Formalizing the Role of Goals in the Development of Domain-Specific Ontological Frameworks

Faezeh Ensan
Faculty of Computer Science
University Of New Brunswick
Fredericton, Canada
faezeh.ensan@unb.ca

Weichang Du
Faculty of Computer Science
University Of New Brunswick
Fredericton, Canada
wdu@unb.ca

Abstract

In this paper we propose a high-level scheme that assists ontology engineers develop appropriate ontological frameworks. By ontological frameworks we mean those structures that specify particular phases and also provide implemented components for developing ontologies. Based on the i conceptual modeling framework, our proposed scheme guides ontology engineers by customizing a suitable ontological framework based on their preferences and their specific domain necessities. In the proposed scheme, We specify the users of an ontological framework, their high-level softgoals as well as the goals that contribute to these softgoals. We exploit business processes and bind them to the goals in order to implement the framework.*

1. Introduction

The field of ontology engineering has witnessed a wide range of algorithms, tools and methodologies for developing and maintaining different types of ontologies. In [11] we have classified different ontological frameworks and have analyzed their features from a domain-centric point of view. By ontological frameworks we mean those proposals that specify particular phases and also provide implemented components for developing ontologies. Several frameworks address the issue of constructing and maintaining domain ontologies in the literature. Maedche et al. [17] have designed a framework for ontology learning. Their framework provides ontology engineers with proper tools for modeling and designing domain ontologies. It also offers semi-automatic techniques for extracting ontological concepts from structured and natural language documents.

Text2Onto [9] is another ontology learning framework which builds on the concepts presented in [17] and complements it by introducing new features. The authors suggest that extracted ontological concepts be represented in a meta-level model; consequently, the framework will be independent of any ontology modeling language. They also suggest methods for adapting an under-development ontology to the new changes that have been applied on the source data set. Ontolearn [19] is another framework for learning domain based ontologies from a set of relevant documents. It includes an algorithm for identifying the proper sense (meaning) of each term in a compound phrase using machine learning techniques. Kotis et al. [16] have introduced HCOME, a human centered methodology for developing domain ontologies, which is supported by the HCONE ontology engineering environment. HCOME supports the end-users, who may not be familiar with ontology building languages and logic, to manipulate and edit their ontology on a daily basis, and also communicate with each other to share their ideas.

The existing models mostly exploit knowledge extraction, integration and maintenance methods to form the best framework for creating suitable domain ontologies. The most important questions that arise are how should a framework employ different components and methods? Which algorithms, techniques and architectures best fulfil the initial necessities of the domain? How can a developer change some parts of a framework based on new requirements and goals? Is a complicated framework always the best choice or an engineer can design a suitable naive version for simple problems and limited budgets? How can domain-specific features influence the framework? Can a general framework be appropriate for all domains or different domains require different operators and evaluation criteria?

Our objective in this paper is to define some methods, guidelines and criteria for creating ontological frameworks, while considering the above discussed important points. This proposed high-level scheme assists ontology engineers to specify the goal of an ontological framework and its capabilities, the domain knowledge it intends to exploit, and

1530-1605/08 $25.00 © 2008 IEEE

also to indicate its dependency to a specific domain. We believe this scheme is a high-level technique, because it has been proposed for developing ontological frameworks which themselves can be utilized in order to create and maintain ontologies.

With the purpose of specifying and tracking the 'intentions' and 'goals' of an ontological framework during the analysis and design phases, we utilize the *i** [22] model and extend it to accommodate suitable features for developing ontology development and maintenance frameworks. *i** is a conceptual framework for modeling processes and indicating the dependencies among several agents of a software system for specifying the goals to be achieved and the tasks to be performed. Our proposal consists of four major phases. In the first phase, we specify specific actors, and their distinct soft goals. Later in the second phase we introduce the popular hard-goals that can realize the specified abstract softgoals. We then analyze the goals and tasks of the ontological framework and propose a set of related domain specific constrains in the analysis phase. Finally, in the implementation phase, we suggest the employment of the business process definition and also UML activity diagrams for modeling an ontological framework.

This paper has been organized in four sections. The next section gives a brief background on the *i** conceptual modeling framework and also other related work. In section 3, we thoroughly describe our scheme for creating domain-specific ontological frameworks. Ultimately, Section 4 provides some concluding remarks and some venues for future work.

2. Background and Related Work

*i** is a conceptual modeling framework that attempts to analyze processes from an intentional view. On the contrary to existing modeling techniques such as work flow diagrams or Petri net-based process models that mostly focus on 'what' and 'how' in a process, *i** considers 'why' in its modeling procedure. Why an agent should perform a specific task? Why a task needs some resources? And so on. *i** provides some modeling elements and introduces two types of diagrams which facilitate tracking and representing 'intention' in its process.

The *i** framework consists of two models: the Strategic Dependency (SD) model and the Strategic Rationale (SR) model. SD models the dependencies among different actors of a process. It consists of a set nodes and links where nodes represent actors and links represent their dependencies. There are four types of dependencies: goal, task, resource and softgoal. The 'goal' dependency occurs when an actor is dependent to another to achieve a specific goal. The depending actor does not have any idea about how the other actor achieves the goal. It is only concerned about the output which is a certain state that represents the requested goal. The dependency among two actors is the 'task' dependency, when an actor is dependent on the other for doing a specific task. Two actors have the 'resource' dependency, when one actor needs another actor to provide it with a specific type of resource. The resource can be an informational resource or a physical resource. Finally, the softgoal dependency is similar to the goal dependency but instead of a 'goal', a 'softgoal' should be satisfied. Softgoals represent non functional requirements like flexibility or user-friendliness usually expressed in informal statements [18].

The strategic rationale model, describes a process in more details. While SD focuses on an external view of dependencies among different actors, SR shows the internal implementation of the demanded tasks and goals of each actor. There are four types of nodes: goals, tasks, resources and softgoals and two types of links: means-end and task decomposition. A means-end relationship indicates that the 'means' node is a means for achieving, performing, obtaining or satisfying the end node. A task-decomposition link shows the sub components of a specific task. In Section 3 we will use *i** in order to extract a suitable domain specific ontological framework.

In addition to the benefits of *i** such as support for business process re-engineering based on the new goals and tracking intention during requirement analysis, we also pursue two other reasons for exploiting the *i** modeling framework in our scheme. Firstly, by employing *i** , the developer of an ontological framework will focus more on the real goals of the project and therefore will be able to create a suitable product based on the project budget, available time, domain-specific features and the number of accessible domain experts and ontology engineers. So far, diverse techniques and algorithms have been proposed for extracting and maintaining ontologies that sometimes make selection a hard task. For example, where there are several ontology creation techniques and tools that can be utilized in a non-distributed situation, a number of architectures and techniques have also been proposed for a distributed environment. Using *i** models, the developer can see and compare different alternatives and investigate their benefits and costs and their fitness to the domain which an ontology should be created for.

The second reason is that using *i** framework, the developer can spot the goals and tasks that have been already implemented and reuse them. For example, there are various algorithms for extracting ontological concepts from natural language text. The developer can use these existing algorithms for the task of 'extract concept from the resource' when the resource is a natural language text. In our methodology, we have introduced 'domain-specific constrains' which can be assigned to the tasks and goals of the *i** model. The developer should select from those al-

ready existing techniques that satisfy the indicated domain-specific features and constrains.

Fernandez-Lopez et al. [12, 10] have provided good surveys of existing methodologies for creating and maintaining ontologies. Uschold and Grninger methodology [21], TOVE [15] and METHOONTOLOGY [13] are some of the well-known methodologies for building ontologies. Uschold and Grninger methodology is comprised of four main activities: identifying scope and purpose of the target ontology; building ontology; evaluating result ontology and providing perfect documentation. Building the ontology itself is comprised of capturing important concepts and relationships, coding these concepts to the final ontology language and also integrates this created ontology with other close and useful ontologies. Nonetheless, we have not seen any methodology or guidance criteria in the literature for creating ontological frameworks that are used to extract and maintain ontologies.

Goal based analysis and specially the *i** framework has been widely used by several methodologies and frameworks in areas other than ontology engineering. Tropos [7] is a software development methodology which adopts the concepts of *i** and extracts requirements, architectural design and detail design of a software system. Santander et. al. in [20] argue the improvement of UML use case diagrams using the *i** organizational models . [6] has used the goal based analysis to identify the risks of a software development/organizational system. Those proposals that apply the *i** framework in ontology creation area are not numerous. They mostly define and design an ontology for expressing the concepts that are related to the goals [14, 8].

3. A Scheme for Creating Domain-Specific Ontological Frameworks

Through the exploitation of the *i** framework, we propose a scheme for creating ontological frameworks. For developing an ontological framework, the 'Who', 'What', 'Why', 'How', 'When' and 'Where' questions should be analyzed in different levels of abstraction [23, 5]. Except for the developer who uses our methodology to create ontological frameworks, there are two other types of users who will use the products of this scheme ('Who'): the end-users who benefit from the created ontology and the users of the ontological framework who manipulate the ontology using the capabilities provided by the framework. There are also two types of entities in the system ('What'): the ontological concepts which are extracted and the informational resources that are used by different components of the framework to develop an ontology. By using the *i** framework we mostly analyze 'Why' an ontological framework should be developed. We also propose some pre-defined tasks and goals in our scheme that can assist the developer of an ontology in answering the How, When and Where questions.

Our methodology can be used for developing ontological frameworks by those customers who may not have a clear idea about the characteristics of the final product that they require. A customer may only know that she/he/it wants a software that develops a suitable ontology, but does not know what is a suitable application for its necessities, the domain features, the budget and the required time. In order to help these customers and the developers of the ontological framework to have a better comprehension of the final product, we have specified the actors of an ontological framework, their main softgoals, and also propose some tasks and goals that can satisfy these high-level softgoals. Commencing the process based on these proposed softgoal, tasks and (hard) goals, the developer can specify other related goals, tasks and subsystems of the final product.

In order to attain implementation issues from the goal analysis models, we have used business processes and also UML activity diagrams. A business process has a specific goal and is responsible for doing some indicated task. Selecting and defining business processes helps developers to improve some particular functionality of the ontological framework in the future revisions of the system. Through our proposed methodology, we specify some goals that are clear enough to be assigned to a business process.

We model the interactions between different business processes using UML activity diagrams. We demonstrate those tasks that can be performed automatically as control classes in the activity diagram. The final framework is comprised of different control and data classes, business processes, workflow diagrams and also domain specific constrains on the tasks.

in section 3.1 we describe a case study from the domain of tourism which has been employed to explain the features of the proposed scheme throughout the paper. We fully describe the four phases of our proposed scheme in Sections 3.2 through to 3.5.

3.1 A Case Study

As a case study, we consider creating an ontological framework for tourism attractions of Canada. Lets suppose that the Canadian Tourism Commission (CTC) [2] has decided to create and maintain an ontology about Canadian tourism attractions. It intends to enhance the existing search engines with this ontology so that they be able to retrieve suitable and important pages related to Canada tourism. (The CTC is referred to as the customer from now on). Assume the customer insists on extracting all of the related Canadian tourism information from web documents which include the official web sites of each Canadian city, the official web sites of tourism attractions of Canadian regions and all of the non-official web sites like weblogs that may

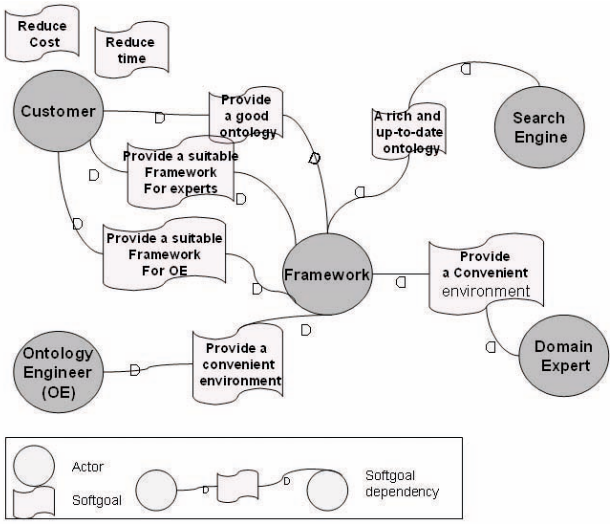

Figure 1. The Actors of an Ontological Framework and Their High-level Softgoals.

have some information about Canadian landmarks and any special events in Canada. Although several ontologies for the classification of tourism attractions already exist in ontology libraries like 'DAML Ontology Library' [1], the customer still prefers to build an ontology from scratch. Consequently other existing ontologies of tourism attraction can not be exploited by the created framework. The end users of the ontology which are created by the framework are search engines that will have the opportunity to enhance their performance on retrieving the information related to queries about Canada tourism.

The customer does not have any clear ideas about the design and implementation issues. For instance it does not know how the application should be deployed, how many ontology engineers or experts are needed, what type of existing open source or commercial tools and algorithms can be exploited? On the other hand it has a clear idea about the budget of the project.

3.2 The Actors and their High-Level Softgoals

We define five types of actors for a typical ontological framework: the customer or stakeholder that supports and owns the project, the ontology engineer who is familiar with the ontology technologies, tools, algorithms and languages and can develop ontology with existing tools or can easily learn to employ new technologies, the domain expert who is completely familiar with the domain of discourse and can resolve any ambiguity about the domain concepts, the end-user who benefits from the ontology which is developed by the framework and finally the framework itself which should be created using the high-level methodology. Figure 1 shows these actors and their high-level soft goals.

In our case study, the Canadian tourism commission is the customer. We can specify five softgoals that every customer aims to reach through the project development phases. As we can see in Figure 1, providing a good ontology, providing a suitable environment for experts and ontology engineers and reducing the cost and time of the project are the goals of any customer. Of course, some of these goals may be more important to some as compared to the others based on the context of the project. For example in some projects, the customer may not worry about the cost of project and can invest as much as needed for the best performance. For such a project, this fact can be taken into account by ignoring the negative effect of costly tasks.

From the ontology engineer and domain expert point of view, a suitable framework should decrease the difficulty of their work; therefore, a suitable framework is one that provides a convenient and easy-to-use environment. The domain expert can be the ontology engineer under some circumstances. In the other words, a single person can play two or more types of roles in the framework.

The end-user in our case study is a search engine that utilizes the created ontology for enhancing its performance. The softgoal of this actor is to access a fresh, rich and up to date ontology.

3.3 The Goals That Contribute to The High-Level Softgoals

Based on the classification of domain specific ontological frameworks provided in [11], we suggest some predefined hard goals that realize the actors' high-level softgoals. In this step, the framework developer can select among the suggested goals or introduce new ones. For each of the suggested goals, the developer should decide about its positive or negative contribution to the softgoals and based on the global business strategies, decide which one should be ignored or be kept in the model. As we can see in Figure 2 the 'provide a good ontology' goal can be satisfied by one of the: 'Extract all available ontological concepts from all of the data sources' or 'Extract ontological concepts from some parts of the available resources'. Furthermore, a good ontology also has to be updated from time to time. ('Evaluate and maintain the ontology').

Observably, all of these goals have a negative contribution to the 'Reduce cost' and 'Reduce time' soft goals. However, their negative contribution is not the same. For example 'Extract all available ontological concepts from all of the data sources' needs more time to be completely

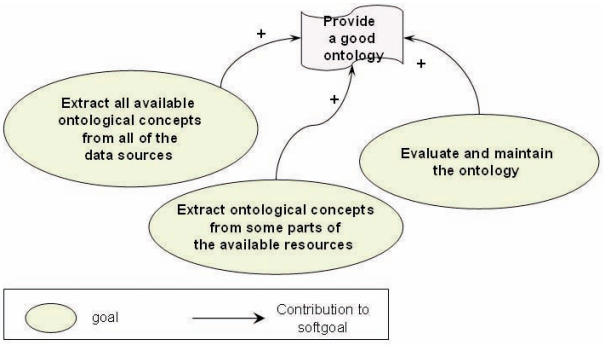

Figure 2. The goals that contribute to the 'Provide a good ontology'

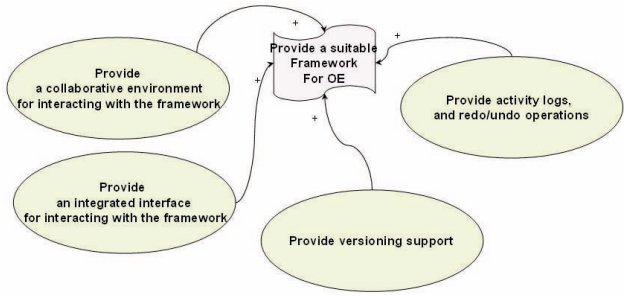

Figure 3. The goals that contribute to the 'Provide a suitable framework for OE' softgoal

performed compared with the 'Extract ontological concepts from some parts of the available resources' goal. The developer can decide to select among them in this stage or keep decomposing and analyzing goals to extract exact tasks that are needed for realizing a specific goal and then make a decision about the strategy that it wants to follow.

In order to analyze the 'Provide a suitable framework for OE' goal we consider two types of interaction that engineers can have with the framework. Ontology engineers can interact with the framework through a collaborative environment in distributed system or through a simple editor when there are few engineers that work in a specific office and can easily mange their work without any conflicts. In fact, this goal helps developer to indicate the distribution of the framework or answer the 'Where' question. Furthermore, a convenient framework provides ontology engineers undo and redo operations and also versioning support. Figure 2 shows the hard goals that contribute to the 'Provide a suitable framework for OE' soft goal.

Domain experts also need to interact with the system. They need some editors that can be easily used without a deep knowledge of the ontological concepts. Obviously our discussion about the need for a collaborative environment can be repeated here for domain experts.

Through the analysis of the system goals; developer should specify the positive or negative contribution of each goal or task to the 'provide a convenient framework' and 'provide a rich and up-to-date ontology'softgoals.

3.4 Analyzing and Decomposing Goals and Tasks

The previous two steps in developing an ontological framework are independent from the problem domain. In this step the domain knowledge will be more utilized in the development process. Suppose that the developers of the framework have made the following decision after the end of the second phase, considering the project budget and resources and also considering the customer priorities and necessities: 'The strategy which is chosen for providing a good ontology is to extract the ontological concept from some parts of the available resource (although there may be several domain related documents and information repositories, the developers decided to only use some useful Web documents that are related to the Canada tourism attraction). Furthermore, since there are several ontology engineers collaborating in a distributed environment, a distributed development tool is necessary. Only one domain expert is designated. The domain expert is familiar with working with user friendly ontology development tools.'

Figure 4 shows some parts of the SR model for the 'framework' actor. As we can see in the figure, for the 'Extract ontological concepts from some parts of the available resources' goal, the framework needs to extract ontological concepts, their taxonomy and the non-hierarchal relationships between them. All of these goals can be realized only when there are suitable extraction algorithms and criteria. According to the model, the framework is dependent on the ontology engineer to provide the suitable algorithms. For evaluating the ontology, the framework is reliant on the domain expert to compare the performance of the search engine before and after using the ontology. In order to enhance the performance, the framework is dependent either on the domain expert to change the Web documents which are exploited by the extracting algorithms or ontology engineers to manually change the ontology or change the employed algorithms.

Figure 5 shows the SR model for the ontology engineer and the 'Find suitable extraction algorithms' softgoal. In this circumstance, the developers decide to only use and select among the already existing extraction algorithms. An

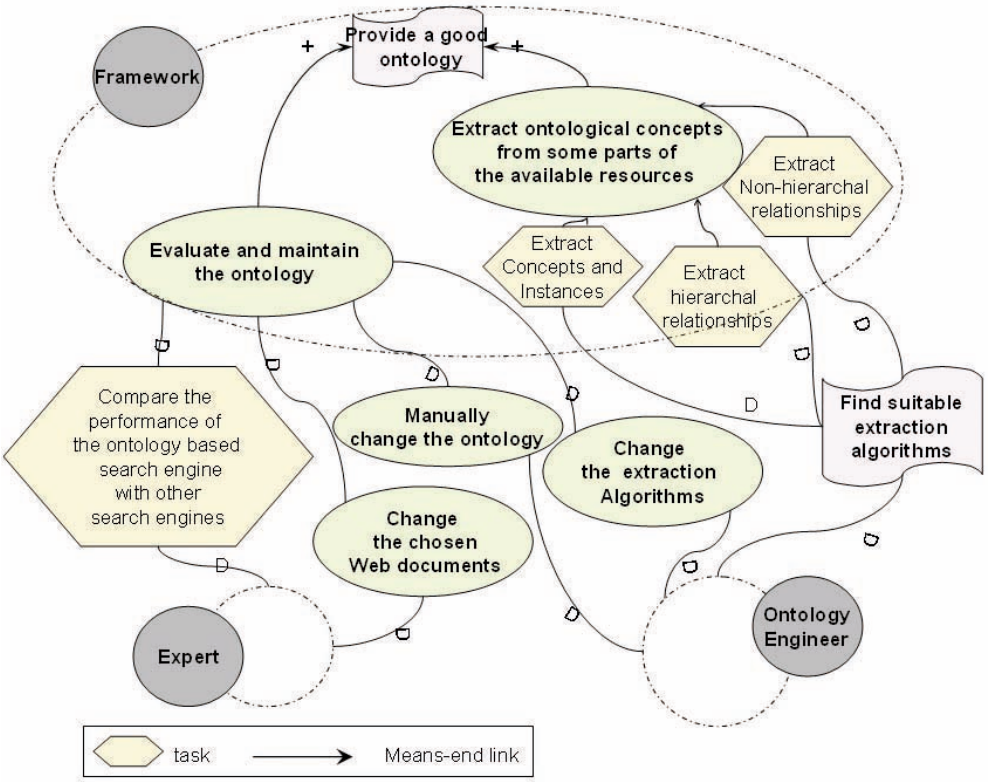

Figure 4. The SR model for the 'provide a good ontology' softgoal.

alternative would be to define and design new extraction algorithms according to the domain necessities. The algorithms should satisfy some domain-specific constrains to be suitable for the framework. The ontology engineer will study different available algorithms based on the identified constrains and select the best ones.

In order to express the domain specific constrains, we introduce some extensions for the concepts of the i^* framework. A 'constrain' is a state that should be held true by a task or by a resource. We use the the same symbol used to show a goal for presenting a 'constrain' in our diagrams with the slight difference that a constrain carries a '$<< constrain >>$' stereotype identifier. Every 'constrain' has some soft or hard features. Features restrict the selection process of the entities that carry that constrain. Analogous to sofgoals, soft feature are those features that do not have a formal definition. We use the softgoal (goal) symbol for soft-features (hard-feature) except that we insert a '$<< feature >>$' annotation to them. Each 'constrain' has a 'task-decomposition' link to the task that causes it (the link between 'Explore different Algorithms based on the satisfaction of constrains and select some of the best' and 'Constrain on Extraction Algorithms' in fig 5). The link between a 'feature' and its 'constrain' is of the 'means-end' type. 'Constrains on Extraction Algorithms' has some features such as the selected algorithms should be able to 'Extract the places, attractions, and operating hours from the context'. Dealing with the constrains and investigating whether they are satisfied or not is up to the ontology engineers in this framework. The framework should save the extracted constrains and allow ontology engineers to manipulate them. Hence, the framework is dependent on the ontology engineers to provide and satisfy the constrains.

Figure 6 shows the SR model for the ontology engineer actor and analyzes the 'Change the extraction algorithms' goal. The ontology engineer is dependent on the expert to find an approximation of precision and recall of each of exiting algorithm so that he/she can rank the algorithms and change them. In the figure, the 'Find an approximation of precision and recall of tourism concepts For each algorithm' has been realized by two tasks. The expert should Select some parts of the ontology concepts and analyze their precision and expand the result to all concept' , then 'Select a set of related tourism concepts and see how much they can be supported by the ontology; expand recall to all concept'. These tasks require that an user friendly ontology browser

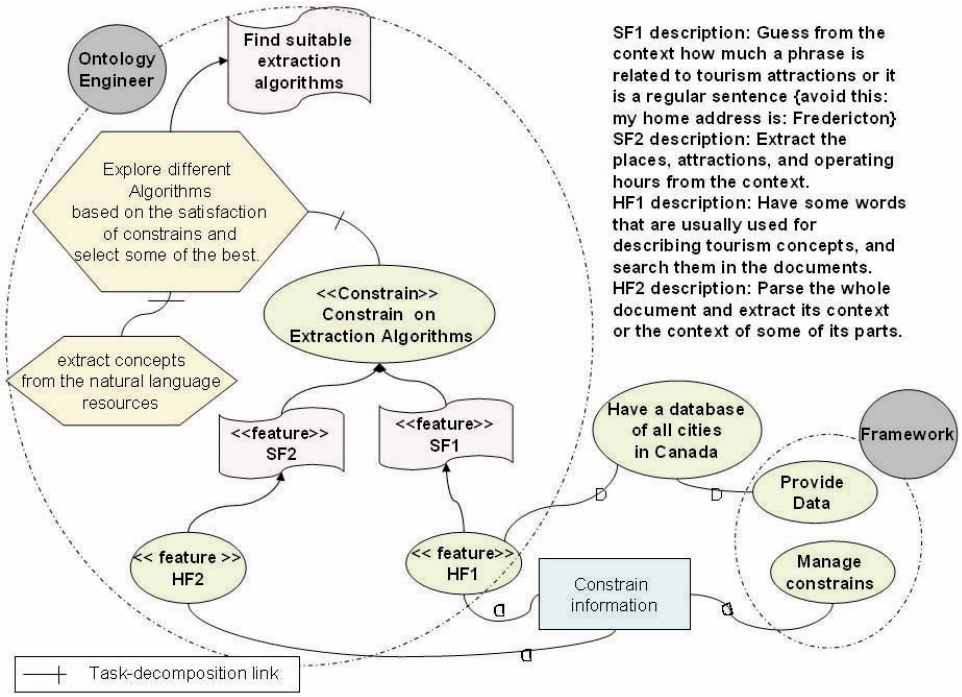

Figure 5. The SR model for the 'Find suitable extraction algorithms' softgoal.

to be available by the framework.

3.5 Specifying the Business Process

The indication of business processes and their related goals, workflows, classes and entities are the last step in our proposed scheme for creating an ontological framework. We have selected the UML activity diagram for analyzing the business processes and their relationships. According to our scheme the partition of the activity diagram represent the business processes. Each partition may include different classes and can communicate with other business processes.

Figure 7 shows an activity diagram for the business process which is assigned to the 'Find an approximation of recall of tourism concepts for each algorithm' goal. The main actor of this business process is an expert who should 'select some concepts related to Canada tourism', 'Investigate whether they exist in the ontology' (we had supposed that the domain expert can work with the ontology editors), 'Calculate the recall' and then 'generalize the result and report it'.

Usually the domain expert, ontology engineers and also tasks and resources need some data exchange among themselves or saving data as well as some services like ontology editors. We have recognized three types of business processes that are usually included in most frameworks:

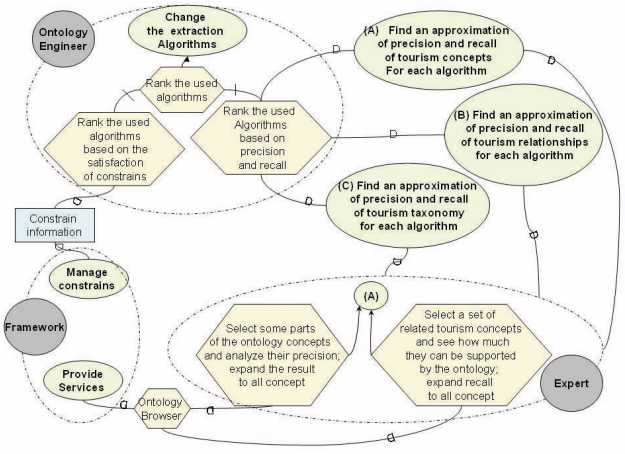

Figure 6. The SR model for the 'Change the extraction algorithms' goal.

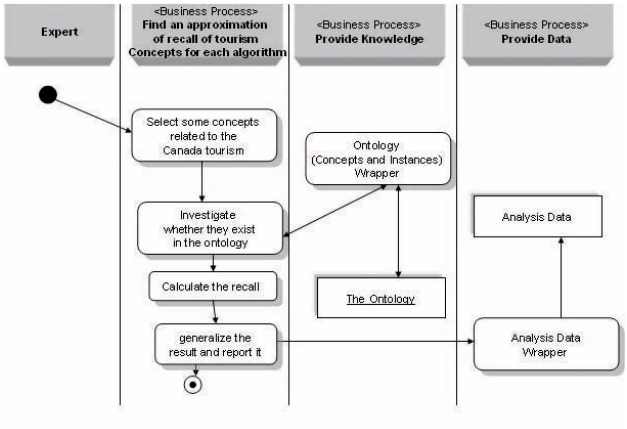

Figure 7. The activity diagram for the 'Find recall of concept extraction algorithms' business process

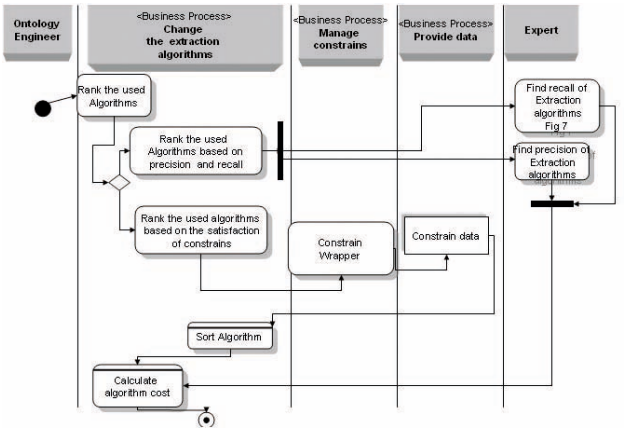

Figure 8. The activity diagram for the 'Change the extraction algorithms' business process

service provider that provides different services to different actors, data provider that manages data repository and knowledge provider that manages knowledge repository. In the tourism example, service provider provides an ontology editor for ontology engineers, data provider, manages a database which includes all of data tables such as 'analysis data' and 'constrain information' and knowledge provider provides the ontology which has been extracted in RDF [4] or OWL [3] syntax.

The activity diagram in Figure 8, specifies the business process associated with the 'Change the extraction algorithms' goal and its communication to other business processes. The 'rank used algorithms' ia a task that realize this goal. This task is decomposed to two subtasks: 'Rank the used algorithms based on precision and recall' and 'Rank the used algorithms based on the satisfaction of constrains'. As it is shown in Figure 8'Rank the used Algorithms based on precision and recall' requires that the expert provide the precision and recall of each algorithm (The activity diagram related to the providing recall of algorithms is shown in the Figure 7. The activity diagram for providing the precision is also similar).

4. Concluding Remarks

In this paper we have designed a scheme that helps developers to build an ontological framework according to the domain features and the customer necessities and preferences. By ontological framework, we mean those structures that provide some implemented components as well as workflows for extracting and managing domain ontologies. Our scheme, which utilizes the *i** conceptual modeling framework, has four steps. In the first step we introduce five important actors in an ontological framework and their high-level softgoals. In the second step we specify those tasks and goals that can realize the high-level goals of the first step. We have extracted these goals and tasks from the analysis of ontological framework features; however, a developer is free to select any of them or introduce new ones. In the third step we analyze the extracted goals in more detail. In this stage we introduced some extension to the *i** model elements like 'constrain' and 'feature'. The last step discusses some implementation issues. In this step, we extract the businesses process and model their relationships and tasks as control and entity classes using the UML activity diagram.

Our main attempt for future research is to suggest a comprehensive methodology for designing a high-level ontological framework for a specific domain. The developers can use this high-level, domain-specific framework and adopt it to their customer needs and preferences using the goal based scheme which is suggested in this paper.

References

[1] Daml ontology library, http://www.daml.org/ontologies/, last visited june 2007.

[2] http://www.corporate.canada.travel/en/ca/index.html, last visited june 2007.

[3] Owl web ontology language overview, http://www.w3.org/tr/owl-features/, last visited june 2007.

[4] Resource description framework (rdf), http://www.w3.org/rdf/, last visited june 2007.

[5] Zachman institute for framework advancement, http://www.zifa.com/, last visited june 2007.

[6] E. Bagheri and A. A. Ghorbani. Risk analysis in critical infrastructure systems based on the astrolabe methodology. In *CNSR '07: Proceedings of the Fifth Annual Conference on Communication Networks and Services Research*, pages 335–344, 2007.

[7] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos. Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3):203–236, 2004.

[8] L. Chung and K. Cooper. Defining agents in a cots-aware requirements engineering approach. In *Proceedings of the Seventh Australian Workshop on Requirements Engineering*, 2002.

[9] P. Cimiano and J. Vlker. Text2onto: A framework for ontology learning and data-driven change discovery. *Natural Language Processing and Information Systems*, pages 227–238, 2005.

[10] O. Corcho, M. Fernandez-Lopez, and A. Gomez-Perez. Methodologies, tools and languages for building ontologies: where is their meeting point? *Data Knowl. Eng.*, 46(1):41–64, 2003.

[11] F. Ensan and W. Du. Towards domain-centric ontology development and maintenance frameworks. In *Proceedings of the Nineteenth International Conference on Software Engineering & Knowledge Engineering (SEKE'2007)*, 2007.

[12] M. Fernandez-Lopez and A. Gomez-Perez. Overview and analysis of methodologies for building ontologies. *Knowl. Eng. Rev.*, 17(2):129–156, 2002.

[13] M. Fernandez-Lopez, A. G.-P. J. P. Sierra, and A. P. Sierra. Building a chemical ontology using methontology and the ontology design environment. *IEEE Intelligent Systems*, 14(1):37–46, 1999.

[14] A. Fuxman, P. Giorgini, M. Kolp, and J. Mylopoulos. Information systems as social structures. In *FOIS '01: Proceedings of the international conference on Formal Ontology in Information Systems*, pages 10–21, New York, NY, USA, 2001. ACM Press.

[15] M. Grüninger and M. Fox. Methodology for the design and evaluation of ontologies. In *IJCAI'95, Workshop on Basic Ontological Issues in Knowledge Sharing, April 13, 1995*, 1995.

[16] K. Kotis and A. Vouros. Human-centered ontology engineering: The hcome methodology. *Knowl. Inf. Syst.*, 10(1):109–131, 2006.

[17] A. Maedche and S. Staab. Ontology learning for the semantic web. *IEEE Intelligent Systems*, 16(2):72–79, 2001.

[18] J. Mylopoulos, L. Chung, and E. Yu. From object-oriented to goal-oriented requirements analysis. *Commun. ACM*, 42(1):31–37, 1999.

[19] R. Navigli and P. Velardi. Learning domain ontologies from document warehouses and dedicated web sites. *Comput. Linguist.*, 30(2):151–179, June 2004.

[20] V. F. A. Santander and J. Castro. Deriving use cases from organizational modeling. In *RE '02: Proceedings of the 10th Anniversary IEEE Joint International Conference on Requirements Engineering*, pages 32–42, Washington, DC, USA, 2002. IEEE Computer Society.

[21] M. Uschold and M. Grüninger. Ontologies: principles, methods, and applications. *Knowledge Engineering Review*, 11(2):93–155, 1996.

[22] E. S.-K. Yu. *Modelling strategic relationships for process reengineering*. PhD thesis, Toronto, Ont., Canada, 1996.

[23] J. A. Zachman. A framework for information systems architecture. *IBM Syst. J.*, 26(3):276–292, 1987.