2012

A task ontology model for domain independent dialogue management

Guoying Liu
University of Windsor

A TASK ONTOLOGY MODEL FOR DOMAIN INDEPENDENT DIALOGUE MANAGEMENT

by

Guoying Liu

A Thesis
Submitted to the Faculty of Graduate Studies
through the School of Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Master of Science at the
University of Windsor

Windsor, Ontario, Canada

2012

A Task Ontology Model for Domain Independent Dialogue Management

by

Guoying Liu

APPROVED BY:

_____

Dr. G. Zhou
Faculty of Education

_____

Dr. J. Morrissey
School of Computer Science

_____

Dr. X. Yuan, Advisor
School of Computer Science

_____

Dr. A. Ngom, Chair of Defense
School of Computer Science

03 February 2012

AUTHOR'S DECLARATION OF ORIGINALITY

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

ABSTRACT

Dialogue systems have been a rapidly growing area in both scientific research and commercial application since 1990s. They can be applied in various fields, such as business, healthcare and education, etc. Due to its complexity, the design and development of a dialogue system is time consuming and costly. It is highly desirable for a generic dialogue system, especially dialogue management that is independent of specific domains. Methods or architecture for domain independent dialogue systems have been proposed by previous research in literature, however each of them has its own limitations and none has been widely adopted. This paper presents a new approach, a task ontology model for domain independent dialogue management. An abstract task ontology model is developed and based on this model a generic dialogue manager is created. Knowledge about a specific task is modeled in its task ontology and retrieved by an ontology reasoning component situated in the dialogue manager. Thus the dialogue manager is task or domain independent. A dialogue system is developed based on the proposed method and experimented with two different tasks: the book borrowing and the online train ticket booking. The experiment results indicate that the dialogue system can be readily applied to tasks from different domains without any modification. This paper has implications on future research and development of domain independent dialogue systems. It also contributes to the knowledge and dialogue system reuse and will have impact on the application of dialogue systems in a wider range of areas.

DEDICATION

This thesis is dedicated to my dear husband and daughter for their endless love and support.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisor, Dr. Xiaobu Yuan for his patience, enthusiasm and instruction on my study and research in the long journey towards the Master's Degree in Computer Science. Without his guidance, I could not image accomplishing this thesis.

I also want to express my appreciation to Dr. Morrissey for her insightful comments and encouragement throughout the work.

In addition, I would like to thank my classmates and friends who kindly helped and encouraged me during the completion of the thesis as well as my colleagues, especially Gwen and Cathy for their continuous support on my study.

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION

A computer system that is able to dialogue with human beings has been a research topic in Artificial Intelligence for many years. However, it is not until the past two decades or so, with the advancement of speech technology, language processing and dialogue modeling, that spoken dialogue systems have been developed and entered into commercial use. [1] In recent years, research on dialogue systems has been further extended to multimodal dialogue systems which support not only speech but also text, graphics, gestures and other user input modes as well as multimedia system output. A multimodal dialogue system typically consists of components that process input information and generate output message as well as a central piece of dialogue management that coordinates the entire system and controls the dialogue process. [2]

1.1 PROBLEM STATEMENT

Dialogue systems have had a wide range of applications, such as customer services, sales, technical support, personalized services and training in various fields, including healthcare, entertainment, government, education and business, etc. [3-4] It has great potential to many other fields as well. On the other hand, the design and development of a dialogue system requires the collaboration of both experts with domain knowledge and computer specialists on dialogue management systems. It is often time consuming and costly to develop a workable spoken or multimodal dialogue system. [2, 5] It is highly desirable to create a generic dialogue system, especially the dialogue management module which is independent of tasks from different domains. [6-7]

The majority of related research in literature focused on components other than the central dialogue management of a dialogue system. Only a few of them addressed the domain independent dialogue management. The first main approach is the agent-based architecture in which spoken dialogue agents are adopted to handle different domains. So adding or applying to a new domain means to construct a new spoken dialogue agent for that domain instead of rebuilding the entire system. [5, 8] The second approach is to create domain experts to control dialogue in each domain. The central module of the system selects the domain and controls the domain experts. [9] The third one is to adopt a service manager for each specific task. [10] All these methods separate what is used to deal with specific domains or tasks from other constituents of the dialogue management module. But the creation of the dialogue agent, domain expert or service manager for a particular task still requires the involvement and collaboration of both dialogue system developers and experts for specific domains. The most recent attempt is the task tree based dialogue management framework [11]. In this framework, a task or domain knowledge is modeled in a task tree, and then the dialogue plan is formed on the basis of the structure of the task tree. It allows domain experts to focus on the building of task tree while system specialists on the creation of dialogue systems. However, different tasks are presented by different task trees and a new dialogue plan would be created for any new task. So the development of a dialogue manager is still dependent on the task or domain.

This thesis proposes a task ontology model for domain independent dialogue management. It intends to apply ontology and task model into the dialogue system design to accomplish the domain independent dialogue management. Ontology has been adopted in various dialogue systems. Besides Natural Language Processing, recent work on the application of ontology in dialogue systems mainly uses ontology as the knowledge source, such as the

ontology-driven dialogue system for agent training [12], the ontology-based framework for health counseling dialogue systems [13], the CookCoach dialogue system in which the ontology for cooking domain can be integrated in the dialogue system [14], and the ontology based requirement model for interactive requirement elicitation [15]. However each of them focuses on a particular domain and none supports domain independent dialogue systems.

Knowledge of task modeling is also combined in the proposed model. The goal of a dialogue management system is to achieve a task [16]. A task, including its activities and their relationships can be represented by a task model [17]. A number of task model representations have been proposed, including HTA (Hierarchical Task Analysis), GOMS (Goals Operators Methods Selectors), GTA (Groupware Task Analysis) and CTT (ConcurTaskTree), etc. These models are mainly designed for task analysis rather than integrating to a dialogue system.

In the proposed model, the knowledge of a specific task is modeled in its task ontology which is independent from dialogue control. An abstract task ontology is created and a generic dialogue manager is developed in which the abstract ontology is used to direct system interaction with users. Some techniques and concepts of task modeling are adopted in the ontology along with those new classes and relations created in the model. The abstract task ontology can be instantiated to task ontologies for tasks in different domains. Through the reasoning component which connects task ontologies, the generic dialogue manager can accomplish a particular task through interaction with users without any modification. Thus the design and development of dialogue management and dialogue systems are isolated from domain knowledge for a specific task; and the dialogue system can be applied to new areas directly. Two case studies on different tasks from different domains are conducted to test and demonstrate the effectiveness of the proposed method.

1.2 CONTRIBUTIONS

This thesis has made the following contributions. It proposes a task ontology model which can be instantiated to task ontology for specific tasks. Based on the task ontology model, the structure of the dialogue manager is also modified to include a generic dialogue management component and an ontology reasoning component. The ontology reasoning component retrieves knowledge specific to a task from its task ontology. Thus the dialogue manager is task independent and a dialogue system based on it can apply to new tasks or modified existing tasks without any modification. Furthermore, the task ontology model contains a variety of types of relationships between tasks and their subtasks, including decomposition, sequential and alternative relationships and their properties. It also takes into account the context for a task completion which allows for handling more complicated tasks in a more flexible way compared to previous approaches.

The proposed method promotes knowledge and dialogue system sharing and reuse among different domains. The adoption of ontology and the proposed task ontology model have implications on future research and development of domain independent dialogue systems. It also has impact on the application of dialogue systems in a wider range of areas.

1.3 STRUCTURE OF THE THESIS

This paper proposes an ontology task model to achieve domain independent dialogue management in a dialogue system. It is structured as follows. Chapter 2 overviews dialogue systems and various approaches to dialogue management. Chapter 3 describes previous work related to this thesis. Section 3.1 focuses on research on domain independent dialogue management. Section 3.2 introduces the concept of ontology, ontology languages and its

application in dialogue systems; and Section 3.3 discusses task models and their presentations. Chapter 4 illustrates the proposed method - the ontology-based task model and the modified dialogue manager based on the proposed model. Section 4.1 introduces the task ontology model and what are new about it compared with previous work. Section 4.2 details the proposed task ontology model, including the concepts, relationships and rules defined in the ontology and its instantiation. Section 4.3 presents the domain independent dialogue manager based on the task ontology model. Complexity analysis is also conducted. Chapter 5 is the implementation of the proposed method. Section 5.1 describes the abstract task ontology. Section 5.2 presents the implementation of the dialogue system based on the dialogue manager introduced in Section 4.3. Chapter 6 studies two application cases from different domains using proposed method. Section 6.1 is about the library borrowing service; and Section 6.2 is the online train ticket booking service. Both sections include the construction of the task ontology for each task and demonstrate how the dialogue system works for different tasks. Chapter 7 draws conclusion and indicates future work.

CHAPTER 2 PRELIMINARY

2.1 DIALOGUE SYSTEMS

Dialogue is a conversation between two or more human beings or machines. [2] A dialogue system is a computer system which communicates with human users in a natural manner. Dialogue systems are involved in dialogue with human users to achieve specific tasks, so they are mainly task oriented. [16]

The majority of the research focused on spoken dialogue systems that support spoken natural language. In recent years, it has been extended to multimodal dialogue systems which support text, audio, video and other modes for both user input and system output. [2] According to Bui [2], a typical multimodal dialogue system consists of the following modules: Input, Fusion, Dialogue Manager, Knowledge Base, Fission and Output. Among these components, the Dialogue Manager plays a key role in a dialogue system. Fig. 2.1 illustrates the architecture of a dialogue system.

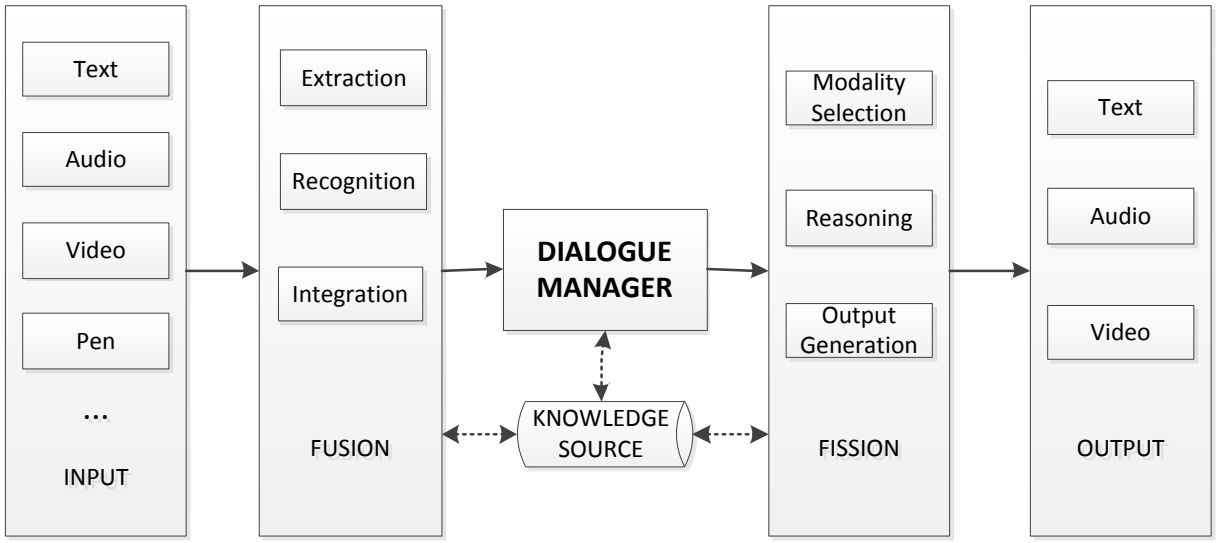

Figure 2.1 Architecture of a Dialogue System

In a multimodal dialogue system, the Input module can handle different input modes, such as text, speech, pen, touch screen and gestures, etc.

The Fusion module extracts, recognizes and integrates various modes of information from Input module, and then sends to Dialogue Manager. It may also consult with Knowledge Base.

The Knowledge Base is used by Dialogue Manager, Fusion and Fission modules. It manages a number of types of knowledge, such as dialogue history, general dialogue knowledge, specific domain knowledge, and information about tasks and users.

The Fission module is responsible for modality selection, reasoning and output coordination. It generates output information in different modes.

The Output module is to present information from Fission module in various output modalities.

The Dialogue Manager is the central management module for both spoken dialogue systems and multimodal dialogue systems. It coordinates with other components, such as Fusion, Fission and Knowledge Base. It also identifies user's intention to help the system understand what the user really wants. It controls dialogue flow through deciding what to respond to user's inputs. [18]

2.2 Dialogue Management Approaches

In the past two decades, a number of approaches have been proposed for dialogue management. Each approach has its own strengths and drawbacks. According to the methods used to control dialogue with users, these approaches can be classified into the following four categories: [1-2]

1) Finite state-based approach

   This approach is the most common and simplest dialogue management strategy. In a state-based dialogue system, the dialogue states and system utterances are predetermined and the dialogue is directed by the system through a set of predetermined sequence of questions. It is suitable for implementing simple dialogue system with well-structured tasks, but lacks of flexibility, naturalness and applicability to other domains.

2) Frame-based approach

   Frame-based approach is an improvement to the finite state-based dialogue management. The system operates like a slot-filling task to gather predetermined set of information. Frame-based dialogue management allows some degree of mixed-initiative and multiple slot fillings, which is more flexible, efficient and natural compared with finite state-based approach. It is capable of handling recognition errors and misunderstandings during dialogue with users.

3) Information state-based and probabilistic approaches

   In information state-based dialogue management, a set of states are predefined and the dialogue state changes according to the update rules and dialogue strategies. As an extension to the information state-based approaches, the probabilistic dialogue management utilizes probabilistic techniques to try to improve the performance of dialogue management. [18] In this type of approaches, dialogue is modeled as a Markov Decision Process (MDP) [19-20] or a Partially Observable Markov Decision Process (POMDP) [21-23] to address the uncertainty in human computer interaction. Other methods, such as Bayesian Networks [24], Decision Networks [25] have been studied by researchers as well.

4) Plan-based or agent-based approaches

Plan-based or agent-based approaches are based on plan-based theories of speech act and dialogue. [26-27] This type of dialogue management supports more complicated dialogue tasks than the above three approaches. Accordingly, it involves higher level of complexity.

The first two types of approaches are the most commonly adopted methods in existing dialogue systems. A number of practical systems have been constructed based on them. The other approaches, including information state-based and probabilistic approaches as well as plan-based or agent-based approaches are relatively complicated and need to be improved for practical use. In this thesis, a frame-based dialogue system is implemented to demonstrate how the domain independent dialogue management can be accomplished by the proposed method.

CHAPTER 3 RELATED WORK

3.1 DOMAIN INDEPENDENT DIALOGUE MANAGEMENT

A number of dialogue systems have been implemented using various dialogue control strategies, such as the Nuance automatic banking system [1], the Philip automatic train timetable information system [28], the Circuit-Fix-It Shop system [29], the E-form classified advertisements for used cars [30], Carnegie Mellon Communicator system [31], MATCH system for multimodal access to city help [32], Immersive Virtual Worlds [33], Virtual Music Center [34], Smartkom public telephone booths [27], and so on. These systems have been deployed in a variety of application areas, including customer services, technical support, personalized services, planning, problem solving and training in the fields of healthcare, entertainment, government, education and business, etc. [3] It is expected that dialogue systems would have a much wider application and acceptance in practical use in future. [7]

On the other hand, it takes a lot of efforts to design and develop a workable dialogue system due to the system complexity. It often requires the collaboration of both experts with domain knowledge and computer specialists on dialogue management systems, which is expensive and time consuming. For a multimodal dialogue system, the building process becomes even more complicated. It is highly desirable to create a generic dialogue system, especially the dialogue management module which can be readily applied to tasks from different domains. [2, 5-6]

A number of research projects have targeted on domain independent dialogue systems in the past decade. Many of them focus on modules related to natural language processing, such as speech recognition or semantic representation, rather than dialogue management. Examples of

these projects include the domain independent spoken dialogue platform that uses key-phrase spotting technique based on combined language model [35], the two-layer architecture of semantic interpretation and domain reasoning for a multi-domain dialogue system [36], robust domain selection against out-of-grammar utterances for the multi-domain spoken dialogue system [37], and a multi-domain spoken dialogue system which is able to respond to user requests across multiple domains against speech recognition errors [38], etc.

A series of generic architecture have been proposed aiming for the design of domain independent dialogue systems. In TRIPS project, Allen et al. [16] suggested a generic dialogue shell – a generic dialogue infrastructure for spoken dialogue system. In this framework, the main components of a spoken dialogue system, including speech recognition, natural language processing, response generation and dialogue management, are constructed in a way that can be adapted to new applications to other tasks or domains by specifying the domain and task operations. Based on this framework, Allen et al. [39] further presented an architecture that separates linguistic and discourse knowledge from task and domain specific information. It clarifies the responsibilities of individual components and improves the system portability to new tasks and domains. Bui [2] illustrated a module-based, general architecture of a multimodal dialogue system. It consists of modules for input processing, output generation and dialogue control. Each module has its own function and works together with other components under the coordination of the dialogue manager. However, the architecture does not address how to build a domain independent dialogue management module. It is a conceptual framework rather than a practical model. More work is needed under this framework in order to create a domain independent dialogue system.

Only a few studies focused on multi-domain or domain independent dialogue management. There are four major types of approaches in literature. The first is the agent-based architecture. Lin et al. [5] suggested a multi-agent architecture for multi-domain spoken dialogue systems. This architecture applies different Spoken Dialogue Agents (SDA) to handle different domains and a User Interface Agent (UIA) for users to access the correct SDA through a domain switching protocol. The SDAs are developed independently and then cooperate with one another to help achieve user's multiple goals. The multi-agent architecture has been adopted by a number of multi-domain systems. [8]

Another approach is to create a service manager for each specific task. [6, 10] The proposed dialogue system architecture consists of input/output manager, dialogue manager and service manager. The service manager handles all domain specific information.

The third method adopts domain experts to support multi-domain dialogue management. Komatani et al. [38] use domain experts to control dialogues in each domain and a central module to select proper domain and control the domain experts in a multi-domain dialogue system. Lee et al. [40] introduces a multi-domain dialogue system using the example-based dialogue modeling framework and the domain spotting technique.

The common problem of the abovementioned approaches is that they all need to develop a new piece of component, such as a new agent, a new service manager or a new domain expert when adding or adapting to a new task or domain in a dialogue system.

The last approach to domain independent dialogue management is the agenda or task tree based generic dialog modeling [7]. In this approach, large tasks are decomposed into smaller and more easily handled subtasks [41-44]. The RavenClaw dialogue management framework is the most recent one based on this approach. [11] It contains a two-tier, plan-based architecture in

which domain-specific aspects of the dialog control logics are separated from domain independent ones. In this framework, a task or knowledge is modeled in a task tree; and then the dialogue plan for the particular task is formed based on its task tree. Different tasks will have different dialogue plans, so the dialogue management is still task specific.

3.2 ONTOLOGY

3.2.1 OVERVIEW

According to Gruber [44], ontology is a formal, explicit specification of a shared conceptualization. A conceptualization is an abstract model of the world that is represented. In other words, ontology provides a shared vocabulary for representing knowledge of a task or a domain. It defines concepts, properties of each concept and constraints on their use within a domain.

People may have different goals to develop ontology, such as:

1) To share common understanding of the structure of information among human beings or software
2) To enable the reuse of domain knowledge
3) To make assumptions on a domain explicit
4) To separate domain knowledge from operational knowledge
5) To analyse common knowledge within a domain [45]

The main purpose of ontology is to support knowledge sharing and reuse. It can be used in a variety of areas, such as human communication, system interoperability and system engineering.

Ontology usually consists of the following components: concepts within a domain, properties of each concept, and relations or constraints between concepts or their properties, etc. According to Noy and McGuinness [45], there are a number of steps typical for ontology construction as follows:

1) Defining classes representing the concepts of the domain that is being described
2) Arranging the defined classes in a taxonomy or superclass-subclass relationship
3) Defining the properties of classes and instances and the constraints on their values
4) Filling in instances, their properties, and the values for each property.

Once all classes, properties, instances and their relationships are properly defined, the ontology, either task ontology or domain ontology is developed.

3.2.2 Ontology Languages

Ontology is constructed by ontology languages. An ontology language usually introduces concepts and their properties, relationships and additional constraints. Ontology languages may be simple, frame-based or logic-based. [46] Simple ontology languages only have concepts and taxonomies. Frame-based ontology languages are more structural compared to simple ones and contain concepts and properties. Logic-based ontology languages are more sophisticated and expressive. It includes two common types. One is the First Order Logic based ontology language, such as Common Logic (Common Logic Standard, http://common-logic.org/), Cycl (the Syntax of Cycl, http://www.cyc.com/cycdoc/ref/cycl-syntax.html), and KIF (Knowledge Interchange Format, http://www.ksl.stanford.edu/knowledge-sharing/kif/). This type of language allows general predicates. Another type is the Description Logic based ontology language that describes knowledge in terms of concepts and their relations. This type of ontology languages has

inference capability. In other words, it is possible to reason about knowledge in Description Logic based ontology languages, such as inference of implicit class membership, inference of implicit subclass and equivalence of classes, and check consistency by using a reasoner. Generally speaking, the richer a language is, the more inefficient it supports reasoning.

Web Ontology Language (OWL), which is recommended by W3C, is one of the Description Logic based languages. [47] It is a Web standard for processing information on the Web. OWL is written in XML and designed to be interpreted by computers. There are three sublanguages of OWL: OWL Lite, OWL DL and OWL Full. Among them, OWL Full is the most expressive language, but no existing reasoner supports its inference. OWL Lite and OWL DL are decidable but their expressivity is sacrificed. [48] So a decidable language, the DL-safe Semantic Web Rule Language (SWRL) was proposed to supplement OWL DL and OWL Lite with Horn-like rules. [49]

It is possible to infer implicit knowledge in an ontology through Description Logic reasoning. There are already many stable reasoners available for OWL DL reasoning, such as Pellet (http://clarkparsia.com/pellet/), FaCT++ (http://owl.man.ac.uk/factplusplus/), HermiT (http://hermit-reasoner.com/), Jena (http://jena.sourceforge.net/inference/) and RacerPro (http://www.franz.com/agraph/racer/). These reasoners can be plugged into the Protégé, an OWL ontology development platform (http://protege.stanford.edu/). Pellet supports reasoning with OWL DL and SWRL DL-safe rules. In this thesis, Protégé will be adopted to construct ontology and Pellet for ontology reasoning.

3.2.3 Application of Ontology in Dialogue Systems

Ontology has been utilized for years for language interpretation and generation in natural language systems. Gatius and Gonzalez [50] presented a dialogue system that used ontology for improving the interaction between the system and the user, the processing of user's interventions and the generation of system's messages. This paper focuses on natural language processing. The ontology is used in input processing and output generation modules, not the dialogue management module in the dialogue system. Sonntag and Romanelli [51] presented an ontological syntactic structure of multimodal question answering results for Question and Answering systems. In this paper, ontology serves as a communication interface between natural language processing components. The study by Milward and Beveridge [52] indicated that ontological information can be used for dialogue clarification. In this study, the ontology is used to explore the distinctions between monologue and dialogue rather than to design a dialogue system.

In dialogue systems, more recent work uses ontology to capture domain knowledge and to provide knowledge source for the system. Flycht-Eriksson and Jonsson [53] proposed an architecture of dialogue systems in which ontologies are applied for various tasks, including question analysis, dialogue interaction and information extraction. In the architecture, ontologies provide the shared knowledge sources for the cooperation of the two main parts of the dialogue systems: Interaction component and Information Processing component.

Van Oijen et al. [12] presented an ontology-driven, goal-based dialogue system for agent training. In the system, ontologies are used for capturing domain knowledge and the knowledge of agent role. They provide a shared data set for concepts and their attributes exchanged between

agents. The use of ontology enables the tight coupling between the agents and the simulation and allows the reuse of agent ability.

Bickmore et al. [13] proposed an ontology-based framework for health counselling dialogue systems. The framework implements an ontology of behaviour change concepts and a task model for the intervention of system and users. The ontology captures the knowledge in health behaviour change counselling. In this approach, the ontology and task model is disconnected and the ontology serves as domain ontology only.

Pardal and Mamede [14] intended to extend a dialogue system to new knowledge domains by using ontologies. They created the Cookcoach dialogue system on the Carnegie Mellon University OLYMPUS framework [54] which incorporates the RAVENCLAW dialogue management [11]. An ontology, OntoChef was developed for the cooking domain. The OntoChef contains modules specifically for cooking domain, including actions, food, recipes, utensils, units and measure. The Cookcoach supports different recipes; however the model only supports cooking domain.

Zhang [15] proposed an ontology-based requirement model for interactive requirement elicitation. In this model, concepts, including function, quality and softgoal along with relationships among these concepts and rules are set up particularly for requirement elicitation. It is hardly to apply the model to tasks in other domains, such as book borrowing service or train ticket booking, etc.

In order to make the POMDP based dialogue manager capable for practical dialogue systems, Young et al. [55] proposed the Hidden Information State (HIS) model. Ontology rules are adopted in the HIS model as task model of dialog scenario and represent groups of similar

belief states as partitions. The model uses tree structure to represent user goal. The tree structure is task specific. So do the ontology rules based on that.

Heinroth et al. [56] created an adaptive spoken dialogue manager, OwlSpeaker which supports multitasking in a dialogue system. In OwlSpeaker, the Spoken Dialogue Ontologies (SDOs) are developed to model knowledge base for different domains. Each SDO contains the Speech and State subclasses and their individuals are arranged in tree like structure. The OwlSpeak allows for switching among different interrelated tasks. However, the purpose of the paper is to support multitasking in a dialogue system, not to create domain independent dialogue system.

None of the abovementioned work supports domain independent dialogue management. Different from previous approaches, this thesis develops a task ontology model which can separate domain knowledge from dialogue control in a dialogue system. So the dialogue management is independent of domain knowledge and the dialogue system based on the model is domain independent.

3.3 TASK MODELS

Task models describe the possible activities and their relationships in tasks. They are produced by task analysis which is the process to gather data about how people perform a task and to acquire an insight understanding of these data. [17]

Task models have been used in supporting the design of user interfaces and interactive systems. The main purposes of task models include:

1) To help better understand the application domain
2) To record the result of interdisciplinary discussion

3) To support effective design and usability evaluation

4) To provide documentation and user support [57]

Task analysis and modeling was first introduced in psychology. The Hierarchical Task Analysis (HTA) by Annet and Duncan [58] was the first important method proposed for task analysis and modeling. [17] Since then, task models have become more formal and a number of representations have been developed, including HTA, Goals Operators Methods Selectors (GOMS) [59], User Action Notation (UAN) [60], Groupware Task Analysis (GTA) [61], task world ontology [62] and ConcurTaskTrees (CTT) [57], etc.

Among the representation methods for task models, HTA is the basis for many other methods. In HTA, tasks are described in terms of operations and plans. Operations are activities to reach a goal while plans are their conditions. Tasks are hierarchically decomposed into a series of subtasks, thus a task hierarchy is modeled. In HTA, goal hierarchy is not explicitly represented.

GOMS represents tasks as well as their goals. This method was first presented by Card et al in 1983[59]. Since then, GOMS analysis has been one of the most widely known task analysis technique and many variants have been developed. [63] In GOMS, activities in a task are described in terms of Goals, Operators, Methods, and Selection rules. Goals are the states for users to pursue; and methods are possible ways to reach a goal. Operators describe the steps of a method. The selection rules are the criteria for a method. However, GOMS is mainly textual. It doesn't consider user errors and the possibility of interruptions. GOMS represents only sequential tasks.

UAN [60] is a user and task oriented notation describing user behavior for the user interface design. UAN identified a number of temporal relationships among tasks, including:

sequence, waiting, repeated disjunction, order independence, interruptibility, one-way or mutual interleavability and concurrency. These relationships can be explicitly and precisely represented in UAN task descriptions. In UAN, tasks that cannot be decomposed are defined as basic tasks. The basic tasks are specified by user actions, system feedback and the state of the system. The main disadvantage of UAN is that it uses textual syntax and lacks of tool support. [17]

GTA focuses on group users or organizations and their activities rather than single users. [61] In GTA, a task model contains three different aspects of the task world, including agents, work and situation. Agents refer to humans or systems that play different roles. Work is performed by tasks; and tasks can be identified at different levels of complexity, such as the unit task and the basic task. Unit tasks and basic tasks can be further decomposed into system actions or user actions. The situation of a task world includes the environment for the performance of a certain task and the description of the objects in the environment. The description of an object includes its structure and attributes.

Welie [62] developed a task world ontology that describes concepts and their relationships based on GTA. The ontology includes concepts of task, goal, role, agent and event. The task can be unit task or basic task. A unit task should consist of one or more basic tasks. These concepts have different relationships between each other, such as uses, triggers, plays, performed_by, has, subtask/subgoal, subrole, influence, responsible and used_by. It has been used for user interface design. The limitation of GTA along with task world ontology and other GTA variants is that they only support sequential tasks. The temporal relation between tasks cannot be precisely specified under this type of task model.

ConcurTaskTrees (CTT) [57] resolves the problem of specification of time relation in GTA. CTT is a graphical notation for the description of tasks and their relationships. It has

hierarchical structure and graphical syntax. CTT has a rich set of operators to specify the temporal relationships between tasks, including enabling, disabling, choice, concurrency, interruption, optionality and iteration, etc. The CTT associated tool, ConcurTaskTrees Environment (CTTE) [64] provides methods to describe single user as well as cooperative task models. So CTT is also an improvement to GOMS.

All abovementioned representations are used mainly as tools for task analysis. They are generally adopted to support the process of system design and do not contain sufficient and necessary information to automatically generate interactive systems. [65] In this paper, a task ontology model is proposed to integrate into the dialogue management system. The task ontology is used to automatically direct the dialogue process.

CHAPTER 4 PROPOSED METHOD

4.1 INTRODUCTION

This chapter presents an ontology-based task model for domain independent dialogue management systems. As stated in Section 3.1, the goal of a domain independent dialogue system is to separate the development of dialogue systems from knowledge specific to a task in a particular domain. In order to accomplish the goal, this paper proposes a new approach - a task ontology model for domain independent dialogue management. In this approach, an abstract task ontology model is developed. Based on this model, the dialogue manager is modified. Knowledge about a specific task is modeled in its task ontology through ontology instantiation of the abstract model. Information in the task ontology can be retrieved by an ontology reasoning component situated within the dialogue manager. Thus the specific task or domain knowledge is isolated from the dialogue management in a dialogue system. The modified dialogue manger is task independent; and the dialogue system based on this method is domain independent and can be readily applied to new tasks or new domains.

The proposed task ontology model is different from previous task modeling approaches discussed in Chapter 3. First, it combines knowledge of task modeling and ontology. In this model, ontology is used to represent concepts and their relationships in a task model; while graphical or textual representations are commonly adopted in previous task modeling approaches. Secondly, the proposed task ontology can be retrieved by dialogue systems directly. It is used to automatically control interaction with the user in a dialogue system; whereas previous task modeling methods are mainly used as tools for task analysis.

This model is expanded from the task world ontology by Welie [61]. The concepts of agent, object and task and their relationships in the task world ontology have been adopted in this model. However the task world ontology does not detail the decomposition of a task and the possible relationships among its subtasks. To overcome this problem, this model uses Hierarchical Task Analysis [17] method to decompose task into a set of subtasks. It adopts the concepts of basic task, unit task and their specifications of user action and system feedback from UAN [59] and CTT [56]. Some of the temporal relationships among tasks presented in UAN [59] and CTT [56], such as sequence and choice, are adapted and modeled into relations in the task ontology model as well. This model also develops concepts, relationships and properties which cannot be represented in previous task models but suitable for an ontology, such as disjoint relationship and data properties. The proposed task ontology model is detailed in Section 4.2. The construction of the ontology follows the steps described in the Section 3.2.1.

How would the task ontology model be used in a dialogue manager to help accomplish the domain independence? Section 4.3 illustrates the structure of the generic dialogue manager based on the proposed task ontology model. It includes a generic dialogue management (DM) component and an ontology reasoning component. The ontology reasoning component is invoked by the generic DM. It retrieves task or domain knowledge from the task ontology and reasons to gather information from dialogue with users. The knowledge about a specific task is modeled in the task ontology. Thus the design and development of the generic DM and the ontology reasoning component is independent of specific tasks or domains.

4.2 Proposed Task Ontology Model

4. 2.1 Concepts

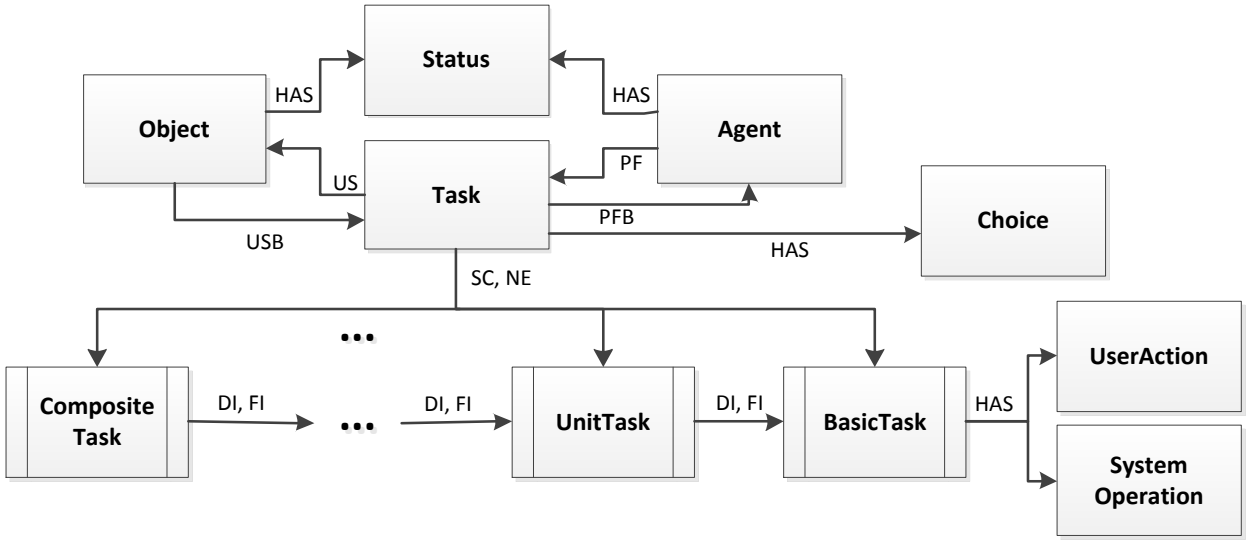

Figure 4.2 the proposed task ontology model

The concepts in this task ontology model are illustrated in Fig. 4.2 with class and subclass diagram notations. Object and Agent represent the context in which a task is executed. Object is used by a task; and agent performs a task. Both Object and Agent have Status. Tasks are activities that help users to achieve their goal. There are three types of tasks: CompositeTask, UnitTask and BasicTask. BasicTask has UserAction and SystemOperation. The UserAction models the action users may take in a task; while SystemOperation is the operation that system would have. Some tasks may have Choice for users to choose and the selection users makes would guide the system to move forward. All concepts created in the task ontology model are described as follows:

1) Task: are activities to reach a certain goal. There are three types of task defined in this model: CompositeTask, UnitTask and BasicTask representing the different decomposition level in a task hierarchy.
    a) CompositeTask: is a task that users want to achieve his goal. CompositeTask is on the top level of task decomposition. It is further decomposed into one or more UnitTasks.
    b) UnitTask: is in the middle of task decomposition hierarchy. There can be one or more levels of UnitTask. UnitTask composes CompositeTask. It is also decomposed into one or more BasicTasks.
    c) BasicTask: is in the bottom of a task hierarchy. BasicTask composes UnitTask. It is atomic and includes UserAction and SystemOperation.
2) Agent: can be humans or machines. Agent performs Task.
3) Object: refers to physical or non-physical entities used by Task.
4) Status: both Object and Agent have a status. The value of Status can be available or unavailable.
5) Choice: are options that users may have in a task. What users choose will direct the next step the system takes.
6) UserAction: refers to actions that users should take in order to fulfil a task. It applies only to BasicTask.
7) SystemOperation: is an action done by system in a task. Similar to UserAction, it is only meaningful for BasicTask.

4.2.2 RELATIONSHIPS

Task is usually represented as a tree-like task hierarchy. In a task tree, the most important relationships between tasks and their subtasks are: (1) supertask and subtask relationship: a task may decompose into several subtasks. This is modeled as DecomposeInto; (2) temporal relationship: the execution order of a set of tasks or subtasks. This is represented as First and Next; and (3) alternative relationship: for some tasks, only one of them would be executed to complete a super task. This kind of relationship is represented with hasChoice.

Besides hasChoice, two other "has" relationships are constructed in the task ontology model. One is hasUserAction. It is used to store information on what kind of actions users should take in order to complete a task. It only applies to BasicTask. Another is hasSystemOperation. It is similar to the relationship hasUserAction which only applies to BasicTask. HasSystemOperation is used to store information on operations the system will take to complete a task.

The proposed model also considers the context for a task. The concepts Object and Agent are defined to represent the objects or tools a task needs and the humans or machines performing a task respectively. Their relationships with Task are represented with Uses, UsedBy, Performs and PerformedBy. The relationship hasStatus is developed to represent the Status of Object or Agent.

These relationships are illustrated in Fig. 4.2. Their notations are listed as follows:

SC: Subclass(x, y), x∈{ Task }, y∈{ CompositeTask , UnitTask, BasicTask }

DI: DecomposeInto(x, y), x∈{CompositeTask}, y∈{ UnitTask }

DI: DecomposeInto(x, y), x∈{UnitTask}, y∈{ BasicTask }

FI: First(x, y), x∈{CompositeTask }, y∈{UnitTask}

FI: First(x, y), x∈{UnitTask}, y∈{BasicTask}

NE: Next(x, y), x∈{Task}, y∈{Task}

US: Uses(x, y), x∈{Task}, y∈{Object}

USB: usedBy(x, y), x∈{Object}, y∈{Task}

PF: Performs(x, y), x∈{Agent}, y∈{Task}

PFB: performedBy(x, y), x∈{Task}, y∈{Agent}

HAS: hasChoice(x, y), x∈{Task}, y∈{Choice}

HAS: hasUserAction(x, y), x∈{BasicTask}, y∈{UserAction}

HAS: hasSystemOperation(x, y), x∈{BasicTask}, y∈{SystemOperation}

HAS: hasStatus(x, y), x∈{Object, Agent}, y∈{Status}

In OWL, concepts are constructed as classes and their relationships are represented by properties. There are two main types of properties: object properties and data properties. In the proposed task ontology model, the relationships of DI, FI, NE, US, USB, PF, and PFB are constructed as object properties and the HAS relationship is developed as data properties, including hasChoice, hasStatus, hasUserAction, hasSystemOperation.

1) SC: is the superclass-subclass relationship. Task is the superclass. Its subclasses, CompositeTask, UnitTask and BasicTask represent different decomposition levels of a task.
2) DI: DecomposeInto. CompositeTask is on the top of a task hierarchy. It can be decomposed into one or more UnitTask. UnitTask is in the middle of a task hierarchy which can be further decomposed into one or more BasicTask.

3) FI: is the first UnitTask containing in a CompositeTask decomposition or the first BasicTask in a UnitTask decomposition. It is used to define the execution order within a CompositeTask or UnitTask.
4) NE: Next defines which is the next task or action the system will take after the current task is completed. Next, along with First represent the sequential relationships among tasks and subtasks.
5) US: Task Uses Object. The completion of a certain task may depend on the Status of relevant objects.
6) USB: Object is UsedBy Task.
7) PF: Agent Performs Task. The Status of Agent may affect the execution of a certain task.
8) PFB: Task is PerformedBy Agent.
9) HAS: there are four types of HAS relationship, including hasStatus, hasChoice, hasUserAction and hasSystemOperation. They are constructed as data properties in the proposed model. The data property allows the task ontology to have values representing the status of an agent or an object, the options users have, the actions users should take, or the operations in system side in a task.
    a) hasStatus: represents the Status of Agent or Object. The value can be Available or Unavailable.
    b) hasChoice: a task may have choice. It represents options users may have in a task and what users choose directs the following actions the system would take. The value can be a list of options presenting to users.

c) hasUserAction: represents actions users should take in a task. It only applied to BasicTask. The value is a string of explanation to users. Depending on the size of information, the value could be a file.

d) hasSystemOperation: represents operations system have in a task. It is only meaningful in a BasicTask. The value is a string of explanation to users. Depending on the size of information, the value could be a file.

4.2.3 RULES

The following SWRL rules are defined for the proposed task ontology model:

1. DecomposeInto(x, x) is invalid

   The relationship DecomposeInto is irreflexive.

2. DecomposeInto(x, y) → DecomposeInto(y, x) is invalid

   The relationship DecomposeInto is asymmetric.

3. First(x, x) is invalid

   The relationship First is irreflexive.

4. First(x, y) ∧ First (y, x) is invalid

   The relationship First is asymmetric.

5. Next(x, y) ∧ Next (y, x) is invalid

   The relationship Next is asymmetric.

6. Next (x, x) is invalid

   The relationship Next is irreflexive.

7. CompositeTask(x) ∧ UnitTask(x) is invalid

   The subclasses of task: CompositeTask and UnitTask are disjoint with each other.

8. CompositeTask(x) ∧ BasicTask(x) is invalid

   The subclasses of task: CompositeTask and basicTask are disjoint with each other.

9. UnitTask(x) ∧ BasicTask(x) is invalid

   The subclasses of task: UnitTask and basicTask are disjoint with each other.

10. Task(x) ∧ Agent(x) is invalid

    The classes Task and Agent are disjoint with each other.

11. Task(x) ∧ Object(x) is invalid

    The classes Task and Object are disjoint with each other.

12. Agent(x) ∧ Object(x) is invalid

    The classes Agent and Object are disjoint with each other.

13. Uses (x, y) → usedBy(y, x)

    Uses and usedBy are inverse object properties.

14. Performs(x, y) →performedBy(y, x)

    Performs and performedBy are inverse object properties.

15. Uses(x, y) →Uses(y, x) is invalid

    The relationship Uses is asymmetric.

16. Uses(x, x) is invalid

    The relationship Uses is irreflexive.

17. Performs(x, y) →Performs(y, x) is invalid

    The relationship Performs is asymmetric.

18. Performs(x, x) is invalid

    The relationship Performs is irreflexive.

Once a task ontology is constructed, these rules can be used by an ontology reasoner to infer information that is not explicitly described in the ontology. The reasoner can help conduct consistency check as well as retrieve implicit knowledge. If an invalid relationship is built in an ontology, the reasoning process will help identify and fix the problem.

Besides rules, other restrictions, including existential, universal and cardinality restrictions are also defined in the proposed task ontology model. They can be used by a reasoner for ontology inference and validation as well. The following restrictions are defined in the task ontology:

1) UnitTask is equivalent to:

   Task and ((DecomposeInto some BasicTask) and (DecomposeInto min 1 BasicTask) and (First min 1 BasicTask))

   UnitTask is Task that can be decomposed into at least one BasicTask and it has at least one BasicTask as its First object property.

2) CompositeTask is equivalent to:

   Task and ((DecompostInto some UnitTask) and (DecomposeInto min 1 UnitTask) and (First min 1 UnitTask))

   CompositeTask is Task that can be decomposed into at least one UnitTask; and it has at least one UnitTask as its First object property.

3) The domain for Uses is Task, and the Range is Object

   Task uses only Object.

4) The domain for usedBy is Object, and the Range is Task

   Object is used only by Task.

5) The domain for Performs is Agent, and the Range is Task

Agent performs only Task.

6) The domain for performedBy is Task, and the Range is Agent

   Task is performed only by Agent.

7) hasChoice min 2 Literal

   The data property hasChoice for a task has at least 2 cardinalities as its value.

8) The domain for the data properties of has UserAction and hasSystemOperation is BasicTask

   Only BasicTask has the data properties of hasUserAction and hasSystemOperation.

9) The domain for the data property hasChoice is Task

   Only Task has the data property hasChoice.

10) The domain for the data property hasStatus is Agent or Object

    Only Agent or Object has the data property hasStatus.

4.2.4 ONTOLOGY INSTANTIATION

In this thesis, ontology instantiation is the task modeling process for specific tasks using proposed task ontology model.

A specific task is commonly represented as a tree-like task hierarchy. First a task and its subtasks are identified and mapped as instances or members into the subclasses of CompositeTask, UnitTask and BasicTask according to their decomposition level in the task tree. Composite tasks are the most complex tasks which sit on the top level of the task tree. It is decomposed into a set of unit tasks. For complicated tasks, it is possible to have more than one level of unit tasks. The relationships between different levels of unit tasks are the same with ones between composite and unit task. The unit tasks are further decomposed into basic tasks which

are the leaves in the task tree. Then the relationships among these tasks and subtasks are determined and mapped into appropriate properties in the task ontology. Two main types of properties are adopted in the proposed task ontology model: one is the object property which links an individual to an individual; and another is the data property that links an individual to a data value. In this step the relationships between instances and proper data value should be identified and added into the ontology. DecomposeInto, Next and First are commonly used for the execution order of a task. The relationship hasChoice is used to represent different options for users to choose and the users' selection would determine the next action the system will take. Thus different types of task flow, inlucing sequential tasks or alternative tasks can be represented in the task ontology model.

For some types of tasks, the context should be considered as well. The class Object is used to model the tools for a task or its subtasks. The availability of the tools is represented as status in the task ontology. Agent is used for human beings, machines or software applications which perform a task or its subtasks. Their status is also represented as hasStatus.

Once an ontology for a specific task is constructed, we can run an ontology reasoner to classify ontology to get inferred classes. Then the ontology will have inferred hierarchy as well as inserted hierarchy. The reasoning process will check consistency as well and indicate any inconsistency among individuals or their properties. Thus any problems that occur during the process can be identified. Once all errors are fixed, a valid ontology for the particular task is established. This task ontology will be used by the domain independent dialogue manager described in the following section to direct the dialogue process between the system and users.

4.3 THE DOMAIN INDEPENDENT DIALOGUE MANAGER

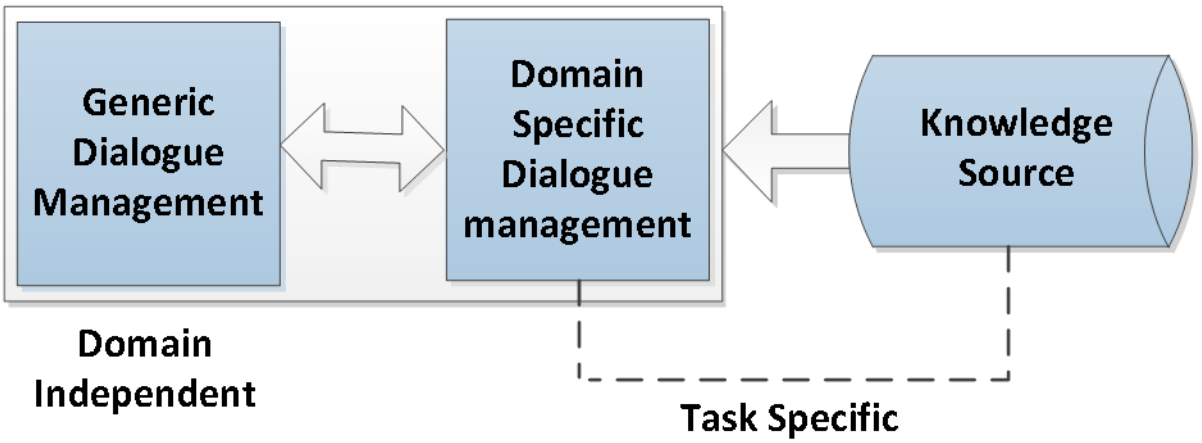

Figure 4.3.1 the structure of the DM based on previous domain independent dialogue management approaches

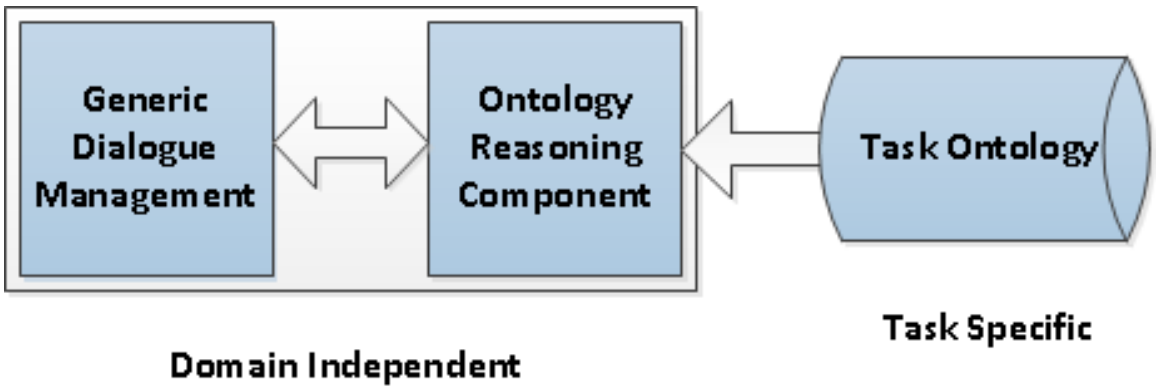

Figure 4.3.2 the structure of the modified DM based on the proposed method

According to Section 3.1, previous approaches to the domain independent dialogue management separated the dialogue manager into a generic dialogue management component and a domain specific dialogue management component. See Fig. 4.3.1. Based on the proposed task ontology model in Section 4.2, the structure of the DM is modified to make the DM domain independent. An ontology reasoning component is added to the DM and replaces the domain specific dialogue

management component in previous approaches. It is able to retrieve domain specific knowledge from the task ontology. Fig. 4.3.2 illustrates the structure of the modified DM. It includes the Generic Dialogue Management and an ontology reasoning component. The ontology reasoning component is invoked by the Generic DM. In Fig. 4.3.2, only the task ontology is specific to a certain task; and all other components are independent of specific tasks or domains.

Fig. 4.3.3 illustrates the dialogue management process of a dialogue system based on the modified dialogue manager. In the beginning of the dialogue, the user can select from a list of tasks available in a dialogue system. Once the user selects a particular task, the dialogue system will find the root of the task, the composite task. Then the system will start to go through the task tree till the entire task is completed.

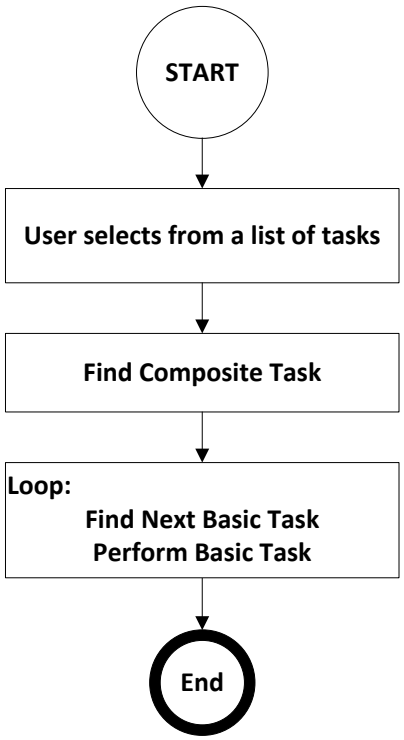

Figure 4.3.3 the dialogue management process in the modified DM

Fig. 4.3.4 is the pseudo code for the loop of finding next basic task and performing the basic task listed on Fig. 4.3.3.

```
findBasicTask(currentTask){
        if (currentTask is not a basic task) {
                if (currentTask has > 1 First properties) {
                        find the hasChoice property;
                        ask user to select one;
                        find the user selected task;
                        set current task to the user selected task;
                } else {
                        find First task;
                        set current task to the First task;
                }
        }
        if (current task is a basic task) {
                perform the task;
                if (current task has Next property) {
                        if (current task has >1 Next properties) {
                                find the hasChoice property;
                                ask user to select one;
                                find the user selected task;
                                set current task to the user selected task;
                        } else {
                                find the next task;
                                set current task to the next task;
                        }
                        findBasicTask(currentTask);
                } else {
                        entire task completed;
                }
        } else {
                findBasicTask(currentTask);
        }
}
```

Figure 4.3.4 pseudo code of the dialogue management process

The task tree is constructed in a way that a subtask will be visited at most once. If the subtask is a basic task, then its object property Next will be visited once and each of its two data properties – hasUseraction and hasSystemOperation will be expanded once. Visiting an object property means to find the subtask which is listed as the value of the property. Expanding a data property means to open the value and deliver to the user in an output modal. If the subtask is a

unit task or composite task, then the object property First will be visited once. Then we can evaluate the dialogue management process as follows.

Let m_1: the number of Unit Task

m_2: the number of Basic Task

c_o: expense visiting an object property

c_d: expense expanding a data property

Then *Total amount of work done:*

Before loop: a constant amount a

After loop: a constant amount b

Each time go through a loop:

for basic task: $c_o + 2c_d$

for unit task: c_o

In worst case, all subtasks will be visited once in the loop, so:

Total amount of time: $(a + b + c_o * m_1 + (c_o + 2c_d) * m_2)$

Time complexity: O $(m_1 + m_2)$

Let $m = \max(m_1, m_2)$

Then *Time complexity:* O (m)

So the time complexity is O (m) in which m is the maximum of the total number of the basic tasks or the unit tasks within a task.

CHAPTER 5 IMPLEMENTATION

The system implementation includes the creation of the abstract task ontology and the development of the dialogue system based on the proposed method. A task ontology can be created through the ontology instantiation of this abstract ontology when the dialogue system is applied to a particular task.

5.1 ABSTRACT TASK ONTOLOGY

Based on the proposed task ontology model described in Section 4.2, an abstract task ontology is created using Protégé 4.1. In the implementation, only 3 levels of task decomposition are considered. For more complicated tasks, the abstract task ontology can be easily expanded to contain multiple levels of unit tasks, such as UnitTask1, UnitTask2, etc. The relationships between different levels of unit tasks and between other classes are the same with those between composite tasks and unit tasks. All rules and restrictions in the ontology stay unchanged as well.

Fig. 5.1.1 presents all concepts or classes developed in the task ontology, including Agent, Choice, Object, Status, SystemOperation, Task and UserAction. The subclasses of Task, such as CompositeTask, UnitTask and BasicTask are also included in the model.

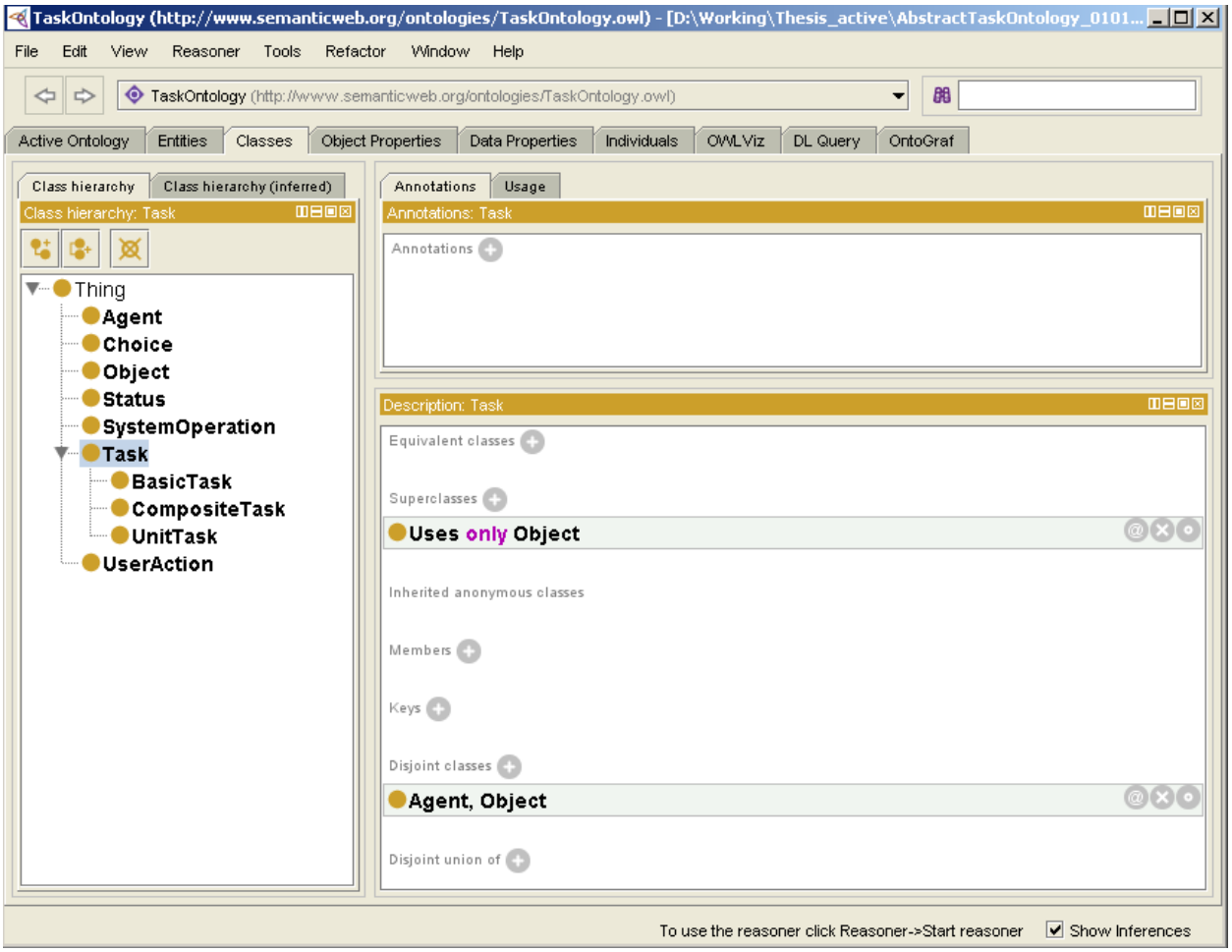

Figure 5.1.1 Classes developed in the task ontology model

All object properties which represent the various relationships among concepts created in the task ontology model are shown in Fig. 5.1.2. They include First, Next, DecomposeInto, Performs, PerformedBy, Uses and UsedBy. The characteristics for each property are also built in the model, such as asymmetric, irreflexive, etc.

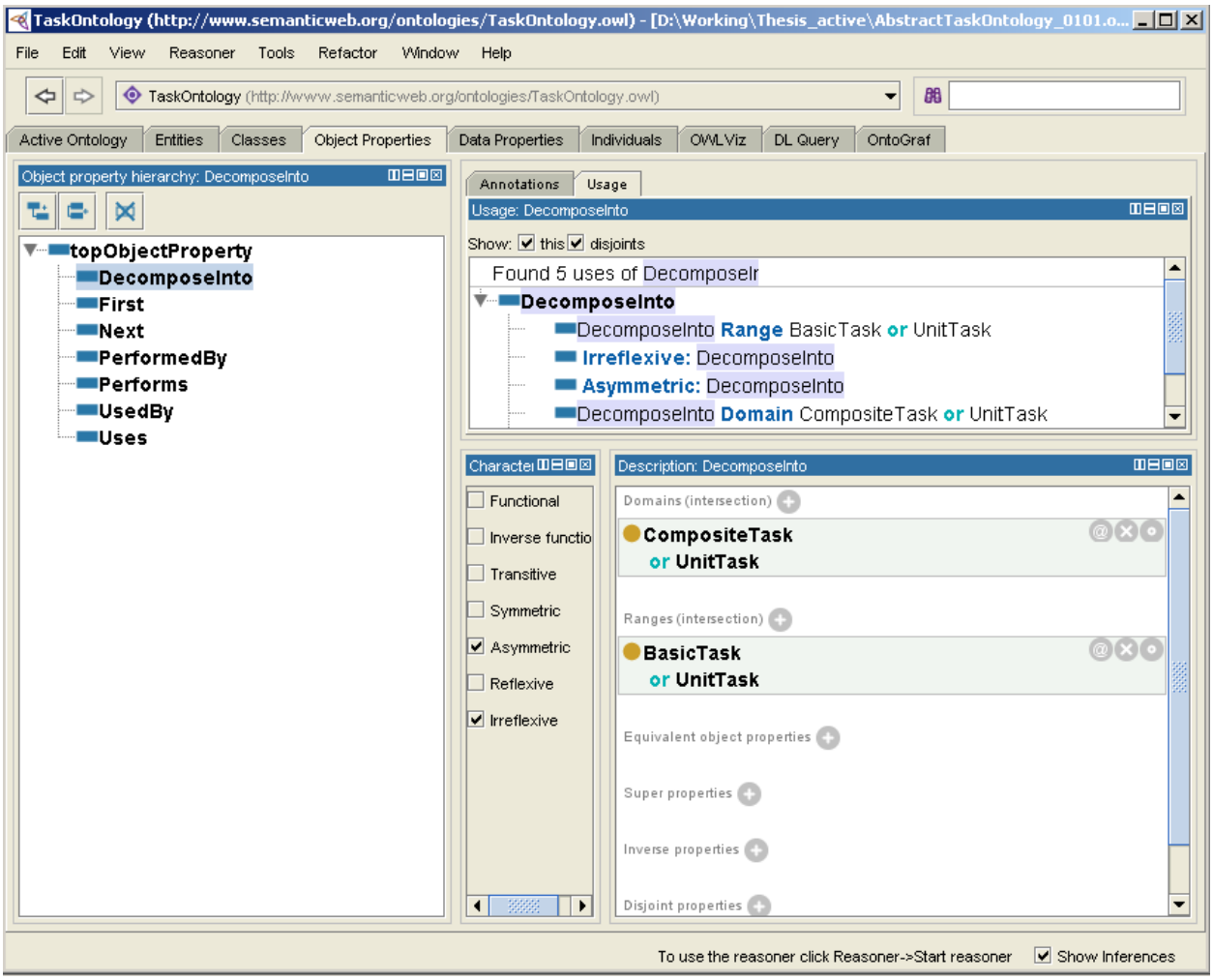

Figure 5.1.2 Object properties developed in the task ontology model

The data type properties are adopted in the task ontology model as well, including hasChoice, hasStatus, hasSystemOperation and UserAction. See Fig. 5.1.3. Constraints on this type of properties are set up in the model. For example, the Domain of hasUserAction is BasicTask.

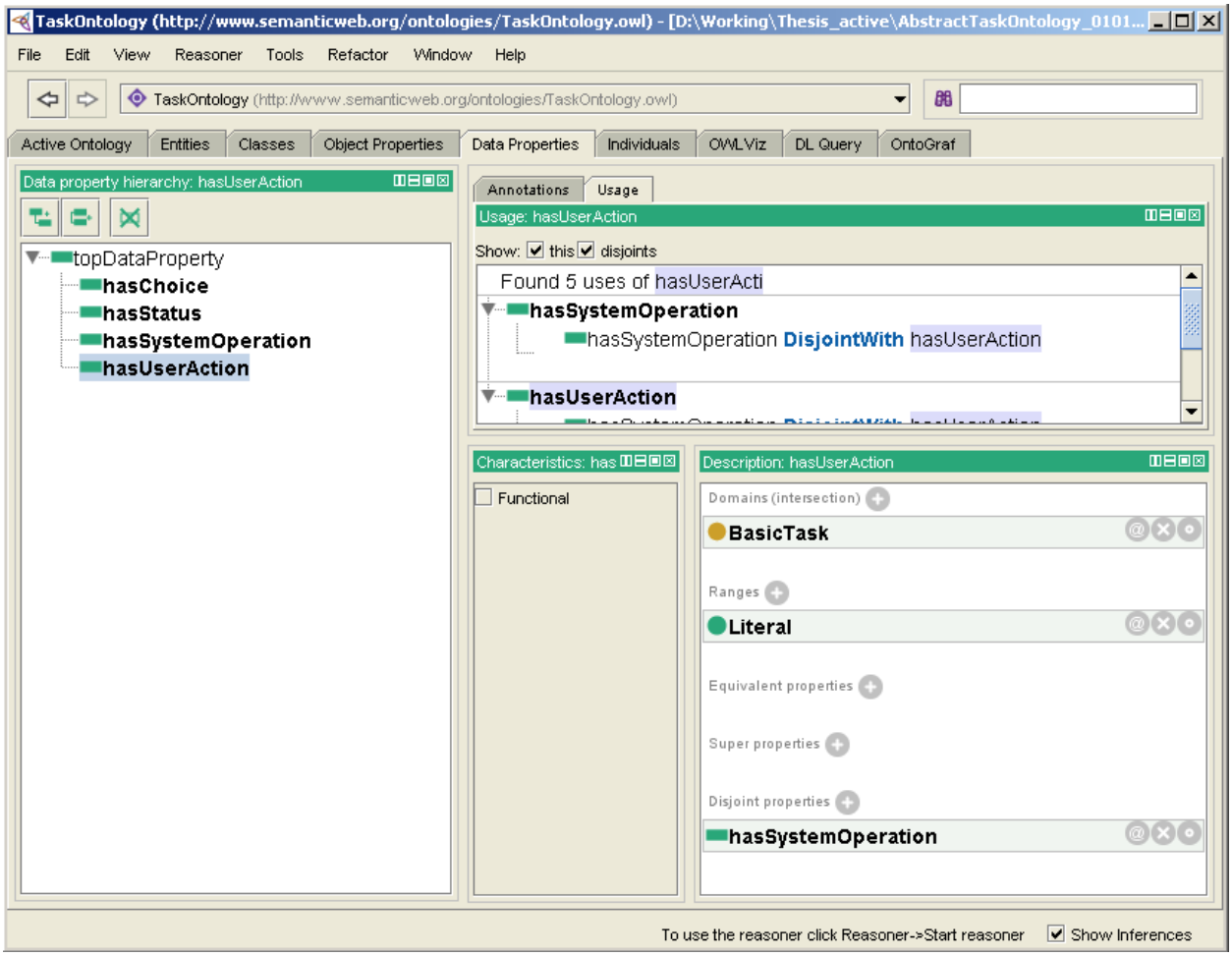

Figure 5.1.3 Data properties developed in the task ontology model

This abstract task ontology can be instantiated to task ontology for specific tasks. In this paper, ontologies for two tasks from different domains are constructed based on the abstract task ontology. One task is the book borrowing service; and another is the online train ticket booking. The Section 6.1 and 6.2 describe the two cases in details respectively.

5.2 The Domain Independent Dialogue System

A domain independent dialogue system is developed based on proposed method described in Chapter 4. The system consists of the proposed dialogue manager, an Input/Output controller and a dialogue interface. The dialogue manager connects with the dialogue interface via the Input/Output controller. Fig. 5.21 illustrates the architecture of the dialogue system implemented in this thesis.

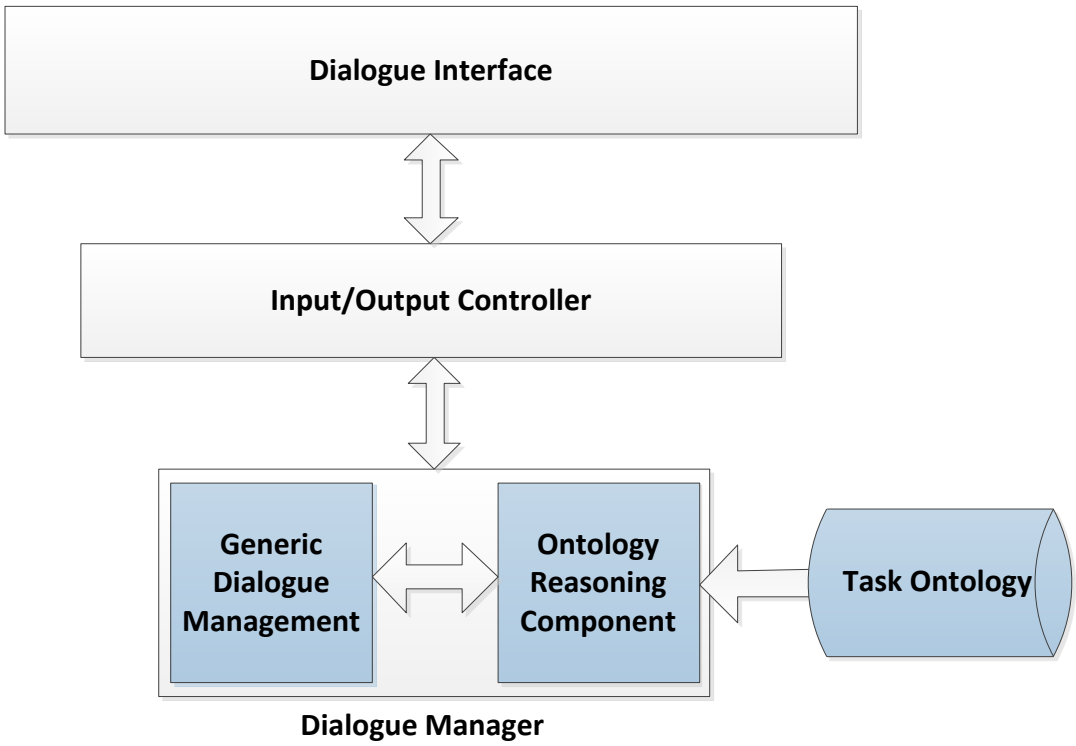

Figure 5.2.1 Architecture of the dialogue system

The dialogue manager adopts the frame-based dialogue management approach. It is used to control the interactions with human users to instruct users to achieve their goal. The dialogue process is modeled as a set of slot-filling tasks. The dialogue manager gathers predetermined information from user responses to fill in the slots. The predetermined information is stored in

the task ontology. The utterances generated by the dialogue manager, the slots and their value options are all retrieved from the task ontology. The dialogue manager controls the dialogue process based on information gathered from users' responses and the ontology reasoning to guide users to complete a specific task.

The dialogue interface is text-based which displays utterances generated by the dialogue system and provides one slot for users to fill in. The utterances are information about what users should do and what the tool he works with is going to do. The tool can be a machine or an application that helps users to accomplish a specific task, such as purchasing a train ticket or borrowing a book. In the train ticket purchasing task, the goal is to buy a ticket and the online train ticket booking program is the tool for users to achieve this goal. Typical system utterances will include the following two parts: one is the action that users are expected to take to get a particular subtask done; and another is the operation that the tool – in this case, the online train ticket booking program will have. For example, the dialogue system may display information like "You should enter the travel information including departure station, arrival station, departure date and returning date into the booking form" to instruct what users are expected to do in a certain step. In the meantime the information on the behaviour of booking program is also displayed to inform users what is going on in the booking program side. In this case, it would be "The system will search train schedules to find trains meeting your travel plan. After that, all available trains will be displayed for you to select". Once user has done a particular subtask under the dialogue system's instruction, he may fill in "Next" to move forward to get instructions on next step towards his goal.

If there are options for users, all possible choices will be displayed on the interface for users to choose from. Users are expected to select one of them to fill in the slot. According to

users' selection, the dialogue system will take the appropriate route to guide users to get the task done. In the train ticket booking example, the dialogue system will display the options of "Round Trip" and "One Way" for users to select the type of ticket he wants to buy. So the interface will display the utterance "What kind of ticket do you want to buy? Please select one from the following option: "Round Trip" or "One Way"". And users are expected to input either "Round Trip" or "One Way" to fill in the slot. Then the users' response is passed to the Input/ Output controller which sends the information to the DM. The DM will determine the next step based on user response as well as information retrieved from the task ontology.

The information of all utterances, slots and options for users comes from the task ontology for a specific task. The reasoning component of the dialogue manager will retrieve the appropriate information from the task ontology and control the dialogue process with users till the task is completed. The information generated by the dialogue manager will be sent to the Input/Output controller which allows the text message displayed on the dialogue interface. Different tasks will have different task ontology. Thus the dialogue system is capable of guiding users to complete various tasks from different domains through interactions with human users.

The dialogue system is developed in Java 6.0 using eclipse IDE 3.7. The OWL API for OWL 2 and Pellet - the OWL 2 reasoner for Java are used in the implementation as well. Fig. 5.2.2 is the flowchart of the dialogue system. Currently the task decomposition in the proposed task ontology model contains three levels. For more complicated tasks, extra levels can be introduced to the task decomposition. In that case, the model can be extended to more than three subclasses to the task class easily. Also, the dialogue system readily supports as many levels as needed.

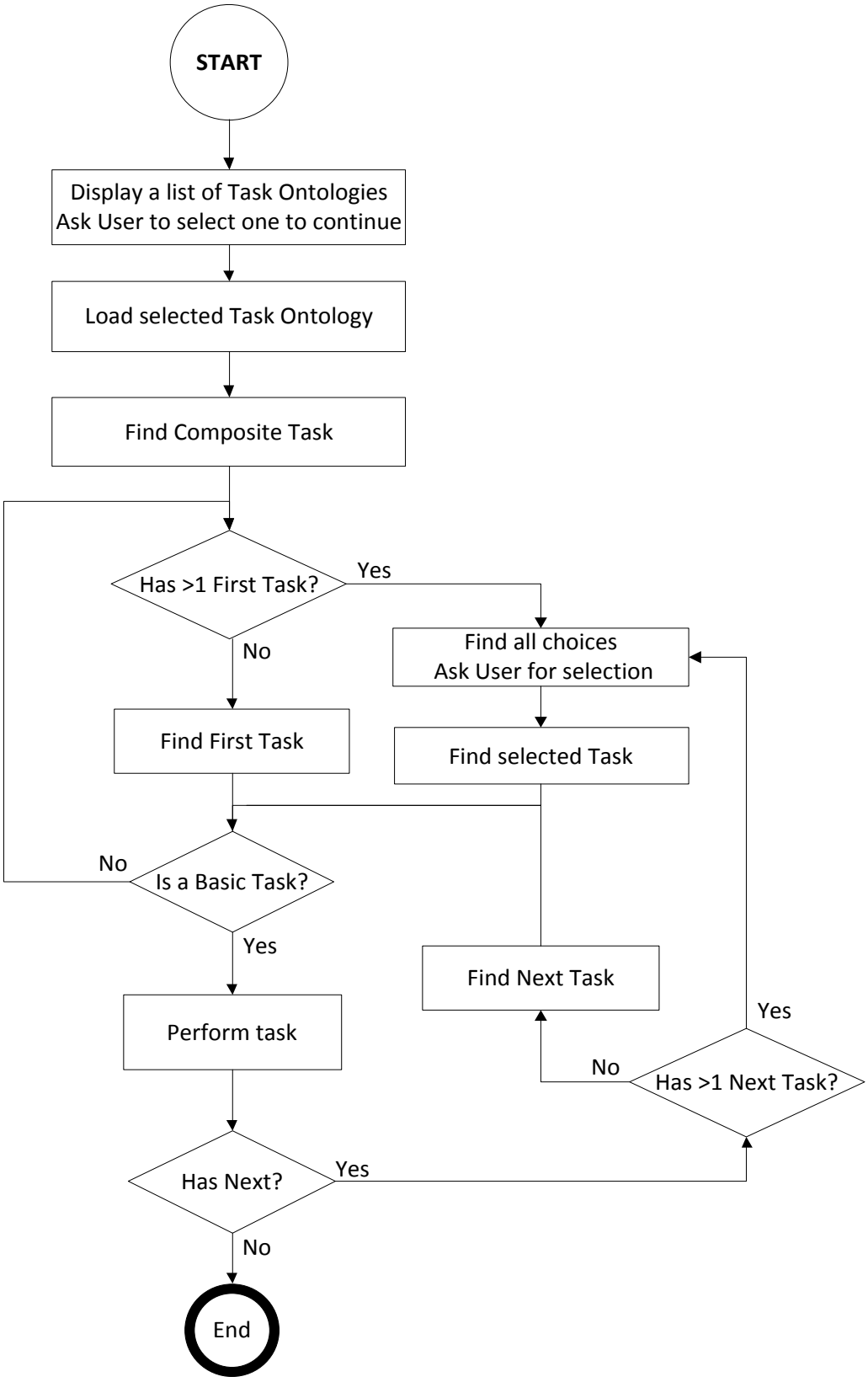

Figure 5.2.2 Flowchart of the dialogue system

The Java GUI tool, WindowBuilder is adopted to create the dialogue interface with users. Fig 5.2.3 shows the interface of the dialogue system.

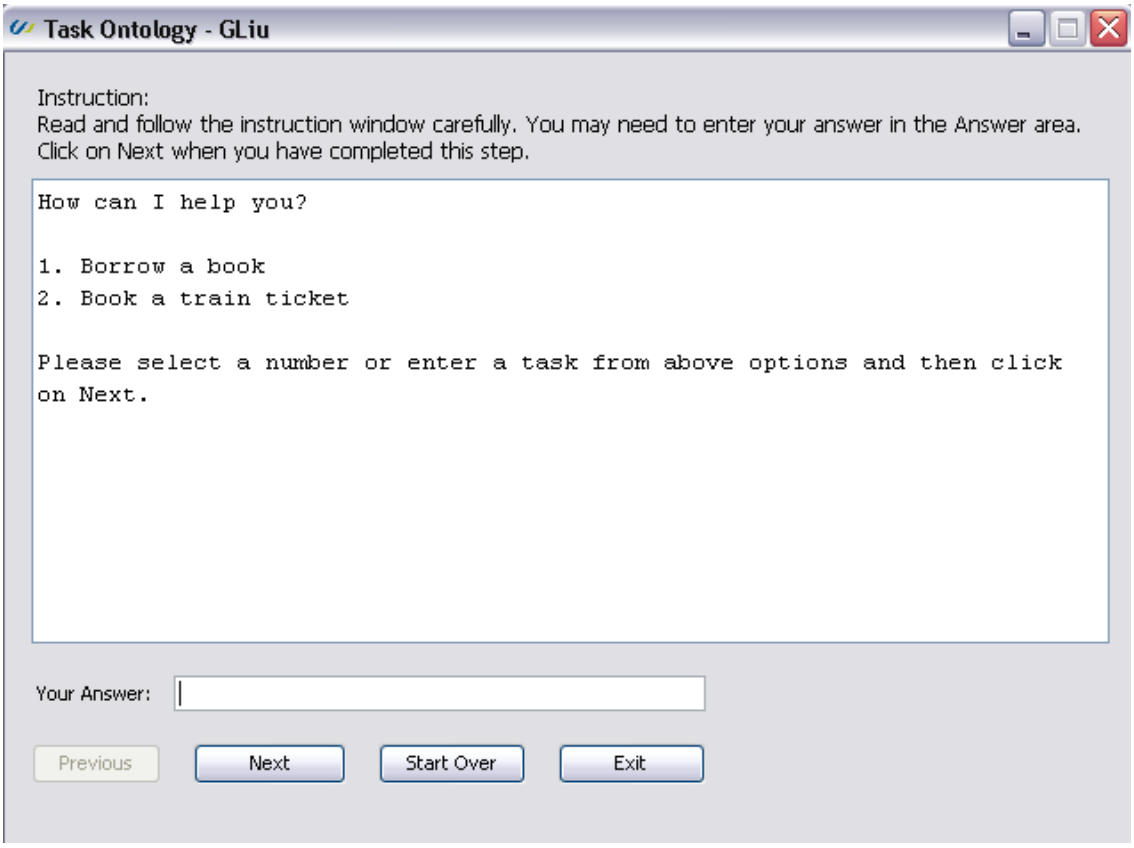

Figure 5.2.3 Dialogue interface of the dialogue system

The institution logo and project name can be added and displayed on the top of the window.

The middle of the interface is a window used to display system utterances to users. The message is mainly retrieved from the task ontology for a certain task. If the size of the message is too large to display in one window, the scroll bar will show up on the right side allowing users to scroll down and up.

Under this window, there is a smaller box for users to fill in their response. Whenever the dialogue system displays options to users, it will be set editable. And the focus of the interface is

also set on that box. Here two methods are provided: by number or by the name of the task. The input text is set as case insensitive which is convenient to users. If users enter an invalid answer, an error message will be displayed besides the box to remind users to re-enter a valid one.

In the bottom of the interface, there are four buttons: Previous, Next, Start Over and Exit for users to choose what they want.

a) The Previous button allows users to go back to one or more steps as they like.

b) The Next is the button for users to continue a task.

c) Start Over: allowing users to start from the beginning at any time.

d) Exit: users can exit any time.

When necessary, some buttons are disabled. For example, in Fig. 5.2.3, the "Previous" button is disabled in this dialogue window because there is no previous step for current interaction. These features facilitate users to interact with the dialogue system effectively.

CHAPTER 6 CASE STUDY

The dialogue system implemented in Chapter 5 is experimented with two tasks from different domains, one is the book borrowing service; and another is the online train ticket booking. Two task ontologies are constructed for them respectively through ontology instantiation of the abstract task ontology created in Section 5.1. The task ontologies provide knowledge about each task to the dialogue system, so the system can be used for the two tasks without any modification. In other words, the dialogue system is task or domain independent. The following sections, Section 6.1 and 6.2 describe the two cases in details respectively.

6.1 BOOK BORROWING SERVICE

6.1.1 TASK ONTOLOGY FOR BOOK BORROWING SERVICE

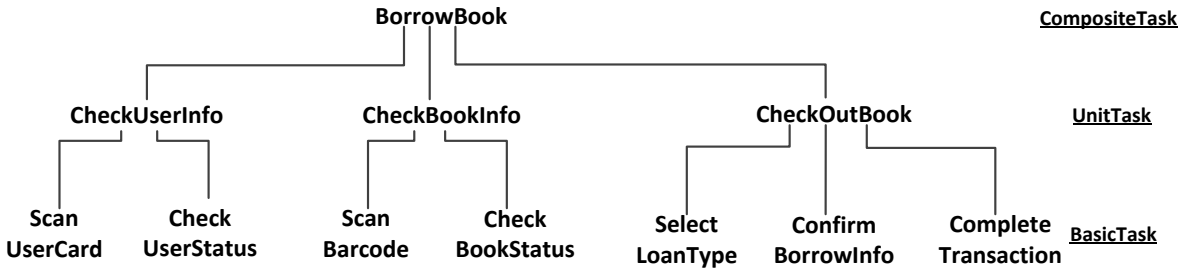

Figure 6.1.1.1 Task decomposition tree for book borrowing service

The task of book borrowing service can be decomposed into a set of subtasks of different hierarchical level. Fig. 6.1.1.1 illustrates the task decomposition hierarchy of the book borrowing service in a library. At the top level, BorrowBook is a CompositeTask. In the task ontology for

the book borrowing service, BorrowBook is an instance (or a member in Protégé) of the subclass CompositeTask. It can be decomposed into three UnitTasks, including CheckUserInfo, CheckBookInfo and CheckOutBook which are instances of the subclass UnitTask. Among these UnitTasks, CheckUserInfo is further decomposed into two BasicTasks: ScanUserCard and CheckUserStatus; CheckBookInfo is decomposed into ScanBarcode and CheckBookStatus; and CheckOutBook is decomposed into SelectLoanType, ConfirmBorrowInfo and CompleteTransaction. All the seven BasicTasks are in the bottom of the task hierarchy and built as the instances of the BasicTask in the task ontology.

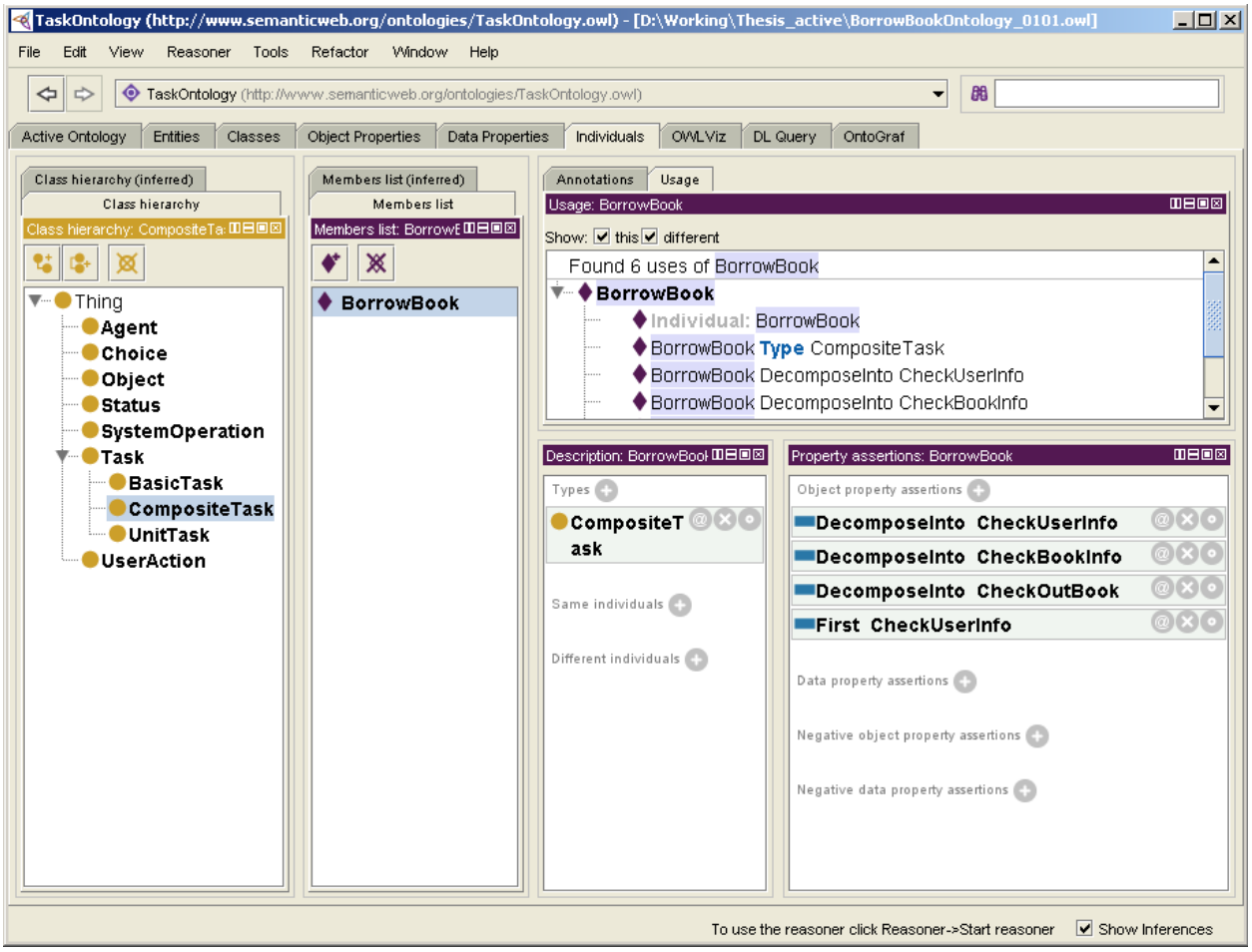

Figure 6.1.1.2 Instance of CompositeTask and its properties

Fig. 6.1.1.2 presents the instance of the CompositeTask, BorrowBook and its relationships with other instances. The relationships are represented as object properties in the ontology for this specific task, including:

DecomposeInto CheckUserInfo

DecomposeInto CheckBookInfo

DecomposeInto CheckOutBook

First CheckUserInfo

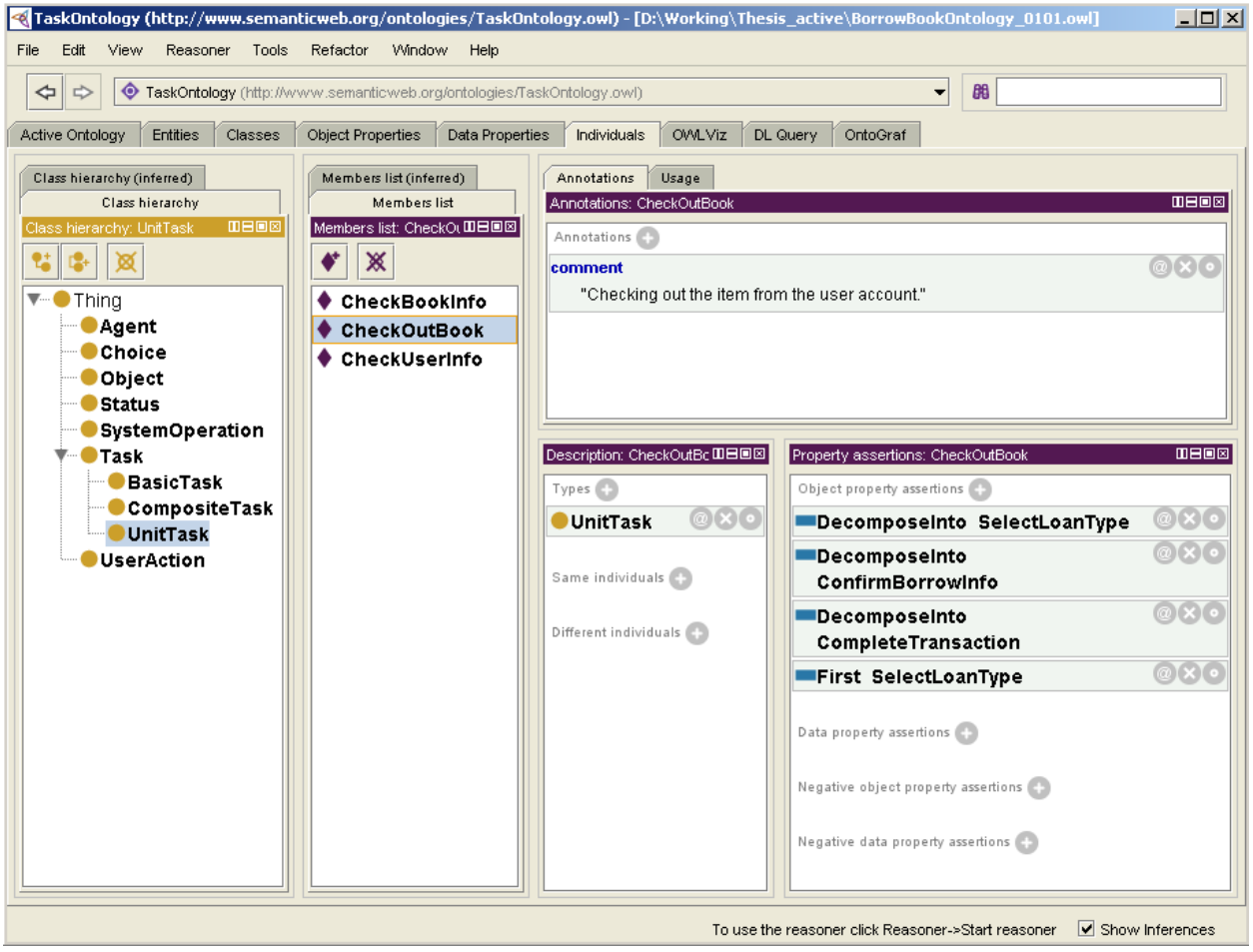

Figure 6.1.1.3 Instances of UnitTask and their properties

Fig. 6.1.1.3 displays the instances of the UnitTask, including CheckUserInfo, CheckBookInfo and CheckOutBook. They have their own object properties. For example, the CheckOutBook has the following object properties:

DecomposeIntoSelectLoanType

DecomposeIntoConfirmBorrowInfo

DecomposeIntoCompleteTransaction

First SelectLoanType

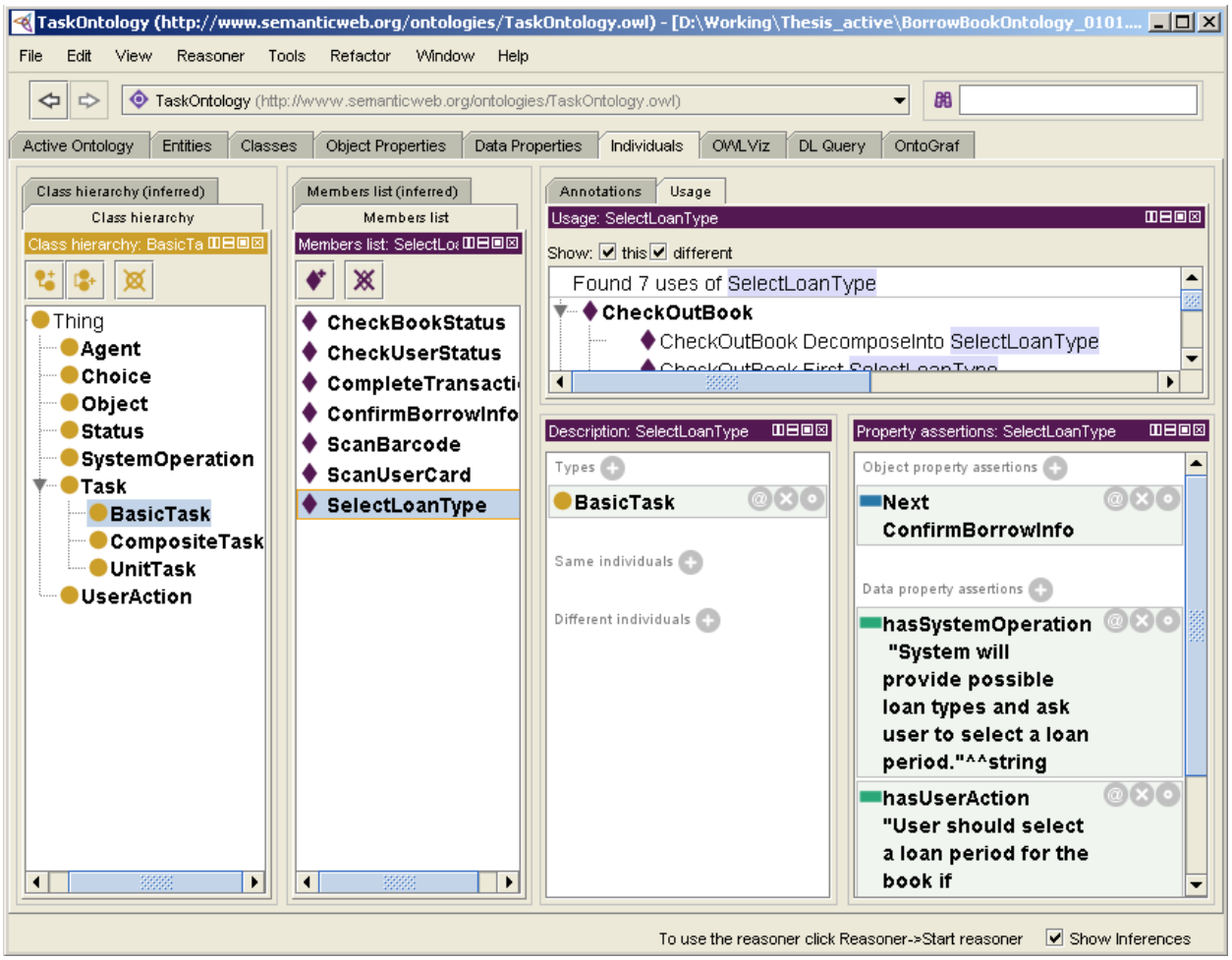

Figure 6.1.1.4 BasicTask and their properties

Instances of BasicTask and their properties are displayed in Fig. 6.1.1.4. Take SelectLoanType as example. It contains both object and data properties. The object property is:

Next ConfirmBorrowInfo

The data properties include:

hasUserAction “User should a loan period for the book if applicable”

hasSystemOperation “System will provide possible loan types and ask users to select a loan period”

Those rules and constraints built in the abstract ontology model apply to this ontology automatically. The instantiated ontology for the book borrowing service is used in the dialogue system created in Chapter 5 to instruct users to complete the book borrowing task through interactions between the dialogue system and users.

6.1.2 Dialogue Process for the Book Borrowing Service

Once the book borrowing service is selected by users, the dialogue system starts retrieving information from the task ontology for this task and gathering information from users through interaction with them. Fig. 6.1.2.1 shows the dialogue interface for the subtask "Scan User Card". It is the first basic task that instructs users' action and explains the system operation.

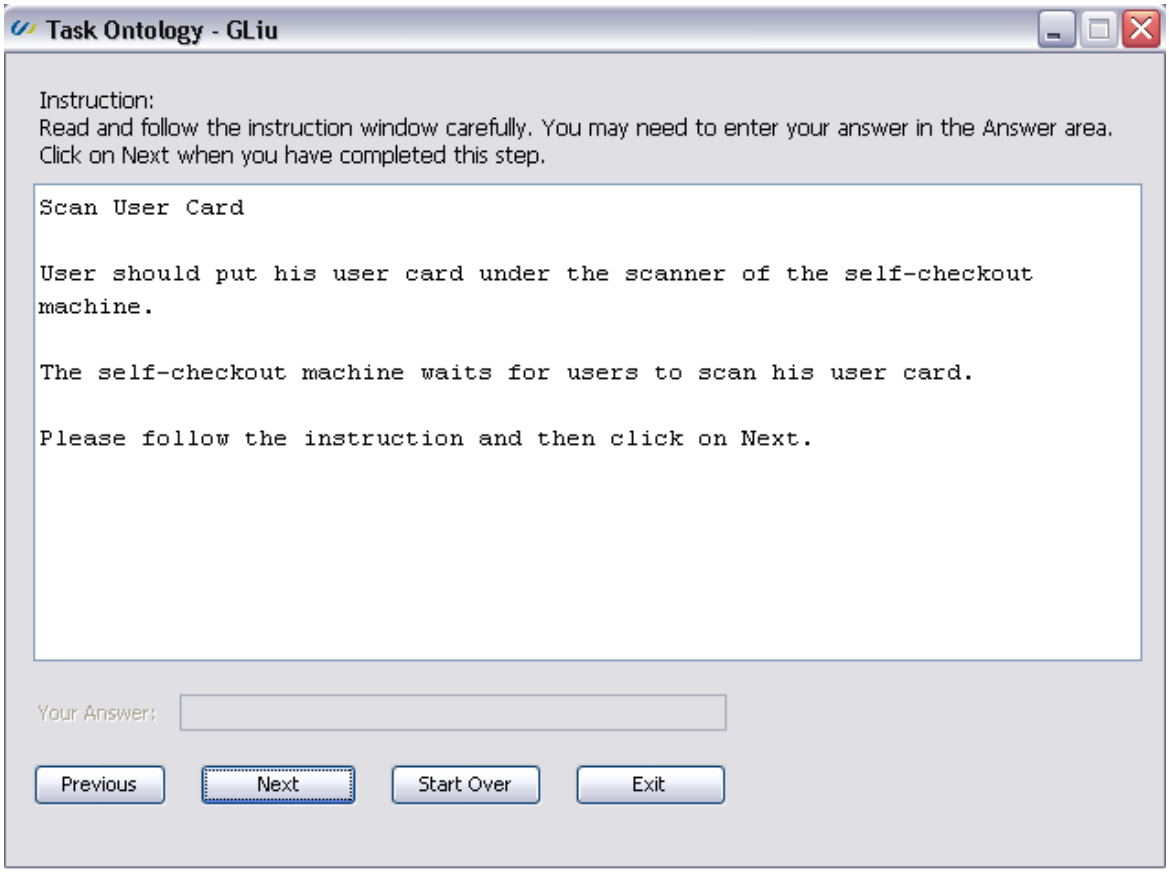

Figure 6.1.2.1 Dialogue interface for the subtask "Scan User Card"

In the large display window, the task name "Scan User Card" comes from the current subtask name set up in the book borrowing task ontology. The instruction for users: "User should put his card under the scanner of the self-checkout machine" and the explanation of system

operations “The self-checkout machine waits for the user to scan his/her card” are the values of the data properties for this subtask stored in the book borrowing task ontology. When necessary, the library which adopts the dialogue system can change the data property value via the ontology editor directly. Also if there is a policy change or other changes affecting the subtasks for this service, the library can modified the task ontology themselves. There is no need for asking dialogue system developers to modify the code.

Because users do not expect to enter an answer in this step, the box for users filling in their response is disabled. Instead, the “Next” is focused which is the button usually selected by a user in this step. Other buttons are enabled. Users can go back to the last step or more previous steps following the buttons. The current task may be interrupted for some reason, so users can click on Start Over or Exit at any time.

Following the instructions by the dialogue system, all basic subtasks for this service will be completed, including Check User Status, Scan Book Barcode, Check Book Status, Select Loan Type, Confirm Borrow Info and Complete Transaction. When the entire task is completed, users have the options to start over or exit from the dialogue system. Fig. 6.1.2.2 is the dialogue interface after a task is finished.

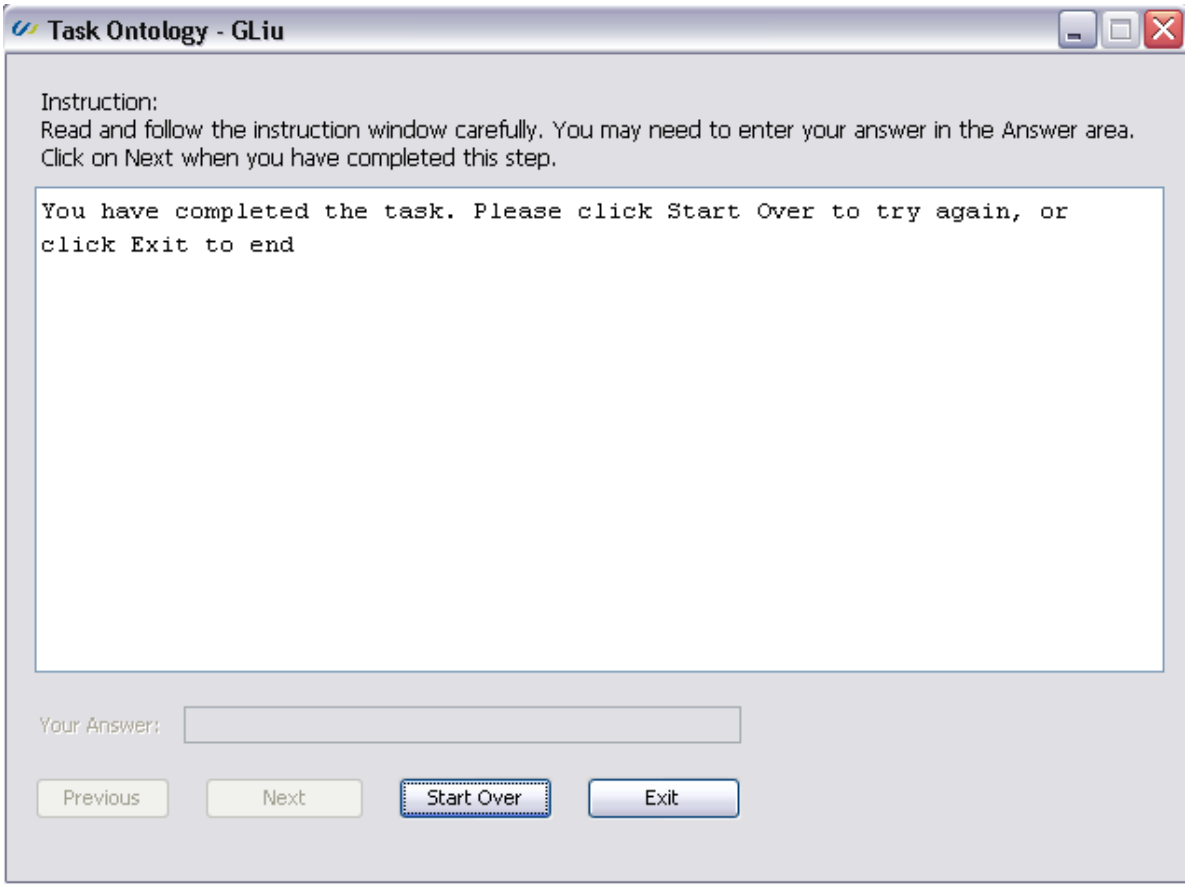

Figure 6.1.2.2 Dialogue interface after a task is completed

6.2 Online Train Ticket Booking Service

6.2.1 Task Ontology for Online Train Ticket Booking Service

Compared with the book borrowing task in Section 6.1, the online train ticket booking task is more complicated. It contains not only sequential subtasks but also alternative subtasks. For alternative subtasks, users can make a choice among all options provided by the system. The selection by the users will determine the next subtask. The proposed task ontology model support this types of subtasks using the class Choice and the data property hasChoice.

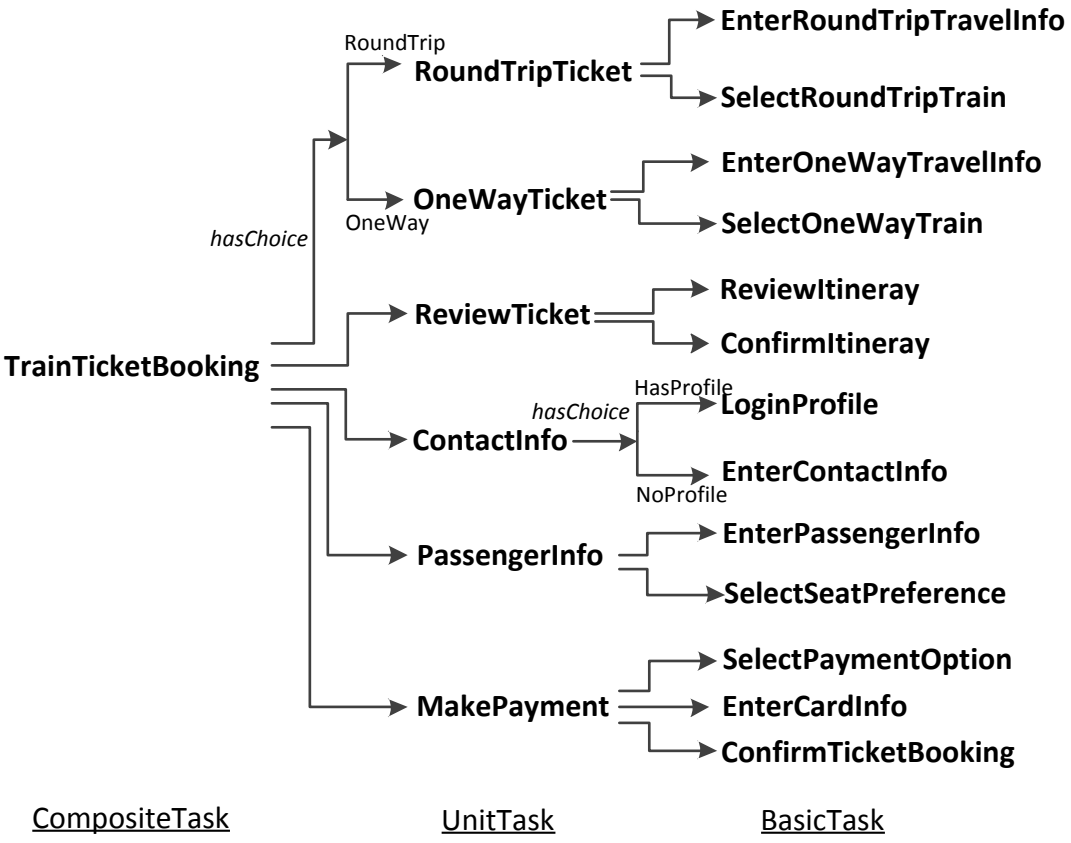

Figure 6.2.1.1 TrainTicketBooking Task Decomposition

Fig. 6.2.1.1 illustrated the task decomposition for the online train ticket booking task as follows:

1) One CompositeTask instance: the TrainTicketBooking

2) Six UnitTask instances:

RoundTripTicket

OneWayTicket

ReviewTicket

ContactInfo

PassengerInfo

MakePayment.

Among them, the RoundTripTicket and OneWayTicket are alternative tasks and which should be executed depends on users' choice

3) Thirteen BasicTask instances:

EnterRoundTripTravelInfo

SelectRoundTripTrain

EnterOneWayTravelInfo

SelectOneWayTrain

ReviewItinerary

ConfirmItinerary

LoginProfile

EnterContactInfo

EnterPassengerInfo

SelectSeatPreference

SelectPaymentOption

EnterCardInfo

ConfirmTicketBooking

These instances compose different UnitTasks. Among them, LoginProfile and EnterContactInfo are alternative tasks which will be executed depending on users' selection.

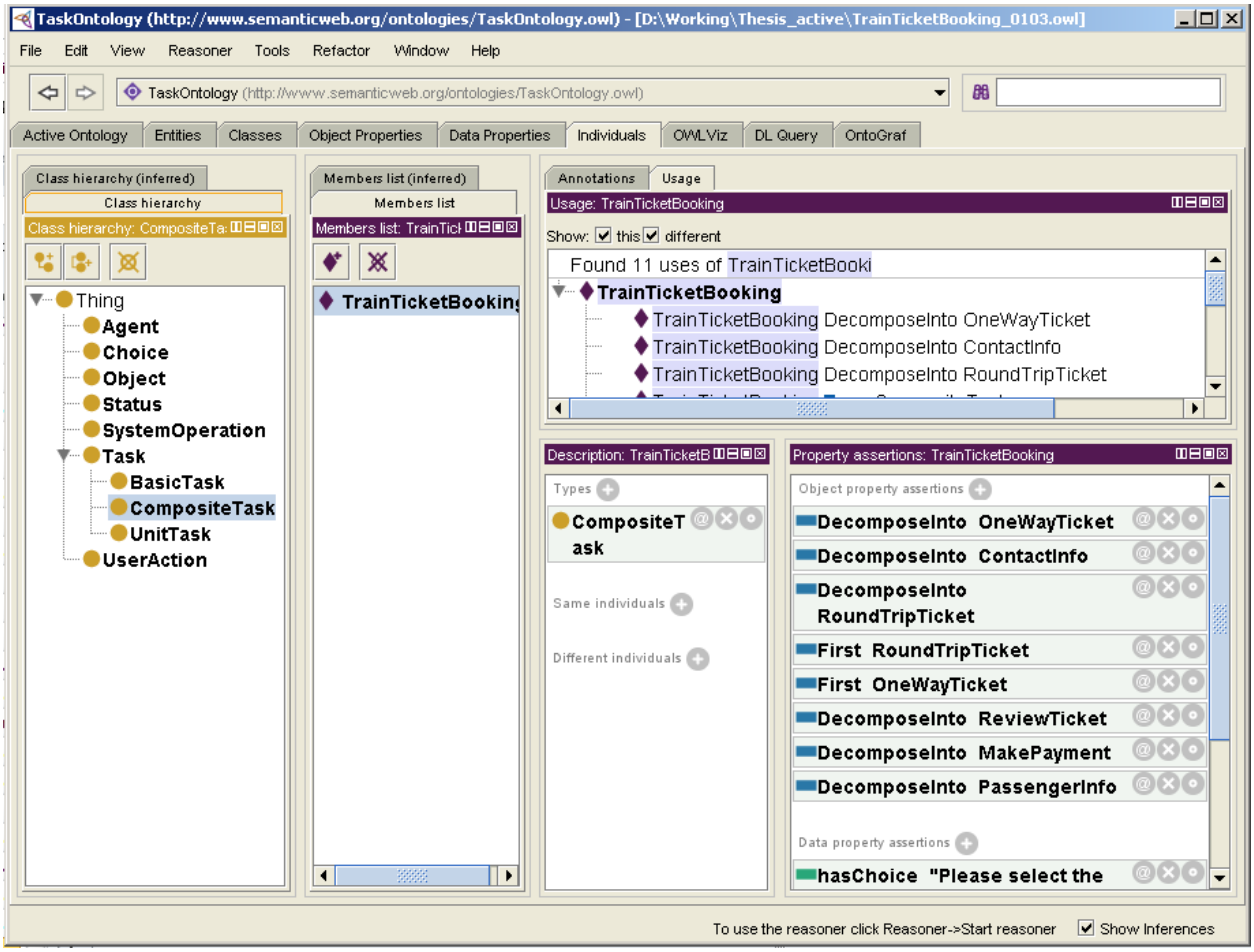

Figure 6.2.1.2 CompositeTask in the TrainTicketBooking ontology

Fig. 6.2.1.2 presents the CompositeTask instance and its object properties and data property as follows:

DecomposeInto RoundTripTicket

DecomposeInto OneWayTicket

DecomposeInto ReviewTicket

DecomposeInto ContactInfo

DecomposeInto PassengerInfo

DecomposeInto MakePayment

First RoundTripTicket

First OneWayTicket

hasChoice “Please select the type of the ticket you want to book:

RoundTripTicket

OneWayTicket”

There are two First properties in the statements. If there are more than one First properties for an instance, the dialogue manager, more specifically the ontology reasoning component will check the hasChoice property to see the value of the options and ask users to select one from the possible options. The value of these options must be exactly the same with the subtasks (instances) they refers to. For other object properties, such as Next, the same rule applies. In the example we are discussing, the data property hasChoice contains two options “RoundTripTicket” and “OneWayTicket” which indicate the two UnitTask instances. Any of them selected by users will be the first subtask in the taskflow for the TrainTicketBooking task.

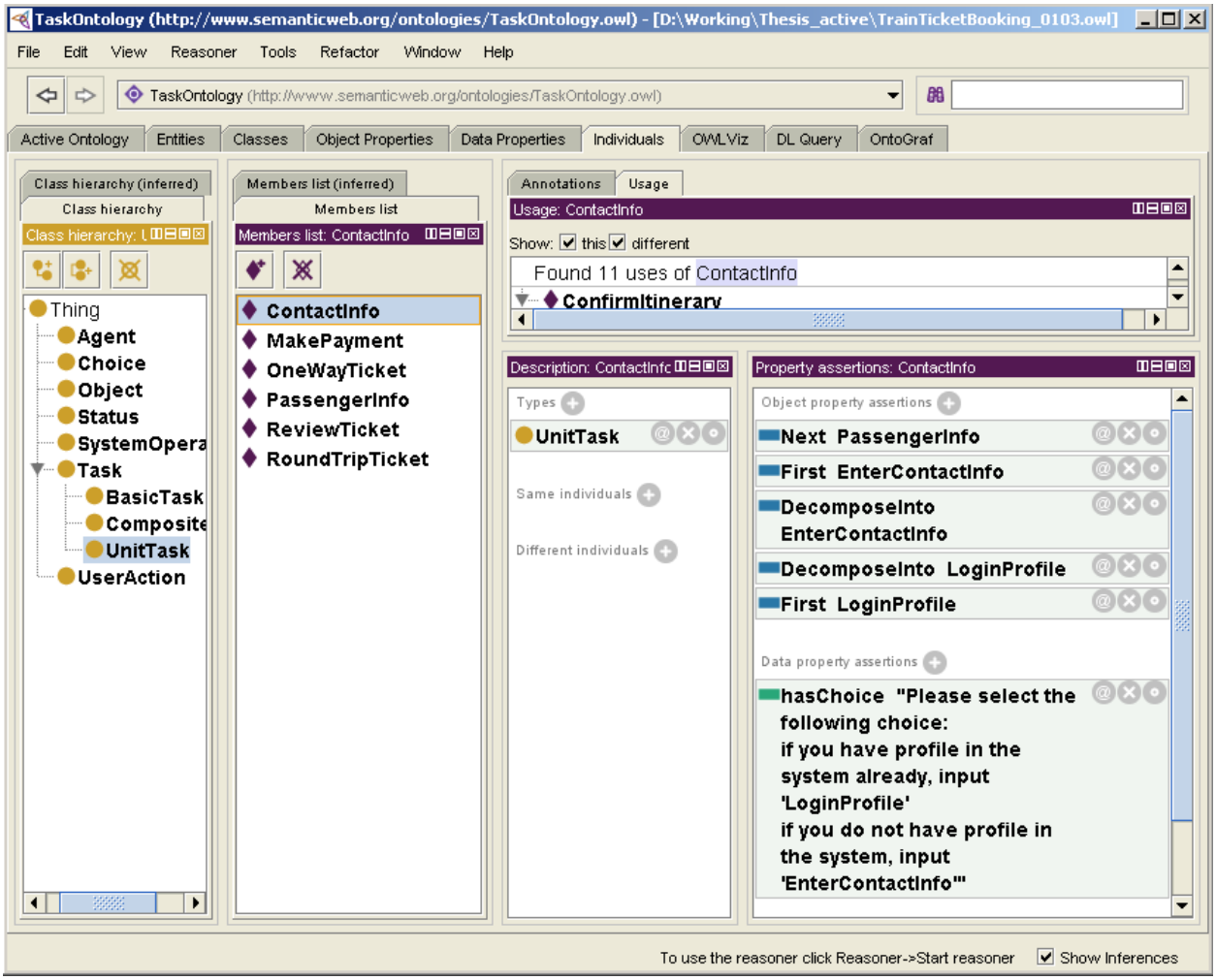

Figure 6.2.1.3 UnitTask instances in the TrainTicketBooking ontology

Fig. 6.2.1.3 shows the UnitTask instances and their properties. Some of them have data properties as well as object properties, such as ContactInfo; whereas, others only have object properties, such as RoundTripTicket, OneWayTicket, ReviewTicket, PassengerInfo and MakePayment. Similar to the hasChoice property in the CompositeTask instance TrainTicketBooking, the UnitTask instance ContactInfo contains the options for users to select. If the user has a profile in the system already, s/he can choose LoginProfile; otherwise, s/he should choose EnterContactInfo to enter his/her contact information to the booking system.

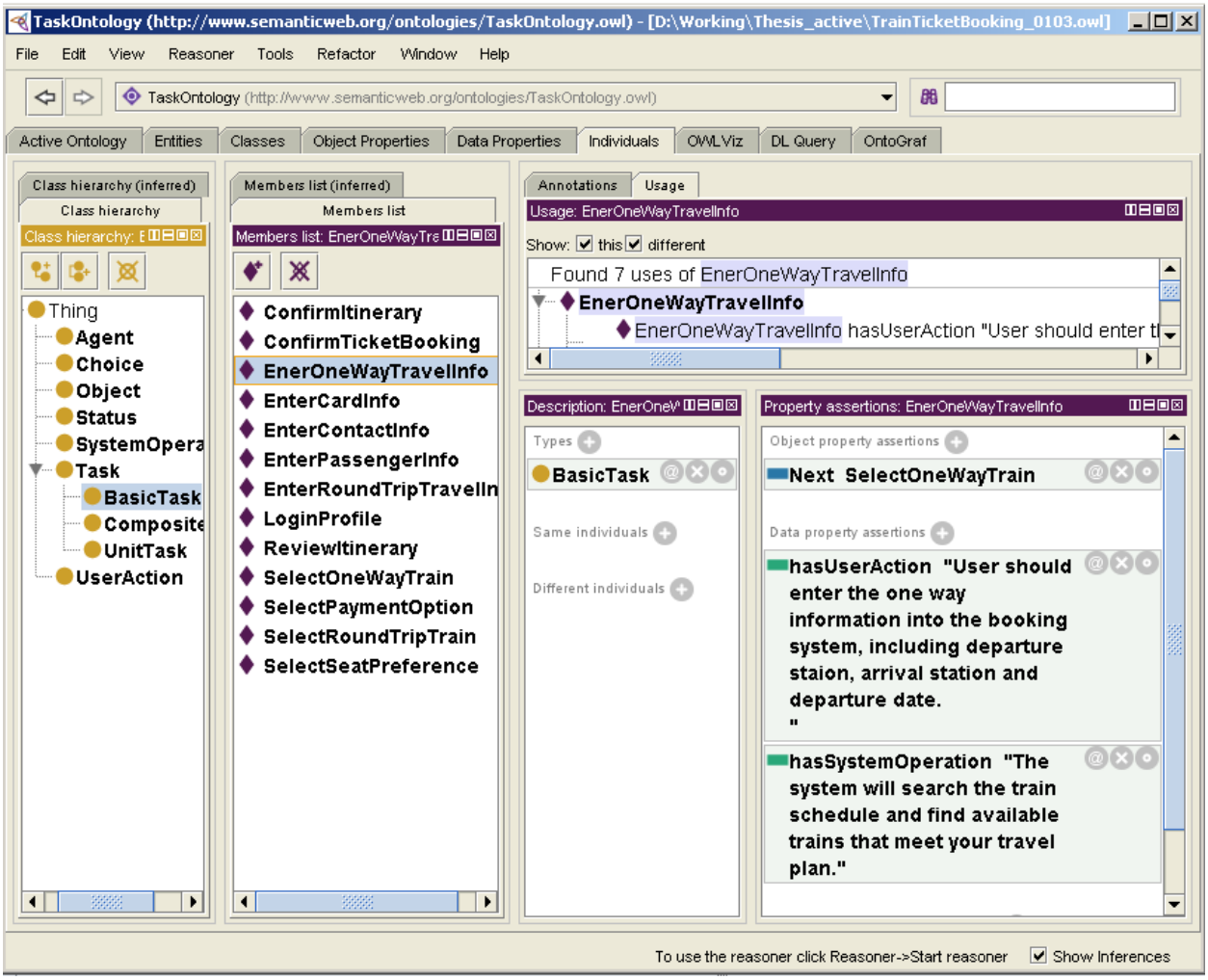

Figure 6.2.1.4 BasicTask instances in the TrainTicketBooking ontology

The BasicTask instances and their properties are shown in Fig, 6.2.1.4. They all have the data properties of hasUserAction and hasSystemOperation which indicate the action users should take and the operations in the system side in a particular BasicTask instance. Except ConfirmTicketBooking, all other instances have the object property Next. That is because ConfirmTicketBooking is the last subtask for the entire task.

6.2.2 Dialogue Process for the Online Train Ticket Booking Service

If the online train ticket booking task is selected by users, the dialogue system will retrieve the knowledge about this task from the train ticket booking task ontology. Fig. 6.2.2.1 is the dialogue interface presented to users for the first subtask in this task.

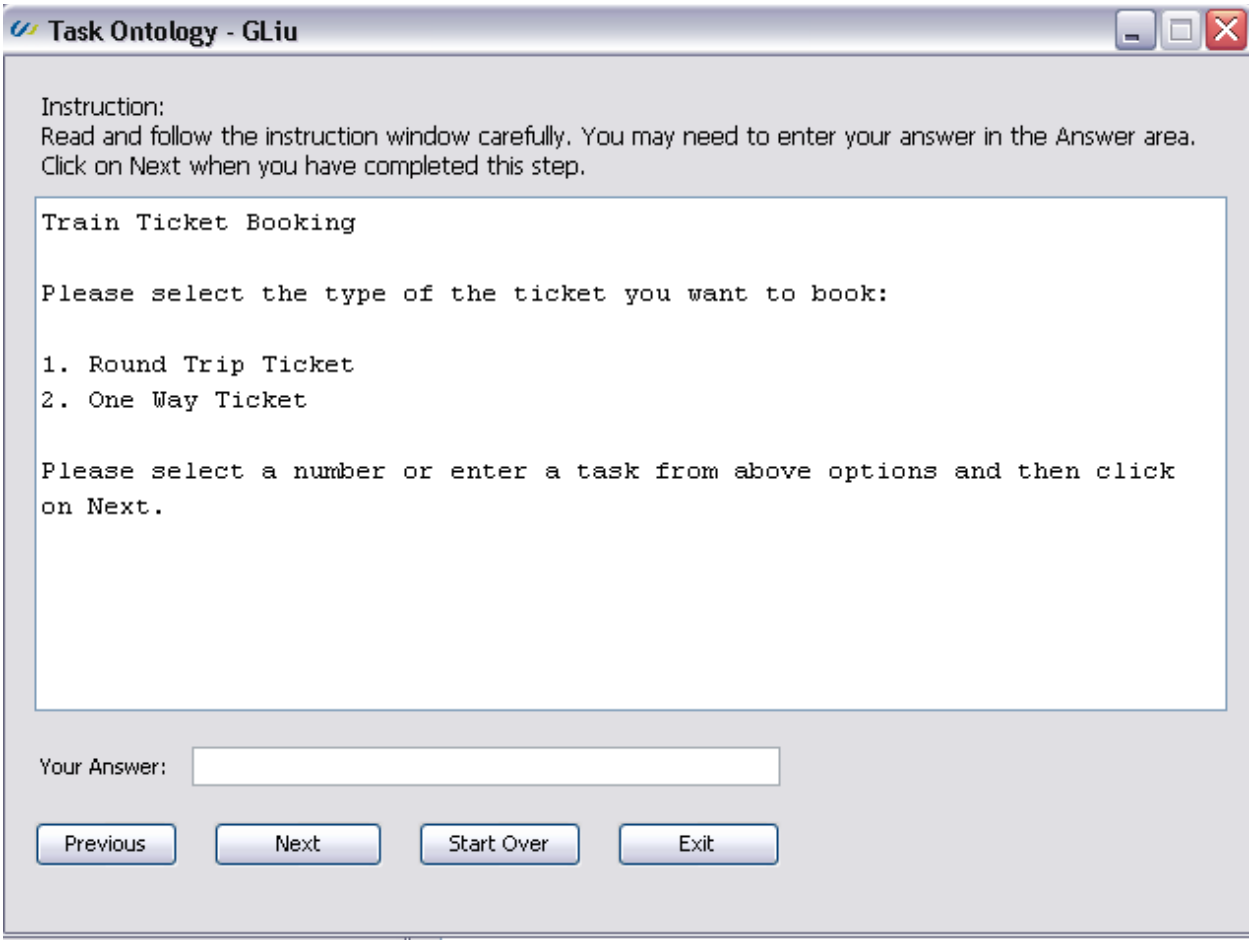

Figure 6.2.2.1 Dialogue interface for the train ticket booking task

In the larger window, two options are provided to users: Round Trip Ticket or One Way Ticket. Users can enter the number or the name of the task they want. The next subtask of the

dialogue system will be determined by users' selection. In this example: either Round Trip Ticket or One Way Ticket.

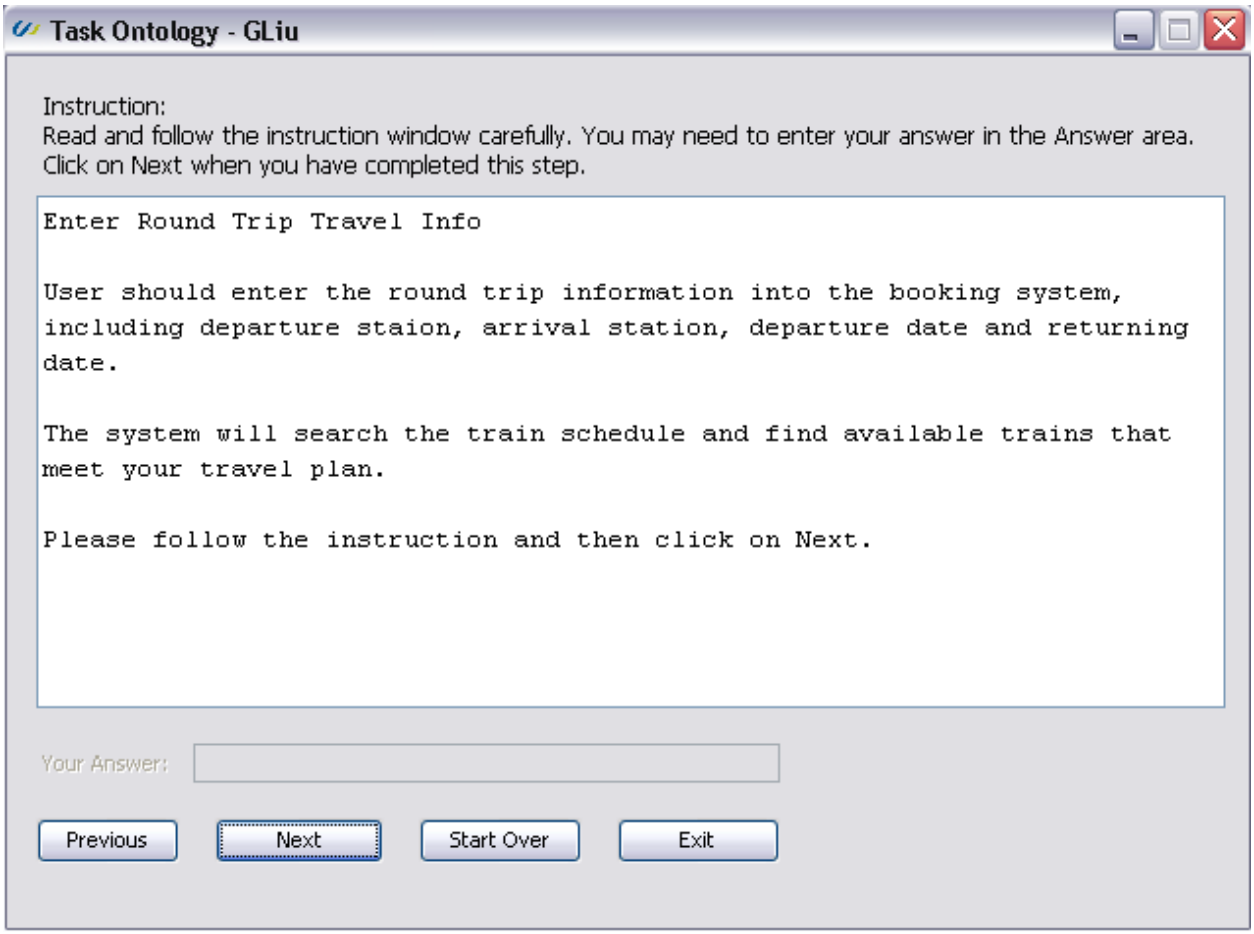

Figure 6.2.2.2 Dialogue interface for the enter round trip travel info task

If the round trip ticket is chosen by users, the system will start the subtask on round trip booking. The first BasicTask is the EnterRoundTripTravelInfo, so it will show up in the dialogue interface (see Fig. 6.2.2.2). Once users have completed this subtask, they should click on Next to continue. Then the dialogue system will instruct users to SelectRoundTripTrain which is the

subtask following the Enter Round Trip Travel Info. Fig. 6.2.2.3 is the dialogue interface for SelectRoundTripTrain.

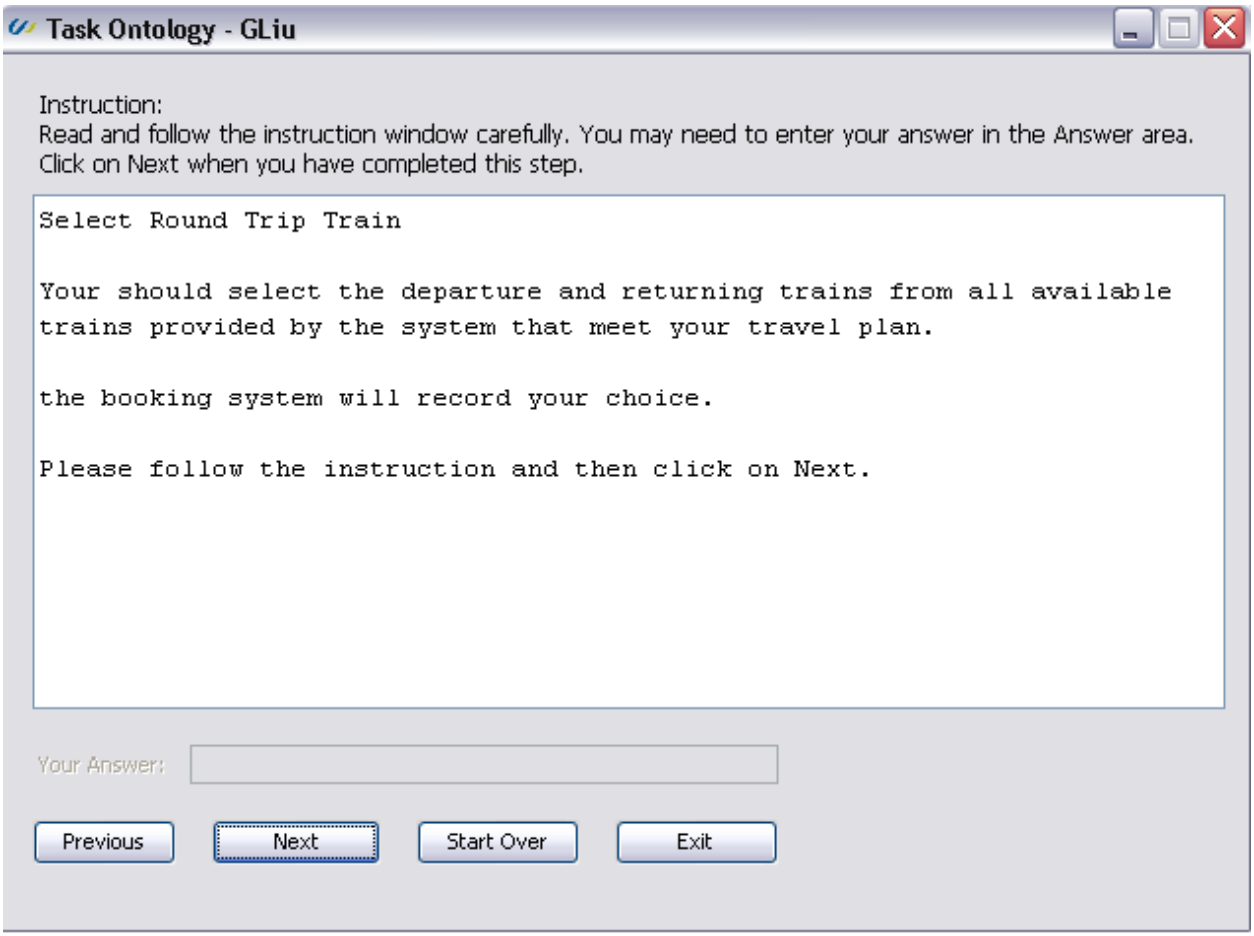

Figure 6.2.2.3 Dialogue interface for the select round trip train task

In the first step, if users select the one way ticket, then the EnterOneWayTravelInfo will be displayed on the dialogue interface instead of the EnterRoundTripTravelInfo. Fig. 6.2.2.4 shows the dialogue interface for the EnterOneWayTravelInfo task.

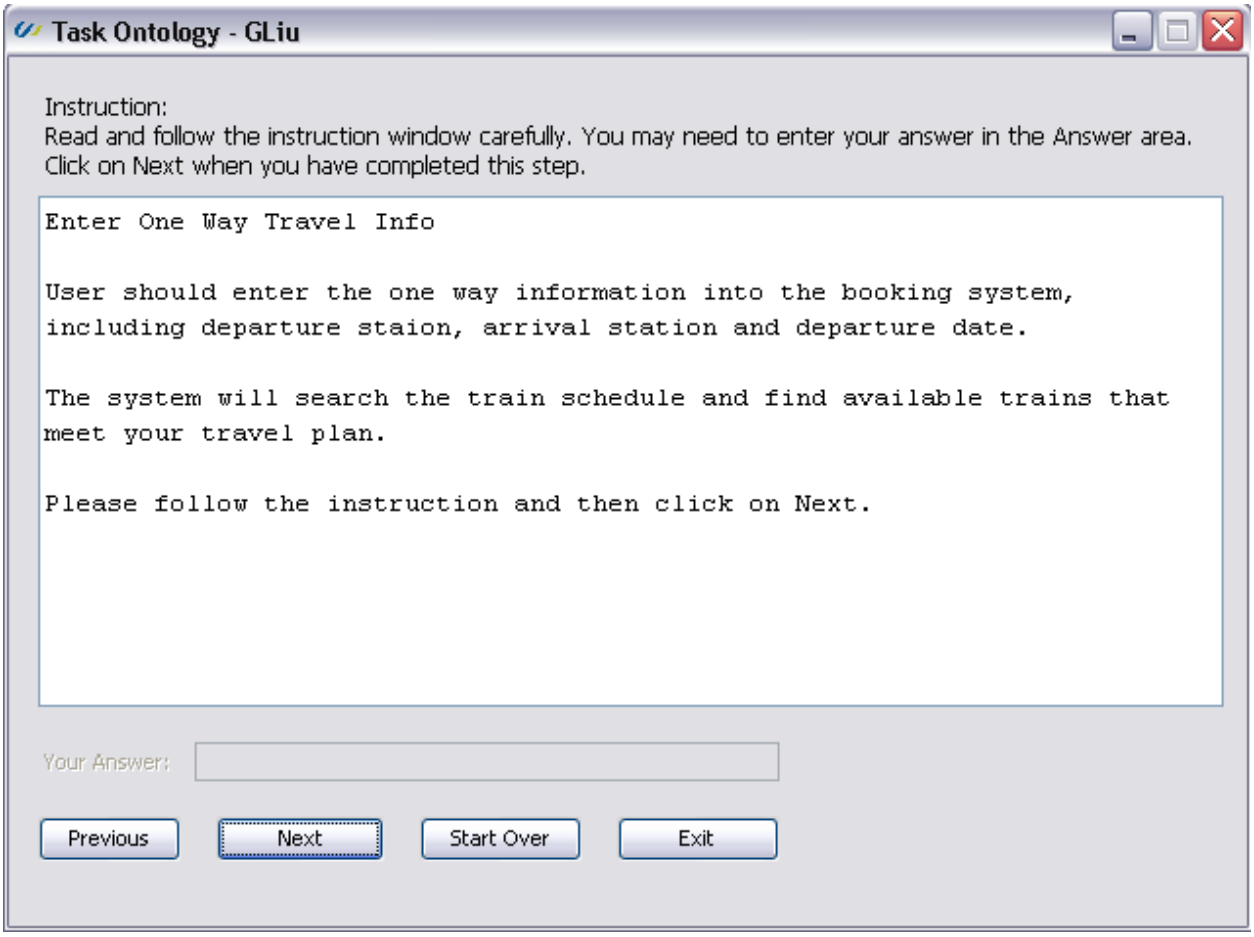

Figure 6.2.2.4 Dialogue interface for the enter one way travel info task

Following the EnterOneWayTravelInfo, the SelectOneWayTrain will be the next subtask. After either SelectRoundTripTrain or SelectOneWayTrain is finished, the ReviewIntinerary will be the next subtask to both. Users will see the dialogue interface shown in Fig. 6.2.2.5.

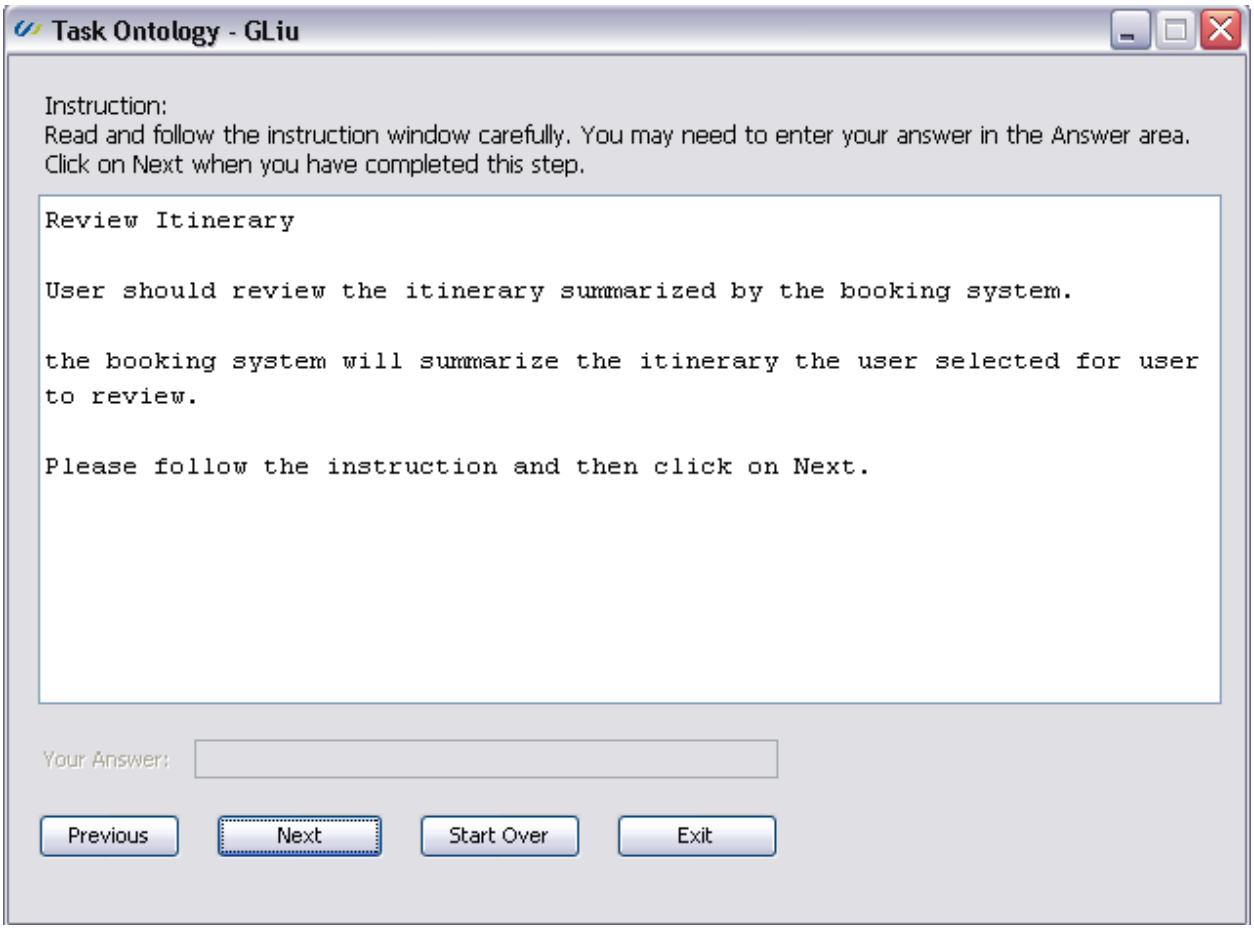

Figure 6.2.2.5 Dialogue interface for the review itinerary

Users can follow the instruction by the dialogue system to complete the entire task. The dialogue interface will provide options of Start Over and Exit for users to continue with other tasks, to repeat the current task or to exit from the dialogue system.

CHAPTER 7 CONCLUSION AND FUTURE WORK

This thesis presents a new approach, the task ontology based dialogue management to achieve domain independent dialogue systems. A task ontology model is created combining knowledge and methods from the areas of ontology and task modeling. Knowledge about a certain task in a domain is modeled in its task ontology through ontology instantiation of the model. The structure of the dialogue manager is also modified to contain an ontology reasoning component retrieving task specific knowledge from the task ontology. So the task or domain knowledge is separated from the development of a dialogue system.

The task ontology model has been used to create task ontology for two tasks from different domains. Based on the model and the modified dialogue manager, a dialogue system has been developed and experimented with the two tasks. The experimentation demonstrates that the dialogue system is capable of handling different tasks. In other words, the dialogue system is task or domain independent.

Dialogue systems have been used in many areas but all are limited to specific domains. On the other hand, the development and maintenance of a dialogue system is expensive. The method developed in this thesis would promote the knowledge reuse and sharing of dialogue systems and help reduce the development cost. Future work may investigate methods automating the task ontology construction to ease the ontology creation process for various tasks.

REFERENCES

[1] M.F. McTear, "Spoken Dialogue Technology: Enabling the Conversational User Interface,"

ACM Computing Surveys, vol. 34, no. 1, pp. 90–169, 2002.

[2] T. H. Bui, "Multimodal Dialogue Management: State of the Art," Centre for Telematics and Information Technology, University of Twente, Enschede, Netherlands, Tech. Rep. TR-CTIT-06-01, 2006.

[3] J. Lester, K. Branting and B. Mott. "Conversational Agents," in *The Practical Handbook of Internet Computing*. M. P. Singh, Ed. New York: Chapman & Hall, 2004, pp. 220-240.

[4] M.O. Dzikovska et al., "Linking Semantic and Knowledge Representations in a Multi-Domain Dialogue System Source," *Journal of logic and computation,* vol.18, no. 3, pp. 405-430, 2007.

[5] B. Lin, H. Wang and L. Lee, "A Distributed Agent Architecture for Intelligent Multi-Domain Spoken Dialogue Systems," *IEICE Trans. Informat. Syst.,* vol. E84-D, no. 9, pp. 1217-1230, 2001.

[6] M. Mourao, R. Cassaca and N. Mamede, "An Independent Domain Dialogue System through a Service Manager", in *Proceedings of ESTAL,* Alicante, Spain, 2004.

[7] C. Lee, S. Jung, K. Kim, D. Lee and G. G. Lee, "Recent Approaches to Dialogue Management for Spoken Dialogue Systems," *Journal of Computing Science and Engineering*, vol. 4, no. 1, pp. 1-22, 2010.

[8] B. Pakucs, "Towards dynamic multi-domain dialogue processing," in *Proc. of the European Conference on Speech, Communication and* Technology, 2003, pp. 741−744.

[9] K. Komatani, S. Ikeda, T. Ogata and H. G. Okuno, "Managing Out-Of-Grammar Utterances by Topic Estimation with Domain Extensibility in Multi-Domain Spoken Dialogue Systems," *Speech Communication*, vol. 50, pp. 863-870, 2008.

[10] F. M. Martins, A. Mendes, M. Viveiros, J. P. Pardal, P. Arez, N. J. Mamede and J. P. Neto, "Reengineering a Domain-Independent Framework for Spoken Dialogue Systems," in *ACL Workshops: Software Engineering, Testing, and Quality Assurance for Natural Language Processing*, Association for Computational Linguistics, 2008, pp. 68-76.

[11] D. Bohus and I. Rudnicky, "The RavenClaw Dialog Management Framework: Architecture and Systems," *Computer Speech and Language*, vol. 23, pp. 332-361, 2009.

[12] J. van Oijen, W. van Doesburg and F. Dignum, "Goal-Based Communication Using BDI Agents as Virtual Humans in Training: An Ontology Driven Dialogue System", *Agents for Games and Simulations II*, Frank Dignum, Ed. Berlin: Springer, 2011, pp. 38-52.

[13] T. W. Bickmore, D. Schulman and C. L. Sidner, "A reusable framework for health counseling dialogue systems based on a behavioral medicine ontology," *Journal of Biomedical Informatics*, vol. 44, no. 2, pp. 183-197, 2011.

[14] J. P. Pardal and N. J. Mamede, "Starting to Cook a Coaching Dialogue System in the Olympus framework," in *Proceedings of the Paralinguistic Information and its Integration in Spoken Dialogue Systems Workshop*, 2011, pp. 255-267.

[15] X. Zhang, "An interactive approach of ontology-based requirement elicitation for software customization." Master's thesis, School of Computer Science, University of Windsor, Windsor, Canada, 2011.

[16] J. Allen, D. Byron, M. Dzikovska, G. Ferguson, L. Galescu and A. Stent, "An Architecture for a Generic Dialogue Shell," *Natural Language Engineering*, vol. 6, no. 3, pp. 1-16, 2000.

[17] A. M. Tarta, "Task Modeling in Systems Design," *Studia Univ. Babes-Bolyai, Informatica*, vol. XLIX, no. 2, pp. 37-44, 2004.

[18] D. Traum and S. Larsson, "The information state approach to dialogue management," in *Current and New Directions in Discourse and Dialogue*, J. van Kuppevelt and R. Smith, Eds. Kluwer, 2003, pp. 325–353.

[19] E. Levin, R. Pieraccini, and W. Eckert, "A stochastic model of human-machine interaction for learning dialogue strategies," *IEEE Transactions on Speech and Audio Processing*, vol. 8, no. 1, pp. 11-23, 2000.

[20] S. Singh, D. Litman, and M. Kearns, "Optimizing dialogue manage-ment with reinforcement learning: xxperiments with the NJFun system," *Journal of Artifcial Intelligence Research*, vol. 16, pp. 105-133, 2002.

[21] J. D. Williams and S. Young, "Partially observable Markov decision processes for spoken dialog systems," *Computer Speech and Language*, vol. 21, pp. 393−422, 2007.

[22] T. Bui, J. Zwiers, M. Poel, and A. Nijholt, "Affective dialogue management using factored POMDPs," in *Interactive Collaborative Information Systems*, R. Babuška and F.C.A. Groen, Eds. Berlin: Springer-Verlag, 2010, pp. 207–236.

[23] S. Varges, G. Riccardi, S. Quarteroni and A. V. Ivanov, "POMDP concept policies and task structures for hybrid dialog management," in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2011, pp. 5592 – 5595.

[24] C.Wai, H. M. Meng and R. Pieraccini, "Scalability and portability of a belief network based dialog model for different application domains," In *Proceedings of the first international conference on Human language technology research*, Association for Computational Linguistics, 2001.

[25] T. Paek and E. Horvitz, "Conversation as action under uncertainty," In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pp. 2000, 455–464.

[26] N. Blaylock and J. Allen, "A collaborative problem-solving model of dialogue," in *6th SIGdial Workshop on Discourse and Dialogue*, 2005.

[27] Y. Wilks, R. Catizone, S. Worgan and M. Turunen, "Review: Some background on dialogue management and conversational speech for dialogue systems," *Computer Speech and Language*, vol. 25, pp. 128-139, 2011.

[28] H. Aust, M. Oerder, F. Seide, and V. Steubbuss, "The Philips automatic train timetable information system," *Speech Communication,* vol. 17, pp. 249–262, 1995.

[29] R. Smith and D. R. Hipp. *Spoken Natural Language Dialog Systems: A Practical Approach*. New York, NY: Oxford University Press, 1994.

[30] D. Goddeau, H. Meng, J. Polifroni, S. Seneff, and S. Busayapongchai, "A Form-Based Dialogue Manager for Spoken Language Applications," in *Proc. ICSLP '96*, vol. 2, 1996, pp. 701-704.

[31] A. I. Rudnicky, E. Thayer, P. Constantinides, C. Tchou, R. Shern, K. Lenzo, W. Xu and A. Oh, "Creating natural dialogs in the Carnegie Mellon Communicator system," in *Sixth European Conference on Speech Communication and Technology*, 1999.

[32] M. Johnston, S. Bangalore, G. Vasireddy, A. Stent, P. Ehlen, M. A. Walker, S. Whittaker and Preetam Maloor, "Match: An architecture for multimodal dialogue systems," in *ACL '02 Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, 2002, pp. 376-383.

[33] D. R. Traum and J. Rickel, "Embodied agents for multi-party dialogue in immersive virtual worlds," in *AAMAS '02 Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 2*, 2002, pp. 766-773.

[34] D. Hofs, R. op den Akker, and A. Nijholt, "A generic architecture and dialogue model for multimodal interaction," in *Proceedings 1st Nordic Symposium on Multimodal Communication*, 2003.

[35] K. Komatani, K. Tanaka, H. Kashima and T. Kawahara, "Domain-independent spoken dialogue platform using key-phrase spotting based on combined language model", In

EUROSPEECH-2001, 7th European Conference on Speech Communication and Technology, Aalborg, Denmark, 2001, pp. 1319-1322.

[36] M. O. Dzikovska, J. F. Allen and M. D. Swift, "Linking Semantic and Knowledge Representations in a Multi-Domain Dialogue System," *Journal of Logic and Computation*, vol. 18, no. 3, pp. 405-430, 2008.

[37] S. Ikeda, K. Komatani, T. Ogata and H. G. Okuno, "Extensibility verification of robust domain selection against out-of-grammar utterances in multi-domain spoken dialogue system," in *INTERSPEECH-2008, 9th Annual Conference of the International Speech Communication Association*, 2008, pp. 487-490.

[38] K. Komatani, N. Kanda, M. Nakano, K. Nakadai, H. Tsujino, T. Ogata and H. G. Okuno, "Multi-domain spoken dialogue system with extensibility and robustness against speech recognition errors," in *Proceeding SigDIAL '06 Proceedings of the 7th SIGdial Workshop on Discourse and Dialogue Association for Computational Linguistics Stroudsburg*, 2009.

[39] J. Allen, G. Ferguson and A. Stent, "An architecture for more realistic conversational systems," in *IUI '01 Proceedings of the 6th international conference on Intelligent user interfaces*, 2001.

[40] C. Lee, S. Jung, S. Kim and G. G. Lee, "Example-Based Dialog Modeling for Practical Multi-Domain Dialog System," *Speech Communication*, vol. 51, pp. 466-484, 2009.

[41] D. Bohus and A. Rudnicky, "RavenClaw: Dialog Management Using Hierarchical Task Decomposition and an Expectation Agenda," In *Proc. of the European Conference on Speech, Communication and Technology*, 2003, pp. 597−600.

[42] T. Bui, M. Rajman and M. Melichar, "Rapid Dialogue Prototyping Methodology," In *Proc. of the international conference on Text, Speech, and Dialogue*, pp. 579−586, 2004.

[43] S. Larsson and D. R. Traum, "Information state and dialogue management in the TRINDI dialogue move engine toolkit," *Natural Language Engineering*, vol. 6, pp. 323−340, 2006.

[44] T. R. Gruber, "A Translation Approach to Portable Ontology Specifications," *Knowledge Acquisition*, vol. 5, no. 2, pp. 199-220, 1993.

[45] N. F. Noy and D. L. McGuinness, "Ontology development 101: A guide to creating your first ontology," Technical Report SMI-2001-0880, Stanford Medical Informatics, 2001.

[46] Enrico Franconi, "Description Logics for Conceptual Design, Information Access, and Ontology Integration: Research Trends," *Networks journal of the philosophy of Artificial Intelligence and Cognitive Science*, special issue on the Semantic Web, No. 2, 2003.

[47] D. L. McGuinness and F. van Harmelen, “OWL Web Ontology Language Overview.” Internet: http://www.w3.org/TR/owl-features , 2004 [Jan. 1, 2012]

[48] G. Dobson and P. Sawyer, "Revisiting Ontology-Based Requirements Engineering in the Age of the Semantic Web," in *Proceedings of the International Seminar on Dependable Requirements Engineering of Computerised Systems at NPPs*, 2006.

[49] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof and M. Dean, “SWRL: A Semantic Web Rule Language Combining OWL and RuleML.” Internet: http://www.w3.org/Submission/SWRL, 2004.

[50] M. Gatius and M. Gonzalez, “Using Ontologies for Improving the Communication Process in a Dialogue System,” in *XXI Congreso de la SEPLN, I Congreso Espa˜nol de Inform´atica (CEDI),* 2005.

[51] D. Sonntag and M. Romanelli, “A multimodal result ontology for integrated semantic web dialogue applications”, in *Proceedings of the 5th Conference on Language Resources and Evaluation (LREC),* 2006.

[52] D. Milward and M. Beveridge, “Ontologies and the structure of dialogue,” in *Proceedings of the Eight Workshop on the Semantics and Progmatics of Dialogue, Catalog’04*, 2004, pp. 69-76.

[53] A. Flycht-Eriksson and A. Joensson, “Ontology-driven information-providing dialogue systems,” in *9th Americas Conference on Information Systems (AMCIS) 2003 Proceedings*, 2003, pp. 2956-2967.

[54] D. Bohus, A. Raux, T. K. Harris, M. Eskenazi and A. I. Rudnicky, “Olympus: an open-source framework for conversational spoken language interface research,” in *NAACL-HLT ’07: Proceedings of the Workshop on Bridging the Gap: Academic and Industrial Research in Dialog Technologies*, 2007.

[55] S. Young, M. Gasic, S. Keizer, F. Mairesse, J. Schatzmann, B. Thomson and K. Yu, “The hidden information state model: a practical framework for POMDP-based spoken dialoguemanagement,” *Computer Speech and Language*, vol. 24, no. 2, pp. 150–174, 2009.

[56] T. Heinroth, D. Denich, and W. Minker, “A Multitasking Approach to Adaptive Spoken Dialogue Management,” *Universal Access in HCI, Part IV, HCII 2011, LNCS 6768*, C. Stephanidis, Ed., 2011, pp. 42 – 51.

[57] F. Patern´o, “Task models in interactive software systems,” In *Handbook of Software Engineering and Knowledge Engineering*, S. K. Chang, Ed. World Scientific Publishing, 2001.

[58] J. Annett and K. Duncan, "Task analysis and training in design," *Occupational Psychology*, vol. 41, pp. 211–221, 1967.

[59] S. Card, T. Moran, T. and A. Newell. The Psychology of Human-Computer Interaction. Lawrence Erlbaum Associates, 1983.

[60] R. Hartson and P. Gray, "Temporal aspects of tasks in the User Action Notation," *Human Computer Interaction*, vol. 7, pp. 1-45, 1992.

[61] G. van der Veer, B. Lenting and B. Bergevoet, "GTA: Groupware Task Analysis - Modeling Complexity," *Acta Psychologica*, vol. 91, no. 3, pp. 297–322, 1996.

[62] M. Welie, "Task-Based User Interface Design." Doctoral dissertation, Graduate School for Information and Knowledge Systems, Vrije Universiteit, Amsterdam, 2001.

[63] B.E. John, B. E. and D. E. Kieras, 'The GOMS family of user interface analysis techniques: Comparison and contrast," *ACM Transactions on Computer-Human Interaction*, vol. 3, pp. 320–351, 1996.

[64] G. Mori, F. Patern'o and C. Santoro, "CTTE: Support for developing and analyzing task models for interactive system design," *IEEE Transactions on Software Engineering*, vol. 28, no. 8. pp. 797- 813, 2002.

[65] D. Navarre, P. Palanque and M. Winckler, "Task models and system models as a bridge between HCI and software engineering," In *Human-Centered Software Engineering*, 1st ed., A. Seffah, J. Vanderdonckt, and M. C. Desmarais, Eds. Springer, 2009, pp. 357-385.

VITA AUCTORIS

NAME: Guoying Liu

PLACE OF BIRTH: Shangrao, Jiangxi, China

YEAR OF BIRTH: 1968

EDUCATION: University of Science and Technology of China, Hefei, China

1986-1991 B.Sc.

Dalhousie University, Halifax, Nova Scotia

2004-2006 MLIS