USING AGENTS IN THE EXCHANGE OF PRODUCT DATA

Udo Kannengiesser and John S. Gero
Key Centre of Design Computing and Cognition, University of Sydney

Abstract: This paper describes using agents in the exchange of industrial product data when predefined translators are not available. A major problem with standard translators is that a seamless data transfer instantly fails when not every translator implements a mapping into or from the standard format. This is frequently the case for large design projects that involve the use of a multitude of heterogenous tools, possibly in evolving configurations over time. This approach to using agents aims to flexibly provide product models in a form adapted to the need of the particular tools when a common data format is not readily available. Experiments show the feasibility of this approach as well as its efficacy and efficiency.

Key words: product data exchange, product modelling, interoperability

1 INTRODUCTION

There has been an increased use of computational tools to support various tasks in product development. Examples include computer-aided drafting (CAD) and manufacturing (CAM) systems and a number of specialised tools for analyses such as finite element analysis (FEA) and spreadsheet analysis. Most computational systems have been developed independently from one another to address the specific needs of each task and use different product data representations. However, industrial product development is a process that involves a complex network of interrelated activities, each of which needs information produced or manipulated by the other. Interoperability – the ability to move data from one representation of a product to another to allow other computational processes to operate on it

has become an area of growing concern as the cost of such interchanges increases (NIST 1999).

Most approaches to the exchange of product data (today commonly subsumed in the notion of product modelling) are founded on a standard data model that is used to translate between the different native formats of the tools. Any object that needs to be made interoperable must be pre-defined in this model and encoded into a standard form. One of the best-known product models is the ISO 10303 standard, informally known as STEP (STandard for the Exchange of Product model data).

Despite the growing use of STEP some practical issues remain. One of them is that interoperability between a set of tools is solely determined by the intersection of their translation capabilities. As many translators have been specialised to implement only certain subsets of the standard data model, in practice a completely seamless data transfer is often not possible (Pratt 2001). Especially large design projects involving a highly diverse set of tools are affected. This problem is aggravated when technological or organisational changes over the duration of a project necessitate the integration of new tools or new exchanges among present tools. In addition, the pace at which ISO standards are developed and implemented is generally very slow and often lags behind the needs and developments in industrial practice (Eisenberg and Melton 1998).

Updating the set of translators for every modification in the tool environment is time-consuming, costly and not always possible within the time constraints of a project. There is a need for a more flexible approach, one that is able to quickly achieve interoperability when a common product model is not readily available. We have developed an agent-based system that can exchange product data among tools that do not share a common format. Experimental results show that this approach can be an effective and efficient way to provide the needed flexibility in the transfer of product data when pre-defined translators are not available. Our conceptual assumptions draw from research in cognitively-based situated agents.

1. SITUATED AGENTS

A characteristic that allows a system or agent to adapt its behaviour to changes in its environment is situatedness, an idea from cognitive science. A situated agent does not simply react reflexively in its environment but uses its interpretation of its current environment and its knowledge to produce an action (Clancey 1997). As a consequence, a situated agent can be exposed to different environments and produce appropriate responses.

Gero and Fujii (2000) have developed a modular architecture for a situated agent, Figure 1. The agent's sensors monitor the environment to produce sense-data relevant for the agent. The sensors receive biases from the perceptor, which "pulls" the sense-data to produce percepts. Percepts are grounded patterns of invariance over interactive experiences. Perception is driven both by sense-data and biases from the conceptor, which "pulls" the percepts to produce concepts. Concepts are grounded in the percepts as well as possible future interactions with the environment. The hypothesizor identifies mismatches between the current and desired situation and decides on actions that when executed are likely to reduce or eliminate that mismatch. Based on the hypothesized action, the action activator decides on a sequence of operations to be executed on the environment by the effectors.

This architecture provides a framework for different modes of cognition and consequently different degrees of flexibility in the agent's actions: A *reflexive* agent uses the raw sensory input data from the environment to generate a pre-programmed response. A *reactive* agent uses percepts to activate its actions, which can be viewed as a limited form of intelligence constrained by a fixed set of concepts and goals. A *reflective* agent constructs concepts based on its current goals and beliefs and uses them to hypothesize possible desired external states and propose alternate actions that will achieve those desired states through its effectors. The agent's concepts may change as a consequence of its experience.

Situated, reflective agents are especially useful in dynamic environments characterised by high heterogeneity and frequent change. One such environment is the fragmented world of industrial design software.

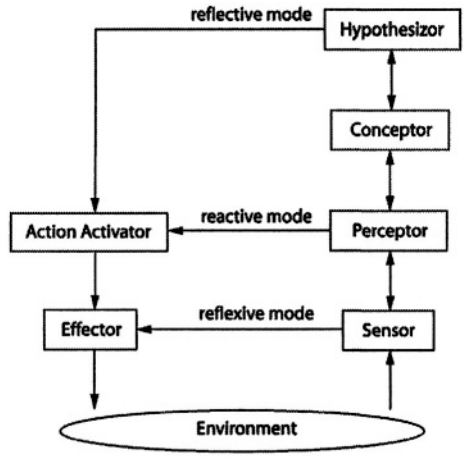

Figure 1. A modular architecture for a situated agent allowing different modes of reasoning.

2. IMPLEMENTATION

We have implemented a system consisting of a situated, reflective agent (which we call the product modelling (PM) agent) and a number of tools, each of which is "wrapped" by a simple, reactive agent, Figure 2. The PM agent maintains all the data of a particular product throughout the different design stages from the initial specification to the released design description. The individual design stages are carried out by the tool agents that modify or add to the product data. The PM agent knows about the tool agents' roles and the current state of the design with respect to a given project plan and accordingly manages all data transfers to and from the tool agents. Cutkosky et al. (1993) have used a similar architecture using agents wrapping design tools and facilitators to exchange their data; however their approach is based on pre-defined standard translators.

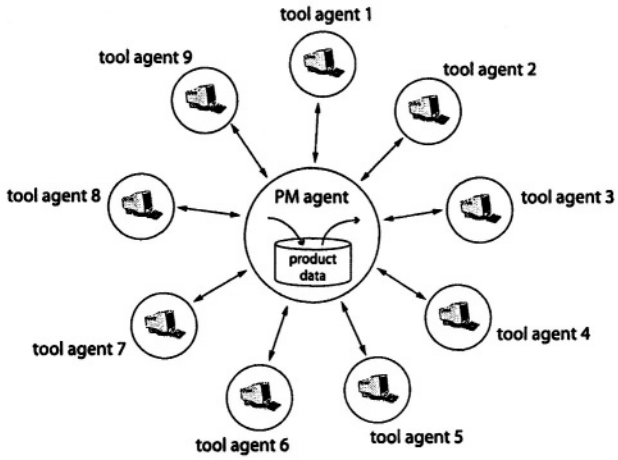

Figure 2. An agent-based system for product data exchange.

All agents are implemented in the rule-base language Jess[1] and connected to their environment by sensors and effectors written in Java. While the tool agents, here, are rather simple consisting of perceptors and action activators, the PM agent has all the modules of a situated, reflective agent, including conceptor and hypothesizor. In addition, it has a neural network (in Java) to represent its memories about its interactions with the tool agents, which it uses to bias the way it interacts with its environment. The neural network is an interactive activation and competition (IAC) network that has the same

[1] Java Expert System Shell (http://herzberg.ca.sandia.gov/jess/)

architecture as the one proposed by McClelland (1981) and includes an unsupervised learning algorithm. In addition, we have given the agent the capability to add new neurons to the IAC network as well as to reorganise the connections among the neurons to update its memory according to its current interpretation of its interactions. This allows the agent to integrate new experiences such as previously unknown formats or new associations between the formats and the tool agents (in case a tool agent replaces its tool by one that uses a different format). The PM agent can construct new formats from interpreting messages represented in unfamiliar formats using a set of generative rules and semantic knowledge.

The messages exchanged among the agents are structured according to specifications developed by the Foundation for Intelligent Physical Agents (FIPA 2004). There are two types of messages: strings representing product models, and propositions or requests to clarify their meanings when an agent fails to understand a product model. The latter type is based on synonyms and hypernyms (super-names) as a means to explain unknown data. This has been inspired by the conceptual foundations of WordNet (Miller 1995). Figure 3 shows all possible agent interactions using the AUML[2] notation.

Failure of the PM agent to provide a tool agent with product data that is represented in the correct format necessitates help from the human user who then has to translate the data manually. Similarly, if the PM agent fails to understand the product data produced by a tool agent, human intervention is required to fill the data into the PM agent's product database. We will use the number of times that human intervention occurs during the design project as an indicator for our system's efficacy.

As the PM agent can learn from its interactions with the tool agents, we expect it to transfer the product data with less and less effort among the same set of tool agents. As a result, there will be fewer messages in the system dealing with expressing lack of understanding or clarifying meanings. We will refer to this type of messages using the linguistic term *repair*. The number of times that repair occurs during the design project will serve as an indicator for our system's efficiency.

Figure 4 shows a possible result of the PM agent learning (parts of) formats and thus increasing interoperability through its interactions. Commencing with a limited amount of knowledge that is sufficient to allow interoperability with some tool agents (tools T3 and T6), we expect the PM agent to construct new knowledge over time to exchange product data with other tool agents, either completely automated (tools T1, T2 and T4) or semi-automated (tool T5). The PM agent might not be able, however, to

[2] AGENT UML (http://www.auml.org/)

learn a tool agent's format if that format is completely different from its previous experiences (tool T7).

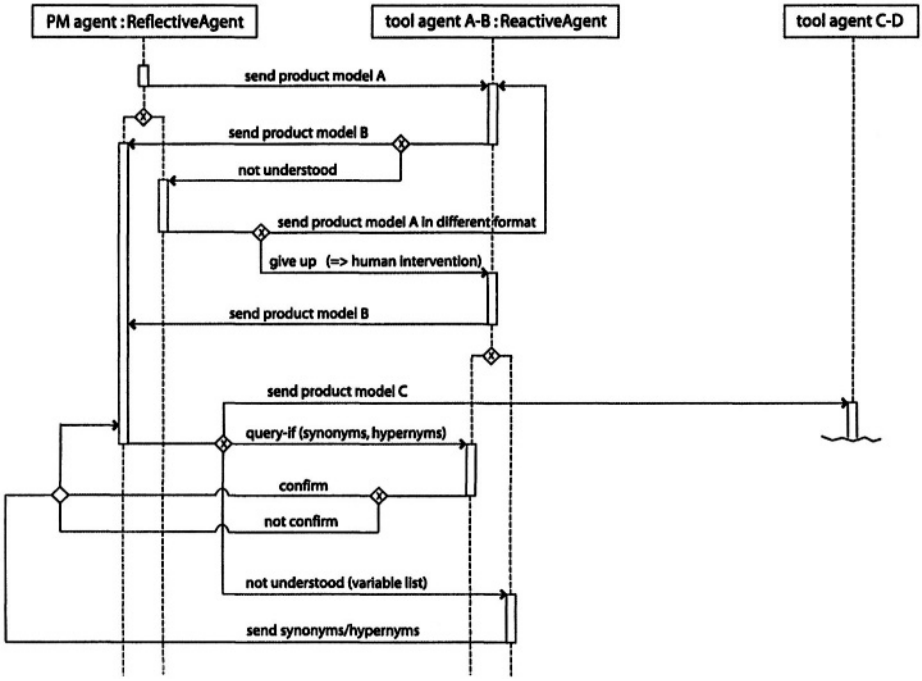

Figure 3. Possible interactions between the PM agent and a tool agent that needs a product model A as its input to produce product model B as its output (tool agent A-B). After the PM agent has successfully parsed product model B (eventually using human intervention), it sends a product model C to tool agent C-D.

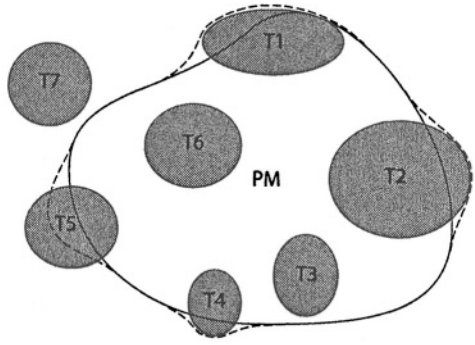

Figure 4. The PM agent's knowledge (PM) covering some of the formats of tool agents (T1 - T7). The dashed line indicates new knowledge gained over a series of interactions.

3. EXAMPLE: DESIGN OF A TURBOCHARGER

Our product example whose design data is to be exchanged among the agents is an exhaust-gas turbocharger for passenger cars, Figure 5. This product is quite complex with a large amount of data describing its structure and behaviour. For reasons of illustrative simplicity we have reduced this complexity to only 268 variables, without limiting the validity of the model.

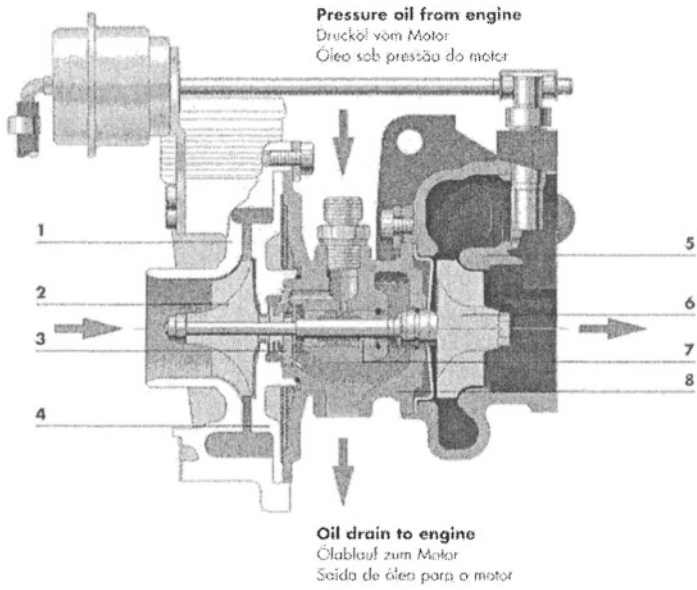

Figure 1. A turbocharger for passenger cars (Source: BorgWarner Turbo Systems): 1. Compressor housing, 2. compressor wheel, 3. thrust bearing, 4. compressor backplate, 5. turbine housing, 6. shaft & turbine wheel assembly, 7. bearing bushing, 8. centre housing.

All data is represented by content, specifically in the form of:

< variable name > < value > < unit of measure >

Different formats can be distinguished by different sets of variable names describing the same set of concepts. For example, depending on the format, the compressor air flow of **0.176 m³/s** at an engine speed of 5500 rpm may be represented as "AF1000 **0.176 m³/s",** "AF_1000 **0.176 m³/s"** or "V_RED_1000 **0.176 m³/s".** We have defined 7 different formats covering each of the 268 product variables.

We have implemented 16 tool agents, each of which has (hard-coded) knowledge about at most two of the 7 formats: one format must be used to represent the tool's input data and one to represent the tool's output data

(these formats can also be identical). Although a tool agent does not have the possibility to swap or mix its input and output format, it can communicate its knowledge about mappings among them (as synonyms or hypernyms) to the PM agent if requested. The PM agent uses its knowledge about the tool agents to provide each of them with the correct product model represented in the correct form. This knowledge is stored in its IAC network as associations between tool agents and the following types of properties:

- function or role in the design project
- behaviour (as an input-output view) of the tool or agent
- input format
- output format
- vendor of the tool or agent

This knowledge is also used as a bias for the correct parsing template to interpret product models produced by the tool agents. If the PM agent's knowledge about the correct format is not sufficient, the PM agent uses its knowledge about other, similar formats or about product semantics to parse the message. Its semantic knowledge consists of pre-coded qualitative relationships among some of the product variables, such as "nozzle inner diameter < nozzle outer diameter". The agent also uses its expectations and perceptions about the product model with respect to the number of variables, their type of values and their units of measurement. It can conclude, for example, that a numeric value with a unit of measure of **"N/mm^2"** possibly indicates "stress". Before adding these constructed assumptions to its knowledge and proceeding with the next design stage, the PM agent first seeks reconfirmation from the tool agent to ensure that its assumptions are correct. If the PM agent is unable to understand a product model using its semantic knowledge, it tries to receive help from the tool agent by communicating with it.

Figure 6 shows the project plan we have established for simulating a part of the design of a turbocharger. Every task is carried out by a different tool agent. We have set up a scenario that includes two iterations that represent necessary reformulation for optimising the product's performance after evaluating the prototype. This permits us to examine if the PM agent becomes more successful and efficient when interacting with the same agents for a second or third time. These are the agents carrying out the tasks 2 to 11 after the **1st** iteration and 3 to 12 after the **2nd** iteration. We have also set up some tool agents to modify their formats to simulate the integration of new tools during the design project and to test the PM agent's ability to adapt to these changes by quickly reorganising its memory.

The scenario requires a total of 68 product data transfers between the PM agent and the tool agents. We have given the PM agent some initial translation knowledge that is sufficient to cover 41 of them, while the

remaining 27 data exchanges will require either additional effort of interpretation and communication by the agents or intervention by the human user. This potential lack of interoperability concerns the input and/or output formats of 7 of the 16 tool agents.

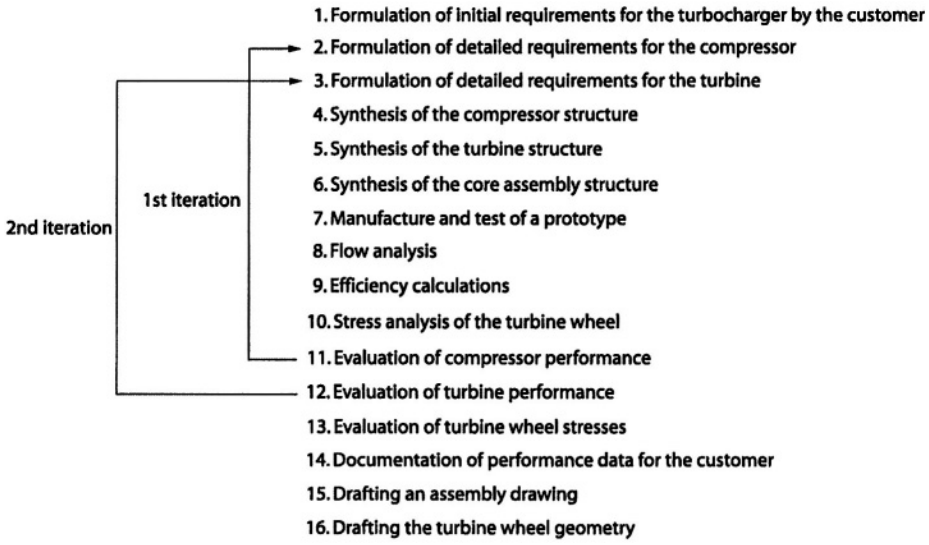

Figure 2. A project plan and scenario for simulating the design of a turbocharger.

4. RESULTS

Table 1. **Interoperability characteristics of the agent-based system compared to a static system using pre-defined translators.**

	Static system using pre-defined translators			Agent-based system			Increase in interoper-ability with the agent-based system
total no. data trans-fers	no. auto-mated trans-fers	no. manual trans-fers	Interoper-ability rate	no. auto-mated transfers	no. manual trans-fers	Interoper-ability rate	
68	41	27	60 %	60	8	88 %	46 %

Table 1 summarises the results of the simulation with regard to its efficacy. The agent-based system has raised the potential rate of interoperability by nearly 50% compared to what a static system would be able to achieve under the same initial conditions.

Table 2 gives an overview of the failures and repairs occurring in exchanging product data throughout the two iterations in the scenario. It also shows the PM agent's knowledge gaps regarding the formats of 7 of the 16 tool agents (labelled A-1 to A-16). As some of these gaps are gradually eliminated through agent interaction, the (potential) lack of interoperability is reduced resulting in fewer failures and less communication.

The PM agent has also been able to reorganise its knowledge after agents A-4, A-5 and A-8 unexpectedly changed their formats after the first iteration. As a result, it has been able to exchange product models with them after the second iteration without further problems or communicative effort.

Table 2. **Failures and repairs get reduced as gaps of knowledge are eliminated during interaction. Failures/repairs are related either to parsing the output of a tool agent or providing it with the correct input. The dark background represents the knowledge gaps pertaining at the beginning of the current sequence.**

	1st sequence (designing 1st prototype)	2nd sequence (optimisation 1)	3rd sequence (optimisation 2)
A-1	1.1		
A-2	2.1	2.2	
A-3	3.1 failed input output repair	3.2	3.3
A-4	4.1 input repair *	4.2 input repair **	4.3
A-5	5.1	5.2 output repair	5.3
A-6	6.1	6.2	6.3
A-7	7.1 failed input failed output	7.2 failed input failed output	7.3 failed input failed output
A-8	8.1	8.2 output repair	8.3
A-9	9.1 output repair	9.2	9.3
A-10	10.1	10.2	10.3
A-11	11.1	11.2	11.3
A-12		12.2	12.3
A-13			13.3 failed input
A-14			14.3
A-15			15.3
A-16			16.3

*** This input repair was due to an over-generalisation of the neural network giving the PM agent an incorrect bias, which led to constructing the product data in an inappropriate format.**

**** This input repair was due to A-4 changing its input format without informing the PM agent of this change.**

5. CONCLUSION

Our experiments have shown that an agent-based approach to product data exchange can provide interoperability without pre-defined translators. It can allow design projects to choose the computational tools appropriate for the needs of the actual design tasks and worry less about the availability of a standard translator. This would make these projects more adaptable to technological and organisational changes. Although our implementation has demonstrated to some extent how using agents in product modelling can increase interoperability, we expect better results when more than one situated agent is used. Our current work focuses on extending the tool agents (which to date are only reactive) to be situated.

Letting situated agents negotiate a shared product model on the fly can potentially constitute a method for pushing future standardisation. After a product model has been agreed upon and successfully used by a set of agents, this model can be used later as the prototype version of a new part of the standard model. Such a method would ground the standard in practice and accelerate its development and implementation.

ACKNOWLEDGEMENTS

This work is supported by a University of Sydney Sesqui Research and Development grant and by an International Postgraduate Research Scholarship.

REFERENCES

Clancey, W.J., 1997, *Situated Cognition,* Cambridge University Press, Cambridge.

Cutkosky, M.R., Engelmore, R.S., Fikes, R.E., Genesereth, M.R., Gruber, T.R., Mark, W., Tenenbaum, J.M. and Weber, J.C., 1993, PACT: An experiment in integrating concurrent engineering systems, *IEEE Computer* **26**(1): 28-37.

Eisenberg, A. and Melton, J., 1998, Standards in practice, *SIGMOD Record* **27**(3): 53-58.

FIPA, 2004, FIPA ACL Message Structure Specification, *Document No. SC00061G* (April 24, 2004); http://www.fipa.org/specs/fipa00061/SC00061G.pdf.

Gero, J.S. and Fujii, H., 2000, A computational framework for concept formation for a situated design agent, *Knowledge-Based Systems* **13**(6): 361-368.

McClelland, D.E., 1981, Retrieving general and specific information from stored knowledge of specifics, *Proceedings of the Third Annual Meeting of the Cognitive Science Society,* pp. 170-172.

Miller, G.A., 1995, WordNet: A lexical database for English, *Communications of the ACM* **38**(11): 39-41.

NIST, 1999, Interoperability cost analysis of the US automotive supply chain, *Planning Report #99-1,* NIST Strategic Planning and Economic Assessment Office, Gaithersburg, MD.

Pratt, M.J., 2001, Practical aspects of using the STEP standard, *Journal of Computing and Information Science in Engineering* **1**(2): 197-199.