Analysis and Design of Multi Agent Knowledge Development Process

Cheah Wai Shiang, Leon Sterling

Department of Computer Science and Software Engineering, University of Melbourne, Victoria, Australia
{w.cheah@pgrad., leonss@}unimelb.edu.au

Abstract

Within multi agent systems (MAS), knowledge plays an important role in agent communication, reasoning and supporting interoperability. It is often considered as an ontology which contains explicit domain knowledge to be used by agents. Although there are many ontology development or engineering methodologies, current efforts to incorporate knowledge into MAS are too focused on computational aspects or ad hoc. Working at the computational model is too low level, and many processes are left implicit to the developer. This paper focuses on engineering the agent knowledge development process. A set of activities is proposed to externalize processes involved in managing agent domain knowledge, preferring software engineering approaches to ad hoc processes. The activities are classified into analysis and design steps together forming a development model suite (e.g. user model, motivation model, task model and design model). With guidance, we have successfully developed agents' knowledge based on a real life application in finding a potential advisor for a graduate student. Finally, it enables the agent developer to use, reuse and maintain the agent knowledge.

1. Introduction

Nowadays, incorporating domain knowledge into multi-agent systems is ad hoc or too focused on computational aspects. Agents can have knowledge by creating personal webs of knowledge through text processing [15] (e.g. WordNet, term disambiguation); agents teaching others semantic concepts through supervised inductive learning [1]; populating the agent domain knowledge with mapping capability through various mapping algorithms like heuristic approach, usage of natural language processing (NLP) [6], usage of machine learning approach [9], dialogue based approach [7], [16] combination of dialog and text processing, [2], combination of dialog and NLP [21], combination of dialog and NLP. We argue that working at the computational model is too low level and many processes are implicit to the developer. Also, the development of agent-knowledge application can be time and effort consuming Selection of different algorithms to use is hard and there exist issues like having longer periods to produce training sets [7]; lacking domain specific terms under WordNet [6]; accuracy of classification and clustering techniques in semantic integration. Other mechanisms to incorporate domain knowledge in multi agent systems are ad hoc. For example, Ganzha proposed a pragmatic approach in describing hotel domain to be used by a multi-agent travel support system [19]; [22] leaves the work of ontology development to the software developer and concentrates on development of multi agent systems according to the usage of ontology within MAS. The ad hoc approach provides maximum flexibility; however, experience gained from the resulting application cannot be easily transferred [22].

This paper focuses on engineering the agent knowledge development process. A set of activities has been proposed to externalize processes involved in managing agent domain knowledge and avoiding ad hoc but more software engineering aspects in working on developing agent knowledge. The activities are classified into analysis and design steps. With complete guidance, we have successfully developed an agent knowledge base for a real life application of finding a potential advisor. Finally, it enables the agent developer to use, reuse and maintain the agent knowledge. Our hypothesis is that multi-agent knowledge consists of characteristics like derivation of agent knowledge from multiple sources; user centricity of agent knowledge; diversity of agent knowledge; agent knowledge is in two forms (e.g. what I understand, what I know) and agent knowledge is reusable. The outcome of our research is to facilitate the software developer or agent developer when dealing with developing knowledge for multi-agent systems. Towards this direction, finding the processes involved is our focus rather than developing new algorithms.

The paper is organized as follows. Section 2 consists of our early experiment in working on computational model towards autonomous knowledge creation by software agents. Issues have been highlighted and the outcome of this experiment contributed towards the main direction of this research. Section 3 discusses current research on knowledge related development mechanisms together with a description of our proposed solution that derived from feature extraction from existing development methods. Section 4 provides a general description of our

1530-0803/08 $25.00 © 2008 IEEE
DOI 10.1109/ASWEC.2008.74

proposed solution. Section 5 discusses design activities in more detail. This includes a discussion of processes involved together with a running example. Section 6 concludes the paper.

2. Preliminary Work

Before we discuss our proposed process oriented approach for developing knowledge for multi-agent systems, we elaborate our preliminary work on knowledge processing based on a computational focus. Our early work prototyped an "advisor finder system". The advisor finder mediates user queries to locate a relevant advisor autonomously. Hence, the agent must understand the advisor domain, and user related to advisor domain. Processes and algorithms are required to incorporate these domains into the agent system. The agent needs to develop the domain knowledge and populate the knowledge through reconciliation capability of a range of universities and academics. The aim of this preliminary work was to experience the processes involved in developing the agent knowledge through population of concepts and instances from semi structured data like web sites, and having the agent manage the diverse concepts autonomously. Everything was built from scratch and on the fly by agent and to agent. The task involved working on parsing component; syntactic similarity measures through approximate string matching and edit distance; knowledge extraction mechanisms (e.g. segmentation) from the web; diversity handling; and concurrent execution and coordination. The computation procedures in our working scenario are listed in Figure 1.

-Obtain concept segment
-Optimum finding by obtaining segment boundary based on heading
-Initiate concurrent execution
-From a segment, parsing instances from hypothesis (e.g. web) based on line coding
-Perform approximate string matching(instances from hypothesis, instances of concept)
-Obtain matched segment boundary
-Extract instances of the matched segment
-Perform syntactic similarity measurement through edit distance
-Loop to other hypothesis (e.g. other community member pages)
-At the ends of lists, perform noise filtering, delete noisy elements
-By now, the agents has confidence on the concept segment and can use the segment name to perform the unmatched hypothesis
-Embed new instances together with segments

Figure 1: Computation Model for developing agent knowledge

Working at the computational level is challenging. Besides concerns about the diversity of the web (e.g. syntactic and semantic level), the accuracy of the populated domain knowledge is a big concern. One of the factors influencing accuracy is the threshold [7] used within the matching or mapping algorithm. Lower threshold will cause a huge number of populated instances either relevant or irrelevant. Higher threshold will produce higher accuracy but lose other relevant knowledge elements. Apart from that, building the relationship among the concept and sub-concept autonomously is not trivial and even a developer can justify the relationship manually. Meanwhile, from the computational model, we have identified interesting findings to model our working example. The outcome from the model has inspired us to continue our effort towards "process oriented or software oriented" agent knowledge development which is the main discussion in this paper.

3. Background

In this section, we describe our background study based on our hypothesis in working on multi-agent knowledge development as described in the introduction. Related research to ours falls under the area of ontology engineering (OE). From the study, we identified two trends in OE, namely engineering conceptualization [24] and engineering development process [12]. Our work is towards the latter trend. Although there exist many diverse methodologies, there is a lack of applicability within application development compared to CommonKADs. As a result, we believe that this is why agent-ontology development still is ad hoc. Also, the approach of incorporating knowledge into multi-agent systems or the agent knowledge development process is still unclear and non-understood. CommonKads [13] has incorporated knowledge into agent systems but the level of agency is undefined. Here, an agent is defined as human or hardware and the focus of the knowledge is problem solving or inference rule with little concern for domain knowledge. Also, the influence on agency towards the knowledge development does not have proper consideration. MAS-CommonKADS [3] incorporated problem solving knowledge into analysis and design of multi-agent systems or produced a methodology for multi agent system development. In this case, CommonKad model suite has been applied to analyze software agents with further extension to protocol engineering and coordination mechanisms. It is not our focus to create another new methodology for software agent. Dileo [8] worked on integrating an ontology into the Multi agent Software Engineering methodology (MaSE). In his work, concepts were extracted from requirement analysis like use cases and sequence diagrams to form a system ontology. The steps are derived from IDEF5 and Methontology. They define purpose and scope of the ontology; collect data; construct an initial ontology; refine and validate the ontology; and using the ontology in MAS. Apart from forming analysis and design activities in agent knowledge development, the significant difference between our work and Dileo's is in integration steps. Finally, our work fills the missing ontology definition activities within MOMBAS methodology [22].

Our proposed agent knowledge development process is represented through the usage of features extracted from the methodologies [11,13,14,17]. To reduce the complexity of knowledge representation, we adopted the lightweight ontology structure based on work from our Agentlab [18]. Instead of working knowledge processing autonomously, the agent developer has opportunity to work on knowledge structure and elements through documentation structure from CommonKads, UML and tabular or profile representation like work from Methontology. Meanwhile, the agent will work within the organization or community. In dealing with community, people with the same interest or knowledge will group together to easily reach consensus during communication. People will obtain their knowledge through experience [20] and we further extend that people will explicitly indicate the knowledge into a particular representation (e.g. semi structural data like web). In this case, working at distributed knowledge and community knowledge is important here. Since this work is looking from software engineering aspect, the analysis steps will be represented through the documentation format from CommonKads and UML.

4. Engineering Multi Agent Knowledge Development

We have divided the agent knowledge development process into analysis and design activities. Analysis activities handle the agent knowledge at a high level of abstraction. It is interesting to show that by borrowing the agent concepts during the analysis, it enables explicit indication of agent knowledge development in a structured and clear manner. The developer or user will analyze the domain knowledge through model sets. The proposed model set corresponding to the analysis activities are user model, motivation model, task model and ontological model. The proposed analysis activities for multi-agent adviser finder consist of activities like knowledge source analysis, knowledge item analysis and knowledge structure analysis. Meanwhile, the design activities will handle the agent knowledge at the applicable level. From our work, the design activities will specify the components required in developing agent knowledge through developing design models. It involves designing specific mechanisms to organize the agent knowledge, having profile based, structuring knowledge deployment to agent like interaction design, with the aim that thee agent knowledge is ready to use by the software agent. Furthermore, it concerns the subsystems involved, interconnected through data, control and other dependencies.

The generalization of the activities is based on our proposed agent knowledge engineering principle. Given a case study, identify the requirement for the application based on the knowledge consideration. Extract the knowledge characteristics of the application(s) and these will turn into agent knowledge characteristics. Based on the knowledge characteristics, invent a model and design process that will facilitate the user and developer in agent knowledge development or enable such knowledge to be handled (reuse existing model or design a new one). Structure the model based on the knowledge development lifecycle (e.g. knowledge identification, knowledge generation, knowledge evolution and knowledge deployment). In order to provide a high level abstraction of the agent knowledge development, we have associated the agent concepts with the analyzing of agent knowledge. The association procedure is described below. Influenced by the ROADMAP methodology, we first model the agent organization, followed by interaction and service. First, the working procedure involves identifying the knowledge source also known as MAS organization knowledge or multi agent domain knowledge through a user model. The user model will model the agent involvement within the organization. Then, we identify the knowledge items based on a concept that user interaction is triggering from motivation towards a task execution. In this case, user motivation will be modelled within a motivation model to execute a particular task that is modeled within the task model. Together, these form an organization knowledge structure for agent systems. Meanwhile, an interaction protocol is used to model the task ontology. At the same time, the user or developer constructs design components required to facilitate the development of knowledge for MAS.

In the rest of this section, we briefly describe the analysis activities involved and proceed into detailed description of design activities in the following section. A more comprehensive description of the analysis activities can be found in [5].

4.1 Analysis Activities

The knowledge source analysis indicates the knowledge that will occur within the MAS organization. It identifies actor(s) involved to produce a knowledge contributor model or user model. The model will capture information like knowledge contributor, scope of knowledge extraction, actor formation and actor boundary. The knowledge item analysis involves identifying knowledge items like concepts and instances required within the agent knowledge. This can be done through analysis of motivation items from the motivation model and analysis of the ingredients from the task model. The knowledge structure analysis involves conceptual layout and knowledge elements. The conceptual layout and elements are derived through a set of activities like identifying level of granularity, populating the conceptual elements, instance analysis, diversity analysis (with mapping requirements like locating the algorithm used during mapping; identifying the concepts used during mapping; mapping process; enriching the mapped concepts) and finally refining and verifying the knowledge elements.

5. Design Activity

In this section, we describe the design activity in detail based on a running example of a multi-agent adviser finder system. It is an agent mediator system that works on the following scenario: Students have Government scholarships available to study for a Ph.D overseas if they are able to find an adviser at a reputable university. To find an adviser, a substantial amount of knowledge is needed which includes "advisor domain" like research areas, research experience, professional activities, etc. These are usually described differently at different institutions.

The design activity starts with having software components to locate the actors and obtain actors' knowledge; manage knowledge repository arrangement; organize the concept layout and elements; serving reconciliation outcome and managing knowledge usage within multi agent systems.

5.1. Design for obtaining organization knowledge

In this section, we describe the design aspect to facilitate process in obtaining the organization knowledge. The steps involved are listed below.

5.1.1 Addressing or directory design

The aim of directory design is to provide a platform for locating the knowledge sources for further processing. Once the actors have been modeled, the locations of the knowledge sources or actors' explicit knowledge need to be traceable. This can be done through designing a directory registration like works in UDDI, reference ontology, directory facilitator. In this project, a simple addressing object has been created to support the directory design. The addressing object will store a list of addresses, actor name and name of the community. A more advanced directory registration can be derived from previous work .

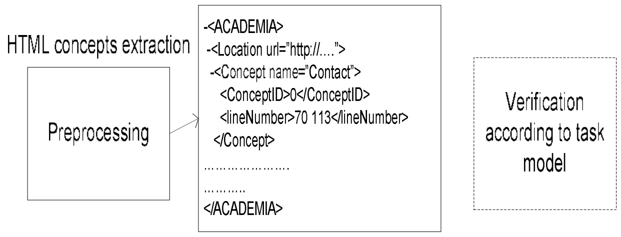
5.1.2 Knowledge extraction

As mentioned before, we assume that semi-structured data like on the Web will contribute towards the detailed allocation of actors' knowledge. In this case, designing a software component for semi-structured data extraction is required. Although many mechanisms or tools have been proposed for knowledge extraction from the web, this is not our direction in working on increasing the recall and precision through advanced algorithms. Based on our early work in autonomous agent knowledge, it seems that the web is too diverse and it is a challenge to have a single algorithm. Also, it is hard to identify and extract concepts, instances, and sub-concepts through the current information retrieval tools. As a result, we are working on a simple semi-automated tool to extract the data in order to engineer the process in knowledge extraction and understand the process involved in building the agent knowledge from scratch. The input to the tool is explicit knowledge (e.g. information on web or defined as knowledge source) and the output is actors knowledge represented in XML. To overcome the diversity of the web, we have proposed a step-by-step and continuous verification of the extracted knowledge items with supported tool. The verification has been done based on the outcome from the analysis activities [5]. The description of the steps involved is given below.

Preprocessing. Preprocessing deals with conversion of actor explicit knowledge. Without preprocessing, HTML is just like a collection of data with different data formation. We assume that the concept will be represented with a heading (e.g. H1, H2 and etc.) and the instances are underneath each concept. Since it is subjective and difficult to explore the relationship from semi-structured data, we focus on concept-instance relationship. An HTML parser[1] and XML beans[2] are used during the preprocessing. HTML parser consists of API that can be used to interpret HTML tags and retrieve the content within a particular tag. Given a URL, the HTML parser has capability to parse HTML tag, HTML link, HTML text and HTML remark. Each tag can be considered as a tree structure (e.g. each tree element consists of a start and end tag) and iteration is used to retrieve a certain tag required by the developer. However, the tree structure is getting complex with the current HTML designs. To simplify the process from traversing from tree elements to the others, we have utilized tag numbers in concept extraction. The concept is surrounding with a start tag number and an end tag number. The start tag number is the number given when the concept name has been identified. Meanwhile, the end tag number is the number given when the next concept name has been identified. This start and end numbers form a block we call the boundary of a concept. For example, given explicit knowledge of the academic, the concept of *contact* falls under a numbering of 70-113

[1]http://htmlparser.sourceforge.net
[2]http://xmlbeans.apache.org

(forming a block). In addition, the item within the block is interpreted as instances for the given concept.



```
-<ACADEMIA>
 -<Location url="http://....">
  -<Concept name="Contact">
    <ConceptID>0</ConceptID>
    <lineNumber>70 113</lineNumber>
  </Concept>
…………………
………
</ACADEMIA>
```

Figure 2: Preprocessing from HTML

Figure 2 shows the output after preprocessing. The output is generated by using XML beans and captures the concepts structure for an academic. XML is used here as it is platform independent and supports semantic description for data. Besides, the structure is well-form and well-defined. From Figure 2, concepts are extracted from explicit knowledge of Rao, consisting of concepts like contact, current research interests, students and current activity. Verification is taken place during preprocessing to remove unwanted concepst as well as ambiguous concepts. Some of the unwanted concept like +61 3 8344 1325, physical location, email, hobby, photo, Out and about, getting the job done..., Chair of software innovation and engineering, links, admin and so on. The verification is based on the outcome from the task model. Apart from that, verification also involves a tag number check to prevent unwanted instances falling into particular concepts during concept instantiation .

Concept extraction is a preliminary stage to form an individual knowledge repository after the preprocessing. The context knowledge derived from the task will become a guideline to extract the relevant concepts in this process. Although the task ingredient is far from complete, having interaction with extracted knowledge can enrich the knowledge of a software developer. For example, finding a potential advisor, advisor here is a representative from an institution and works within the education context. The education context for an academic is derived from concepts such as publication, supervision, teaching, and administration. In this section, we do not deal with raw data like HTML but structured knowledge (e.g. XML forms) derived from preprocessing. A concept extraction component has been developed for this stage. The concept extraction component relies on XML Beans for parsing the XML file. As mentioned before, the task ingredient or education context has become a guideline for the concept extraction. As a result, non-verification is required in this stage due to one-to-one mapping between the contextual knowledge and the extracted concepts. For the moment, the software developer is required to manually input the concepts for extraction through our concept extraction component.

Concept instantiation component looks for instances that fall underneath the concepts. It facilitates the instance analysis as described previously. The input consists of concepts that had been extracted in the previous stage and dedicated outcomes derived from the preprocessing. People may be curious why we don't extract everything during the preprocessing. One of the reasons is to reduce the unnecessary filtering and time spent during the preprocessing. The other reason is we need more precise outcome through checked tag number. The processes involve searching through the corresponding tag number within a particular concept. Once obtained, the component will extract the instances by using the HTML parser. The instance verification is required to filter unwanted extraction like space, duplicate instances and unorganized instances.

Global integration. The function of global integration is to position the concepts within a standard ontology. The global integration is another enrichment method to agent knowledge through inheriting or referencing the knowledge from domain expert like ACM computer science classification, mathematical ontology, biology and so on. It relates the extracted concepts into a more formalized stage in which the standard ontology will act as global reference and provide an annotation service for concepts. The information from the global ontology is derived from ingredient analysis from the task model [5]. For example, a concept of research interest may consist of information sources from ACM classification list, conference topic of interest and so on. Here, we have tried to integrate the ACM classification list into our working example for the concept of research interest. The classification system consists of 122k, more than 200 classes and has at most four level of granularity forming a single classification XML file. Due to the complexity in traversing from concept to sub-concept, we have constructed our local global ontology that is suitable for our context with conditions of 5 main concepts software, data, mathematic computing, information system and computing methodology, reduced sub-concepts and 3 levels of granularity. The challenge for global integration is to have an effective searching algorithm as well as the diversity of concept representation by individual. In this project, we have utilized SQL query to ease of searching for the concepts. However, the current ontology has failed to annotate the application at hand. This has risen by [travel ontology] that we cannot find any ontology that is suitable for what we want to do. Although it fails to capture the overall knowledge level for an application, it has added some value to agent knowledge.

5.2. Knowledge storage design

The knowledge storage acts as repository to provide storage at the knowledge level. Logically the knowledge storage design is shown in Figure 3. The physical view of the storage design is shown in Figure 4. In the knowledge storage design, two important concepts have been defined. There are “agent understands” and “agent knows”. These concepts have been implicitly specified within current mapping tools. Prompt, Chimara, S-Match, Sambo focus on matching at the concept level; GLUE, iMapper, Anemone focus on matching at the instances level as well as the concept level. An agent must understand the knowledge structure and have the capability to interpret the knowledge structure. The conceptual space is the agent knowledge model structure and it defines “what an agent can understand”. It also defines agent knowledge model formation in which forming agent concepts, sub-concepts, attributes will carry on. The individual knowledge repository consists of knowledge from a particular individual, represented in XML file. As a result, we will have groups of XML files among the individuals. Since the individual is located within a community, the group of individuals that have the same location will form a larger repository. In this case, each community will be represented as a folder. Finally, all the community having a same interest will form a cross organization viewpoint.

The agent knowledge repository is a process to form “what an agent knows” through knowledge structure instantiation. The process involves forming the instances for each of the concepts within the knowledge structure. Each concept is hosted under a dedicated folder together with the file name according to given concept name. For example, under the agent knowledge repository for a concept named “project”, it consists of 10 Xml files. In this case, there exist 10 individuals that can provide the knowledge required by the agent.

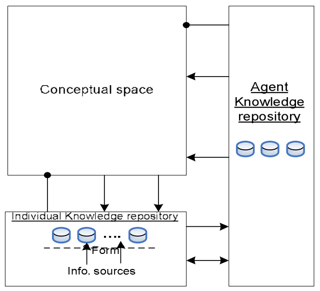


Figure 3: logical view design

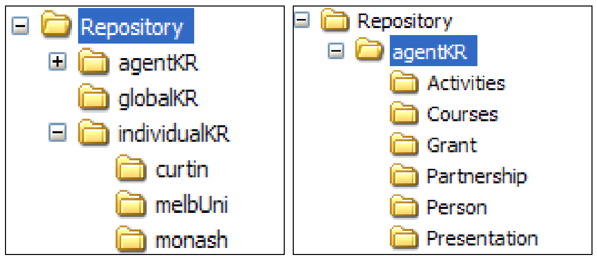
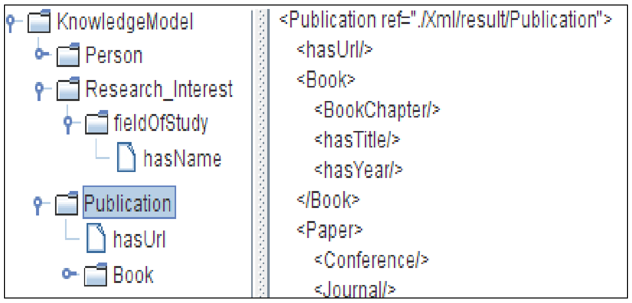


Figure 4: Physical view design for knowledge storage. agentKR- agent knowledge repository; individualKR-actor knowledge repository; globalKR-global knowledge repository

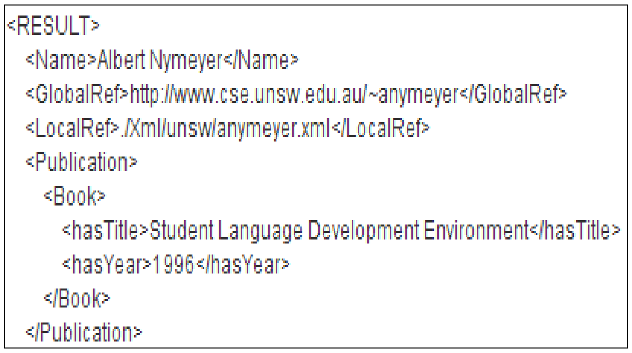
5.3. Knowledge structure or conceptual design

The knowledge structure design or conceptual design focusses on designing the layout for agent knowledge. We believe that the knowledge layout will grow and evolve from time to time due to the changes within the organization or community within the organization. The current ontology layout is too complex and difficult to traverse by agents [18]. As a result, the structure must be simple, lightweight and easy to traverse by agents. We propose to use a simple taxonomy as knowledge layout. Initially, the agent knowledge consists of concepts and instances. Then it will continue to be refined until it forms a complete structure of agent knowledge with expansion capability. Figure 5 shows the agent knowledge structure and Figure 6 shows the agent knowledge structure with elements associated with the concept of “publication” as indicated in Figure 5.



```
<Publication ref="./Xml/result/Publication">
  <hasUrl/>
  <Book>
    <BookChapter/>
    <hasTitle/>
    <hasYear/>
  </Book>
  <Paper>
    <Conference/>
    <Journal/>
```

Figure 5: Agent Knowledge Structure



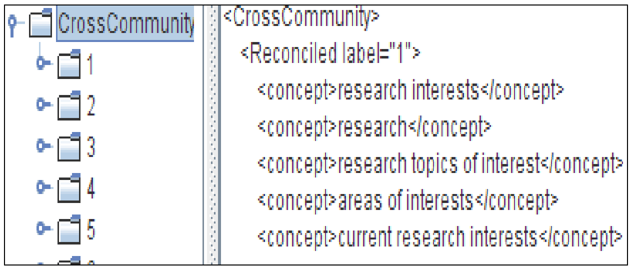
```
<RESULT>
  <Name>Albert Nymeyer</Name>
  <GlobalRef>http://www.cse.unsw.edu.au/~anymeyer</GlobalRef>
  <LocalRef>./Xml/unsw/anymeyer.xml</LocalRef>
  <Publication>
    <Book>
      <hasTitle>Student Language Development Environment</hasTitle>
      <hasYear>1996</hasYear>
    </Book>
  </Publication>
```

Figure 6: Agent Knowledge Structure and Element

5.4 Knowledge reconciliation design- Populating the Agent Knowledge

The knowledge reconciliation design works on the basic idea surrounding maintenance and reusability of the mapped concepts. From the previous study, we identified that storage and maintenance of knowledge is needed to enable the usability and reusability of agent knowledge. We believe that this is not an exception during reconciliation. Methontology [11] is the only ontology methodology working in this direction. Here, a set of tabular forms or tables will store the concepts, instances, properties, axiom during the ontology development process. The description provided for the knowledge elements is clearly defined and presented in a well form.

We proposed a set of profiles to locate the reconciliation elements. The usage of profile is taking the advantage of structural representation (e.g. XML) and is also platform independent. During the diversity analysis, the developer will initiate a concept and receive input from individual knowledge repository to perform matching between the knowledge elements within the individual knowledge repository and the initial concept. We use the reconciled profile to store the initial concept. The layout of the reconciled profile is shown in Figure 7. The reconciled profile consists of a classification of concepts with similar meaning but with different representation. For example, from our working environment (e.g. finding a potential advisor from 60 data sets), we have identified that “service”, “previous work”, “research papers”, “recent publication”, “current activity” have similar instances. Although the concept representation is different, the instances are the same. This indicates how diverse people are in representing their knowledge. Another example like “lecturing”, “current teaching”, “course”, “teaching in year”, “subject”, “classes”, “lectures”.



```
<CrossCommunity>
  <Reconciled label="1">
    <concept>research interests</concept>
    <concept>research</concept>
    <concept>research topics of interest</concept>
    <concept>areas of interests</concept>
    <concept>current research interests</concept>
```

Figure 7: ReconciledProfile

Having worked on the reconciled profile, this involves processes to record the concepts that contributed by the individual actor for purposes of tracking. We have proposed two profiles towards this activity. They are K-distribution profile and C-profile. For the moment, the C-profile is based on tabular form. However, it is easily transform into XML structure as listed below. The C-profile is compulsory and K-distribution profile is optional. Depending on the situation, sometimes only a C-profile is needes, other times both may be needed.

The C-profile or contributor profile is a profile that records individuals that contribute towards a particular concept within a particular community. For example, a concept of research interests is contributed by individuals within a community of MelbUni like Adrian, Alistair, Harald, Udaya[3]. The C-profile is the smallest unit in tracking concept locality. It provides a reference to the individual knowledge repository during execution. Keep in mind that the C-profile is not simply a directory but a place for individuals to position itself to consensus. Figure 8 (right) shows the C-profile for individuals under MelbUni. Each community must have their individual C-profile. We have constructed our C-profile by using Microsoft Access for fast prototyping. The C-profile consists of three sections, the member list, description list and concept distribution list. The member list (in the middle) has highlighted the individual that participated within the community. It consists of reference points to the corresponded knowledge repository. The description list indicates the number of unique concepts that represent under a particular context, here we defined as education context. For example, C1 or category 1 is described as the consensus reaching subjective to concept-interest representation among the academics. Finally, the concept distribution list indicates how the individual said they know for a particular concept within a particular consensus category.

The K-distribution profile also known as knowledge distribution profile indicates the community that contributed towards a particular concept or knowledge point that can be looked for, before further processing. Figure 8 (left) shows the K-distribution profile. The number of concepts is closely related to concept reconciled profile. The K-distribution profile consists of concept, community name and concept category (refer to C-profile). From Figure 8 (left), we can interpret that the concept of “*research interests*” reconciled group (e.g. from concept reconciled profile) known among individual from MelbUni, UTS, Curtin, Monash, RMIT and UNSW. Steps towards creating a K-distribution profile are:

1. Having concept reconciled profile and C-profile as input
2. Obtain reconciled concepts (group of concept under a particular reconciliation) from reconciled profile and perform mapping to C-profile.
    a. Mapping can be performed from description list or,

[3] Note first names of Computer Science academics from the University of Melbourne are used here.

   b. From concept distribution list
3. Once found, locate the concept category and fill it into K-distribution profile. If the concept does not exist, leave the column empty.

```
<?xml version="1.0" encoding="UTF-8"?>
<CrossCommunity>
  <Concept label="1">
   <melbUni>C1</melbUni>
   <uts>C1</uts>
   <curtin>C2<curtin>
   <monash>C2</monash>
   <rmit>C2</rmit>
   <unsw>C1</unsw>
  </Concept>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<ConceptProfile>
<consensus id=C1>
 <Description>
 The consensus reaching subjective to interests
 representation among the academia
 </Description>
 <element name="research interests">
    <instance>adrian.xml</instance>
    <instance>alistair.xml</instance>
 </element>
 <element name="current research interests">
    <instance>rao.xml</instance>
    <instance>egeman.xml</instance>
 </element>
</consensus>
</ConceptProfile>
```

Figure 8: (left) Example of K-distribution profile and (right) Concept Profile in XML form

5.5. Deployment Design

Work has been done by introducing MOMBAS methodology in incorporating ontology into agent oriented software engineering. Although MOMBAS [22] has proposed a set of activities to integrate the ontology into agent designs like internal design, interaction design, organization design, the work is fall within our proposed deployment section. No agent knowledge development process has been introduced or incorporated into the work in MOMBAS. Here, we will describe how the outcome from the analysis and design activities will contribute towards the incorporation of agent knowledge into agent systems.

As described in the user model [5], the MAS organization consists of two types of actors. They are actors who act as students and actors who act as academics or potential advisors. Meanwhile, we also modeled the interaction with policy like student has full access to the concept structure, and partial right to access the instances or knowledge element. By default, the actors (e.g. academics) will share their knowledge freely among each others. In order to cater the huge number of actors involved (e.g. academics), we have adopted an agent mediator architecture to work on the application. Two proposed scenarios are shown in Figure 12 below. The first scenario involves multi agent interaction, while the second scenario involves human agent interaction as shown in Figure 9. The second scenario is human and agent interaction.

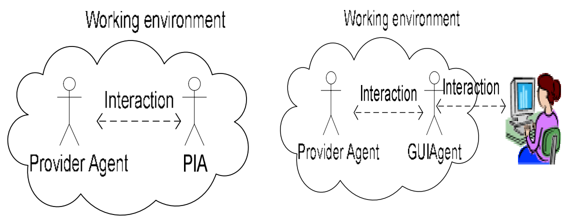


Figure 9: Multi agent system, Human and agent interact through knowledge model (right).

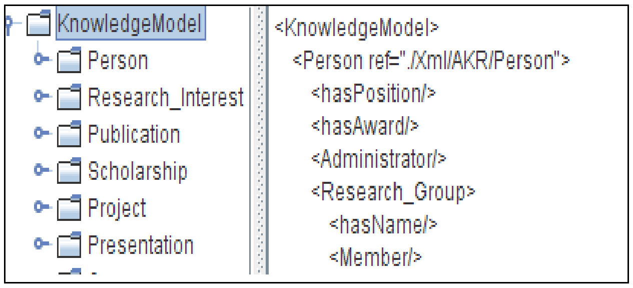
In multi agent interaction, a personal information agent (PIA) has been dedicated to perform the task of finding a potential advisor. The PIA uses motivation knowledge derived through motivation analysis. Once activated, by default the PIA would have complete knowledge for performing the task required when it entered a working environment. Logically, the PIA will inherit the knowledge model in a particular working environment. However, the knowledge it has is just a knowledge shell without knowing instances. The inheritance mode depends on what role an agent plays in the environment as well as the policy restriction that have been modeled in the user model. From client and service aspect, a client agent will inherit with knowledge without instance. Meanwhile, a service provider will inherit with complete knowledge and instances. This will introduce reusability of the knowledge model. The overall execution of the agents are traversing the agent knowledge model, obtaining annotation to dedicated agent knowledge repository, comparing the returned knowledge items, ranking it and presenting to user or providing response to PIA. In human-agent interaction, the operation is still the same but more dynamic and complex agent development (e.g. degree of autonomy) is introduced. In the following section, we will describe the internal architecture of the mediator based on the Figure 9 (left) with student agent as PIA and provider agent as mediator. This involves a process in loading separate knowledge repository representation based on the described scenario. Table 1 shows the deployment of agent knowledge within our working example.

Provider Agent	Personal Intelligent Agent
Load policy profile Load agent knowledge structure ---connected reconciled Profile ---connected K-distribution Profile ---connected concept Profile ---connected individual knowledge repository ---**connected agent knowledge repository	Load policy profile Load motivation script ---connected agent knowledge structure

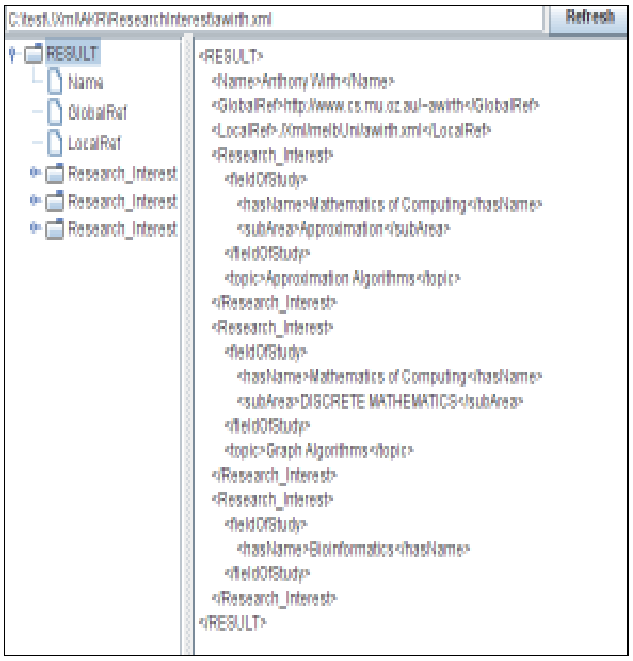
Table 1: Deployment Design

During the agents' interaction, a received request (e.g. motivation item or concept) will pass through several checks from reconciled profile, K-distribution profile, concept profile until further retrieving the relevant instances that occur within a particular individual knowledge repository. This is time consuming and computationally intensive. As a result, annotated agent knowledge structure to individual knowledge repository is ineffective. We need more reusability mechanisms and effective solution for handling agent request. Preprocessing has been done to annotate a particular concept with dedicated instances within the individual knowledge repository as listed in the following steps.
Steps 1 to 8 represent execution involved in preparing the agent knowledge repository or preprocessing period. Step 9 onward involves process for constructing agent knowledge repository.
1)Give a particular motivation item as input
2)Check concept reconciliation and baseline profiles
3) Obtain reference on reconciled item
4) Check K-distribution profile (if exists, optional)
  4a) Obtain concept category path
5) Check on C-profile
6) Obtain member item
7) Traverse KM storage like community repository and individual knowledge repository
7a) Obtain concept instance based on reconciled reference
8) Write the outcome into XML file, named with identified individual
9) Processing to form agent knowledge repository
9a) Give agent knowledge model and motivation item as input 9b) Traverse the agent knowledge model to find the relevant concept 9c) Obtain the concept block (consisting of concept and sub-concepts) 9d) Verify and embedded the block structure into the XML file from Step 8. 9e) Finalize agent knowledge repository

From now on, instead of having references to K-distribution profile, concept profile and individual knowledge repository, loading the agent knowledge structure will only require to annotate reconciled profile and agent knowledge repository. Figure 10 below represented the agent knowledge for provider agent. The mediator agent uses the knowledge in further processing.



```
<KnowledgeModel>
  <Person ref="./Xml/AKR/Person">
    <hasPosition/>
    <hasAward/>
    <Administrator/>
    <Research_Group>
      <hasName/>
      <Member/>
```



```
<RESULT>
  <Name>Anthony Wirth</Name>
  <GlobalRef>http://www.cs.mu.oz.au/~awirth</GlobalRef>
  <LocalRef>./Xml/melbUni/awirth.xml</LocalRef>
  <Research_Interest>
    <fieldOfStudy>
      <hasName>Mathematics of Computing</hasName>
      <subArea>Approximation</subArea>
    </fieldOfStudy>
    <topic>Approximation Algorithms</topic>
  </Research_Interest>
  <Research_Interest>
    <fieldOfStudy>
      <hasName>Mathematics of Computing</hasName>
      <subArea>DISCRETE MATHEMATICS</subArea>
    </fieldOfStudy>
    <topic>Graph Algorithms</topic>
  </Research_Interest>
  <Research_Interest>
    <fieldOfStudy>
      <hasName>Bioinformatics</hasName>
    </fieldOfStudy>
  </Research_Interest>
</RESULT>
```

Figure 10: Agent knowledge structure and annotation to agent knowledge element – Provider Agent

Putting it all together, the design model for handling the agent mediator knowledge is shown in Figure 11.

6. Conclusions

The main objective of this research is working on agent knowledge development mechanisms. Since knowledge plays an important component in agent systems, we believe that there should be an easy way to incorporate the knowledge development into multi agent systems. The mechanism should be clear, explicit and reusable by others. It can turn into guidance, pattern and working on high level of abstraction. Two working mechanisms have been proposed and described. This can range from having a computational model in dealing with autonomous agent knowledge development (e.g. of AI approach) to software engineering aspect through working on "process-oriented" approach. In future, we would like to develop a more structured way to develop agent knowledge. Furthermore, the agent knowledge development process will fertilize from activities like knowledge discovery, knowledge generation, knowledge evolution and knowledge deployment.

Acknowledgements

Thanks for useful discussions from members of the Intelligent Agent Lab at the University of Melbourne. The first author was supported by a Malaysian Government scholarship. The research was partially supported by ARC Linkage Grant LP0454027.

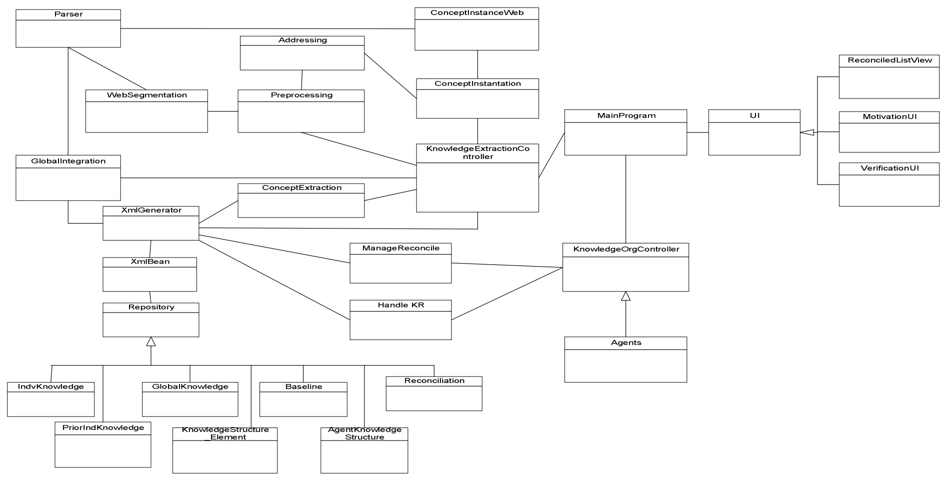


Figure 11: Design Model of Agent Knowledge Development Process for Mediator Agent

7. References

[1] William A. B.(2004). Learning to share meaning in a multi agent system, Journal of Autonomous Agents and Multi Agent System, **8**(4), pp. 165-193, 2004.

[2] Bailin S. C., Truszkowski W. (2003). Ontology Negotiation: How Agents Can really get to know each other, Proc. First International Workshop on Radical Agent Concepts, McLean, VA, USA, pp. 320-324, 2003.

[3] Henderson-Sellers B., Giorgini P.(2005). Agent oriented methodologies, In Agent-Oriented Methodologies, (eeds. P. Giorgini and B, Henderson-Sellers), pp. 46-79 ,2005.

[4] Henderson-Sellers B., Tran Q. N., Debenham J.(2005). An etymological and metadamodel based evaluation of the terms "goal and task on agent oriented methodologies," Journal of Object Technology, **4**(2), pp. 131-150, 2005.

[5] Wai-Shiang C., Sterling L. (2007). A Dedicated Model in developing agent system with reconciliation capability, Proc. 9th International Conf. of Information, Integration and Web based Application and Service Jakarta, pp. 33-45, 2007.

[6] Fossati D., Ghidoni G., Eugenio B. D., Cruz I., Xiao H., Subba R., (2006). The Problem of ontology alignment on the web: a first report Proc, Workshop on Web as Corpus, Proc. 11th Conference of the European Association of Computational Linguistics. Trento, Italy.

[7] Diggelen, J. V. (2006). Achieving Semantic Interoperability in Multi-agent Systems-A Dialogue-based Approach. PhD thesis, University Utrecht.

[8] Dileo J., Jacobs T. DeLoach S.(2002). Integrating ontologies into Multiagent Systems Engineering, Proc.Fourth International Conf. on Agent Oriented Information Systems, **59**.

[9] Doan A., Madhavan J., Dhamankar R., Domingos P., Halevy A., (2000). Learning to Match ontologies on the semantic web, VLDB, **12**(4), pp. 303-319, 2000.

[10] Djuric D., Devedzic V., Gasevic D.(2007). Adopting Software Engineering Trends in AI, IEEE Intelligent Systems, **22**(1), pp. 59-66, 2007.

[11] Lopez M. F., Gomez-Perez A., Sierra J.P.(1999). Building a Chemical Ontology Using Methontology and the ontology design environment, IEEE Intelligent Systems, **14**(1), pp. 37-46, 1999.

[12] López, M. F. (1999). Overview of the methodologies for building ontologies, Proc.Workshop on ontologies and problem solving methods (KRR5), IJCAI-95 .

[13] Schreiber G., Wielinga B., Hoog R., Akkermans H., Velde W. V.(1994). CommonKADS:A Comprehensive Methodology for KBS Development, IEEE Expert:Intelligent Systems and Their Applications, **9**(6), pp. 28-37, 1994.

[14] Kotis K., Vouros G. A.(2006). Human Centered Ontology Engineering: the HCOME methodology, International Journal of Knowledge and Information Systems, **10**(1)**,** pp. 109-131, 2006.

[15] Kotis K. (2005). Using Simple ontologies to build personal webs of knowledge, Proc. 25th International Conference of the British Computer Society. Cambridge, UK.

[16] Laera L., Tamma L. V., Euzenat J., Bench-Capon T., Payne T. (2006). Arguing over ontology alignments, Proc. International Workshop on Ontology Matching, Athens, GA.

[17] Duen-Ren L., I-Chin W., Kuen-Shieh Y. (2005). Task based K-support system: disseminating and sharing task-relevant knowledge, Journal of expert systems with applications, pp. 408-423, 2005.

[18] Hristozova M., Sterling L.. (2002). An eXtreme method for developing lightweight ontologies, Proc.,Workshop on ontologies in agent systems, AAMAS 2002.

[19] Ganzha M., Gawinecki M., Paprzycki M., Gasiorowski R., Pisarek S., Hyska W. (2006). Utilizing semantic web and software agents in a travel support system, In Semantic Web Technologies and eBusiness: Idea Publishing Group, pp. 325-359, 2006.

[20] Laclavik M., Babik M., Balogh Z., Hluchy L. (2006). AgentOWL:Semantic Knowledge Model and Agent Architecture, Computers and Artificial Intelligence, **25(5)**, pp.419-437, 2006.

[21] Orgun B., Dras M., Cassidy S., Nayak A.(2005). DASMAS-Dialogue based Automation of Semantic interoperability in Multi-Agent Systems. Australian Ontology Workshop, **58,** pp. 75-82, 2005.

[22] Tran Q. N., Low G. C. (2006). A methodological framework for ontology centric oriented software engineering, Journal of Computer Systems, Science and Engineering , **21**(2).