THE DESIGN OF A JADE COMPLIANT MANUFACTURING ONTOLOGY AND ACCOMPANYING RELATIONAL DATABASE SCHEMA

J. JANSE VAN RENSBURG AND H. VERMAAK

Abstract

To enable meaningful and consistent communication between different software systems in a particular domain (such as manufacturing, law or medicine), a standardised vocabulary and communication language is required by all the systems involved. Concepts in the domain about which the systems want to communicate are formalized in an ontology by establishing the meaning of concepts and creating relationships between them. The inputs to this process in found by analysing the physical domain and its processes. The resulting ontology structure is a computer useable representation of the physical domain about which the systems want to communicate. To enable the long term persistence of the actual data contained in these concepts and the enforcement of various business rules, a sufficiently powerful database system is required. This paper presents the design of a manufacturing ontology and its accompanying relational database schema that will be used in a manufacturing test domain.

Keywords: Ontology, relational database, multi-agent architecture.

1. INTRODUCTION

Communication within and between software platforms and data persistence are two core components of any software engineering project. Although this article is not focussed on the software platform in which the ontology and database will be used, a short introduction to the system architecture is necessary to better understand why the ontology and database are designed as they are.

The software platform described in this project is based on the multi-agent [1] software model. Agents exist within the system and each one is responsible for certain functions. Agents can be thought of as plug-ins in a software system. Agents communicate with one another using a particular language with a well defined vocabulary (ontology).

The software platform chosen to host the agents is called JADE[2]. This is an extension to the very popular JADE [3] agent platform that is written in the JAVA computer programming language. The following agents have been identified as necessary in the system:

- Security Agent. This agent handles low level system functions and security operations
- Database Agent. This agent connects to the database and acts as a proxy to it for other agents
- Device Agent(s). Each manufacturing device in the factory is controlled by its own device agent. This agent will control and monitor the device during manufacturing
- User Interface Agent(s). This agent regulates user interaction with the system
- XMLWebService User Interface Proxy Agent. This agent acts as a web service proxy to User Interface Agents. It presents a web service portal for a user to access the system
- AgentDevice Device Interface Proxy Agent(s). This agent acts as a proxy for agents running on remote platforms that do not share the system ontology, but are part of a production plan as a user device that performs some action during manufacturing
- Production Planner Agent. This agent schedules orders into an overall production plan
- Production Execution Agent. This agent takes a production plan and executes it by creating appropriate device agents and sending each one its part of the overall plan to execute
- Product Agent(s). Each product to be manufactured is represented by this agent. The agent keeps track of what steps the product has been through and where it should go next. Tracking data is also compiled (from such sources as RFID and barcode scanners)

Section 2 provides a look at the manufacturing test domain where the practical application of the system will be. In section 3, an overview of ontology and relational database theory is given. Section 4 describes the design of the ontology and database schema. In section 5, future work in this project is listed.

2. MANUFACTURING TEST DOMAIN

The manufacturing test domain is located in the RGEMS[4] laboratory on the CUT [5] campus. A relatively simple demonstration manufacturing system is available to test and develop various technologies. The overall goals of the demonstration system are to:

- Develop technologies that enable components from various manufacturers to seamlessly work together
- Develop the system to be reconfigurable by using reconfigurable components. The term “reconfigurable” as applied to the system means that the system is capable of quickly changing to enable it to manufacture different, but related products
- Develop technologies that can be applied to small manufacturing plants within South Africa

2.1 Current Configuration

The current configuration of the test domain is composed of three main categories.

2.1.1 Manufacturing Components

Various conveyors, robots, controllers, actuators and sensors are installed on a manufacturing line.

2.1.2 Communication Network

Controller devices (such as PLCs) in the system are accessed through a high speed gigabit Ethernet network.

OPC[6] is an industrial communication protocol using Ethernet as transmission medium and is used to communicate between software applications on computers and OPC compatible devices in a standardised way. Most manufactures of industrial controllers support the OPC protocol. For devices that do not support OPC, converter solutions have been found.

A device on the network's inputs and outputs are represented on an OPC server as tags. A tag has a name, data type, data quality and current data value. Access to the address space by clients is password protected. Authenticated clients can subscribe to tags and receive updated data from the device when new data is available on the device (e.g. a proximity sensor is triggered). Clients can also write data to a device through OPC (e.g. a tag is set to “1” and the device in turn activates a particular actuator).

Figure 1 shows the OPC address space in the test domain. The input register folder on an Allen-Bradley ML1500 PLC is highlighted.

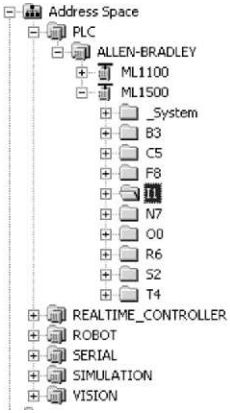

Figure 1: Test Domain OPC Address Space

2.1.3 Information Technology

Servers responsible for database storage, OPC access, Internet access and application development and execution are well established within the laboratory.

2.2 Configuration Expansion

At the time of writing the system is being expanded with various new components. A second KUKA Robotics 6 axes robot is to be installed and the current conveyor system will be expanded with modular Bosch conveyors and transfer units. Several RFID tag readers will be added to the system to track product flows. High quality camera systems will also be installed to monitor product quality and system reconfiguration. A process of making every component OPC visible is well underway. The new hardware additions to the system will allow for a more complicated manufacturing path.

2.2.1 Current Development Projects

A number of student projects are ongoing to achieve the system goals:

- Projects to develop machine vision quality control systems as well as the development of reconfigurable assembly cells are ongoing
- The project of the author of this article is to develop a software platform capable of controlling and monitoring the manufacturing process using a multi-agent software architecture
- A web-based remote monitoring system is under development. This project will allow remote monitoring and partial control over the manufacturing process within the lab by interacting with the manufacturing process control platform. The remote monitoring system is being designed from the outset to integrate into the current RGEMS SharePoint web platform

2.3 Domain Observations

Taking the domain into account the following observations can be made concerning the design of the ontology and database:

- Both designs need to be flexible due to the reconfigurable nature of the test domain and the wide variety of possible configurations within the domain
- Support for OPC devices and humans as devices in the manufacturing line is required
- Security is very important considering the fact that potentially dangerous industrial devices can be controlled through the system

3. ONTOLOGY AND RELATIONAL DATABASE THEORY OVERVIEW

3.1 Ontology

The most widely used definition of an ontology is that it is an explicit specification of a conceptualisation. An ontology does not store information about the particular state of a domain, it describes knowledge about the domain. A knowledge base describes the particular state of a domain in the terms defined within the ontology in use in the domain [7].
In [8] an ontology is defined as: “Pragmatically, a common ontology defines the vocabulary with which queries and assertions are exchanged among agents. Ontological commitments are agreements to use the shared vocabulary in a coherent and consistent manner. The agents sharing a vocabulary need not share a knowledge base; each knows things the other does not, and an agent that commits to an ontology is not required to answer all queries that can be formulated in the shared vocabulary. In short, a commitment to a common ontology is a guarantee of consistency, but not completeness, with respect to queries and assertions using the vocabulary defined in the ontology.”
Various ontologies exist (mostly medical and natural science related) and are represented in different ontology languages. Ontology languages permit various levels of information representation and inference capabilities. Using an appropriate ontology language and inference engine, new knowledge can be inferred from the structure of the ontology and the state of the domain.
Various factors influenced the choice of the selected ontology language and base vocabulary:

- Compatibility with the selected multi-agent platform and existing platforms
- Time needed to develop the ontology in the selected language
- Tools available for design and implementation

The ontology implementation is described in more detail in section 4.1.

3.2 Relational Database

The relational database is a very popular way of persisting large volumes of data in a consistent manner for long periods of time according to certain rules. Software applications interact with relational databases by issuing SQL (Structured Query Language) statements. SQL statements are submitted to the database management system, which manages the retrieval, insertion or updating of data in the actual database. The relational database design is described in more detail in section 4.2.

4. DESIGNS

4.1 Ontology

4.1.1 Ontology Representation Language

The chosen ontology representation language is the built-in Protégé [9] frames based ontology language. Protégé is a widely used graphical ontology engineering tool created by Stanford University.

4.1.2 Basis for the Ontology

The upper-level ontology used as the base of the manufacturing ontology vocabulary is the JADE basic ontology and a Protégé compatible implementation is supplied with the OntologyBeanGenerator[10] plug-in. The JADE basic ontology consists of the following root classes:

- AgentAction: Actions that agents can perform
- Concept: Concepts in the domain
- Predicate: Predicates in the domain

The JADE basic ontology includes the following built-in Predicate classes (that can also be extended) that are used to communicate statements concerning Agent Actions related to Concepts:

- Done
- Equals
- FalseProposition
- Result
- TrueProposition

4.1.3 Ontology Implementation

The ontology is frame based and is constructed using class frames and slots. A class frame can contain any number of slots and these slots can have restrictions (facets) placed on them such as data type and cardinality. Slots can be of any data type present in the domain (including class frames). Class frames inherit identity and slots from their parent class frames, with the root class THING being common to all classes.

The manufacturing ontology is created by extending the JADE basic ontology class frames. Agent Actions and Concepts are mostly based on the agents identified in the multi-agent platform. These agents are in turn based on the domain observations. The main categories include:

- Device
- Security
- Planner
- Product
- Database
- Production Execution

Figure 2 shows the root class frames of the actual ontology design. Only a very limited view of the ontology is shown.

Figure 2: Ontology Root Class Frames

Figure 3 shows the DeviceAction root class frame expanded to show its sub-classes

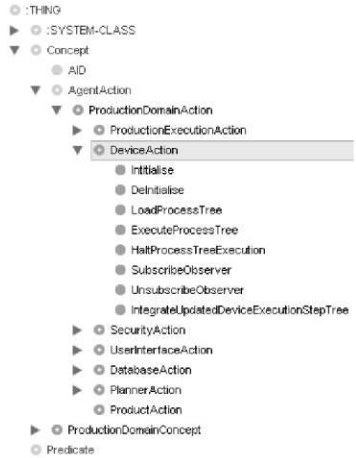

Figure 3: DeviceAction Root Class Frame

Figure 4 shows the DeviceConcept root class frame and its subclasses.

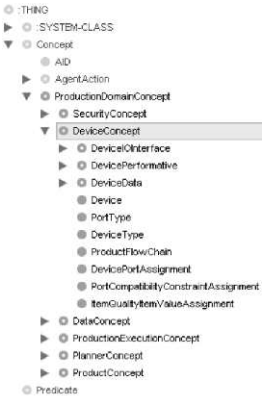

Figure 4: DeviceConcept Root Class Frame

Slots (with their facets) common to all ProductionDomainAction class frames are shown in Figure 5

OperationToBePerformedByUser	Class with superclass Operation
ProductionDomainActionName	String
ProductionDomainConceptOperationAppliesTo	Class with superclass ProductionDomainConcept
ProductionDomainActionDescription	String
ProductionDomainContextConcept	Class with superclass ProductionDomainConcept

Figure 5: Common ProductionDomainAction Slots

4.1.4 JAVA Code Generation

After the ontology is created in Protégé, the OntologyBeanGenerator can be executed to generate the JAVA ontology classes as can be seen in Figure 6.

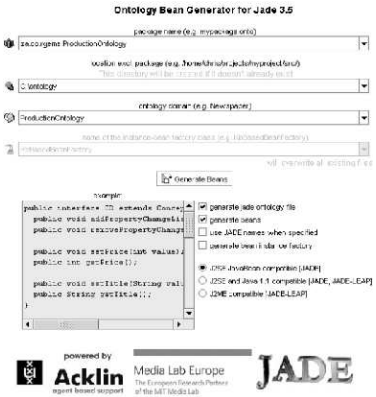

Figure 6: OntologyBeanGenerator Code Generation

4.1.5 Communication between Agents Using the Ontology

All the agents in the system share the same ontology and communicate with each other by sending and receiving populated ontology classes encoded in the FIPA SL content language [11]. The FIPA SL content language is a standardised way to send and receive information within FIPA[12] compliant multi-agent systems. The JAVA classes generated by the OntologyBeanGenerator are compatible with the JADE basic ontology and can therefore be properly encoded into the FIPA SL content language for transmission. Adhering to these standards means that communication to any other FIPA compliant multi-agent platform is theoretically possible.

4.1.6 Security

The security model chosen for the system is based on the widely used Role-Based Access Control (RBAC) [13] model. An implementation of such an RBAC model was adapted for use from[14]. The RBAC model assigns permissions to roles and these roles are then assigned to users in the software system. A role is a collection of permissions given to a user to execute specific actions within the system. Each entity interacting with the system is represented by an internal user. This allows fine grained control over what the interacting entity may do within the system as well as tracking what the entity did. Security is one area where the database and ontology complement each other. Each agent within the system executes as a particular user, for a particular session. This means that for that session the agent is bound by the permissions of its user. The Security Agent in the system handles these assignments and the enforcement of user permissions.

Each action executed within the system is ultimately executed by an agent sending an Agent Action object to another agent. Each Agent Action object in the ontology inherits security slots. These slots are used to uniquely identify the user permission. The user permissions are generated beforehand and stored in the database. Based on the user permissions of the requesting agent, the request can either succeed or fail.

As an example, suppose the Production Execution agent wishes to start the execution of a production order. It prepares a “LoadProcessTree” object (see Figure 3) by filling its “ProductionStepTree” slot (not shown) with the appropriate steps a device is supposed to execute. The security slots seen in Figure 5 are also populated:

- A domain Operation the action entails such as "Execute”, “Delete”, “Create”, “Change” and “Halt”. These operations are the same root operations defined in the relational databases RBAC user operations
- Domain object type on which the action is to be executed (e.g. "Device")
- A Domain object type to provide context to the request (e.g. "Device")

The context object type is only has to be unique if further refinement is needed to specify a permission.

When the Device agent receives the Agent Action object it first queries the Security agent with the following data:

- ID of the agent that sent the request
- Type of Agent Action object sent by the requesting agent
- The security slot data present in the original Agent Action request object

The Security Agent then checks what user the original requesting agent is executing as by checking the currently active session table in the database. It then compares the request against its pre-compiled permission table (created from the relational database at system start-up to increase performance) and responds to the Device agent with a Result Predicate object. Depending on the result, the Device agent will execute the request or send a failure Result predicate object back to the Production Execution agent.

4.2 Relational Database

During the modelling of the relational database, it was found that a recursive relationship between entities (in most cases within the same entity table) in the database design is a very powerful modelling technique and the only way to represent certain domain concepts and their relationships. The database schema consists of about 60 tables and represents the following main domain concepts and relationships. Only the most important concepts and relationships are discussed.

4.2.1 Devices in the Manufacturing Line

- The way in which devices are physically arranged. Many different physical arrangements can be stored to allow for reconfiguration of the device layout in the manufacturing plant
- A hierarchy of devices. Devices may physically or conceptually contain other devices. Arbitrary collections of devices can be created in this way (e.g. Assembly cell, Robot arm with attached camera or Room 1)
- A map specifying how devices are connected to one another creating a path for assemblies to follow through compatible "ports" on devices. A port is a way of specifying a type of physical interaction a device can support (e.g. a gripper). A device may contain many such ports. A port is only compatible with a subset of other ports that are specified by the system engineer. This concept was adapted from [15].
- An abstract device definition with concrete implementations. Currently two core types of devices are implemented: OPC and User. OPC devices are communicated with by OPC tags on OPC servers.

User devices represents humans working in the manufacturing line (e.g. human in manufacturing line with touch-screen device), performing some function in the manufacturing process

4.2.2 Production Plans

- A tree of steps to be performed to manufacture a product. This tree contains actions to be executed on devices and allows for branches of parallel execution of steps
- Each production step has certain conditions that must be true before, during and after it executes. These conditions are built by chaining conditions (a specified data value on a device output combined with a specified data quality for that reading) by using logical operators. In this way the manufacturing process executes by adhering to precisely defined rules

4.2.3 Products (Assemblies)

A flexible method to represent products and how they are constructed from parts by a production plan is integrated into the database. Any product configuration can be represented: a product consisting of only a single part, to a complicated product consisting of many sub-assemblies. To achieve this, a part and product is essentially the same concept called an assembly. A product is the final form of a particular assembly. An assembly can consist of other assemblies. A part assembly is constructed of two or more parts. This part collection forms an assembly that can be connected to another assembly on one of its sub-parts.

4.2.4 Security

As previously mentioned, the security model in the system is an adapted RBAC model. The security schema consists of the following concepts and relationships:

- Users and the User Groups they belong to, Roles assigned to User Groups, Permissions assigned to Roles and Operations assigned to permissions. A Permission's associated Operation is associated with an object from the domain (Concepts in the ontology design) completing the permissions definition
- Agents are defined
- An active session table is maintained that lists the User a particular Agent is executing as
- A boot table is defined that lists what agents to start and as which user the particular agents should execute as
- An audit log table is maintained that logs permission usage by users

5. FUTURE WORK

Future work involves implementing the multi-agent system using the ontology and database implementations. During implementation, minor refinements to the ontology and database schema are expected. The system is then to be tested on the manufacturing components in the test domain.

6. ACKNOWLEDGEMENTS

The work presented in this article is part of the Advanced Manufacturing Technology Strategy (AMTS) [16] project AMTS-07-11-P. The National Research Foundation (NRF), AMTS and CUT Innovation Fund are thanked for their funding and support of this work

7. REFERENCES

http://www.rgems.co.za, last accessed in January 2010

Central University of Technology (CUT), http://www.cut.ac.za, last accessed in January 2010.

About OPC - What is OPC?, http://www.opcfoundation.org/Default.-aspx/01_about/01_whatis.asp?MID=AboutOPC, last accessed in October 2008.

Introduction to Ontologies and Semantic Web, http://www.obitko.com/tutorials/-ontologies-semantic-web/introduction.html, last accessed in January 2010.

T Gruber, "Toward Principles for the Design of Ontologies Used for Knowledge Sharing," International Journal Human-Computer Studies , vol. 43, no. 5-6, pp. 907-928, August 1993.

The Protégé Ontology Editor and Knowledge Acquisition System, http://protege.stanford.edu/, last accessed in December 2008.

OntologyBeanGenerator 4.1, http://protegewiki.stanford.edu/index.php/-OntologyBeanGenerator_4.1, last accessed in June 2009.

FIPA SL Content Language Specification, SC00008I, March 12, 2002.

Foundation for Intelligent Physical Agents, http://www.fipa.org/, last accessed in May 2010.

R Sandhu, D Ferraiolo, and R Kuhn, "The NIST Model for Role-Based Access Control: Towards a Unified Standard," in Proceedings of the fifth ACM workshop on Role-based access control, Berlin, Germany, July 2000, pp. 47-63.

Fine Grained Role Based Access Control (RBAC) system, http://www.sqlrecipes.com/database_design/fine_grained_role_based_access_control_rbac_system-3/, last accessed in January 2010.

N Lohse, H Hirani, and S Ratchev, "Equipment Ontology for Modular Reconfigurable Assembly Systems," International Journal of Flexible Manufacturing Systems, vol. 17, no. 4, pp. 301-314, October 2005.

Advanced Manufacturing Technology Strategy, http://www.amts.co.za/-main.htm, last accessed in September 2010.

National Research Foundation, http://www.nrf.ac.za/, last accessed in September 2010.