Distributed Agents Architecture Applied in Assembly Domain

Munir Merdan; Gottfried Koppensteiner; Alois Zoitl & Bernard Favre-Bulle

Automation and Control Institute
Vienna University of Technology, Austria
merdan@acin.tuwien.ac.at

Abstract

The need for agility of production structures is continuously growing due to increasing complexity of products and decrease of product life cycle. This paper proposes a distributed knowledge-based architecture based on a multi-agent paradigm applied in the assembly domain. The proposed approach is modeled by a society of cognitive agents, which control a set of heterogeneous resources and cooperate together in order to achieve their own aims as well as the system's aim. The assembly knowledge is encapsulated in a rule-based system, which allows an efficient generation of assembly steps. Ontologies are semantic tools that provide a framework to represent this knowledge. The ontology is shared among agents and serves as the instrument to define the vocabulary used by the agents during their interactions.

Keywords: Multi-agent System, Assembly, Ontology, Knowledge-based Architecture

1 Introduction

Current manufacturing systems are faced with the growth in the variety of products and at the same time with a decreasing product life cycle. This increases the complexity of the manufacturing and is especially presented in assembly domain, which involves a lot of manual work and is for a major part of the manufacturing cost responsible [26]. There has been a clear recognition of the need for agile, knowledge intensive assembly systems that can easily absorb the required changes in product volumes, variety and manufacturing organization [1]. Because of their rigid structure and limited adaptation capabilities centralized hierarchical architectures of conventional manufacturing systems can't dynamically manage the high degree of complexity and change. Few approaches, Holonic [2], Multiagent [3] and Bionic architecture [4] are proposed as promising paradigms to handle the combinatorial complexity of manufacturing systems using distributed autonomous decision building blocks (holons, agents and cells).

In this paper, using the Multi-agent paradigm in combination with ontology-driven solutions, we present a distributed architecture in which sets of agents collaborate in order to achieve their own as well as the system goal. The proposed architecture offers the direction towards solving the interoperability problems among heterogeneous enterprise levels. We use the assembly domain as a test field.

The paper is structured as follows. The multi-agent architecture is described in the second section. The third section gives an overview on the ontology. The system architecture is elaborated in the forth section. The fifth section describes the workflow. Finally, the section six gives the conclusion and outlines the future work.

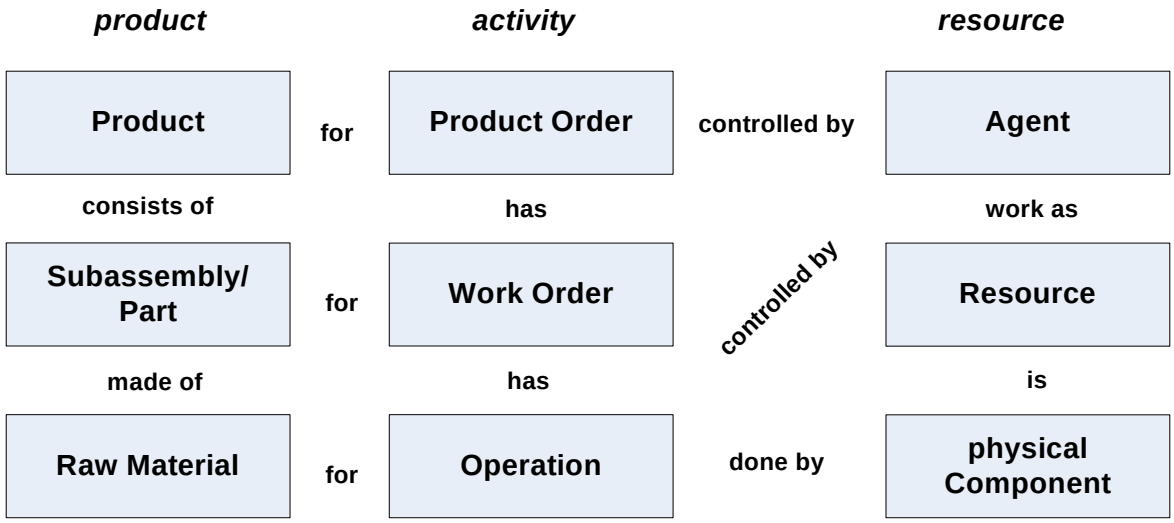
2 Multi-agent system

A multi-agent system is a society of autonomous agents, where each agent acts based on its enclosed behaviors, which represent adequate agent's responds to different inputs from the manufacturing environment [6,7,8]. In a multi-agent system, a number of heterogeneous agents work independently or in a cooperative and interactive manner to solve problems in a decentralized environment [5]. Agent technology provides a good framework for integration of knowledge within a production environment. In our multi-agent system, each agent is an autonomous semantic entity responsible for the commitment of the local data described in its knowledge base. Two different approaches for agent encapsulation in agent-based manufacturing systems are known: the functional decomposition approach and the physical decomposition

approach [7]. In the functional decomposition approach, agents are used to encapsulate modules assigned to functions for order, supply, etc. In the physical decomposition approach, agents are used to represent entities in the physical world, such as robot, conveyor, pallet, etc. Each agent has individual behavior and the capacity to make its own local decisions. The agent has knowledge about his domain of application, about strategies, which can be used to achieve a specific goal, and knowledge about (other) agents involved in the system. The crucial element in the decision component is the rule-based system, which applies declarative knowledge, expressed in a set of rules, to regulate the agent's behavior. Agents communicate and negotiate with each other in order to perform the operations based on the available local information or to solve possible conflicts. Inter-agent communication capability provides the essential means for agent collaboration. In order to ensure the correct understanding of the exchanged messages, agents must have the same presentation of the environment, or at least that part of the shared environment about which they are exchanging information with each other. Ontologies are from vital importance for enabling knowledge interoperations between agents and at the same time for a fluent flow of different data from different entities. Ontology is defined as an explicit specification of conceptualization [9]. Explicit specification of conceptualization means that ontology is a description of the concepts and relationships that can exist within a multi-agent system. Ontologies define a formal semantic for information and provide a shared understanding of a domain, an understanding that can be communicated within application systems [27].

3 Ontology

A mechanical assembly is a composition of parts interconnected forming a stable unit. Parts are defined as components, described by set of attributes, properties, constrains and relations to other parts. A subassembly is a non-empty subset of parts that either has only one component or every part has at least one surface contact with another part in the subset [10]. Numerous ontologies covering a wide range of domains have been developed and presented [11]. As a basis for our ontology we take the "Machine Shop Information Model" [12] developed at the National Institute of Standards and Technology (NIST) as a part of efforts that support the development of standard data interfaces. Our ontology is also heavily influenced by the OZONE ontology [13], the Enterprise Ontology [14] and ADACOR ontology [15]. The benefit of using an existing ontology is that the system is interoperable, in terms of passing data, with any other system that uses the same data model. We use the ontology to link product designs, assembly planning processes and required assembly equipment together. Our proposal is built on three basic layers: product, activity and resources (Figure 1). There is roughly correlation between our ontology and FABMAS ontology [16] as well as MASON ontology [17], which are also based on three layers.



A *product* is presented as hierarchy of subassemblies and parts together with all their properties and relationship between them. This relationship between parts represents how these parts should be assembled to complete the product.
An *activity* is the basic action, which specifies how the world is changed. An *activity* describes how the product is going to be produced and how its production relates to all other entities in production environment.
Resources are represented as entities able to perform a certain activity.
Our concept split up a product order activity into sets of work orders, each work order being described as a list of operations. In order to simplify the generation operation of the process plan, we adopt a top-down function model of the product [18]. Each product type is described through its own assembly process plan.
Assembly steps are presented as levels, what means that parts from the lower level will be assembled first (Figure 2).

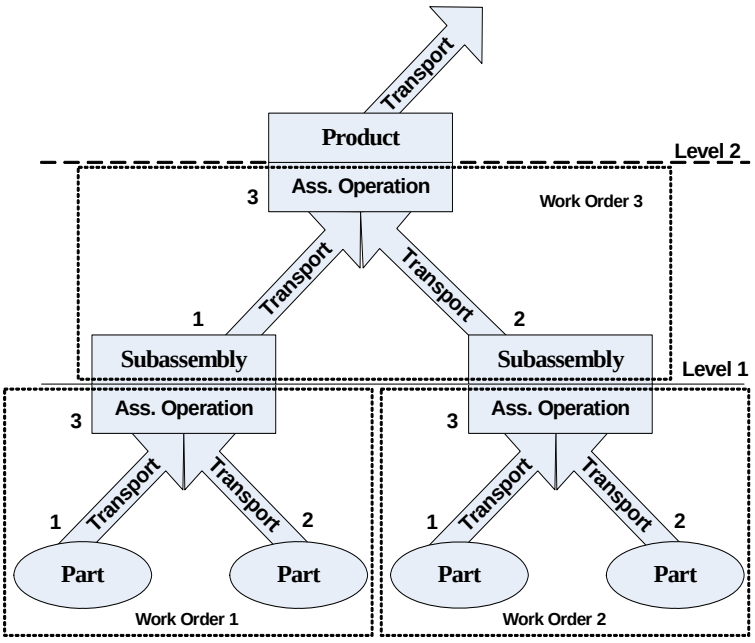


Figure 2. Assembly process plan

There is also a hierarchy within the same level, where it's distinguished which part should be transported and assembled before others. An assembly process plan further contains work orders (steps), which are defined as a set of operations. In our framework, we distinguish two types of operations: part transportation and part assembly.

We assume that each assembly operation would be performed by another resource what means that between two assembly operations it will be necessary to transport a part to another destination. In the case that two sequenced assembly operations could be performed by the same machine, the transport operation will be automatically deleted by the rule-based system.
We use Protégé-2000 [19] as an integrated software tool used by system developers and domain experts to develop the knowledge base.

4 System Architecture

In order to appropriately distribute the functionality and intelligence of the system into specific agents, we decided to generate specific agents based on the activities that could be done within the system (Figure 3). Our Testbed architecture consists of an automatic storage system with a handling unit for the extraction of the parts, a pallet transfer system with redundant paths, as well as a portal robot for the final assembly. The composite parts of the pallet transfer system are: switch units, index units that fix the pallet in a defined position for the handling units, identification units (RFID) for identification of passing pallet units and conveyors. Each unit is controlled by its own machine agent. The communication framework and the strategy model have been built on top of Java Agent Development Environment (JADE) framework [24]. The Contact Agent is created at the start-up of the JADE system and it is always active. Its main responsibilities are to receive a product order and create the appropriate order and supply agents. This agent also creates a machine agent in Protege and one JADE agent for each resource in the system.
The order agent is launched by the contact agent, which means a new order is received. The order agent is responsible for controlling and guiding a single order through the shop floor and for the shipment of finished order to the customer. The order agent queries the ontology for the production plan of the ordered product. From this plan it

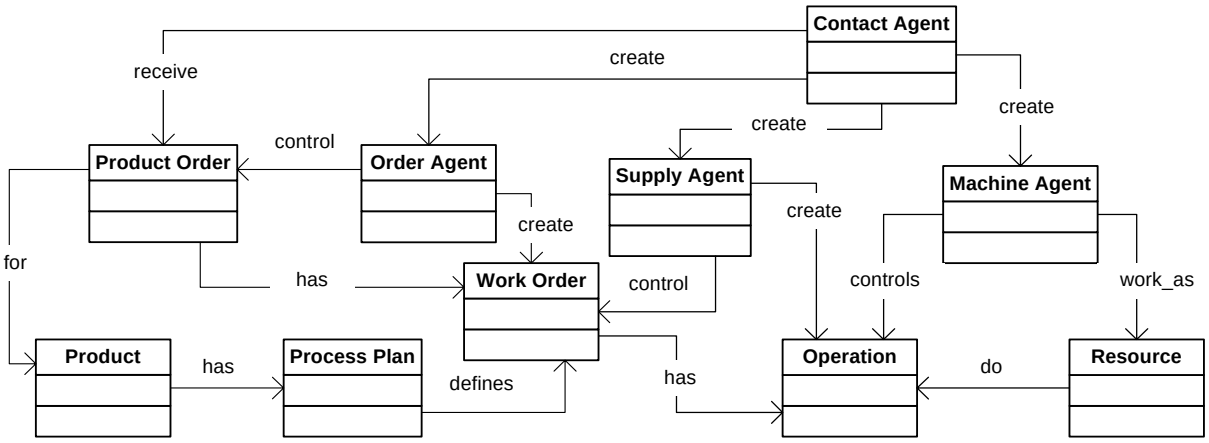


Figure 3. The main agents classes of the system and their activities

generates work orders and one for these work orders responsible supply agent. The supply agent splits each work order on specific operations. Suitable resources, controlled by machine agent, perform these operations. The control of the resources and negotiation with supply agent about operation allocation are the main responsibilities of the machine agent.

4.1 Agent Behavior

Our architecture is based on agents that have a rule-based behavior. Rules are considered as if-then statements applied to the knowledge base. The knowledge base is constantly updated with new facts. If all defined conditions for a particular rule on the left hand side of the rule are satisfied the rule's actions on the right hand side will be executed. This will again add new facts in the knowledge base and set conditions for firing of new rules.
For an easer handling of rules and having a better overview over created rules, each rule is assigned to an appropriate agent.
The simplified rule where a supply agent checks, if there are enough available parts for an order product in storage, is presented bellow. Once an order is issued, the order agent sends the message "Check_Parts" to the supply agent. It will check if there are enough available parts in storage and based on the results undertake appropriate actions.

```
(defrule SA_checks_available_Parts
(object (is-a Supply_Agent) (OBJECT ?sa) (re-
ceive ?message))
(object (is-a Message)(OBJECT ?message)(content
"check parts"))
(object (is-a Order_Agent)(OBJECT ?orderAgent)
   (responsible_for ?productOrder))
(object (is-a Product_Order)(OBJECT ?productOrder)
   (quantity ?q)(product ?product))
(object (is-a Product)(OBJECT ?product)(parts
$?part))
          =>
(foreach ?part $?pl (bind ?pn (slot-get ?part name))
(bind ?x (count-query-results find_Part ?part))
(if (< ?x ?q) then
  (printout t "Not enough Parts of " ?pn " available" )
  else (send_ message
       (make-instance of Message
            (performative "INFORM")
            (sender ?sa)
            (receiver  ?orderAgent)
 (content "parts available")))))
```

The reasoning is implemented using the Jess expert system shell [20]. Jess is essentially a Java reimplementation of a subset of the earlier CLIPS shell [21]. Jess is a simple but powerful tool, used for building a number of industrial Expert System applications [22]. We chose the Jess role engine because of its active development and support, its easy interaction with Java programs, its powerful scripting language, and its expressiveness. JessTab [23] is used as a plug-in for Protégé that allows us to use Jess and Protégé together. JessTab provides a Jess console window where it's possible to interact with Jess while running Protégé.

4.2 Communication

The communication between the agents is done by sending messages. The JADE architecture enables agent communication through message exchange based on agent communication language (ACL) [25]. A message has two sections: message header and message content. The header

contains the information regarding the sender, receiver(s), subject, date, and time that the message is sent by the sender, date and time that the message is received by the receiver, and the priority. The message content contains information regarding the intent. There are two types of content: illocutionary and perlocutionary [34]. The illocutionary message is used to inform other agents, such as the information about the action that has been accomplished. The perlocutionary message is used to request actions from other agents, such as a call for proposal when the resource is available and would like to do something. An incoming message is asserted into the knowledge base as an incoming message fact.

5 Workflow

When a product order comes into the system, it is received by the contact agent. The product order contains information about the costumer, the product, the quantity and the due date. The contact agent sets the status of the order and issues the system ID that will escort the order and will be contained in the names of correlated order and supply agent as well as work orders and operations. After this, the supply and order agent are being created. The created order agent checks if the due date of the order is achievable, calculates a priority for the product order, and sends the message "check parts" to the supply agent. When it receives this message, the supply agent checks if all parts for the orders are available and reserves these parts, otherwise the order must be pending and waits for the parts arriving. After the order agent receives the message that all parts are available, it uses the process plan to decompose the assembly process of the product in work orders and informs the supply agent to take control over them. The supply agent decomposes each work order into operations and starts negotiation with available resources about the completion of these operations. As said before, each of the work orders consists at least one transport operation and one assembly operation. Operations have to be accomplished by different resources located at different places at the shop floor. Each resource possesses its own schedule and its own capabilities to perform different operations. The schedule is not calculated in advance, but arises from the concurrent and contingent interactions. In order to achieve optimal utilization agents negotiate with each other. For example, the assembly operation will be allocated to the resource with the smallest utilization. For the same reason, as soon as pallets are free it will send the message “free” to all supply agents looking for available transport operation. When there are more operations from different product orders, which are competing for the resource, the one with highest priority will be performed. Since the agents in our architecture are self-centered and suffer from the lack of global perspective, it is hart to predict the performance of the whole system. In order to achieve best possible scheduling, there is still more work to be done on optimization of the rules that govern the agent behavior. Further research and experiments are needed to extend the current work and to address its possible shortcomings.

6 Conclusion

In this paper we presented a multi-agent system that uses the ontology driven solution in combination with intelligent agents in order to solve the interoperability problems within the manufacturing life cycle. This system has been applied in an assembly domain. The main advantage of our concept is that in this knowledge-based system it does not have to be said, how a problem has to be solved, but the problem and the goal have to be described. Further advantage of this type of knowledge-based system is related to the simple and very comprehensive way to represent the reasoning capability of one agent. There is also no need to explicitly program the interactions of the whole system since this emerges as a by-product of the individual goals and capabilities of the constituent agents.
The work presented represents only the first step of our effort toward achieving knowledge-based agent system in an assembly domain. The future work will be conducted to integration of this system with our transport system and experiments in the real world. Furthermore, it will be also necessary to design appropriate scheduling algorithms to face the dynamic scheduling problems of modern manufacturing systems.

Acknowledgment

The authors would like to acknowledge the financial support from the Austrian Science Fund (FWF), Projects No. FWF-P19644-N13.

References

[1] Heilala J, & Volvo P, "Modular Reconfigurable Flexible Final Assembly Systems In Electronic Industry", Assembly Automation Workshop, Netherlands, 11'-12" May 2000

[2] Brussel H. V., Wyns J., Valckernaers P., and Bongaerts L., "Reference architecture for holonic manufacturing systems: PROSA," *Comput. Ind*., vol. 37, pp. 255–274, 1998.

[3] Jennings N. R. and Wooldridge M., "Applications of intelligent agents," in *Agent Technology*, N. R. Jennings and M. J. Wooldridge, Eds: Springer, 1998, pp. 3–28.

[4] Okino N., Bionic Manufacturing System-*Flexible Manufacturing Systems Past-Present-Future*, 1993, pages 73–95.

[5] Brenner, W.; Zarnekow, R.; Wittig, H.: Intelligent Software Agents, Springer Verlag, Heidelberg, 1998

[6] Parunak H.V.D.: Practical and Industrial Applications of Agent-Based Systems, Environmental Research Institute of Michigan (ERIM), 1998.

[7] Shen W., Norrie D.H.: Agent-based Systems for Intelligent Manufacturing : A State-of-the-Art Survey, *Knowledge and Information Systems, an International Journal*, Vol.1, No.2, 1999, p. 129-156.

[8] Parunak H.V.D. Manufacturing experience with contract net, distributed artificial intelligence. London: Pitman; 1987. p. 285–310.

[9] Gruber T. R., „A translation approach to portable Ontologies" *Knowledge Acquisition*, 1993

[10] Homem de Mello L.S. and Sanderson A.C., "Representations of Assembly Sequences" 1 l* IJCAI, 1989, pp. 1035-1040.

[11] http://www.daml.org/ontologies/ Accessed January 2006.

[12] McLean, C., Lee Y. T., Shao G. and F. Riddick. 2005. Shop Data Model and Interface Specification, NISTIR 7198. National Institute of Standards and Technology, Gaithersburg, MD.

[13] Smith S. and Becker M., An Ontology for Constructing Scheduling Systems; Working Notes of 1997 AAAI Symposium on Ontological Engineering, AAAI Press, March, 1997

[14] Uschold M., King M., Moralee S. and. Zorgios Y (1998) The Enterprise Ontology The Knowledge Engineering Review , Vol. 13, *Special Issue on Putting Ontologies to Use* (eds. Mike Uschold and Austin Tate).

[15] Leitão P., An Agile and Adaptive Holonic Architecture for Manufacturing Control, doctoral dissertation, Dept. Electrical and Computer Eng., Univ. of Porto, Portugal, 2004.

[16] Mönch L., Stehli M., „ManufAg: a multi-agent-system framework for production control of complex manufacturing systems" Springer-Verlag 2005

[17] Lemaignan, S. Siadat, A. Dantan, J.-Y. Seme-nenko, A. „MASON: A Proposal For An Ontology Of Manufacturing Domain" IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications (DIS'06) pp. 195-200

[18] Zhu D., Zhao L., Zhang J.. "A Method of Generating Assembly Plan Based on Level Hierarchy Connection Relation Model", *Journal of East China Shipbuilding Institute.*14(1): 71-75, Jan 2000.

[19] Stanford Medical Informatics, Stanford University. Prot´eg´e Website. http:// protege.stanford.edu. Accessed December 2006.

[20] Sandia National Laboratories, Jess: the Rule Engine for the JavaTM Platform. Available at: http:liherzberg.ca.sand~agovljessi, last visited March 2007.

[21] CLIPS, a tool for building exprt systems. Available at: http:ilwww.ghg.netlclipsi CLIPS.html, last vis. September 26th 2003.

[22] Friedmm-Hill E. J., "Jess, the expert system shell for the Java Platform, Y. 6.la4, user's manual", available at: hnp:iihenberg.casmdia.govljessi, last visited September 26th 2003.

[23] Eriksson H. The JESSTAB Approach to Protégé and JESS Integration. In Proceedings of the IFIP 17th World Computer Congress - TC12 Stream on Intelligent Information Processing, pages 237–248. Kluwer, B.V., 2002.

[24] JADE - Java Agent Development Framework, http://jade.tilab.com/, Accessed January 2006.

[25] Bellifemine F. L., Caire G., Greenwood D., Developing Multi-Agent Systems with JADE, Wiley & Sons; 2007

[26] Delchambre, A. (1996). CAD Method for Industrial Assembly: Concurrent Design of Products, Equipment, and Control Systems, John Wiley & Sons Ltd.

[27] Fensel, D. Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce, Springer-Verlag 2004.