Supporting Development of Software Agents By Integrating Heterogeneous Repositories Based on Ontologies

Noriaki Izumi
Dept. Computer Science
Shizuoka University
3–5–1 Johoku Hamamatsu
Shizuoka 432–8011 Japan
izumi@cs.inf.shizuoka.ac.jp

Takahira Yamaguchi
Dept. Computer Science
Shizuoka University
3–5–1 Johoku Hamamatsu
Shizuoka 432–8011 Japan
yamaguti@cs.inf.shizuoka.ac.jp

ABSTRACT

This paper proposes an integrated support methodology for constructing software agents by using heterogeneous repositories. In order to model agents' behavior and their inference process and to implement them as software applications, heterogeneous repositories in different granularities are integrated based on ontologies. By devising a framework, which picks the main concepts of repositories up and make correspondence among them, our framework achieves the unified reuse of existing repositories of modeling libraries and software components. We have implemented the prototype system by JAVA and confirmed that it supports us in various phases of agent application development including agent-model construction, definition of agents' inference process and an implementation of agent software applications.

Keywords

Ontologies, Common KADS, Process Handbook, Agent development

1. INTRODUCTION

Due to the recent development of the Internet computing environment, it becomes very important to achieve the rapid adaptation of the agent's task by adopting new computing environments and employing new technologies. Furthermore, according to the recent trends of e-business, the roles of ontologies become very important as the key concepts not only for the exchange information among people and computers, but also for employing a variety of heterogeneous reusable components such as inference models, agent resources, software libraries, and so on.

Because of the above context, a lot of research and development projects, which construct repositories of various concepts and ideas relating to agent activities, have been activated. Those repositories can be divided into the following three main types:

- repositories of concepts and activities in order to model human activities,
- formal repositories corresponding to formalize abstract concepts,
- software repositories for the construction of applications.

In the field of MS (Management Science) as the first type, one of the famous results is the e-business Process Handbook project[9] carried out by MIT. The e-business Process Handbook is a substantial contribution as a business repository, which contains approximately 4,600 definitions of business activities from abstract processes to the specialized one to the business over the Internet. Its formality, however, is not strict since the most part of the definitions are described by natural language.

From the viewpoint of the formality on the process specification, there is the enterprise ontology[12] of Edinburgh University in the field of artificial intelligence. Its formality is very strong and it contributes the reuse of business models nevertheless it covers only so general and abstract concepts that it is very hard to construct concrete business models with operability.

On the other hand, a lot of software libraries for building EC applications have been proposed[10, 15, 16]. Most of them are originally developed as repositories for the agent applications and extended to ones for the business applications[17, 18, 19]. Furthermore, the special platform for the development of enterprise applications is provided. Those libraries and platform offer the strong framework for the construction of real applications. There are, however, no clear relationships between software components and business models.

Owing to the difference of purposes and viewpoints among those repositories, the integrated support is hardly performed in the construction and the re-engineering of business applications on the existing domain.

So, in order to achieve the unified support in the construction of agent models and applications, the computing framework, which integrates the different sorts of repositories, should be developed.

From the standpoint of the above-mentioned background, we propose the development methodology of agent applications based on ontologies with reusable repositories such as e-business process handbook and software libraries.

In order to construct the business models and to implement them on the existing domain, we rebuild the heterogeneous repositories into two repositories on different-grain-levels: the agent specification repository on the level of business activities and the agent software repository on the level of software applications.

The former one is for modeling agents' activities from specification descriptions, and the latter one is for constructing agent applications from the agent models.

As the main characteristic of this work, we provide the integrated support of modeling and implementation by employing primitives of PSMs (Problem Solving Method), which provides the correspondence between the models and applications. For the purpose on the integration of heterogeneous repositories obtained over the Internet, we employ Common KADS inference primitives and WordNet as standard ontologies.

We have implemented the prototype system by JAVA and confirmed that it supports us in various phases of agent application development including agent model manifestation, detailed agent model definition and an implementation of agent software applications.

2. OVERVIEW OF PROPOSED DEVELOPMENT SUPPORT

In order to achieve the unified treatment of models on different levels such as making agent models clear, implementing them as the detailed inference process, building software applications and so on, our research aims at the establishment of integrated support from the analysis level to the implementation level. Our standpoint of the integration and the reuse of heterogeneous repositories is to obtain the key structure of each repositories, which corresponds to noun concepts, and to relate them each other to bridge the whole structure of processes and activities including verb concepts.

The key idea to achieve our aim is how to extract the verb and noun concepts and their relationship as the common structure of information from the heterogeneous repositories. The noun concepts can be divided into two main categories: higher level concepts and system implementation level concepts. We call them 'business objects' and 'software objects', for each. The verb concepts can be divided into three main categories: concepts for human-level behaviour, abstract-level primitives for problem solving, and fine-grain size method for sysmtem implementation. We call them 'activity', 'PSM: Problem Solving Method', and 'action' for each.

The target of development support is the construction of agent application as a agent models obtained from specification documents, which contains the facility such as communication with users through network via E-mail or Web, and the file system sharing with the users for customer relationships and data management.

For the purpose of the efficient development, we employ the methodology of software patterns of analysis and design. Due to a variety of patterns, we also employ the MIT's e-business Process Handbook as a standard ontology for classification and organization of activities.

Brief description of agent application development through the repositories integration by our framework is as shown in Figure 1.

At first, in order to pick the key concepts up from specification documents, we construct the agent specification repository including ontologies of noun and verb concepts. Noun concepts are extracted from the e-business Process Handbook and classified into the business object ontology according to WordNet[2] as the general concept ontology provided by Princeton University. Verb concepts of e-business Process Handbook are re-arranged by using frequency information of noun-concepts-appearance. By making reference to the history database of the correspondence between words of documents and ontologies of nouns and verbs, an agent model is constructed by the agent specification repository.

Next, as a repository for transplantation of the agent model to an agent software application, we provide the agent software repository.

In order to correspond activities of agent models to software modules, we employ the inference catalogue of Common KADS[11, 14] as the standard concepts of PSMs. Common KADS is utilized for analysis and development of knowledge systems and offers the language and primitives to clarify conceptual models. Based on the primitives of the inference catalogue, we construct a library of typical patterns of the input-output relations and the module structure by rebuilding JAT (JAVA Agent Template)[10] provided by Stanford University.

In order to standardize the format of patterns, we construct a software object ontology that provides the classification of software objects with control and data structures. Each pattern is formalized by the vocabulary of the software object ontology and Common KADS.

By consulting the software object ontology and REPOSIT[6] provided by Shizuoka University as a library of the implementation patterns, the agent model, obtained on the previous stage, are supplemented with control structures of software codes. According to the frequency and history of corresponding among three libraries of JAT, Common KADS and REPOSIT, a detailed model of the software application are obtained based on the software object ontology.

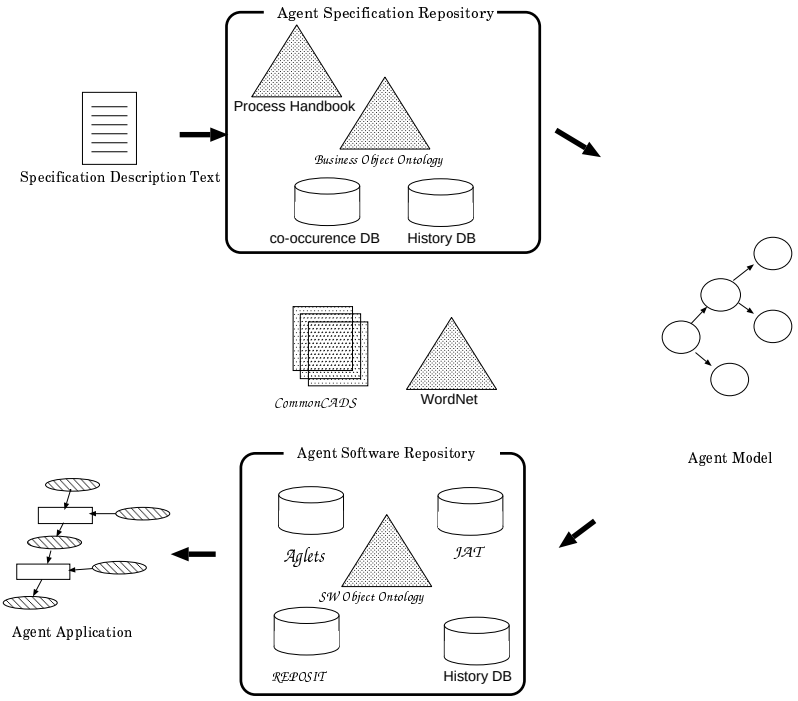

Figure 1: Overview of Development Support

3. CONSTRUCTION OF AGENT MODELS

In this work, we employ the e-business Process Handbook of MIT, called Process Handbook for short, as a library classifying agents' activities on their inference processes. There still remains an issue in the extraction of a required process due to the difference of the viewpoints between specification descriptions and process repositories. In order to bridge the difference, we construct the agent specification repository by employing WordNet as a general lexical repository. First, the agent specification repository is provided as a key structure bridging the specification documents and Process Handbook. Then, the wrapper framework is constructed as an extract method of required activities from a repository of Process Handbook.

3.1 Building Object Ontology

In order to classify the noun concepts extracted from Process Handbook, we employ the WordNet as a general ontology that contains over 17,000 concepts. However, if we utilize WordNet as it is, the number of candidate explodes because of the variety of the word's meaning and the ambiguity of the word in a document. When the abstraction degree of a requirement is high, a verb concept of an activity in the goal is often vague for specification makers due to the difference of viewpoints on the definitions. In contrast, a noun concept of the activity is comparatively clear and appears regularly in the document.

So, in order to classify the noun concepts appearing in Process Handbook, we choose the major concepts with respect to the degree of abstraction and frequency by using WordNet. As the criteria to select a major noun concept, we pay attention to roles of input-output objects in the definition of Process Handbook. Based on the above standpoint, we construct the object ontology in the following way:

- concentrating a substructure for each verb concept from the process hierarchy,
- extracting a noun hierarchy from the concentrated substructure,
- counting the number of the appearance of each noun concept,
- adding the number counted for the upper concepts,
- merging all the noun hierarchy by introducing the abstract concepts according to the WordNet.

According to the WordNet's structure, we define the substructure of concepts obtained above as an upper ontology, provided that the priority of the meaning is given to the application domain over the relation of WordNet. Furthermore, when we fix the upper ontology for constructing the domain ontology, concept drift, that is a kind of semantic shift on a specific domain, often occurs and causes inefficiency on building ontologies. Due to reduce the cost of construction, we employ the methodology for resolving the concept drift[13]. Figure 2 shows the structure of the object ontology obtained.

3.2 Determining Agent's Activities

When obtaining the definition of the business activity corresponding to a requirement document, it is difficult to utilize the hierarchical structure of the process handbook because

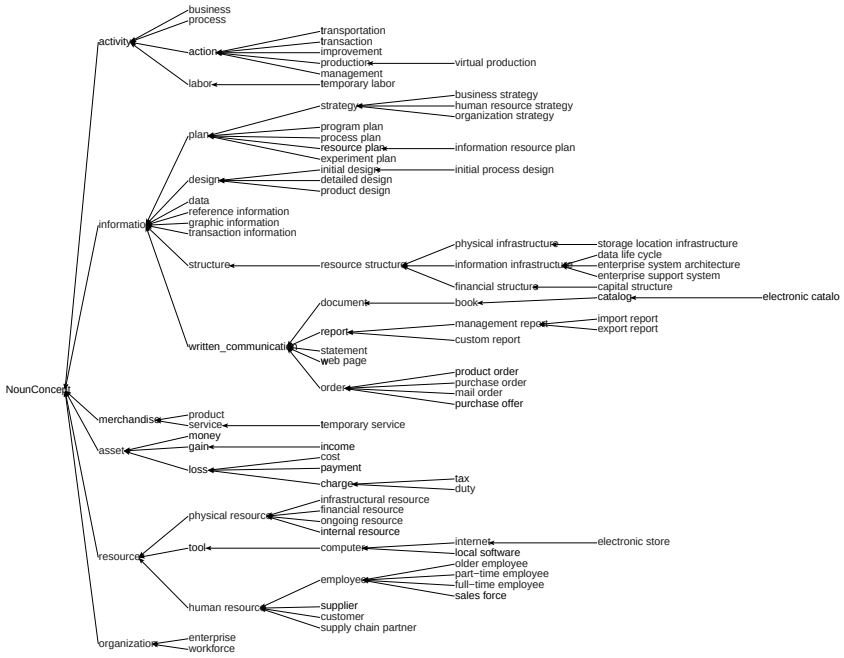

Figure 2: Business Object Ontology (a part of it)

of the gap between words of actual documents and process handbook. On this account, we employ the object ontology, as an upper ontology, which bridges the variety of words in documents and nouns in Process Handbook. In order to identify an agent's activities from a sentence given by a user, we devise an extraction mechanism as a wrapper for Process Handbook based on the object ontology.

The wrapper tool is composed of the databases of the following information. First, the co-occurrence information of noun concepts in the definition of an agent's activity is obtained and classified with respect to the structure of the object ontology. Then, the information is made accessible as the database of the co-occurrence. At the same time, the frequency information is also available in collecting the co-occurrence one. By using both of the co-occurrence and frequency information, the wrapper tool help us to search the definition of activities in the space of Process Handbook. The proto-typing tool is shown in Figure 3.

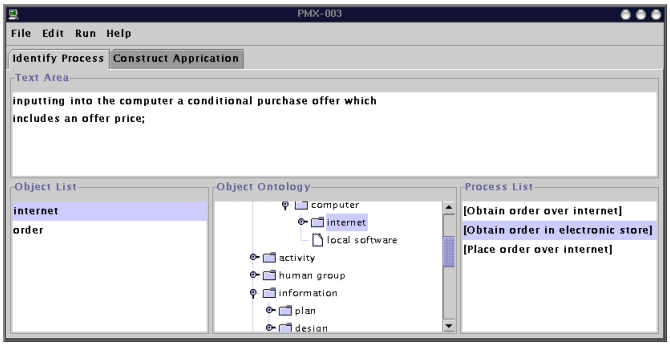

Figure 3: The Wrapper Tool

4. BUILDING SUPPORT OF AGENT APPLICATIONS

4.1 Building Data Structure Ontology for Reuse of Libraries

In the same way of building the object ontology, the data structure ontology (Figure 4) is constructed as a domain independent ontology from the libraries intended to employ. The software object ontology gives words for expanding domain ontologies such as building a set, picking up an atom of a set, indicating a calculation stage, data structures for implementation details and so on.

On the purpose of the developing agent applications from the agent model obtained above, a detailed definition of each activity of the model is required. In order to give the activities the operational information that is used for application development, we prepare the library of the application template, which defines the structure of the part of application in the fashion of the knowledge system development with respect to the data structure ontology. By constructing the business software repository with the reusable template of REPOSIT, Common KADS, JAT and historical databases, the agent application is obtained.

4.2 Model Refinement Based on Application Templates

We consider the structure of agent applications based on the agent architecture to be composed by the inference engine to attain a task, the sensor to get the information of the outside and the effector to carry out their task. The sensor is characterized by the following three functions:

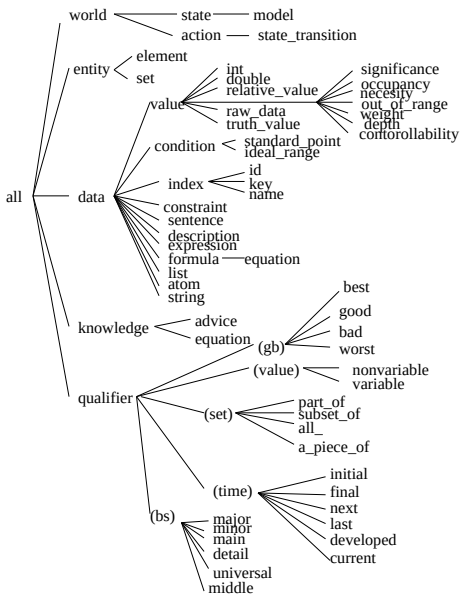

Figure 4: Data Structure Hierarchy

(1) the function that accesses the inside and the outside resources of the agent,

(2) the function that examines the place and the contents of resources,

(3) a function to acquire a message from the user and to interpret the message.

The effector is defined by two of the next:

(a) the function to form and modify the inside and the outside resources of the agent,

(b) a function to make and to send a message to the user.

The framework of the combination with the above functions and the inference engine is organized as agent templates by referring to JAT (JAVA Agent Template) of the Stanford University. Furthermore, detailed templates, corresponding to eight types of communication models given by Common KADS, are formed as interaction templates with the user and resources (Figure 5).

4.3 Building Reusable Templates for Implementation of Applications

From the importance of a unified language for the reflection of the change on a business model, we rebuild and extend inference primitives of Common KADS into "REPOSIT (REusable Pieces Of Specification-Implementation Templates)" which combines declarative semantics employed in Common KADS and procedural semantics like Prolog. A unit of a description in REPOSIT, defined as a relationship among input, output and reference knowledge, is called a "unit function".

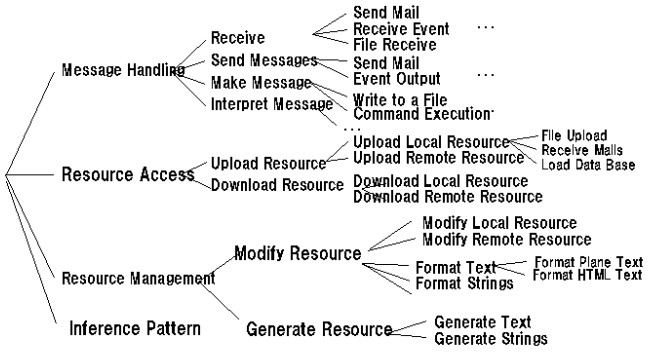

Figure 5: Primitives of Application Templates

A set of unit functions is rebuilt into the method ontology by abstracting knowledge types of input, output and reference.

Furthermore, patterns of a combination of unit functions, which appear frequently in the development process, are gathered, sorted out and constructed as a method library based on the following standpoint:

(1) providing refinement policies,

(2) standardizing a way of the knowledge (data) management,

(3) classifying the adding patterns of control structures given to specifications.

In order to keep a correspondence between descriptions of specifications and implementations, REPOSIT supports step-by-step operationalization of an abstract description into a detailed implementation model, as the following way (Figure 6):

a. selecting a pattern of the method library according to a task type of a knowledge system,

b. concreting knowledge type of input, output and reference by using the software object ontology and the obtained business model as the requirement specification,

c. adding a control structure to the description with the obtained information of knowledge type,

d. selecting a pattern for each unit function of the description and continuing the above process.

Finally, we've provide 22 methods on the abstract-pattern level, 92 methods including prolog-build-in functions on the program-code level, and 69 methods on the middle-level.

5. EXPERIMENTAL STUDY

5.1 Personal Agent Construction

In order to consider the validness and usability of proposed framework, we've implemented the above mechanism by JAVA

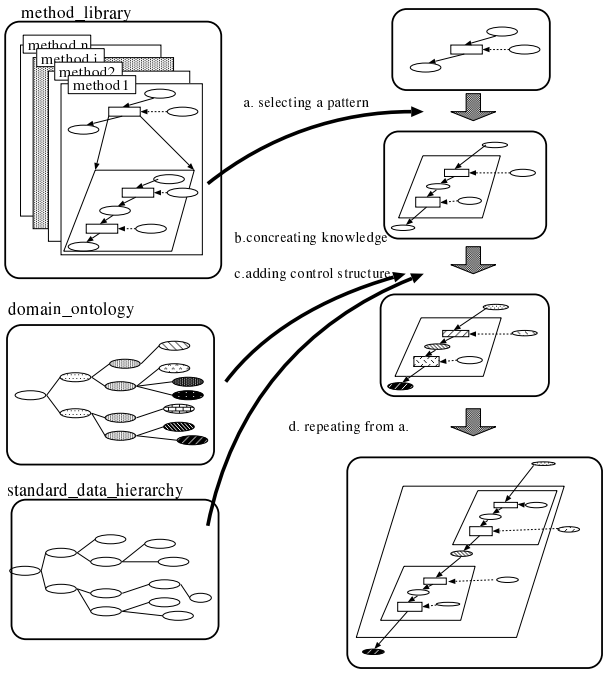

Figure 6: Overview of a Development Process

into the proto-typing tool and applied it into case studies of constructing agent applications from description documents. We take an agent's task to support the management of the large-scale lecture held all together as this case study.

We suppose the agent to act like a human secretary; therefore he is personified as in the same conditions as human users on UNIX OS who have their own home directory on UNIX, an E-mail address and URL. The agent is made start to perform by receiving E-mails on his address or by special commands via GUI of JAVA applets. In this case, we provide the following functions as the input/output mechanisms: reading/writing function of the file system including HTML files, sending/receiving function of E-mails, exchanging function of data on JAVA applets, general UNIX commands and so on (Figure 7).

Developers selected as users couldn't make the software system, which satisfied the requirement in advance with the C language or JAVA.

The outline of the task that the agent, developed by a user with the agent wizard's help, has done and the contents of a class subject where the agent was actually used are shown in the followings:

A use result in advanced programming is as follows: Class size: 100 people * 1 class, Main task carried out: Reports reception via E-mails (15 exercises 1382 mails for total), Evaluation of the reports received (same as above), Making records of submission, evaluation, marks and so on (15 exercise for 100 students; Updated averagely five times.), Information presentation to Web (10 exercises, updated 5 times averagely) ,

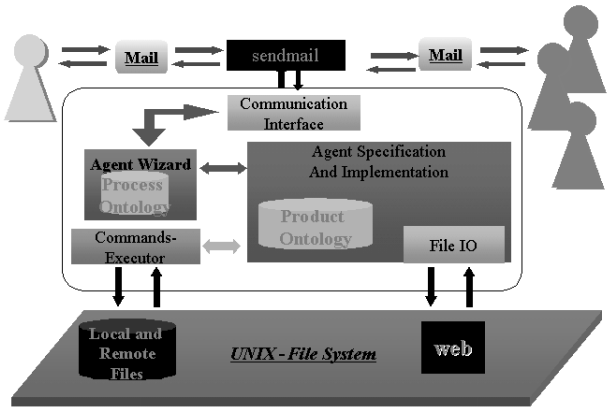

Figure 7: An Overview of the Agent Development

A use result with computer literacy is as follows: Class size: 50 people * 4 classes, Main tasks carried out: Reports reception via E-mails (5 subjects 1935 mails for total), Evaluation of the reports received (same as above), Making records of submission, evaluation, marks and so on (3 exercises for 200 students; Updated averagely 3 times for each exercises). Information presentation to Web (10 exercises, updated 5 times averagely).

Roughly speaking about the result, 52 methods of the implementation-level are used averagely for each task. Implementation has been completed around 18 hours after receiving the specifications on the advanced programming. At this time, the agent wizard interacted with a user about the task-specific requirement such as evaluation of programs. Furthermore, Implementation has been completed around 3 hours after receiving the specifications on the computer literacy. It seems that an experience of the agent development at the former lecture has been propagated to the next lecture.

5.2 e-Business Application Construction

As another experimental study, in each case study, some patent texts obtained from the Internet are used as a specification document. We have compared between the models of case studies provided by Process Handbook and the ones built by the proto-type tool (Figure 8).

Roughly speaking about the result, approximately 70 % activities of each case study model are determined from patent texts, and, each implementation has been completed around 18 hours after receiving the patent text of claims and details as specifications.

The above means that the cost of the application development has been reduced more than 50% as compared with by hand, that reuse of the system has been performed about the common structure of agent applications, and that main agent structure could be reused if we have stacked and open some experience to the public at our library. Now, we are analyzing and investigating about the deeper experimental studies on the development on heterogeneous systems and

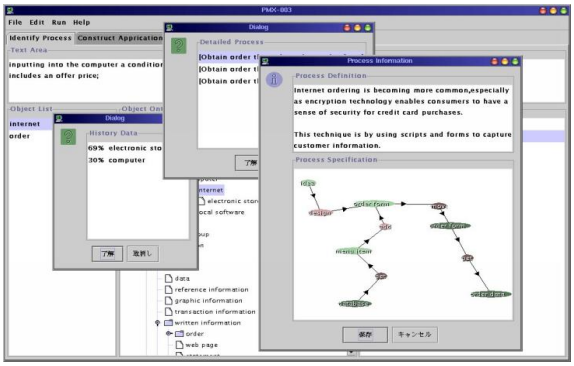

Figure 8: Execution of Supporting Tool

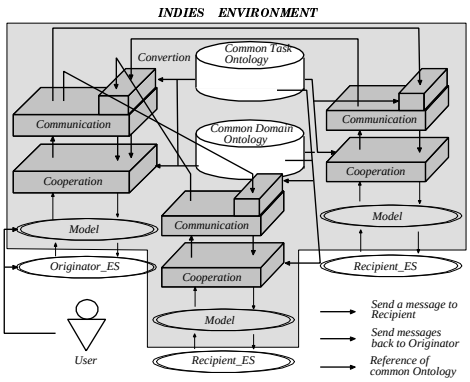

Figure 9: An Overview of INDIES

the result will be open until the conference.

6. DISCUSSIONS

The above result of the experimental study means that the cost of the application development has been reduced more than 50% as compared with by hand, that reuse of the system has been performed about the common structure of agent applications, and that main agent structure could be reused if we have stacked and open some experience to the public at our library. Now, we are analyzing and investigating about the deeper experimental studies on the development on heterogeneous systems and the result will be open until the workshop.

As comparison with related work, there are three main fields of research areas: clarifying specifications, building an application and reusing existing libraries.

First, as discussed in Introduction, numbers of work on analyzing business specification has been done by MIT, Edinburgh University and so on. Their work is very significant as a fundamental research; however, most of them are around abstract and general framework. Recently, Process Handbook is revised into e-Business Process Handbook and provides a hundred of specification as case studies. But most of them are just defined by natural language text. So, our work can be regarded as the integrated work to utilize the related work.

Second, a lot of works are carried out on building applications Including software engineering field[1, 3], but they use the different type of tool and languages on the different phases of development. So, models and languages should be unified into on framework as we proposed.

Our framework is based on a standpoint that it is difficult to automate the whole agents' activities, but possible to do many part of it. Recently, a few of the researchers consider the modeling methodology of the whole enterprise structure as a multi-agent system[8]. It is worth paying attention but still remain on the abstract structures of defining agent roles.

Finally, a lot of projects concentrate on reusing libraries, ontologies and applications and provide a number of repositories. One of our previous works is on interoperation of the heterogeneous expert systems[5]. Because each expert system is modeled by its own vocabulary, it needs a conversion facility so that it can understand the messages sent from other expert systems. In the work, we employ a specification-sharing(SS)-based cooperation, called assisted coordination[4]. The shared specification comes from REPOSIT library which serves as a common structure of noun and verb concepts named the common domain ontology and the common task ontology. As the methods of modeling, operationalizing, cooperating and communicating (wrapping) distributed expert systems come up, we put them together into an interoperation environment for distributed expert systems INDIES(Figure 9).

In our previous work, a number of significant lessons obtained in exchanging the messages among the heterogeneous expert systems. However the way to construct the common ontologies is still remain as the future work. Our current work can be regarded as a new approach to reuse heterogeneous repositories based on ontologies.

7. CONCLUSION

As conclusion, the computing environment, which supports dynamic construction of agent models and applications from specification document, should be developed for the purpose to perform the re-engineering agents' inference processes according to the changes of situations in their computing environments and their tasks. From standpoint that the heterogeneous repositories should be integrated to achieve the unified support of the application development, we have proposed the framework of the extraction of the required information based on ontologies with reusable repositories such as e-business process handbook, Common KADS and REPOSIT. In order to construct the agent models and to implement them as the actual agent including software applications, we develop two repositories on different-grain-levels: the agent specification repository on the level of agent activities and the agent software repository on the level of software applications. For the purpose on the integration of heterogeneous repositories obtained over the Internet, we employ Common KADS inference primitives and WordNet as standard ontologies.

We have implemented the prototype system by JAVA and confirmed that it supports us in various phases of agent application development including agent model manifestation, detailed agents' inference process and an implementation of agent software applications. Furthermore, we are

re-organizing our product in order to open it to the public.

8. REFERENCES

[1] P.Code, D.North, M.Mayfield, "Object Models:Strategies, Patterns, & Applications", Yourdon Press, 1997.

[2] C.Fellbaum ed: "Wordnet", The MIT Press, 1998.

[3] M.Fowler, "Analysis Patterns: Peusable Object Models", Addison-Wesley, 1997.

[4] M.R.Genesereth and S.P.Ketchpcl: Software Agents, CACM.ol.37.No.7. (1994) 48–53.

[5] N.Izumi, A.Maruyama, A.Suzuki, T.Yamaguchi: "An Interoperative Environment for Developing Expert Systems" Proc. EKAW'99 LNAI.1621, pp.335–340, Springer–Verlag (1999).

[6] N.Izumi, A.Maruyama, A.Suzuki, T.Yamaguchi: "Design and Implementations of Reusable Method Library for Development of Expert Systems", Journal of JSAI, 14, 6, 1061–1071, (1999), in Japanese.

[7] N.Izumi, T.Yamaguchi: "Developing Software Agents Based on Product Ontology and Process Ontology", Proc. ECAI-00 Workshop on Applications of Ontologies and Problem–Solving Methods (2000) 8-1—8-6.

[8] E.A.Kendall, "Role Models: Patterns of Agent Analysis and Design", British Telecom Journal Special Issue on Decentralized Business Systems, (1999).

[9] The MIT Process Handbook Project: http://ccs.mit.edu/ph

[10] C.J.Petrie: "Agent-Based Engineering, the Web and Intelligence" IEEE Expert, 1996. URL: http://java.standord.edu/

[11] Guuns Schreiber, et al: Knowledge Engineering and Management: The CommonKADS Methodology,MIT Press (1999).

[12] M.Ushold, et al: The Enterprise Ontology, Knowledge Engineering Review,Vol.13,Special Issue on Putting Ontologies to Use(1998).

[13] T.Yamaguchi: Constructing Domain Ontologies Based on Concetp Drift Analysis, IJCAI Workshop on Ontologies and Problem-Solving Methods: Lessons Learned and Future Trends , 13-1 - 13-7, (1999.8)

[14] See http://www.commonkads.uva.nl

[15] See http://www.trl.ibm.com/aglets/

[16] See http://java.sun.com/j2ee/

[17] See http://www.ftp.com/

[18] See http://www.genmagic.com/

[19] See http://www.objectspace.com/