A Collaborative Framework to realize Virtual Enterprises using 3APL

Gobinath Narayanasamy[1], Joe Cecil[2], and Tran Cao Son[1]

[1]Computer Science Department
New Mexico State University, USA
{gonaraya,tson}@cs.nmsu.edu
[2] Department of Industrial Engineering
New Mexico State University, USA
jcecil@nmsu.edu

Abstract. In this paper, we propose a collaborative framework to realize a Virtual Enterprise (VE) for the domain of Micro Assembly. The framework is developed using 3APL technologies [5] and employs the idea of viewing Web-Service composition as a planning problem [10]. We describe the implementation of the framework and experiment with two micro assembly work cells.

1 Introduction

In today's business world, being innovative and withstanding competitive pressure from contemporary business vendors are a key to success for any business vendors. With dynamic nature of consumer demands, business vendors often need a sophisticated mechanism to tap those momentous market demands. One such mechanism which will facilitate as well as satisfy the business vendors need is the concept of a Virtual Enterprise (VE). A VE is a conglomeration of different business vendors under one hood (to meet the market demands arising from consumers) by sharing their own resources and expertise, which – sometime – cannot be provided by a single business vendor. Each of the business vendors participating in a VE has different resource capabilities. Here a resource is anything that is necessary for the production of a product. It can be a machine, a software program, a component, a service, etc. Each resource might have a cost associated with it. Furthermore, there might be a resource, which can be used in the assembly of a product and is available in several places. The diversifying nature of a VE causes heterogeneity which slows down the process of forming collaborations among the vendors.

Our goal is to develop a framework that facilitates collaborations and seamless flow of information exchange among the partners in a VE. We explore this idea using the agent technologies provided by the 3APL framework [5].

We develop a prototype VE using the proposed collaborative framework in the Micro Assembly domain. Micro Assembly is the domain where parts in micron sizes are assembled using computer enabled micro assembly work cells. We target this domain for the following reasons: (*i*) it is considered as a better alternative to Micro

Electro Mechanical Systems (MEMS) where parts having varying material properties cannot be manufactured; (*ii*) it is a completely new methodology for developing products in manufacturing sector, and hence, not many business vendors possess the whole range of tools and resources to accomplish micro assembly related tasks such as micro assembly planning, simulation and actual physical implementation; and (*iii*) each business vendor possesses different micro assembly resources which – when used together – can accomplish most of the customer requests related to Micro Assembly.

As many parts in the Micro Assembly domain are assembled using computer programs and the VE is virtually available on the internet, we need a multi-agent development platform in which agents with various capabilities can be created. Each agent should have their own belief, capabilities, goals, and rules for reasoning. This platform should also facilitate the agent communication and collaboration. This is the reason why we choose 3APL as our implementation platform.

The paper is organized as follows: Section 2 provides a review of some past and recent developments of Virtual Enterprises using agent based approaches. Section 3 highlights the 3APL framework. Section 4 describes the collaborative system design. Section 5 discusses the development of collaborative framework using 3APL. Section 6 discusses VE formation for Micro Assembly domain using the proposed collaborative framework and Section 7 is the conclusions.

2 Literature Review

In this section, background information about virtual enterprises as well as a review of agent based systems is provided. Other issues such as agent communications, agent interaction protocols, and distributed problem solving approaches in agent based systems are also discussed.

In [12], it is observed that Co-operative or Concurrent Engineering (CE) techniques are the reason for forming collaborative working environment among company levels. In [3], a consortium of companies is called a Virtual Enterprise (VE) which allows for the development of a working environment to manage all or part of different resources towards achieving a common goal. Common information definition and sharing problem while forming Virtual Enterprises are discussed in [4]. The paper also discusses the issues of interaction among the companies that will agree upon a contract to form virtual enterprise.

In [8], the concept of forming Virtual Enterprises using agent based systems is proposed. In this conceptualization, partners of a virtual enterprise are considered as software agents. This paper also discusses different agent communication protocols such as KQML and KIF. A significant agent communication protocol proposed by US Defense Advanced Research Projects Agency's (DARPA) Knowledge-Sharing Effort known as Knowledge Query Management Language (KQML) is presented in [7]. The language includes variety of primitives, assertives, and directives which allow agents to query other agents, subscribe to other agents services, or find other agents for distributed problem solving. KQML assumes that each agent is built with its own knowledge bases. This allows other agents to extract information from the knowledge base of that particular agent. In the context of micro assembly, a sample KQML message between two agents, *Path_Planner* and *Service_Locater*, the former requests the latter for information about providers of a certain service with the help of ontological information is

```
(tell   :sender Path_Planner  :receiver Service_Locater
        :ontology Micro_Assembly_ontology  content publish(Service)).
```

In [6], Knowledge Interchange Format is emphasized. KIF is a language for interchanging knowledge between heterogeneous programs. KIF has a declarative semantics which allows agents to understand a KIF representation without any interpreters. It allows expressing arbitrary sentences using first order predicate calculus. It has constructs to represent knowledge in the domain, represent non monotonic reasoning rules and define objects, functions and relations. KIF has been employed in the development of the Process Specification Language (PSL), a language specifically designed to facilitate correct and complex exchange of process information among manufacturing systems [13].

In [10], it is observed that web services markup will allow agent technologies to efficiently capture the 'meta' data associated with the services and reason about them. This paves way for agent technologies to perform automated web services discovery, execution, composition and interoperation. In automated web services discovery, the software agent automatically discovers the web services based on user constraints, which is performed manually in the current World Wide Web (WWW). In automated web services execution, the software agent discovers the web services based on user constraints, understands the requirements for the services, and executes them automatically. In automated web services composition and interoperation, the software agent selects the required web services, compose and interoperate them to accomplish the requested complex task.

In [11], a need is identified to automate the process of discovering, executing, composing, and monitoring services. Automation refers to no human intervention and allows for the use of software agents. For a software agent to automatically process and execute a service, a machine understandable description of the service is required. One such language which provides descriptions that are machine understandable is OWL-S which is evolved as a collaborative work of BBN Technologies, Carnegie Mellon University, Nokia, Stanford University, SRI International, and Yale University.

In [2], the importance of using ontologies in manufacturing domain is explained. The paper emphasis on the need for developing richer ontological structures especially to the manufacturing domain so that more sophisticated intelligent applications can be developed.

3 3APL Language

An Abstract Agent Programming Language (3APL) developed at Universiteit Utrecht is a new agent oriented programming language for developing agents with cognitive capability, as given in [5]. The language comes with programming constructs that allows developing agents with complex mental states. A 3APL agent developed using this language is a tuple of the form <B, G, P, A> where,

- B is Belief base,
- G is Goal base,
- P is a set of Practical reasoning rules and
- A is an Action base.

Each of this component is briefly explained next.

3.1 Belief Base

A belief base encodes the agent knowledge about its operating environment and is a set of first order sentences. For example, a belief that a robot at room A is represented by the atom *at*(*Robot*, *RoomA*); other belief that a robot is not at the room *x* then it is at the room next to *x* is expressed by the sentence[1] $\forall x,y(\neg at(Robot,x) \wedge nexto(x,y) \Rightarrow at(Robot,y))$. Notice that a belief base can contain non-grounded sentences.

3.2 Goal Base

A goal base consists of *goals-to-do* goals. 3APL considers goals of procedural type. Under this view, a goal can be considered as an imperative program. A goal defines a plan of actions for an agent to execute. The language allows for the definition of simple and complex goals.

Simple goals (also called basic goals) are of three types: basic action, test goal, and achievement goal. For example, a simple goal like *inquireUDDI*() allows an agent to inquire the UDDI registry.

Complex goals (also called composite goals) are composed from basic goals and are used to specify complex actions such as sequences of actions, disjunctive goals, or non-deterministic choices, etc. Conventional programming constructs such as ';' and '+' are used to create complex goals. For example, "$goal_1$; $goal_2$" defines a sequence of goals and "$goal_1$+$goal_2$" defines a disjunctive goal.

3.3 Practical Reasoning Rules

A 3APL agent can manipulate its goals by using practical reasoning rules. These reasoning rules allow an agent to find plans, which help him/her achieve its goals. They also allow the agent to monitor its goal base. These rules facilitate the *practical reasoning* which an agent can use to decide (*i*) to adopt a plan for achieving a goal; (*ii*) to revise a plan if necessary. The set of practical rules is built from semi-goals and first order formulas where semi-goals are defined similar to goals using a new set of variables.

[1] The sentence might or might not be valid.

A practical reasoning rule is of the form

$$\pi \leftarrow \varphi \mid \pi',$$

where

- π is the head of a the rule,
- φ is the guard of the rule and
- π' is the body of the rule,
- Global variables are free first order variables in the head of the rule, and
- Local variables are non global first order variables in the body of a rule.

A practical rule $\pi \leftarrow \varphi \mid \pi'$ says that if the agent adopts some goal or plan π and believes that φ is true, then it may consider adopting π' as a new goal.

3.4 Action Base

Action base defines the set of primitive actions (or basic actions) that an agent can execute. This set of basic actions defines the capabilities of an agent with which an agent can change its mental state of belief about its working environment.

4 Framework Design

We follow the idea behind the design of this system follows the model proposed in [8] and [9]. We view each partner in a VE as an agent who has its own knowledge about the environment, its actions (basic and complex), its set of practical rules, and its own goals. A VE is a collection of agents who collaborate to achieve a common goal. As we have discussed above, most activities in the Micro Assembly domain are controlled by computer programs. As such, each partner is implemented as a software agent who can offer their services (or actions) to others. Our framework facilitates the communication between agents and allows users of the system to simulate the VE. The overall design of our framework is depicted in **Figure** 1.

Central to our system is a central manager agent which is a 3APL agent. This agent facilitates the communication between different agents and creating solutions for users' requests.

An agent can advertise its services in a service directory, which is implemented as a part of our system. A 3APL service directory agent provides other agents in the system the capability to find service provider(s) that can satisfy their needs. This agent communicates with other agents through the agent manager. In our implementation, each service is specified by its inputs and its execution method.

One issue in a collaborative framework is the semantically differences between different agents. This is also an issue in our framework. We follow others by addressing this issue using ontologies and develop ontologies for the Micro Assembly domain. To incorporate ontologies into our system, a 3APL agent is developed. This agent also communicates with other agents through the agent manager. We call this the meta-information of services.

We note that in [1], design and development of ontologies for physical devices are explained.

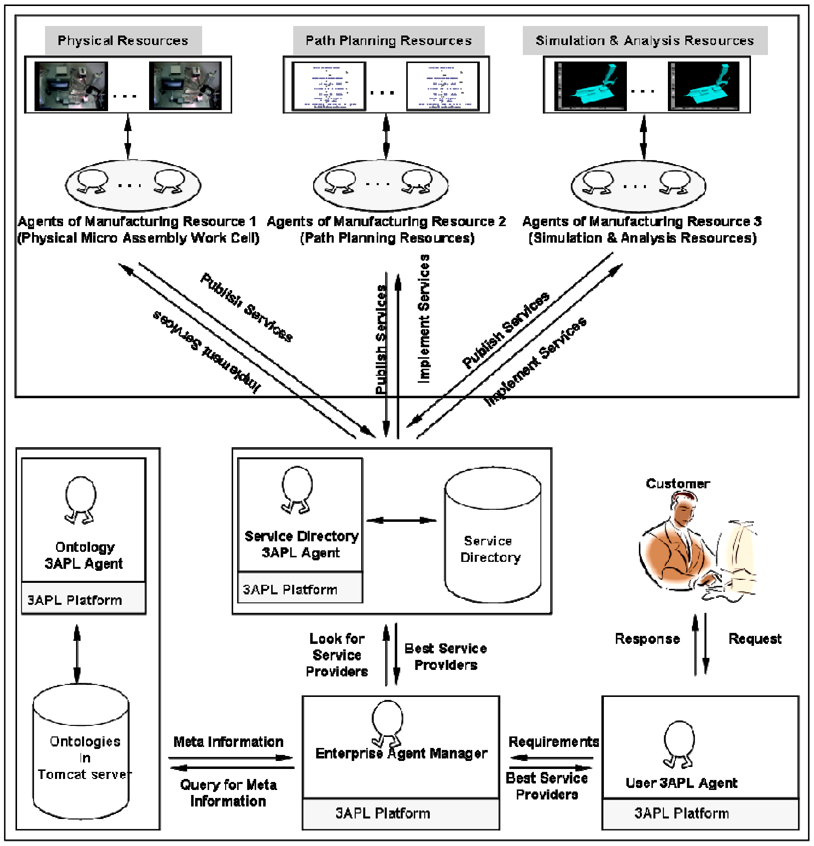


Figure 1. Collaborative System for Virtual Enterprise

5 Framework Implementation

This section discusses the implementation of the collaborative framework as shown in Figure 1. It consists of following agents:

1. User Agent
2. Virtual Enterprise Agent (or Enterprise Agent Manager)
3. Ontology Agent
4. Service Directory Agent and

5. Service Provider Agents

All these agents are implemented using 3APL and they run in 3APL platform. Plug-in programming construct is provided by 3APL platform so that agents can use the plug-in as their working environments and access the methods available in them. With the help of plug-ins, agents in 3APL platform can access the external JAVA methods, virtually allowing an agent to *execute* a service provided by another agent. For each agent in the our system, an associated plug-in is developed to assist the formation of Virtual Enterprise in real time. Detailed descriptions of 3APL agents used in the collaborative system are given below.

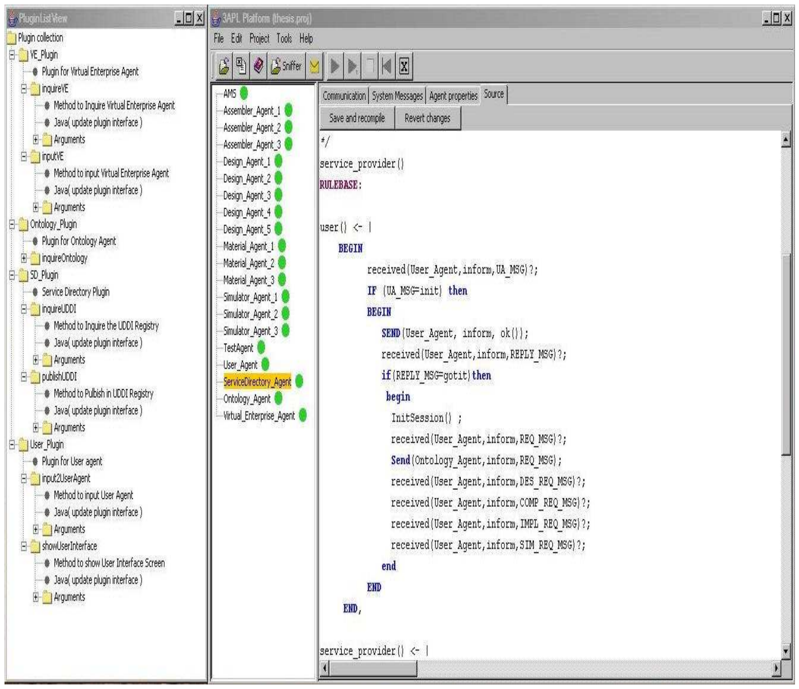
5.1 User Agent

User Agent provides the user interface to the collaborative system. This agent is probably the simplest agent in the system. It acts on behalf of real world entities such as human users, software applications, or even other business vendors who may need to accomplish a task.

5.2 Virtual Enterprise Agent

The Virtual Enterprise Agent coordinates the various activities in the collaborative framework. It is responsible for processing users' requests (from the user agent) and providing an initial solution (i.e. *plan*) for these requests. In the course of finding this solution, it queries the Ontology agent for meta-information and uses this information to find a list of best available service providers by querying the Service Directory agent.

The Virtual Enterprise Agent also serves as a search engine for other agents who need to find service providers for their own needs. **Figure** 2 shows a view of collaborative framework implemented in 3APL platform with developed plug-ins and participating software agents



```
Plugin collection
  VE_Plugin
    Plugin for Virtual Enterprise Agent
    inquireVE
      Method to Inquire Virtual Enterprise Agent
      Java( update plugin interface )
      Arguments
    inputVE
      Method to input Virtual Enterprise Agent
      Java( update plugin interface )
      Arguments
  Ontology_Plugin
    Plugin for Ontology Agent
    inquireOntology
  SD_Plugin
    Service Directory Plugin
    inquireUDDI
      Method to Inquire the UDDI Registry
      Java( update plugin interface )
      Arguments
    publishUDDI
      Method to Pulbish in UDDI Registry
      Java( update plugin interface )
      Arguments
  User_Plugin
    Plugin for User agent
    input2UserAgent
      Method to input User Agent
      Java( update plugin interface )
      Arguments
    showUserInterface
      Method to show User Interface Screen
      Java( update plugin interface )
      Arguments
```

```
AMS
Assembler_Agent_1
Assembler_Agent_2
Assembler_Agent_3
Design_Agent_1
Design_Agent_2
Design_Agent_3
Design_Agent_4
Design_Agent_5
Material_Agent_1
Material_Agent_2
Material_Agent_3
Simulator_Agent_1
Simulator_Agent_2
Simulator_Agent_3
TestAgent
User_Agent
ServiceDirectory_Agent
Ontology_Agent
Virtual_Enterprise_Agent
```

```
*/
service_provider()
RULEBASE:


user() <- |
    BEGIN
          received(User_Agent,inform,UA_MSG)?;
          IF (UA_MSG=init) then
          BEGIN
             SEND(User_Agent, inform, ok());
             received(User_Agent,inform,REPLY_MSG)?;
             if(REPLY_MSG=gotit)then
              begin
               InitSession() ;
               received(User_Agent,inform,REQ_MSG)?;
               Send(Ontology_Agent,inform,REQ_MSG);
               received(User_Agent,inform,DES_REQ_MSG)?;
               received(User_Agent,inform,COMP_REQ_MSG)?;
               received(User_Agent,inform,IMPL_REQ_MSG)?;
               received(User_Agent,inform,SIM_REQ_MSG)?;
             end
          END
     END,

service_provider() <- |
```

Figure 2. Collaborative System for VE using 3APL

5.3 Ontology Agent

The Ontology Agent in the collaborative system provides the necessary meta-information for the VE agent to further process the input from the user agent. For demonstration purpose, some sample ontologies are created using Stanford's Protégé editor. **Figure** 3 displays a part of the ontology developed for the Micro Assembly domain.

The ontologies developed for the collaborative system are deployed in a Tomcat web server. Any modifications to the existing ontologies are done through the ontology agent. This is achieved by means of a Ontology plug-in developed to assist the ontology agent. Ontology plug-in contains some basic functions for querying and modifying existing ontologies.

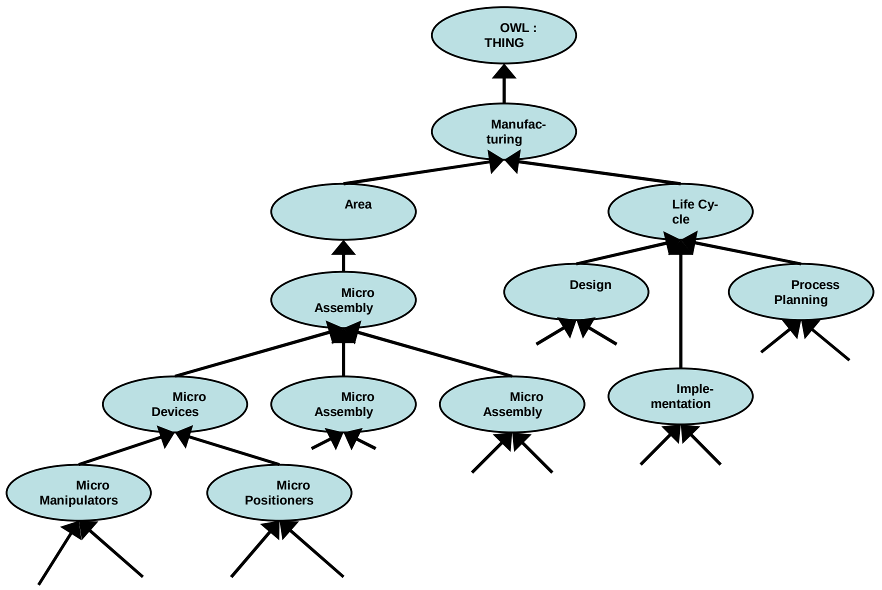


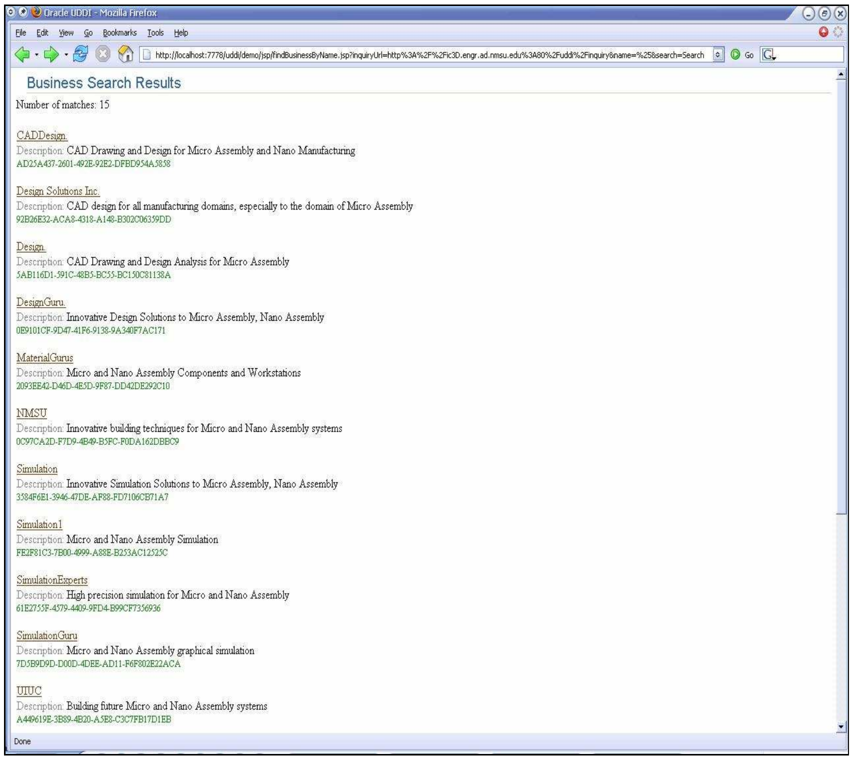
Figure 3. Sample Ontology

5.4 Service Directory Agent

The Service Directory Agent in the collaborative system is used to maintain a service directory where service provider agents will publish their services. This will facilitate other agents in the collaborative system, especially VE agents, to access the available services and use them to process the user agent's input. Oracle UDDI registry is used as the service directory in this collaborative system. Oracle UDDI registry comes along with the Oracle Application Server 10g. In this UDDI registry, instead of saving normal WSDL descriptions for services, OWL-S descriptions of services are saved. Requests from other agents for available services in the UDDI registry are made through this service directory agent. A service directory plug-in is developed for the agent to accomplish this task. The plug-in is developed with methods to connect to the service directory, publish OWL-S services in the service directory and inquire for available services. A screen shot of oracle UDDI registry is shown with some sample services is shown in **Figure** 4.

5.5 Service Provider Agent

Real business services in the collaborative system are provided by the service provider agents. Services provided by these agents range from software resources to actual physical implementation. Along with describing the service capabilities, the configurations of actual physical implementations are also described using OWL. A sample OWL description of a physical work cell can be accessed at http://128.123.245.156:9090/ontology/Device.owl


Oracle UDDI - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
Go
Done

http://localhost:7778/uddi/demo/jsp/findBusinessByName.jsp?inquiryUrl=http%3A%2F%2Fic3D.engr.ad.nmsu.edu%3A80%2Fuddi%2Finquiry&name=%25&search=Search

Business Search Results

Number of matches: 15

CADDesign.
Description: CAD Drawing and Design for Micro Assembly and Nano Manufacturing
AD25A437-2601-492E-92E2-DFBD954A5858

Design Solutions Inc.
Description: CAD design for all manufacturing domains, especially to the domain of Micro Assembly
92B26E32-ACA8-4318-A148-B302C06359DD

Design.
Description: CAD Drawing and Design Analysis for Micro Assembly
5AB116D1-591C-48B5-BC55-BC150C81138A

DesignGuru.
Description: Innovative Design Solutions to Micro Assembly, Nano Assembly
0E9101CF-9D47-41F6-9138-9A340F7AC171

MaterialGurus
Description: Micro and Nano Assembly Components and Workstations
2093EE42-D46D-4E5D-9F87-DD42DE292C10

NMSU
Description: Innovative building techniques for Micro and Nano Assembly systems
0C97CA2D-F7D9-4B49-B5FC-F0DA162DBBC9

Simulation
Description: Innovative Simulation Solutions to Micro Assembly, Nano Assembly
3584F6E1-3946-47DE-AF88-FD7106CB71A7

Simulation1
Description: Micro and Nano Assembly Simulation
FE2F81C3-7B00-4999-A88E-B253AC12525C

SimulationExperts
Description: High precision simulation for Micro and Nano Assembly
61E2755F-4579-4409-9FD4-B99CF7356936

SimulationGuru
Description: Micro and Nano Assembly graphical simulation
7D5B9D9D-D00D-4DEE-AD11-F6F802E22ACA

UIUC
Description: Building future Micro and Nano Assembly systems
A449619E-3B89-4B20-A5E8-C3C7FB17D1EB

Figure 4. Oracle UDDI registry showing sample services for the collaborative system

This allows the Virtual Enterprise agent to know more about the actual hardware implementation of devices. The collaborative system contains multiple service providers who will serve the needs of a user agent. Publication of services by these agents is accomplished through the service directory plugin, which provides methods for publishing the services into the UDDI registry.

6 Example Scenario

In this section, an example scenario is provided from the Micro Assembly domain to the collaborative system. Micro Assembly is considered as an alternative to MEMS based product development, where it is difficult to manufacture a product with different parts having varying properties. As explained in previous sections, it is completely a new area of product development where business vendors have limited number of sophisticated infrastructures and resources to accomplish a complete micro assembly based product development. In this application scenario, a user agent wants to assemble various micron sized parts (for eg. cams) on micron sized pins. Here, the goals of user agent are identification and formation of partnerships with potential business vendors and execution of their associated services.

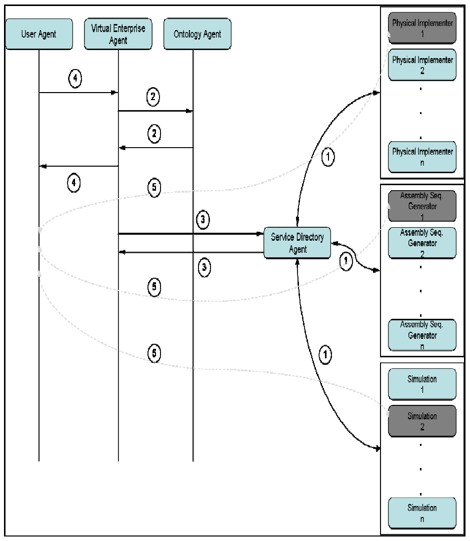


Figure 5. Interactions among the agents in the collaborative system

Possible interactions that will happen in this collaborative framework are listed below (refer to figure 5) and are elaborated subsequently.

1. Interactions between Service Directory Agent and Service Provider Agents.
2. Interactions between Virtual Enterprise Agent and Ontology Agent.
3. Interactions between Virtual Enterprise Agent and Service Directory Agent.
4. Interactions between User Agent and Virtual Enterprise Agent.
5. Interactions between Service Provider Agents and User Agent.

6.1 Service Directory Agent ←→ Service Provider Agents

To demonstrate this interaction, a set of service provider agents have been designed and implemented. These include service directory agents capable of providing

1. Services based on software applications such as assembly sequence generators, 3D path planners and virtual prototyping and analysis Environments
2. Services based on actual physical resources such as micro assembly work cells.

A brief description of some of these resources is provided along with their OWL and OWL-S descriptions.

In order to assemble micron sized parts on micron sized pins, two micro assembly work cells as shown in figure 6, having different assembling capabilities are designed and developed. An ontology is developed to describe the capabilities in terms of work cell specifications. For example, work cell 1 is developed with gripper having the capability of assembling pins and cams in the size range of 100 – 200 microns (diameter) and a few millimeters in length. Due to the page limit, all OWL descriptions and grounding files necessary for the operation of the example are omitted. They are accessible from http://web.nmsu.edu/~gobinath/file.htm.

The maximum and minimum gripping force exerted by the gripper on its target object and its operating conditions are also described by an OWL element. The type of parts that the gripper can handle is given by the following OWL element

<parts_it_handle rdf:resource="#Cams"/>
<parts_it_handle rdf:resource="#Pins"/>

Similar to first micro assembly work cell, the second micro assembly work cell with tweezers is also described using OWL. This can be accessed at the URL http://128.123.245.156:9090/mawc2.owl. **Figure** 6 display two work cells used in our experiment.

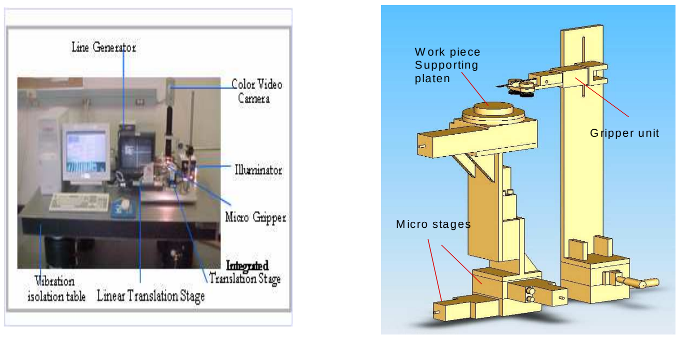


Figure 6. Micro Assembly Work Cells (Left: Work Cell 1, Right: Work Cell 2)

The assembly services of these two micro assembly work cells are made available as web services. As the assembly service requires physical components (cams and pins in this case) to be assembled, a software validation program is developed to validate the dimensions of input components with the capability of the respective micro assembly work cell. For example, in micro assembly work cell 1, the validation program validates the input by comparing the dimensions of the gripper and the parts to be assembled. If the validation program returns the positive results, further steps will be taken to ship the parts to the respective work cell location. This validation program is also made available as web services whose grounding information in OWL-S format is given in the above mentioned URL.

Apart from the work cells, virtual prototyping environments have been developed which form part of the VE resources. **Figure** 7 provides a snapshot of two virtual environments, which can be used to study alternates assembly and path plans, etc.

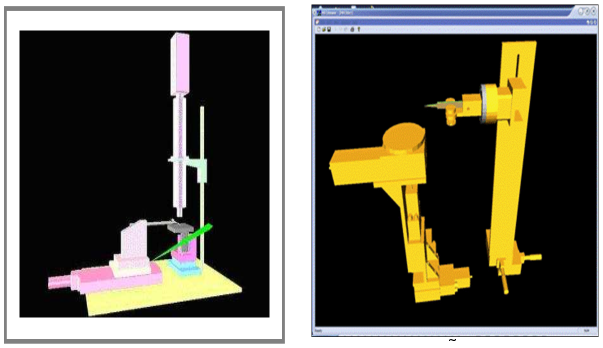


Figure 7. Different Virtual Environments (Virtual Environment 1, Virtual Environment 2)

These virtual environments are also accessible via web services. Service grounding information for one of the these VEs is described in OWL-S format and is available at http://web.nmsu.edu/~gobinath/file.htm.

Some of the software resources within the collaborative framework include micro assembly sequence generators as well as 3D path planners. Grounding information for one of the micro assembly sequence generators (determining an optimal sequence of assembling a target set of micro parts) using Genetic Algorithm is detailed below in 3APL format.

Sample message transfers that will take place during the interaction between a service provider agent (say, Micro Assembly Work Cell Provider) and the service directory (SD) agent while publishing a service are listed below:

Send(SD_Agent, inform, publish ()),
Send(SD_Agent, inform, serviceName (Micro Assembly Work Cell)),
Send(SD_Agent, inform,
    serviceDescription (http://128.123.245.156:9090/ontology/Implementer.owl))
Send(SD_Agent, inform, requires (path planning))
Send(SD_Agent, inform, requires (simulation))

After receiving these messages from the service provider agent, service directory agent publishes the service in the Oracle UDDI registry.

6.2 User Agent ←→ Virtual Enterprise Agent

In this interaction, the user agent sends the input requirements to the virtual enterprise agent. Below are some sample input requirements to the VE agent:

Send (VE_Agent, inform, domain (Micro_Assembly)),
Send (VE_Agent, inform, input ()),
Send (VE_Agent, inform, radius (pin1, 0.5)),
Send (VE_Agent, inform, radius (pin2, 0.5)),
Send (VE_Agent, inform, radius (pin3, 0.5)),
Send (VE_Agent, inform, radius (cam1, 0.6)),
Send (VE_Agent, inform, radius (cam2, 0.6)),
Send (VE_Agent, inform, radius (cam3, 0.6)),
Send (VE_Agent, inform, goal ()),
Send (VE_Agent, inform, on (cam1, pin1)),
Send (VE_Agent, inform, on (cam2, pin2)),
Send (VE_Agent, inform, on (cam3, pin3))

This sequence of message states that the user would like to assemble three pins (*pin1*, *pin2*, *pin3*) of radius 0.5 into three cams of radius 0.6 by placing *pin1* on *cam1*, *pin2* on *cam2*, and *pin3* on *cam3*.

6.3 Virtual Enterprise Agent ←→ Ontology Agent

For the VE agent to process users' request, it needs to create a plan for doing it and who can provide the necessary services required to execute this plan. This information is available in the meta-information managed by the Ontology agent. The VE agent first queries the Ontology agent for meta-information about the services available in the system and devises a plan to achieve the goals of the users (as done in [10]).

In our experimental scenario, ontology for the Micro Assembly domain is developed and deployed in a Tomcat Application Server (refer to **Figures** 2 and 3). Some sample 3APL messages for this interaction are given below

Send (Ontology_Agent, inform, queryForMeta (Micro_Assembly))
Send (Ontology_Agent, inform, whatis (pin1))
Send (Ontology_Agent, inform, whatis (cam1))

Once the Ontology Agent receives the input from the VE agent, the Ontology Agent processes the input to find the corresponding ontology (in this case the ontology of Micro Assembly domain) and queries the ontology to find possible relationships between the input and the concepts it contained using the ontology plug-in. For sample input messages from VE agent, the ontology agent responds by sending the following messages,

Send (VE_Agent, inform, metaInfo (Micro_Assembly))
Send (VE_Agent, inform, steps ())
Send (VE_Agent, inform, physical_implementation ())
Send (VE_Agent, inform, planning ())
Send (VE_Agent, inform, simulation ())
Send (VE_Agent, inform, isObject (pin1, true))
Send (VE_ Agent, inform, isObject (cam1, true))

6.3 Virtual Enterprise Agent ←→ Service Directory Agent

With the meta information and the original input, the VE agent now requests the service directory agent for service providers. The sample messages of this interaction are given below.

Send (SD_Agent, inform, serviceProviderfor (physical_implementation))
Send (SD_Agent, inform, serviceProviderfor (planning))
Send (SD_Agent, inform, serviceProviderfor (simulation))

After receiving these messages, the service directory agent searches the UDDI registry for available service providers. In a UDDI registry, there may be more than one service provider who can serve the user agent's input request. Those service providers are known as potential partners in VE context. From the list of potential service providers, the service directory agent should choose one best service provider for the user agent. Before the selection of a best service provider, the Service directory agent will check for the requirements for each of the potential service providers. The requirements for a service provider may be correct inputs or even some services from other service providers. If all the requirements of a service provider are satisfied and it also satisfies the requirements of user agent, the service directory agent will announce the service provider as best partner. If user agent's requirement does not match with the service providers' requirements, then service directory agent will announce the unavailability of service providers. After finding the service providers, the service directory agent returns the access point URLs of each of the identified business vendors to the VE agent. Message transfers during this interaction are

Send (VE_Agent, inform,
    accessPointURL (http://128.123.245.156:9090/ontology/Implementer.owl)),
Send (VE_Agent, inform,
    accessPointURL (http://128.123.245.156:9090/ontology/planning.owl)),
Send (VE_Agent, inform,
    accessPointURL (http://128.123.245.156:9090/ontology/simulator.owl)),

The resulting access point URLs are then sent to User Agent for execution.

6.5 Service Directory Agent ←→ User Agent

After obtaining the access point URLs of service provider agents, the User agent executes the services available at the service provider sites.

7 Conclusion and Future Work

In this paper, a collaborative system is developed to form a Virtual Enterprise for the domain of Micro Assembly. 3APL language is used to develop the agents which constitute the collaborative system. Ontology for Micro Assembly domain is developed to provide a common ground to share the information contained in it among the agents. Although it is still an ad-hoc development, this prototypical system demonstrates that agent technologies can be very useful in VE development, a rather new area to agent researchers. In the future, we would like to study and develop methodologies for a systematic development of VE in the Micro Assembly domain.

References

1. Bandara, A., Payne, T., Roure, D., Clemo, G., An Ontological Framework for Semantic Description of Devices, *ISWC 2004*, Poster Session, Hiroshima, Japan, 7 - 11 Nov 2004.
2. Borgo, S., P. Leitão, The Role of Foundational Ontologies in Manufacturing Domain Applications, R. Meersman, Z. Tari et al. (eds.) OTM Confederated International Conferences, ODBASE 2004, Ayia Napa, Cyprus, 2004, LNCS 3290, pp. 670-688.
3. Camarinha-Matos, L. M., Asfarmanesh, H., Virtual Enterprise Modeling and Support Infrastructures: Applying Multi-Agent System Approaches in *Multi- agent Systems and Applications*, in LNAI 2086, Springer, July 2001.
4. Hardwick, M., Spooner, D. L., Rando, T., and Morris, K. C. 1996. Sharing manufacturing information in virtual enterprises. *Commun. ACM* 39, 2 (Feb. 1996), 46-54. http://doi.acm.org/10.1145/230798.230803
5. Hindriks, K. V., De Boer, F. S., Van Der Hoek, W., and Meyer, J.-J. Ch. Agent Programming in 3APL, *Autonomous Agents and Multi-Agent Systems, ACM*, 2:4, 357–401, 1999.
6. Genesereth, M. R. and Fikes, R. E. Knowledge Interchange Format (KIF) Version 3.0, *Reference Manual*.
7. Munindar P. Singh. Agent Communication Languages: Rethinking the Principles, *Computer*, vol. 31, no. 12, pp. 40-47, December, 1998.
8. Petersen, S. A., Gruninger, M., An Agent-based Model to Support the Formation of Virtual Enterprises, *Int. ICSC Symposium on Mobile Agents and Multi-Agent in Virtual Organizations and E-Commerce (MAMA '2000)*, in Australia, 11-13 Dec. 2000.
9. Petersen, S. A., Rao, J., Matskin, M., AGORA Multi-agent Architecture for Implementing Virtual Enterprises, *Norsk Informatikkonferanse* NIK2003, Oslo, Norway, 2003.
10. McIlraith, S., Son, T. C., and Zeng, H. Semantic Web Services, *IEEE Intelligent Systems*, vol. 16, no. 2, pp. 46-53, March/April, 2001.
11. The OWL Services Coalition, "OWL-S: Semantic Markup for Web Services", http://www.daml.org/services/owl-s/1.0/owl-s.html.
12. Wilbur, S., Computer Support for Co-operative Teams: Applications in Concurrent Engineering, *IEEE Colloqium on Current Development in Concurrent Engineering Methodologies and Tools*, June 1994.
13. M. Grüninger and C. Menzel. The Process Specification Language (PSL) Theory and Applications, *AAAI Magazi*, 63-74, *Fall 2000*.