A Framework for Decision Support Systems Adapted to Uncertain Knowledge

Zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften

von der Fakultät für Informatik
der Universität Fridericiana zu Karlsruhe (TH)

genehmigte

Dissertation

von

Yi Yang

aus Shanghai

Tag der mündlicen Prüfung:	18. June 2007
Erster Gutachter:	Prof. Dr. Jacques Calmet
Zweiter Gutachter:	Prof. Dr. Pascale Kuntz-Cosperec

To my darling Yan,
whose love and support made everything possible.

Acknowledgements

First and foremost, I am grateful to my advisor Prof. Dr. Jacques Calmet for the academic guidance and strong support he gave me throughout my stay at the Institute of Algorithms and Cognitive Systems. I particularly appreciate his patience and willingness to help at all time without which this work would not have been completed.

There are a number of other people whose contributions to this thesis should also be acknowledged. I would like to thank Prof. Dr. Pascale Kuntz-Cosperec for accepting to be my second advisor. Thanks my examiners, Prof. Dr. Jürgen Beyerer and Prof. Dr. Ralf H. Reussner. Thanks Henning Burchardt, Thilo Mie, Stefan Röhrich, Carmen Stüber and Helga Scherer, whose assistance in proofreading, printing or binding the thesis was a great help.

I send special thanks to DFG for the financial support during my studies, which helped me both in allowing me to focus on my research, and in travel funds that helped me present my works and cooperate with other researchers.

I am deeply grateful to my parents, Zhiming and Lianzhen, whose love and support has been with me throughout my life. And to my parents in law, Baosen and Lin, for caring for my wife during the pregnancy.

Finally and most importantly, my most sincerest thanks to my wife, Yan, for her love, support and understanding, which makes completing the Ph.D. study possible and worthwhile. Thank you Yan for bearing and caring for our son (Kaiwen) during this major endeavour.

Deutsche Kurzfassung

Folgende Probleme stellen Systeme zur Entscheidungsfindung vor große Herausforderungen: Erstens die Unsicherheit bei der Entscheidungsfindung beim Vorliegen von unscharfem Wissen; zweitens die Komplexität der Anwendungsbereiche, die eine Verwertung von Wissen aus vielen Bereichen erfordert; und drittens die dauernde Änderung der Systemumgebung, die den Systemen eine große Anpassungsfähigkeit abverlangt. Diese Arbeit präsentiert einen ontologiebasierten und serviceorientierten Ansatz für Systeme zur Entscheidungsfindung, welcher verschiedene Methoden integriert und optimiert, um jede der genannten Herausforderungen angehen zu können.

Die meisten bekannten Lösungen sind nur darauf zugeschnitten, einer dieser Herausforderungen gewachsen zu sein. So können beispielsweise entscheidungstheoretische und analytische Ansätze sehr gut mit unsicherem Wissen umgehen, wohingegen Expertensysteme und wissensbasierte Systeme besser mit der Komplexität gebietsübergreifender Anwendungen zurecht kommen. Aber kaum eines von ihnen kann ein praktisches Rahmenwerk zu den Entscheidungsfindungssystemen anbieten, das in der Lage ist, alle drei Probleme gleich gut zu lösen.

Im Mittelpunkt der Arbeit steht die Entwicklung eines ontologiebasierten unscharfen Modells — *OntoBayes*. Es ermöglicht eine Integration von Ontologien, Bayesschen Netzwerken und Influenz-Diagrammen in der Web-Ontologien-Sprache OWL, um die Vorteile von allen Methodiken zu kombinieren, da sie sich gegenseitig gut ergänzen können. OntoBayes setzt sich aus zwei Teilen zusammen: der Wissensbasis und den Entscheidungsmodellen.

Ontologien verfügen über eine hervorragende Modellierungsfähigkeit für komplexe Einsatzgebiete, sie können jedoch unscharfes Wissen nicht repräsentieren. Im Gegensatz dazu haben Bayessche Netze eine ausgezeichnete Fähigkeit zur Darstellung von unscharfem Wissen, sind aber wiederum stark beschränkt bei der Repräsentation von Wissen aus komplexen Anwendungsgebieten.

Diese Arbeit erweitert OWL durch Hinzufügen neuer Annotationen zur Darstellung von Wahrscheinlichkeiten und Abhägigkeitsrelationen, um Bayessche Netze in Form der Ontologien repräsentieren zu können. Diese integrierte Wissensbasis bildet den "'Wissens-Anteil"' der Entscheidungsunterstützungssysteme.

Um jedoch die Entscheidungsfindung zu verwirklichen, sind zusätzliche Erweiterungen der Entscheidungsmodelle unbedingt erforderlich. Bei solchen Erweiterungen spielen Ontologien auch eine sehr wichtige Rolle. Basierend auf Ontologien, insbesondere auf Bayesschen Netzen, lassen sich unterschiedliche Entscheidungsmodelle mit individuellen Anwendungsontologien spezifizieren. Als Beispiele solcher Anwendungsontologien seien hier Influenz-Diagramme, Entscheidungsnetzwerke sowie Markov-Entscheidungsprozesse genannt.

Jede Anwendungsontologie verfügt über eine Menge von Annotationen zur Definition von aufgabenspezifischen Konzepten. Hieraus können spezialisierte Entscheidungsmodelle gebildet werden. Diese werden von Systemen verwendet, um bestimmte Aufgaben mit unscharfem Wissen zu bewältigen. Diese Erweiterungen von OntoBayes bilden dann den wesentlichen Bestandteil der Entscheidungsmodelle der Systeme. Kombiniert mit dem OntoBayes-Modell sind Systeme zur Entscheidungsfindung nun in der Lage, mit Unsicherheit und Komplexität umzugehen. In dieser Arbeit werden Influenz-Diagramme verwendet, um Entscheidungsmodelle bilden zu können. Für OntoBayes wurde ein Protégé Plugin *OWLOntoBayes* implementiert, mit dem die Benutzer die Wissensbasis und die Entscheidungsmodelle leichter verarbeiten können.

Die Eigenschaft der Anpassungsfähigkeit kann die Leistung der Systeme in erheblichem Maße verbessern. Diese Arbeit optimiert die Methodik der *Virtuellen Wissensgemeinschaft* (engl. virtual knowledge community, VKC) und setzt den Ansatz der *Service-Orientierte Architektur* (SOA) ein, um diese Eigenschaft in die Systeme einzubringen.

VKCs bieten eine Plattform zum Wissensaustausch an und ermöglichen es, das Wissen der sich ständig ändernden Systeme rechtzeitig zu aktualisieren. Die Aktualisierung von Wissen führt zur Anpassung der Entscheidungsmodelle und Entscheidungsprozesse. Die existierende Methodik ist jedoch eingeschränkt durch ihre Unfähigkeit, unscharfes Wissen zu behandeln, insbesondere beim Austauschen von bayesschen Informationen. Dies wird dadurch optimiert und erweitert, dass sich die bekannten Konzepte von VKCs für struktuelle und numerische bayessche Informationen jeweils mit zusätzlichen Operationen ausbauen lassen. Auch die Informationen von Influenz-Diagrammen können mit Hilfe der VKCs ausgetauscht werden. Damit VKCs wirklich mit OntoBayes zusammenarbeiten können, müssen VKCs in OWL angepasst werden, da OWL die formalle Sprache zur Wissensrepräsentation von OntoBayes ist.

Der Ansatz von SOA ermöglicht mehr Flexibilität der Systeme. Jeder Prozess besteht aus vielen kleinen Web-Diensten, die von unterschiedlichen Systemkomponenten geliefert werden. Mit der Änderung der Systemumgebung werden neue Prozesse oder Arbeitsabläufe nach Bedarf durch Selektion und Zusammensetzen von neuen Web-Diensten rekonstruiert, und lassen sich daher zur Entscheidungsfindung an die gegebene Änderung anpassen. Somit können Systeme zur

Entscheidungsunterstützung nicht nur die drei oben genannten Herausforderungen angehen, sondern auch mit lose gekoppelten, verteilten Komponenten arbeiten und plattformunabhängig bleiben.

Basiert auf OntoBayes, VKCs, Multiagent-Systemen und Web Services wurde ein Prototyp eines Systems zur Entscheidungsunterstützungen implementiert. Potenzielle Anwendungsgebiete dafür sind beispielsweise in Banken, in der Medizin oder im Katastrophenmanagement und der Katastrophenversicherung zu finden. Um die Durchfürbarkeit des Systems testen zu können, sind einige Szenarios für Katastrophenversicherung entworfen und evaluiert worden. Eine Graphische Benutzerschnittstelle wurde implementiert, um das Bedienen und das Testen des Systems zu vereinfachen.

Contents

Chapter 1

Introduction

1.1 Motivation

During the 1950s and 1960s the concept of decision support was investigated from two aspects: the theory of organizational decision making and the techniques of interactive computer systems. After that it became an area of research that focuses on computerized system supporting decision making activities [Pow03]. A DSS (*Decision Support System*) can be defined as "a computer program that provides information in a given domain of application by means of analytical decision models ..." [KM95].

The main challenges for DSSs are uncertainty, adaptivity, knowledge management, collaboration, intelligence and explanatory power etc. [DL05, Hol01, Gro96, Nak06]. There are too many kinds of DSSs nowadays and it is almost impossible to give a succinct survey about them. In general they are designed to address only one or two of the above challenges. Most of them are designed and developed based on precision and certainty. They are infeasible to work under uncertainty. They provide unreliable solutions based on the assumption of closed environments. For most real applications, uncertainty is an inevitable feature and can not be ignored. Agents do not act within a static and close environment but within a dynamic and open one. The available information is mostly incomplete and often imprecise because agents almost never have access to the whole truth implied by their environment. Agents must therefore act under uncertainty, and must be able to make optimal decisions with limited computational resources.

The main *goal* of this dissertation is to design a theoretical framework for DSSs to address all of these challenges, particularly the challenge to decision making under uncertainty.

1.1.1 A Motivating Example

This work is one of the interdisciplinary projects of the postgraduate college "natural disaster", which is financed by DFG, the German Research Foundation. Considering this background a simple decision problem from the application domain of catastrophe insurance will be briefly introduced below as a motivating example.

Example 1.1 *Mr. Bob has bought a house in Shanghai and this house is not far away from the HuangPu River. Now he is trying to decide whether he needs a flood insurance for this house. If he needs it, then he must know the potential flooding insurance products that he can choose from. And finally he must decide which one among them he should buy.*

Probably Bob can make use of some different existing DSSs to get helpful information and to make proper decisions in the face of uncertainty. For example, the GIS (*Geoinformationssystem*) can provide him information about the flooding risk of his house by using risk card or map; the portal of insurance companies or agencies can provide him information about available products of flooding insurance; the homepage of an insurance expert can help him to evaluate these products etc.. But there are still many problems can not be solved by these systems.

The first problem is the limited capability of human judgments of uncertainty (or probability). In this example Bob tries to make rational decisions under uncertainty — the flooding risk. For example he is informed by GIS that the probability of a 100-year flood in this area is 20%, and a 100-year flood was just happened one year ago. What does the figure 20% mean for Bob? Can Bob interpret this information correctly? Unfortunately, according to the study of human judgments of probability, judged probabilities do not conform to the equations of probability theory [BAH90]. Probably Bob will think that the occurrence probability is too low or "lightnings won't strike the same place twice". Such biases could lead to wrong decisions. To reduce the biases of human judgment by decision making under uncertainty, DSSs must be adapted to uncertain knowledge.

The second problem is the limited capability of knowledge management of human, particularly for domain specific knowledge. In this example Bob can probably only take the flooding risk and the premium of an insurance product into account, in spite of the fact that there are many other factors having impact on the decision of buying it. For example the risk coverage of an insurance, the financial status of the buyer and so on. How to structure all of these factors and how to make an optimal decision based on them? That is obviously too difficult for Bob, except when he is an insurance expert. From this perspective DSSs must be knowledge-based and integrated with decision theories.

The third problem is the limited capability of human to react to open and dynamical environments. When Bob makes a decision to consume a service, prob-

ably some new and better services are issued. How can Bob assert that he is kept up to date? For such situations DSSs must be adaptive to the perpetual changes of the environment.

There are other unsolved problems besides these discussed above, e.g. the system support of collaboration between Bob and domain experts, the explanatory power of DSSs for their recommendations and so on. Nowadays no existing DSSs can help Bob solve these problems at the same time. Therefore a framework for DSSs will be investigated and proposed in this work both from practical and theoretical perspectives, in order to solve the problems. In the next section the underlying methodologies of such a framework will be surveyed.

1.2 The Underlying Methodologies

The proposed framework in this work touches the following research fields: ontologies, Bayesian networks, influence diagrams, virtual knowledge communities, multiagent systems and web services.

From the local perspective of a DSS, each of these methodologies has its special features and can address the challenges mentioned in the last section, respectively. But from the global perspective each of them has its own limitations and incompatibility. They can not be automatically integrated into the system without any optimization or improvement. Therefore an integrated solution for DSSs is desired to meet the goals of the work.

The key reason for using ontologies in DSSs is that they enable the representation of background knowledge about a domain in a machine understandable form. Ontologies can formally and explicitly specify a shared conceptualization [Gru93]. They can excellently represent the organizational structure of large complex domains, but their application is bounded because of their inability to deal with uncertainty [KP98].

In order to overcome this limitation, Bayesian Networks will be introduced into the framework. Bayesian networks are widely used graphical model for probabilistic knowledge representation under uncertainty [Jen96]. Two kinds of information can be represented in a BN (*Bayesian Networks*): structure and numerical information. The former is in principle a DAG (*Directed Acyclic Graph*) where nodes are random variables and arcs between nodes imply the dependency (or conditional) relation between the variables. The latter is the Bayesian probabilistic information of random variables.

In comparison with ontologies, BNs have their excellent ability to represent uncertain knowledge in a sound mathematical way. But they are very limited because of their inability to represent complex structured domains. Obviously they can complement themselves via a sound combination aiming at taking advantages

of both. Inspired by this approach, an ontology-driven uncertainty model — OntoBayes — will be proposed in this thesis.

OntoBayes has two parts: a knowledge part and a decision model part. The former is an integration of the certain and uncertain knowledge based on ontologies and BNs respectively, while the latter can describe different decision models based on influence diagrams. An ID (*Influence Diagram*) combines a BN with additional node types for decision choices (or actions) and utilities. Utilities are used to quantify the preferences of different choices. A rational agent can make a best decision among all choices according to the principle of MEU (*Maximum Expected Utility*). MEU indicates that a rational decision under the explicit uncertainty of the situation is the decision with the greatest expected utility, i.e. the maximization of the expected benefits of the possible outcomes.

OntoBayes makes use of OWL (*Web Ontology Language*)[1] as the underlying ontology modeling language and extends it with additional annotations according to the semantics of BNs and IDs. Nowadays OWL is the broadly accepted ontology language of the semantic web. OWL builds upon RDF (*Resource Description Framework*) with more descriptive power. It aims to reach maximal compatibility with XML, RDF and existing ontology languages and logic frameworks. A Protégé[2] plugin *OWLOntoBayes* will be implemented for the OntoBayes model. It allows users to edit and codify BNs and IDs into OWL files with a graphical user interface.

Virtual knowledge communities will be applied to enable the collaboration between agents in DSSs. The concept of a VKC (*Virtual knowledge community*) was introduced as a means for agents to share knowledge about a topic [MC04]. It aims to increase the efficiency with which information is made available throughout the society of agents. VKCs can provide a virtual place for corporate knowledge retrieval, sharing decision models and building virtual teams.

Corporate knowledge was defined as the overall knowledge detained by agents within a system and their ability to cooperate with each other in order to meet their goal [MHC04]. Decision making based on corporate knowledge has become crucial for a society made of distributed agents each possessing its own knowledge, particularly when the knowledge is uncertain. During a process of decision making agents can profit from the corporate knowledge of their society better than only from their sole knowledge. Agents are required to know not only what it knows but also what they know, and are expected to make maximum use of the knowledge.

In this thesis we will extend VKCs with semantic features of BNs and IDs, in order to make them compatible with the OntoBayes model. Due to fact that the

[1] http://www.w3.org/TR/owl-guide

[2] Protégé is an ontology editor. http://protege.stanford.edu/

previous implementation of VKCs is based on RDF, we need to adapt VKCs to suit the needs of OntoBayes which is totally built on OWL.

The Multiagent paradigm and web services are the selected overall technical foundations of the framework. The DSS is implemented based on intelligent agents which have the characteristics of reactivity, proactivity and sociability. These characteristics make them suitable as software entities for the delegation of diverse decision making tasks and for collaborative work with each other. We will design and implement the system in which all components are built as web services so that users or system agents can integrate them and perform agent tasks helping users according to various knowledge sources.

The world of web services was characterized as loosely coupled distributed systems based on service oriented computing. The use of web services could be considered as actions that the agent may take to meet its goals. In general, decision making is not a simple event but a process leading to the selection of a course of action among several alternatives. Agents need to select and to compose different actions in order to make a decision. From the point of view of service, the process of decision making consists of different services provided by the agent itself or by external agents. This viewpoint supports us to implement the DSS based on web services. A service oriented architecture will be designed to facilitate the implementation of the DSS. Such an architecture makes the DSS more adaptive and flexible.

Throughout the thesis, we will make use of some simple made-up examples in the application domain of catastrophe insurance, to demonstrate the feasibility of the framework and to test the implementation of the DSS. In fact, even if the application study is specific, the framework presented in this thesis could serve as a basis to build different DSSs working under uncertainty with a probabilistic approach.

1.3 Structure of the Thesis

This dissertation is structured as follows. Chapter 2 introduces general background of the theoretical foundations upon which the framework is based: ontologies, which are used as an underlying knowledge representation paradigm for DSSs; BNs, which are used to model uncertain knowledge with conditional dependency and probabilities; IDs, which are used to represent general class of decision problems and to evaluate optimal policies for these problems. Additionally two important uncertainty approaches, the probabilistic approach and the fuzzy approach, will be discussed.

Chapter 3 introduces general background of DSSs and presents some important challenges first. Then an advanced and abstract framework for DSSs is pro-

posed to address these challenges. It is characterized by a pillar-based view, a layered view, a decision theoretical view and a process view, respectively. Finally, main features of the framework will be summarized.

Chapter 4 introduces the motivation as well as the design of the OntoBayes model. Afterwards integrating BNs and IDs into OWL will be investigated, respectively. The integration of BNs and OWL solves the problem of incorporating uncertainty into ontologies. It is one of the main purposes of this doctoral work. Eventually a survey of related works for incorporating uncertainty and ontology will be discussed.

Chapter 5 is devoted to investigate how to address the emerging paradigm of knowledge dissemination and collaboration in Decision Support Systems through VKCs. At first, the motivation and definition of VKCs will be presented. Then an overview of all basic concepts using to model VKCs will be introduced according to [Ham04]. Afterward it will be investigated how to make use of VKCs to facilitate knowledge dissemination, particularly with emphasis on knowledge sharing and decision models sharing in the OntoBayes model. At last it will be demonstrated how to utilize VKCs for supporting decision making in terms of collaboration and adaptivity.

Chapter 6 will first describe a service oriented architecture according to the proposed framework, in order to guide the implementation of a DSS. A simple overview of the implementation will be provided next. After that the DSS will be tested with a designed use case in decision support for purchasing catastrophe insurance products.

Chapter 7 is the last chapter of this thesis. It is devoted to summarize contributions and to outline the future works and open research lines.

The appendix provides detailed descriptions for the OWL extensions in the OntoBayes model.

Chapter 2

Preliminaries and Definitions

This chapter is devoted to introduce three important theoretical foundations used throughout this thesis, namely ontologies, Bayesian networks (BNs) and Influence Diagrams (IDs). Among them, ontologies are most important, because the work of Chapter 4 focuses on building an ontology-driven model by integrating BNs and IDs into ontologies. Besides these three methodologies, we will introduce the probabilistic and fuzzy approaches for dealing with uncertainty, with emphasis on the probabilistic one.

2.1 Ontologies

In this section we will introduce the methodology of ontologies, from its historical derivation to its new position in the domain of computer science. Formal definitions, features and categorizations of ontologies will be given first. Afterward the most important formal languages for representing ontologies in the last decade will be surveyed in Section 2.1.3 and 2.1.4. The focus of the survey is OWL, which is the selected underlying formal ontology language for this work.

2.1.1 History, Motivation, Definition and Features

The very term *Ontology*[1] derives from philosophy originally and is the study of being in the world. The word itself is composed of two Greek words: *ontos* and *logos*. The former stands for "being" and the latter for "treatise" [GPFLC04]. Ontology as a philosophical discipline is originally used to distinguish between essence and existence. It is often used as a synonym of "metaphysics" which

[1] In order to avoid confusing the term 'ontology' in the field of philosophy and information science, it was proposed in [GG95] to use the words 'Ontology' (with capital 'o') and 'ontology' (with lowercase 'o') to distinguish them respectively.

refers to the work of Aristotle for explaining the nature of the world in the field of philosophy [Smi03]. Different philosophers have different interpretations about Ontology, but all of them agreed with the point that the essentials of Ontology must be the capture of the essence of entities and their classification in reality. The only debate is with which division to classify them[2].

Since the early nineties the term Ontology has gained new currency in the field of information science. On the basis of the commonness between philosophy and AI (*Artificial Intelligence*) [McC95, Smi03], the AI researchers adopted the term ontology for knowledge representation by drawing the results of the works carried out over the last 2000 years. The main motivation to use ontologies in AI is to facilitate knowledge sharing and reuse in a computational way [Fen01]. There are several pioneer projects devoted to develop a library of reusable ontologies, for instance the project Knowledge Sharing Effort (KSE) [NFF+91] and Cyc [GL90].

Definitions of Ontologies

Many different definitions of ontologies were given in the nineties. Neches et al. defined ontologies in [NFF+91] as follows: "An ontology defines the basic terms and relations comprising the vocabulary of a topic area ...". A few years later, a more simple definition was introduced by Gruber in [Gru93]:

Definition 2.1 *An ontology is an explicit specification of a conceptualization.*

The term "conceptualization" was introduced by Genesereth and Nilsson in the field of AI in [GN87]. They pointed out that a conceptualization is the most important step for formal knowledge representation. It includes the individuals, concepts, and other entities presumed to exist in the world and the interrelationships between them. Gruber claimed in [Gru95]: A "conceptualization" is an abstract and simplified view of the world to be presented by wish. But it still became the center point criticized because of the ambiguousness of this word by using it [Smi04] and the incompleteness of foundational relations[3] [BDS04]. Studer et al. extended Definition 2.1 in [SBF98] as follows:

Definition 2.2 *An ontology is a formal, explicit specification of a shared conceptualization.*

This definition is widely spread and accepted in the field of information science, even though there are still debates about conceptualization. It will be the underlying definition of ontologies adopted in this thesis. In comparison with Definition

[2]There existed many divisions, for example substantialists and fluxists, adequists and reductionists. More detailed descriptions can be found in [Smi03, GPFLC04].

[3]Foundational Relations such as part-of, member-of, partition-of etc. are required for correlating different biomedical ontologies.

2.1 there are two words more in Definition 2.2: ‘formal’ and ‘shared’. According to [Fen01, SBF98], the former points out that ontologies must be presented in formal language, not in natural language. The latter emphasizes that the captured knowledge within ontologies should be commonsense, in order to facilitate or insure the knowledge sharing between different communication parties.

Features of Ontologies

From the definitions as mentioned above some inherent and elementary features of ontologies can be summarized as follows:

- **Individuals** play a central role in an ontology. An individual exists as a single, separate thing or being. The set of individuals is countable. Normally each individual has a clear identity to make them distinguishable from others, even though they have common properties.
- **Concepts** are abstract groups or collections of individuals which have the same properties. The concept space is normally continuous and infinite. The instantiation of a concept results in its individuals.
- **Properties** are alway associated with concepts or individuals.
- **Relationships** exist between concepts or individuals and can be specified based on the assignment of properties in ontologies.
- **Constraints** are used to specify concepts or individuals more precisely and explicitly. They can also be used to describe exceptions in the ontological world.

Every ontology language must support the definition and description of these key features (or elements) to facilitate the ontological conceptualization.

2.1.2 Ontology Categories

There are many works devoted to categorize ontologies depending on the levels of generality or other viewpoints, such as the categorizations presented in [MVI95, vHSW97, Gua98, LM01, SPO04]. Among them the one based on Gurino’s work [Gua98] was the most accepted and adopted by other researchers because of the introduction of the terms “top-level ontology” and “domain ontologies”. According to his work there are four kinds of ontologies. From general to specific they are top-level ontologies, domain ontologies, task ontologies and application ontologies. This classification is depicted in Figure 2.1 which is borrowed from [Gua98]. The arrows in the figure represent the specialization relationship.

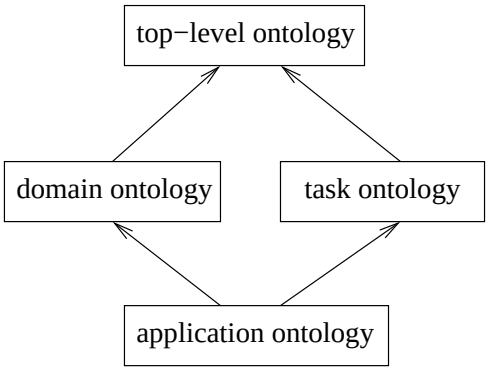

Figure 2.1: Types of ontologies according to their level of generality.

- **Top-level ontologies** classify high-level and domain-independent ontologies which provide very general concepts and link all other (domain specific) ontologies. They are often characterized as representing common sense concepts. The main problem in the knowledge world is the semantic heterogeneity [Col02] because similar terms may differ significantly in meaning in different ontology-driven systems. This problem makes the integration of different information systems difficult. To deal with this problem, the IEEE Standard Upper Ontology[4] Working Group is trying to build top-level ontologies which can provide generic concepts such as time, space and so on.

- **Domain ontologies** provide vocabularies about concepts and their relationships within a specific domain. They specialize or extend concepts defined in a top-level ontology to build reusable knowledge entities that can be shared between different domains [AS03].

- **Task ontologies** provide concepts and relationships associated with generic tasks or activities.

- **Application ontologies** provide application-dependent concepts and provide the vocabulary for given applications related to a certain task or activity.

This work makes use of the first two kinds of ontologies to construct an ontology-driven decision support system (see Chapter 6). The term “top-level ontologies” will be simplified as “upper ontologies” in this thesis because it is often mentioned as a synonym for “upper-level ontologies”. In [DOS03] an upper ontology

[4]http://suo.ieee.org

is defined as “a set of integrated ontologies that characterizes a set of basic commonsense knowledge notions” .

Lightweight and Heavyweight Ontologies

Nowadays ontologies play a more and more important role in many natural scientific or economic domains, especially in new emerging domains like the Semantic Web [BLHL01]. But most applications in this field make use of ontologies as a synonym for “taxonomies”, for instance the Yahoo! Directory, UNSPSC[5]. Such taxonomies can also provide a hierarchical classification of concepts from general to special, but only with `SubClassOf` and `SuperClassOf` relations (or the *is-a* relation). In order to distinguish ontologies from taxonomies, the ontology community proposed *lightweight* and *heavyweight* ontologies. Lightweight ontologies are closed to taxonomies and consist of concepts, concept taxonomies, relationships (not only restricted to the *is-a* relation) between concepts and properties defined on concepts, whereas heavyweight ontologies can provide more semantic modeling within ontologies by means of adding axioms and constraints to lightweight ontologies [GPFLC04].

2.1.3 Semantic Networks and RDF(S)

The paradigm of semantic networks is one of the most important knowledge representation methodologies inspired by the semantic association model of human memory in the field of cognitive psychology. This model pointed out the existence of associative relations which link the semantically connected concepts in the human memory. A *semantic network* is a directed graph composed of named nodes and labeled edges, where each node represents a concept with a name and a unidirectional edge represents the associative relations between nodes [Rei91].

RDF is an XML (*Extensible Markup Language*) based language of the W3C (*the World Wide Web Consortium*) recommendation to describe resources about anything, but its main utility is for metadata descriptions in the Semantic Web [McB04]. It is designed to describe resources in a minimally constraining, flexible way [KC04].

RDF Graph

A RDF *graph* is composed of *triples*, each consisting of a subject, a predicate and an object as shown in Figure 2.2 (from [KC04]), where

[5]UNSPSC is an internationally established classification system of the merchandise-economy. http://www.unspsc.org/

Figure 2.2: The graphical representation for a generic RDF triple.

- a **subject** is a resource described with or without[6] an URI (*Uniform Resource Identifier*),
- a **predicate** defines attributes or relations used to describe a resource (or resources). The predicate is also known as the *property* of the triple, and
- an **object** is a resource described with or without an URI, or a literal, where a literal is a string or fragment of XML and is used to identify values such as numbers and dates by means of a lexical representation [KC04].

A triple specified as above represents a *statement* of a relationship between the things denoted by the nodes that it links. The notion of RDF statements corresponds to the structure of simple English sentences. Resources, properties and statements are the underlying three components for the RDF data model [GPFLC04].

From the perspective of graph theory an RDF graph simply is a directed graph with labels on both nodes and edges. In comparison with the description of semantic networks above, RDF graphs are obviously equivalent to the knowledge representation paradigm of semantic networks [CK01].

RDF Schema

The RDF data model provides a generic method to allow modeling of object models in a semantic way, but without any clearly defined semantics. This limitation is lifted by the schema language RDFS (*RDF Schema*), also known as the RDF Vocabulary Description Language [BG04]. RDFS is built on top of RDF and extends it with some simple but basic primitives for modeling RDF classes and properties hierarchically. The combination of RDF and RDFS is then known as RDF(S).

According to [BG04] the RDFS primitives can be grouped into basic classes and properties, container classes and properties, collections, reification vocabulary, and utility properties. RDFS makes use of the specifications, *domain* and *range*, to define subject and object resources associated with each RDF property,

[6] In case no URI description is given, the subject is represented by a blank node which means either that people have no ideas about the name of the node or that the name does not exist at all. In fact not every thing in the real world needs to have its own name.

respectively. On the one hand these specifications enable inferences about the types of things based on the statements; on the other hand they serve as a vocabulary documentation in the web by using XML namespaces [SEMD02].

OIL and DAML+OIL as RDF(S) Extensions

RDF and RDF Schema are supposed to be the most basic KR (*Knowledge Representation*) formalisms for specifying resources in the semantic web. But due to their limited expressivity [AvH04], they can not fully satisfy the need of semantic webs for more modeling power. Therefore a joint initiative was formed (between the years 2000 and 2001), in order to create a richer language, named DAML+OIL[7]. This name is a combination of the names of DAML-ONT[8] and OIL[9].

DAML-ONT stands for *DAML Ontology Language*, where DAML is an acronym for the DARPA project *DARPA Agent Markup Language*. OIL stands for *Ontology Interchange Language* and *Ontology Inference Layer*. This initiative was funded by the European Union IST project On-To-Knowledge and combines techniques from three different communities [HH00, BGH01, FHvH$^+$00]:

- *frame-based systems* for epistemological modeling and frame-based primitives,
- DL (*Description Logic*) for the formal semantics and reasoning support, and
- *web standard languages* with XML and RDF syntax.

OIL has different layers. From low to high they are core OIL, Standard OIL, Instance OIL and Heavy OIL. The higher layer is established on the basis of a lower layer and is more functional and complex [GPFLC04].

Similar to OIL, DAML+OIL was also developed as an extension of RDF(S) with more powerful concepts for describing ontologies, nevertheless it has no layered structures [PS02, CvHH$^+$01]. It extends RDF(S) with DL-based KR primitives. In comparison with RDF(S), DAML+OIL has the following additional features [GPFLC04, AvH04]:

- Ability of inference: DAML+OIL allows reasoning with the expressions `disjointWith`, `TransitiveProperty`, `UnambiguousProperty`, `inverseOf` , `equivalentTo` and so on.

[7] http://www.daml.org/2001/03/daml+oil-index.html
[8] http://www.daml.org/2000/10/daml-ont.html
[9] http://www.ontoknowledge.org/oil/

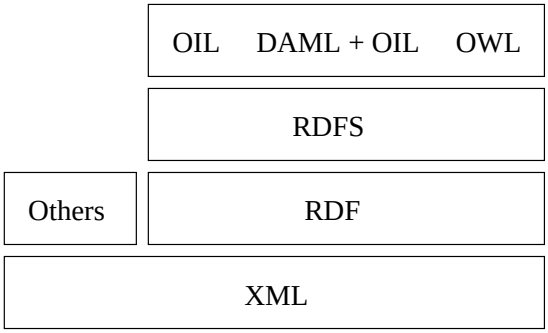

Figure 2.3: XML based ontology markup languages.

- Boolean combinations of classes: DAML+OIL allows class expressions involving `disjointUnionOf`, `unionOf`, `intersectionOf`, or `complementOf`.
- Cardinality restriction: DAML supports cardinality, minCardinality, and maxCardinality for limiting the number of statements with the same subject and predicate.
- Local scope of properties: DAML has the expression `Restriction` which allows local restrictions on certain properties for dealing with exceptions, whereas RDF(S) can only specify restrictions in a global scope.

Figure 2.3 is borrowed from [Dae05]. The most important ontology markup languages are presented in the figure, where the word "others" claims that there are also other XML-based ontology languages, for example XOL (*Ontology Exchange Language*)[10]. The relationships between these languages are clearly illustrated here. In the next subsection OWL will be introduced in detail, because it is the most important and popular ontology language for the semantic web nowadays and it is also the underlying specification language for the OntoBayes model (see Chapter 4).

2.1.4 Web Ontology Language

OWL is developed by the Web Ontology (WebOnt) Working Group[11]. In February 2004 it was officially announced by W3C as a semantic web standard for facilitating to process the content of information instead of just presenting information to humans [MvH04]. Now OWL is the broadly accepted ontology language of the semantic web. It aims to reach maximal compatibility with XML, RDF(S)

[10] http://www.ai.sri.com/pkarp/xol/
[11] http://www.w3.org/2001/sw/WebOnt/

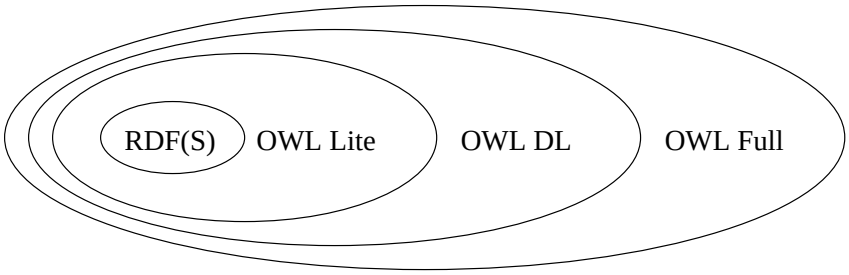

Figure 2.4: Three species of OWL and their set-relations.

and existing ontology languages and logic frameworks. Like DAML+OIL, *OWL* builds upon RDF(S) (see Figure 2.3) with more descriptive power. It derives from DAML+OIL and covers most of its features, therefore it is similar to DAML+OIL. But there are still some differences between them [GPFLC04]. Before demonstrating these similarities and differences, it is important first to investigate the various language elements of OWL in detail.

Three Divisions of OWL

As illustrated in Figure 2.4, OWL is divided into three layers represented by three increasingly expressive sublanguages, respectively, in order to reflect compromises between expressivity and implementability [APM+06].

- **OWL Lite** extends RDF(S) and captures the most essential and basic features of (lightweight) ontologies, in order to facilitate building taxonomies (or hierarchical ontologies) by means of some simple constraints. The motivation to set up this layer is to provide a straightforward implementability for developers with a minimal useful subset of language features [MvH04], but a restricted expressivity.

- **OWL DL** retains the vocabulary of OWL Lite and extends it with new language primitives, in order to make use of the expressivity and computational efficiency as much as possible at the same time. In fact it includes the complete OWL vocabulary, but with some constraints which ensure exploiting the formal underpinnings and the computational tractability of Description Logic. Therefore OWL DL has an efficient reasoning support but is limited to the full compatibility with RDF(s) [AvH04].

- **OWL Full** allows to use all primitives of OWL and gives the users full flexibility to combine them with RDF(S), as long as the result is legal RDF [AvH04]. On the one hand OWL Full is full compatible with RDF(S), but on

the other hand it is very difficult to achieve complete (or efficient) reasoning support, and therefore no computation guarantees are given.

The Primitives of OWL

Syntactically all primitives of OWL are based on the XML syntax. According to features described in Section 2.1.1, most of them can be roughly grouped as follows:

- Primitives for defining concepts
  - `owl:Class` is used to define concepts in ontologies which are known as classes in OWL.
  - `owl:Thing` and `owl:Nothing` are predefined classes and used to specify the most as well as the least general classes, respectively. The former contains all individuals and the latter is empty.
  - `oneOf` can only be used in conjunction with `owl:Class` to define enumerated classes.
  - `owl:equivalentClass` is used for defining equivalent concepts.
  - `disjointWith` states the disjointness of numerous concepts.
  - `owl:intersectionOf`, `owl:unionOf` and `owl:complementOf` are used to define boolean operations of conjunction, disjunction and negation respectively.
- Primitives for defining individuals
  - `owl:sameAs` states the identity between individuals.
  - `owl:differentFrom` states that an individual differs from others.
  - `owl:allDifferent` can only be used in conjunction with `owl:distinctMembers` to state that numerous individuals differ from each other.
- Primitives for defining properties
  - `owl:ObjectProperty` and `owl:DatatypeProperty` are the only two property types in OWL. The former relates objects to other objects, whereas the latter relates objects to datatype values. The datatype properties can use `owl:dataRange` for setting enumerated datatypes or simply use predefined XML Schema datatypes[12].

[12] http://www.w3.org/2001/XMLSchema

  - `owl:equivalentProperty` is used to define equivalent properties.
  - `owl:TransitiveProperty`, `owl:SymmetricProperty` and `owl:inverseOf` can be used to define transitivity, symmetricity and inverse between properties.
  - `owl:FunctionalProperty` defines that a property has a unique value for each object.
  - `owl:InverseFunctionalProperty` indicates that two different objects can not share the same value.

- Primitives for defining constraints
  - `owl:Restrictions` is used to specify the constraints on concepts.
  - `owl:onProperty` indicates which property is restricted with regard to a class.
  - `allValueFrom` and `SomeValueFrom` define the logical universal quantifier and existential quantifier respectively.
  - `hasValue` is used to define that the value of a property is a certain individual, but is not allowed in OWL Lite.
  - `cardinality`, `maxCardinality` and `minCardinality` define restricted cardinality. In the OWL DL and OWL Full synopses the cardinality is arbitrary, whereas in OWL Lite it can be only restricted to either 0 or 1.

Besides these most important ones there are also some primitives for giving header information, for versioning or other special uses in OWL:

- `owl:Ontology` is used to group all headers and versioning information to facilitate the ontology management.
- `owl:imports` is used for importing other ontologies into the current one. It is transitive: If *A* imports *B* and *B* imports *C*, then *A* imports *C*.
- `owl:versionInfo`, `owl:priorVersion`, `owl:backwardCompatibleWith` and `owl:incompatibleWith` are used for versioning.
- `owl:DeprecatedClass` and `owl:DeprecatedProperty` are used to provide the common feature of deprecation by versioning.
- `owl:AnnotationProperty` allows annotations on classes, properties, individuals and ontology headers.

Figure 2.5 presents all primitives in OWL and the corresponding ones in DAML+OIL. It is taken from [GPFLC04], but with slight changes because the original specification is based on the W3C draft documentation for OWL (in 2003) which are out of date. More detailed explanations about the primitives of OWL can be found in [SWM04, MvH04, BvHH+04].

Similarities and Differences between OWL and DAML+OIL

After the introduction of the basic language elements of OWL, now it is possible to summarize the similarities and differences between OWL and DAML+OIL. OWL is almost equal to its predecessor DAML+OIL due to the historical background introduced above. The similarities generally exist at three levels:

- At the level of exchange language syntax the languages are similar. Both of them make use of RDF/XML-based syntax for defining their underlying exchange syntax. OWL renames most of the primitives of the DAML+OIL (see Figure 2.5).
- At the level of abstract syntax [PSHH04] both languages are influenced by the frame paradigms. OWL DL and OWL Lite have a frame-like abstract syntax similar to OIL.
- At the level of formal semantic [Hor05] they are designed based on DL, in order to integrate the key features of DL into OWL and DAML+OWL, the expressivity, automated reasoning and the compositionality [LRDS04].

OWL is so close to DAML+OIL that it can be easily transformed into DAML+OIL [GGP+02]. In spite of major similarities there are still some important differences between them which should be taken into account:

- OWL is more close to OIL than DAML+OIL. It has different layers (see Figure 2.4), while DAML+OIL is more DL-like.
- The primitives are syntactically similar, but still different (see Figure 2.5).
  - OWL adopts the RDF(S) primitives, for example `rdfs:subClassOf`, `rdfs:subPropertyOf` etc., while DAML+OIL renames them.
  - OWL allows defining symmetric properties with `owl:SymmetricProperty` as opposed to DAML+OIL.
  - OWL does not allow qualified number restriction, for example `daml:hasClassQ` etc..
  - OWL makes use of two primitives `owl:unionOf` and `owl:disjointWith` instead of the simple primitive `daml:disjointUnionOf` in DAML+OIL.

OWL DL

```
Class expressions allowed in: rdfs:domain,rdfs:range,rdfs:subClassof
                              owl:intersectionOf,owl:equivalentClass,
                              owl:allValuesFrom,owl:someValuesFrom
Values are not restricted (0..N) in: owl:minCardinality,owl:maxCardinality,owl:Cardinality

owl:DataRange, rdf:List, rdf:first, rdf:rest, rdf:nil

owl:hasValue (daml:hasValueOf)
owl:oneOf (daml:oneOf)
owl:unionOf (daml:unionOf), owl:complementOf (daml:complementOf)
owl:disjointWith (daml:disjointWith)
```

OWL Lite

```
owl:Ontology (daml:Ontology),
owl:versionInfo (daml:VersionInfo),
owl:imports (daml:imports),
owl:backwardCompatibleWith,
owl:incompatibleWith, owl:priorVersion,
owl:DeprecatedClass,
owl:DeprecatedProperty

owl:Class (daml:Class),
owl:Restriction (daml:Restriction),
owl:onProperty (daml:onProperty),
owl:allValuesFrom (daml:toClass) (only with class identifiers and named datatypes),
owl:someValuesFrom (daml:hasClass) (only with class identifiers and named datatypes),
owl:minCardinality (daml:minCardinality, restricted to {0,1}),
owl:maxCardinality (daml:maxCardinality, restricted to {0,1}),
owl:cardinality (daml:cardinality, restricted to {0,1})

owl:intersectionOf (only with class identifiers and property restrictions)

owl:ObjectProperty (daml:ObjectProperty),
owl:DatatypeProperty (daml:DatatypeProperty),
owl:TransitiveProperty (daml:TransitiveProperty),
owl:SymmetricProperty,
owl:FunctionalProperty (daml:UniqueProperty),
owl:InverseFunctionalProperty (daml:UnambiguousProperty),
owl:AnnotationProperty

owl:Thing (daml:Thing),
owl:Nothing (daml:Nothing)

owl:inverseOf (daml:inverseOf),
owl:equivalentClass (daml:sameClassAs) (only with class identifiers and
                                        property restrictions),
owl:equivalentProperty (daml:samePropertyAs),
owl:sameAs (daml:equivalentTo),
owl:differentFrom (daml:differentIndividualFrom),
owl:AllDifferent, owl:distinctMembers
```

RDF(S)

```
rdf:Property
rdfs:subPropertyOf
rdfs:domain
rdfs:range (only with class identifiers and named datatypes)
rdfs:comment,rdfs:lable,rdfs:seeAlso,rdfs:isDefinedBy
rdfs:subClassOf (only with class identifiers and property restrictions)
```

Figure 2.5: Primitives of OWL Lite and OWL DL in comparison with DAML+OIL.

Protégé as An Ontology Editor

This work utilizes the tool Protégé[13] in Version 3.2.1 for developing ontologies in the application domains of natural disaster and catastrophe insurance. As an ontology editor Protégé will mainly be used to satisfy the knowledge representation requirements for users. The graphical visualization of ontologies is realized by using the plugin of *OWLViz*[14] which is integrated with the Protégé-OWL plugin and is implemented by the University of Manchester. For the consistency checking the reasoner RACER[15] can be used.

A new Protégé plugin *OWLOntoBayes* for completing all ontological extensions in OntoBayes (see Chapter 4) is implemented within a student research project [Her07]. The plugin allows users to construct domain specific Bayesian Networks and Influence Diagrams with a graphical user interface (see Chapter 6).

2.2 Uncertainty

There are many systems that are designed and developed based on precision and certainty. They provide unrealizable solutions based on the assumption of closed environments. For the most real applications uncertainty is inevitable and can not be ignored.

2.2.1 Uncertainty Categories

Information *imperfection* is the most difficult, but unavoidable problem faced by agents in an open environment. According to Smets' approach [Sme96] it can be generally grouped into imprecision, inconsistency or uncertainty.

- **Imprecision** presents the ambiguity, vagueness or approximation of information.
- **Inconsistency** expresses that contradictory conclusions can be drawn based on given information or statements.
- **Uncertainty** is caused by a lack of knowledge about the environment when agents need to decide the truth of statements. Uncertainty can be distinguished objectively and subjectively. *Objective uncertainty* relates to randomness which likely qualifies the occurrence possibility of an event, whereas *subjective uncertainty* depends on the subjective opinions of agents about the truth value of information.

[13] http://protege.stanford.edu/
[14] http://www.co-ode.org/downloads/owlviz/co-ode-index.php
[15] http://www.racer-systems.com/products/tools/index.phtml

Imprecision and inconsistency are essential properties related to information content whereas uncertainty is a property of the relation between the information and our knowledge about the world.

Besides the classification based on Smets' approach, another viewpoint describing perspectives on computational perception and cognition under uncertainty, is proposed in [SG00]. Two broad categories of uncertainty, *U-Type One* and *U-Type Two,* are suggested:

- The first type of uncertainty deals with information arising from the random behavior of physical systems.

- The second type of uncertainty deals with information arising from human perception and cognition processes.

The first type has been investigated for centuries with efforts of statistical theory. The statistical methodologies are very useful to model this type, but lack the sophistication to process the second type. In order to deal with the second type, several effective methods were been proposed, e.g. fuzzy logic, neural networks and so on.

In the next two sections we will give an simple overview of the fuzzy and probabilistic approaches for representing uncertainty. Alternative approaches of uncertainty modeling and logical inference exist, e.g. the Dempster-Shafer approaches [Dem67, Dem68, Sha76] etc.. In this work, we make use of the probabilistic approach, BNs, to deal with uncertainty for the framework of DSSs.

2.2.2 The Fuzzy Approach

Fuzzy logic is derived from fuzzy set theory, which was introduced by Lotfi A. Zadeh in 1965. It was developed for looking at vagueness in a mathematical way. Its basic idea is to allow expressing of the membership relation between an object and a set by a membership function ranging from 0 to 1. By contrary the classic (or crisp) set allows the membership function to take on only two values 0 and 1. Let X be a set of objects, with generic elements noted as x. Then a fuzzy set can be formally defined as follows [Zad65]:

Definition 2.3 *A fuzzy set A in X is characterized by a membership function $f_A(x)$ which maps each object $x \in X$ to a real number in the interval $[0,1]$.*

The value of $f_A(x)$ represents the "the grade of membership" of x in A with the implication that the nearer the value is to 1, the higher is the degree of membership.

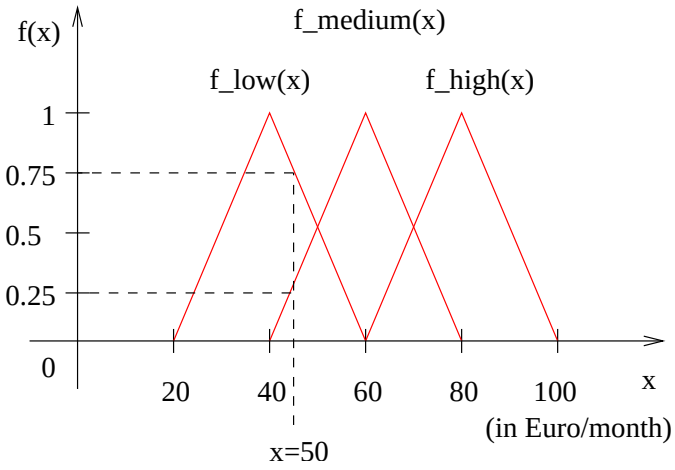

Figure 2.6: Fuzzification for the premium of an insurance product.

A number of elementary operations can be used for modeling fuzzy systems. Basic operations in sets (union, intersection, complement) that correspond to boolean operators (OR, AND, and NOT) have their equivalent in fuzzy sets and fuzzy logic. These basic operations satisfy some axioms, for example, De Morgan's law, Monotonicity, Associativity, Commutativity and so on [Zad65]. In addition to logical operations, there are also algebraic operations. For instance, the algebraic product, algebraic sum, algebraic difference, convex combination and so on [Men01].

Fuzzy Modeling

In principle a fuzzy system can be modeled by the following basic steps according to [Ibr01]:

- The first step is to identify the inputs and outputs of the system using fuzzy variables described in linguistic terms. For example, "Premium", "Risk of Coverage" as inputs and "Buy Product" as an output for decisioning to purchase an insurance product.

- The second step is the *fuzzification* which comprises the process of transforming crisp values into degrees of membership with fuzzy sets described with linguistic terms. For example, for the fuzzification of an insurance product premium $x = 50$ €/month, the three membership functions $f_{low}(x)$, $f_{medium}(x)$ and $f_{high}(x)$ as depicted in Figure 2.6, can be used. They characterize a low, medium and high premium fuzzy set, respectively. The given premium $x = 50$ €/month belongs with a degree of $f_{low}(x) = 0.75$ to the

fuzzy set "low" and with a degree of $f_{medium}(x) = 0.25$ to the fuzzy set "medium".

- The core step is to build a *fuzzy inference machine* with a set of rules relating the input condition to the output responses. These rules can be expressed as a list of IF-THEN pairs: IF <fuzzy proposition> THEN <fuzzy proposition>, where the first fuzzy proposition of the pair is the input condition, and the second fuzzy proposition is the output response. For instance, IF *Premium=low* AND *Risk of Coverage=full* THEN *Buy Product=yes*.

- The last step is the *defuzzification* which is a process to convert the fuzzy value to a single crisp value, because the result of all rules that have fired within the inference machine is a fuzzy set. In order to apply the fuzzy set in the application, it is required to reduce it to a crisp value representing the set. There are many methods of defuzzification, each with various advantages and drawbacks [BBSS+00]. Among them *center of gravity* and *maximum* methods are the two most used. But the final choice of defuzzification methods can only be decided based on the user's experience and the way of thinking.

2.2.3 The Probabilistic Approach

The probabilistic approach was introduced in the 1600's and originally was investigated for analyzing games of chance in a mathematical way [OGD80]. After that it has been widely developed and applied to many fields. In the field of AI many research efforts were made to use probabilistic methods for dealing with uncertainty in KBS (*Knowledge-based Systems*) [DHN90].

Interpretation of Probability

There are four main interpretations of *probability*: the frequency theory, the propensity theory, chance and Bayesianism (see [Wil05] for a survey). They can be grouped broadly into objective or subjective probability.

- **Objective probability** (or physical probability) is a classical approach to probability and was developed based on the classical definition of probability that Pierre Simon Laplace identified in his work *Théorie analytique des probabilités* [dL96]. It defines the probability of an event as the "limit" of its relative frequency in a large number of trials.

  Mathematically it can be represented as follows: If a random experiment can result in n mutually exclusive and equally likely outcomes and if m of

these outcomes result in the occurrence of the event A, the probability of A is defined by $P(A) = \frac{m}{n}$.

Objective probability reflects the physical interpretation and includes the frequency and propensity interpretation.

- **Subjective probability** (or Bayesian probability)[16] is an alternative way to measure the probability of an event relying on the subjective opinions of agents, where the probability is often interpreted as a degree of belief about an event or a proposition. For example two agents can have different degrees of belief about the same proposition even though they have the same background knowledge.

  Subjective probability reflects the mental interpretation and includes the chance and Bayesianism interpretation.

Foundations of Probability

As mentioned above there are many ways to interpret probability. Therefore it is important to define common and basic concepts independently from the different interpretations, in order to understand the basis of probability theory. This is usually done by relying on *Kolmogorov's axioms*. Before introducing the axioms, it is necessary to define the probability space mathematically which is the foundation of probability theory.

Definition 2.4 *A probability space* $(\Omega, \mathscr{A}, P)$ *is a measure space such that*

$$P(\Omega) = 1,$$

where the non-empty set Ω *is the sample space,* $\mathscr{A}$ *the* σ*-algebra on* Ω *and* P *the probability measure (or probability) [Bau02].*

Each element of the sample space Ω is an atomic event ω[17]. Each element of the σ-algebra $\mathscr{A}$ is an *event* A which is a collection of *atomic event*s determined by some set-algebraic rules governed by the laws of Boolean algebra. Two events, A and B are said to be *mutually exclusive* or *disjoint* if $P(A \cap B) = 0$. For example, union, intersection, complement and De Morgan's law. A probability P is a positive function

$$P : \mathscr{A} \to \mathbb{R}.$$

It is a mapping from σ-algebra $\mathscr{A}$ to the space of real numbers $\mathbb{R}$ restricted to the interval $[0, 1]$.

[16] Subjectivists, also known as Bayesians or followers of *epistemic probability*, which emphasizes the close link between knowledge and probability.

[17] In the literature, atomic events are often referred to *sample points* and define a random variable as a function taking an atomic event as input and returning a value from the appropriate domain.

Kolmogorov's Axioms

The following three axioms are known as the *Kolmogorov's axioms* developed by the Russian mathematician Andrei Kolmogorov, who built up the rest of probability theory based on the simple foundations as explained above [PP02].

1. All probabilities are between 0 and 1. For any event A,

$$0 \leq P(A) \leq 1. \tag{2.1}$$

2. The probability of the certain atomic event in the entire sample space equals 1:

$$P(\Omega) = 1. \tag{2.2}$$

   It means also that there are no atomic events outside the sample space.

3. If two events A and B are mutually exclusive, then

$$P(A \vee B) = P(A) + P(B). \tag{2.3}$$

From these axioms some consequences can be deduced for calculating probabilities. For example it can be extended to the addition law of probability (or the sum rule),

$$P(A \vee B) = P(A) + P(B) - P(A \wedge B), \tag{2.4}$$

the inclusion-exclusion principle,

$$P(\Omega \backslash A) = 1 - P(A), \tag{2.5}$$

and the probability of the impossible event

$$P(\emptyset) = 0. \tag{2.6}$$

Events are also known as propositions by Bayesianism. In this work we use the term "random variables" to simplify the notation, by allowing to speak about $P(X = x)$ as a function of x instead of having to take into consideration a huge number of unlinked events [Dan06]. In the algebraic axiomatization of probability theory, the primary concept is not that of probability of an event or of a proposition, but rather that of a *random variable*. In this thesis we only take *discrete* random variables into account, which include boolean random variables as a special case, with values from an enumerable domain. We use an uppercase letter as the first letter for random variables and lowercase ones for their values. The domain of the variable will be denoted as $dom(X)$. For instance, the domain of the variable "premium" is given by $dom(Premium) = \{low, medium, high\}$.

2.3 Bayesian Networks

As a probabilistic approach, BNs are selected for modeling uncertain knowledge in this work.

2.3.1 Definitions

BNs[18] provide a natural way to represent and reason about uncertain knowledge on a formal theoretical basis. Before we introduce the definition of BNs formally, we will first define some terminologies which are adopted from standard graph theory and will be used to facilitate the explanation of BNs.

Definition 2.5 (Directed Graph) *Given a set of vertices (or nodes) $\mathscr{V}$, a directed graph (or digraph) is a graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ with $\mathscr{E} \subseteq \mathscr{V} \times \mathscr{V}$.*

All edges in a digraph are directed. They are often called the arrows of the digraph. The number of nodes in the digraph will be denoted as $\sharp\mathscr{V}$.

Definition 2.6 (Directed path) *A directed path P in a digraph is a sequence of consecutive directed edges. It can be denoted as $P = X_1 \rightarrow ... \rightarrow X_n$ (or $P = (X_1, ..., X_n)$), where $X_i \in \mathscr{V}, i = 1, ..., n$.*

We make use of $\sharp P$ to denote the length of a path P. It equals the number of edges in the path, i.e., for a path with n nodes the length is $n-1$. The ith element of the path is a node and can be denoted as $P[i]$. For example for the path $P = (X_1, X_2, X_3), X_1, X_2, X_3 \in \mathscr{V}$, $\sharp P$ is 2 and $P[1]$=X_2.

Definition 2.7 (DAG) *A DAG is a digraph without any directed path $X_1 \rightarrow ... \rightarrow X_n$ such that $X_1 = X_n$.*

For a node $X \in \mathscr{V}$, the *parents* of X are nodes from which there is an arrow in $\mathscr{G}$ going to X. The set of parents is denoted as $par(X)$. The set of *children* of X (denoted as $chi(X)$) are nodes reached by an arrow starting from X. The *descendants* of X (denoted as $des(X)$) is the set of nodes which are offspring, or offspring of offspring, etc. of the given variable X, while the *ancestors* of X (denoted as $anc(X)$) are the variables which are parents, or parents of parents, etc. of X.

Professor Jensen, one of the most influential researchers in this field, defined BNs in [Jen96, Jen01] as follows:

Definition 2.8 *A BN consists of the following:*

[18] BNs are also known as *Bayesian nets, probabilistic networks, Bayesian belief networks* or *belief networks.*

- *A set of variables and a set of directed edges between variables.*
- *Each variable has a finite set of mutually exclusive states.*
- *The variables together with directed edges from a DAG.*
- *To each variable X with parents $Y_1,...,Y_n$, there is attached the potential table $P(X|Y_1,...,Y_n)$.*

A BN is a probabilistic graphical model whose main strength is due to the fact that it can represent both qualitative information as well as quantitative information. According to this strength Definition 2.8 can be refined as follows [Wil05]:

Definition 2.9 *Given a finite set $\mathscr{V}$ of discrete variables*[19]*, a BN $\mathscr{B} = (\mathscr{G}, \mathscr{S})$ on $\mathscr{V}$ consists of a qualitative component $\mathscr{G}$ and a quantitative component $\mathscr{S}$:*

- *The qualitative component is a DAG $\mathscr{G} = (\mathscr{V}, \mathscr{E})$, where $\mathscr{V}$ and $\mathscr{E}$ are respectively the sets of vertices*[20] *and directed edges in the graph.*
- *The quantitative component contains a set of probability specifications $\mathscr{S}$. For each variable $X \in \mathscr{V}$, $\mathscr{S}$ specifies the probability distribution of X.*

This work uses this *definition* as its underlying theory basis for uncertainty modeling. We will begin our explanation with the quantitative part first.

2.3.2 The Quantitative Information

The quantitative part of a BN contains the numerical information representing probability distributions associated with Bayesian variables. This part describes how the variables relate to each other quantitatively. There are two types of probability distributions in BNs: either unconditional or conditional. Such distributions are represented as probability tables.

Unconditional Probability

In order to describe what is known about a variable X ($X \in \mathscr{V}$) in the absence of any other evidence, Bayesian probability uses the *unconditional* (or *prior*) probability. It is written as $P(X)$. An unconditional probability is normally the purely subjective assessment of an experienced expert. For example, given a Bayesian variable *Term* independent on any other variables and $dom(Term) =$

Term=short	Term=long
0.4	0.6

Table 2.1: The unconditional probability table for $P(Term)$.

$\{short, long\}$, then a possible probability distribution of $P(Term)$ can be represented as in Table 2.1.

The unconditional probability of a set of variables $X_1, ..., X_n$ in a conjunction can be written as $P(X_1 \wedge ... \wedge X_n)$ (or $P(X_1, ..., X_n)$ for short). It can be expressed by the JPD (*joint probability distribution*) of these variables. For example, given a variable *RiskCoverage* and $dom(RiskCoverage) = \{low, medium, high\}$, then the JPD of $P(Term, RiskCoverage)$ can be represented by a 2×3 table with 6 probability entries.

Conditional Probability

Once agents have observed some evidences which influence over the predefined variables, unconditional probabilities are no longer appropriate. In this case, we use *conditional* (or *posterior*) probability. The notation used is $P(X|Y)$, where $X, Y \in \mathscr{V}$. It can be read as "the probability of A given B". Conditional probabilities can be defined in terms of unconditional probabilities:

$$P(X|Y) = \frac{P(X \wedge Y)}{P(Y)} \tag{2.7}$$

where $P(Y) > 0$. For example, $P(Premium = low|RiskCoverage = high) = 0.1$ indicates that if the risk coverage of an insurance product is observed to be high and no other information is available yet, then the probability that the product has a low premium will be 0.1. Equation 2.7 can also be written as

$$P(X \wedge Y) = P(X|Y)P(Y), \tag{2.8}$$

which is called the *product rule* [RN03]. Starting from Equation 2.7 and 2.8 we can derive *Bayes' theorem*:

$$P(X|Y) = \frac{P(Y|X)P(X)}{P(Y)}. \tag{2.9}$$

The conditional probability $P(X|Y)$ indicates the dependency (or influence) relation between variables X and Y: X depends on Y (or Y influences X). In order

[19] It is possible to work with BNs involving continuous variables, for example, variables subject to Gaussian distribution. In this work we only take into account discrete variables with finite states.

[20] The set $\mathscr{V}$ of vertices is the set of variables on which the BN is defined.

	Premium=low	Premium=medium	Premium=high
RiskCoverage=low	0.6	0.2	0.1
RiskCoverage=medium	0.3	0.5	0.2
RiskCoverage=high	0.1	0.3	0.7

Table 2.2: The conditional probability table for $P(Premium|RiskCoverage)$.

to represent the quantitative information of $P(X|Y)$, the CPT (*Conditional Probability Table*) is introduced. For instance, $P(Premium|RiskCoverage)$ expresses that the premium of an insurance product depends on its risk coverage. Table 2.2 presents the CPT of $P(Premium|RiskCoverage)$.

A useful generalization of the product rule (Equation 2.8) is the *chain rule*. Let $X_1,...,X_n$ be a set of *n* variables, then the JPD $P(X_1,...,X_n)$ can be written as a product of *n* conditional probabilities via repeated application of Equation 2.8:

$$P(X_1,...,X_n) = P(X_1,...,X_{n-1})...P(X_2|X_1)P(X_1). \tag{2.10}$$

Let $par(X)$ denote all parents of variable X (see Page 26), then Equation 2.10 can be formulated as follows:

$$P(X_1,...,X_n) = \prod_{i=1}^{n} Pr(X_i|par(X_i)). \tag{2.11}$$

Definition 2.10 (Conditional independence) *Given $X,Y,Z \subseteq \mathscr{V}$, X and Y are conditional independent given Z, iff*[21]

$$P(X|Y,Z) = P(X|Z). \tag{2.12}$$

In this work we make use of Dawid's notation $(X \perp Y|Z)_p$ or simply $(X \perp Y|Z)$ to denote conditional independence [Daw79]. Some properties can be followed from Definition 2.10, for example, symmetry, decomposition, weak union, contraction and intersection [Pea00].

2.3.3 The Qualitative Information

The qualitative part of a BN describes the structure information represented by a DAG over its vertices, where the vertices here correspond to variables of a BN (whence the common symbol $\mathscr{V}$). Edges in a DAG denote a certain relationship which holds on pairs of variables. Normally the relationship indicates the statistical dependency between variables, but it varies with the application. For example, it can be interpreted as causal relationship in a causal network [Pea00].

[21] "iff" is shorthand for "if and only if".

The Dependency

In this work, an edge $X \rightarrow Y$ indicates the *dependency* or influence relation between variables X and Y: Y depends on X or X influences Y. The notation $X \rightarrow Y$ can express the same meaning as the notation for conditional probability $P(Y|X)$. For example $P(Premium|RiskCoverage)$ (see Table 2.2) can be expressed here with the graphical notation $RiskCoverage \rightarrow Premium$.

Definition 2.11 (Markov Blanket) *A Markov Blanket of a node X in a BN $\mathscr{B}$ is a set of nodes, which is composed of $par(X)$, $chi(X)$ and $par(Y_i)$, $\forall Y_i \in chi(X)$.*

The Markov Blanket of X can be denoted as $mb(X)$. It is important because it is the only knowledge that is needed to predict the behavior of that node X in a BN. Based on Definition 2.11 the following two theorems can be considered (and can be proven [Pea88]).

Theorem 2.12 (Markov Condition) $P(X \perp Y|par(X)), \forall X \in \mathscr{V}$ *and* $Y \in \mathscr{V} \backslash des(X)$.

Theorem 2.13 $P(X \perp Y|mb(X)), \forall Y \in \mathscr{V} \backslash (mb(X) \cap \{X\})$ *in a BN* $\mathscr{B}$.

Hypothesis and evidence

Evidence is information about a certain situation via observation. If the variable represented by a node is *observed*, then the node is said to be an *evidence* node. There are three types of evidences that can be applied to BNs:

- Hard evidence is an instantiation of a variable E with $E = e$. It means, that a variable is with a degree of 100% in one state, and with a degree of 0% in all other states [BF05].
- Soft evidence means $E \neq e$, i.e., it is only known that E does not have state e [BF05].
- Virtual evidence is a method widely adopted in BNs inference and was presented in [Pea97]. It is the likelihood of a variable's distribution. The likelihood is presented by the probability of observing E being in state e [Din05].

As opposed to observed variables, variables whose values are not known are called *hidden* variables. Based on the definition of observed variables (or evidences[22]) and Bayes' theorem (Equation 2.9), the essence of BNs can be expressed via the following formula:

$$P(H|e) = \frac{P(e|H)P(H)}{P(e)}, \tag{2.13}$$

[22] In this work when we speak about evidences, they are automatically considered as hard evidences.

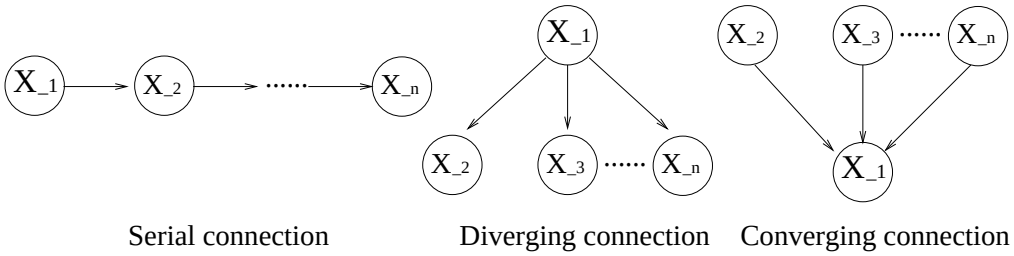

Figure 2.7: Three types of connections in BNs.

where H is a hypothesis. It shows how to compute the belief about a hypothesis upon observing evidence *e* in the Bayesian inference [Pea00].

d-separation

The purpose of introducing d-separation is to describe how a change of certainty in one variable may change the certainty for other variables.

In Figure 2.7 three types of connections in BNs are illustrated. The first type is serial connection. The influence of an evidence *e* can be transmitted through the serial connection unless any variable in the connection is instantiated. The second type is diverging connection. The influence of an evidence *e* can only be passed to the children $chi(X_1)$ unless the state of X_1 is known. The third type is converging connection. The influence of an evidence *e* can only be passed to the parents $par(X_1)$ if either X_1 or any variable of $des(X_1)$ is instantiated. Based on these three types of connections we can define d-separation as follows [Jen96]:

Definition 2.14 (d-separation) *Given $X, Y \in \mathscr{V}$ $(X \neq Y)$ in a BN, X and Y are d-separated if for all paths between X and Y, there is an intermediate variable Z such that either*

- *the connection is serial or diverging and Z is instantiated, or*
- *the connection is converging, and neither Z nor any of $des(Z)$ has received evidence.*

A and *B* are *d-connected*, if they are not d-separated. If X and Y are d-separated, then changes in the belief of X have no impact on the belief changing of Y.

Proposition 2.15 *$\forall Y \in MB(X)$ in a BN, if Y is instantiated, then X is d-separated from Z, where $\forall Z \in \mathscr{V} \setminus \{X\} \cap MB(X)$.*

2.3.4 Bayesian Reasoning

Bayesian reasoning is statistical inference for computing the belief change (or the posterior probability distribution) for a set of query variables by given observations. Simply it is a process used to draw conclusions from evidences. The algorithms in BNs make use of the principles of probabilistic reasoning and Bayes' theorem. A lot of algorithms have been developed for both exact and approximate reasoning. For a survey of them refer to [GH02].

Exact Reasoning

Approaches of exact reasoning are based on exploration of the causal structures in BNs for efficient computation. There are many algorithms for exact reasoning in BNs. These include the variable elimination algorithm [ZP94], the polytree algorithm [Pea97], the clique-tree algorithm [LS88], the junction tree algorithm[23] [Cow98] and so on.

The computation complexity of inference algorithms depends on the structure of a BN. When the structure is in polytree, it can be linear in the size of the network, where the size is defined as the number of CPT entries. But if the structure of a BN is not in polytree, then the algorithm in general is NP-hard[24] [Coo90].

Approximate Reasoning

Approaches of approximate reasoning are normally based on stochastic simulation which aims to give fast, accurate approximations to posterior probabilities in BNs by reducing the time and space complexity of exact reasoning approaches. It is adequate for very large networks. Like for the exact reasoning, there are also many algorithms for the approximate approaches, for example, importance sampling algorithms [Hen87, FC89, FF94, CD00] and Markov Chain Monte Carlo (MCMC) methods [RC04]. Generally the approximate algorithms have been proven to be NP-hard [DL93].

Each exact or approximate algorithm has different properties and is adequate for different classes of inference problems. It is not easy to select one among them for a given BN due to the *algorithm selection problem* [Ric76]. Many research efforts are devoted to deal with this problem based on empirical methods [Bor96], the approach of algorithm analysis [Wil97] and the approach of system analysis [NJ96]. In this work we do not focus on developing, optimizing or selecting any

[23] The clique-tree algorithm and the junction tree algorithm together are also known as the clustering algorithms.

[24] Roughly speaking, NP-hard is a mathematical term to indicate that it is impossible to compute a solution within reasonable time.

inference algorithms, instead we make use of an existing tool which provides reasoning support for our DSS.

Backward vs. Forward reasoning

All reasoning algorithms can be grouped into two types: forward (or deductive) reasoning and backward (abductive) reasoning.

- The *forward reasoning* simply aims to draw new conclusions by given initial facts. It does not increase the qualitative knowledge base in BNs but changes the quantitative part, therefore it is also known as *belief updating*. Normally it is unnecessary to use Bayes' theorem in the forward reasoning.
- The *backward reasoning* simply aims to explore the best explanations by given hypothesis (or goals). It is more complex because it tries to favor one conclusion above others. This means that it attempts either to falsify alternative explanations, or to show the likelihood of the favored conclusion. The backward reasoning is also known as *belief revision*. If the queried variables are all hidden variables, then it is known as the MPE (*Most Probable Explanation*). Bayes' rule is often applied in the backward reasoning.

2.3.5 Features of Bayesian Networks

Based on the probability theory, BNs possess the following nice features [Bru02]:

Consistency Consistency means that reasoning results in BNs are free from paradoxes and internal contradiction. The results do not change and do not depend on the route along which the available information is processed.

Unique Once the prior knowledge is fixed, BNs lead to unique conclusions. This means that there is only a unique way from "input" to "output".

Plausibility The probabilistic approach extends crisp logic by uncertainty represented through probability. Based on this approach the quantitative techniques of BNs can replicate the essential features of plausible reasoning (reasoning under conditions of uncertainty) which seems more logical to a human being and is mathematically sounder.

So far we have reviewed the basic concepts of BNs. The value of this methodology lies entirely in what can be done with it. In [Gly01] three utilities of BNs are pointed out:

Predication The use of BNs for predication is obvious due to its natural way of forward reasoning.

Control This functionality can also be understood under the term *diagnosis*. The underlying technique is backward reasoning.

Discovery In many cases the structure of the network for representing a domain application can be discovered from experiments, observations, data, and background knowledge.

In this work we are only interested in the first utility due to their significant meaning for DSS.

2.4 Influence Diagrams

IDs[25] were introduced by Howard and Matheson [HM84] and have visibly affected the development of decision making under uncertainty in the field of AI, for example in applications of medical diagnosis, control systems and so on [Bou05]. An ID combines a BN with additional node types for choices (or actions) and utilities. Before giving the formal definitions of IDs, it is meaningful to introduce some important concepts about utilities.

2.4.1 Utility

Probability theory (inclusive BNs) provides an agent's belief on the basis of evidences in a quantitative way. But it is still required to find a framework to describe the desired action of the agent and its rational behavior. In order to deal with this aspect, *Influence Diagrams* makes use of *utility* to model and assess the preference among decision outcomes.

Choice, Preference and Utility

Let C be a finite set of *choices*[26] $\{c_1, ..., c_n\}$ that an agent as a decision maker can select, then a *preference* relation on C can be introduced for ranking all decision choices. This relation is defined as follows.

Definition 2.16 (Preference) *An agent's preference on C is a binary relation $\succeq$ over $\{c_1, ..., c_n\}$ (the choice set of C) with $\succeq \subseteq V_D \times C$, i.e. it is a subset of all ordered pairs*[27] *(c_i, c_j), where $c_i, c_j \in C$ and $i \neq j$.*

[25] They are also known as *decision networks* or *decision graphs* (see [Fen01]).
[26] They are also known as *alternatives* or *actions* in some literatures.
[27] It is often the case that an ordered pair will be called simply a pair.

We can write $(c_i, c_j) \in \succeq$ or $c_i \succeq c_j$ to indicate that the choice c_i is at least as good as another choice c_j. More precisely we can refine the preference $\succeq$ by means of $c_i \succ c_j$ and $c_i \sim c_j$, where $c_i \succ c_j$ and $c_i \sim c_j$ mean c_i is strictly preferred or equivalent to c_j respectively. There are some semantic constraints on preferences:

- Reflexivity: $\forall c \in C$,
$$c \succeq c.$$
- Transitivity: $\forall c_i, c_j, c_k \in C$ and $i \neq j \neq k$,
$$c_i \succeq c_j, c_j \succeq c_k \Rightarrow c_i \succeq c_k.$$
- Orderability: $\forall c_i, c_j \in C$, there must be a preference between c_i and c_j.
$$(c_i \succ c_j) \cup (c_j \succ c_i) \cup (c_i \sim c_j).$$

There are also other constraints, for example, continuity, substitutability, monotonicity and decomposability. A survey of them can be found in [RN03].

Rather than ranking the choices just by setting preferences on them, we can quantify them with the help of utility which is considered as a numerical rating and can be assigned to each possible choice c_i of C. The utility can be expressed as a *utility function* which maps each choice c_i to a real number.

Definition 2.17 (Utility Function) *A utility function on C is a map*

$$u : C \to \mathbb{R}.$$

The existence of the utility function follows the utility principle defined as follows:

$$u(c_i) > u(c_j) \Leftrightarrow c_i \succ c_j \text{ and}$$

$$u(c_i) = u(c_j) \Leftrightarrow c_i \sim c_j.$$

An axiomatic extension of the ordinal concept of utility to uncertain payoffs is the EU (*Expected Utility*). When an agent chooses c as its decision from the choice set C, then all possible outcomes of the decision can be denoted as $Out_i(c)$, where the index i ranges over the different outcomes. Now we can define the expected utility of the choice c (denoted as $EU(c)$) as follows:

$$EU(c) = \sum_i Pr(Out_i(c))U(Out_i(c)) \tag{2.14}$$

A rational agent should make a best decision among all choices (or actions) according to the principle of *MEU* which indicates that a rational decision under

conditions of uncertainty is the decision with the greatest expected utility, i.e. the result of the maximization of the agent's expected utility. Formally it means that $\forall c \in C$, $\exists c' \in C$, $c \neq c'$ such that $EU(c') \succeq EU(c)$. Simply this fact can be expressed as follows:

$$MEU(C) = \max_{c \in C} EU(c). \tag{2.15}$$

Combined Equation 2.15 and 2.14 we can derive

$$MEU(C) = \max_{c \in C} \sum_i Pr(Out_i(c))U(Out_i(c)). \tag{2.16}$$

2.4.2 Formal Definition of IDs

Based on the definitions from [HM05, Jen01, LD06] we can formally define IDs as follows:

Definition 2.18 *Given a set of nodes $\mathscr{V}$ associated with the following function:*

$$f : \mathscr{V} \rightarrow \{\bigcirc, \square, \diamond\},$$

then an ID is a DAG $\mathscr{G}_{id} = (\mathscr{V}, \mathscr{E})$ with

$$V_C = \{v \in \mathscr{V} | f(v) = \bigcirc\},$$
$$V_D = \{v \in \mathscr{V} | f(v) = \square\},$$
$$V_U = \{v \in \mathscr{V} | f(v) = \diamond\},$$
$$E_C = \{e = (v,w) \in \mathscr{E} | v \in \mathscr{V} \backslash V_U \text{ and } w \in V_C\},$$
$$E_D = \{e = (v,w) \in \mathscr{E} | v \in \mathscr{V} \backslash V_U \text{ and } w \in V_D\},$$
$$E_U = \{e = (v,w) \in \mathscr{E} | v \in \mathscr{V} \backslash V_U \text{ and } w \in V_U\},$$
$$\text{and } \mathscr{E} = E_D \cup E_C \cup E_U.$$

Definition 2.18 expresses that an ID includes three types of nodes: decision nodes V_D, chance nodes V_C and value nodes V_U, i.e. $\mathscr{V} = \{V_D\} \cup \{V_C\} \cup \{V_U\}$.

- *Chance nodes V_C* represent propositional (or random) variables that are not controlled by the decision maker. Like the nodes in BNs each of them is associated with a probability distribution over its domain of values. Graphically they are represented by circles $\bigcirc$.
- *Decision nodes V_D* represent the possible choices available to the decision maker. Graphically they are represented by boxes $\square$.

- *Utility nodes*[28] V_U represent the agent's utility function over $par(V_U)$. They are not allowed to have children nodes. Graphically they are represented by diamonds $\diamond$.

  Originally, utility nodes were designed without states. In the extended framework MOID (*Multi Objective Influence Diagrams*) [DH04], they are allowed to have multiple states for expressing different objectives.

Additionally there are three types of edges (or arcs) for expressing different relationships [TS90].

- *Informational arcs* E_D are arcs into decision nodes and indicate which information is known to the decision maker at decision time.

- *Conditional arcs* E_C are arcs into chance nodes and indicate the probabilistic dependency on the associated variables.

- *Functional arcs* E_U are arcs into a utility node[29] and indicate which variables are functionally dependent on the utility node. More simply, they are variables used as decision criteria for utility nodes.

Based on Definition 2.18 two additional assumptions are necessary for evaluating IDs.

Remark 2.19 (Total order) *If there are totally n decision nodes in an influence diagram $\mathscr{G}_{id}$, then there must be a path $P = (D_1, \ldots, D_n)$ in $\mathscr{G}_{id}$.*

This assumption allows decision making in sequence.

Remark 2.20 (No-forgetting) *Let I_i denote the set of all observed chance nodes when making a decision on the decision node D_{i+1}, then I_i must be available when decision D_j is made (for $j > i+1$).*

The *no-forgetting* assumption expresses that the decision maker must remember the past observations and decisions. I_0 is the set of variables observed before the first decision is taken. And I_i ($i > 0$) indicates that all variables are observed after the $(i-1)$th decision but before the ith decision. It is obviously that all parents of D_i ($par(D_i)$) are also parents of D_j.

Figure 2.8 illustrates a simple example of deciding to buy an insurance product by means of an ID. There are two chance nodes, one decision node and one

[28] They are often called *value nodes* in the literature. According to [RN03] we use this term to maintain the distinction between utility and value functions.

[29] Generally there is only one utility node in an ID. When there are more than one, then the associated utility is a sum or product function of all utility nodes [LD06].

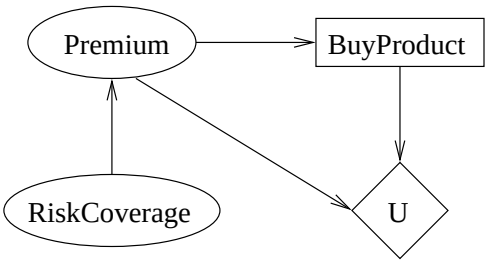

Figure 2.8: A simple Influence Diagram.

U(yes,low) = -100
U(no,low) = 50
U(yes,medium) = 40
U(no,medium) = 10
U(yes,high) = 80
U(no,high) = -80

Table 2.3: The utility table for $U(Premium, BuyProduct)$.

utility node. They are defined as follows: $dom(Premium) = \{low, medium, high\}$, $dom(RiskCoverage) = \{low, medium, high\}$, and $dom(BuyProduct) = \{yes, no\}$. The utility node U can be specified as $U(par(U)) = U(BuyProduct, Premium)$ in Table 2.3. This table reflects the preferences of a decision maker.

2.4.3 Evaluating Influence Diagrams

The main objective of evaluating IDs is to use the quantification of a decision maker's preferences to determine an optimal solution for a decision problem.

Policy, Strategy and Solution

Before we investigate the different evaluating algorithms, it is necessary to introduce some related terms that are defined as follows according to [Jen01]:

Definition 2.21 (Policy) *Given an ID $\mathscr{G}_{id} = (\mathscr{V}, \mathscr{E})$ and for each decision node $D \in V_D$, there is a policy δ which is a mapping:*

$$\delta : dom(par(D)) \rightarrow dom(D).$$

A decision node has many different policies dependent on the different configurations of its parents. For example, $\delta : (Premium = low) \rightarrow BuyProduct = yes$ is a policy of the decision node *BuyProduct* in Figure 2.8, and $\delta : (Premium =$

$low) \rightarrow BuyProduct = no$ is also one. Generally we can simply denote all policies of D as δ_D. All policies of *BuyProduct* then is the set $\delta_{BuyProduct}$.

Definition 2.22 (Optimal policy) *Given an ID $\mathcal{G}_{id} = (\mathcal{V}, \mathcal{E})$, a policy δ^* for a decision node D is optimal iff*

$$EU(\delta^*) \geq EU(\delta), \forall \delta \in \delta_D.$$

The value of a policy δ is the expected utility given that the decision nodes are executed according to δ.

Definition 2.23 (Strategy) *Given an ID $\mathcal{G}_{id} = (\mathcal{V}, \mathcal{E})$, a strategy for $\mathcal{G}_{id}$ is a set of policies δ_i,*

$$\delta_i : dom(par(D_i)) \rightarrow dom(D_i)$$

one for each decision node $D_i \in V_D$, $i = 1, ..., \sharp V_D$.

Definition 2.24 (Solution) *A solution for an ID is a strategy that maximizes the expected utility.*

Evaluating Algorithms

There are different algorithms for evaluating IDs. The foundation of all evaluating algorithms is the probabilistic inference algorithm. The intuitive way to evaluate IDs is transforming them into BNs. After that all Bayesian inference algorithms can be used to evaluate them.

Cooper proposed a method that primarily aims to reduce IDs to BNs [Coo88]. He set up some rules for converting decision nodes and utility nodes to chance nodes. After construction has been completed, the problem of finding an optimal decision in IDs is automatically reduced to the same problem in BNs for which a number of solving algorithms exist.

The algorithm proposed by Shachter and Peot [SP92] is actually very similar to the original one from Shachter [Sha86], but it is more efficient. It simply utilizes the feature of Bayesian independency and the total order constraint for decision nodes. The optimization process is recursive for each δ. It iterates over the decision nodes from the last one to the first one.

Based on previous work of Zhang and Poole [ZP92, ZP94] an algorithm was proposed in [Zha98] which divides an regular ID into two independent partitions, the parents of each decision node and the rest of the nodes. The basic techniques of the algorithm are the stepwise decomposition of IDs and the standard Bayesian inference methods. After the decomposition an optimal policy for each decision can be evaluated. Also based on inference of BNs, Xiang and Ye [XY01] proposed a similar method which is as efficient as others mentioned above, but simpler.

Besides the approaches using BNs there are other methods such as Shenoy's valuation-based networks [She92]. More detailed descriptions of these algorithms can be found in [RN03, Cro04].

An Abstract Algorithm

According to [RN03] algorithms for evaluating an solution of an ID can be abstractly described by the following steps:

1. Instantiate all available chance nodes $par(D)$ for current decision node D.
2. For each instantiation of decision node $D = d$:
   (a) Update the probabilities for $par(U)$ by using any standard probabilistic inference algorithm, where U is a utility node associated with D.
   (b) Calculate the expected utility $EU(d)$.
3. Return $D = d'$ if $EU(d')$ is the highest utility of D, otherwise $D = d$.

Normally the evaluation algorithm for finding a solution needs to solve the whole model by exploring all the possible combinations of decision nodes and chance nodes. For some situations it may be not necessary for evaluating the whole model. For example, only a part of some significant information might already be enough to identify the best choice for the next decision step. In this case the computation will be much faster than in the case when all policies are evaluated. For instance the tool *GeNIe*[30] provides algorithms for both policy evaluation and the best choice calculation.

[30] http://genie.sis.pitt.edu/

Chapter 3

A Framework for DSSs

Since the early 1970s, works centered on DSSs[1] technology and applications have developed significantly. Not only individuals but also workgroups, teams and especially virtual organizations make use of DSSs to solve decision problems, which are appearing in almost any real life applications.

This chapter gives an overview of definitions and categorizations about DSSs first. Then some significant system requirements and challenges will be pointed out. Section 3.2 will propose an abstract framework for DSSs adapted to probabilistic uncertain knowledge. The framework builds upon a number of theoretical and technical foundations (part of them already introduced in Chapter 2), to deal with these generic requirements and challenges of DSSs. A concrete implementation of this framework, both at the theoretical level and at the technical level, will be investigated in detail in the following chapters. Section 3.3 summarizes some basis features of the proposed framework, and sharpen views towards the deployment of the framework for building real applications of DSSs.

3.1 Definitions, Categorizations and Challenges

Before we introduce some definitions of DSSs, it is meaningful to give a clear interpretation of *decision making*. An abstract view of decision making is to make a choice among several alternatives. A more sophisticated view is that it is a process of selecting a course of action among alternative choices. Holtzman pointed out that "making a decision means designing and committing to a strategy to irrevocably allocate valuable resources" [Hol89]. One interesting point in this definition

[1] In the literature knowledge based systems are sometimes considered as a synonym for DSSs because they formalize domain knowledge to enable decision making based on mechanized reasoning. But we distinguish between KBSs and DSSs by the fact that DSS can be knowledge driven, but not necessarily so.

is that an actual allocation of resources can be understood under taking action, which is obviously not included in the decision making process. Another interesting point is the word “irrevocable”, which indicates the coherent importance by making any decision[2]. In fact decision making is a process rather than a single activity.

3.1.1 Definitions of DSSs

There are many definitions of DSSs, but none is universally accepted, because of the different context information according to the development, usage, and intention and so on. Klein and Methlie defined DSSs as follows:

> A computer program that provides information in a given domain of application by means of analytical decision models and access to databases, in order to support a decision maker in making decisions effectively in complex and ill-structured (non-programmable) tasks [KM95].

Druzdzel and Flynn define DSSs in [DF03] as “interactive computer based systems that aid users in judgment and choice activities”. Power pointed out that “the term DSS remains a useful and inclusive term for many types of information systems supporting decision making” [Pow97]. In a broader view, DSSs can be abstractly described as any methodology[3] to support decision making. In this sense, it is not necessary and also impossible to give a unique definition of DSSs. In this work DSSs are simply considered as interactive computer-based systems that support the decision making process.

In spite of no common accepted definition of DSSs we can summarize some basic elements of decision problems which must be taken into account when designing DSSs:

- *Decision maker* is an agent (or a set of agents) which can make a decision for a given decision problem. An agent can be a person, a machine or a software entity. In any decision making process the identification of decision makers is always the first important step, because all other elements related to the process are dependent on it. There may be many other agents affected by the decision and each of them has its special role, for example domain experts, knowledge engineers and so on.

[2] Some actions can only be recovered without regard to the cost of recovery. In contrast some actions can not be recovered physically, for example a time loss.

[3] It means that DSSs must not be necessarily computer based.

- *Decision alternatives and outcomes* are other essential elements of a decision. Decision alternatives are available decision options at the time of the decision. When a decision maker chooses an alternative, then he is subject to a corresponding outcome. Decision outcomes are not all equally attractive, therefore a decision maker needs to evaluate them based on their desirability. This implies a measure of preferences over them. How to do that was already shown in Section 2.4.1.

  In a formal system decisions can be modeled with variables, and the alternatives for each decision can be specified with a domain of values of the decision variable. For example in an ID there are decision nodes with specified domain values.

- A *Decision context* consists of a decision domain and of a decision situation. The former concentrates on the conceptual level, whereas the latter addresses the individual level.

  – A *Decision domain* needs to be identified for each decision problem due to the fact that all decision problems are domain specific. One important requirement for DSSs is to provide a generic domain model (or many models) related to each decision problem. These models should be abstract, generic enough for achieving the commonness and reusability.

  – A *Decision situation* needs to be identified for each decision problem due to the fact that all decision problems are situation specific. It means that decision problems vary over situations. Different decision makers differ themselves from situation to situation. Even for a same decision problem, different decision makers can decide for different alternatives.

3.1.2 Categories of DSSs

Based on different viewpoints many classifications of DSSs were proposed in the last decade [Pow03]. One approach to categorize DSSs is based on different interactive behavior between users and systems to support decision making. They are either passive or active [CJW98].

- A *passive* DSS is a system that aids to support decision making by simplifying and reducing non-structured problems to well-defined tasks that can be predefined in a system without any ambiguity. Most of traditional DSSs are passive and not adequate for real and complex applications.

- An *active* DSS is a system that is able to respond to changes or exceptions and is able to exhibit goal-directed behavior by taking the initiative, in order to solve decision problems.

Another approach for classifying DSSs is according to the dominant component in the system. In [Pow02] there are five generic types of DSSs identified:

- A *communications-driven* DSS is a system that supports decision making with the emphasis on communications and collaboration. It aims to cooperative and collaborative decision making based on one or more groupware.

- A *data-driven* DSS is a system that support decision making with the emphasis on accessing to and manipulating a time series of internal data or external data. For example GIS (*Geographic Information Systems*) are data-driven.

- A *document-driven* DSS is a system that supports decision making with the emphasis on retrieval and management of unstructured documents in a digital format.

- A *model-driven* DSS is a system that supports decision making with the emphasis on accessing to and manipulating decision models that are normally constructed by statistical, financial, OR (*Operation Research*) or simulation methods.

- A *knowledge-driven* DSS is a system that supports decision making with the help of a knowledge base. Normally this kind of DSS combines KBS and other methodologies for decision making. The framework that will be proposed in next section, for building DSSs combines the knowledge-driven aspect and others, for example agent-based aspect, service-oriented aspect and so on (see Section 3.3).

3.1.3 Challenges of DSSs

A sound DSS must be able to address the following important *challenges*.

Uncertainty As discussed in Section 2.2 uncertainty is one of the most daunting challenges in an open and dynamic environment for decision making. Not only the imperfection of information, but also the uncertain nature of correlations between decisions and outcomes, causes uncertainty. Therefore, a good DSS must be able to work under uncertainty.

Adaptivity Another daunting challenge is the perpetual change in the environment. Therefore, DSSs have to scale up and adapt to changing knowledge, workflow, and operational setting [DL05]. Particularly, an adaptive DSS should be flexible enough so that a decision making process can be quickly reconstructed or modified without high cost. From the perspective of system engineering, all components of the system should be loosely coupled, in order to increase their reusability.

Knowledge Management KM (*Knowledge Management*) is a discipline concerned with the representation, processing, distribution and improving of knowledge by humans, machines, organizations and societies. KM in DSSs appears to be more and more important because decision making is a knowledge intensive activity with knowledge throughout the whole decision making process: problem identification, data (or evidence) gathering, diagnosis or predication and so on. The more proficient decision makers are in KM, the more competitive they are within the global knowledge society [Hol01].

Due to the heterogeneity of information resources the effectiveness and efficiency of knowledge management can only be ensured when it relies on the establishment of a common and formal language. The difficulty here is to select a formal representation language which makes a tradeoff between expressiveness and tractability, because the more expressive a formal language is, the less tractable it is, and vice-versa.

- *Expressiveness* is a schema-level or conceptual level issue for modeling decision processes, models and other relevant knowledge in a better way. The background knowledge behind decision making problems in certain application domain must be expressible.
- *Tractability* is a data-level or individual level issue for providing better data exchange, query and integration.

Collaboration To enable decision making to be efficient DSSs must offer a platform for collaboration with teamwork of all participating agents in the decision process. A decision maker needs to collaborate with these agents in getting the knowledge they need and solving decision problems they have, with careful coordination, cooperation, negotiation and even synchronization of activities. From this perspective, the collaboration must be designed into systems from the start and cannot be patched in later [Gro96].

Intelligence One of the decisive factors to estimate the support capabilities of a DSS are its intelligent behaviors. Such intelligences are embedded in the whole decision making process and in all of the system components, for example knowledge management, algorithms, reasoning and so on.

Explanatory The explanatory power of a DSS refers to its ability of explaining its action. Two characteristics are related to this challenge: *transparency* and *flexibility* [Nak06].

- Transparency refers to the ability of DSSs that the decision maker or other users are allowed to have an insight into the underlying mechanism for decision support. A black box for such mechanisms is not desired.
- Flexibility refers to the viewpoint of the UI (*User Interface*) of DSSs, because DSSs are highly interactive by the fact that DSSs do not replace humans but rather support them by augmenting their limited computational and cognitive capability. Therefore, a *user-friendly* and flexible user interface is very important. UIs are not rigid, but open and flexible for a wide variety of end-user interactions according to their divergent demands, for instance interaction via dialogue, argumentation and so on [DF03].

3.2 A Framework for DSSs

The challenges as discussed above shape up our view towards the development of an advanced and abstract framework for DSSs. It must be able to address these challenges in an open and dynamic decision environment, particularly with emphasis on uncertain knowledge. This *framework* can be characterized by different views: a pillar-based view, a layered view, a decision theoretical view and a process view.

3.2.1 The Pillar-based View

As illustrated in Figure 3.1, a DSS rests on a 6-pillar *framework* and each pillar serves as a building block of the system. We can group all pillars into two classes — either theoretical or technical foundations — in the left-to-right sequence. The first four pillars are the theoretical foundations for decision making.

- *Ontologies* can be used to address the challenge of KM in a DSS. As explained in Section 2.1 ontologies as a formal methodology can facilitate knowledge representation, acquisition, sharing and exchange. Due to balanced expressiveness and tractability of OWL, we use it as the underlying KR language for the system and develop ontology-based tools (which are at least compatible with OWL) for KM in the system.

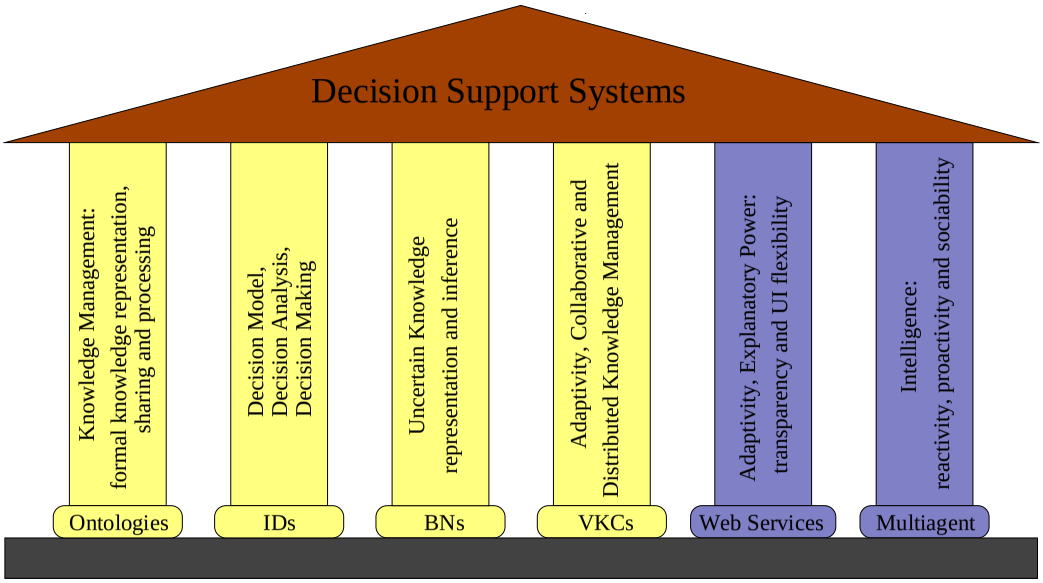

Figure 3.1: The proposed framework of DSSs with all theoretical and technical building blocks.

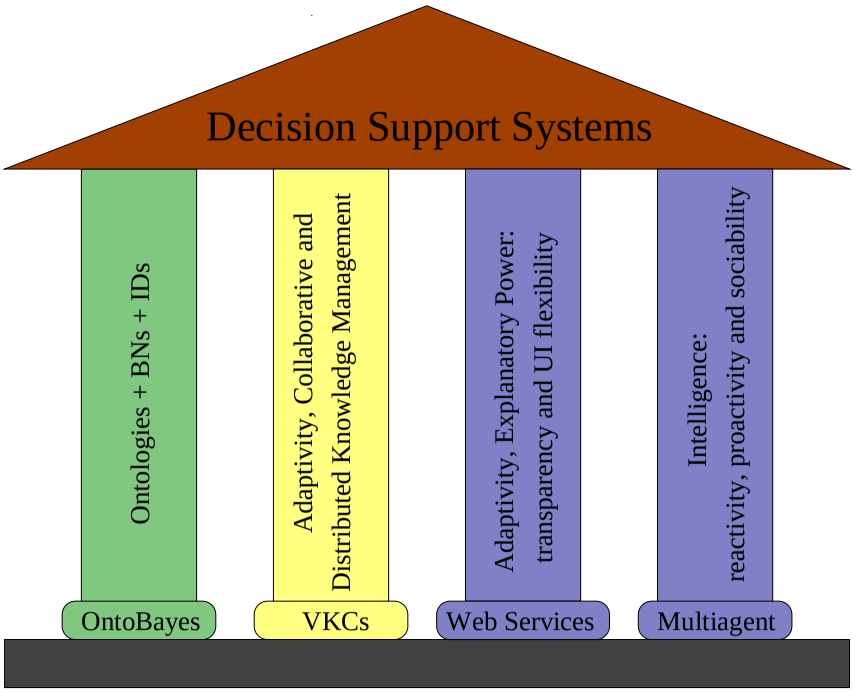

Figure 3.2: The proposed framework of DSSs with an integrated approach — OntoBayes.

- BNs are the underlying method for addressing the challenge of uncertainty in a DSS. They can represent uncertain knowledge in a graphical way and provide inference power in probabilistic reasoning.

- IDs are used as an underlying method for representing decision models, analyzing the models for supporting decision making.

- VKCs (*Virtual Knowledge Communities*) are used to address the challenges of collaboration and adaptivity of a DSS. The concept of a VKC enables us to model corporate knowledge as the amount of knowledge provided by individual agents [MC04]. Additionally, VKCs can enhance KM in a DSS in a distributed way. A detailed introduction to VKCs and the investigation of integrating VKCs into DSSs will be described in Chapter 5.

The last two pillars are considered as the most important technical foundations to implement the system. A simple explanation of these technologies with regard to a DSS will be given as follows. More detailed descriptions of the design of a DSS based on them will be given later in Chapter 6.

- The *Multiagent* paradigm is one of the selected overall technical foundations in DSSs. It is used to address the challenge of system intelligence. This introduces *intelligent agents* as a powerful metaphor in the field of DSSs. These agents link certain decision problem types and the decision environment in which they operate. Intelligent agents are defined as agents capable of autonomous action in situated environment in order to meet their design objectives [Woo99]. They possess the following basic characteristics: *reactivity*, *proactivity* and *sociability*. These basic characteristics make them suitable as software entities for the delegation of diverse decision making tasks and for collaborative work with each other.

- *Web Services* as another selected overall technical foundation in DSSs are used to address the challenge of system adaptivity, explanatory power and UI flexibility. We will design the system architecture in which all components are built as web services so that users or system agents integrate them and perform agent tasks helping users according to various knowledge sources. The whole system will be developed as a SOA (*Service Oriented Architecture*), which provides the system with the following features: loose coupling, implementation neutrality, flexible configurability, long lifetime and granularity [SH05].

In this work we propose an ontology driven uncertain model, OntoBayes, which integrates BNs and IDs into ontologies, in order to preserve all advantages of them. Based on this integration, a DSS now rests on a 4-pillared framework, as

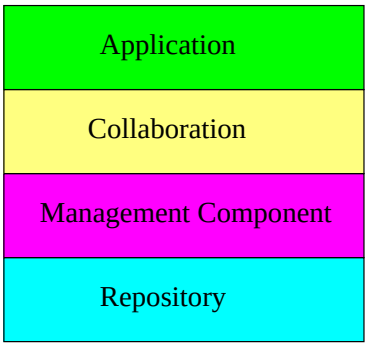

Figure 3.3: The framework of DSSs with a layered view.

illustrated in Figure 3.2. In the next chapter we will demonstrate how to integrate them into OntoBayes based on OWL and how to use OntoBayes for facilitating decision support.

3.2.2 The Layered View

In addition to the pillar-based view, the proposed framework for DSSs has a layered view as shown in Figure 3.3. This *framework* consists of four generic layers: a repository layer, a management component layer, a collaboration layer and an application layer. Each layer has its own functionalities and can communicate with the layers that are directly adjacent to it. In a bottom up approach the hierarchical layers construct a system from the back end to the application front end.

- The *repository layer* is the lowest layer consisting of different kinds of repositories. A repository is a place to store and maintain data; it can be centralized or distributed over a network. Traditionally, the lowest layer in a system is the database layer which has a similar functionality. Nowadays the trend is to build repositories within the system rather than to use simple databases because more and more information systems are not simply data-based, but heterogeneous and hybrid. They cover different types of information resources such as knowledge, models and services etc. It is impossible to store them in a unique database, but it is easy to classify, store, maintain and access them through different kinds of repositories.

- The *management component layer* is an important and complicated layer of the framework. It is the backbone of the system and sustains the running system. There are lots of components required to provide different operations and functionalities to the system. From the point of view of a

service oriented system these components must be loosely coupled and independently managed, in order to insure a flexible configurability, lasting lifetime and granularity of the system. Each management component provides its own services that can be used within the system or be published to construct a business process for instance. In this layer one finds at least the repository management which is responsible for accessing, storing and maintaining different repositories in the repository layer.

- The *collaboration layer* is a virtual place where agents can meet together and collaborate with each other, in order to facilitate corporate knowledge acquisition for supporting decision making. The backbone of this layer are VKCs.

- The *application layer* is the front end of the system. Users or agents will discover and select services provided by the system to build their own applications. Applications can be used either within the constructed new system to design different management tools or for third parties which are the service consumers located outside the system, e.g. a knowledge worker from a partner company who wants to access the knowledge repository.

An implementation based on this layered framework will be discussed and demonstrated in Chapter 6.

3.2.3 The Decision Theoretical View

In order to better understand the abstract framework of DSSs, we will discuss it from a decision theoretical viewpoint. This viewpoint has three different perspectives: *descriptive*, *normative* and *prescriptive*[4] [BRT88].

- The *normative* perspective focuses on how rational agents (humans or software entities) **ought** to make optimal decisions by using axioms. The main disciplinary background of this perspective are statistics, mathematics and economics, for example, expected utility etc..

- The *descriptive* perspective gives emphasis to the **actual** way that ordinary agents are observed to make (rational or non-rational) decisions independently of the theoretical axioms, because the cognitive capability of humans is limited. The main disciplinary background of this perspective are

[4]There are some discussion about the different perspectives. While some researchers considered the normative and prescriptive approaches as the same, other researchers pointed out there is a fourth approach: the constructive approach, which refers to build rationality models for decision makers from their answers to preference-related questions [BMP+06].

Perspective	Characteristics	Ways to get decision models
descriptive (actually do or done)	exogenous rationality, generic and context free	observations and empirical findings
normative (ought)	exogenous rationality, generic and context free	postulate based on a priori established norms
prescriptive (could)	endogenous rationality, specific and situation aware	unveil the system of values of decision makers

Table 3.1: Differences between perspectives of decision making.

psychology and behavioral sciences [Dil98]. For example, the qualitative information of BNs is descriptive.

- The *prescriptive* perspective emphasize how agents **could** (should and can) make decisions reflecting the practical domain, i.e. that decisions are coherent with the decision situation. It selectively fuses the descriptive and normative perspectives by exploiting the logical consequences and the empirical findings, respectively, but behaves more advantageously with some systematic reflection. Generally this approach aims to develop techniques and aids for supporting and improving human decision making [Bro89, Sar94]. The main disciplinary background of this perspective are operation research and management science.

According to the descriptions above and [BMP+06] we can summarize the main differences between these approaches in Table 3.1.

The *framework* proposed in this section fuses all of these three perspectives to provide high performance DSSs. Two important building blocks, BNs and IDs are both descriptive and normative. Another one, ontologies, which will be used to facilitate knowledge management in a DSS, is more descriptive, because most decision relevant knowledge is descriptive, for example, decision models and processes. In the collaboration layer we can utilize VKCs to gather context aware and decision situational information, in order to give agents more practical decision support with a prescriptive approach. Therefore, our framework has a *hybrid* decision making approach.

3.2.4 The Process View

Generally a DSS can provide two levels of decision support, the low and high levels:

- The *low Level* support regards a DSS more as a query system which can provide information according to the queries. Decision makers deploying this

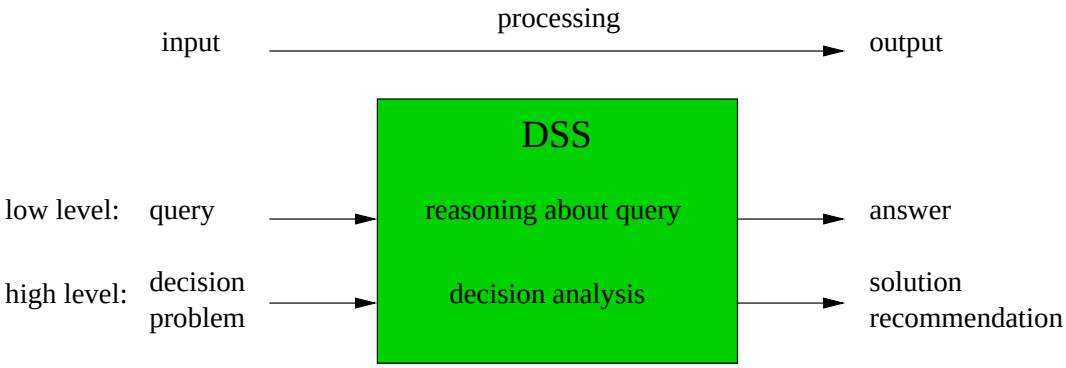

Figure 3.4: Two levels of decision support with an input-output view.

kind of support normally are dealing with some simple decision problems that can be solved directly or with minimal analysis efforts by themselves.

In this level ontologies play a key role, because most queries are for knowledge acquisition and can be easily answered with ontological reasoning. Due to the uncertainty factor, it is also possible that a decision maker wants to know some probabilistic information. The probability queries in a DSS can be answered by using BNs, but they still belong to ontological queries because of the integration of BNs and ontology with OntoBayes.

- The *high level* support regards a DSS more like a decision analytic system which can recommend solutions according to the decision problems via analyzing and forecasting of the current and future environment. Due to their limited cognitive and computational capability, human decision makers are not able to solve complex decision problems (under limited time), therefore they need such a high level support with decision analysis.

  In this level normative decision theory (consisting of IDs, probability theory and utility theory) plays a key role because it is the underlying decision analytic methodology in a DSS with probabilistic uncertain knowledge.

In Figure 3.4 we illustrate these two levels with an input-output view. The box between the input and output refers to the processing in the DSS. The low level support is so simple and intuitive that it does not need to be explained any more. On the contrary the high level support is more complicated because it includes a dynamical decision analysis process. Inspired by the decision analysis cycle [Hol89] and Simon's model of the decision process [Sim60] we split the *decision analysis process* into the following phases corresponding to Figure 3.5:

1. *Basis development* is the first phase of the process. Broadly speaking, the term *basis* refers to all basic information related to a decision problem. The

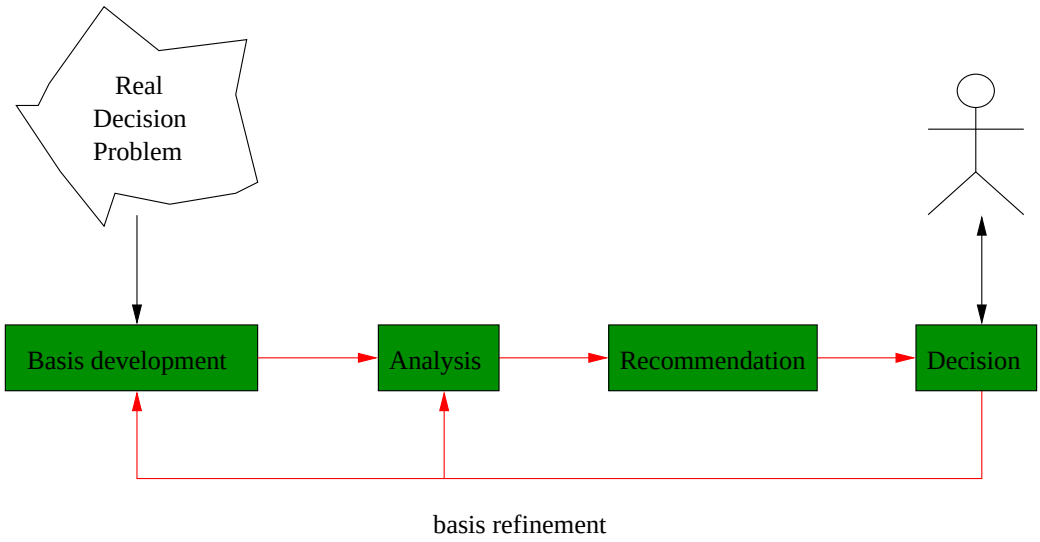

Figure 3.5: A decision analysis process.

following steps are required for such information gathering:

(a) Identify the context of the decision problem, i.e. the decision domain and situation (as described in Page 43).

(b) Identify decision makers and other participants in the process (e.g. domain expert and knowledge engineer).

(c) Identify or develop the decision model. In DSSs a decision model is a formal model of a decision problem in a comprehensive form, e.g. models represented in IDs and encoded in OWL. In the case that the system can identify a formal decision model for a given decision problem in its model base, the analysis process goes directly to the next step, otherwise it is required to develop a formal decision model with the help of domain experts and knowledge engineers.

(d) Identify the decision analysis method. When the system identifies which formal model it can use for solving the problem, it can identify the corresponding analysis method, i.e. which reasoning and evaluation method can be used for analyzing this model. Parsing the ontology of the model will be involved in this step, in order to identify the formal model based on predefined upper ontologies.

2. *Analysis*, the second phase in the decision analysis cycle, consists of the following two steps:

(a) *Deterministic analysis* aims to reduce the size of the original model by eliminating unimportant variables or identify more evidence via similarity analysis. In this step more domain-specific knowledge to determine the ranges on some variables is required. It is here that we take explicit advantage of ontologies via incorporating entity type into BNs and IDs.

(b) *Probabilistic analysis* consists of (1) probabilistic and risk-attitude assessment, and (2) policy evaluation. The first task depends strongly on the decision context (the domain and the situation) of the specific decision maker's problem. The second task is the central task in the analysis phase where we can take advantages of utilizing maximum expected utility.

3. *Recommendation*, the third phase in the decision analysis cycle, gives a report of the analysis results to the decision maker. Such a report includes a recommended solution or a ranking of alternatives for the decision maker, and when necessary, with explanations.

4. *Decision phase* is the phase where the decision maker makes a choice not only based on the recommendation of the system, but also taking the feasibility of the recommendation into account. The final decision will be returned to the system for basis appraisal.

The steps as described above are interrelated and can be repeated as needed throughout the process.

3.3 Features of the Framework

After the detailed description of the proposed framework above we can summarize some basic *features* as follows:

Ontology-driven From the integrated view in Figure 3.2 we can see that the framework is ontology-driven, because after the integration with OntoBayes, the whole KM in the system is ontology-based. VKCs as distributed KM are also ontology-driven [YC06b]. The decisive point of KM is its knowledge representation, because the development of other components of KM depends on the selected methodology of KR. Based on the proposed framework, all knowledge (include uncertain knowledge) of DSSs makes use of OWL as the underlying KR language. Therefore all knowledge intensive activities must be ontological.

Agent-based One of the selected basic technical foundations is MAS (*Multi Agent Systems*) (see Figure 3.1). It means that the framework will be implemented under the multiagent paradigm; therefore it is agent-based.

Service-oriented Another technical foundation is SOA. It means that the framework will be implemented under Web Services paradigm, therefore it is service-oriented.

Uncertainty adaptive Introducing BNs into the framework allow DSSs to deal with probabilistic uncertainty in a mathematical sound way, therefore it is adapted to uncertainty.

Decision-analytic Incorporating BNs and IDs into the framework make the utilization of the principles of decision theory, probability theory, and decision analysis to decision problems possible. Systems based on these principles are decision analytic [DF03].

In principle the framework is generic and flexible enough to be used for a variety of design decision problems. The basic features as mentioned above shape up views towards the deployment of the framework for building real applications of DSSs, for example the application domain of catastrophe insurance (see Section 6.3).

Chapter 4

The OntoBayes Model

In the last chapter we introduced some important challenges of DSSs and proposed an abstract framework to address them. The building blocks of the framework based on four pillars (as theoretical and technical foundations) were illustrated in Figure 3.2. Among these pillars, the one called OntoBayes is the most important and considered as a core model of the proposed framework. *OntoBayes* is an ontology-driven uncertainty model, which integrates BNs and IDs into ontologies for preserving the advantages of them. OntoBayes is devoted to address the following challenges of DSSs: uncertainty, knowledge management (with emphasis on knowledge representation), specification and development of formal decision models for decision analysis. In this chapter, the OntoBayes model will be introduced and completed based on the previous works [YC05, YC06a, YC06b].

The motivation of the research effort about OntoBayes will be given in Section 4.1, as well as a simple overview. Afterward integrating BNs into OWL will be investigated in Section 4.2. This section is the main part of this chapter, because it solves the problem of incorporating uncertainty into OntoBayes which is one of the main purposes of this doctoral work. In Section 4.3 we will investigate the integration of IDs and OWL. At last a survey of related works for incorporating uncertainty and ontology will be given in Section 4.4.

4.1 Motivation and Overview of OntoBayes

A key reason for using ontologies in DSSs is that they enable the representation of background knowledge about a domain in a machine understandable form. After the introduction to ontologies in Section 2.1 we knew the main feature of ontologies is that they can excellently represent the organizational structure of large complex domains. But their application is bounded because of their inability to deal with uncertainty [KP98].

In order to allow agents to work with uncertainty, an extension of ontologies, which has the capability of capturing uncertain knowledge about concepts, properties and relations in domains and of supporting reasoning with inaccurate information, is mandatory. Along this direction, researchers have attempted in the past to use different non-probabilistic and probabilistic methodologies for incorporating uncertainty into ontologies. In Section 2.2 it was stated that the uncertainty modeling in this work will follow the probabilistic direction, because "the only satisfactory description of uncertainty is probability" [Lin87]. As discussed in Section 2.3 BNs have the excellent ability to represent uncertain knowledge in a sound mathematical way. But they are very limited because of their inability to represent complex structured domains.

Comparing the main advantage and disadvantage of BNs and ontologies it is obvious that they can complement themselves via a sound combination aiming at taking advantages of both. This is one of the key reasons for proposing OntoBayes — an integrated approach for building the domain knowledge of agents, including certain and uncertain knowledge.

Besides domain knowledge related to decision problems, agents in DSSs still need (formal) decision models to support them when making decision. As described in Section 2.4 the methodology of IDs has the excellent ability to represent decision models in a graphical way and to analyze them with mathematical algorithms. Therefore, to complete the task of decision making support, OntoBayes is extended with the integration of IDs. The reason for selecting IDs is that there are natural links between IDs and BNs as explained in Section 2.4 and 2.3. Users can easily construct decision models by using IDs based on BNs, due to their similar structures.

Overview of OntoBayes

Abstractly the OntoBayes model is designed with two parts: a knowledge part and a decision model part. The former is an integration of certain and uncertain knowledge based on ontologies and BNs respectively, while the latter can describe different decision models based on IDs.

In order to facilitate the use of OntoBayes in DSSs, particularly to facilitate the share and reuse of knowledge and decision models for decision makers, a formal language for representing the knowledge part and the decision model part is necessary and important. We make use of OWL as the underlying KR language for OntoBayes. Therefore the design of OntoBayes must be implemented based on OWL. For that, we need extend OWL with the features of BNs and IDs. In the following sections we will describe these extensions in details.

4.2 The Extension of BNs

The first step for building the OntoBayes model is to extend BNs in OWL. According to Definition 2.9, the extension of BNs follows two perspectives: the qualitative and quantitative perspective. But before investigating the extension according to these perspectives, it is significant to extract the essentials of BNs at a high and general level with an upper ontology. The upper ontology will be considered as the underlying abstract specification for the Bayesian extension.

4.2.1 An Upper Ontology

In Section 2.1.2 it was pointed out that ontologies can be categorized into different levels with regard to their generality. At the highest and most abstract level it is upper ontologies which are used to refer to top-level ontologies in this work.

In Figure 4.1 a simplified view of an upper ontology is illustrated to capture the essentials of a BN. The graphical notions used here are based on the RDF-Triples as shown in Figure 2.2. Ellipses with solid line represent ontological concepts, whereas ellipse with dashed line represent predefined XML schema datatypes. Each arrow with a label indicates the relationship between two concepts. The numbers on the label are the cardinality constraints.

This figure introduces the very general and commonsense concepts and their properties in the Bayesian world. As mentioned above there are two perspectives in the Bayesian world: the qualitative and quantitative perspective. In the figure we make use of two boxes with dashed blue edges and green edges to frame the qualitative and quantitative information in the upper ontology, respectively.

From the qualitative perspective a BN consists of a number of chance nodes[1] As depicted in the figure there is a red arrow with a label `dependsOn` associated with chance nodes. It indicates the only relationship between different chance nodes — the (statistical) dependency. Every chance node in a BN is either conditional or unconditional depending on whether it depends on other chance nodes or not. A chance node is a discrete variable[2] which has a domain of finite and mutually exclusive states. In this work the domain is simplified as the datatype `string`. A chance node is an evidence node when it is instantiated with an observed state.

From the quantitative perspective each chance node of a BN is assigned with at least one corresponding joint probability distribution, but only one distribution is set as `active`. This active distribution will be used as a default distribution when

[1] In place of using the more common term “nodes”, here we makes use of the term “chance nodes”, because it can facilitate to extend BNs to IDs which classify nodes into three types: chance nodes, value nodes and decision nodes.

[2] As mentioned in Chapter 2 we consider that all nodes of a BN in the context of our work have only discrete domain of values, in order to facilitate the Bayesian extension in OWL.

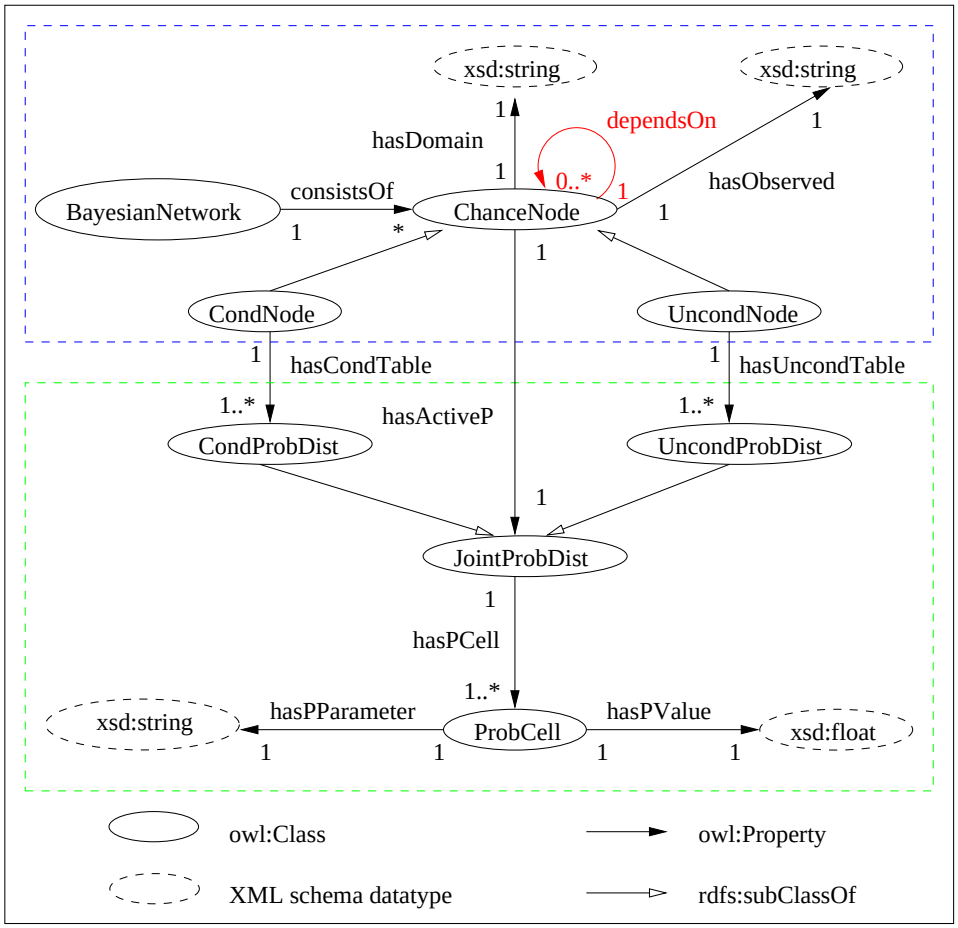

Figure 4.1: The simplified upper ontology for specifying Bayesian Networks in OWL.

executing a reasoning mechanism on the BN. As depicted in Figure 4.1 probability distributions in this work are represented in the form of probability tables. Therefore each such distribution consists of a number of table cells: `ProbCell`. Each cell contains two elements: a parameter for one of all possible instantiation combinations of $X|par(X)$ (see Section 2.3.2) of the given Bayesian variable X and a corresponding probability value. In the case that a chance node has no parent it is simply a possible instantiation of X. Such a parameter and its probability value will be specified as the datatype of `string` and `float` (between 0 and 1) in OWL, respectively. As described in Table 2.2, there are nine instantiation combinations for $P(Premium|RiskCoverage)$. For example the parameter of the instantiation $P(Premium = high|RiskCoverage = high) = 0,2$ is the string "`Premium=high|RiskCoverage=high`" and its corresponding probability value

Concept Name	Annotation in OWL
BayesianNetwork	<owl:Class rdf:ID=‘‘BayesianNetwork’’>
ChanceNode	<owl:Class rdf:ID=‘‘ChanceNode’’>
CondNode	<owl:Class rdf:ID=‘‘CondNode’’>
UncondNode	<owl:Class rdf:ID=‘‘UncondNode’’>
JointProbDist	<owl:Class rdf:ID=‘‘JointProbDist’’>
CondProbDist	<owl:Class rdf:ID=‘‘CondProbDist’’>
UncondProbDist	<owl:Class rdf:ID=‘‘UncondProbDist’’>
ProbCell	<owl:Class rdf:ID=‘‘ProbCell’’>

Table 4.1: Concepts and their annotations used for the Bayesian extension in OWL.

is the float value “0.2”.

The graphical notations simplify the description of BNs, but can not fully express the complete specification in OWL, for example the axiom (or constraint) of the disjointness between the concepts CondNode and UncondNode. In order to describe the upper ontology more detailed and explicitly, a syntactical specification of OWL annotations is required.

Property Name	Annotation in OWL
consistsOf	<owl:ObjectProperty rdf:ID="consistsOf">
hasCondTable	<owl:ObjectProperty rdf:ID="hasCondTable">
hasUncondTable	<owl:ObjectProperty rdf:ID="hasUnCondTable">
dependsOn	<owl:ObjectProperty rdf:ID="dependsOn">
hasActiveP	<owl:ObjectProperty rdf:ID="hasActive">
hasPCell	<owl:ObjectProperty rdf:ID="hasCell">
hasObserved	<owl:DatatypeProperty rdf:ID="hasObserved">
hasDomain	<owl:DatatypeProperty rdf:ID="hasDomain">
hasPParameter	<owl:DatatypeProperty rdf:ID="hasParameter">
hasPValue	<owl:DatatypeProperty rdf:ID="hasPValue">

Table 4.2: Properties and their annotations used for Bayesian extension in OWL.

The OWL annotations for all important concepts and properties modeled in the upper ontology are specified in Table 4.1 and 4.2. The first four concepts in Table 4.1 are used for specifying the qualitative part of a BN, whereas the others for specifying the quantitative part. Table 4.2 contains all object and datatype properties used in the upper ontology. The detailed description of each OWL

class and property of the Bayesian extension will be given in Appendix A.1 and A.2[3].

4.2.2 The Qualitative Extension

The most basic step to construct the qualitative part of a BN is to specify random variables of the BN in an ontology explicitly as well as all dependency relations between them, because of the following reasons:

- The dependency relations are not automatically specified when modeling ontologies.
- The dependency modeling can indicate which random variables are dependent. It means that BNs can be easily extracted from an ontology based on this specification.
- A common dependency modeling method is more applicable than a domain specific method like the one proposed in Ding's work [20] which makes use of BNs to overlap different ontologies over a single domain. In order to reach this goal Ding and Peng design some set-theoretic approach based rules for dependency modeling within the original language. However these rules can not satisfy the requirements of our model, which aims at facilitating probabilistic reasoning to support decision making under uncertainty. We need a more generic dependency modeling than the set-theoretic approach.

In order to solve the problems just mentioned we introduce an additional property element `<owl:ObjectProperty rdf:ID="dependsOn"/>` to markup dependency information in an OWL ontology. Before we give a formal definition of dependency in OntoBayes, we introduce some notations which are influenced by the Object Oriented Programming (OOP) approach. We denote an object property o between a domain class X and a range class Y as $o(X,Y)$. It is considered as an available operation of the subject class X to the object class Y . A datatype property d of the class X will be denoted as $X.d$, where d is considered as a class attribute of X. Now we can define a dependency relation between two properties in an ontology as follow.

Definition 4.1 *A dependency is a pair $X \rightarrow Y$, where each of X and Y is either a datatype property $X.d$ or an object property $o(X,Y)$. It is read as "X depends on Y".*

[3]The entire OWL encoding for the upper ontology of BNs is to be found under the web link `http://iaks-www.ira.uka.de/home/yiyang/DSS/UpperOntologies`.

This definition clearly points out that each Bayesian variable in the OntoBayes model is either an object property or a datatype property. The main reason for using properties to model random variables in an ontology-driven BN is that they enable more precise context modeling than those introduced in [YC05, GHKP04]. Context information can be delivered based on the natural abstract structure behind an object or a datatype property — a RDF-triple which consists of a subject, a predicate and an object (see Section 2.1.3). These context information provide the following features:

- To avoid possible errors when extracting a Bayesian structure from ontologies.

  For instance, in the application domain of catastrophe insurance we model two object properties `LiveIn` and `LocatedAt` associated with the domain class `Person`. These properties have the same range class `Location`. The object property `LiveIn(Person,Location)` indicates the permanent official address while `LocatedAt(Person,Location)` stores the present address. Another class `NaturalDisaster` in the example has a datatype property `OccurrenceProbability`. If we directly define that the datatype property `OccurrenceProbability` depends on the ontology concept `Location`, then insurance companies can not distinguish the occurrence probability of a natural disaster between customer's resident location and actual location. In fact they only care about the occurrence probability at the resident location of a customer, because there are the real properties (e.g. the house) to be insured. To avoid such confusion by modeling we decide to specify dependency between properties, not between classes. Therefore in accord with the notations described above, the correct dependency modeling here for the datatype property `NaturalDisaster.OccurrenceProbability` should be that `NaturalDisaster.OccurrenceProbability` depends on `LiveIn(Person,Location)`.

- To facilitate agents to find executable actions as well as to construct their decision models (i.e. IDs) from the Bayesian knowledge.

  There are two kinds of Bayesian variables in OntoBayes: datatype properties or object properties. Datatype properties will not be taken into account, because they are more like attributes to their associated classes and not executable. On the contrary agents or users can selectively convert object properties into decision nodes according to their executability. For example the object properties `Buy(Customer,Product)` and `Sell(Provider, Product)` as Bayesian variables can be considered as two executable actions for a customer or an insurance agent, respectively, because they enable to

```
<owl:Class rdf:ID="Product.Premium">
     <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="dependsOn"/>
        </owl:onProperty>
        <owl:hasValue rdf:resource="#Product.RiskCoverage"/>
     </owl:Restriction>
     ......
     <rdfs:subClassOf>
          <owl:Class rdf:ID="CondNode"/>
     </rdfs:subClassOf>
</owl:Class>
.......
<owl:Class rdf:ID="Product.RiskCoverage">
     ......
     <rdfs:subClassOf>
          <owl:Class rdf:ID="UncondNode"/>
     </rdfs:subClassOf>
</owl:Class>
```

Figure 4.2: A partial OWL encoding for specifying the Bayesian variables `Product.Premium`, `Product.RiskCoverage` and the dependency between them.

express actions formally and naturally in the form of "action:=subject-verb-object".

Figure 4.2 shows the encoding for the dependency denotation *Product.Premium* $\rightarrow$ *Product.RiskCoverage* in OWL. The partial encoding indicates that the Bayesian variables `Product.Premium` and `Product.RiskCoverage` are subclasses of the class (of the upper ontology) `CondNode` and `UncondNode` respectively. More details are illustrated in Figure 4.8 in Section 4.2.5.

4.2.3 The Quantitative Extension

The qualitative extension of OWL alone is not enough for modeling our ontology-driven BNs. It is required to specify Bayesian variables with associated quantitative information, namely probability tables. For this purpose we define four OWL classes: `UncondProbDist`, `CondProbDist`, `JointProbDist` and `ProbCell`. As described in Section 4.2.1 the first two classes are defined to identify the unconditional probability and the conditional probability respectively. They are subclasses of `JointProbDist` and disjoint. `JointProbDist` has an object property `hasPCell` associating another class `ProbCell`.

Table 4.3 and Figure 4.3 are used to analyze the representation of an unconditional probability table in an ontological way. According to them we know that

ProbCell	hasPParameter(xsd:string)	hasPValue(xsd:float)
cell_1	Product.RiskCoverage=low	0.2
cell_2	Product.RiskCoverage=middle	0.5
cell_3	Product.RiskCoverage=high	0.3

Table 4.3: The unconditional probability table for the Bayesian variable *Product*.*RiskCoverage* (a datatype property).

```
<owl:Class rdf:ID="Product.RiskCoverage">
    <UncondProbDist rdf:ID="table_1">
            <hasPCell>
              <ProbCell rdf:ID="cell_3">
                <hasPParameter rdf:datatype="#string"
                >Product.RiskCoverage=high</hasPParameter>
                <hasPValue rdf:datatype="#float"
                >0.3</hasPValue>
              </ProbCell>
            </hasPCell>
            <hasPCell>
              <ProbCell rdf:ID="cell_2">
                <hasPParameter rdf:datatype="#string"
                >Product.RiskCoverage=middle</hasPParameter>
                <hasPValue rdf:datatype="#float"
                >0.5</hasPValue>
              </ProbCell>
            </hasPCell>
            <hasPCell>
              <ProbCell rdf:ID="cell_3">
                <hasPParameter rdf:datatype="#string"
                >Product.RiskCoverage=low</hasPParameter>
                <hasPValue rdf:datatype="#float"
                >0.2</hasPValue>
              </ProbCell>
            </hasPCell>
    </UncondProbDist>
</owl:Class>
```

Figure 4.3: The partial OWL encoding for the unconditional probability table illustrated in Table 4.3.

ProbCell	hasPParameter(xsd:string)	hasPValue(xsd:float)
cell_1	Product.Premium=low| Product.RiskCoverage=low	0.6
cell_2	Product.Premium=low|Product.RiskCoverage=middle	0.3
cell_3	Product.Premium=low|Product.RiskCoverage=high	0.1
cell_4	Product.Premium=middle|Product.RiskCoverage=low	0.2
cell_5	Product.Premium=middle|Product.RiskCoverage=middle	0.5
cell_6	Product.Premium=middle|Product.RiskCoverage=high	0.3
cell_7	Product.Premium=high|Product.RiskCoverage=low	0.1
cell_8	Product.Premium=high|Product.RiskCoverage=middle	0.2
cell_9	Product.Premium=high|Product.RiskCoverage=high	0.7

Table 4.4: The conditional probability table for $P(Product.Premium|Product.RiskCoverage)$.

the Bayesian variable `Product.RiskCoverage` has an unconditional probability table which is an instance `table_1` of the class `UncondProbDist`. This instance is associated with three instances of the class `ProbCell` via the object property `hasPCell`. They are `cell_1`, `cell_2` and `cell_3`. Each cell consists of a parameter and a probability value.

The conditional probability table can be constructed in a similar way as the unconditional one. Table 4.4 and Figure 4.4 are used to analyze the representation of a conditional probability table in an ontological way. According to them we know that the Bayesian variable `Product.Premium` has a conditional probability table which is an instance `table_2` of the class `CondProbDist`. This instance is associated with nine instances of the class `ProbCell` via the object property `hasPCell`. They are `cell_1`, `cell_2`, ... and `cell_9`. The illustrated encodings in Figure 4.4 is not complete because it contains only two of the nine table cells: $P(Product.Premium = low|Product.RiskCoverage = low) = 0.6$ and $P(Product.Premium = high|Product.RiskCoverage = high) = 0.7$. The other seven can be similarly encoded.

The encodings in Figure 4.4 and in Figure 4.3 are so similar that there is almost no differences between them, because we simplify the OWL specification for parameters of table cells with XML schema datatype `string`. Based on this simplification, the modeling of probabilistic information in an ontology is easier and more efficient (we will discuss some related works in Section 4.4 to show this feature). The only difference here is the class "indicator" `CondProbDist` and `UncondProbDist` that can identify whether a probability table is conditional or unconditional.

For all cells of a conditional table, we need parse their parameters. Therefore, we implement a small parser to analyze the parameter of each table cell, in or-

```
<owl:Class rdf:ID="Premium">
     <CondProbDist rdf:ID="table_2">
          <hasPCell>
            <ProbCell rdf:ID="cell_1">
              <hasPValue rdf:datatype="#float"
              >0.6</hasPValue>
              <hasPParameter rdf:datatype="#string"
              >Product.Premium=low|Product.RiskCoverage=low>
              </hasPParameter>
            </ProbCell>
          </hasPCell>
          ......
          <hasPCell>
            <ProbCell rdf:ID="cell_9">
              <hasPParameter rdf:datatype="#string"
              >Product.Premium=high|Product.RiskCoverage=high
              </hasPParameter>
              <hasPValue rdf:datatype="#float"
              >0.7</hasPValue>
            </ProbCell>
          </hasPCell>
     </CondProbDist>
</owl:Class>
```

Figure 4.4: The partial OWL encoding for the conditional probability table illustrated in Table 4.4.

der to make it suitable for Bayesian reasoning according to Bayes' theorem (see Equation 2.9).

4.2.4 The Graphical Representation

The presentation of the knowledge part of OntoBayes was so far done through some sort of markup language in OWL. There is, in addition, a graphical representation. There are in fact two graphical models in the knowledge part of OntoBayes: an OWL and a Bayesian graph (or network) models. The former is a directed graph which is built on the graph data model of RDF, as illustrated in figure 2.2, and additionally has a markup of dependency relations between properties. This graph model can visualize all possible information of a specified ontology such as the class hierarchy or the dependency relations for instance. But it exhibits so many different relations that it is challenging to visualize any significant overview of such graphs in realistic cases. Therefore we extract the Bayesian graph model from this model, in order to clearly show the dependency relations which are more interest for decision makers. These two models can be also dis-

Figure 4.5: The graphical representation for a generic dependency triple.

tinguished through their nodes. Indeed, while the nodes in the OWL graph model only consist of classes and datatypes, the nodes in the Bayesian graph model are properties.

The underlying structure of any expression for BNs in OntoBayes is also a collection of triples, each consisting of a subject, a predicate and an object, where the predicate is constantly the dependency relation `dependsOn` and the subject and object are either an object properties or a datatype property from the ontological knowledge. Such a triple is called a *dependency triple*. There are two differences from an RDF triple in the graphical representation. The first one is that we omit the predicate as a label of an arc because the predicate here is unique. The second one is that the arrow of arcs is not from subject to object, but from object to subject, because the intuitive meaning of an arrow in a properly constructed network is usually that X has a direct influence on Y [RN03]. We illustrate it in Figure 4.5 as opposed to a generic RDF triple in figure 2.2.

We can simply construct an OWL graph model with the help of RDF and dependency triples. Figure 4.6 shows some concepts and their properties as well as the relations in the domain of insurance and natural disaster. The ellipse is a graphical notation for OWL classes and the rectangle is used for data types. A label on an arrow line refers to a property. The dashed line models the influence relation. For example $Product.Premium \rightarrow Buy(Customer, Product)$ means the premium of a product has influence on a purchase action. In fact an influence relation is an inverse property of a dependency relation.

Using dependency triples we can build a Bayesian graph model by the following simple rules.

- First we extract all dependency triples from an OntoBayes ontology and represent them separately according to figure 4.5.
- Next, all triples will be merged: all nodes with a same identifier are composed into one single node. For example, if there are two triples $A \rightarrow B$ and $B \rightarrow C$, they can be merge into a Bayesian Network with only one node B such as $A \rightarrow B \rightarrow C$.

In figure 4.7 we illustrate the BN extracted from Figure 4.6. The ellipses are Bayesian variables and the dashed lines specify the influence relations between them.

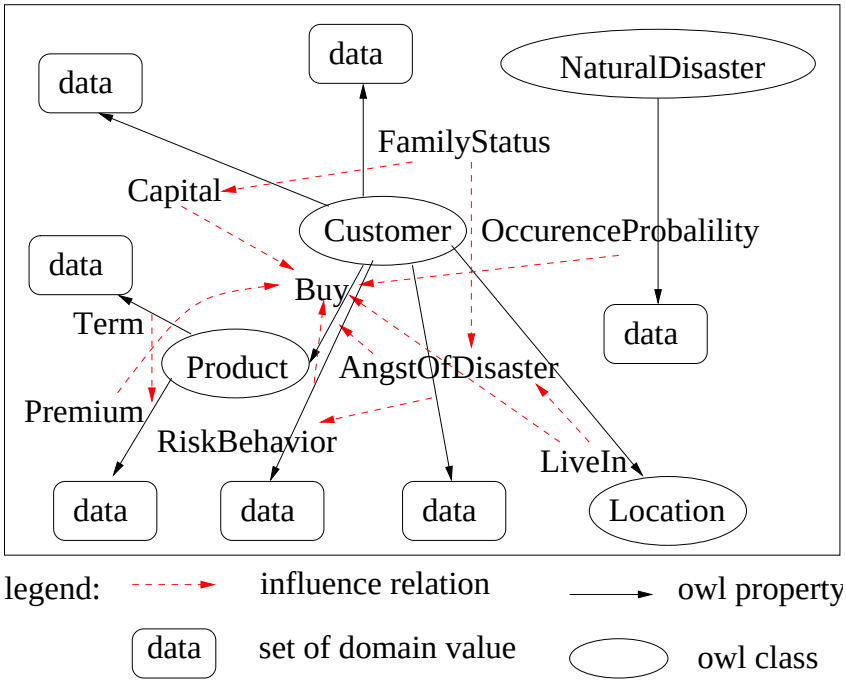

Figure 4.6: The OWL graph model for an insurance ontology.

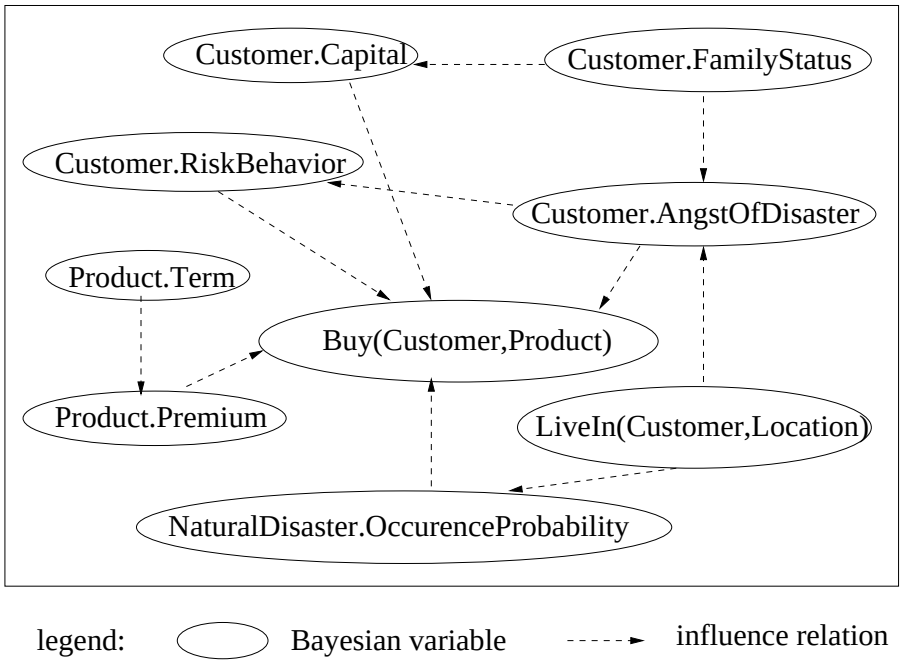

Figure 4.7: The Bayesian graph model extracted from the insurance ontology.

4.2.5 Construction of the Knowledge Part

The upper ontology illustrated in Figure 4.1 can be used to guide the development of domain specific BNs. The underlying OWL file for encoding this upper ontology will be published through an URI for making its access from users (particularly for domain experts and knowledge engineers) possible and easier. Users can import the file to another OWL file in which they can specify the domain specific Bayesian knowledge by using Protégé. After importing it there are many predefined concepts for creating domain specific Bayesian variables and probability tables. As described in the last sections all domain specific Bayesian variables are subclasses of the predefined class `ChanceNode` and all probability tables associated with them are composed with instances of `JointProbDist` and `ProbCell`.

Figure 4.8 depicts how to use this upper ontology to construct domain specific Bayesian knowledge. According to this figure it is clear that the knowledge part of OntoBayes consists of ontological knowledge (the yellow zone) and Bayesian knowledge (the blue zone). They are related to each other. Generally if we want to build Bayesian knowledge in Ontobayes, we need to build first ontological knowledge. Based on the ontological knowledge we can further construct Bayesian knowledge. The Bayesian upper ontology is at the highest level in the whole application, from bottom to top. As shown in Figure 4.8 we want to construct the Bayesian domain knowledge for the application of catastrophe insurance. We can do that directly with the help of the upper ontology, but the Bayesian knowledge could be more sound and factual when we build it upon the given validated ontological knowledge (e.g. from domain experts). According to this idea a Protégé plugin, *OWLOntoBayes*, is implemented for building the Bayesian knowledge in OntoBayes. The plugin allows the user to construct a BN in agreement with the given upper ontology via drag and drop interface [Her07].

4.3 The Extension of IDs

Decision models are comprehensible for agents, only when they provide a semantic understanding of their structure and sound syntactical specification behind the semantics. As mentioned before we make use of IDs as the underlying theory for the decision model part of OntoBayes due to their excellent graphical expressiveness and their understandability. In this section we will demonstrate how to integrate IDs into OntoBayes in an ontological way. But before that, it is meaningful to extract the essentials of IDs at a high and general level with an upper ontology, which will be considered as the underlying abstract specification for the OWL extension.

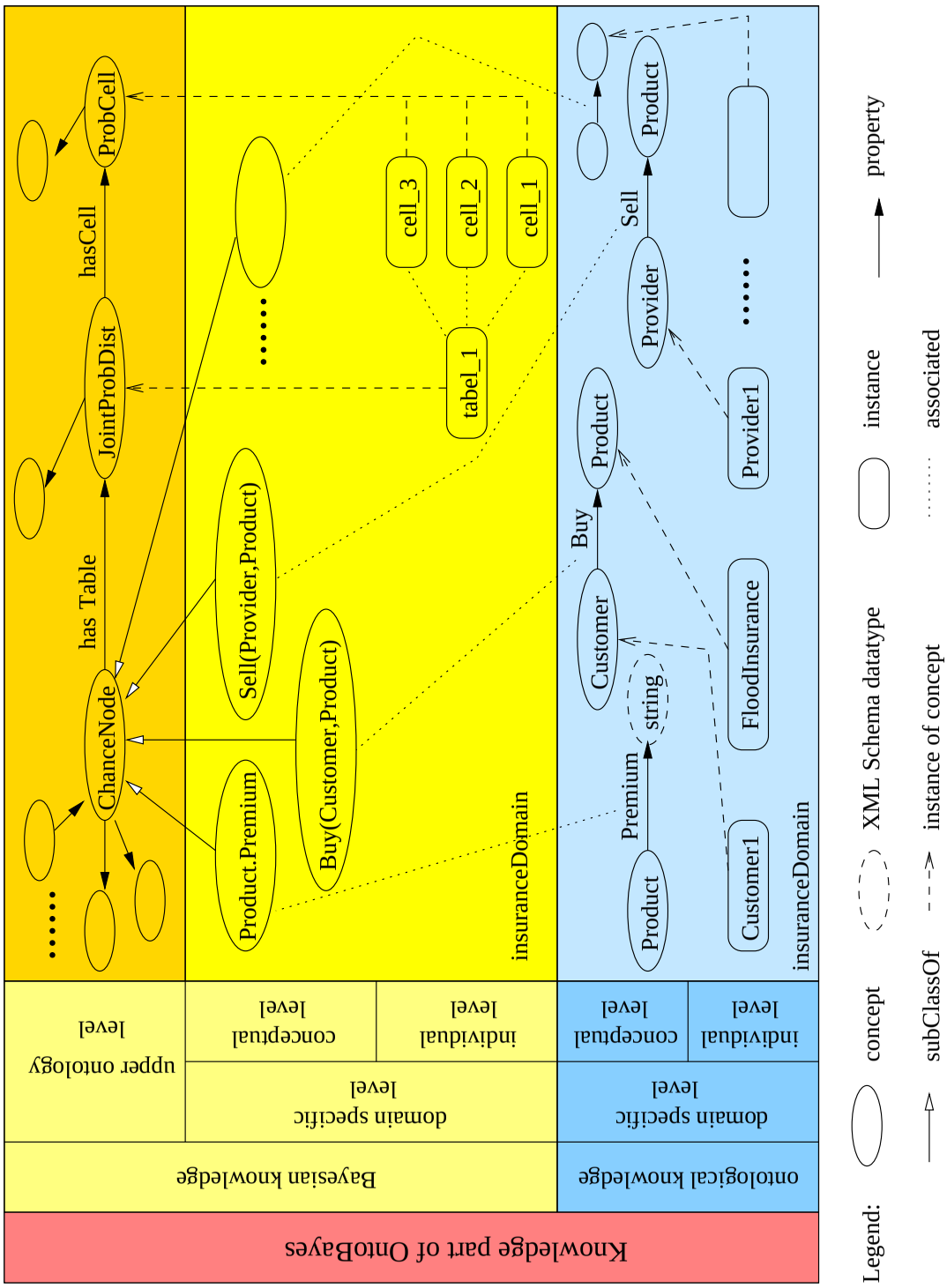

Figure 4.8: Construction of knowledge part in OntoBayes by using an upper ontology of BNs.

4.3.1 An Upper Ontology

In Figure 4.9 a simplified view of an upper ontology is depicted to define an ID. The graphical notations used here are similar to them illustrated in Figure 4.1, but have three types of arcs more. This figure introduces the very general and commonsense concepts and their properties in the world of IDs. According to Definition 2.18, an ID consists of three type of nodes: decision nodes, utility nodes and chance nodes. The relations between them are depicted in the figure as three arc types: informational arcs (the green arrow), conditional arcs (the red arrow) and functional arcs (the yellow arrow). The nodes and arcs between the nodes together build the core of the upper ontology. More details about them were introduced in Section 2.4.

This upper ontology is more complicated that the one for BNs. But there are a lot of similarities between them, due to the fact that in principle an ID can be seen as an extension of a BN with two additional kinds of nodes: decision nodes and utility nodes. From this perspective each chance node in IDs has exactly the same specification as the chance node illustrated in Figure 4.1, particularly the specification for probability tables associated to each chance node. Indeed we take over the quantitative part of Figure 4.1 to specify the same information of chance nodes in IDs. In this section we will focus on the OWL extensions for decision nodes and utility nodes.

As depicted in the upper ontology each decision node in an ID has exactly a set of alternatives that is simply labeled in an XML Schema datatype `string`. The difficult task for completing the specification of decision nodes is to deal with the informational arcs which links other nodes into a decision node. Such informational arcs are colored in green. As depicted in Figure 4.9 there are only two types of nodes having informational arcs: decision nodes and chance nodes, because of the formal definition of IDs. In OntoBayes making a decision in a given decision model equals choosing an alternative for a decision node of the model. The informational arcs related to this decision node imply all information resources that should be observed when making the decision. In the context of chance nodes these nodes will become evidence nodes to this decision node. And in the context of decision nodes these informational arcs can be used to indicate the sequence of all decision nodes in the decision model.

Similar to a chance node, each utility node is associated with a set of tables, not for specifying probability but for specifying utility. Only one utility table is active and will be taken into account when evaluating the ID. Each utility table consists of a number of table cells which can be constructed in an analogous way to the cells of a probability table. The main difference here is the XML Schema datatype `float` will not be restricted between 0 and 1.

The graphical notations simplify the description of IDs, but can not fully ex-

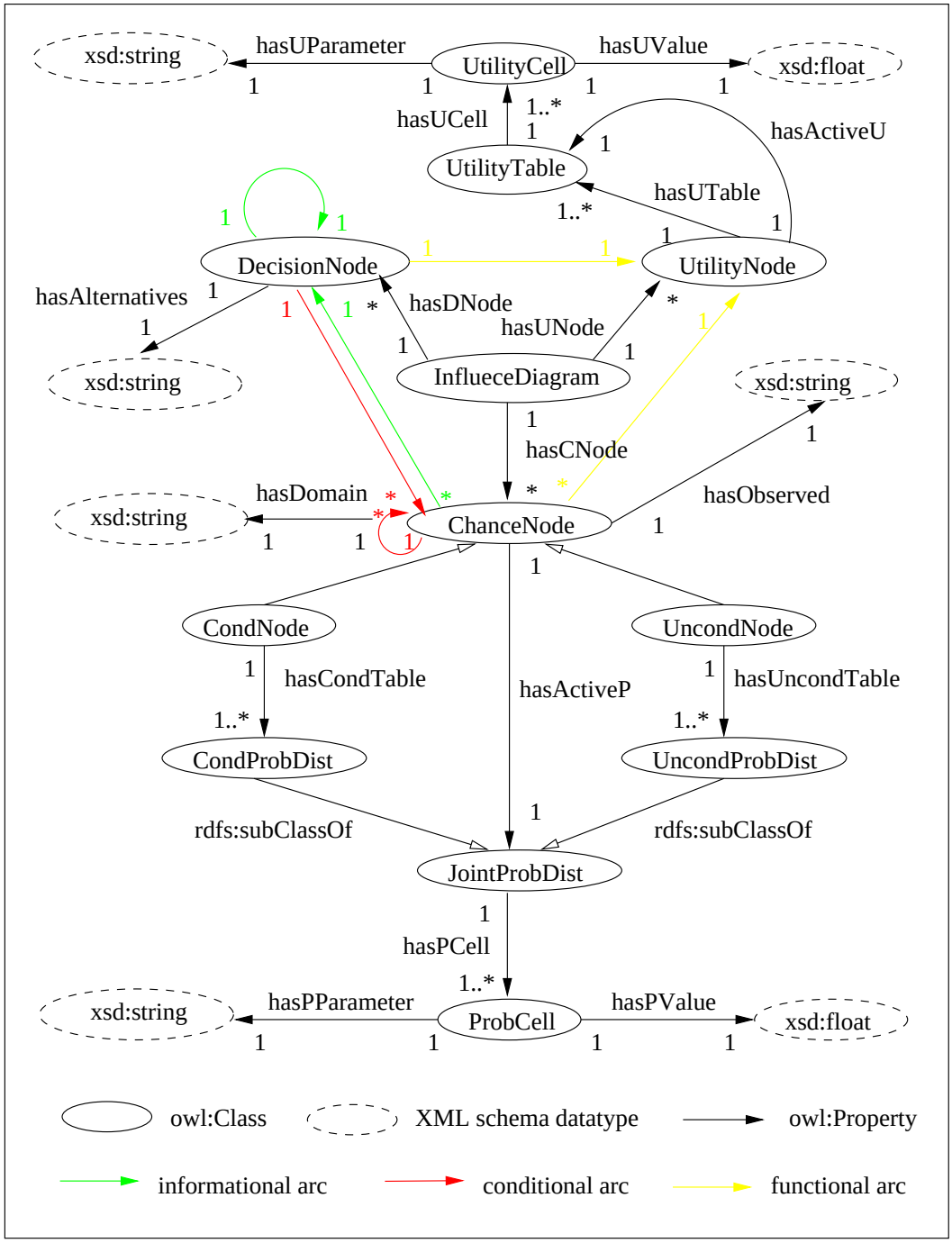

Figure 4.9: The simplified upper ontology for specifying IDs in OWL.

press the complete language specification. In order to describe the upper ontology more detailed and explicitly, syntactical annotations in OWL of IDs are required.

The entire annotations for all important concepts and properties modeled in the upper ontology are specified in Table 4.5 and 4.6, respectively. We need to point out that the three important arc types — informational, conditional and functional — will be specified as object properties `isKnownBy`, `influenceOn` and `attributeBy` in OWL, respectively, where `influenceOn` is an inverse object property of `dependsOn` specified in the Bayesian extension.

The detailed description of each class and property of the OWL extension for IDs will be given in Appendix A.3 and A.4[4].

4.3.2 Using the upper ontology

In this subsection we will investigate how to use the upper ontology (introduced in the last section) for representing IDs in OWL formally, with emphasis on decision nodes and utility nodes. A simplified ID describing a decision model for buying a catastrophe insurance product is illustrated in Figure 4.10. The used graphical notations are almost the same as that used in Section 2.4. The main difference is that arcs in the figure are colored to distinguish the three different types of them. There are two dashed green arcs. They are used to indicate the *no-forgetting* assumption in IDs as mentioned in Remark 2.20.

As specified in the upper ontology each decision node just needs to remember which nodes are informational related to itself directly. It means that the decision node D_i don't need to remember what the decision node D_j ($i \neq j$) observed. But according to Remark 2.19 we knew that there is a directed path throughout all decision nodes. It means that all decision nodes in an ID are sequential. Therefore agents can find out the sequential relation between decision nodes D_i and D_j. Under the known decision sequence information observed before can be transfered to the later decision node. For example the decision node `Decision_n` in the figure has all information observed at the decision nodes `Buy(Customer,House)` and `Buy(Customer,Product)`. In Figure 4.11 we can identify these informational arcs and knowledge associated with them through the OWL annotation `isKnownBy` encoded in the specification of these two decision node.

In Table 4.7 a utility table for the utility node `U2` is specified. The utility values are represented in different integer numbers. From minus to plus they reflect the preferences of all decision choices in the table from low to high. For example, under the condition `Product.Premium=low`, the decision alternative `Buy(Customer,Product)=yes` has the lowest utility -100 and will be not pre-

[4]The entire OWL encoding for the upper ontology of IDs is to be found under the web link `http://iaks-www.ira.uka.de/home/yiyang/DSS/UpperOntologies`.

Concept Name	Annotation in OWL
InfluenceDiagram	<owl:Class rdf:ID=‘‘InfluenceDiagram’’>
DecisionNode	<owl:Class rdf:ID=‘‘DecisionNode’’>
UtilityNode	<owl:Class rdf:ID=‘‘UtilityNode’’>
UtilityTable	<owl:Class rdf:ID=‘‘UtilityTable’’>
UtilityCell	<owl:Class rdf:ID=‘‘UtilityCell’’>
ChanceNode	<owl:Class rdf:ID=‘‘ChanceNode’’>
CondNode	<owl:Class rdf:ID=‘‘CondNode’’>
UncondNode	<owl:Class rdf:ID=‘‘UncondNode’’>
JointProbDist	<owl:Class rdf:ID=‘‘JointProbDist’’>
CondProbDist	<owl:Class rdf:ID=‘‘CondProbDist’’>
UncondProbDist	<owl:Class rdf:ID=‘‘UncondProbDist’’>
ProbCell	<owl:Class rdf:ID=‘‘ProbCell’’>

Table 4.5: Concepts and their annotations used for IDs in OWL.

Property Name	Annotation in OWL
InformationalArc	<owl:ObjectProperty rdf:ID="isKnownBy">
CondiationalArc	<owl:ObjectProperty rdf:ID="influenceOn">
FunctionalArc	<owl:ObjectProperty rdf:ID="attributeOf">
hasDNode	<owl:ObjectProperty rdf:ID="hasDNode">
hasCNode	<owl:ObjectProperty rdf:ID="hasCNode">
hasUNode	<owl:ObjectProperty rdf:ID="hasUNode">
hasUTable	<owl:ObjectProperty rdf:ID="hasUTable">
hasUCell	<owl:ObjectProperty rdf:ID="hasUCell">
hasActiveU	<owl:ObjectProperty rdf:ID="hasActiveU">
hasCondTable	<owl:ObjectProperty rdf:ID="hasCondTable">
hasUncondTable	<owl:ObjectProperty rdf:ID="hasUnCondTable">
hasPCell	<owl:ObjectProperty rdf:ID="hasPCell">
hasActiveP	<owl:ObjectProperty rdf:ID="hasActiveP">
hasUValue	<owl:DatatypeProperty rdf:ID="hasPValue">
hasDomain	<owl:DatatypeProperty rdf:ID="hasDomain">
hasObserved	<owl:DatatypeProperty rdf:ID="hasObserved">
hasUParameter	<owl:DatatypeProperty rdf:ID="hasPParameter">
hasAlternatives	<owl:DatatypeProperty rdf:ID="hasAlternatives">
hasPParameter	<owl:DatatypeProperty rdf:ID="hasPParameter">
hasPValue	<owl:DatatypeProperty rdf:ID="hasPValue">

Table 4.6: Properties and their annotations used for IDs in OWL.

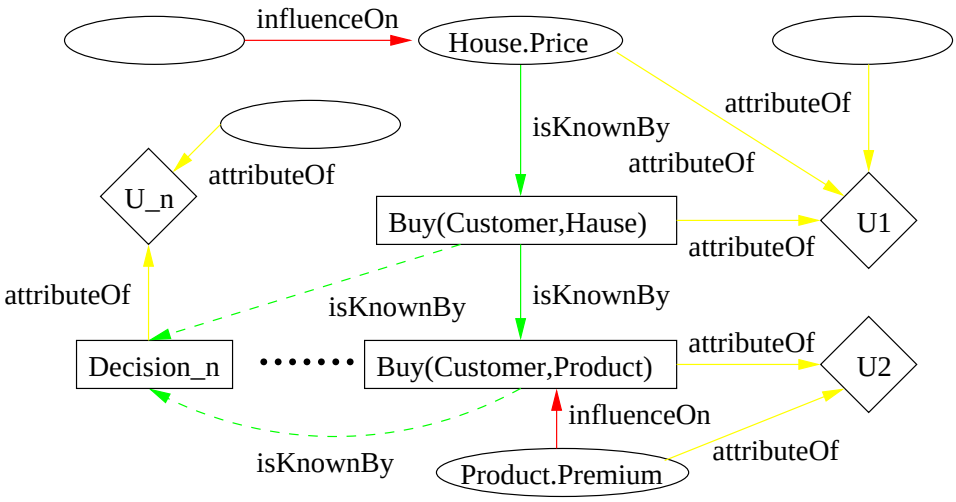

Figure 4.10: A simplified ID used for demonstrating the OWL extension of IDs.

```
<owl:Class rdf:ID="Buy(Customer,Product)">
     ......
     <rdfs:subClassOf>
          <owl:Class rdf:ID="DecisionNode"/>
     </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Buy(Customer,Hause)">
     <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="isKnownBy"/>
        </owl:onProperty>
        <owl:hasValue rdf:resource="#Buy(Customer,Product)"/>
     </owl:Restriction>
     ......
     <rdfs:subClassOf>
          <owl:Class rdf:ID="DecisionNode"/>
     </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Hause.Price">
     <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="isKnownBy"/>
        </owl:onProperty>
        <owl:hasValue rdf:resource="#Buy(Customer,Hause)"/>
     </owl:Restriction>
     ......
     <rdfs:subClassOf>
          <owl:Class rdf:ID="ChanceNode"/>
     </rdfs:subClassOf>
</owl:Class>
......
```

Figure 4.11: The partial OWL encoding for specifying the decision node `Buy(Customer,House)` and `Buy(Customer,Product)` illustrated in Figure 4.10.

UtilityCell	hasUParameter(xsd:string)	hasUValue(xsd:float)
cell_1	Buy(Customer,Product)=yes,Product.Premium=low	-100
cell_2	Buy(Customer,Product)=no,Product.Premium=low	50
cell_3	Buy(Customer,Product)=yes,Product.Premium=middle	40
cell_4	Buy(Customer,Product)=no,Product.Premium=middle	10
cell_5	Buy(Customer,Product)=yes,Product.Premium=high	80
cell_6	Buy(Customer,Product)=no,Product.Premium=high	-80

Table 4.7: The utility table for the utility node `U2` associated with two attributes: the decision node `Buy(Customer,Product)` and the chance node `Product.Premium`.

ferred for decision makers. As illustrated in Figure 4.10, $U2$ has two attributes: the decision node `Buy(Customer,Product)` and the chance node `Product.Premium`. The underlying OWL encoding for this table is shown in Figure 4.12. With the help of the annotation `attributeOf` we can identify all attributes of a utility node. Encoding for utility tables in OWL can be constructed similarly to probability tables. A simple syntax parser is necessary and implemented, in order to decompose parameters of each utility able cell into separated attributes.

4.3.3 Construction of the Decision Model Part

The upper ontology illustrated in Figure 4.9 can be used to guide the development of domain specific decision models (i.e. IDs). The OWL file for encoding this upper ontology will be published through URI for making its access from users (particularly for domain experts and knowledge engineers) possible and easier. Users can import the file to another OWL file in which they will specify a domain specific ID by using Protégé. After importing it there are many predefined concepts for creating domain specific decision nodes, chance nodes with probability tables and utility nodes with utility tables. As shown in Figure 4.11 and 4.12 all domain specific nodes in an ID are subclasses of the predefined classes `ChanceNode`, `DecisionNode` or `UtilityNode`. And each utility table associated with a utility node are composed with an instance of `UtilityTable` and a number of instances of `UtilityCell`.

Figure 4.13 depicts how to construct a decision model based on the knowledge part of OntoBayes by using the upper ontology of IDs. This figure clearly illustrates that the decision model part and the knowledge part of OntoBayes are related to each other. Often a decision model is involved in many different domains. Therefore we need different domain knowledge before we try to build a decision model in OntoBayes. The approach here is similar to the one discussed

```
<owl:Class rdf:ID="Buy(Customer,Product)">
     <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="attributeOf"/>
        </owl:onProperty>
        <owl:hasValue rdf:resource="#U2"/>
     </owl:Restriction>
     ......
     <rdfs:subClassOf>
          <owl:Class rdf:ID="DecisionNode"/>
     </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Product.Premium">
     <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="attributeOf"/>
        </owl:onProperty>
        <owl:hasValue rdf:resource="#U"/>
     </owl:Restriction>
     ......
     <rdfs:subClassOf>
          <owl:Class rdf:ID="ChanceNode"/>
     </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="U2">
    <UtilityTable rdf:ID="table_1">
         <hasUCell>
           <UtilityCell rdf:ID="cell_1">
             <hasUParameter rdf:datatype="#string"
             >Buy(Customer,Product)=yes,Product.Premium=low
             </hasUParameter>
             <hasUValue rdf:datatype="#float"
             >-100</hasUValue>
           </UtilityCell>
         </hasUCell>
         ......
         <hasUCell>
           <UtilityCell rdf:ID="cell_6">
             <hasUParameter rdf:datatype="#string"
             >Buy(Customer,Product)=no,Product.Premium=high
             </hasUParameter>
             <hasUValue rdf:datatype="#float"
             >-80</hasUValue>
           </UtilityCell>
         </hasUCell>
    </UtilityTable>
</owl:Class>
```

Figure 4.12: The partial OWL encoding for the utility table illustrated in Table 4.7.

in Section 4.2.5. Whereas there we make use of ontological knowledge to support the construction of Bayesian knowledge, here we make use of the Bayesian knowledge to support construct decision models. The figure in fact extends Figure 4.8 with the green zone — the decision model part of OntoBayes. In the figure the upper ontology of IDs is at the highest level in the whole application, from bottom to top. We can construct an ID for any application domains directly with the help of the upper ontology, but it will be more convenient[5], efficient and factual when we build the ID upon the given validated Bayesian knowledge. Such Bayesian knowledge can be composed from many different BNs related to different domains. The plugin *OWLOntoBayes* allows users to construct an ID based on the given BNs via a drag and drop interface.

Features of OntoBayes

We can summarize some features of the OntoBayes model as follows:

- OntoBayes is an ontology-driven model. It means that all features of an ontology-based system are remained in OntoBayes, e.g. formal representation, reusability etc..
- OntoBayes is OWL compatible. The description in the last sections showed the formal OWL Annotations for BNs and IDs. OWL is the most widely used language in semantic webs nowadays. Therefore, OntoBayes has good potentials in semantic webs.
- OntoBayes has an intuitive ability of graphical representation. Ontologies, BNs and IDs, all of them can be good represented graphically. Therefore OntoBayes is really easier to be understood and to use.
- OntoBayes is a model for dealing with uncertainty. The integration with BNs and IDs provides the ability of uncertainty modeling in a probabilistic approach.

In the next section we will introduce some related approaches which aim at incorporating uncertainty into ontologies, particularly with the emphasis on OWL.

4.4 Related Works

There has been some attempts to incorporate non-probabilistic and probabilistic methodologies of representing uncertainty into ontologies, such as fuzzy logic and

[5]The natural link between BNs and IDs provides such convenience. Most nodes in an ID are chance nodes which can be directly taken over from BNs, together with the associated probability tables.

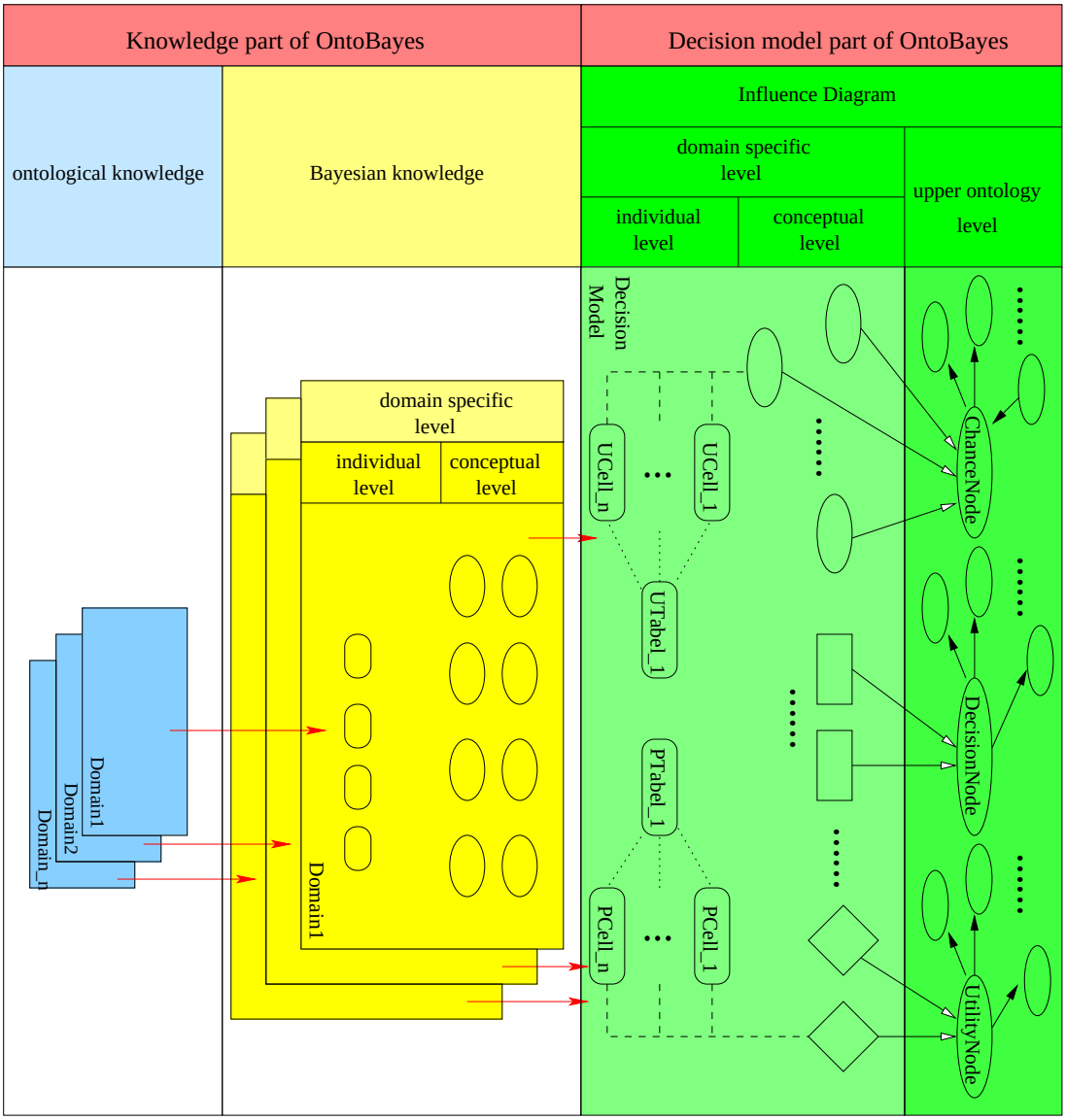

Figure 4.13: Construction of the decision models part based on the knowledge part of OntoBayes by using the upper ontology of IDs.

probability approaches. This section gives a survey of the most relevant works in the last decade, especially those related to OWL. In Section 4.4.1 works based on the probabilistic approach will be discussed. Then the fuzzy approach for extending OWL with uncertainty modeling is discussed in Section 4.4.2.

4.4.1 The Probabilistic Approaches

The probabilistic approaches for the ontology (or other KR) paradigm are those most related to our research effort. Among them, mainly works based on OWL are investigated.

Probabilistic frame-based systems

The approach that seems closest to ours is the old work of Koller and Pfeffer [KP98]. Building on their earlier works, *P-CLASSIC* [KLP97] and *Object-Oriented Bayesian Networks* [KP97], they proposed *probabilistic frame-based systems*, which integrate BNs into a frame-based system to preserve the advantages of both.

BNs are widely used for modeling uncertainty because of their excellent graphical expressiveness and computational power, but they are inadequate for representing large and complex domains [ML96]. On the contrary, frame-based systems have excellent ability to represent large complex domains with their organizational structure, but show limitations due to their inability to deal with uncertainty [KP98]. It is clear that they can complement themselves very well. The complementariness motivates the authors to propose the approach of probabilistic frame-based systems.

From the perspective of its main objective, the OntoBayes model is a similar approach, but it adopts the more recent methodology that consists of proposing ontology-driven uncertain knowledge base. Beside this objective, OntoBayes possesses decision models which aim to support agents making decisions under uncertainty for given problems. From the perspective of the implementation details, OntoBayes makes use of totally different modeling languages, definitions and translation rules both syntactically and semantically.

BayesOWL

Ding proposed in [Din05] a probabilistic ontology approach, *BayesOWL*. It makes use of BNs as the underlying uncertainty theory to extend OWL with probabilistic features, in order to facilitate ontology mapping in the semantic web [DPP04].

The probabilistic modeling in BayesOWL is represented via some additional language markups, which can be simply reflected in an upper ontology as illus-

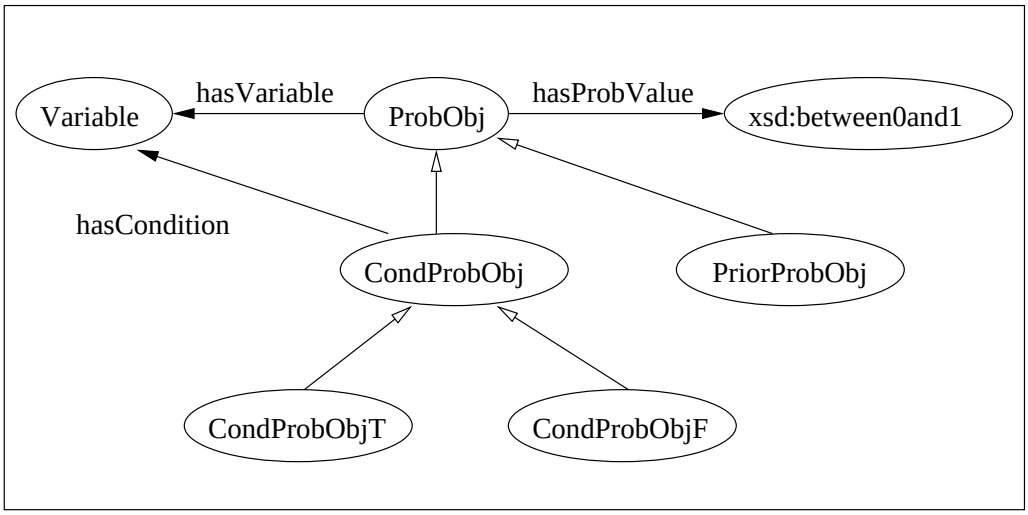

Figure 4.14: The upper ontology for the probabilistic extension in BayesOWL.

A	¬A
0,8	0,2

	B	¬B
A	0,4	0,6
¬A	0,7	0,3

Table 4.8: A simple unconditional probability table `P(A)` and conditional probability table `P(A|B)`.

trated in Figure 4.14. The figure provides an abstract view of the approach for extending Bayesian probability in OWL. Three classes, *PriorProbObj*, *CondProbObjT* and *CondProbObjF*, are used to instantiate probability distributions over a variable. The first class is devoted to specify unconditional probabilities and the last two for expressing conditional probabilities. For example Table 4.8 describes two simplest probability tables for `P(A)` and `P(A|B)`. To encode these tables in OWL, these classes with associated properties are used, e.g. *hasVariable* for queried variables, *hasProbValue* for probabilistic values and *hasCondition* for condition variables of the queried one [DP04].

Indeed, it is unnecessary to give the full joint probability distributions of `P(A)` and `P(A|B)` in BayesOWL, because it only allows two-valued (either true or false) random variables in the Bayesian modeling and based on Kolmogorov's axioms it is very easy to calculate that `P(¬A) = 1 - P(A)`, `P(¬B|A) = 1 - P(B|A)` and `P(¬B|¬A) = 1 - P(B|¬A)`. By reason of this feature BayesOWL introduces *CondProbObjT* and *CondProbObjF* to express conditional probabilities instead of just using one constructor *CondProbObj*.

In fact for representing `P(A)` and `P(A|B)` the only necessary information is

```
<PriorProbObj rdf:ID="P(A)">
  <hasVariable><rdf:value>A</rdf:value></hasVariable>
  <hasProbValue>0.8</hasProbValue>
</PriorProbObj>
<CondProbObjT rdf:ID="P(B|A)">
  <hasCondition><rdf:value>B</rdf:value></hasCondition>
  <hasVariable><rdf:value>A</rdf:value></hasVariable>
  <hasProbValue>0.4</hasProbValue>
</CondProbObjT>
<CondProbObjF rdf:ID="P(B|(not)A)">
  <hasCondition><rdf:value>notB</rdf:value></hasCondition>
  <hasVariable><rdf:value>A</rdf:value></hasVariable>
  <hasProbValue>0.7</hasProbValue>
</CondProbObjF>
```

Figure 4.15: Encoding for Table 4.8 with additional markups.

colored in red in Table 4.8 according to Kolmogorov's axioms. Figure 4.15 presents the encoding of these red marked information for describing these two tables in a machine-readable way.

The main objective of BayesOWL is to provide a method to support ontology mapping by translating an OWL ontology to a BN, e.g. concept satisfiability, concept overlapping and concept subsumption [DP04]. The approach involves augmenting and supplementing OWL semantics with additional language markups for supporting uncertainty reasoning and representing based on BNs. This work is the first published important research effort in the field of the Bayesian extension geared for the semantic web, but due to the special objectives its application potential is very limited. The solutions provided in BayesOWL can meet its own requirements, but not for more complex applications such as DSSs. There are some weak points in the approach.

- BayesOWL restricts itself to the lightweight ontology. It means that only the ontology taxonomy can be translated into BNs. But Many complex domains like disaster management and catastrophe insurance can be correctly modeled only based on the heavyweight ontology.

- The Bayesian extension allows only two-valued random variables. Maybe it is sufficient for ontology mapping, but for real applications it is impossible to specify all Bayesian variables only with boolean values. For example the simple unconditional table represented in Table 4.3 can not be specified in BayesOWL because of this limitation. Therefore a multi-valued approach is absolutely required.

- The additional markups are only allowed in OWL Full. Thus it is difficult to guarantee consistency with the ontological uncertainty modeling. For retaining the ontological reasoning support, it is better to keep the uncertainty extension at the level of OWL DL.
- The language markups can only be used to specify the quantitative information of BNs. For applications relying on BNs it is also necessary to be able to specify the qualitative part.

These weaknesses are solved in the OntoBayes model. In our approach heavyweight ontologies and multi-valued random variables are allowed. In order to retain the full compatibility with OWL, OntoBayes implement the Bayesian extension under OWL DL, which means that the consistency and the ontological reasoning support are guaranteed. As described in Section 4.2, OntoBayes provides both qualitative and quantitative extensions in OWL. Besides, the main purpose of BayesOWL is very different from our approach.

The Bayesian approach of SOCAM

Gu et al. proposed an ontology-based middleware, SOCAM (*Service-Oriented Context-Aware Middleware*), for building context-aware mobile services in intelligent environments [GWPZ04]. The context information in such environments is inevitably uncertain. The uncertain context modeling of SOCAM is based on probabilistic ontologies using BNs [GPZ04].

In order to incorporate BNs into common ontologies, Gu et al. proposed a similar approach to BayesOWL. They slightly modified the existed annotations of BayesOWL, to markup arbitrary conditional probability, for instance using one class constructor *CondProb* to replace *CondProbObjT* and *CondProbObjF* in BayesOWL. But they did not overcome the limitation of two-valued random variables. As mentioned in Ding's approach, it is not sufficient for more complex applications. Generally the way to the quantitative extension of BNs is totally different from our approach.

For specifying qualitative Bayesian information they introduce an additional RDF element `rdfs:dependsOn` which allows to capture dependency between properties. This dependency extension in OWL is similar to the qualitative extension in OntoBayes, but we don't introduce new RDF element to OWL, but just define an object property `dependsOn` with existing OWL primitives. Therefore our extension is fully compatible with OWL DL, whereas the one in SOCAM is not.

The main similarity between the Bayesian extension in SOCAM and OntoBayes is that both of them only allow to specify the dependency between ontology properties, not between classes. The motivation behind such a specification is the

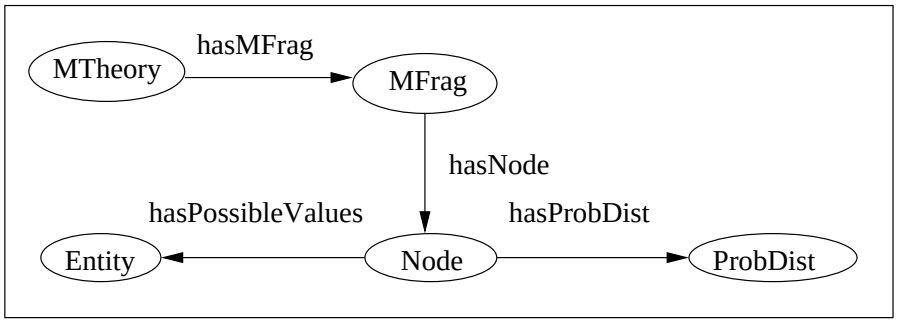

Figure 4.16: A general view of the probabilistic ontology specified in MEBN and PR-OWL.

same — to obtain more context information. In fact SOCAM has stronger context awareness requirements than OntoBayes. The context awareness is one of the main objectives in SOCAM, therefore a new OWL primitive `owl:classifiedAs` was designed for classifying context information and further for supporting context reasoning. In OntoBayes the context information support will be seen as an additional feature of DSSs, but not as the main feature. All of the new primitives in SOCAM cause the incompatibility of the Bayesian extension with OWL DL.

Like BayesOWL, SOCAM did not provide any decision mechanisms which can be directly used by agents for supporting decision making. This missing feature is fully realized in OntoBayes and considered to be one of main features of DSSs.

PR-OWL

In [dCLL05] a probabilistic ontology approach, PR-OWL, was proposed. It aims to justify the lack of uncertainty support in common ontology formalisms and to improve the semantic interoperability in open environments, e.g. in the semantic web vision [CLL06]. Before giving a comparison with OntoBayes and the approaches mentioned above, it is important to investigate and interpret this approach first.

PR-OWL is implemented based on the definition of *probabilistic ontologies* [Cos05] and makes use of MEBN (*Mulit-Entity Bayesian Network*) [LC05, CL06] as its underlying logic basis. Syntactically PR-OWL provides a number of OWL constructs for building probabilistic ontologies, whereas semantically it must accord with the MEBN theory. Figure 4.16 (slightly modified the original one from [CLA06]) presents a general view of the extension from MEBN to PR-OWL by omitting many details of the actual implementation. This figure demonstrates five classes for specifying the most general concepts involved in defining a proba-

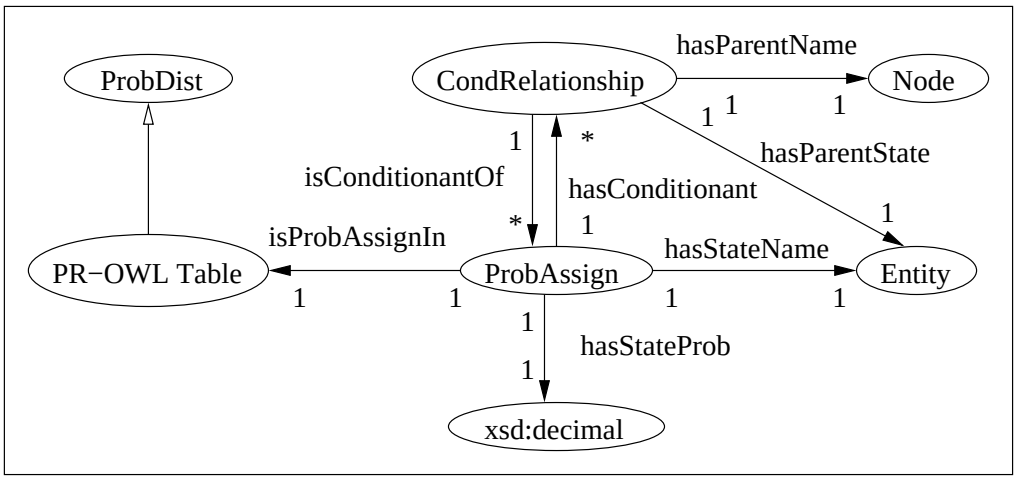

Figure 4.17: The simplified upper ontology of PR-OWL.

bilistic ontology based on the MEBN theory. A probabilistic ontology consists of at least one individual of class *MTheory*. A *MTheory* is formed by a collection of *MFrag*s defined by a set of random variables and other related information [LC05]. In MEBN these variables are represented as *Node*s and can be classified into three types: input, resident and context nodes. Each node has correspondent states and probabilistic distributions, either conditional or unconditional.

In fact the implementation in PR-OWL is more complicated than it was illustrated above. Figure 4.17 tries to capture the essentials of the core part of the probabilistic extension with a simplified upper ontology. The main form of probability distributions in the Bayesian world is probability tables. Costa et al. stated that each class *PR-OWL Table* is a subclass of *ProbDist* and consists of a number of table cells represented by class *ProbAssign*. Such a cell has a probability value assigned for a state of a random variable (i.e. a node) given the states of its parents nodes, when the variable has a conditional probability distribution. In order to specify the conditional table cells, Costa et al. tried to make use of the class *CondRelationship* which actually only expresses the n-ary relation between the involved variable, the table cells, parent nodes of the variable and their states, because in OWL only binary relations can be directly constructed. This n-ary construct leads to the increased complexity when comprehending and using the PR-OWL to build Bayesian probability tables. However the most difficult problem, how to enable the product rule by giving the states of all parent variables to construct a conditional probability table, is still unclear in this approach. A hard coded solution (e.g. a syntax parser) is also unavoidable here. According to [CLL06] PR-OWL provides the following key features:

- PR-OWL makes use of additional language markups at the level of OWL DL to facilitate the integration of probability into ontologies. Therefore it has full compatibility with OWL DL.
- The enhanced expressivity of MEBN based on the first order logic allows to model more complex problems with BNs, especially for representing entity types.
- The flexibility of using PR-OWL for different Bayesian probabilistic tools based on different probabilistic technologies.

In order to achieve the first feature, the backward compatibility with the base language of OWL, PR-OWL must increase the syntactical complexity to model the probabilistic extension in OWL. Figure 4.18 shows how Table 4.8 can be encoded in PR-OWL based on the given upper ontology[6] from [Cos05]. In comparision with Figure 4.15 and Figure 4.3 it has more than 80 lines and obviously is too complex and difficult to be comprehended by users. Based on this comparison the conclusion can be drawn that OntoBayes provides the better solution by expressing the Bayesian probability in OWL. It provides a simple modeling option in the syntactical level but which is still expressive enough. The approach of BayesOWL has a simplest syntax but can not overcome the limitation of two-valued expressions.

The second feature of PR-OWL is only useful when BNs are the only method applicable for problem solving. On the one hand, identifying entity types in BNs can be easily solved in OntoBayes or other methodologies which combine the techniques of BNs and ontologies, because ontologies distinguish their knowledge both at the conceptual and at the individual levels. They provide an excellent feature to classify entities and their instances. On the other hand the MEBN theory is not mature enough for real applications in large complex domains. There are many methods which can be used to enhance the ontological reasoning and expressivity by incorporating first order logic into ontologies (and also in OWL [GHM05]). They can be standardized and better accepted in the semantic web than PR-OWL, because in the semantic world ontologies currently (and in the future) play the key role, not BNs. From this perspective the OntoBayes model provides an open formalism that is also fully compatible with OWL and gives knowledge workers the possibility to adapt the probabilistic ontology with customized first order logic methods.

The third feature is considered as a big benefit of PR-OWL in [CLL06]: "... *That level of flexibility can only be achieved using the underlying semantics of*

[6]This example is encoded with the help of Protégé according to the upper ontology which can be downloaded under the following webpage:
http://mason.gmu.edu/ pcosta/pr-owl/pr-owl.owl

```
<ProbAssign rdf:ID="ProbAssign_B1">
<hasConditionant>
<CondRelationship rdf:ID="CondRelationship_A">
<hasParentState rdf:resource="#true"/>
        <isConditionantOf> <ProbAssign rdf:ID="ProbAssign_notB1">
        <isProbAssignIn>
        <PR-OWLTable rdf:ID="PR-OWLTable_AB">
        <isProbDistOf> <Resident rdf:ID="B">
        <hasProbDist rdf:resource="#PR-OWLTable_AB"/>
        <hasParent><Resident rdf:ID="A"> <isParentOf rdf:resource="#B"/>
        <hasProbDist>
        <PR-OWLTable rdf:ID="PR-OWLTable_A">
        <isProbDistOf rdf:resource="#A"/>
        <hasProbAssign>
        <ProbAssign rdf:ID="ProbAssign_notA">
        <hasStateProb rdf:datatype="http://www.w3.org/2001/XMLSchema#decimal"
            >0.2</hasStateProb>
        <hasStateName rdf:resource="#false"/>
        <isProbAssignIn rdf:resource="#PR-OWLTable_A"/>
        </ProbAssign></hasProbAssign>
        <hasProbAssign>
        <ProbAssign rdf:ID="ProbAssign_A">
        <hasStateProb rdf:datatype="http://www.w3.org/2001/XMLSchema#decimal"
           >0.8</hasStateProb>
        <hasStateName rdf:resource="#true"/>
        <isProbAssignIn rdf:resource="#PR-OWLTable_A"/></ProbAssign>
        ......
        <ProbAssign rdf:ID="ProbAssign_notB2">
        <hasStateName rdf:resource="#false"/>
        <isProbAssignIn rdf:resource="#PR-OWLTable_AB"/>
        <hasStateProb rdf:datatype="http://www.w3.org/2001/XMLSchema#decimal"
                    >0.3</hasStateProb>
        <hasConditionant>
        <CondRelationship rdf:ID="CondRelationship_notA">
        <hasParentState rdf:resource="#false"/>
        <hasParentName rdf:resource="#A"/>
        <isConditionantOf rdf:resource="#ProbAssign_notB2"/><isConditionantOf>
        <ProbAssign rdf:ID="ProbAssign_B2">
        <hasConditionant rdf:resource="#CondRelationship_notA"/>
        <hasStateProb rdf:datatype="http://www.w3.org/2001/XMLSchema#decimal"
                          >0.7</hasStateProb>
        <hasStateName rdf:resource="#true"/>
        <isProbAssignIn rdf:resource="#PR-OWLTable_AB"/></ProbAssign>
        ......
        </ProbAssign></hasProbAssign>
        <hasProbAssign rdf:resource="#ProbAssign_B2"/>
        <hasProbAssign rdf:resource="#ProbAssign_notB1"/>
        <hasProbAssign rdf:resource="#ProbAssign_B1"/>
        </PR-OWLTable></isProbAssignIn>
        <hasConditionant rdf:resource="#CondRelationship_A"/>
        <hasStateName rdf:resource="#false"/>
        <hasStateProb rdf:datatype="http://www.w3.org/2001/XMLSchema#decimal"
             >0.6</hasStateProb>
        </ProbAssign></isConditionantOf>
        <isConditionantOf rdf:resource="#ProbAssign_B1"/>
        <hasParentName rdf:resource="#A"/></CondRelationship></hasConditionant>
        <isProbAssignIn rdf:resource="#PR-OWLTable_AB"/>
        <hasStateProb rdf:datatype="http://www.w3.org/2001/XMLSchema#decimal"
         >0.4</hasStateProb>
        <hasStateName rdf:resource="#true"/>
</ProbAssign>
```

Figure 4.18: Part of the encoding for Table 4.8 in PR-OWL.

first-order Bayesian logic (MEBN) ...". Unfortunately it only possesses the bounded flexibility, if the Bayesian extension in OWL must be bounded with MEBN. The real flexibility can only be achieved when the extension only relies on the syntax and semantic of BNs, neither more nor less. Only such an ontological extension of BNs can be easily applied to diverse Bayesian tools because they have common understanding on Bayesianism. From this perspective the OntoBayes model possesses more flexibility than PR-OWL.

OWL_QM

Pool et al. proposed another probabilistic extension to OWL, OWL_QM (*OWL Quiddity*Modeler*), for eliciting and representing PRM (*Probabilistic Relational Model*) [PFCA05]. A PRM aims to model uncertainty (in the form of probabilistic distributions) about the values of attributes of objects in a certain domain of discourse. Based on the general features of relational models it can express much more information than common BNs [FGKP99]. QM is a representation language for PRM provided by IET[7] and based loosely on frames.

OWL_QM is a similar approach to PR-OWL but makes use of a different underlying uncertain modeling language, PRM. It extends OWL with a number of PRM constructs to represent quiddity facets, slot-chains, variable discretizations and probabilistic distributions and tables [PFCA05]. On the whole OWL_QM is only an implementation of PRM in OWL. In comparison with PR-OWL it provides a smaller extension with less parsing and reasoning supports, but similarly it can not utilize the advantages of ontologies to facilitate knowledge elicitation for decision making. The only way to make the most of probabilistic ontologies is to integrate the probabilistic methodologies and ontologies into a whole model (like the OntoBayes model), not just simply represent the probabilistic techniques in a certain ontology language, e.g. in OWL or in DAML+OWL and so on.

Fukushige's approach

The work of Fukushige [Fuk04, Fuk05] proposed a vocabulary for Bayesian extensions in RDF and a corresponding probability calculation framework. This proposal distinguishes three kinds of probabilistic information encoded in RDF: probabilistic distributions (with unconditional and conditional probabilities), observations (with observed data), and probabilistic beliefs (with posteriors). The motivation of introducing different kinds of these information in the vocabulary is not only to facilitate the representation of the basic description of Bayesian information, but also to facilitate the representation of changing information by belief

[7] It stands for Information Extraction and Transport and is a service company specialized in BNs solutions.

updating. Unlike Fukushige's work, in OntoBayes or other approaches mentioned above we do not distinguish them directly, instead, we focus on the mathematical foundations of BNs with discrete variables, where such variables have either unconditional or conditional probabilistic distributions over their states (or values). Similar to the approaches of PR-OWL and OWL_QM, Fukushige's work simply focused on making use of RDF as the underlying formal language to facilitate the usage of BNs in the semantic web, not preserving the advantages of ontologies.

Other approaches

Giugno and Lukasiewicz proposed P-*SHOQ*(D) for dealing with probabilistic ontologies in the semantic web [GL02]. It is a probabilistic extension of *SHOQ*(D), which extends *SHQ* with individuals and concrete datatypes [HS01]. The syntax of P-*SHOQ*(D) is based on conditional constraints [Luk98], whereas the semantical part is based on lexicographic entailment from probabilistic default reasoning [Luk01]. This approach provides probabilistic reasoning features of consistency, concept satisfiability, concept overlapping etc.. It is useful for supporting ontology mapping, but not sufficient for more generic applications with ontological engineering.

Helsper and Gaag proposed a methodology for building BNs through ontologies [HvdG02]. The main goal of their work is to investigate how important it is to integrate ontologies in a system-engineering approach for developing probabilistic networks [HvdG03]. Based on an oesophagus ontology they demonstrated how to construct an oesophagus BN, but very informally from the theoretical and technical viewpoints. Another deficiency of this method is that it only concentrates on building BNs with the help of ontologies, but not embedding them in ontologies.

Holi and Hyvönen proposed a method in [HH04, HH05] to represent an overlap between concepts and to compute it from the ontology taxonomy. They made use of BNs as the underlying uncertainty technique for representation and computation. A graphical notion for modeling the concept subsumption can be represented easily in RDF(S). But this method results in an extremely large and complicated BN when too many relations and overlaps among concepts are translated.

4.4.2 The Fuzzy Approaches

Besides the probabilistic approaches discussed above, there are also other approaches to uncertainty extensions for ontologies such as fuzzy logic. Stoilos et al. proposed a method for extending OWL with fuzzy set theory, *Fuzzy OWL* (or *f-OWL*), in order to capture, represent and reason with uncertain information in the semantic web [SST$^+$05].

Fuzzy OWL is a fuzzy extension of OWL DL with additional degrees to OWL facts. It makes use of crisp OWL's syntax as its building blocks. In order to express the fuzzyness degree added to the facts, it introduces an additional element `<owlx:degree>`. The reasoning feature is implemented by combining the syntactical extensions with f-*SHOIN*, which extends *SHOIN* to the fuzzy case by letting concepts and roles denote fuzzy sets of individuals and relations among them respectively. In f-*SHOIN* the fuzzy knowledge base contains fuzzy *TBox*, *RBox* and *ABox*, where each *TBox*, *RBox* and *ABox* is a finite set of fuzzy concept axioms, fuzzy role axioms and fuzzy assertions respectively [SST$^+$05]. A reasoning engine for Fuzzy OWL, *FiRE*, was proposed in [SSSK06].

Another fuzzy approach for extending OWL, FOWL, was proposed by Gao and Liu in [GL05]. They provide a number of new vocabularies for encoding fuzzy constructs, axioms and constraints, in order to map them to fuzzy DL. Besides the vocabularies, some rules are specified to translate OWL to FOWL, because from the viewpoint of fuzzy set, some common OWL concepts are also special fuzzy concepts. But their work still has the lack of syntactical parser and a reasoning machine for FOWL.

Chapter 5

Virtual Knowledge Communities

This chapter is devoted to investigate how to address the emerging paradigm of knowledge dissemination and collaboration in decision support systems through VKCs.

In Section 5.1 we will give the motivation and definition of VKCs. Afterward an overview of all basic concepts using to model VKCs will be introduced according to [Ham04] in Section 5.2. Section 5.3 and 5.4 are devoted to investigate how to make use of VKCs to facilitate knowledge dissemination, particularly with emphasis on knowledge sharing in the OntoBayes model. We will follow the notations introduced in the last Chapter for denoting datatype or object properties in OntoBayes. In the last section we will demonstrate how VKCs can be utilized for supporting decision making in terms of collaboration and adaptivity.

5.1 Motivations and Definitions

Before we introduce what VKCs are, we will first describe what corporate knowledge is and investigate how decision making support can profit from corporate knowledge. Then the AOA (*Agent Oriented Abstraction*) paradigm will be introduced as a grounding theory behind the facilitation of corporate knowledge in the society of agents, since it was demonstrated in [MC04] that corporate knowledge can take advantages of AOA. At last we will define VKCs and explain the natural link between corporate knowledge and VKCs.

Corporate Knowledge and Decision Making Support

Nowadays most DSSs put much greater emphasis on knowledge management. This is clearly justified. Making a decision once one has the right knowledge is often the easy part. Under an open, dynamic and uncertain environment, decision

making based on corporate knowledge has become crucial for a society made of distributed agents each possessing its own knowledge, particularly when the knowledge is uncertain.

Corporate knowledge was defined as the overall knowledge detained by agents within a system and their ability to cooperate with each other in order to meet their goal [MHC04]. During a process of decision making agents can make decisions more easily, precisely and rationally based upon corporate knowledge than only based on their sole knowledge. Agents are required to know not only "what it knows" but also "what they know", and are expected to make maximum use of their knowledge.

Agent Oriented Abstraction

The AOA paradigm [CME04] covers the concepts of agents, annotated knowledge, utility functions and society of agents. Indeed, AOA is based on Weber's classical theory in Sociology [Web86]. AOA assumes that agents are entities consisting of both a knowledge component and a decision making mechanism. The former is partitioned into four components, also called annotations: ontology, communication, cognition and security. The latter is related to its tasks and goals. It generates utility functions and is based upon the knowledge component. Chiefly, agents can be defined in terms of knowledge and action's utility. The AOA model can be abstractly summarized by a number of basic definitions. A detailed description is to be found in [CME04].

In [MC04] applications of the AOA model to the abstract modeling of corporate knowledge are investigated. Corporate knowledge was defined as the amount of knowledge provided by individual agents. To avoid the separation between agents and knowledge, it was considered that agents have explicitly represented knowledge and communication ability. A knowledge company was modeled as a scenario to demonstrate the corporate knowledge modeling within the AOA. The concrete implementation for corporate knowledge within AOA was introduced in [MHC04].

Communities, Virtual Communities and VKCs

Traditionally, information is mostly centralized within a uniform information structure. This viewpoint is not truly compliant with the nature of knowledge that is subjective, distributed and contextual [BBC02]. From the perspective of the knowledge information society, modern knowledge management often focuses on the constitution of communities of practice and communities of interest [FO01]. In the real life society or organization, a *community* can be seen as a group in which individuals come together around a shared purpose, interest, or goal, but

the communication between the individuals are often offline, face-to-face, for example communities of practice [Wen98] and so on.

Nowadays such communities are becoming more and more *virtual* due to the internet revolution, particularly due to the web 2.0 revolution which mainly focuses on the online collaboration and sharing among users. Built upon the supports of modern IT infrastructures more and more virtual organizations or enterprises can go beyond the physical distance and organizational boundaries, in order to improve their efficiency and ability to support sharing of resources in a timely fashion as well as to maximize their economical profits. From this perspective, a *virtual community* can be seen as a society of individual agents coming together around a shared purpose, interest, or goal, but using computer supports rather than face-to-face interactions for their communications [RS01]. Most virtual communities exist therefore purely in cyberspace [KKBB07].

The concept of virtual communities can be supported in a *virtual knowledge community* in order to bring the concerned agents together to share their knowledge with each other. VKCs can be abstractly defined as a mean for agents to share knowledge about a topic [MHC04]. It aims to increase the efficiency with which information is made available throughout the society of agents.

From the point of view of corporate knowledge management, agents can be individuals, software assistants or automata. Agents possess knowledge and processes within the society which tends to make agents produce and exchange knowledge with each other. These processes are distributed throughout the society and contribute through their own intrinsic goals to solve a unique high level challenge, for example solving a decision problem. This provides the link between corporate knowledge and VKCs [MC04]. Generally VKCs can be seen as the realized concept for corporate knowledge.

5.2 Basic Concepts of VKCs

In this section we will give a simple overview of the basic concepts according to the first prototype of VKCs [Ham04]. In this prototype there are two main modeling for VKCs: *agent modeling* and *community modeling*.

Agent Modeling

The agent modeling has four key notions: personal ontology, knowledge instances, knowledge cluster and mapper. A *Personal ontology* represents the knowledge of an agent. It describes the taxonomy of the relationships between the concepts and predicates what an agent understands. The *knowledge instances* are instances of objects defined into the personal ontology. It was assumed that an agent's knowl-

edge consists of both its personal ontology and knowledge instances according to its personal ontology. The *knowledge cluster* as a sub-part of an ontology can be shared among agents. It is defined by a head concept, a pointer to the different parts of knowledge existing in the cluster. The *mapper* chiefly contains a set of mapping from personal terms to mapped terms, and allows agents to add such mappings, and use the mapper to normalize or personalize a given knowledge cluster or instance. It facilitates knowledge sharing among agents with regards to the heterogeneity of knowledge.

Community Modeling

The community modeling has also some key notions: domain of interest, community pack and community buffer. A *domain of interest* exists in each virtual knowledge community and is similar to the concept of ontology for agents. It is given by a community leader who created the community. The *community pack* is what defines the community. It consists of a community knowledge cluster, a *normalized ontology* which contains at least the head of the community cluster, and the identification of the leaders of the community. The *community buffer* can record messages which are used by the member of a community to share their knowledge. This approach is compatible with blackboard systems, but still has its difference, because agents cooperate to solve their respective problems, not towards a unique goal.

Knowledge Community Process

Agents' actions related to knowledge communities are the following ones: initiate, reorient, leave, terminate and join a community as well as exchange knowledge. Every agent can initiate a community by creating a topic and a community buffer and advertising about this community. Advertising is done through a specific agent called *community of communities*, which has a central directory of all communities. All agents of the system are members of this community. At the same time of initiating a community, a community park is also created. It contains a knowledge cluster of the initiator, a normalized ontology, and the identification of the initiators. The information will be posted to the community buffer.

Community reorientation is needed because the knowledge can not be uniquely considered at design time. It should evolve over time. Reorientation consists of sending a new community cluster to the community of communities.

Agents can leave a community voluntarily, but it could be also forced out by the leaders. When a leader leaves a community, a new leader is required, if it is the unique leader in this community.

Community termination consists of erasing the community buffer and its reference posted to the community of communities during the community's lifetime.The community can only be terminated by one of the community leaders.

5.3 Knowledge Sharing through VKCs

The main objective to use VKCs in DSSs is to enable decision making based on corporate knowledge, i.e. DSSs need VKCs working as a platform for facilitating knowledge exchange between different individual agents. Before we demonstrate how VKCs can facilitate knowledge sharing, it is necessary to make clear which kinds of information can and will be exchanged through VKCs, particularly in the context of OntoBayes.

DSSs built upon the proposed framework (see Chapter 3) possess the OntoBayes model. As described in the last chapter OntoBayes consists of two parts: a knowledge part and a decision model part. The former refers to normal ontological knowledge (which is certain) and Bayesian knowledge (which is uncertain), whereas the latter refers to different (formal) decision models corresponding to different decision problems. In this chapter we will show that we can utilize and adapt VKCs for sharing information in both parts of OntoBayes, not only the ontological and Bayesian knowledge, but also the decision models.

As mentioned in Chapter 4 the OntoBayes model is ontology-driven, because the selected underlying representation language both for the knowledge part and for the decision part is OWL. In fact the main challenge for knowledge sharing now is how to integrate OWL with VKCs syntactically and to take the semantical consistency of BNs and IDs into account at the same time.

5.3.1 A Simple Scenario

To demonstrate how to exchange ontological knowledge, Bayesian knowledge and decision models through VKCs, a simplified scenario of information sharing between actors in the insurance field is as follows.

There are three kinds of basic actors operating in the field: insurance companies (or providers), insurance agents and insurance customers.

- Insurance companies offer a product range. Nowadays the companies can not only utilize insurance agents as their selling channels, but also sell their products directly to customers via web services [1].

[1] A company can open its own B2C-site to offer its own products without paying any provision to insurance agents.

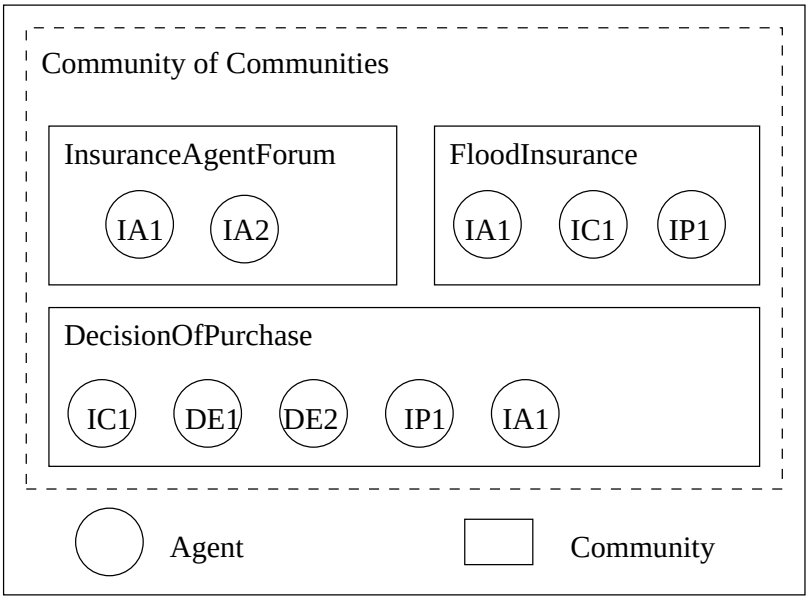

Figure 5.1: The simplified view of virtual knowledge communities of the catastrophe insurance scenario.

- Insurance agents[2] offer a selling and distribution channel. Traditionally, one agent is "hooked" up to only one insurance company, nowadays they can easily switch from one company to another via web services[3].
- Insurance customers can buy a product (i.e. an insurance policy) either from an insurance agent or from an insurance company directly. For a customer it makes no real differences whether he or she buys the product from whom. The most important factor for making a purchase decision is the premium of an insurance policy.

In this scenario, most agents can be divided into these three actor groups described above. To simplify the description we denote "IP", "IA" and "IC" for these three actor groups of insurance companies, of insurance agents and of insurance customers, respectively. In order to distinguish different agents having the same role, we add numbers to these notations. For example "IA1" and "IA2" have the same role of insurance agents, but they are different individuals. Besides these actors, there could be other participants in the scenario, for example, domain experts, knowledge engineers and so on.

[2]Here the insurance agents differ from the term "agents" which is very often used in this thesis. The latter refers to the multiagent paradigm.

[3]Web services offer a new distribution channel for insurance companies, but they also offer agents more flexibility in offering products and services from different product providers.

In the community of communities there are three VKCs illustrated in Figure 5.1. In the VKC "InsuranceAgentForum" there are only two agents: "IA1" and "IA2". They want to sharing information about their products. In the VKC "FloodInsurance" there are three agents: "IA1", "IC1" and "IP1". It may be created by the agent "IC1" for finding appropriate insurance products against flooding. The VKC "DecisionOfPurchase" has many participants: "IA1", "IC1", "IP1", "DE1" and "DE2", where "DE1" and "DE2" are domain experts for making purchase decisions and want to help "IA1" complete its decision model.

We will make use of these VKCs to illustrate the knowledge exchange of ontologies, BNs and IDs in the following sections, respectively.

5.3.2 The Exchange of Ontological Knowledge

For facilitating the exchange of normal ontological knowledge, we can make use of the basic concepts introduced in Section 5.2. The basic unit of ontological knowledge can be exchanged in any community is the RDF triple illustrated in Figure 2.2. In [HYC06] we have illustrated a concrete e-business scenario for showing how to exchange ontology-driven knowledge through VKCs in details. Here we just simply demonstrate the knowledge exchange between agents in the VKC "InsuranceAgentForum" in Figure 5.2.

In this VKC there are two agents: "IA1" and "IA2", and both of them are insurance agents, but independent of each other. Messages in the community buffer are structured through simplified notations of ontologies with concepts, their instances and relations. For example, the notation `InsuranceProvider: {IP1}` denotes that the concept `InsuranceProvider` has an instance `IP1`. The dashed-arrowed line between agents and the community buffer shows the operation in an exchange process.

The agent "IA1" created this community and wrote a message in the community buffer. This message contains the information about the premium and the risk coverage of an insurance product "EarthquakeInsurance" provided by "IP1". Besides this agent there is another agent "IA2" who has also interest in this community. It joined the community and posted a message. This message contains the same ontology structure, but with different instances.

After the message input both of them read the messages posted from other agents in this community, to complete a simple knowledge exchange process. After the knowledge exchange, both of "IA1" and "IA2" know that they can provide more concrete insurance products for their customers, for instance insurance against flood or against earthquake with a given premium.

According to the basic concepts of VKCs there are simply two levels of knowledge exchange in ontologies: at the level of knowledge cluster and at the level

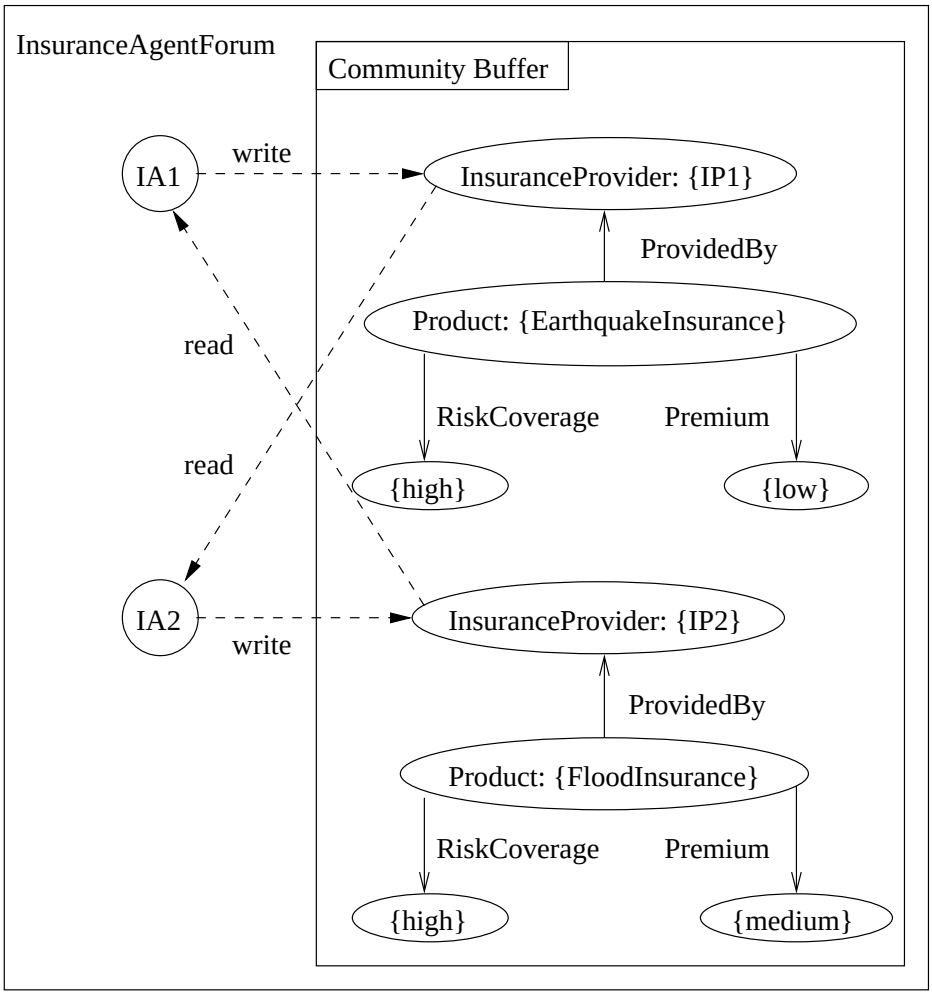

Figure 5.2: A simple example for ontological knowledge exchange.

of knowledge instance. The former concentrates on the structure of ontologies, whereas the latter focuses on the instances of ontologies.

5.3.3 The Exchange of Bayesian Knowledge

As described in Section 4.2 Bayesian knowledge in OntoBayes is also ontology-driven. Syntactically it is specified in OWL, therefore it can be shared between agents through VKCs in principle. But some adaption is required due to the different semantic between ontologies and BNs. According to Definition 2.9 and the upper ontology of BNs (as illustrated in Figure 4.1), the basic exchangeable unit of Bayesian knowledge is abstracted in Figure 5.3.

Each exchangeable basic unit must contain two parts of information: quali-

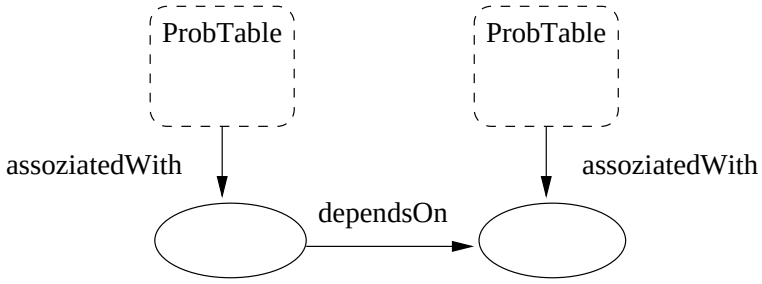

Figure 5.3: The basic exchangeable unit of Bayesian knowledge.

	Levels of knowledge exchange	
	Knowledge cluster	**Knowledge instance**
Ontological knowledge	Ontological structure: concepts, relations	Ontological instances: instances of concepts
Bayesian knowledge	Bayesian structure: Bayesian variables and the dependency relation beteen them	Bayesian instances: instances of Bayesian variables and their probability distribution

Table 5.1: The difference of knowledge exchange between BNs and ontologies

tative and quantitative. The former is the structure basic unit, whereas the latter is the numerical basic unit. Each numerical unit is simply a probability table associated to a variable (represented by ellipse) in the qualitative part. In fact each probability table in OntoBayes is an instance of the class `CondProbDist` or `UncondProbDist` according to the extended OWL specification in Section 4.2.

The Knowledge Exchange

Table 5.1 shows the differences of knowledge exchange between BNs and ontologies both at the level of knowledge cluster and knowledge instance. In comparison with the ontological knowledge exchange, the knowledge cluster in Bayesian knowledge contains the qualitative information of BNs, i.e. the variables and the dependency relations between them. And the knowledge instance contains both the qualitative and quantitative information, i.e. the instances and the probabilistic distributions of the variables.

In Figure 5.4 we demonstrate the knowledge exchange between agents in the VKC “FloodInsurance”. The agent “IC1” created this VKC for finding a suitable insurance product against flooding. It posted a piece of its Bayesian knowledge in a message to indicate that an action for buying an insurance product depends on the premium of the product. This VKC drew attention of another agent “IP1”. So

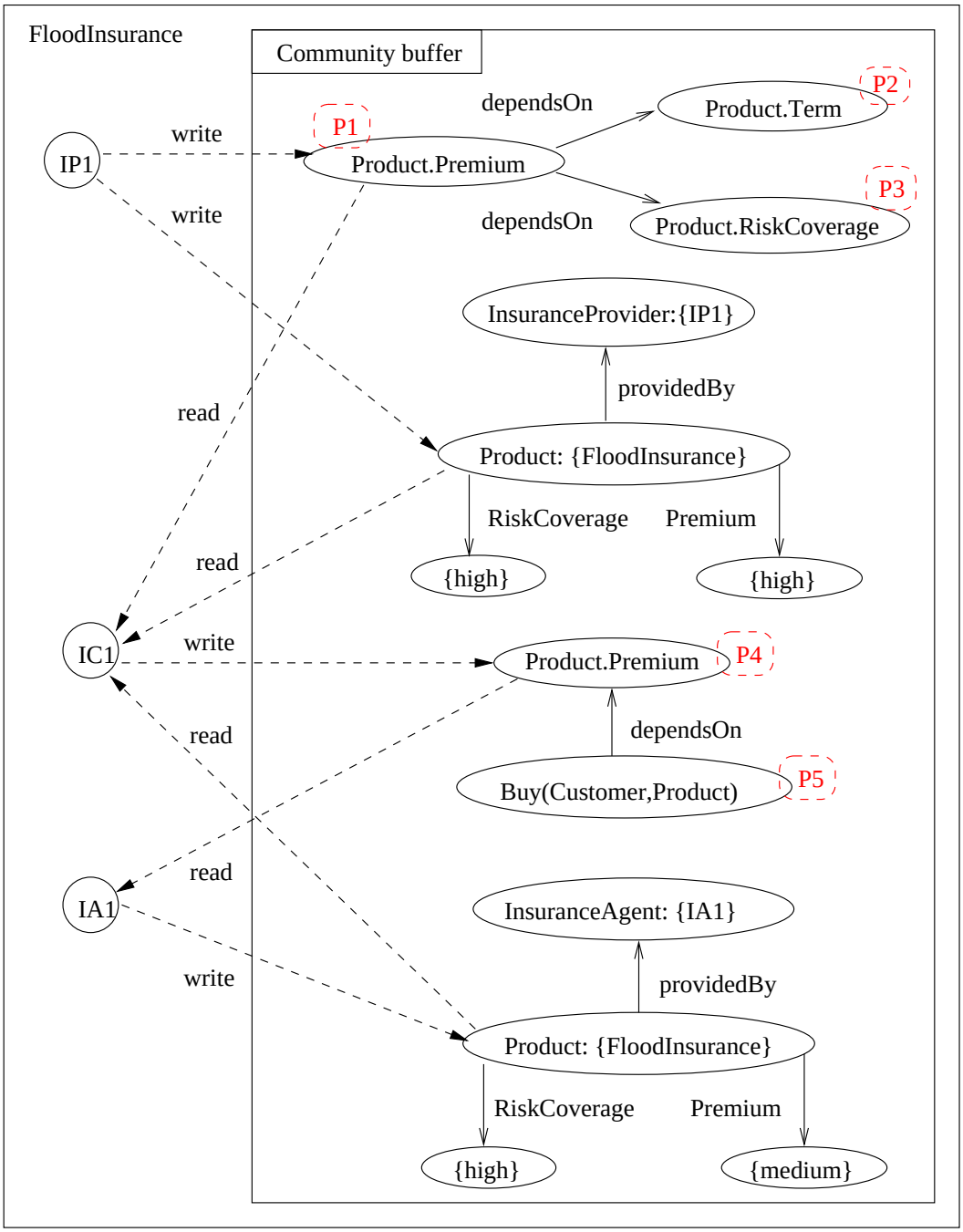

Figure 5.4: The knowledge exchange in the VKC “FloodInsurance”.

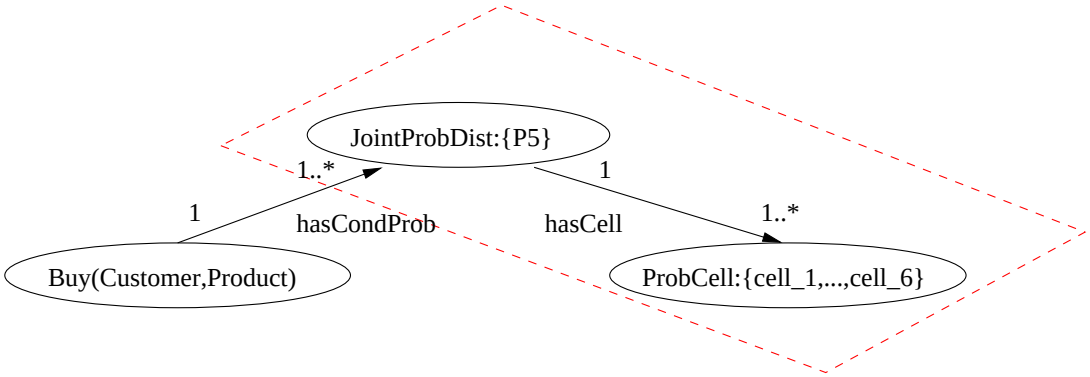

Figure 5.5: The knowledge sharing of the probability table P5 in the VKC "Flood-Insurance".

it joined this VKC and posted two messages: one is for its Bayesian knowledge about the dependency relation between the premium, term and risk coverage of an insurance product — the premium of an insurance product depends on the term and risk coverage of the product; another one is for its offer of a flood insurance policy. Besides "IC1" and "IP1" there is another agent "IA1" who has also interest in the VKC and wrote a message, because after the knowledge exchange with "IA2" in the VKC "InsuranceAgentForum" (see Figure 5.2) it knew that it can offer a flood insurance policy to "IC1", but provided by "IP2" with different policy constraints comparing with the one of "IP1". Therefore it wrote a message to provide such information to "IC1".

As illustrated in the figure both of Bayesian knowledge and ontological knowledge are exchanged between agents in this VKC, because these agents are built upon OntoBayes and have an integrated approach of their knowledge. It must be pointed out that at the same time of exchanging the qualitative information of Bayesian knowledge, the corresponding quantitative information must be exchanged too. In order to obtain a good overview of the figure, we simplified the quantitative information as red boxes labeled with P. In fact each of such boxes denotes a probabilistic table associated with a Bayesian variable, either conditional or unconditional.

As pointed out in Section 4.2 the probabilistic information of a BN is specified in OWL, i.e. that it is ontology-driven. Therefore, it can be exchanged through VKCs in principle as well as the normal ontological concepts, i.e. exchange based on RDF-triples. For example, the conditional probability table P5 in the VKC "FloodInsurance" can be illustrated in Figure 5.5 as a bundle of instances in the RDF-triple

$$JointProbDist \xrightarrow{hasCell} ProbCell\cdot$$

Generally we specified in the upper ontology of BNs (see Figure 4.1) that a prob-

ability table has arbitrary number of cells (at least one). But theoretically the number of cells in a table is computable based on the state of related variables. For example, if the variable `Buy(Customer,Product)` has two states: {`yes, no`} and `Product.Premium` has three states: {`low, medium, high`}, then `P5` has exactly 6 ($= 2 \times 3$) cells as shown in Figure 5.5. More detailed syntactical representation of such tables is described in Section 4.2.3 previously. Here we just want to emphasize the semantic feature — the *completeness* of probability tables — when exchanging them in VKCs. It means that a probability table is only exchangeable when its entire cells are complete. For instance, `P5` has 6 cells and can not be exchanged if anyone of these 6 cells is not involved in the exchange process.

The Knowledge Evolvement

After the exchange each agent can adapt its old knowledge base respectively to its knowledge cluster and personal ontologies with the new information. The feature introduced in the above example is illustrated in Figure 5.6 and Figure 5.7 respectively. They show the updated knowledge of the agent "IC1", including both of ontological and Bayesian knowledge.

"IC1" can evolve its ontological knowledge by merging the messages inputed by "IP1" and "IA1" respectively. The result of the merge is shown in Figure 5.6 which includes both conceptual and individual knowledge. For example the concept "Product" is now linked to two different concepts, `InsuranceAgent` and `InsuranceProvide`, with the object property `providedBy`. This is the knowledge evolvement at the conceptual level. At the individual level "IC1" knows now that there are two providers who can offer him the product instance `FloodInsurance`. One provider is "IA1" and the other is "IP1". Each provider offers an `Floodinsurance` with different constraints. In the figure we distinguish the different offers with the green and blue color, respectively.

"IC1" can also complement its Bayesian knowledge about the Bayesian variable `Product.Premium` with the dependency relations to other variables `Product.Term` and `Product.RiskCoverage`. The probabilistic distributions about each variable must be taken into account too.

Most Variables can simply take over the probability tables posted in the VKC. But some variables can not do that, because these probabilistic information is not compatible in the following situations:

- One variable has both conditional and unconditional probability tables in a VKC. For example the variable `Product.Premium` had two probability tables `P1` and `P4` in the VKC "FloodInsurance". `P1` is the conditional one posted by the agent "IP1", whereas `P4` is unconditional and posted by the agent "IC1".

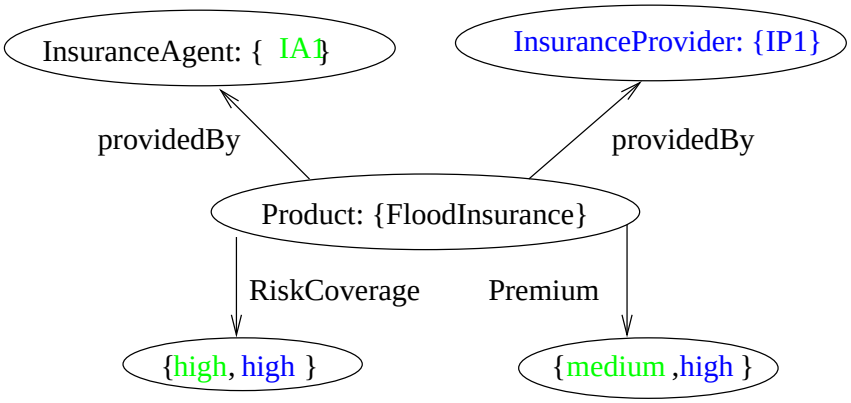

Figure 5.6: Ontological knowledge evolvement of the agent "IC1" following the knowledge exchange in the VKC "FloodInsurance".

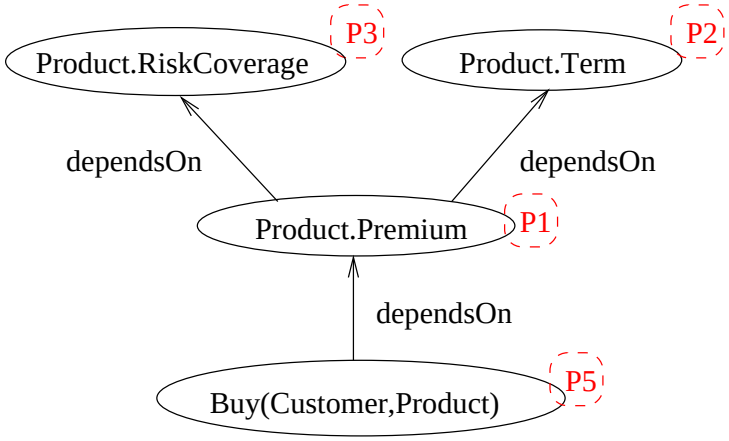

Figure 5.7: Bayesian knowledge evolvement of the agent "IC1" following the knowledge exchange in the VKC "FloodInsurance".

In this case the variable will take over the conditional one, because the structural type of the variable in a BN is changed from unconditional to conditional. Therefore in Figure 5.6 the variable `Product.Premium` can only take over the conditional probability table `P1`.

- One variable has more than one probability table in a VKC, but with the same type (either conditional or unconditional).

  In this case the variable will take over all the tables, because as specified in OntoBayes we allow a Bayesian variable to have more than one probability table. But among them, only one table is active.

We mentioned that the completeness of probability tables must be guaran-

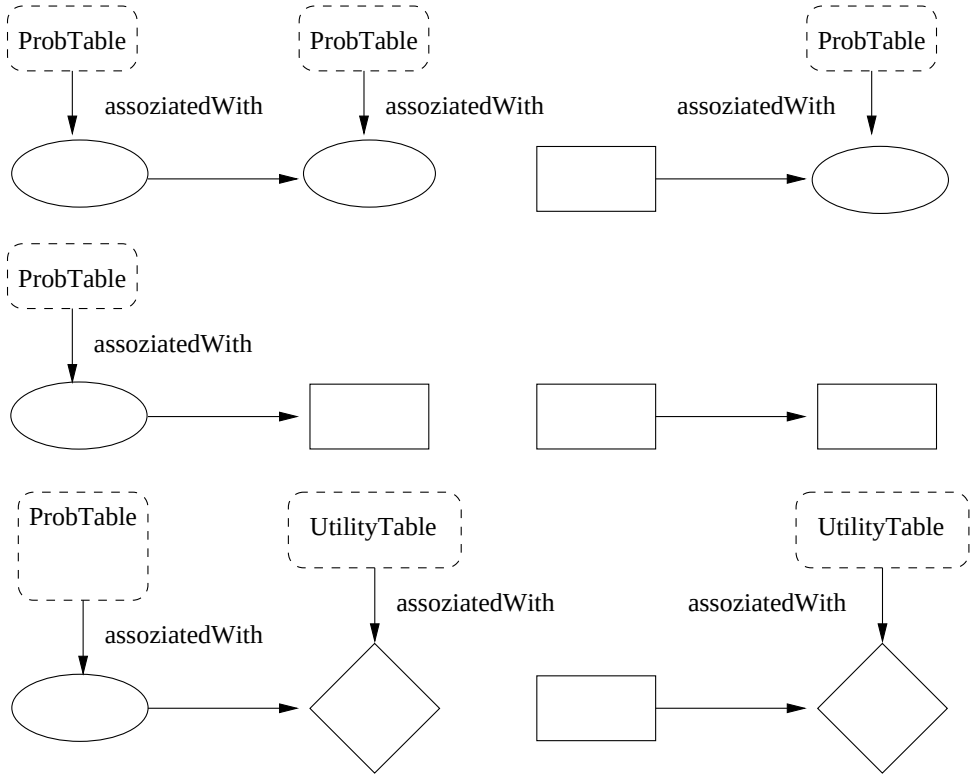

Figure 5.8: The basic exchangeable unit of IDs in VKCs.

teed during the quantitative Bayesian knowledge exchange. Except that, there is another semantical restriction for the qualitative Bayesian knowledge exchange according to the definition of BNs: the DAG condition. Every BN must be a DAG. Before the knowledge exchange it can be insured that the Bayesian knowledge taken by an agent is sound. It is expected that the situation remains the same after the knowledge exchange. In order to make it sure, the qualitative knowledge evolvement of a BN must be checked with the DAG condition. It means that any new variable added into a BN will be guaranteed not to cause a circle in the BN.

5.4 Decision Model Sharing through VKCs

Similarly as for the Bayesian knowledge, decision models in OntoBayes are represented based on IDs and are also ontology-driven. They can be specified in OWL formally. Therefore decision models can be shared between agents through VKCs too. According to Definition 2.18 and the upper ontology of IDs (as illustrated in Figure 4.9), the basic exchangeable units of IDs are abstracted in Figure 5.8.

There are totally 6 basic exchangeable units of IDs and three types of nodes

	Levels of knowledge exchange	
	Knowledge cluster	**Knowledge instance**
Ontological knowledge	Ontological structure: concepts, relations.	Ontological instances: instances of concepts.
Bayesian knowledge	Bayesian structure: Bayesian variables and the dependency relation between them.	Bayesian instances: instances of Bayesian variables and their probability distribution.
Decision models with IDs	Structure of IDs: chances nodes, decision nodes and utility nodes and the relations between them.	Instances of ID: instances of all kinds of nodes, the probability distributions of chance nodes, the utility function of utility nodes.

Table 5.2: The difference of knowledge exchange between ontologies, BNs and IDs.

in such units: chance nodes, decision nodes and utility nodes (for more detail descriptions see Page 36). The exchange of such units is obviously more complex than the exchange of Bayesian units, due to the additional two node types: decision nodes and utility nodes.

The chance nodes can be exchanged as when exchanging Bayesian units, because they are in fact Bayesian variables associated with probabilistic distributions. For the decision nodes we can simply make use of the method for exchanging ontological knowledge as described in Section 5.3.2, because they are not associated with any extra information and each of them can be considered as a "subject" or an "object" node in a RDF-triple. But, for the utility nodes we need to take the associated utility functions (simplified as utility tables in OntoBayes) into account as when dealing with the chance nodes and their probability tables.

The Exchange of Decision Models

Table 5.2 extends Table 5.1 to show the differences of knowledge exchange between ontologies, BNs and IDs, both at the level of knowledge cluster and knowledge instance. In comparison with the ontological and Bayesian knowledge exchange, the knowledge cluster in IDs contains more structural information and the knowledge instance contains not only the instance of nodes, but also the probability tables and utility tables.

In Figure 5.9 we demonstrate the knowledge exchange of decision models between agents in the VKC "DecisionOfPurchase". The agent "IC1" created this VKC for completing its decision model to support making a purchase decision.

U(Buy(Customer,Product),Product.Premium)	Utility Value
U(yes,small)	-100
U(no,small)	50
U(yes,medium)	40
U(no,medium)	10
U(yes,high)	80
U(no,high)	-80

Table 5.3: The utility table U1 for $U(par(U)) = U(Product.Premium, Buy(Customer, Product))$.

Based on the Bayesian knowledge that it obtained from the VKC "FloodInsurance" (see Figure 5.7), it can construct a simple decision model by converting the Bayesian variable `Buy(Customer,Product)` to a decision node. Then "IC1" inputed this model as a message in the community buffer. In order to complete this model, a utility node with a utility table related to the decision node `Buy(Customer,Product)` is required.

As illustrated in the figure, a domain expert "DE1" (may be an economist) joined this VKC to help "IC1" complete the model. It posted a message that contains a utility node `U` with a utility table `U1`. This utility node has two parent nodes: the decision node `Buy(Customer,Product)` and the chance node `Product.Premium`. The utility table is specified as

$$U(par(U)) = U(Buy(Customer, Product), Product.Premium)$$

in Table 5.3.

Another domain expert "DE2" (may be a psychologist) who knew that the risk behavior of an agent is an important factor to make a purchase decision. It joined the VKC too and posted its knowledge according to the decision problem. This message contains a utility node `U` associated with a chance node `Customer.RiskBehavior`. And the domain value of the chance node is specified as $dom(Customer.RiskBehavior) = \{aversion, neutrality, seeking\}$. Chance nodes in IDs are in fact variables in BNs, therefore they must be specified with probability tables as explained in Section 5.3.3. In the VKC "DecisionOfPurchase" the chance node `Customer.RiskBehavior` possesses a probability table `P6`.

According to $dom(Customer.RiskBehavior)$ the utility table `U2` of the node `U` is constructed in Table 5.4. The way to exchange utility tables is similar to exchange probability tables as mentioned in the last section, because as specified in Section 4.3 the underlying encoding of these tables is OWL. It means that they can be exchanged based on RDF-triples. In order to explain it more clearly, we illustrate the utility table `U2` in Figure 5.10. `U2` is an instance of the ontology

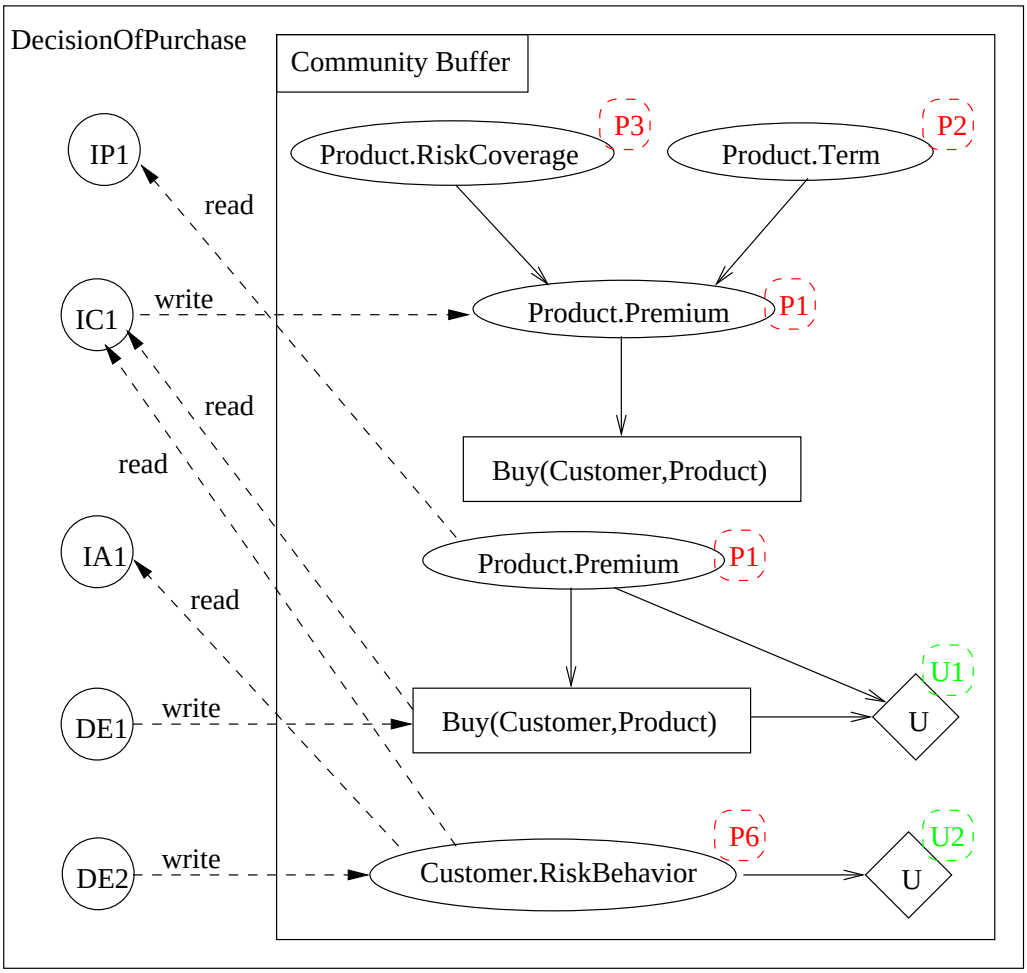

Figure 5.9: The decision models exchange in the VKC "DecisionOfPurchase".

U(Costomer.RiskBehavior)	Utility Value
U(aversion)	-50
U(neutrality)	0
U(seeking)	50

Table 5.4: The utility table U2 for $U(par(U)) = U(Costomer.RiskBehavior)$.

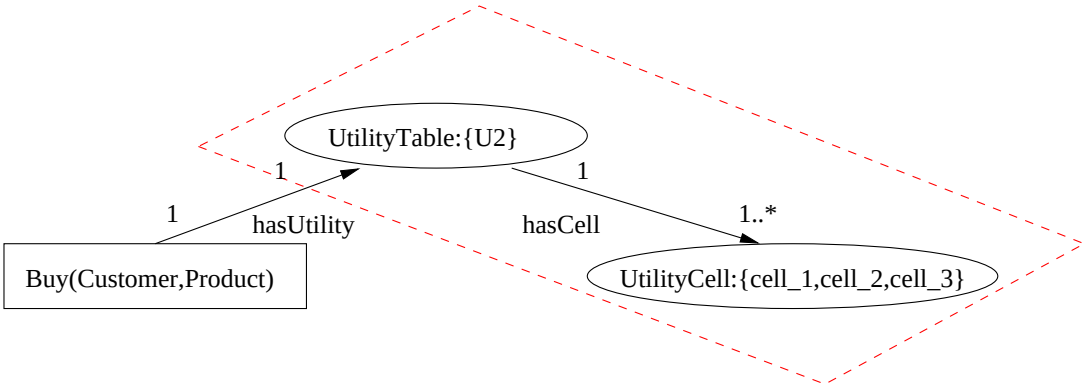

Figure 5.10: The knowledge sharing of the utility table `U2` in the VKC "DecisionOfPurchase".

class `UtilityTable` which is associated with another class `UtilityCell` based on the predicate `hasCell`. Each utility table has at least one cell as specified in the figure. The instance `U2` possesses exactly 3 instances of `UtilityCell`: `cell_1`, `cell_2` and `cell_3`, according to $dom(Customer.RiskBehavior)$. Like probability tables, the semantic feature — the *completeness* of utility tables — must be taken into account when exchanging them in VKCs.

The Evolvement of Decision Models

After the exchange of decision models each agent can adapt its old decision model base respectively. In Figure 5.11 we illustrate that the agent "IC1" completed its decision model for purchasing a catastrophe insurance product after the exchange in the VKC "DecisionOfPurchase".

Before the exchange process the agent "IC1" only knew that it can make a decision on the node `Buy(Customer,Product)`, but did not know how it should take this decision. After the exchange it has a complete model with the help of two domain experts "DE1" and "DE2". Now "IC1" knows that it should make a decision based on the utility measure of the node `U`. This utility node merged two utility tables posted in the VKC "DecisionOfPurchase". The question is how to merge them? As mentioned in Section 2.4 the most used method to deal with this problem is to create an associated utility which is a sum or product function of these utility nodes. as illustrated in Figure 5.11, "IC1" can merge `U1` and `U2` into one utility node `U` with a sum function. For example, if the risk behavior of "IC1" is identified as `risk seeking`, then according to Table 5.4 we must add 50 to all utility values in `U1`. The resulted table is represented in Table 5.5.

We mentioned that the completeness of utility tables must be guaranteed during the exchange. Except that, there is another semantical restrictions for exchang-

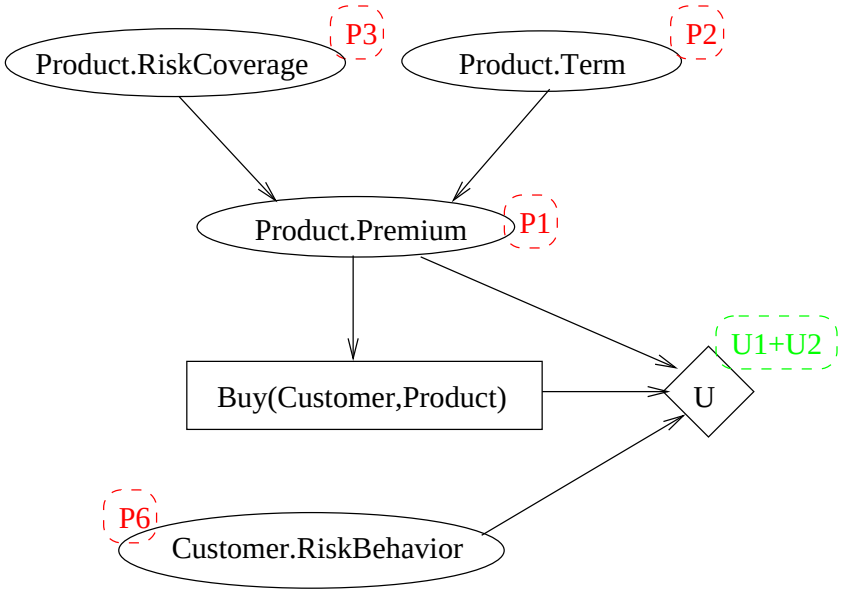

Figure 5.11: The agent "IC1" complete its decision model for purchasing catastrophe insurance product after the knowledge exchange in the VKC "DecisionOfPurchase".

U(Buy(IC1,Product),Product.Premium,IC1.RiskBehavior)	Utility Value
U(yes,small,seeking)	-100+50
U(no,small,seeking)	50+50
U(yes,medium,seeking)	40+50
U(no,medium,seeking)	10+50
U(yes,high,seeking)	80+50
U(no,high,seeking)	-80+50

Table 5.5: The utility table U1+U2 for $U(par(U)) = U(Product.Premium, Buy(IC1, Product), IC1.RiskBehavior)$ with identified risk behavior: *seeking*.

ing decision models according to the definition of IDs: the DAG condition and the total order condition (see Page 37) of IDs. Every ID must be a DAG. Before the knowledge exchange it can be insured that an ID taken by an agent is a DAG, and the same condition will be expected after the exchange. In order to make it sure, for any new node added into an ID it will be checked whether it causes a circle in the ID.

The total order of an ID indicates that there must be a directed path through all decision nodes in the ID. To avoid violating this semantic feature of IDs, after adding any new decision node to an ID, it must be checked that there is a direct path throughout all decision nodes including the new one.

5.5 Collaboration through VKCs

In the last sections we demonstrated how to enable distributed KM through VKCs, with emphasis on knowledge sharing in the OntoBayes model. In this section we will show how to facilitate the collaboration and adaptivity through VKC.

Collaboration for Virtual Teams

As discussed in Section 5.1, most organizations or enterprises have distributed structures in different places or even virtually, in order to keep their global competence. Instead of teams with face-to-face communications, virtual teams for collaborative problem solving are required in such organizations, due to the physical distance (different locations and time zones) and to organizational boundaries.

In Figure 5.12 a simplified DSS is illustrated to provide a generic view for *collaboration* through VKCs. In the system there are many components: agent management, knowledge management, reasoner and so on. A system user, the decision maker, can get support through the system interface to solve a decision problem. The system assigns a domain expert and a knowledge engineer to the decision maker, in order to facilitate the *basis development* (which is the first step in the decision analysis process presented in Figure 3.5) of the problem. The system component “Agent Management” creates three agents, “DM”, “DE” and “KE” for representing these three human users in the system respectively. These three agents build a virtual team “VKC_1” that only aims to deal with this decision problem. But the basis development can not be completed only with the help of the knowledge expert and the knowledge engineer. It still needs the domain and situational knowledge relevant for this problem. Therefore, the system creates two system agents “KM1” and “KM2” and assigns them to the virtual team “VKC_1”, where “KM1” is responsible for finding related domain knowledge in the system repositories and “KM2” answers for gathering evidences according to

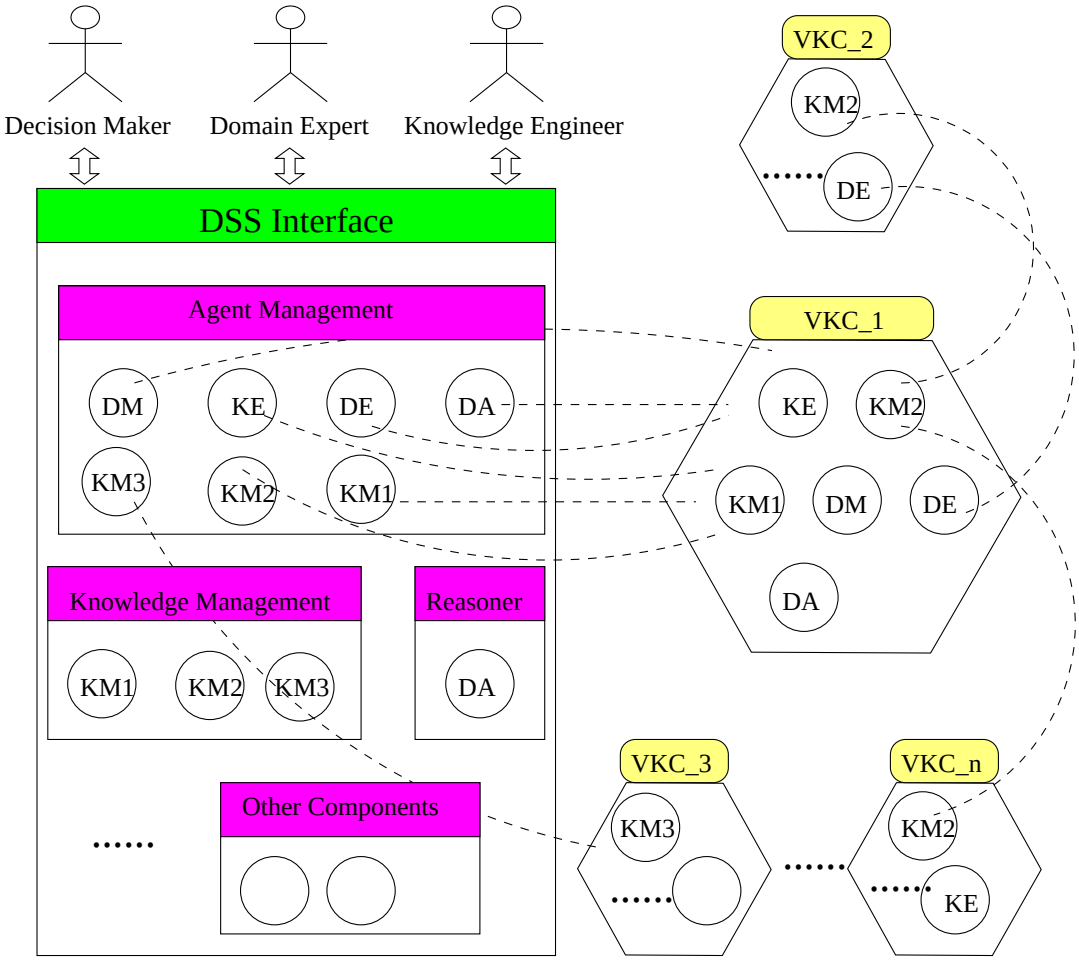

Figure 5.12: Agents collaborate with each other via VKCs in a simple DSS.

the problem. "KM1" and "KM2" join the virtual team "VKC_1" and together with the human user agents they can build a formal decision model for solving this problem. According to the decision analysis process the team still needs an additional agent "DA" for decision analysis. "DA" is created by "Agent Management" and possesses the reasoning ability for the given decision model. Now the team is complete. Each member has its special functionality and takes a unit of task which is decomposed from the original decision problem.

All members of the virtual team "VKC_1" can work together collaboratively without regards to the physical distance and organizational boundaries. For example the decision maker "DM " is an European who wants to buy a house near the "HuangPu" river in Shanghai. He must decide whether a flood insurance is necessary for him, and when necessary who can provide such products and which one he should buy. Therefore it asks the DSS for decision making support. The system

assigns “DM” two helpers from Europe, “DE” and “KE”, due to the confidence problem (he may rely on an European expert more than a local one). Additionally an agent “KM3” answers for the product query from “DM”. Together with other agents as mentioned above, a virtual team to support “DM” is established. They don’t need face-to-face communication any more.

It is pointed out in Figure 5.12 that an agent can be involved in many different communities, according the principle of VKCs. For example, the agent “KM2” can participate in both “VKC_2” and “VKC_n” at the same time, in order to gather more evidences through knowledge sharing in VKCs.

Adaptivity from the Perspective of KM

As discussed in Section 3.1.3 one challenge for DSS is the *adaptivity*, which has two perspectives: one is the knowledge management and the other is the system engineering. VKCs can be used to deal with the former, whereas the latter can be accomplished through the implementation of the system based on SOA (see Chapter 6).

Most systems will provide the feature of adaptivity in a centralized approach — the monitoring of a system landscape. On the contrary systems layered with VKCs have a distributed approach to provide this feature. Agents in VKCs are distributed. They can selectively join different communities based on their own interests at the same time. They can work more proactively, because they don’t wait for the report of knowledge changing from any central system component like a monitor. They can update their knowledge and also sharing their new observations about knowledge of the system environment with other agents by actively participating into different communities. Changes can be delivered to the right community, for the right agents and at the right time, based on different domain of interest. Then agents can adapt their decision models to given decision problems.

For example, in a disaster management DSS there are some agents connected to sensors for gathering information about water gauge. They will directly deliver the changes to an assigned controller who can decide whether to open the water gate or to keep it closed. But, they can also share this change with other agents through VKCs. They can create a new VKC to give other agents an opportunity to join the VKC and obtain the changes. Or they can selectively join some communities where there are already agents having interest in the knowledge about changing water gauge.

It’s quite the opposite in a monitoring system where agents behave more reactively, because they can and must update their knowledge when the system monitor observes something has either change or is new.

Chapter 6

Implementation and Test

This chapter aims to design and implement a DSS based on the framework proposed in Chapter 3 and to test the feasibility of the DSS with some evaluation scenarios. In the first section an architecture is designed for guiding the implementation. The next section is devoted to give an overview of the implementation from the general technical perspective. In the last section the DSS will be evaluated with emphasis on the main objectives of this work — decision making with uncertain knowledge, (formal) knowledge integration in OntoBayes, and knowledge sharing through VKCs.

6.1 A System Architecture

According to the layered framework illustrated in Figure 3.3 a service oriented architecture is designed for DSSs. There are four colors: cyan, magenta, yellow, and green. As depicted in Figure 6.1 this architecture has many components and each of them can be assigned to a layer of the framework according to the color representing it. Components in cyan, magenta, yellow, and green belong to the repository, management, collaboration and application layer respectively. In the following subsections the architecture will be described in a bottom up approach.

The Repository Layer

In the lowest layer, the repository layer, there are at least two kinds of repositories: a service repository and an OntoBayes repository.

- The *Service repository* is used to store service descriptions. All services published by the DSS can be retrieved from this repository.
- The *OntoBayes repository* is in fact the public knowledge and decision model storage of the system. In the OntoBayes model there are three kinds

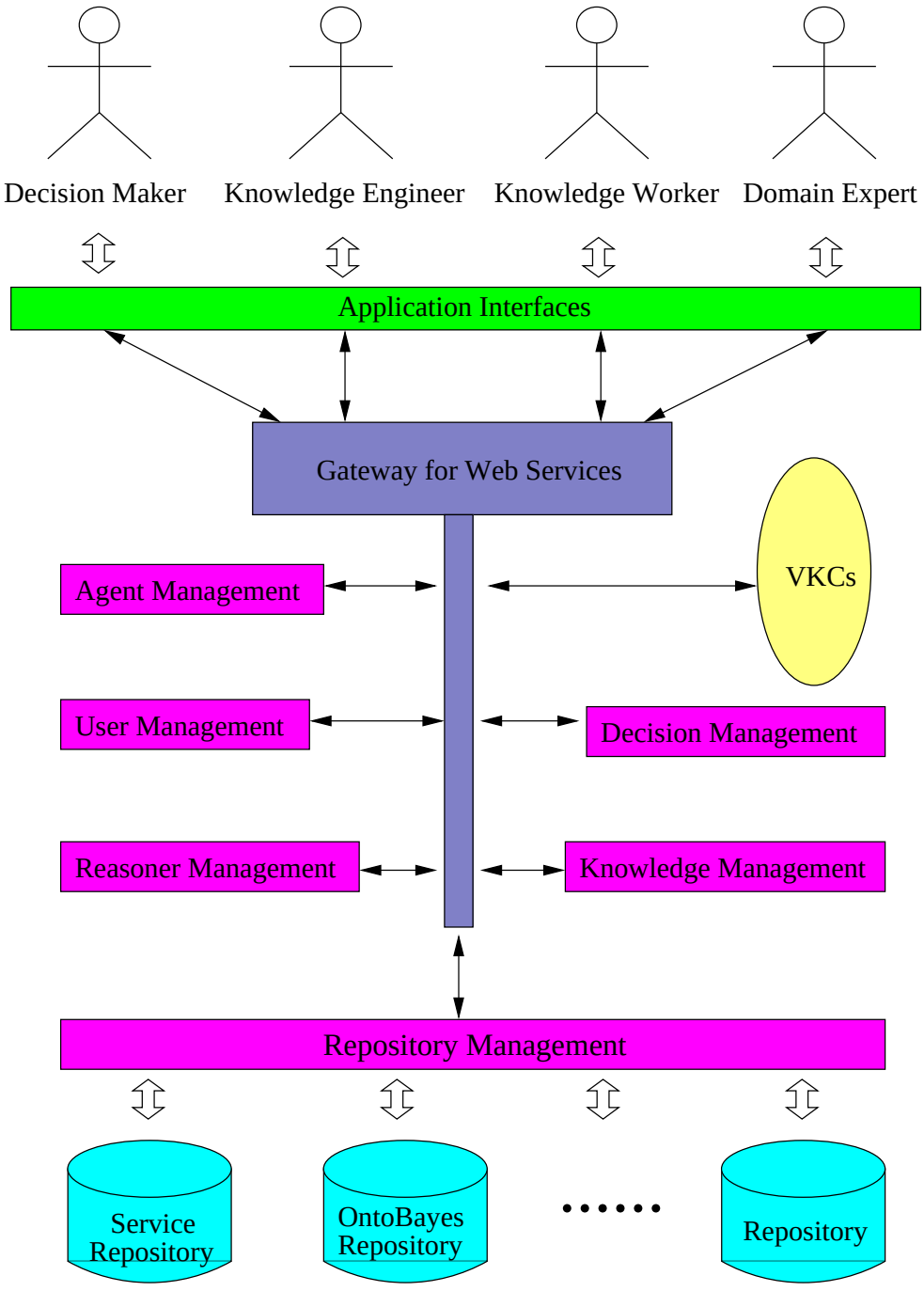

Figure 6.1: An architecture according to the layered framework (see Figure 3.3) for DSSs.

of OWL files: the ontological knowledge, the Bayesian knowledge and decision models. They can be distinguished with the upper ontologies described in Chapter 4.

- Other repositories can be built for the system or for users on demand. For example repositories for storing policy, business processes and so on.

The Management Component Layer

In the management layer there are several types of management components and each of them has its own functionalities and provides different services.

- A *repository management* provides basic services to manage all kinds of repositories, e.g. operations for access, recovery and long term preservation of repositories. These operations are very basic such as *read, write, modify* etc., but are not able to interpret the content in the repositories. Advanced operations like these are provided by the knowledge management component. For example, operations for recognizing the contents of a file in a repository will be provided by knowledge management, not in this layer.

- A *decision management* takes responsibility for managing all activities directly related to decision making. These activities are for instance to manage different decision making processes, to distinguish different decision support levels (simple query vs. decision analysis) and so on. The decision analysis process described in Section 3.2.4 will be implemented in this component.

- A *knowledge management* should provide services to manage the knowledge life cycle: knowledge generation, knowledge codification, knowledge transfer (also known as knowledge sharing), and knowledge application [AL01]. We take only knowledge generation and codification into account in this system component, because in this work knowledge transfer is theoretically designed based on VKCs as discussed in Chapter 5 and VKCs build the collaboration layer of the system.

  For knowledge generation operations such as the acquisition, synthesis, and creation of knowledge are required. In fact all knowledge is stored in different repositories in format of certain files, e.g. OWL-file and so on. Therefore we need tools to recognize the content of these files and to index them, as well as to query them.

  For knowledge codification tools to represent knowledge by converting it into formal formats are required, because only the formal representation can

make knowledge easily accessible and transferable. In this thesis knowledge codification is theoretically based on the work of the integrated model *OntoBayes* as introduced in Chapter 4. The introduced underlying formal representation language for OntoBayes is OWL, therefore a tool must be able to edit and codify knowledge (ontologies, BNs and IDs) into OWL files.

- A *reasoner management* has the responsibility for assigning algorithms for reasoning about a BN or evaluating an optimal policy of an ID. Building a reasoner management component in a DSS separately is useful if there are many algorithms available for solving a decision problem. For example agents can make use of an exact reasoning to query a Bayesian variable, but it can also make use of an approximate reasoning to do this work.

  To enable the DSS work, at least two algorithms must be managed by this component: one for reasoning about BNs and the other for evaluating IDs.

- A *user management* provides authentication and authorization services for users. Each user has a different role in the system and has also different rights. For example, there are knowledge workers, domain experts, knowledge engineers, administrators, decision makers (i.e. the end users) and so on. Users may consume services provided by the system only when they are granted permission.

- An *agent management* takes responsibility to manage all agents. For example, to create agents, to terminate agents and so on. There are two kinds of agents running in the system: system agents and user agents. The former run on each management component and have its functionalities. They are created by the system. The latter are created by users with assigned tasks.

The Collaboration Layer

In the collaboration layer there are VKCs which provide a virtual place for agents to meet and work together. As discussed in Chapter 5 two main functionalities will be provided in this layer: corporate knowledge support for decision making through knowledge exchange and collaborative working of virtual teams.

The Application Layer

The front end of the system, the application layer, provides user interfaces based on web services to support decision problem solving. Decision support in the application front end will be realized through the utilization of web services provided by the system back end.

In fact each user interface is a service consumer and the system back end is thus a service provider. As depicted in Figure 6.1 the service gateway connects the back end and the application front end of the DSS. It enables a logical integration of service consumers and providers and a logical connection between the back and front ends of the system.

Repositories, all management components as well as VKCs behind the system are connected to the gateway through a service bus. Services that they provided can be published in the service repository and consumed by users via the gateway. Applications can be constructed dynamically and flexibly by selecting services on demand and composing them into a executable process.

6.2 Implementation

In accord with the architecture illustrated above a prototype of a DSS is mainly implemented within the interrelated works of Guillou [Gui07] and Hering [Her07]. The prototype is developed with the programming language Java. In this section we only give a simple overview of the implementation, not the details, because it is not possible to describe the concrete implementation work of the whole system in details within the limited place in this thesis. More details such as UML (*Unified Modeling Language*) diagrams can be found under the web link `http://iaks-www.ira.uka.de/home/yiyang/DSS/UMLs` as well as in [Gui07] and [Her07].

In order to facilitate the implementation, we make use of many existing tools and API (*Application Programming Interface*). Most of them are open source, but there are also commercial softwares. In the following subsections we will describe what are implemented in the prototype and with which techniques (open sources or commercial softwares).

6.2.1 An Application Interface

According to the architecture user interfaces in the application layer are required to facilitate the interaction between users and the DSS. In the prototype a GUI (*Graphical User Interface*) is implemented based on the technique of *Java Swing*. As depicted in Figure 6.2 the GUI consists of five tabs: the main tab, the repository management tab, the VKC tab, the user management tab and the agent management tab.

Among them the main tab is the main user interface for decision making support. This tab is set as the default display of the GUI. In this tab the following operations are implemented:

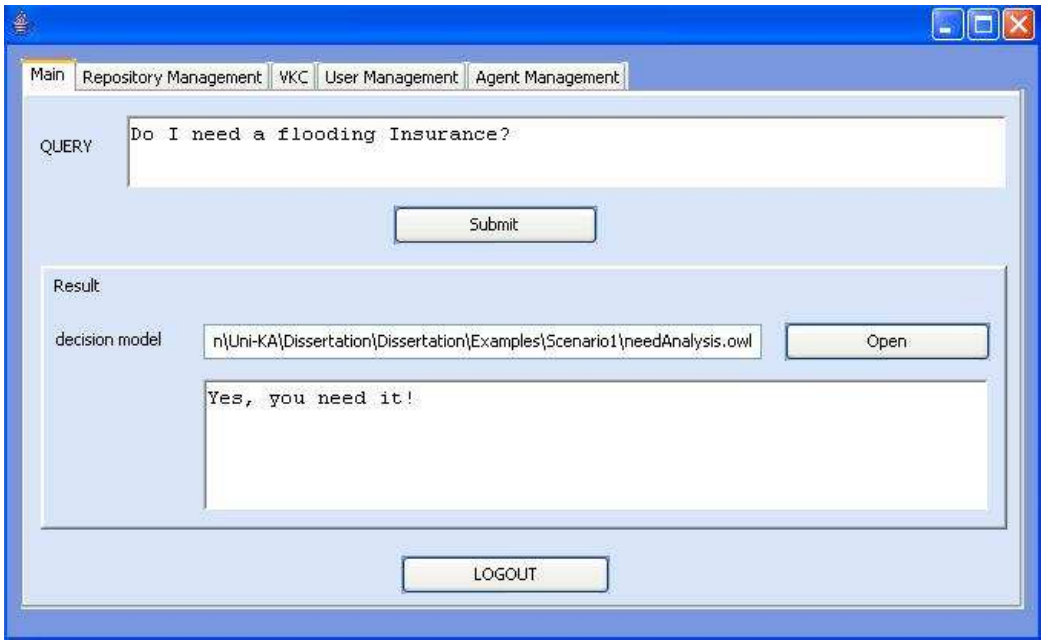

Figure 6.2: The main tab of the GUI.

- Operation for submitting queries. There is a query panel. A user can type a query in this panel and submit it to the DSS.

- Operation for displaying query results. There is a result panel. An input query will be processed in the system back end and its answer will be displayed in the result panel.

- Operation for opening an OWL file with Protégé. When the submitted query is a decision problem which needs high level decision support, besides the query results, a decision model used to solve the problem is returned. And users can explore the decision model with the Protégé and a plugin *OWLOntoBayes*.

Other tabs of the GUI will be introduced in the next subsection.

6.2.2 Management Components

In this subsection we will survey the implementation of different management components of the DSS.

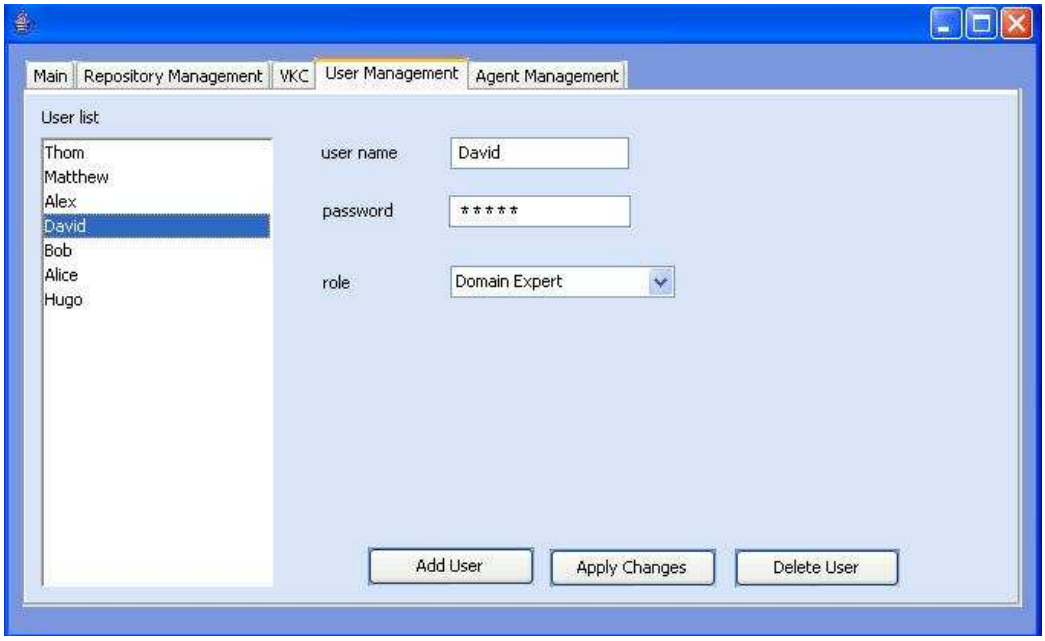

Figure 6.3: The user management tab of the GUI.

User Management

In the user management component operations for authentication and authorization are implemented on JAAS (*Java Authentication and Authorization Service*) 1.0 which is a set of Java packages for providing services to authenticate and enforcing access controls upon users.

In this component all users of the DSS are managed and controlled. A user can log in the system only when his login information (the username and the password) can be authenticated here. After the successful authentication his access right will be loaded through an authorization service of this component, according to the predefined user profile.

In Figure 6.3 the user management tab is illustrated. In this tab the following operations are implemented:

- Operations for assigning roles to a user. Each user has a role, for example end user, domain expert, and administrator and so on. Different roles are assigned with different access rights. The access right of each user determines which operations are allowed for him in the DSS. For example an end user can view public repositories, but he is not allowed to modify them, because such operations belong only to an administrator.

- Operations for modifying user information such as password and so on.

- Operations for adding and deleting users. These operations are only allowed for system administrators.

Repository Management

The repository management component is implemented based in the open source content repository *Apache Jackrabbit*[1] *1.0*. Jackrabbit 1.0 is an implementation of the JCR (*Java Content Repository*) API in accord with the JSR (*Java Specification Request*) 170. JSR 170 specifies standard APIs to access content repositories in a uniform manner with the Java technology. Following basic operations are implemented:

- Operations for creating or deleting a repository.
- Operations for opening (or viewing) a repository.
- Operations for adding or deleting a file or a folder in a repository.

In Figure 6.4 the repository management tab of the GUI is illustrated. This tab contains sub-tabs. The most left sub-tab, “repositories”, gives an overview of all repositories of the DSS. According to the architecture there are at least two repositories in the system: a service repository and an OntoBayes repository. All web services files are stored in the service repository. And all OWL files are stored in the OntoBayes repository. These OWL files are used to model the ontological knowledge, Bayesian knowledge and decision models (i.e. IDs). Therefore they are categorized into three folders: ontology, BN and ID. The detailed information about each repository can be displayed in a separated sub-tab.

As illustrated in the figure users can create their own repositories for their own purposes. For example, there is a repository called ”Bob’s repository” and it stores Bob’s private content. Only Bob can access and modify this repository and other users have no right to access it, except the administrators.

Reasoner Management

In the reasoner management component there are only two algorithms used for the system: one for BNs and the other for IDs. The former is an exact algorithm, the clustering algorithm [Pea97], for reasoning about BNs, whereas the latter is for evaluating IDs and was abstractly described in Section 2.4 (see Page 40).

The implementation of these algorithms are provided by Netica-J[2]. Netica-J is the Java version of Netica API which allows to build BNs and IDs, to do

[1] http://jackrabbit.apache.org/

[2] http://www.norsys.com/netica-j.html

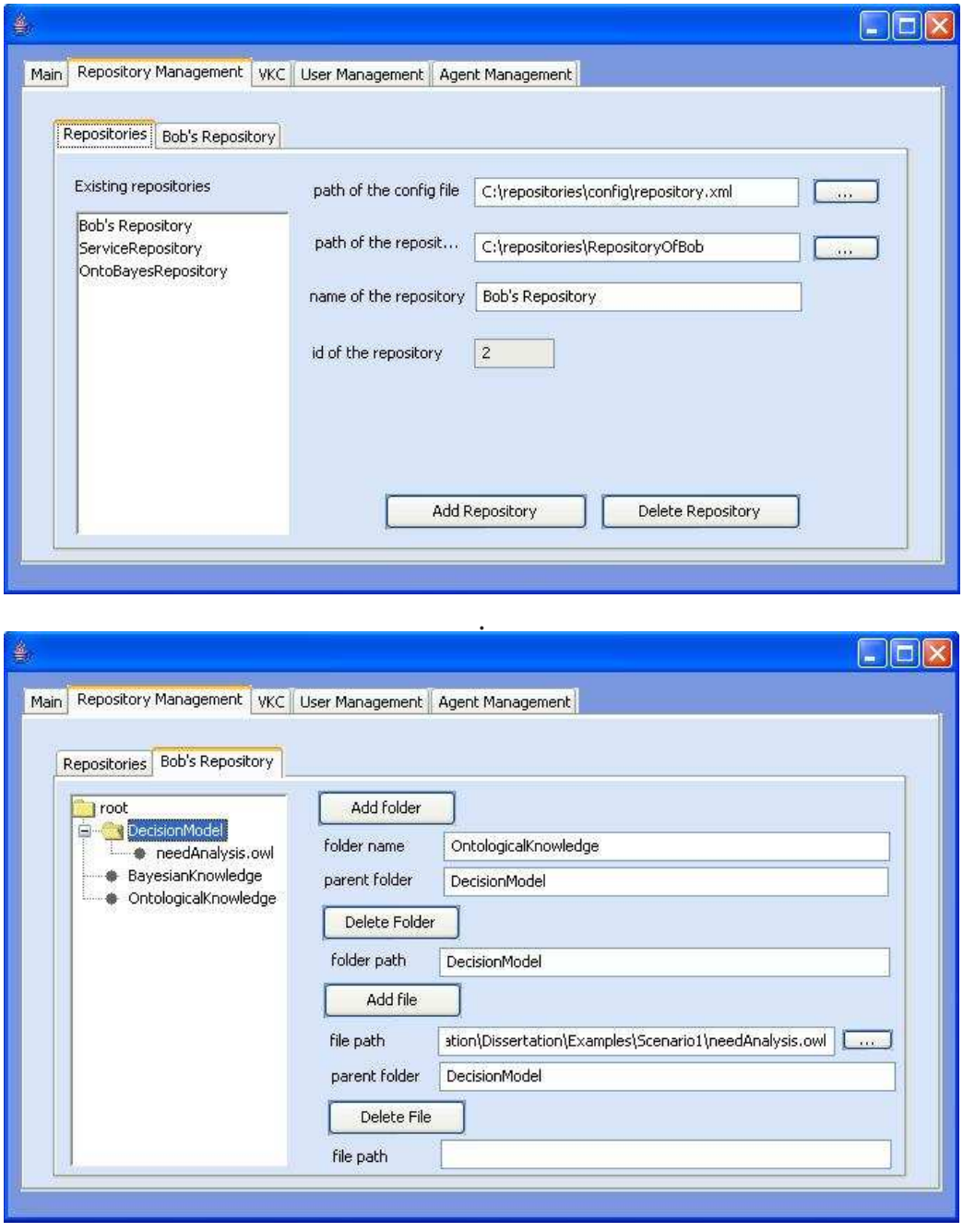

Figure 6.4: The repository management tab of the GUI.

probabilistic inference and to evaluate optimal policy. It is a commercial product provided by Norsys Software Corporation. For using algorithms provided by Netica-J we implement a bridge which can set connections between OntoBayes and Netica. The OWL files in OntoBayes representing BNs and IDs will be converted into Bayes nets and decision nets in the format of Netica, respectively. Every reasoning request for OntoBayes will be sent to Netica through the bridge, and the results will be returned in the same way.

In this prototype the main purpose implementing this component is not to provide many selective and comparable algorithms for BNs and IDs, but to provide the minimal set of algorithms that can guarantee the reasoning ability of the system. From this perspective one algorithm for each method is enough to test the feasibility of the system. Further functionalities of this component (as mentioned in Section 6.1) will be complemented in future works.

Agent Management

The implementation of the agent management component is based on JADE (*Java Agent DEvelopment Framework*)[3] 3.4. JADE provides an agent platform which simplifies the implementation of all basic specifications of FIPA (*Foundation for Intelligent Physical Agents*) compliant MAS. In this component the following operations are implemented:

- Two classes of agents: user agents and system agents. User agents represent users to execute operations for them in the system, because the whole system is agent-based and all communications in the system are between agents, not between users and system components directly. When a user logs in the system successfully, a corresponding user agent will be automatically created in the DSS. This agent will take over the user profile and can execute operations assigned either by the user or by the system. For example, to join a VKC or all VKCs for finding useful information.

  System agents are created for different components and assigned with predefined tasks, operations and services. The DSS can create as many agents as it needs, depending on the system burden. If there are many service requests to one component, the component can create many agents to take over the requests concurrently.

- Operations for assigning roles to an agent. According to the services provided by different system components, agents will be assigned different roles similar to user's ones.

[3] http://jade.tilab.com/

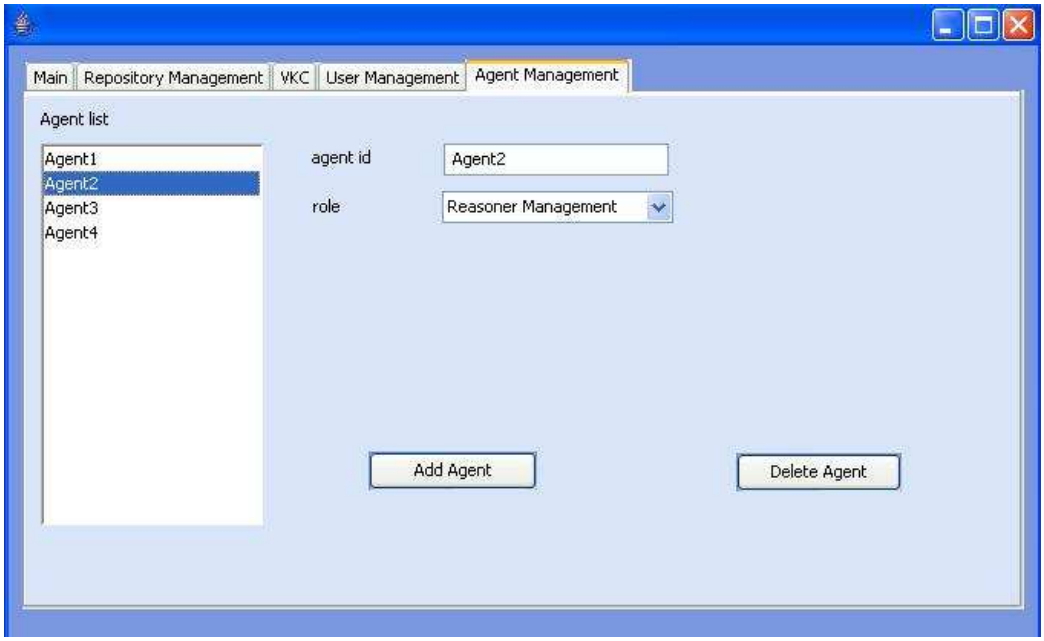

Figure 6.5: The agent management tab of the GUI.

- Operations for creating or killing agents. For avoiding to waste the system resource and to make the system efficient, the agent management component needs to suppress agents in time after they do finish their task.
- Operations for monitoring all agents. All agents are tracked by the system. JADE provides a centralized approach to track them, the central directory with a unique identifier for each agent.

A user interface is implemented for facilitating the management of agents. It is illustrated in Figure 6.5.

Knowledge Management

As mentioned in the last section only knowledge generation and codification are implemented in the knowledge management component, because knowledge transfer will be implemented in the collaboration layer through VKCs.

For knowledge generation an index engine and a query engine are implemented in this prototype. They can help agents search domain specific knowledge (and decision models) in the repositories. The index engine can index all OWL files with meaning tags, in order to make the retrieval of these files easily and quickly. And the query engine is used to answer the ontological queries, as well as to find situational knowledge — evidences of domain specific BNs and

IDs. The indexes are hardcoded and the query engine is implemented based on the Jena SPARQL[4] query engine. SPARQL is a query language for RDF and is a standard of W3C. The syntax of SPARQL is SQL-like and easy to use and understand.

For knowledge codification a Protégé plugin, OWLOntoBayes, is implemented in this prototype. The implementation of this plugin is based on the Protégé API and Jena OWL API. This plugin enables to edit and codify BNs and IDs into OWL files with a GUI. The way to codify them in OWL was introduced in Chapter 4.

Figure 6.6 and Figure 6.7 illustrate the GUI of the plugin. There are many functionalities implemented in this plugin. Among them the most important are the following ones:

- Operations for creating or deleting chance nodes. As described in Section 4.2 and Section 4.3 the basement of BNs and IDs in OntoBayes are ontologies. The properties of an ontology are used to create chance nodes and decision nodes. As shown in Figure 6.6 and Figure 6.7 the properties are listed in the left panel. They can be moved to the right panel and displayed as nodes.

- Operations for creating or deleting decision nodes. We can construct an ID based on a given BN. We can get a menu of operations. One important operation is allowing to convert a chance node into a decision node.

- Operations for creating or deleting utility nodes. These nodes can be directly created by using a similar menu as mentioned above.

- Operations for creating or deleting arrows between nodes. The types of an arrow can be identified automatically depending on the type of nodes which the arrow goes into.

- Operations for editing states of a chance node or a decision node.

- Operations for editing probability tables of a chance node.

- Operations for editing utility tables of a utility node.

We select the variable `FloodingInsurance.Premium` as an example to test the operations for editing states and probability tables. In Figure 6.8 there are two panels. The left one is for editing states and the right one is for probabilities. Before a probability table can be specified for a Bayesian variable, the state space of

[4] http://www.w3.org/TR/rdf-sparql-query/

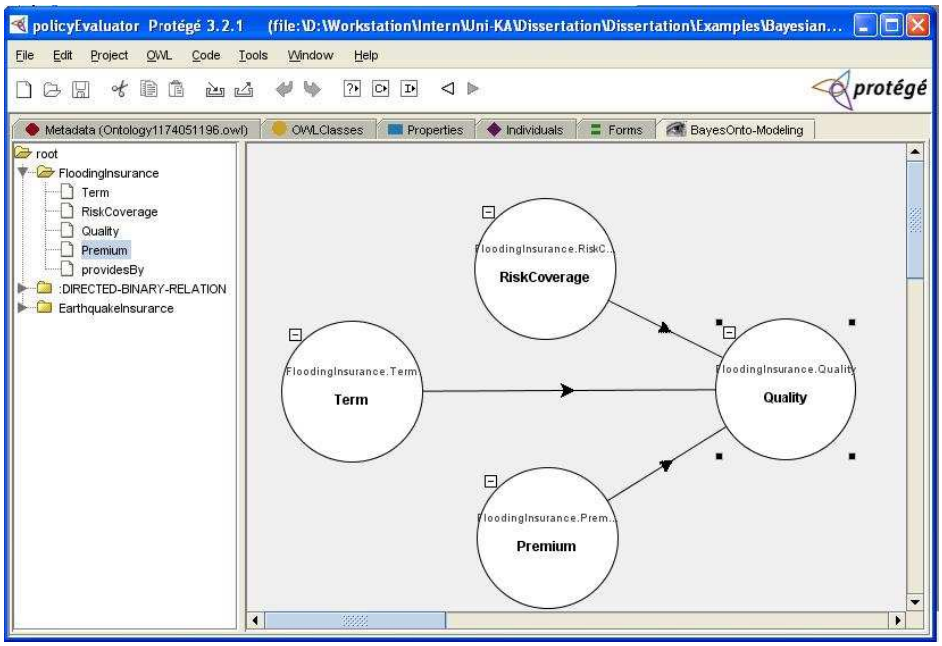

Figure 6.6: The Bayesian knowledge for evaluating insurance policy.

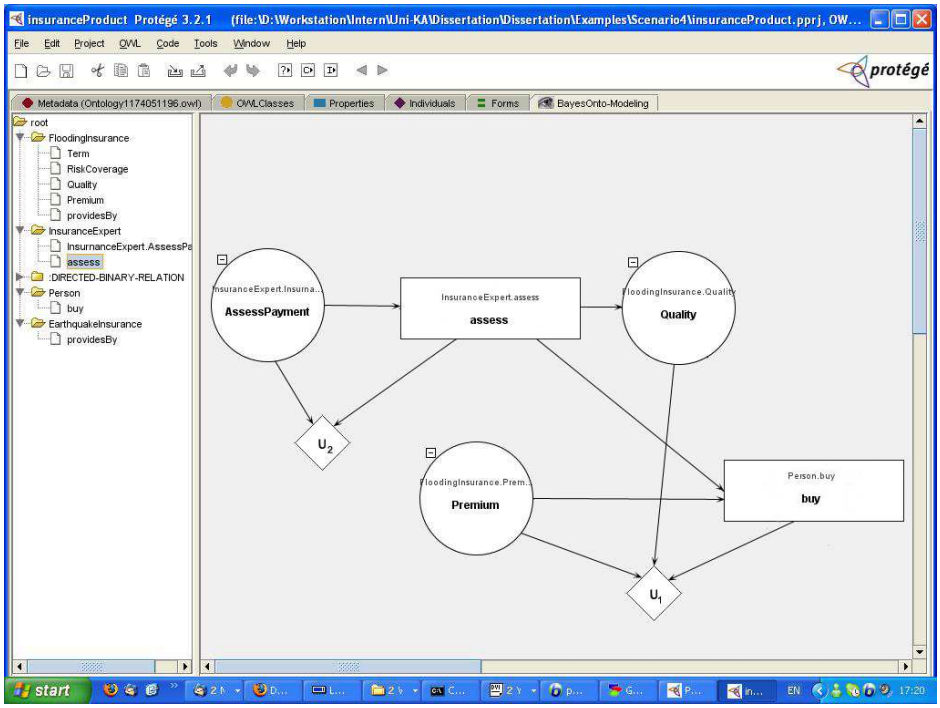

Figure 6.7: The decision model for buy a flooding insurance.

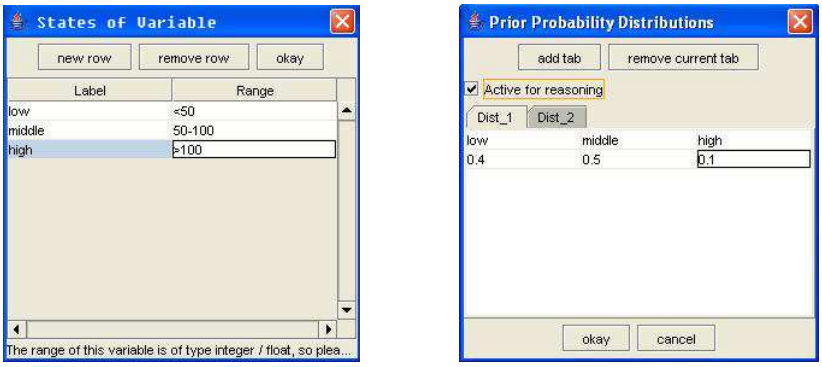

Figure 6.8: Construct the state of the variable and its probability table

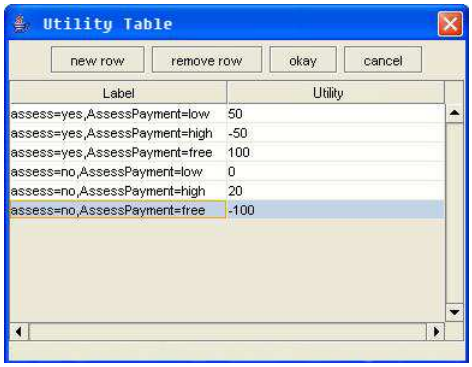

Figure 6.9: Construct the utility table of a utility node.

the variable must be defined first. Due to the fact that we only take discrete variables into account, in the left panel we can define the variable `FloodingInsurance.Premium` has three discrete states {`low,middle,high`}. For making applications more practical, we can define each state with intervals of real number. For example the premium has the state `low` when it is lower than 50 Euro per month.

After the states of a variable are completely defined, we can specify its probability tables. The probability table of the variable `FloodingInsurance.Premium` is specified in the right panel. As shown in the panel, we can specify many probability tables for this variable, but only one is active when reasoning about a BN. The active table can be selected via the check box shown above. The utility table of a utility node can be constructed similarly. Figure 6.9 presents the utility table of the utility node `U2` in Figure 6.7.

As shown above nodes in BNs or in IDs can be easily constructed via the plugin. At the same time they can be encoded in OWL files according to the upper ontologies defined in Chapter 4. The plugin is the main tool for them to construct the knowledge and decision model bases of the DSS. The main user

groups of this plugin are knowledge engineers and domain experts. But end users can also make use of them to view decision models for having an insight into the underlying mechanism for decision support.

Decision Management

The following functionalities are implemented in the decision management component in the prototype:

- Recognition of two different levels of decision support according to Figure 3.4. An interpreter is implemented to analyze the input message from users and distinguish whether it is a service request for the low level or for high level decision support.

- Construction of the decision analysis process depicted in Figure 3.5. In the case that a user requests a high level support, the decision analysis process will be involved. A set of agents will be created for this process. They have different unit of tasks: basis development, analysis, recommendation and basis refinement. For the basis development the decision context will be analyzed by an interpreter (agent), according to the decision problem. Two agents are created by the knowledge management component: decision model identifier and situational knowledge identifier. The former will identify a necessary decision model in the OntoBayes repository. When a decision model is identified, the latter will query the whole OntoBayes repository for finding situational knowledge — evidences. For analysis the decision model with given evidences, an agent created by the reasoner management component will assigned with an adequate algorithm to the process. After that results will be delivered to the user by the recommendation agent. It will also monitor the action that the user will take to refine the basis.

In the first prototype this component is implemented with a hardcoded solution, because a full automatic solution can only be guaranteed by invoking techniques such as natural language processing etc.. This is but beyond the scope of this doctoral work.

6.2.3 Virtual Knowledge Communities

The implementation of VKCs in this prototype is a major extension of the works of Calmet et al. [CME04, MC04, Ham04]. The following works are implemented to extend VKCs for archiving the full integration of VKCs and OntoBayes.

- Extension of normalized ontology into OntoBayes model. Each normalized ontology of the old work contains only one head of the community cluster. But to be compatible with the OntoBayes model there can be at most three heads in each community buffer: one for ontological knowledge, one for Bayesian knowledge and the other for decision models. Therefore the normalized ontology are extended to work with three heads.

- Extension of knowledge cluster and knowledge instances into OntoBayes model. The difference of knowledge cluster and knowledge instances between ontological knowledge, Bayesian knowledge and decision models are shown in Table 5.1 and Table 5.2. Their existing implementation can only work with ontological knowledge, but not for others. Therefore they are extended to cover all differences, e.g. extensions of Bayesian instances and IDs' instances etc..

- DAG check. The precondition for a legal BN or a legal ID is that each of them must be a DAG. After the Bayesian knowledge or decision model exchange the system must check whether the resulted new BN or ID is still a DAG.

- OWL compatibility. The underlying formal representation language for OntoBayes is OWL, but the previous implementation of VKCs is based on RDF. In order to integrate VKCs with OntoBayes, VKCs must be compatible with OWL. But we can not just translate or convert OWL files into RDF files. Such a solution will violate the semantic meaning captured in an OWL file, because OWL is built upon RDF and has more descriptive power. Therefore we decide to improve the implementation of VKCs to reach the OWL compatibility.

  This part is implemented by using APIs of AgentOWL[5], Protégé and Jena[6]. Jena is a semantic web framework for Java application development. The exchangeable knowledge and decision models in OntoBayes are implemented in the Java API `JenaOWLModel`. Additionally two parsers are implemented for the semantic restrictions mentioned in Chapter 5: one for OWL-based probability tables and the other for OWL-based utility tables. They are used to parse the string expression of each cell of the tables and to ensure that a table is only exchangeable when it is complete (no cells are ignored during the exchange process).

- A GUI for the management of VKCs. As illustrated in Figure 6.10 some basic operations of VKCs are available in the GUI. For example a user agent

[5]http://agentowl.sourceforge.net/
[6]http://jena.sourceforge.net/

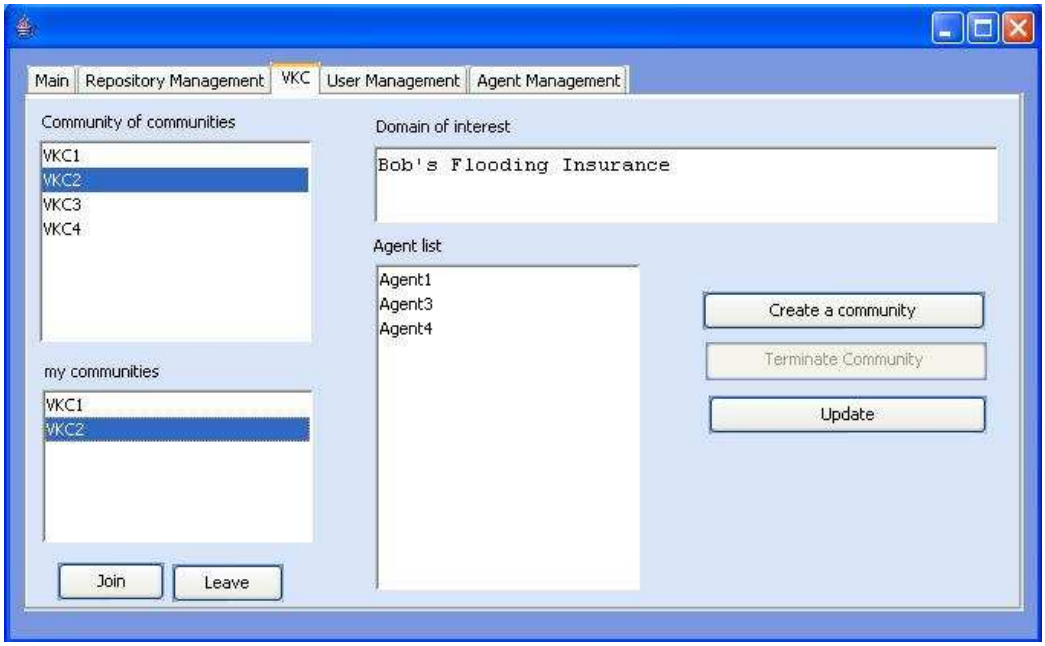

Figure 6.10: The VKC management tab of the GUI.

can create a new VKC or join an existing one. It can terminate a VKC which belongs to itself, i.e. it is the community leader of the VKC.

6.2.4 The Web Service Gateway

The web service gateway of the DSS is implemented as a J2EE Web Server with the help of *Apache Tomcat 6.0*[7] and *Apache Axis2*[8]. Tomcat is a Java Servlet container for implementing *Java Servlet* and JSP (*JavaServer Pages*), which are developed by Sun Microsystems aiming to make the web-based applications possible, simple, dynamical, rapid and platform-independent. Axis is a SOAP (*Simple Object Access Protocol*)[9] engine server which plugs into Tomcat and has additional supports for WSDL (*Web Services Description Language*)[10]. SOAP is an XML-based communication protocol started by W3C and is designed for data interchange between systems and for execution of RPC (*Remote Process Call*). Initially SOAP was entirely based on HTTP, but now it has been extended to a generic conveyor of information on top of a variety of transport protocols [ACKM04].

A WSDL document defines services as collections of network endpoints, or

[7] http://tomcat.apache.org/index.html
[8] http://ws.apache.org/axis/index.html
[9] http://www.w3.org/TR/soap/
[10] http://www.w3.org/TR/wsdl

ports. Each WSDL specification consists of two parts: an abstract part and a concrete part. The former contains abstract information about defined data types, a unit of communication, an action supported by the service and a collection of operations. The latter defines the concrete protocol binding and all related endpoints, where each endpoint is defined as a combination of a binding and a network address [ACKM04].

The use of WSDL is simplified through the help of Axis, because Axis provides two commands: `java2WSDL` and `WSDL2java`. The first one can be used for building WSDL from Java classes, and the second one for building Java APIs and deployment descriptor templates in accord with web services specified in WSDL. These methods together make it easier to develop the DSS in a SOA approach.

The DSS can build all services based on the implemented Java APIs through the commond `java2WSDL`. These services will be published in the service repository and can be used for applications. In our system the application interface is also implemented in Java. In fact it is a service consumer (or client). It sends service requests via SOAP to the gateway of the system. The system as a service provider will analyze the requests and return the corresponding services back. After getting the services, the applications can convert these services by using `WSDL2java` into Java classes which can be directly used for applications.

For example to use the main GUI as shown above, the user “Bob” is required to log in first. He inputs his user name and his password in the login panel and then the GUI (the client) sends an authentication service request with the given login information (encoded in a SOAP message) to the system. In the gateway of the system this SOAP message will be decoded and analyzed, then delivered to a user management agent. The agent identifies Bob with his password successfully and returns a confirmation service to the client. Then “Bob” is allowed to log in to the system and to use other functionalities provided by the system.

6.3 Test

In this section we only test the feasibility of the prototype of the DSS. A use case is constructed according to the decision problems described in Example 1.1 (see page 2), in order to facilitate the test.

The whole use case will be demonstrated in five scenarios according to the sequential steps that Bob will go through to solve these problems by using the DSS. Based on the analysis of these scenarios we can test the main objectives of the framework proposed in Chapter 3 — decision making with uncertain knowledge, (formal) knowledge integration in OntoBayes, and knowledge sharing through VKCs.

- Decision making with uncertain knowledge. This is a common feature of

most decision analytical systems, but is very important and basic. In this use case, there are many uncertain factors having influence on decision making: the risk of flooding, the relation between the financial status of a customer and the premium of an insurance product etc.. Therefore decision analytical systems are widely used for solving these problems. Scenario 1, 3, 4 and 5 can be used to test that our DSS is able to make optimal decisions based on given IDs with predefined probability tables and utility tables.

In fact for this objective, a wide variety of BN-based tools are available, e.g. Hugin, Netica and so on [Mur02]. The usual steps are: 1) build or find a decision model (an ID); 2) find (or set) evidences; 3) evaluate an optimal policy. Among these steps the first two are the most important, because the last step is mathematically calculable and the result is unique when the decision model and evidences related to the model are given. Our DSS also follows these steps for making decision under uncertainty. But the way to deal with the first two steps in our DSS is unique and differs from others, because of the next two points.

- Knowledge integration in OntoBayes. This is a unique feature of our DSS compared to other DSSs. Most existing tools provide GUI for building BNs or IDs and specifying them in a certain syntactical format, such as XML etc.. With the implemented Protégé plugin *OWLOntoBayes* we can also easily edit BNs and IDs, not only their structures but also the numerical information, e.g. probability and utility tables. And additionally we can encode them in OWL files formally, i.e. not only at the syntactical level, but also at the semantical level, as introduced in the OntoBayes model (see Chapter 4).

  In the OntoBayes model BNs and IDs are not just represented in an ontological way (in OWL), but incorporated into ontologies. This integration provides coherent connections between ontologies, BNs and IDs. The connections are implemented through the way of constructing chance nodes and decision nodes. As described in Chapter 4, these nodes are properties of ontologies. They indicate the required context information for given decision problems. And this context information can be identified via ontology reasoning or queries. This incorporation enables to find evidences for BNs and IDs automatically, not manually. We will show this feature in all five scenarios.

- Knowledge sharing through VKCs. This is another unique feature of our DSS compared to other DSSs. In Scenario 4 and 5 we will show how to facilitate knowledge sharing via VKCs, to utilize VKCs for establishing virtual teams related to given decision problems, and how to build or complete

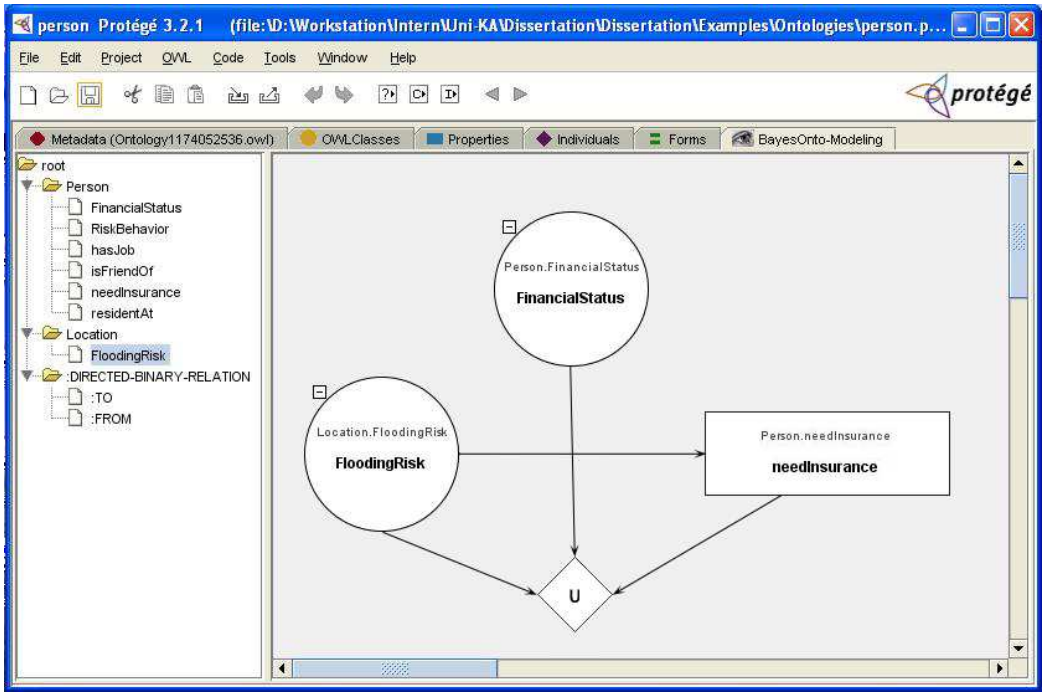

Figure 6.11: The decision model for insurance need analysis.

decision models by sharing knowledge through VKCs.

Our DSS has low level decision support — query to the knowledge base. Scenario 2 will show this feature by using the formal ontological query language SPARQL based on the knowledge base (ontologies) of the system.

6.3.1 Scenario 1

The first problem faced by Bob is that he must find out whether he needs a flooding insurance at all. He inputs a message in the query panel: “Do I need a flooding insurance?”.

The DSS finds a decision model (an ID) `needAnalysis.owl` in the OntoBayes repository which matches this decision problem. This ID contains two chance nodes `Location.FloodingRisk` and `Person.FinancialStatus`, a decision node `need(Person,FloodingInsurance)`, and a utility node `U`. They are illustrated in Figure 6.11. The arrows going to `U` point out that the chance nodes are two decision criteria to evaluate the expected utility. The probability tables of `Location.FloodingRisk` and `Person.FinancialStatus` and the utility table of `U` are specified in the OWL file. These numerical information can be viewed and modified with the GUI illustrated in Figure 6.8 and 6.9, respectively. The

arrow going to `need(Person,FloodingInsurance)` points out that the information about the flooding risk must be known for making this decision.

Now the system is trying to find evidences about these chance nodes. The notations of chance nodes and decision nodes indicate the context information which should be retrieved through ontology queries. According to the user profile provided by user management, the system tries to find the location and financial status of Bob. The following queries are sent to the query engine:

```
SELECT ?Location
WHERE {?Person:Username "Bob" . ?Person:residentAt
                                ?Location}
SELECT ?FinancialStatus
WHERE {?Person:Username "Bob" . ?Person:FinancialStatus
                                ?FinancialStatus}
```

The result of the first query is `Shanghai` and the second query is `no matches found` based on the ontology file `person.owl` in the OntoBayes repository. Now another query is shipped to the engine:

```
SELECT ?FloodingRisk
WHERE {?Location:CityName "Shanghai" . ?Location:FloodingRisk
                                       ?FloodingRisk }
```

Based on another ontology file `riskCard.owl` the result `high` will be returned to the system[11]. Now the whole ID with all numerical information (the tables) and all evidences can be evaluated by using the algorithm assigned by the reasoner management. The expected utility of the decision node `need(Person,Flooding-Insurance)` is computed: `(need=yes,62.5)` and `(need=no, -55)`. The decision can be made based on the highest expected utility `(need=yes,62.5)`.

All of these queries are processed in the system background. Bob will only get a recommendation message in the result panel: “Yes, you need a flooding insurance.” and a URI of the used formal decision model. But he has still the possibility to have an insight into the decision model, when he opens it with Protégé and the plugin *OWLOntoBayes*.

6.3.2 Scenario 2

Now Bob needs to know all potential flooding insurance products that he can choose from. He inputs a message: “Which flooding insurance products are

[11]We simplified the necessary information of a location for identifying the flooding risk in our scenario. In the reality it needs the exact information such as country, city, street and so on. But the simplification has no impact on the quality of this test scenario.

there?". The DSS identifies that it is a low level support request. Therefore this message will be translated into the following SPARQL query by the interpreter of the DSS.

```
SELECT ?PolicyName
WHERE {?FloodingInsurance:PolicyName ?PolicyName }
```

Based on the OWL file `insuranceProduct.owl` in the repository the query engine returns the result: `PolicyA, PolicyB, PolicyC`.

6.3.3 Scenario 3

Now Bob must decide to choose one product among `PolicyA`, `PolicyB`, `PolicyC`. He inputs a new message: "Which one should I buy from the proposed products 'PolicyA', 'PolicyB' and 'PolicyC'?" The system identifies that it is a high level support request. It means that a decision model is needed for this decision problem. There is a simple decision model `personalBuyModel.owl` in the repository of Bob which can be used to analyze this problem.

This decision model is illustrated in Figure 6.12 as an ID. It has similar structure to the one depicted in Figure 6.11. But the decision node in this ID is `buy(Person,FloodingInsurance)` and the chance nodes are `FloodingInsurance.Quality` and `FloodingInsurance.Premium`. These two chance nodes are the decision criteria of the utility node `U`. To evaluate this ID we need evidences. The following queries are used to find evidences of the chance nodes in the ID for the given insurance product `PolicyA`.

```
SELECT ?p
WHERE {?FloodingInsurance:Premium ?p .
                        ?FloodingInsurance:policyName "PolicyA"}
SELECT ?p
WHERE {?FloodingInsurance:Quality ?p .
                        ?FloodingInsurance:policyName "PolicyA"}
```

The result of the first query is `middle` and of the second query is `no matches found` according to the ontology file `insuranceProduct.owl` in the OntoBayes repository. Similar queries can be constructed for `PolicyB` and `PolicyC`. The results can be simplified as follows:

```
PolicyB.Premium=low,  PolicyB.Quality=no matches found
PolicyC.Premium=high, PolicyC.Quality=no matches found
```

Now the ID `personalBuyModel.owl` has all information that it needs and can calculate the expected utility of the decision node `buy(Person,FloodingInsurance)` according to the specified utilities of `U`. The expected utility for `buy(Bob,`

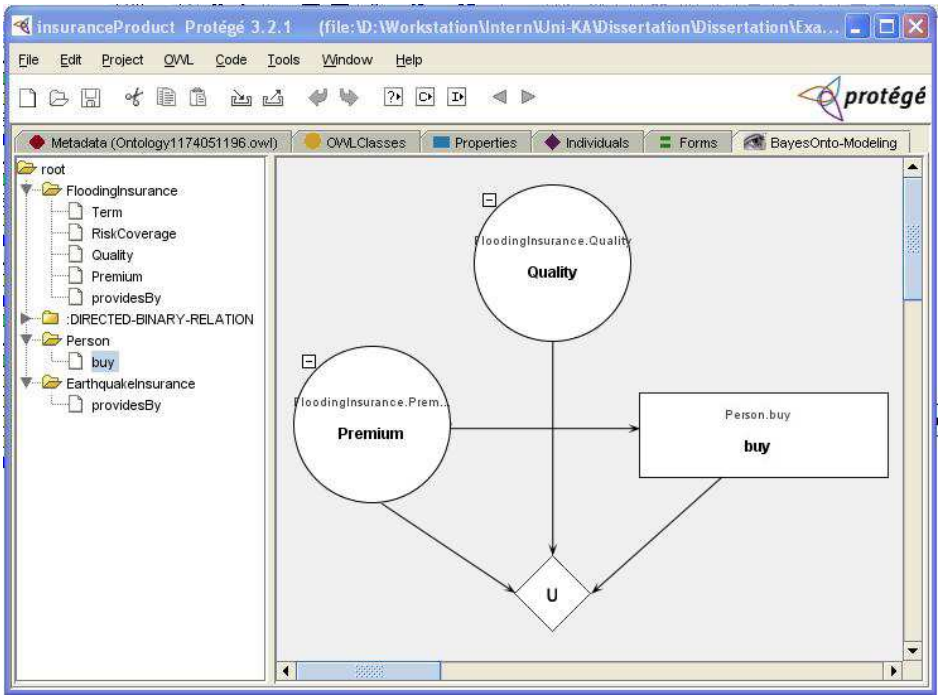

Figure 6.12: The decision model for buy a flooding insurance.

`PolicyA`) are `(buy=yes,36)` and `(buy=no,34)`. For `buy(Bob,PolicyB)` they are `(buy=yes,116)` and `(buy=no,-78)`. For `buy(Bob,PolicyC)` they are `(buy=yes,-32)` and `(buy=no,114)`. According to the highest expected utility two optimal decisions are returned to Bob: "`PolicyA` or `PolicyB` is your optimal choice".

6.3.4 Scenario 4

It is obvious that Bob can not really solve his problem of purchasing an insurance product only based on the decision model `personalBuyModel.owl` from his own repository. He needs more support from the system, e.g from VKCs. He sends a request to the system for creating a virtual team *VKC1* in the VKCs. This virtual team has the domain of interest "Bob's flooding insurance" and the normalized ontology of Bob's decision mode `personalBuyModel.owl` (Figure 6.12). The system assigns a domain expert of assessing insurance policy, Mr. David, to this virtual team. David joins *VKC1* and shares his decision model `assessPolicy2.owl` to help Bob make an optimal decision. Bob can complete his decision model `personalBuyModel.owl` with `assessPolicy2.owl`. The new decision model was illustrated in Figure 6.7 and is specified in `buyModel.owl`.

In fact David adds a new decision node `assess(InsuranceExpert,Flood-`

ingInsurance) to Figure 6.12. This decision node has a utility node U1 which has a determinate variable InsuranceExpert.AssessPayment as its decision criterion. With the added nodes the new model is informing Bob that David can evaluate the insurance products and give a quality report about them. Based on the quality evaluation Bob can make a better decision. But, for this service Bob must pay.

The new decision model is a sequential decision problem. Before Bob makes a decision to buy one product, he must decide first whether he needs the service of product quality evaluation. To make the first decision, Bob needs to know the payment for David — the evidence of the chance node David.AssessPayment. An instance of the chance node was given in the VKC1 when David submitted his decision model: David.AssessPayment=high. Now the decision model can be evaluated based on this evidence. The expected utility for assess(David,FloodingInsurance) are (assess=yes,17.5) and (assess=no,95.8). Therefore Bob decides not to take David's service, because it is too expensive. The decision problem is not solved. Bob still has no idea which one he should buy between PolicyA and PolicyB, but at least he got a new decision model.

6.3.5 Scenario 5

Bob sends his personal agent to join all VKCs and to find useful information for his problem. In *VKC2* (having the domain of interest "Alice's Flooding Insurance") he observed that the payment for the domain expert, Mr. Hugo, is low. He asks Hugo for joining his virtual team. Hugo joins it and provides the same payment of the quality evaluation service as for Alice. Now Bob analyzes his decision model again, but with the new evidence: Hugo.AssessPayment=low. The expected utility for assess(Hugo,FloodingInsurance) are (assess=yes,117.5) and (assess=no,75.8). According to the highest expected utility Bob decides to take Hugo's service.

Hugo gets the payment of Bob and evaluates PolicyA and PolicyB. The domain knowledge for evaluating the quality of insurance products was illustrated in Figure 6.6 and specified in assessPolicy.owl. The figure is a BN and indicates that the quality of an insurance product depends on three criteria: the term, the risk coverage and the premium of an insurance product. The premiums of PolicyA and PolicyB were identified in Scenario 3: PolicyA.Premium=middle and PolicyB.Premium=low. The remaining evidences that Hugo needs to know are about the nodes FloodingInsurance.RiskCoverage and FloodingInsurance.Term for these two products, respectively. With the following queries to insuranceProduct.owl Hugo can get the results for PolicyA: PolicyA.RiskCoverage=high and PolicyA.Term=long.

```
SELECT ?t
WHERE {?FloodingInsurance:Term ?t .
                        ?FloodingInsurance:policyName "PolicyA"}
SELECT ?r
WHERE {?FloodingInsurance:RiskCoverage ?r .
                        ?FloodingInsurance:policyName "PolicyA"}
```

With similar queries the results for PolicyB can be retrieved too: PolicyB.RiskCoverage=low and PolicyB.Term=long. Now the BN (assessPolicy.owl) can update the probabilities of all of its nodes, given these observations. The probability distribution of the node FloodingInsurance.Quality given the observations of PolicyA and PolicyB are (PolicyA.Quality=good,0.68), (PolicyA.Quality=bad,0.32), (PolicyB.Quality=good,0.3) and (PolicyB.Quality=bad,0.7).

Based on these results Hugo gives the report to Bob with his evaluation result: PolicyA.Quality=good and PolicyB.Quality=bad. Now Bob can evaluate his decision model again based on the new evidences reported by Hugo. The expected utility for buy(Bob,PolicyA) are (buy=yes,150) and (buy=no,80). For buy(Bob,PolicyB) they are (buy=yes,40) and (buy=no,80). According to these utilities the system recommends to Bob to buy PolicyA, not PolicyB. The problem is solved finally.

6.3.6 Concluding Remarks

In this section we presented a series of scenarios for testing the feasibility of the DSS, with a focus on the following main features.

- Decision making with uncertain knowledge. Our DSS is decision analytical which combines utility theory and probability theory. The uncertain knowledge modeling is based on BNs and the decision analysis is based on IDs. Scenario 1, 3, 4 and 5 demonstrated that our system can make optimal decisions with uncertain knowledge like other methodologies based on IDs.

  Comparing Scenario 3 to Scenario 5, a conclusion can be drawn: More evidences can be found, more precise decision can be made in an uncertain environment. With the decision model used in Scenario 3 Bob got a recommendation from the DSS with two choices: buy PolicyA or PolicyB. But it is still unclear which one is better. If he must make a decision at that time, he will probably choose PolicyB, because its premium is lower than PolicyA. In fact PolicyA is better than PolicyB as shown in Scenario 5. The evidences about the quality of these products helped Bob get a better and precise recommendation from the DSS.

- Knowledge integration in OntoBayes. As shown in the scenarios, the underlying formal representation language of OntoBayes is OWL, in spite of the fact that it represents three different methodologies. To identify the factual interpretation for an OWL file we make use of upper ontologies described in Section 4.2 and 4.3. The formal knowledge codification of OntoBayes enables to identify the domain specific and situational knowledge used for BNs and IDs automatically.

- Knowledge sharing through VKCs. Knowledge management tools can enhance knowledge by a variety of ways including generation, codification and transfer. But "without a culture that supports the rewards of knowledge sharing, the tools can be useless" [Rug97].

  In Scenario 4 Bob created a VKC for building a virtual team of dealing with his decision problem. He took the advantage of knowledge sharing through VKCs by completing his decision model with the shared model of the participant David. And in Scenario 5 he sent its personal agent to participate in all exiting VKCs for obtaining more useful knowledge. This is exactly the corporate knowledge approach for decision making. The result of this scenario shown that Bob did profit from corporate knowledge, because he got an optimal solution which he will never know when he did not utilize VKCs. Together with Scenario 4 and 5 we demonstrated how VKCs can facilitate knowledge sharing and how decision making can profit from corporate knowledge.

The exchanged knowledge and decision models through VKCs in the presented scenarios in fact can only take place in the community buffer and hardly be observed. But for testing the feasibility of the VKCs approach we must record these exchanged unit in OWL files. All OWL files used in the scenarios above can be found under the web link `http://iaks-www.ira.uka.de/home/yiyang/DSS/Testdata`.

Chapter 7

Conclusions

The main *goal* of this dissertation was to design and develop a DSS adapted to uncertain knowledge. As discussed in Section 2.2 uncertainty is one of the most important, daunting, but inevitable challenges in an open and dynamic environment for decision making. Not only the imperfection of information, but also the uncertain nature of correlations between decisions and outcomes, cause uncertainty. Therefore, a sound DSS must be able to work under uncertainty. Besides uncertainty, there are other important challenges to address. As mentioned in Section 3.1.3 they are adaptivity, knowledge management, collaboration, intelligence and the explanatory power.

In order to meet the goal, the following *research objectives* were established and achieved:

1. The development of an advanced and abstract framework for DSSs to address all these challenges in an open and dynamic environment, particularly with the emphasis on uncertain knowledge. This framework is characterized by different views: a pillar-based view, a layered view, a decision theoretical view and a process view (see Section 3.2).

   The pillar-base view points out six important methodologies of the framework. They are ontologies, BNs, IDs, VKCs, MAS and Web Services. The first four methodologies are the theoretical foundations of the framework, whereas the last two are the technical building blocks for implementing it. Features and functionalities of these methodologies with regard to DSSs are described in Section 3.2.1.

   The layered view shows the hierarchical layers from the system back end to the application front end in a bottom up approach. There are four generic layers: a repository layer, a management component layer, a collaboration layer and an application layer (see Section 3.2.2). These layers guided the design of the architecture depicted in Figure 6.1.

The decision theoretical view summarizes normative, descriptive and prescriptive perspectives of the framework (see Section 3.2.3). It points out that the proposed framework has a hybrid decision making approach with all of these three perspectives.

The process view points out a low level and a high level decision support of the framework (see Section 3.2.4). The former regards a DSS more as a query system which can provide information according to the input queries, whereas the latter regards a DSS more like a decision analytic system which can recommend solutions according to the decision problems via analyzing and forecasting current and future environment. The low level support is relative simple and intuitive. On the contrary the high level support is more complicated because it needs a dynamical decision analysis process (depicted in Figure 3.5) which is inspired by the decision analysis cycle [Hol89] and Simon's model of the decision process [Sim60].

2. The design, development and use of the OntoBayes model in decision support for solving decision problems having complex structure and uncertainty (see Chapter 4). OntoBayes integrates the methodologies of ontologies, BNs and IDs into an ontology-driven model, in order to preserve all advantages of them.

   OntoBayes is designed with two parts: a knowledge part and a decision model part. The former includes ontological and Bayesian knowledge, while the latter specifies different decision models based on IDs. The selected underlying formal KR language of OntoBayes is OWL. In order to enable the integration of BNs and IDs into ontologies, two upper ontologies (depicted in Figure 4.1 and 4.9 respectively) are given for capturing essentials of BNs and IDs. Moreover, a number of OWL annotations are specified based on these upper ontologies in OntoBayes.

   As described in Section 4.2.5 and 4.3.3 in OntoBayes the ontological knowledge, Bayesian knowledge and decision models are not separate, but interrelated. The interrelation between them facilitates knowledge acquisition through ontological queries which are very important to find evidences for decision making under uncertainty as shown in Section 6.3.

   The utilization of the OntoBayes model provides formal supports for decision making. From the syntactical perspective the OntoBayes model totally rests on OWL which is a formal language for modeling ontologies. From the semantical perspective OntoBayes not only retains the usual semantic of ontologies, but also incorporates the semantics of BNs and IDs into ontologies. In order to distinguish the specifications of different OWL files in the OntoBayes model, upper ontologies for BNs and IDs are suggested.

These upper ontologies are predefined in the head information of OWL files in OntoBayes as described in Section 4.2 and 4.3. Therefore an OWL file can be easily recognized whether it specifies a BN or an ID.

The most related works based on the probabilistic approach and on the fuzzy approach are investigated and surveyed in Section 4.4. The survey shows that the OntoBayes model is an original work and differs very from other proposals or methods in the domain of incorporating uncertainty into ontologies. The knowledge part of OntoBayes has similarities with them, but the decision model part is unique and decisive for decision making support.

3. The adaption and use of VKCs in decision support for the solution of knowledge transfer. VKCs provide a virtual platform to share knowledge, decision models and to facilitate the collaboration in a distributed way. A simple scenario was designed and introduced to demonstrate it (see Chapter 5).

   VKCs are not a simple extension of the previous implementations performed in the group and starting with Hammond's work [Ham04]. From the theoretical perspective the previous works do not comply with the new requirements and solutions described in Chapter 5. VKCs are expected to be fully integrated with OntoBayes which incorporates uncertain knowledge and decision models into ontologies. This challenge can not be addressed with a simple extension of the previous works of VKCs. Most important concepts about VKCs such as normalized ontology, knowledge cluster and knowledge instances, must be extended and redesigned for OntoBayes in accord with the semantics of BNs and IDs.

   From the technical perspective the previous works are not adequate for the knowledge exchange based on the underlying modeling language OWL. Hammond implemented VKCs in his master thesis with a UML-based approach. The resulted prototype can only be used to exchange RDF-based ontological knowledge. In order to integrate VKCs with OntoBayes, VKCs must be compatible with OWL, not only with RDF. The solution of converting OWL files into RDF files is not practical because OWL is build upon RDF and has more descriptive power. Therefore the implementation of VKCs must be improved to reach the OWL compatibility.

4. The development and test of a prototype of a DSS according to the proposed framework. An agent-based and service-oriented architecture is designed and proposed in Section 6.1. The MAS paradigm makes agents suitable as software entities for the delegation of diverse decision making tasks and for collaborative works. The SOA approach enables features of loose coupling, implementation neutrality, flexibility and adaptivity of DSSs.

Based on the system architecture (depicted in Figure 6.1) a prototype is implemented by using many open sources and commercial softwares. This prototype is able to support decision making with the OntoBayes model and VKCs under uncertainty. A Protégé plugin *OWLOntoBayes* is implemented for the OntoBayes model. It allows users to edit and codify BNs and IDs into OWL files with a user friendly GUI. Additionally an application interface is implemented to facilitate the utilization as well as the test of the prototype (see Section 6.2).

The test part allows to evaluate the feasibility of the prototype (see Section 6.3). A simple made-up use case in the application domain of catastrophe insurance is created. This use case is demonstrated in five scenarios according to the sequential steps that the decision maker solved these problems by using the prototype. All five scenarios show the unique feature of the implemented DSS — knowledge integration in OntoBayes. Among them, Scenario 1, 3, 4 and 5 successfully evaluate that users can make optimal decisions based on given BNs and IDs with predefined probability tables and utility tables. Particularly with Scenario 4 and 5, it is evaluated that the prototype is feasible to facilitate knowledge sharing via VKCs, to utilize VKCs for establishing virtual teams related to given decision problems, to build or complete decision models by sharing knowledge through VKCs.

7.1 Future Works

The implementation of the first prototype of the DSS will be extended and improved. More reasoning algorithms for BNs and IDs will be implemented. After that users can compare the computation results by using different algorithms for same decision problems. More decision making (or analysis) processes are expected to be incorporated into the system. Different processes provide different perspectives for making a decision about a same problem. They will give users or other service consumers more flexibility when selecting provided web services to support decision making.

In collaboration with the catastrophe insurance projects of the postgraduate school “Natural Disasters” applicable ontologies and decision models about natural disasters and catastrophe insurances will be developed and evaluated. This should allow real applications of the prototype in decision support concerning low probability-high loss-events like flooding and so on.

Open research lines to be considered, based on the experience from the doctoral work are:

- Enhancement of the decision model part of the OntoBayes model with other

BN-based methods for decision making. The current development of the decision model part in OntoBayes only allows to specify ontology-driven IDs. But theoretically this part can be extended to contain other BN-based methods such Markov decision process, decision network and so on. By means of domain specific upper ontologies these methods can be described in OWL enabling thus a semantical understanding.

- Integration of decision making or analysis processes with a formal business process language. In order to execute the decision analysis process proposed in the framework, a hardcoded solution is implemented in the prototype. But a more elegant and efficient solution is to execute such processes automatically. For that, processes can be integrated with an executable process language, e.g. BPEL (Business Process Execution Language).

- A trust mechanism for agents involved in knowledge sharing through VKCs. The current development of VKCs does not include any trust mechanism which can measure or control the knowledge transfered by agents in VKCs. This lack can result in unreliable knowledge sharing through VKCs and lead to incorrect decisions. To avoid that, VKCs must be enhanced with a trust mechanism.

- Methods to assess achieved decisions based upon corporate knowledge with possible conflicting information. Once obtaining corporate knowledge, it is possible to get conflicting information from different agents. For example a variable can be observed with different evidences, or with different probability distributions, from different information sources. Therefore methods for assessing such conflicting information are necessary.

Appendix A

Preliminary OWL Annotations for OntoBayes

OWL Annotations used to extend BNs and IDs into ontologies in the OntoBayes model are described in details in the following sections.

A.1 Class Annotations of BNs

<owl:Class rdf:ID="BayesianNetwork">

Description:

This class represents a BN as an ontological concept. A generic BN consists of many chance nodes.

Subclasses: None

Properties (with BayesianNetwork as its domain):

consistsOF (multiple ChanceNode)

<owl:Class rdf:ID="ChanceNode">

Description:

This class represents discrete variables of a BN as chance nodes. It is the super class of all domain specific nodes in a BN. Each chance node has domain values specified in the datatype of string. When the state of a chance node is observed, then it is an evidence variable. Each chance node is associated with at least one probability table. Among these tables one is selected to be active when reasoning about the BN

of the node. A chance node may have dependency relations to others. Based on the dependency relation we can divide chance nodes into two types: conditional and unconditional nodes.

Subclasses:

`<owl:Class rdf:ID=‘‘CondNode’’>`
`<owl:Class rdf:ID=‘‘UncondNode’’>`

Properties (with ChanceNode as its domain):

hasObserved (single xsd:string)
hasDomain (single xsd:string)
dependsOn (multiple ChanceNode)
hasActiveP (single JointProbDist)

<owl:Class rdf:ID="CondNode">

Description:

This class represents conditional chance nodes of a BN which has parents in the BN. It is the subclass of `ChanceNode` and inherits all properties of a generic chance node. Every conditional chance node has at least one CPT.

Subclasses: None

Properties (with CondNode as its domain):

hasObserved (single xsd:string)
hasDomain (single xsd:string)
dependsOn (multiple ChanceNode)
hasActiveP (single JointProbDist)
hasCondTable (multiple CondProbDist)

<owl:Class rdf:ID="UncondNode">

Description:

This class represents unconditional chance nodes of a BN which has no parents in the BN. It is the subclass of `ChanceNode` and inherit all properties of a generic chance node. Every unconditional chance node has at least one unconditional probability table.

Subclasses: None

Properties (with UncondNode as its domain):

hasObserved (single xsd:string)
hasDomain (single xsd:string)
dependsOn (multiple ChanceNode)
hasActiveP (single JointProbDist)
hasUncondTable (multiple CondProbDist)

<owl:Class rdf:ID="JointProbDist">

Description:

This class represents the joint probability distribution of a Bayesian variable (the chance node). In this work joint probability distributions are representable by probability tables. According to the type of the node associated with the probability table, it can be divided into two subclasses: unconditional and conditional probability distributions (or tables). Each table consists of a number of table cells.

Subclasses:

`<owl:Class rdf:ID=''CondProbDist''>`
`<owl:Class rdf:ID=''UncondProbDist''>`

Properties (with JointProbDist as its domain):

hasPCell (multiple ProbCell)

<owl:Class rdf:ID="CondProbDist">

Description:

This class represents conditional probability tables of a BN. It is the subclass of `JointProbDist` and inherits all properties of `JointProbDist`.

Subclasses: None

Properties (with CondProbDist as its domain):

hasPCell (multiple ProbCell)

<owl:Class rdf:ID="UncondProbDist">

Description:

This class represents unconditional probability tables of a BN. It is the subclass of `JointProbDist` and inherits all properties of `JointProbDist`.

Subclasses: None

Properties (with UncondProbDist as its domain):
hasPCell (multiple ProbCell)

<owl:Class rdf:ID="ProbCell">

Description:
This class represents cells of a probability table. To simplify the annotations in OWL, we specify them with two associated datatype properties: one for representing probability values and the other for parameters. To distinguish cells of a CPT from the cells of an unconditional probability table, a syntax parser is implemented in the work.

Subclasses: None

Properties (with ProbCell as its domain):
hasPParameter (single xsd:string)
hasPValue (single xsd:float)

A.2 Property Annotations of BNs

<owl:ObjectProperty rdf:ID="consistsOf">

Description:
This object property is the link between a BN and its chance nodes. It indicates the membership of a chance node, i.e. to which BN a node belongs.

<owl:ObjectProperty rdf:ID="dependsOn">

Description:
This object property specifies the dependency relation between two chance nodes *A* and *B*. If *A* depends on *B*, then *A* is a conditional chance node associated with CPTs and *B* is an unconditional chance node associated with unconditional probability tables.

<owl:ObjectProperty rdf:ID="hasCondTable">

Description:

This object property links a conditional chance node to a CPT. It is allowed in this work that a conditional chance node have more than one CPTs.

<owl:ObjectProperty rdf:ID="hasUncondTable">

Description:

This object property links an unconditional chance node to a CPT. It is allowed in this work that an unconditional chance node have more than one unconditional probability tables.

<owl:ObjectProperty rdf:ID="hasActiveP">

Description:

This object property indicates that a probability table is active when reasoning about a BN. Due to the fact that chance nodes in this work may have many probability tables, it is important to set exact one table for the need of reasoning computation.

<owl:ObjectProperty rdf:ID="hasPCell">

Description:

This object property is the link between a probability table and the cells of the table. It can indicate to which table a cell belongs.

<owl:DatatypeProperty rdf:ID="hasDomain">

Description:

This datatype property specifies the domain value of a discrete variable (the chance node) of a BN using `xsd:string`. The domain value consists of many mutual and exclusive states.

<owl:DatatypeProperty rdf:ID="hasObserved">

Description:

This datatype property indicates the observed state of a chance node using `xsd:string`. A chance node is a discrete variable in a BN and has many states. Once a state is observed, the probability value of the

state will be changed to 100%. This mean the state of the variable is an evidence.

<owl:DatatypeProperty rdf:ID="hasPValue">

Description:

This datatype property specifies the probability value of a probability table cell using `xsd:float`. And the value is restricted to the interval [0,1].

<owl:DatatypeProperty rdf:ID="hasPParameter">

Description:

This datatype property specifies the parameter of a probability table cell using `xsd:string`. The parameter of a cell is a possible combination of states of variables. In fact each probability table cell has a binary relation with a parameter and a corresponding probability value.

A.3 Class Annotations of IDs

Due to the fact that the class and property annotations of IDs in OWL are extended based on the annotations of BNs, we will only describe the additional ones in the following sections.

<owl:Class rdf:ID="InfluenceDiagram">

Description:

This class is used to specify a generic ID in OWL. An ID has three kinds of node: chance nodes, decision nodes and utility nodes.

Subclasses: None

Properties (with InfluenceDiagram as its domain):

hasUNode (multiple UtilityNode)
hasDNode (multiple DecisionNode)
hasCNode (multiple ChanceNode)

<owl:Class rdf:ID="DecisionNode">

Description:

This class is used to specify decision nodes of IDs in OWL. Each decision node represents a set of possible alternatives available to decision makers. These alternatives are specified with `xsd:string`.

Subclasses: None

Properties (with DecisionNode as its domain):

hasAlternatives (single xsd:string)
isKnownBy (single DecisionNode)

<owl:Class rdf:ID="UtilityNode">

Description:

This class is used to specify utility nodes of IDs in OWL. A utility node may have many utility tables, but is allowed to have only one active table for evaluating IDs.

Subclasses: None

Properties (with UtilityNode as its domain):

hasUTable (multiple UtilityTable)
hasActiveU (single UtilityTable)

<owl:Class rdf:ID="UtilityTable">

Description:

This class is used to specify utility tables in OWL which can reflect the preferences of a decision maker. Similar to probability tables, a utility table consists of many table cells.

Subclasses: None

Properties (with UtilityTable as its domain):

hasUCell (multiple UtilityCell)

<owl:Class rdf:ID="UtilityCell">

Description:

This class is used to specify cells of a utility table. Similar to cells of probability table, such a utility table cell has a parameter and a utility value.

Subclasses: None

Properties (with UtilityCell as its domain):

hasUParameter (single xsd:string)
hasUValue (single xsd:float)

A.4 Property Annotations of IDs

<owl:ObjectProperty rdf:ID="hasDNode">

Description:

This object property is the link between an ID and its decision nodes. An ID may have many decision nodes.

<owl:ObjectProperty rdf:ID="hasCNode">

Description:

This object property is the link between an ID and its chance nodes. An ID may have many chance nodes.

<owl:ObjectProperty rdf:ID="hasUNode">

Description:

This object property is the link between an ID and its utility nodes. An ID may have many utility nodes.

<owl:ObjectProperty rdf:ID="isKnownBy">

Description:

This object property is used to specify informational arcs in an ID. Such arcs are arcs into decision nodes and indicate which information is known to a decision maker at a decision time.

<owl:ObjectProperty rdf:ID="influenceOn">

Description:

This object property is used to specify conditional arcs in an ID. It is an inverse property of `dependsOn` and indicates the probabilistic dependency on the associated variables.

<owl:ObjectProperty rdf:ID="attributeOf">

Description:

This object property is used to specify functional arcs in an ID. Functional arcs are arcs into a utility node and indicate which variables are functionally dependent on the utility node.

<owl:ObjectProperty rdf:ID="hasUTable">

Description:

This object property is used to link a utility node to a utility table. A utility node may have many utility tables.

<owl:ObjectProperty rdf:ID="hasUCell">

Description:

This object property is used to link a utility table to a utility table cell. A utility table consists of many cells.

<owl:ObjectProperty rdf:ID="hasActiveU">

Description:

This object property is used to specify an active utility table which is involved in the evaluation algorithm of an ID.

<owl:DatatypeProperty rdf:ID="hasAlternatives">

Description:

This datatype property is used to specify possible alternatives of a decision node using `xsd:string`.

<owl:DatatypeProperty rdf:ID="hasUValue">

Description:

This datatype property is used to specify the utility value of a utility table cell using `xsd:float`.

<owl:DatatypeProperty rdf:ID="hasUParameter">

Description:

This datatype property is used to specify the parameter of a utility table cell using `xsd:string`. A utility table cell is completed in a binary relation with the property `hasUValue` and this one.

List of Figures

List of Tables

List of Abbreviations

Abbreviation Page

Bibliography

[ACKM04] Gustavo Alonso, Fabio Casati, Hurumi Kuno, and Vijay Machiraju. *web services: concepts, architectures and applications*. Springer, 2004.

[AL01] M. Alavi and D. Leidner. Knowledge management and knowledge management systems: Conceptual foundations and research issues. *MISQ Review*, 25(1):107–136, 2001.

[APM+06] Gabriel Amores, Guillermo Pérez, Pilar Manchón, Fernando Gómez, and Jesús González. Integrating owl ontologies with a dialogue manager. *Procesamiento del Lenguaje Natural*, 37:153–160, 2006.

[AS03] Muthukkaruppan Annamalai and Leon Sterling. Guidelines for constructing reusable domain ontologies. In Stephen Cranefield, Timothy W. Finin, Valentina A. M. Tamma, and Steven Willmott, editors, *Proceedings of the Workshop on Ontologies in Agent Systems (OAS'03) at the 2nd International Joint Conference on Autonomous Agents and Multi-Agent Systems*, volume 73 of *CEUR Workshop Proceedings*, pages 71–74. CEUR-WS.org, 2003.

[AvH04] Grigoris Antoniou and Frank van Harmelen. Web ontology language: Owl. In S. Staab and R. Studer, editors, *Handbook on Ontologies*, pages 67–92. Springer, 2004.

[BAH90] M. H. Birnbaum, C. J. Anderson, and L. G. Hynan. *Cognitive biases*, chapter Theories of bias in probability judgment, pages 477–498. Elsevier Science Publishers, 1990.

[Bau02] Heinz Bauer. *Wahrscheinlichkeitstheorie*. Walter de Gruyter, 5 edition, 2002.

[BBC02] M. Bonifacio, P. Bouquet, and R. Cuel. Knowledge nodes: the building blocks of a distributed approach to knowledge management. *Journal of Universal Computer Science*, 8(6):652–661, 2002.

[BBSS+00] I. Baturone, Á. Barriga, S. Sànchez-Solano, C. J. Jimènez-Fernàndez, and D. R. Lòpez. *Microelectronic Design of Fuzzy Logic-Based Systems*. CRC Press, January 2000.

[BDS04] Thomas Bittner, Maureen Donnelly, and Barry Smith. Individuals, universals, collections: On the foundational relations of ontology. In *Proceedings of the 3rd International Conference on Formal Ontology and Information Systems (FOIS'04)*, pages 37–48, Turin, Italy, November 2004. IOS Press.

[BF05] C.J. Butz and F. Fang. Incorporating evidence in bayesian networks with the select operator. In *Proceedings of the 18th Conference of the Canadian Society for Computational Studies of Intelligence (Canadian AI 2005), Advances in Artificial Intelligence*, volume 3501/2005 of *LNCS*, Victoria, Canada, May 2005. Springer.

[BG04] Dan Brickley and R.V. Guha. Rdf vocabulary description language 1.0: Rdf schema, February 2004. Available from: `http://www.w3.org/TR/rdf-schema/`.

[BGH01] Sean Bechhofer, Carole A. Goble, and Ian Horrocks. Daml+oil is not enough. In *Proceedings of the 1st Semantic Web Working Symposium (SWWS'01)*, pages 151–159, California, USA, 2001.

[BLHL01] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5):34–43, 2001.

[BMP+06] Denis Bouyssou, T. Marchant, M. Pirlot, A. Tsoukias, and P. Vincke. *Evaluation and decision models with multiple criteria*. Springe, 2006.

[Bor96] B. J. Borghetti. Inference algorithm performance and selection under constrained resources. Master's thesis, AFIT/GCS/ENG/96D-05, 1996.

[Bou05] Craig Boutilier. The influence of influence diagrams on artificial intelligence. *Decision Analysis,*, 2(4):229–231, December 2005.

[Bro89] Rex V. Brown. Toward a prescriptive science and technology of decision aiding. *Annals of Operations Research*, 19(1):465–483, December 1989.

[BRT88] David E. Bell, Howard Raiffa, and Amos Tversky. *Decision Making: Descriptive, Normative, and Prescriptive Interactions.* Cambridge University Press, 1988.

[Bru02] Herman Bruyninckx. Bayesian probability. 2002.

[BvHH+04] Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. Owl web ontology language reference, February 2004. Available from: `http://www.w3.org/TR/owl-ref/`.

[CD00] J. Cheng and M.J. Druzdzel. Ais-bn: An adaptive importance sampling algorithm for evidential reasoning in large bayesian networks. *Journal of Artificial Intelligence Research*, 13:155–188, 2000.

[CJW98] Christer Carlsson, Tawfik Jelassi, and Pirkko Walden. Intelligent systems and active dss. In *Proceedings of the 31nd Annual Hawaii International Conference on System Sciences*. IEEE Computer Society, 1998.

[CK01] Wolfram Conen and Reinhold Klapsing. Logical interpretations of rdfs - a compatibility guide. Working Paper, November 2001.

[CL06] Paulo C. G. Costa and Kathryn B. Laskey. Multi-entity bayesian networks without multi-tears. Technical Report C4I-05-07, Department of Systems Engineering and Operations Research, George Mason University, Fairfax, VA, USA, January 2006.

[CLA06] Paulo C. G. Costa, Kathryn B. Laskey, and Ghazi Alghamdi. Bayesian ontologies in ai systems. In *Proceedings of the 4th Bayesian Modelling Applications Workshop*, Cambridge, MA, USA, July 2006.

[CLL06] Paulo C. G. Costa, Kathryn B. Laskey, and Kenneth J. Laskey. Probabilistic ontologies for efficient resource sharing in semantic web services. In *Proceedings of the second workshop on Uncertainty Reasoning for the Semantic Web (URSW'06)*, Athens, GA, USA, November 2006. CEUR-WS.org.

[CME04] J. Calmet, P. Maret, and R. Endsuleit. Agent-oriented abstraction. *Revista (Real Academia de Ciencias, Serie A de Matematicas)*, 98(1):77–84, 2004. Special Issue on Symbolic Computation and Artificial Intelligence.

[Col02] Robert M. Colomb. Use of upper ontologies for interoperation of information systems: A tutorial. Technical Report 20/02 ISIB-CNR, Institute of Biomedical Engineering, National Research Council, Padova, Italy, November 2002.

[Coo88] G. F. Cooper. A method for using belief-network algorithms to solve decision-network problems. In *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence (UAI'88)*, pages 55–63, 1988.

[Coo90] G. F. Cooper. The computational complexity of probabilistic inference using bayesian belief networks. *Artificial Intelligence*, 42(2-3):393–405, 1990.

[Cos05] Paulo C.G. Costa. *Bayesian Semantics for the Semantic Web*. PhD thesis, George Mason University, Fairfax, VA, USA, July 2005.

[Cow98] Robert Cowell. Introduction to inference for bayesian networks. In *Proceedings of the NATO Advanced Study Institute on Learning in graphical models*, pages 9–26, Erice, Italy, 1998. Kluwer Academic Publishers.

[Cro04] Mark Crowley. Evaluating influence diagrams. August 2004. Available from: `http://www.cs.ubc.ca/~crowley/papers/aiproj.pdf`.

[CvHH+01] Dan Connolly, Frank van Harmelen, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. Daml+oil (march 2001) reference description, December 2001. Available from: `http://www.w3.org/TR/daml+oil-reference`.

[Dae05] Anusch Daemi. *Entropiebasierte Bewertung von Ontologien*. PhD thesis, University Karlsruhe, 2005.

[Dan06] P. Dangauthier. Philosophical foundations of probabilities. Technical report, The french national institute for research in computer science and control (INRIA), September 2006.

[Daw79] A. P. Dawid. Conditional independence in statistical theory. *Journal of the Royal Statistical Society*, 41(1):1–31, 1979.

[dCLL05] Paulo Cesar G. da Costa, Kathryn B. Laskey, and Kenneth J. Laskey. Pr-owl: A bayesian ontology language for the semantic web. In *Proceedings of the first workshop on Uncertainty Reasoning for the Se-*

mantic Web (URSW'05), pages 23–33, Galway, Ireland, November 2005.

[Dem67] A. P. Dempster. Upper and lower probabilities induced by a multivalued mapping. *Annals of Mathematical Statistics*, 38:325–339, 1967.

[Dem68] A. P. Dempster. A generalization of bayesian inference. *Journal of the Royal Statistical Society*, B 30:205–247, 1968.

[DF03] Maret J. Druzdzel and Roger R. Flynn. Decision support systems. In Miriam Drake Marcia J. Bates, Mary Niles Maack, editor, *Encyclopedia of Library and Information Science*, pages 794 – 802. Marcel Dekker, Inc., second edition edition, 2003.

[DH04] M. Diehl and Yacov Y. Haimes. Influence diagrams with multiple objectives and tradeoff analysis. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 34(3):293–304, 2004.

[DHN90] R. O. Duda, P. E. Hart, and N. J. Nilsson. Subjective bayesian methods for rule-based inference systems. In *Readings in uncertain reasoning*, pages 274–281, San Francisco, CA, USA, 1990. Morgan Kaufmann.

[Dil98] Stuart M. Dillon. Descriptive decision making: Comparing theory with practice. In *Proceedings of the 33rd Annual Conference of the Operational Research Society of New Zealand (ORSNZ'98)*, 1998.

[Din05] Zhongli Ding. *BayesOWL A Probabilistic Framework for Semantic Web*. PhD thesis, University of Maryland, Baltimore County, USA, December 2005.

[DL93] P. Dagum and M. Ludy. Approximating probabilistic inference in bayesian belief networks is np-hard. *Artificial Intelligence*, 60(1):141–153, 1993.

[dL96] Marquis de Laplace. *A Philosophical Essay on Probabilities*. Dover Publications, January 1996.

[DL05] Marek J. Druzdzel and Tze-Yun Leong, editors. *Challenges to Decision Support in a Changing World*. AAAI Press, 2005.

[DOS03] Michael C. Daconta, Leo J. Obrst, and Kevin B. Smith. *The Semantic Web : a guide to the future of XML, Web services, and knowledge management*. Wiley, Indianapolis, IN, USA, 2003.

[DP04] Zhongli Ding and Yun Peng. A probabilistic extension to ontology language owl. In *Proceeding of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04)*, volume 4, page 40111.1, Washington, DC, USA, January 2004. IEEE Computer Society.

[DPP04] Z. Ding, Y. Peng, and R. Pan. A bayesian approach to uncertainty modelling in owl ontology. In *Proceedings of 2004 International Conference on Advances in Intelligent Systems - Theory and Applications (AISTA'04)*, Luxembourg, November 2004. IEEE Computer Society.

[FC89] Robert M. Fung and Kuo-Chu Chang. Weighing and integrating evidence for stochastic simulation in bayesian networks. In *Proceedings of the 5th Annual Conference on Uncertainty in Artificial Intelligence (UAI'89)*, pages 209–220. Elsevier, 1989.

[Fen01] Dieter Fensel. *Ontologies: A silver Bullet for Knowledge Management and Electronic Commerce*. Springer, 2001.

[FF94] Robert M. Fung and Brendan Del Favero. Backward simulation in bayesian networks. In *Proceedings of the 10th Annual Conference on Uncertainty in Artificial Intelligence (UAI'94)*, pages 227–234, Seattle, Washington, USA, 1994. Morgan Kaufmann.

[FGKP99] Nir Friedman, Lise Getoor, Daphne Koller, and Avi Pfeffer. Learning probabilistic relational models. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI '99)*, pages 1300–1309, San Francisco, CA, USA, 1999. Morgan Kaufmann.

[FHvH+00] Dieter Fensel, Ian Horrocks, Frank van Harmelen, Stefan Decker, Michael Erdmann, and Michel C. A. Klein. Oil in a nutshell. In *Proceedings of the 12th European Workshop on Knowledge Acquisition, Modeling and Management (EKAW'00)*, pages 1–16, London, UK, 2000. Springer-Verlag.

[FO01] Gerhard Fischer and Jonathan Ostwald. Knowledge management: Problems, promises, realities, and challenges. *IEEE Intelligent Systems*, 16(1):60–72, 2001.

[Fuk04] Y. Fukushige. Representing probabilistic knowledge in the semantic web. In *position paper for the W3C Workshop on Semantic Web for Life Sciences*, Cambridge, MA, USA, 2004.

[Fuk05] Y. Fukushige. Representing probabilistic relations in rdf. In *Proceedings of Workshop on Uncertainty Reasoning for the Semantic Web (URSW) at the 4th International Semantic Web Conference (ISWC)*, Galway, Ireland, November 2005.

[GG95] N. Guarino and P. Giaretta. Ontologies and knowledge bases: Towards a terminological clarification. In N. J. I. Mars, editor, *Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing*, pages 25–32. IOS Press, 1995.

[GGP+02] Jennifer Golbeck, Michael Grove, Bijan Parsia, Aditya Kalyanpur, and James Hendler. New tools for the semantic web. In *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW'02)*, LNCS, pages 392–400, Siguenza, Spain, October 2002. Springer.

[GH02] H. Guo and W. Hsu. A survey of algorithms for real-time bayesian network inference. In *Proceedings of the joint workshop on Real-Time Decision Support and Diagnosis Systems in AAAI/KDD/UAI 2002*, Edmonton, Alberta, Canada, 2002.

[GHKP04] T. Gu and D. Q. Zhang H. K. Pung. Towards an osgi-based infrastructure for context-aware applications in smart homes. *IEEE Pervasive Computing*, 3(4):66–74, 2004.

[GHM05] Michael Gruninger, Richard Hull, and Sheila McIlraith. A first-order ontology for semantic web services. In *Proceedings of W3C Workshop on Frameworks for Semantics in Web Services 2005 (FSWS'05)*, Innsbruck, Austria, June 2005. W3C.

[GL90] R. V. Guha and Douglas B. Lenat. Cyc: a mid-term report. *AI Magazine*, 11(3):32–59, 1990.

[GL02] Rosalba Giugno and Thomas Lukasiewicz. P-shoq(d): A probabilistic extension of shoq(d) for probabilistic ontologies in the semantic web. In *Proceedings of the European Conference on Logics in Artificial Intelligence (JELIA'02)*, volume 2424 of *LNCS*, pages 86–97, London, UK, 2002. Springer.

[GL05] Mingxia Gao and Chunnian Liu. Extending owl by fuzzy description logic. In *Proceedings of the 17th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'05)*, pages 562–567, Hong Kong, China, November 2005. IEEE Computer Society.

[Gly01] Clark Glymour. *The Mind's Arrow: Bayes Nets and Graphical Causal Models in Psychology*. The MIT Press, 2001.

[GN87] Michael R. Genesereth and Nils J. Nilsson. *Logical foundations of artificial intelligence*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1987.

[GPFLC04] Asunción Gómez-Pérez, Mariano Fernández-López, and Oscar Corcho. *Ontological engineering*. Springer, 2004.

[GPZ04] Tao Gu, Hung Keng Pung, and Da Qing Zhang. A bayesian approach for dealing with uncertain contexts. In G. Kotsis, editor, *Proceedings of the 2nd International Conference on Pervasive Computing (Pervasive'04), in the book "Advances in Pervasive Computing"*, volume 176, Linz/Vienna, Austria, April 2004. Austrian Computer Society.

[Gro96] Barbara J. Grosz. Aaai-94 presidential address: Collaborative systems. *AI Magazine*, 17(2):67–85, 1996.

[Gru93] T. R. Gruber. A translation approach to protable ontology specification. *Knowledge Acquisition*, 5(2):199–220, 1993.

[Gru95] Thomas R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies*, 43(5-6):907–928, 1995.

[Gua98] N. Guarino. Formal ontology and information systems. In N. Guarino, editor, *Proceedings of the 1st International Conference on Formal Ontologies in Information Systems (FOIS'98)*, pages 3–15, Trento, Italy, June 1998. IOS Press.

[Gui07] Cedric Guillou. The corporate knowledge power for decision support systems. Master's thesis, University of Karlsruhe, 2007.

[GWPZ04] T. Gu, X. Wang, H. Pung, and D. Zhang. An ontology-based context model in intelligent environments. In *Proceedings of Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS'04)*, pages 270–275, San Diego, California, USA, January 2004. The Society for Modeling and Simulation International.

[Ham04] Marc Hammond. Virtual knowledge communities for distributed knowledge management: A multi-agent-based approach using jade.

Master's thesis, University of Karlsruhe and Imperial College London, 2004.

[Hen87] Max Henrion. Some practical issues in constructing belief networks. In Laveen N. Kanal, Tod S. Levitt, and John F. Lemmer, editors, *Proceedings of the 3rd Annual Conference on Uncertainty in Artificial Intelligence (UAI'87)*, pages 161–17. Elsevier, 1987.

[Her07] Philipp Hering. Von ontologien zu bayes-netzwerken: Entwicklung einer modellierungsumgebung zur reprsentation unscharfen wissens. Master's thesis, University Karlsruhe, 2007.

[HH00] Frank Van Harmelen and Ian Horrocks. Questions and answers on oil. *IEEE Intelligent Systems*, 15(6):69–72, 2000.

[HH04] Markus Holi and Eero Hyvönen. Probabilistic information retrieval based on conceptual overlap in semantic web ontologies. In *Proceedings of the 11th Finnish Artificial Intelligence Conference (FAIS'04)*, number 2, pages 83–97, Vantaa, Finland, September 2004. Finnish Artificial Intelligence Society.

[HH05] Markus Holi and Eero Hyvönen. Modeling degrees of conceptual overlap in semantic web ontologies. In *Proceedings of Workshop on Uncertainty Reasoning for the Semantic Web (URSW'05) at the 4th International Semantic Web Conference (ISWC'05)*, Galway, Ireland, November 2005.

[HM84] R. A. Howard and J. E. Matheson. Influence diagrams. In R.A. Howard and J.E. Matheson, editors, *Readings on the Principles and Applications of Decision Analysis*, pages 721–762, Menlo Park, California, April 1984. Strategic Decisions Group.

[HM05] Ronald A. Howard and James E. Matheson. Influence diagrams. *Decision Analysis*, 2(3):127–143, September 2005.

[Hol89] Samuel Holtzman. *Intelligent decision systems*. Addison-Wesley, 1989.

[Hol01] C. W. Holsapple. Knowledge management support of decision making. *Decision Support Systems*, 31(1):1–3, May 2001.

[Hor05] Ian Horrocks. Owl: A description logic based ontology language. In *Proceedings of the 11th International Conference on Principles and Practice of Constraint Programming (CP'05)*, volume 3709

of *LNCS*, pages 5–8, Sitges (Barcelona), Spain, October 2005. Springer.

[HS01] I. Horrocks and U. Sattler. Ontology reasoning in the SHOQ(D) description logic. In *Proceedings of 17th International Joint Conference on Artificial Intelligence (IJCAI'01)*, pages 199–204, Seattle, Washington, USA, August 2001. Morgan Kaufmann.

[HvdG02] Eveline M. Helsper and Linda C. van der Gaag. Building bayesian networks through ontologies. In *Proceedings of the 15th Eureopean Conference on Artificial Intelligence (ECAI'02)*, pages 680–684, Lyon, France, July 2002. IOS Press.

[HvdG03] Eveline M. Helsper and Linda C. van der Gaag. Ontologies for probabilistic networks. Technical report, Institute for information and computing sciences, Utrecht University, 2003.

[HYC06] Dong Huang, Yi Yang, and J. Calmet. Modeling web services policy with corporate knowledge. In *Proceeding of 2006 IEEE International Conference on e-Business Engineering (ICEBE'06)*, pages 216–223. IEEE Computer Society, October 2006.

[Ibr01] A.M. Ibrahim. Bringing fuzzy logic into focus. *IEEE Circuits and Devices Magazine*, 17(5):33–38, September 2001.

[Jen96] Finn V. Jensen. *An introduction to Bayesian networks*. UCL Press, 1996.

[Jen01] Finn V. Jensen. *Bayesian networks and decision graphs*. Springer, corr. print. edition, 2001.

[KC04] Graham Klyne and Jeremy J. Carroll. Resource description framework (rdf): Concepts and abstract syntax, February 2004. Available from: `http://www.w3.org/TR/rdf-concepts/`.

[KKBB07] Joon Koh, Young-Gul Kim, Brian Butler, and Gee-Woo Bock. Encouraging participation in virtual communities. *Communications of the ACM*, 50(2):69–73, 2007.

[KLP97] Daphne Koller, Alon Y. Levy, and Avi Pfeffer. P-CLASSIC: A tractable probablistic description logic. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI'97)*, pages 390–397, Providence, Rhode Island, July 1997. AAAI Press.

[KM95] Michel R. Klein and Leif B. Methlie. *Knowledge-based Decision Support Systems: with applications in business*. Wiley, 2 edition, 1995.

[KP97] Daphne Koller and Avi Pfeffer. Object-oriented bayesian networks. In *Proceedings of the 13th Annual Conference on Uncertainty in Artificial Intelligence (UAI'97)*, pages 302–313, Rhode, Island, August 1997.

[KP98] Daphne Koller and Avi Pfeffer. Probabilistic frame-based systems. In *Proceedings of the 15th. National Conference on Artificial Intelligence (AAAI'98)*, pages 580–587, Madison, Wisconsin, USA, July 1998. AAAI Press.

[LC05] Kathryn B. Laskey and Paulo C. G. Costa. Of klingons and starships: Bayesian logic for the 23rd century. In *Proceedings of the 21st conference on Uncertainty in Artificial Intelligence (UAI'05)*, pages 346–354, Arlington, Virginia, 2005. AUAI Press.

[LD06] M. Luque and F. J. Diez. Decision analysis with influence diagrams using elvira's explanation facilities. In *Proceedings of the 3rd European Workshop on Probabilistic Graphical Models (PGM'06)*, pages 179–187, Prague, Czech Republic, September 2006.

[Lin87] Dennis V. Lindley. The probability approach to the treatment of uncertainty in artificial intelligence and expert systems. *Statistical Science*, 2(1):17–24, 1987.

[LM01] O. Lassila and D. Mcguinness. The role of frame-based representation on the semantic web. Technical Report KSL-01-02, Knowledge Systems Laboratory. Standford University, Standford, California, 2001.

[LRDS04] P. W. Lord and I. R. Horrocks R. D. Stevens, C. A. Goble. Description logics: Owl and daml+oil. In *Genetics, Genomics, Proteomics And Bioinformatics*. Wiley, 2004.

[LS88] S.L. Lauritzen and D.J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society*, 50(2):157–224, 1988.

[Luk98] Thomas Lukasiewicz. Probabilistic deduction with conditional constraints over basic events. In *Proceedings of the 6th International*

Conference in Principles of Knowledge Representation and Reasoning, pages 380–391, San Francisco, California, 1998. Morgan Kaufmann.

[Luk01] Thomas Lukasiewicz. Probabilistic logic programming under inheritance with overriding. In J. S. Breese and D. Koller, editors, *Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence (UAI'01)*, pages 329–336, Seattle, Washington, USA, August 2001. Morgan Kaufmann.

[MC04] Pierre Maret and Jacques Calmet. Modeling corporate knowledge within the agent oriented abstraction. In *Proceedings fo the International Conference on Cyberworlds (CW'04)*, pages 224–231. IEEE Computer Society, 2004.

[McB04] Brian McBride. The resource description framework (rdf) and its vocabulary description language rdfs. In S. Staab and R. Studer, editors, *Handbook on Ontologies*, pages 51–66. Springer, 2004.

[McC95] John L. McCarthy. What has ai in common with philosophy? In *The 1995 International Joint Conference on AI (IJCAI'95)*, pages 2041–2042, 1995.

[Men01] J.M. Mendel. *Uncertain rule-based fuzzy logic systems: Introduction and new directions*. Prentice Hall, 2001.

[MHC04] Pierre Maret, Mark Hammond, and Jacques Calmet. Virtual knowledge communities for corporate knowledge issues. In *Proceedings 5th International Workshop on Engineering Societies in the Agents World (ESAW'04)*, volume 3451/2005 of *LNCS*, pages 33–44, Toulouse, France, October 2004. Springer.

[ML96] Suzanne M. Mahoney and Kathryn B. Laskey. Network engineering for complex belief networks. In *Proceedings of the 12th Annual Conference on Uncertainty in Artificial Intelligence (UAI'96)*, pages 389–396, San Francisco, CA, 1996. Morgan Kaufmann.

[Mur02] Kevin P. Murphy. *Software Packages for Graphical Models: Bayesian Networks*, 2002.

[MvH04] Deborah L. McGuinness and Frank van Harmelen. Owl web ontology language overview, February 2004. Available from: `http://www.w3.org/TR/owl-features/`.

[MVI95] R. Mizoguchi, J. Vanwelkenhuysen, and M. Ikeda. Task ontology for reuse of problem solving knowledge. In N. J. I. Mars, editor, *Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing*, pages 46–57, Amsterdam, 1995. IOS Press.

[Nak06] Robbie T. Nakatsu. Explanatory power of intelligent systems. In Jatinder N.D. Gupta, Guisseppi A. Forgionne, and Manuel Mora T., editors, *Intelligent Decision-Making Support Systems: Foundations, Applications and Challenges*, Decision Engineering, pages 385–401. Springer, 2006.

[NFF+91] Robert Neches, Richard Fikes, Tim Finin, Thomas Gruber, Ramesh Patil, Ted Senator, and WIlliam Swartout. Enabling technology for knowledge sharing. *AI Magazine*, 12(3):36–56, August 1991.

[NJ96] A. E. Nicholson and N. Jitnah. Belief network algorithms: a study of performance using doamin characterisation. Technical report, Department of Computer Science, Monash University, 1996.

[OGD80] I. Olkin, L. J. Glaser, and C. Derman. *Probability Models and Applications*. McMillian & Co., New York, 1980.

[Pea88] Judea Pearl. On the definition of actual cause. Technical Report R-259, Department of Computer Science, University of California, Los Angeles, 1988.

[Pea97] Judea Pearl. *Probabilistic reasoning in intelligent systems: Networks of Plausible Inference*. Morgan Kaufmann, 2 edition, 1997.

[Pea00] Judea Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge university press, 2000.

[PFCA05] Michael Pool, Francis Fung, Stephen Cannon, and Jeffrey Aikin. Is it worth a hoot? qualms about owl for uncertainty reasoning. In *Proceedings of the workshop of Uncertainty Reasoning for the Semantic Web at the 4th International Semantic Web Conference (ISWC'05),*, pages 1–11, Galway, Ireland, November 2005.

[Pow97] D. J. Power. What is a dss? *The On-Line Executive Journal for Data-Intensive Decision Support*, 1(3), October 1997.

[Pow02] Daniel J. Power. *Decision support systems: concepts and resources for managers*. Quorum Books, April 2002.

[Pow03] Daniel J. Power. A brief history of decision support systems. *DSSResources.COM*, 2003. Available from: `http://dssresources.com/history/dsshistory.html`.

[PP02] Athanasios Papoulis and S. Uònònikrishòna Pillai. *Probability, random variables, and stochastic processes*. McGraw-Hill, 2002.

[PS02] Peter F. Patel-Schneider. Two proposals for a semantic web ontology language. In Ian Horrocks and Sergio Tessaris, editors, *Proceedings of 2002 International Description Logic Workshop (DL'02)*, number 53, Toulouse, France, April 2002. CEUR-WS.org.

[PSHH04] Peter F. Patel-Schneider, Patrick Hayes, and Ian Horrocks. Owl web ontology language semantics and abstract syntax, February 2004. Available from: `http://www.w3.org/TR/owl-semantics/`.

[RC04] Christian P. Robert and George Casella. *Monte Carlo Statistical Methods (Springer Texts in Statistics)*. 2. Springer, 2004.

[Rei91] U. Reimer. *Einführung in die Wissensrepräsentation: Netzartige und schema-basierte Repräsentationsformate*. Teubner, Stuttgart, 1991.

[Ric76] J. R. Rice. The algorithm selection problem. *Advances in computers*, 15:65–118, 1976.

[RN03] Stuart J. Russell and Peter Norvig. *Artificial intelligence: A Modern Approach*. Prentice Hall, 2. ed. edition, 2003.

[RS01] Frank T. Rothaermel and Stephen Sugiyama. Virtual internet communities and commercial success: individual and community-level theory grounded in the atypical case of timezone.com. *Journal of Management*, 27(3):297–312, 2001.

[Rug97] Rudy Ruggles. Knowledge tools: Using technology to manage knowledge better. Technical report, 1997.

[Sar94] V. V. S. Sarma. Decision making in complex systems. *Systems Practice*, 7(4):399–407, 1994.

[SBF98] Rudi Studer, V. Richard Benjamins, and Dieter Fensel. Knowledge engineering: Principles and methods. *Data & Knowledge Engineering*, 25(1-2):161–197, 1998.

[SEMD02] S. Staab, M. Erdmann, A. Mdche, and S. Decker. An extensible approach for modeling ontologies in rdf(s). In Rolf Grtter, editor, *Knowledge media in healthcare: opportunities and challenges*, pages 234 – 253. Idea Group Publishing, Hershey, PA, USA, 2002.

[SG00] A. M. Gupta Solo and M. M. Gupta. Perspectives on computational perception and cognition under uncertainty. In *Proceedings of IEEE International Conference on Industrial Technology 2000*, volume 2, pages 221– 224, Goa, India, January 2000. IEEE Computer Society.

[SH05] Munindar P. Singh and Michael N. Huhns. *Service-Oriented Computing*. Wiley, 2005.

[Sha76] G. Shafer. *The Mathematical Theory of Evidence*. Princeton University Press, 1976.

[Sha86] Ross D. Shachter. Evaluating influence diagrams. *Operations Research*, 34(6):871–882, 1986.

[She92] Prakash P. Shenoy. Valuation-based systems for bayesian decision analysis. *Operations Research*, 40(3):463 – 484, 1992.

[Sim60] Herbert Alexander Simon. *The New Science of Management Decision*. Prentice Hall, 1960.

[Sme96] Philippe Smets. Imperfect information: Imprecision and uncertainty. In Amihai Motro and Philippe Smets, editors, *Uncertainty Management in Information Systems: From Needs to Solution*, pages 189–224. Kluwer Academic Publishers, 1996.

[Smi03] Barry Smith. *The Blackwell Guide to Philosophy of Computing and Information*, chapter 11, pages 155–166. Blackwell, Oxford, 2003.

[Smi04] Barry Smith. Beyond concepts: Ontology as reality representation. In *Proceedings of the 3rd International Conference on Formal Ontology and Information Systems (FOIS'04)*, pages 73–84, Turin, Italy, November 2004. IOS Press.

[SP92] R. D. Shachter and Mark A. Peot. Decision making using probabilistic inference methods. In *Proceedings of the 8th conference on Uncertainty in Artificial Intelligence*, pages 276–283, San Francisco, CA, USA, 1992. Morgan Kaufmann.

[SPO04] Salim K. Semy, Mary K. Pulvermacher, and Leo J. Obrst. Towards the use of an upper ontology for u.s. government and military domains: An evaluation. Technical Report MTR 04B0000063, The MITRE Corporation, Bedford, Massachusetts, November 2004.

[SSSK06] G. Stoilos, N. Simou, G. Stamou, and S. Kollias. Uncertainty and the semantic web. *IEEE Intelligent Systems*, 21(5):84–87, 2006.

[SST+05] Giorgos Stoilos, Giorgos Stamou, Vassilis Tzouvaras, Jeff Z. Pan, and Ian Horrocks. Fuzzy owl: Uncertainty and the semantic web. In *Proceedings of the International workshop on OWL: Experience and Directions (OWL-ED'05)*, Galway, Ireland, November 2005.

[SWM04] Michael K. Smith, Chris Welty, and Deborah L. McGuinness. Owl web ontology language guide, February 2004. Available from: `http://www.w3.org/TR/owl-guide/`.

[TS90] J.A. Tatman and R.D. Shachter. Dynamic programming and influence diagrams. *IEEE Transactions on Systems, Man, and Cybernetics*, 20(2):365–379, 1990.

[vHSW97] G. van Heijst, A. Th. Schreiber, and B. J. Wielinga. Using explicit ontologies in kbs development. *International Journal of Human-Computer Studies*, 46(2-3):183–292, 1997.

[Web86] Max Weber. *Economy and society*. University of California Press, 1986.

[Wen98] Etienne Wenger. *Communities of practice: Learning, meaning, and identity*. Cambridge University Press, 1998.

[Wil97] E. M. Williams. *Modeling intelligent control of distributed cooperative inferencing*. PhD thesis, Department of Electrical and Computer Engineering, Air Force Institute of Technology, 1997.

[Wil05] Jon Williamson. *Bayesian Nets and Causality: Philosophical And Computational Foundations*. Oxford University Press, 2005.

[Woo99] Michael Wooldridge. Intelligent agents. In Weiss Gerhard, editor, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, chapter 1, pages 27–78. The MIT Press, 1999.

[XY01] Y. Xiang and C. Ye. A simple method to evaluate influence diagrams. In *Proceedings of the 3rd International Conference on Cognitive Science*, 2001.

[YC05] Yi Yang and Jacques Calmet. Ontobayes: An ontology-driven uncertainty model. In *Proceeding of the International Conference on Intelligent Agents, Web Technologies and Internet Commerce (IAWTIC'05)*, volume I, pages 457–464, Vienna, Austria, November 2005. IEEE Computer Society.

[YC06a] Y. Yang and J. Calmet. From the ontobayes model to a service oriented decision support system. In *Proceeding of International Conference on Computational Intelligence for Modelling, Control and Automation 2006 (CIMCA'06)*, page 127. IEEE Computer Society, November 2006.

[YC06b] Yi Yang and J. Calmet. Ontobayes approach to corporate knowledge. In *Proceeding of the I6th International Symposium on Methodologies for Intelligent Systems (ISMIS'06)*, volume 4203/2006 of *LNCS*, pages 274–283. Springer, September 2006.

[Zad65] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338–353, June 1965.

[Zha98] Nevin Lianwen Zhang. Probabilistic inference in influence diagrams. *Computational Intelligence*, 14(4):475–497, November 1998.

[ZP92] Nevin Zhang and David Poole. Stepwise-decomposable inuence diagrams. In *Proceedings of the 4rth International Conference on Knowledge Representation and Reasoning*, pages 141–152, 1992.

[ZP94] Nevin Lianwen Zhang and David Poole. A simple approach to bayesian network computations. In *In Proceedings of the Tenth Canadian Conference on Artificial Intelligence*, pages 171–178, May 1994.

Index

Curriculum Vitae

1977.02.03	Born in Shanghai, China
1983 – 1989	Elementary school in Shanghai
1989 – 1992	Middle school in Shanghai
1992 – 1995	High school in Shanghai
1995 – 1999	Studies in structural engineering at TongJi-University Bachelor: July 1999
2000 – 2004	Studies in computer science at the University of Bremen Diploma: August 2004
since October 2004	Ph.D. student at the University of Karlsruhe, Institute for Algorithms and Cognitive Systems