1 Introduction

1.1 The Need of Knowledge-Based Systems

Data, information, knowledge, understanding, and wisdom (Ackoff 1989) drive the society, economy, and science. Data refers to a collection of symbols without any meaning beyond its existence. Information refers to a set of data which have been given a meaning by formulating relations between the data elements in a given context. Knowledge constitutes a collection of information with the intention of a certain kind of use. Understanding (or reasoning) refers to an analytic and cognitive process, which takes some knowledge as its input to infer new knowledge as its output by some kind of interpolation. In contrast to understanding, wisdom is an extrapolative and non-deterministic process to provide (i) understanding where there was no understanding before and (ii) a kind of knowledge which cannot be inferred solely by analytical means from available knowledge. Wisdom not only calls upon all previous levels of consciousness, but also extends to human programming such as moral and ethical codes. Wisdom is a human feature and therefore very different from data, information, and knowledge, which can be stored, processed, and even extended by computers by algorithmic reasoning.

Knowledge comes in two forms, either tacit or explicit (Nonaka and Takeuchi 1995). Tacit knowledge is implicit – it is difficult to grasp for the individual holding it therefore hard to communicate. To be of value to other individuals, knowledge has to be made explicit by some kind of articulation, codification, and storage by means of some media to facilitate communication to others. Technical reports, patents, journal articles, monographs, textbooks, or encyclopedias are classical media where the information and knowledge contained is codified by means of natural language. Despite its explicit representation, knowledge is often hard to access and difficult to process, because natural language representations often lack precision and coherence resulting in texts of ambiguous meaning.

This deficiency is not necessarily a consequence of the imperfect presentation skills of the author, but is rather due to the lack of a common vocabulary and common understanding, which is a prerequisite for a shared memory and shared meaning across different domains of discourse (Konda et al. 1992). Typically, this unavoidable shortcoming of natural language knowledge representations is remedied by the intellectual skills, i.e., the wisdom of the reader, who is often well-trained in the domain providing the context of the text.

The codification of tacit knowledge has to go beyond the use of natural language to facilitate sharing, use, and reuse of information and knowledge. The scientific and engineering disciplines have come up with very specific ways to address this representation problem well before computers have been introduced and used for

W. Marquardt et al., *OntoCAPE*, RWTHedition, DOI 10.1007/978-3-642-04655-1_1,

information and knowledge management. For reasons of conciseness, we will focus here and in the remainder of this book only on the engineering perspective.

Engineering knowledge – that is general domain knowledge of the different engineering disciplines as well as specific knowledge evolving, for example, during an engineering design project – is not only represented and stored as text documents in natural language. Rather, more structured and formalized means of representation are being used, such as sets of linked text documents with a prescribed structure, structured worksheets, forms and tables, mathematical models, graphical sketches, technical drawings, and the like. These auxiliary, and at least to some extent formalized, representational schemes are supposed to enhance the expressiveness of natural language, thereby facilitating access and use of information and knowledge. However, the exchange of information and knowledge – be it within a project team or between different organizations – is still hampered because there is no general agreement on the precise syntax and semantics of these representational formalisms.

The problem has not only a logical, but also a technological dimension: That is, the syntactic and semantic heterogeneity is aggravated by the diversity of electronic means and formats for storage, communication, and processing of information and knowledge. For example, during the individual stages of a process and plant design project, information is created and manipulated by diverse software tools and stored in heterogeneous proprietary formats, such as electronic documents, data bases, Computer-Aided Design (CAD) and Computer-Aided Engineering (CAE) systems, simulation files, or asset management tools. The lack of integration between these software tools and their associated data stores unavoidably creates a significant overhead for the project engineers, since much time has to be spent on re-entering of data, interpreting and understanding the data, manually reconciling overlapping data sets, and searching for data. NIST, the National Institute of Standards and Technology in the U.S., has recently analyzed the efficiency losses resulting from inadequate interoperability among computer-aided design, engineering, and software systems (Gallaher et al. 2004). According to this study, insufficient interoperability causes costs of 15.8 billion dollars in the US capital facilities industries, compared to a hypothetical scenario where the exchange of data and the access to information are not restricted by technical or organizational boundaries.

In order to improve this situation, we need to introduce new methods and tools that enable computer-based information and knowledge management in interdisciplinary and cross-institutional engineering projects – methods and tools, which help to cope with the diverse and therefore heterogeneous application software infrastructure. As a *prerequisite* for the solution of these extremely demanding interoperability problems, one has to define a common vocabulary in order to establish a shared understanding of concepts and terms. Such a shared understanding

constitutes the basis of the *shared memory of an organization*[2], regardless of whether the organization is a project team, an enterprise, or a collection of cooperating institutions. Regardless of its technical realization, a shared memory can cover data, information, and – at least to some extent – knowledge implemented by algorithmic reasoning, but not wisdom in the sense of the definitions introduced above. Also, the management of data, information, and knowledge relies on a carefully designed and implemented shared memory. Its realization constitutes a true scientific and technological challenge, given the complexity of the engineering domains and industrial design projects.

There is a much simpler, but still similar problem, which has been successfully addressed in the past decades – namely, the design and implementation of software systems for the mathematical modeling and simulation of technical systems. The approach to this particular problem might serve as a useful role model for the development of future knowledge-based systems supporting engineering work processes. We will briefly draw on this analogy next:

Modeling and simulation systems[3] have originated from an academic environment. A broad acceptance and routine use in the chemical and process industries can be traced back to the 1980s, when commercial systems became widely available. The success of these systems relied (and still relies) on three distinct modules: editors with tailored representational schemes for the formulation of mathematical models; libraries providing reusable mathematical models for standard devices; and, finally, numerical solution techniques for the simulation-based evaluation of the physicochemical knowledge encoded in the models.

We conjecture that for the successful implementation of knowledge-based tools in engineering projects, three comparable modules are required: Knowledge editors providing a semantically rich formalism for knowledge representation; knowledge libraries with self-contained and easily accessible chunks of reusable knowledge; and, finally, efficient reasoning capabilities to interpret the knowledge encoded in the system and to derive new knowledge from it. In fact, mathematical modeling and simulation are themselves a kind of knowledge-based application, working with chunks of reusable knowledge formalized in a special way, namely by means of mathematical equations. The seamless integration of mathematical modeling in the traditional sense with information modeling and knowledge representation to support model-based engineering work processes is widely considered to be an emerging trend (Marquardt et al. 2000; Subrahmanian and Rachuri 2008; Venkatasubramanian 2009).

---

[2] Such interoperability problems in collaborative and distributed chemical engineering are covered in depth in the recent monograph edited by Nagl and Marquardt (2008). Their solution relies on the concept of a shared memory of an organization, but has to reach far beyond.

[3] There is a vast literature on this subject, often with emphasis on a certain engineering domain. We exemplarily cite the book of Braunschweig and Gani (2002) for a chemical engineering perspective.

As a result of the research and development related to the Semantic Web – the next-generation internet – semantic technologies are ready for application in other scientific disciplines. They are based on *ontologies* and include exactly the three enabling building blocks, namely knowledge editors, knowledge representation (or modeling) languages, and powerful reasoning algorithms.

1.2 The Role of Ontologies

In the context of this book, ontologies are primarily seen as a means to efficiently build the knowledge-based software necessary to effectively support engineering work processes. Such software – also referred to as 'intelligent systems', 'artificial intelligence', 'AI systems', or 'expert systems' – comprises two basic software components: the *knowledge base*, which contains generic domain knowledge as well as concrete facts about the case under consideration, and the *inference engine* (also known as *reasoner*), which processes the knowledge and facts stored in the knowledge base and autonomously inferences a solution for the case at hand.

Traditionally, intelligent systems were built from scratch. For large systems, however, this proceeding turned out to be too costly and time consuming. Particularly, the construction of the knowledge bases proved to be the main cost-driver that hindered the further development of intelligent systems in the late 1980s. Neches et al. (1991) diagnosed: "knowledge base construction remains one of the major costs in building an AI system […] As a result, most systems remain small to medium in size. […] The cost […] will become prohibitive as we attempt to build larger and larger systems."

To overcome this economic barrier, Neches et al. (1991) proposed a new approach for the building of intelligent systems: "Building knowledge-based systems today usually entails constructing new knowledge bases from scratch. It could be instead done by assembling reusable components. System developers would then only need to worry about creating the specialized knowledge […] new to the specific task of the system […] In this way, declarative knowledge […] and reasoning services would all be shared among systems."

Besides the obvious *economic benefits* that can be achieved by reusing existing knowledge components, the strategy has other considerable advantages:

- First to mention is the *reduced error rate* of the software: The robustness of a software system increases to the extent to which well-tested parts can be reused (Neches et al. 1991). Plus, due to the continuous revision of the knowledge components, the number of remaining errors will decrease with each reuse cycle.
- A further advantage results from a mandatory change of system architecture required by the new approach: Traditionally, the knowledge representation was heavily intertwined with the reasoning services and the program code in order to optimize the performance of the overall system. As a re-

sult, the knowledge was only accessible to developers with programming experience; domain experts (i.e., the actual knowledge holders) had to get acquainted with the program code first before being able to enter knowledge into the system or to maintain and customize the knowledge base to their particular needs. In practice, this often proved too great an obstacle for the users to overcome. The new approach, by contrast, enforces a strict separation of knowledge base, inference engine, and application-specific program logic. This novel software architecture enables a domain expert to focus on the representation of the knowledge and shields him or her from the implementation details. Therefore, the domain expert is likely to create a knowledge base of *improved quality*. At the same time, the *maintainability* of the entire system is *enhanced*, since reasoner, program code, and knowledge base can be maintained independently by software engineers, application programmers, and domain experts, respectively.

Within the suggested approach, *ontologies* have the function of providing a consensual knowledge representation, which can be reused and shared across software systems and by different groups of users. *Domain ontologies*, in particular, aim at capturing the knowledge of an entire application domain, such as physics, chemistry, or engineering. Note that, in order to be widely applicable, the knowledge represented in an ontology must be generic; that is, the ontology is expected to provide "a conceptual foundation for a range of anticipated tasks", but not to "include vocabulary sufficient to express all the knowledge relevant to those tasks" (Gruber 1995). Thus, to convert an ontology into a knowledge base for a particular application, the knowledge must be specialized and customized.

1.3 The Reusability-Usability Trade-off Problem

Principally, any ontology has to meet two major goals: to be *usable* and to be *reusable*.

- According to the IEEE Standard Glossary of Software Engineering Terminology, *reusability* is defined as "the degree to which a software module or other work product can be used in more than one computing program or software system" (IEEE 1990). *Ontology reusability*, in particular, can be defined as "the adaptation capability of an ontology to arbitrary application contexts" (Pâslaru-Bontaş 2007), including those contexts "that were not envisioned at the time of the creation of the ontology" (Russ et al. 1999). Note that it is neither feasible nor desirable to design an ontology that is equally appropriate for all application contexts (Borst 1997); rather, the goal of reusability is to come up with an ontology that can be adapted to a preferably large number of applications.

- *Usability*, on the other hand, denotes the degree to which the software component is useful for a specific task or application. The term also has the connotation of "ease of use", pertaining to the effort required by a user to utilize a given (software) system. By definition, an ontology is never ready for use, but must always be adapted and refined to a knowledge base for the envisioned application. Therefore, the goal of *ontology usability* can be phrased as minimizing "the effort required to customize the ontology so that it can be used by humans or machines in a given application context" (Pâslaru-Bontaş 2007).

A subtle but important difference between ontology usability and reusability is pointed out by Jarrar and Meersmann (2002):

> "Increasing the reusability of knowledge implies the maximization of using this knowledge among several kinds of (autonomously specified) tasks, while increasing ontology usability could mean just maximizing the number of different applications using an ontology for the same kind of task".

Consequently, it is difficult to simultaneously achieve high degrees of usability and reusability: Specializing in one kind of task makes the ontology more useable for this particular task, but it also decreases the likelihood of its reusability; a highly abstract ontology, on the other hand, may be applicable to a variety of different tasks, but it is unlikely to prove very useful for any of these without extensive modification and detailing. This challenge is known as the *reusability-usability trade-off problem* (Klinker et al. 1991) in the literature.
This trade-off problem has to be one of the drivers for research on ontologies, not only for academic, but also for very practical reasons.

- This problem is academically challenging and rewarding. It constitutes an exciting research problem at the interface between computer science and its applications in science and engineering. Its solution is of great significance to both the theory and practice of ontological engineering.
- The development and maintenance of any major IT system requires a significant effort. The software industry has established development processes based on proven technologies to reduce cost to the extent possible. The introduction of a new paradigm and associated technologies not only requires some reference systems, which demonstrate the improved capabilities from a technological or even from an end-user's perspective. Rather, the economical advantage over established software technologies has to be clearly demonstrated in order to motivate a software company to take the significant risk of integrating the principles and technologies of ontological engineering into their software engineering processes.

Based on this assessment, an appropriate solution to the reusability-usability trade-off problem should be considered as the major enabler for a future use of ontologies in the software industries. Therefore, this trade-off problem has shaped the

major guiding principle for our research, the results of which are presented in this book.

1.4 Objective and Outline of the Book

This book presents OntoCAPE, a general-purpose ontology for applications in the domain of computer-aided process engineering (CAPE). CAPE is a sub-discipline of chemical engineering focusing on IT methods and tools to support the design, planning, construction, commissioning, and operation of chemical process systems and plants. We will discuss the architecture of OntoCAPE, thereby putting particular emphasis on the design rationale we followed. We will show how OntoCAPE reconciles the trade-off between reusability and usability, and is thus broadly applicable to a variety of chemical and process engineering tasks with only moderate customization effort.

The content of his book is organized as follows.

Chapter 2 reviews the scientific background and establishes the terminology required for discussing ontologies, thus providing the basis for the subsequent chapters. It starts off by contrasting the similar but different perceptions of 'ontology' in the areas of philosophy and computer science. Next, the specification of ontologies through informal and formal languages is discussed; the latter option is further elaborated by describing the modeling capabilities of formal ontology languages. Having established these basic facts, it is argued that an ontology must be both formally and informally specified in order to be of practical use. The modeling language OWL is briefly introduced for the sake of completeness. Moreover, it is clarified what differentiates a "true" ontology (i.e., a reusable knowledge representation, as defined in Sect. 1.2) from so-called pseudo-ontologies and lightweight ontologies. The chapter closes with a classification of ontology types according to their respective functions.

Chapter 3 gives an overview on the scope and content of the OntoCAPE ontology. Initially, a short overview is given on its three structural elements – layers , modules and partial models – by which the ontology is organized. Furthermore, the representation and dissemination of OntoCAPE are presented. Finally, the scope and content of the individual parts that constitute OntoCAPE are briefly summarized.

Chapter 4 introduces the Meta Layer, which is located on top of the OntoCAPE ontology. The Meta Layer explicitly represents the underlying design principles of OntoCAPE and introduces common standards for the design and organization of the ontology. In particular, domain-independent root concepts and a theory of mereotopology are introduced on the Meta Layer.

Chapter 5 introduces a number of key concepts, which establish the principles of general systems theory and systems engineering, according to which the ontology is organized. Important systems-theoretical and physicochemical primitives com-

plement the mereotopological concepts adopted from the Meta Layer. It also establishes the means to model a system from a particular viewpoint. These viewpoints are used to partition the representation of complex systems into manageable parts and to emphasize a certain perspective the system is viewed from. Furthermore, a representation of vectors and higher-order tensors is suggested. We introduce a concept for coordinate system, which serves as a frame of reference for the observation of system properties. Finally, we establish the principles of network theory to lay a solid foundation for a structured representation of any kind of complex system showing a network character.

Chapter 6 holds fundamental notions such as space, time, physical dimensions, SI-units, mathematical relations, etc., which do not directly belong to the CAPE domain but are required for the definition of or as supplements to the domain concepts. Since OntoCAPE is not supposed to conceptualize domains beyond the scope of CAPE, this partial model is only rudimentarily elaborated.

Chapter 7 collects all the concepts which are required to provide an abstract description of materials processed in a chemical plant. This partial model comprises the essential concepts for the description of pure chemical substances and mixtures thereof at the macroscopic and atomic scales. Mechanisms and stoichiometry of chemical reactions are presented next. Finally, the principles and concepts for a rigorous description of the thermodynamic behavior of materials in a certain physical context are described.

Chapter 8 presents all those concepts which are directly related to the processing of materials and to plant operations. This part of the ontology is of particular interest for chemical process design. The concepts are modeled on a conceptual as well as on a more concrete, application-oriented level by adding classes and relations needed for a specific use of the ontology. This includes the extension towards two alternative classification schemata for unit operations as well as exemplary descriptions of typical process units.

Chapter 9 defines the notions required for a representation of mathematical models. It introduces the basics concepts for mathematical modeling, including model variables and equations, as well as concepts for representing the composition of a model from sub-models and their connections. Some specialized types of models are presented, including models for representing the behavior of materials and process units as well as models for the estimation of investment costs.

Chapter 10 presents the major design principles that guided the development of OntoCAPE: These principles, which subsume the plethora of recommendations stated in the literature, are *coherence*, *conciseness*, *intelligibility*, *adaptability*, *minimal ontological commitment*, and *efficiency*. The principles are defined individually, and their general implication on ontology design is critically assessed. Finally, we describe the translation of these principles into concrete design decisions to be taken during the realization of OntoCAPE.

Chapter 11 gives a review of related work. The earlier efforts in information modeling at the authors' institute are summarized first, since the results of this research have laid the foundation for the development of OntoCAPE. Next, related

work of other research groups is reviewed and compared with OntoCAPE. The review is confined to ontologies which are of particular relevance in the context of this work. In particular, only those ontologies are considered, which bear close resemblance to OntoCAPE with respect to both scope and level of complexity, or which had a significant influence on the development of OntoCAPE.

Chapter 12 describes some software applications, which have been realized on the basis of OntoCAPE and thus demonstrate the ontology's potential for use and reuse. At first, two early applications in the area of mathematical modeling and simulation are presented. Next, an ontology-based knowledge management system is described, which has been based on version 2.0 of OntoCAPE. The last example refers to an ongoing project, which realizes information integration and management across the plant lifecycle in an industrial setting. The last part of the chapter gives an assessment of the improvement achieved by transitioning from OntoCAPE version 1.0 to version 2.0 by means a few quantitative measures.

Chapter 13 concludes the book with a brief summary of the major results, with an assessment of the design rationale, and with a review of the continuous improvement process chosen. Future research opportunities are identified, particularly an extension of OntoCAPE in scope to also cover work processes.

1.5 References

Ackoff RL (1989) From data to wisdom. *J. Appl. Syst. Anal.* **16**:3–9.

Borst P, Akkermans JM, Top JL (1997) Engineering ontologies. *Int. J. Hum Comput Stud.* **46**:365–406.

Braunschweig B, Gani R, eds. (2002) *Software Architecture and Tools for Computer Aided Process Engineering*. Elsevier Science B.V.

Gallaher MP, O'Connor AC, Dettbarn Jr JL, Gilday LT (2004) *Cost Analysis of Inadequate Interoperability in the U.S. Capital Facilities Industry*. Technical Report (NIST GCR 04-867), National Institute of Standards and Technology, Gaithersburg, Maryland.

Gruber TR (1995) Toward principles for the design of ontologies used for knowledge sharing. *Int. J. Hum Comput Stud*. **43** (5/6):907–928.

IEEE (1990) *IEEE Standard Glossary of Software Engineering Terminology*. IEEE Standard 610.12-1990, Institute for Electrical and Electronics Engineering, New York.

Jarrar M, Meersmann R (2002) Scalability and knowledge reusability in ontology modeling. In: *Proceedings of the International conference on Infrastructure for e-Business, e-Education, e-Science, and e-Medicine SSGRR2002*.

Klinker G, Bhola C, Dallemagne G, Marques D, McDermott J (1991) Usable and reusable programming constructs. *Knowl. Acquis.* **3** (2):117–135.

Konda S, Monarch I, Sargent P, Subrahmanian E (1992) Shared memory in design: A unifying theme for research and practice. *Res. Eng. Des.* **4**:23–42.

Marquardt W, von Wedel L, Bayer B (2000) Perspectives on lifecycle process modeling. In: Malone MF, Trainham JA, Carnahan B (eds.): *Foundations of Computer–Aided Process Design*. AIChE:192–214.

Nagl M, Marquardt W, eds. (2008) *Collaborative and Distributed Chemical Engineering: From Understanding to Substantial Design Process Support.* Springer, Berlin.

Neches R, Fikes R, Finin T, Gruber T, Patil R, Senator T, Swartout WR (1991) Enabling technology for knowledge sharing. *AI Mag.* **12** (3):36–56.

Nonaka I, Takeuchi H (1995) *The Knowledge Creating Company.* Oxford University Press, New York.

Pâslaru-Bontaş E (2007) *Contextual Approach to Ontology Reuse: Methodology, Methods and Tools for the Semantic Web*. PhD Thesis, FU Berlin.

Russ T, Valente A, MacGregor R, Swartout W (1999) Practical experiences in trading off ontology usability and reusability. In: *Proceedings of the 12th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop*. SRDG Publication.

Subramanian E, Rachuri S (2008) Guest editorial: special issue on "engineering informatics". *Trans. ASME, J. Computing and Information Science in Engineering*. **8**:010301-4.

Venkatasubramanian V (2009) Drowning in data: informatics and modeling challenges in a data-rich and networked world. *AIChE J.* **55**:2–8.