Agent-Based Interoperability without Product Model Standards

Udo Kannengiesser and John S. Gero
University of Sydney
Sydney NSW 2006, Australia
{udo, john}@arch.usyd.edu.au

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. The agent-based approach developed and presented in this paper aims to flexibly provide product models in a form adapted to the need of the particular tools when there is no prior agreement on a common data format. Experiments show the feasibility of this approach as well as its efficacy and efficiency.

1 Introduction

Today's industrial product development faces a multitude of challenges arising from the dynamic environment in which it operates. Globalised markets, specialised customer demands and more regulations on safety, liability and sustainability have been major factors for increased product complexity and high requirements on quality. Furthermore, the need to produce at low cost and within short time frames has forced companies to find ways to make their processes more efficient.

Besides re-organising processes and strategies, industry has been increasingly reliant on using information technology to cope with these challenges. Systems for computer-aided design and manufacturing (CAD/CAM), various engineering analyses such as finite element analysis (FEA) or economic analysis are widespread today. However, most computational tools have been developed as stand-alone applications for a particular process, ignoring that this process has to fit into an interdependent network of processes, each of which needs as its input the results of others. Incompatible data formats required by the different tools have been recognised as a major issue for product development in most industrial areas (Eastman 1999). As a consequence, the transfer of product models is often done manually using paper-based documents (drawings, reports), which leads to rework and increases the risk of errors. A study conducted by NIST (1999) has estimated the annual cost of poor interoperability in the American automotive industry to be one billion US dollars.

1.1 Issues with standard approaches

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 predefined in this model and encoded into a standard form. One of the best-known product modelling efforts is the ISO 10303 standard, informally known as STEP (STandard for the Exchange of Product model data), which defines a neutral data model to be used to translate from one native tool format to the other. Another product modelling initiative is the International Alliance for Interoperability (IAI), a consortium formed by the architecture, engineering and construction (AEC) industry. The IAI aims to specify all objects in building projects that need to be made interoperable, which has resulted in a set of data classes called Industry Foundation Classes (IFCs) (IAI 2003). IFCs, like STEP, rep-

resent a neutral format that is used as a central element for every data translation.

There have been a number of reports on the use of STEP, IFCs or other standard product models in various individual design projects. However, there seems to be a slow acceptance of standardised product data exchange in industry. This is due to a number of reasons.

First, product model standardisation is an ongoing effort, and to date many data types already used by existing design tools are not yet available in STEP or IFCs. The processes of standards bodies such as ISO and IAI are organised to seek broad consensus among their members to achieve standards of high quality and general acceptance. As a result, the pace at which these standards are developed and released is extremely slow and always lags behind the practices and technological developments in industry (Eisenberg and Melton 1998; Tolman 1999). This organisational problem is complemented by the nonappropriability of open standards, which has resulted in poor private funding (NIST 1999).

Many tool vendors have also been reluctant for various technical and market strategy reasons to invest into the development of translators for STEP or IFCs. A major market risk has been the uncertainty about the industry's acceptance of a particular standard. In addition, tool vendors tend to prevent the adoption of a neutral standard format as this would make it easier for their customers to change to the software of competitors (NIST 1999). To date there has not been sufficient market pressure to enforce greater software support for STEP and IFCs.

Furthermore, as most tools have been developed for specific types of applications (project phase, industry sector or knowledge domain), they often do not share the same subset of the common standard format (Pratt 2001). However, interoperability between a set of tools is solely determined by the intersection of their capabilities to translate into or from the common standard. As a consequence there is an instant loss of data if that data is not supported by both ends of the exchange – the translator of the sending tool and the translator of the receiving tool. Interoperability within a design project thus becomes more and more fragile (if not impossible) with an increasing number of tools involved in that project. This results in difficulties in adapting to technological or organisational changes over the duration of a project that require the integration of new tools or new exchanges among present tools.

1.2 Recent research towards more flexibility

Traditional approaches to interoperability assume that a product model can be defined that is generally applicable across a set of projects for similar classes of products or within specific industry sectors. The fundamental issue here, however, is that design projects are often unique and can rarely be described using a pre-defined "one-fits-all" product model. Some researchers have therefore proposed to move the specification process of the standard closer to the actual product development processes. This has the additional benefit that design projects would be less dependent on the slow procedures of international standards bodies. Most of these approaches pre-define a minimal core set of basic data types. The designer generating the description of a new product can then flexibly extend these abstract types to model the current design.

Clayton et al. (1999) have proposed a software prototype that can be used by designers to specify not only the structure but also the function and behaviour of design components. It offers a graphical user interface for the designer to semi-automatically generate a comprehensive product model that can be seen as emerging during the design process. However, the focus here is on facilitating the evaluation process of the design rather than supporting data exchange across different tools or design stages.

An approach proposed by Stouffs and Krishnamurti (2002) provides a framework for representing product data based on a standardised syntax rather than semantics. It defines primitive data types that can be combined using formal compositional operators to form more complex data types. The resulting canonical representation allows comparing, mapping and translating different product models by designers. The specification of product models is done manually.

A similar idea is the domain-independent core representation for product data presented by Szykman et al. (2001). It is based on a high-

level classification of artefact properties into function, behaviour and structure. The simplicity and extendibility of this generic product model aims to increase its adoption by users and vendors and to facilitate interoperability across different domains. It also attempts to provide a basis for the integration of functional and behavioural (besides structural) product information in next-generation design tools. Szykman and his co-authors hint towards the possibility of specialising the core model only according to the specific needs for data exchange in a design project. A comparable approach using a generic ontology has been applied by Dartigues and Ghodous (2002) for data transfer between commercial design and process planning tools.

Haymaker et al. (2003) have proposed a framework that allows designers to define generic reasoning mechanisms, called "Perspectors", which automate the production of a particular geometric view of the design. Perspectors are then integrated into a network that maintains consistency among all dependent views when the geometry of the product is modified during the design process. Once they have been written, they can be re-used later, in subsequent design projects. However, there is no mechanism to automate retrieving or generating suitable Perspectors or organising them into a dependency network.

Our research is in line with these approaches in that it develops the standard when it is needed, i.e. at runtime rather than *a priori*. However, it aims to automate the generation of a standard using computational agents that use their previous experience to effectively and efficiently come to an agreement on that standard.

2 Foundations

The research presented in this paper is based on ideas in cognitive science and their integration into computational agents.

2.1 Situatedness

Designing is an activity during which the designer performs actions in order to change the environment. By observing and interpreting the results of their actions, they then decide on new actions to be executed on the environment. This means that the designer's concepts may change according to what they are "seeing", which itself is a function of what they have done. One may speak of a recursive process, an "interaction of making and seeing" (Schön and Wiggins 1992). This interaction between the designer and the environment strongly determines the course of designing. This idea is called situatedness, whose foundational concepts go back to the work of Dewey (1896) and Bartlett (1932).

In experimental studies of designers some phenomena related to the use of sketches, which support this idea, have been reported. Schön and Wiggins (1992) found that designers use their sketches not only as an external memory, but also as a means to reinterpret what they have drawn, thus leading the design in a new direction. Suwa et al. (1999) noted, in studying designers, a correlation of unexpected discoveries in sketches with the invention of new issues or requirements during the design process. They concluded that "sketches serve as a physical setting in which design thoughts are constructed on the fly in a situated way".

An idea that fits into the notion of situatedness has been proposed by Dewey in 1896 (Clancey 1997) and is today called constructive memory. Its relevance in the area of design research has been shown by Gero (1999). Constructive memory is best exemplified by a quote from Dewey via Clancey: "Sequences of acts are composed such that subsequent experiences categorize and hence give meaning to what was experienced before". The implication of this is that memory is not laid down and fixed at the time of the original sensate experience but is a function of what comes later as well. Memories can therefore be viewed as being constructed in response to a specific demand, based on the original experience as well as the situation pertaining at the time of the demand for this memory. Therefore, everything that has happened since the original experience determines the result of memory construction. Each memory, after it has been constructed, is added to the existing knowledge (and thus the situation) and is now available to be used later, when new demands require the construction of further memories. These new memories can be viewed as new interpretations of the augmented knowledge.

The advantage of constructive memory is that the same external demand for a memory can potentially produce a different result, as newly

acquired experiences may take part in the construction of that memory. Constructive memory can thus be seen as the capability to integrate new experiences by using them in constructing new memories. As a result, knowledge "wires itself up" rather than being fixed, and actions based on that knowledge can be altered in the light of new experiences.

Constructing a memory can be described as being governed both by what was initially there (the original experience) and by what the current situation (made up by previous experiences and memories and the currently focussed concepts) makes with it. At this level of abstraction, constructive memory can be viewed in a similar way as the process of interpretation, which Gero and Fujii (2000) have described as consisting of two parallel processes interacting with each other: A *push process,* where the production of an internal representation is driven ("pushed") by the sensed data, and a *pull process*, where the interpretation is driven ("pulled") by some of the designer's current concepts, which has the effect that the original data is biased to match the current expectations about what the interpretation should be.

2.2 Situated agents

A distinguishing feature of a situated agent is its autonomy in a stronger sense, which implies its ability to learn from experience (Russell and Norvig 1995). Specifically, a situated agent grounds its knowledge in its interactions with the environment (Bickhard and Campbell 1996; Clancey 1997). Grounded knowledge can be viewed as internal representations in a first-person perspective as opposed to third-person encodings. Situated agents use their current knowledge, their current goals and sense-data from the external environment to construct this new knowledge in accordance with the push-pull idea. Some initial encoded knowledge, however, is necessary to initiate this kind of processing, which can be viewed as a set of innate abilities or biases.

Another concept of a situated agent is its rationality. Rationality constrains the agent's actions to conform to its current goals and beliefs (Cherniak 1986). Combined with the above concept of autonomy this has the effect that the agent can potentially act in ways it has never acted before as it learns and acquires new experiences. One can view this as the agent's ability to use its augmented knowledge and goals to augment its state space of possible actions to be executed on the environment. This makes the agent highly adaptive even to situations it has not encountered before.

Gero and Fujii (2000) have proposed a framework that describes the processes involved in the interaction of a situated agent with its environment. Based on this framework, Maher and Gero (2002) have derived a modular agent architecture as depicted in Figure 1.

The agent's sensors monitor the environment to produce sense-data relevant for the agent. Sense-data can be thought of as raw data representing attributes of the sensed entities in the environment. 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.

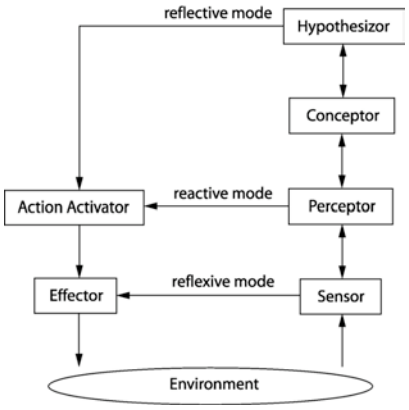


Figure 1. A modular architecture for a situated agent allowing different modes of reasoning (after Maher and Gero (2002)).

This architecture distinguishes different modes or levels of cognition (indicated in Figure 1) that impose different constraints on the agent's flexibility when interacting with the environment (Maher and Gero 2002):

Reflexive Mode: The agent uses the raw sensory input data from the environment to generate a pre-programmed response. No reasoning is required here, and the agent's action is a direct consequence of the sensed data.

Reactive Mode: The agent uses percepts to activate its actions, which can be viewed as a limited form of intelligence. Reasoning is done using sensation and perception processes and knowledge about possible sequences of actions. No goals or concepts are involved here, even though an observer might ascribe goal-directedness to the agent's behaviour. The mapping from the perceptor to the action activator is pre-programmed to match a fixed set of goals ascribed to the agent by the agent designer.

Reflective Mode: The agent constructs concepts and uses them to hypothesize possible desired external states as goals and propose and execute actions that aim to achieve those desired states. This mode of cognition provides the agent with the highest capacity for flexible behaviour, as the agent can choose its actions upon "reflection" on its goals and concepts.

2.3 Common ground

When agents communicate they have to share a common body of background knowledge in terms of concepts and terminology. Cognitive science has frequently referred to this shared (or mutual) knowledge as *common ground*. Common ground has been described as the set of presuppositions "any rational participant [in a conversation] is rationally justified in taking for granted, for example, by virtue of what has been said in the conversation up to that point, what all the participants are in a position to perceive as true, whatever else they mutually know, assume, etc." (Karttunen and Peters 1975). In contrast to a common ontology as used in traditional agent communication, common ground is defined from the bottom up based on the individual agents rather than imposed from the top down. It has often been defined recursively (Clark and Marshall 1981):

A and B mutually know that p = definition
(r) A knows that p and that r′.
(r′) B knows that p and that r.

The infinite recursion implied in this model of common ground has often been seen as an issue: How do agents manage to assess their common ground within a reasonable amount of time? The traditional Artificial Intelligence (AI) approach to agent communication has avoided this issue by idealising common ground to become an irreducible ontology that is common and available to all agents (Wooldridge 2000).

Clark and Marshall (1981) have advocated using a heuristics to avoid recursion. Their co-presence heuristics uses evidence of certain states of affairs – things that are "co-present" to both agents – and assumptions about the agents' general cognitive abilities as a basis to infer common ground as "a single mental entity instead of an infinitely long list of even more complex mental entities" (Clark and Marshall 1981). This common ground can be classified into personal and communal common ground (Clark 1996).

Personal common ground refers to objects or concepts that are directly co-present in the current interaction. This includes physical co-presence, such as a soccer ball that is located between two opponents, and symbolic co-presence, such as the representation of a soccer ball introduced in a conversation. In both cases, the soccer ball is considered as part of the personal common ground among the two agents.

Communal common ground refers to objects or concepts that are indirectly co-present through evidence that both agents are members of the same community within which these objects or concepts can be assumed to be universally known. For example, it can be assumed that all members of the community of chess players know what "castling" means. Community membership can overlap; for example, one may simultaneously be a member of the community of chess players, the community of Australians and other communities and sub-communities. Evidence of community membership can be provided directly through communication or indirectly through inference from the agent's physical features (e.g. dress, location or

possessions) or behaviours (e.g. dialect, jargon or skill) (Clark 1996).

How do the agents use their common ground? Cognitive studies have shown that speakers tailor their messages to the common ground they share with their addressees, which has commonly been referred to as "audience design" (Clark and Murphy 1982). In turn, an addressee uses the same common ground to construct the meaning of the message as intended by the speaker (Fussell and Krauss 1989). As opposed to a common ontology, common ground is constructed during the process of interaction between particular agents. Brennan and Clark (1996) have found that two agents create "conceptual pacts" as temporary agreements on terms for particular objects, which are only valid in the conversation between these two agents. When the agents communicate with different agents, new conceptual pacts about the same objects are interactively established that may not be the same as in the previous situation.

It is unrealistic to expect that communicative actions are always accurate and perfectly adapted to the knowledge of the agents. One reason is that the construction of common ground is always an approximation that at times proves incorrect. In addition, the sensory-motor capabilities of one or the other agent might be impaired or the communication channel noisy. Horton and Keysar (1996) have also identified that time pressure can reduce listener adaptation of a speaker's communicative actions.

The potential inadequacies in constructing common ground require the collaborative effort of both agents involved in the communication to establish a shared understanding. Many researchers have therefore viewed communication as a joint activity (Grosz and Sidner 1990; Cohen and Levesque 1994; Clark 1996), to which the individual agents contribute by producing communicative actions turn by turn. Of particular importance is feedback, which can be positive indicating understanding or negative indicating failure of understanding. In case of negative feedback, the communicative actions will typically be concerned with eliminating the lack of understanding (and thus re-establishing common ground), which is generally referred to as *repair* (Schegloff et al. 1977). This is done via reformulating or explaining the communicative action causing the problem. During the course of a conversation, the common ground among the agents gradually increases.

3 Conceptual model

Interacting agents use internal representations of the world that have been referred to as mental models (Norman 1983). Socially interacting agents use mental models of their social world, i.e. of themselves and other agents. We have shown in an earlier paper (Gero and Kannengiesser 2003) how the function-behaviour-structure (FBS) schema is useful to support these mental models. This Section briefly summarises the basic features of the FBS schema.

3.1 The FBS schema in design research

Gero (1990) has proposed a representation schema for design knowledge, i.e. knowledge about existing (physical) or to be designed (imaginary) objects, based on the notions of function (F), behaviour (B) and structure (S):

- *Function* (F) describes the teleology of an object.
- *Behaviour* (B) describes the attributes that are derived or expected to be derived from the structure (S) of an object.
- *Structure* (S) describes the components of an object and their relationships.

The FBS schema includes both the semantics (stated as functional requirements) and the syntax of the design (stated as a structural description produced by the designer) and links them via behaviour. Specifically, the agent ascribes function (F) to behaviour (B) by establishing a teleological connection between the agent's goals and observable or measurable effects of the object. Work on causal reasoning in qualitative physics (De Kleer and Brown 1984; Kuipers 1984) has been the most influential for the understanding of the structure-behaviour link in today's design research. Here the structure (S) is often interpreted as the components and relationships of physical devices, and the derivation of behaviour (B) is then based on physical laws and heuristics.

The FBS schema is sufficiently general to cover all interpretations of objects and thus to support the construction of mental models that are grounded in the interactions of the agent using them. This is most obvious for interpreta-

tions of the teleology (F) of an object, since it is closely connected to the agent's current goals that are likely to change for one agent as well as differ for different agents. However, the interpretation of behaviour (B) and structure (S) of an object is also situated. Take the example of a mobile phone as the artefact; an electrical engineer typically views its structure (S) in terms of electronic circuits and the resulting behaviour (B) in terms of technical performance variables such as the Specific Absorption Rate (SAR) or the ability to browse Wireless Application Protocol (WAP). These and other behaviours (B) would be relevant also for the software engineer; however this kind of specialist usually sees software components as the structure (S). The user of the same mobile phone is likely to lack the technical knowledge or interest to interpret the structure (S) and behaviour (B) in these terms and would rather view them as shape or colour (as structure (S)) and properties related to usability (as behaviour (B)).

Once a number of experiences with objects has been gained and represented in the FBS form, the agent is able to generalise by clustering sets of like experiences. When the agent needs to access its knowledge about a particular object, it can derive a large part of this knowledge without much computational effort from its generalized experiences. This derived knowledge may even add (default) assumptions about an object where information gained from directly interacting with that object is missing.

The FBS schema provides a uniform set of constructs to model objects at all levels of generality and thus significantly supports this generalisation. Gero (1990) has introduced so-called design prototypes that represent generalised design knowledge, from which specific design objects can be instantiated. Design prototypes are useful for a design agent to start designing even if only incomplete information is available about the function, behaviour and structure of the object to be designed.

3.2 Modelling agents

Using the FBS schema to model agents stresses its support of mental models even more. Figure 2 (bottom and upper right) shows 19th century drawings of the so-called "mechanical duck" designed by the French engineer Jacques de Vaucanson (1709-1782) and first exhibited in 1739. The visible surface structure (S) of the duck consisted of a full-bodied model of a duck made out of copper. Parts of the structure (S) were hidden from the observer inside that body and included a collection of weights, levers and hundreds of other moving parts similar to a mechanical clock. This mechanism allowed the duck to flap its wings, to spread its tail, to quack, to reach for food, to eat and digest grain and to excrete the remains. These behaviours (B) were designed for the purpose (F) of demonstrating that the anatomy of a duck can be copied in a mechanical way.

We can safely assume that Jacques de Vaucanson knew the structure (S) of his design in every detail. In contrast, most visitors of the exhibition were probably left alone with their observations of the duck's behaviour (B) without knowing more about its actual mechanical structure (S) than its surface structure (the duck is now shown as a "black box" in Figure 2, upper left). Given this lack of knowledge to explain the behaviour (B) and the life-like physical appearance of the duck, some of the visitors were perhaps tempted to assume "mental states" as the mechanism driving the duck's actions. This corresponds to adopting an intentional rather than an omniscient physical stance (Dennett 1987) and ascribes (a strong notion of) agency to the mechanical duck (Wooldridge and Jennings 1995). In this case, the structure (S) of the duck regarded as an agent is made up of "mental constructs" held to be the driver of some of the duck's behaviours (B). Examples may include:

- Wanting to eat causes the duck's neck to reach down for food
- Being scared causes the duck to flap its wings
- Having a perception of people's attention causes the duck to quack

This intentional view of structure (S) may be seen as "superstitious" or "unscientific". However, this has been found to be a typical feature of mental models (Norman 1983) and is compensated by its ability to explain or predict the object's (or agent's) behaviour (B) in an effective and efficient way. The causality in the link between structure (S) and behaviour (B) of an agent still exists. However, it is no longer estab-

lished by physical laws but by the principle of rationality that, just like a law, governs all the actions of an agent (Newell 1982).

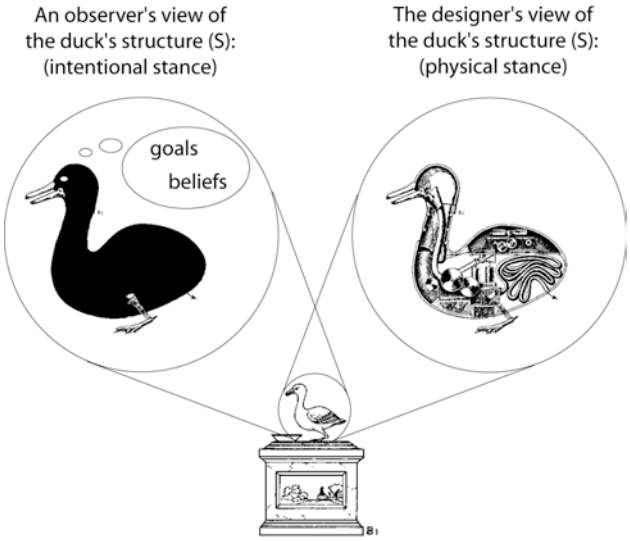
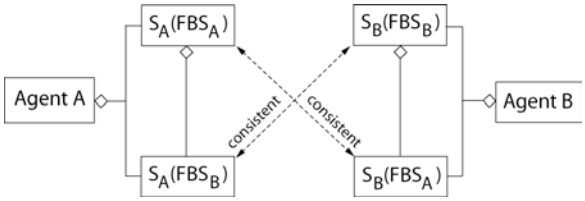


Figure 2. Different views of the structure (S) of Jacques de Vaucanson's "mechanical duck". (assembled from drawings published in Strandh (1979))

Figure 3 summarises the view of an agent in terms of FBS. It distinguishes between two kinds of structure (S): One refers to the "fixed" parts of the agent (S^f), i.e. those components or processes that are given at its inception and that are not subject to significant change. This type of structure is the same as for objects that have no agency, and typically includes visible "outside" components such as the sensors and effectors of the agent. The other kind of structure refers to the "situated" parts of the agent (S^s), i.e. those internal representations or processes that are constructed by the agent's interaction with its environment. The situated structure of an agent may be interpreted as its grounded concepts, beliefs, goals etc., which are often hidden from the observer (and thus depicted "inside" the fixed structure in Figure 3). The behaviour (B) of an agent is often interpreted as a "black-box" or "input-output" view, i.e. the way the agent acts given a set of conditions. A common interpretation of the function (F) of an agent is its social role within a group of agents.

3.3 Modelling situated interactions

Interactions between agents that conform to the ideas of situatedness and common ground can be modelled using the FBS view of agents. Gero and Kannengiesser (2003) have proposed a model of common ground as depicted in Figure 4 using a UML-style notation. Here the knowledge of two agents is represented as the FBS models they have constructed of each other (including of themselves). Common ground then encompasses those parts of an agent's FBS model that are consistent with the corresponding FBS model constructed by the other agent.



S_A (FBS_A) = FBS model of agent A held by agent A
S_A (FBS_B) = FBS model of agent B held by agent A
S_B (FBS_B) = FBS model of agent B held by agent B
S_B (FBS_A) = FBS model of agent A held by agent B

Figure 4. Pairs of consistent FBS models that establish the common ground of two agents.

All knowledge that is considered part of common ground according to Clark and Marshall's (1981) co-presence heuristics is co-present either directly in the current interaction (personal common ground) or indirectly via generalised experience with previous interactions with the same or similar agents (communal

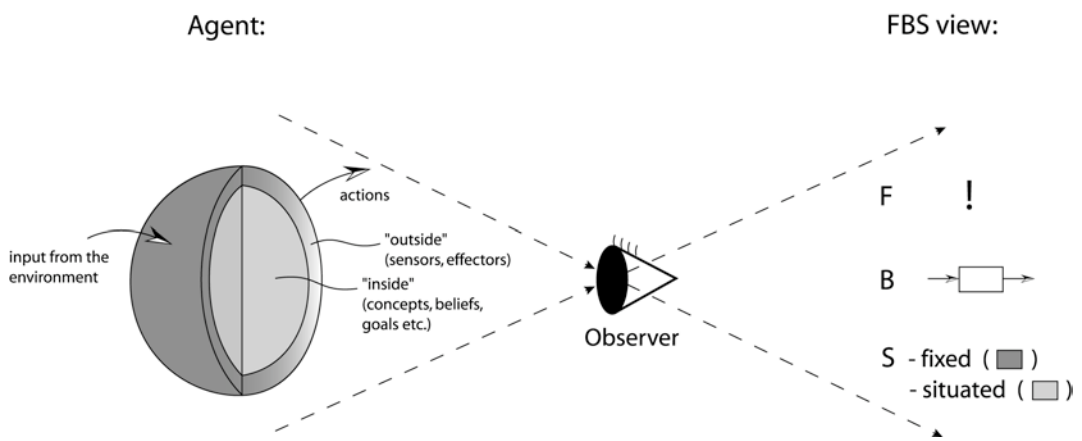


Figure 3. An FBS view of an agent.

common ground). Personal common ground thus uses a specialised FBS model of the other agent constructed from the current interaction with that agent, while communal common ground uses a generalised FBS model that is retrieved using cues provided by the specialised FBS model.

This conception of constructing common ground resembles Gero's (1990) model of designing with design prototypes and shares some of its benefits. Generalised FBS models are constructed from sets of like direct experiences (specialised FBS models), which reduces computational complexity. Generalised FBS models can also be used to complement the current (specialised) FBS model of an agent by deriving additional FBS properties that are typical within that agent's community. This is useful when not much is known about that agent.

Figure 5(a) shows the (specialised) FBS models that an agent (0) has constructed of itself as well as of agents it has previously interacted with (1 – 4). The Figure uses a less formal but more intuitive way to represent the relationship between FBS_0 and $FBS_{1\text{-}4}$ as the latter being nested in the former. The FBS models are represented as circles of different sizes to indicate the different amounts of knowledge the agent (0) has acquired about these agents. For example, some agents (1 and 2) are better known than others (3 and 4), and the best-known agent is certainly agent 0 itself.

Let us now assume that an agent (5) comes into play and starts interacting with agent 0, Figure 5(b). As shown in the Figure, agent 0 has rather little knowledge about that agent. Its FBS model of agent 5 might perhaps only be founded on the very first moments of that agent's advent, if it has never interacted with it before. In order to complement this FBS model, agent 0 has to derive additional knowledge from experiences generalised from interactions with agents that are similar to agent 5. Figure 5(c) depicts the provenance of this generalised knowledge using arrows from the FBS models of similar agents. A major part of this knowledge is expected to come from agent 0's FBS model of itself.

Although generalisations are an efficient way for an agent to construct a set of assumptions that are likely to serve as a sufficient starting point for the intended interaction, some of these default assumptions may turn out to be incorrect, usually via a negative reaction by the other agent. In this case, the agents will engage in repair to collaboratively adjust agent 0's FBS model of the other agent.

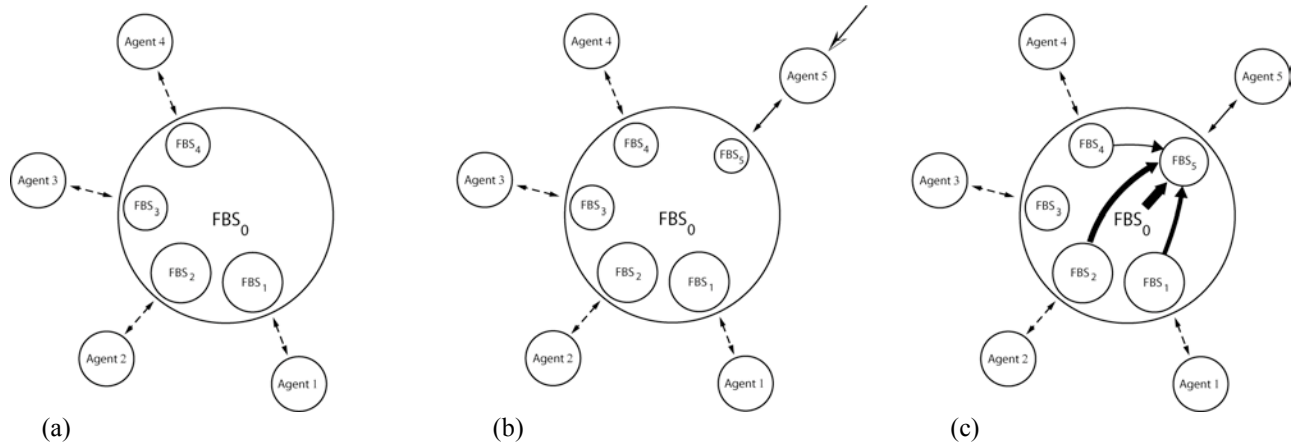


Figure 5. New FBS models are constructed using generalisations of previously constructed FBS models. The size of the circle for each FBS is an indication of the amount of grounding of this FBS model of the other agent. The width of the arrows is an indication of the confidence of the potential applicability of the originating FBS model in constructing or supporting the FBS model of a new agent. Double arrows represent agent interactions (previous interactions in dashes), (a) model at some state, (b) the interaction with a new agent, agent 5, initiates a new FBS representation of that agent, and (c) knowledge sources used to construct agent 0's FBS model of agent 5.

4

Implementation

To implement and test our conceptual ideas, we have set up an agent-based testbed that simulates product data exchange among a set of tools. This Section gives an outline of the testbed in terms of the product example, the architecture of the agent-based system, the process scenario and the implementation of the agents.

4.1 Product example

We have chosen an exhaust-gas turbocharger for passenger cars as the product whose data is to be exchanged during the process of its design, Figure 6. This product is complex with a large amount of data describing its structure and behaviour. For reasons of illustrative simplicity we have reduced this complexity to only 299 variables, without limiting the validity of the model. Although our product example has been chosen from the mechanical engineering domain, the conceptual ideas underpinning our approach can easily be extended to other domains such as building design.

We have organised the components shown in Figure 6 into a 3-layer tree structure, Figure 7. Product models referring to the leaf nodes represent mainly structure (S) variables such as geometry and material, while those referring to higher-level nodes represent mainly behaviour (B) variables such as pressure, temperature, air flow etc.

All product models are 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 m^3/s at an engine speed of 5500 rpm may be represented as "AF5500 0.176 m^3/s", "AF_5500 0.176 m^3/s" or "V_RED_5500 0.176 m^3/s". We have defined 7 different formats covering each of the 299 product variables.

4.2 System architecture

In this testbed, a number of agents are conceived as "wrappers" of design tools, Figure 8. We shall refer to these agents as "tool agents". Tool agents not only control their encapsulated tools but also have knowledge about their role in the design process and how they can use their tools to execute the tasks expected from them. As such, they can be viewed as a collection of individual design experts. In order to integrate their capacities into a distributed product develop-

ment system, we have added an intermediary agent that maintains all the data of a particular product throughout the different design stages from the initial specification to the released design description. Specifically, every tool agent is responsible for a specific design stage and modifies or adds to the product data. The intermediary 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. This architecture resembles federated approaches such as PACT (Cutkosky et al. 1993).

Figure 6. 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

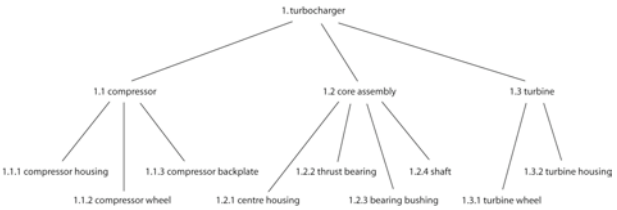


Figure 7. Assembly structure of the turbocharger

4.3 Process scenario

Figure 9 shows the project plan we have established for simulating the design of the turbocharger. Every task is carried out by a different tool agent. We have set up a scenario that includes two iterations that represent reformulation for optimising the product's performance after evaluating the prototype. This permits us to examine if repeated interactions between the same agents become more successful and efficient over time. We have also set up some tool agents to replace others during the design project to simulate the runtime integration of new tools with different formats. This allows testing the agents' ability to adapt to these changes. The scenario requires a total of 68 product data transfers between the mediator agent and the tool agents.

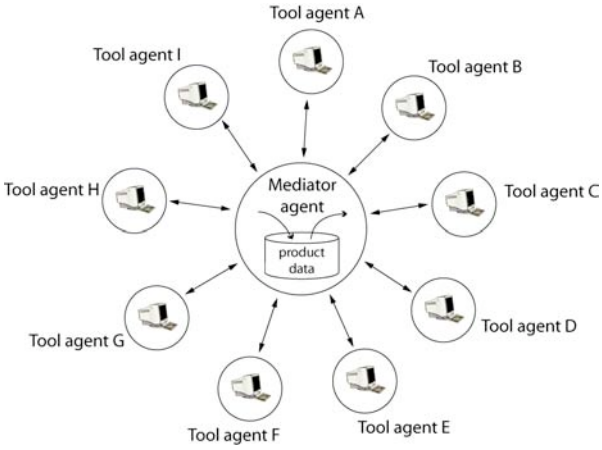


Figure 8. Architecture of a testbed for agent-based product data exchange.

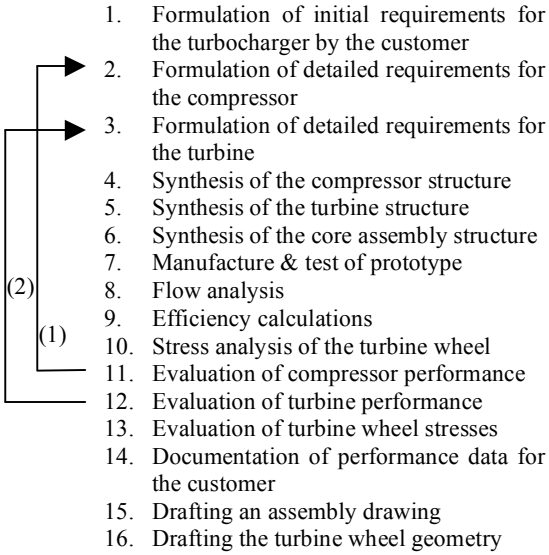


Figure 9. A project plan for simulating the design of a turbocharger with 2 iterations occurring during the process.

4.4 Implementation of agents

All agents are implemented in the rule-base language Jess (Java Expert System Shell) and connected to their environment by sensors and effectors written in Java. We have classified the agents according to the most sophisticated mode of reasoning that their architecture allows (cf. Section 2.2). *Reactive agents* only have perceptors and action activators and thus cannot reason reflectively. Most tool agents have been implemented as reactive agents. *Reflective agents* have all the necessary modules, including conceptor and hypothesizor, to reflect on their interaction in a situated way. In addition, reflective agents have neural networks (written in Java) to represent their generalised FBS models of the other agents. The neural networks are interactive activation and competition (IAC) networks that have the same architecture as presented by McClelland (1981). We have selected this type of neural network because it can produce generalisations and default assumptions. We have also given the agents the capability to add new neurons to their IAC network as well as to reorganise the connections among the neurons to update their memory in the light of new experiences. The mediator agent and some of the tool agents are reflective agents.

The agents have knowledge about the turbocharger according to their field of expertise, which encompasses not only semantic but also syntactic knowledge (i.e. the format in which the product is represented). The amount and use of syntactic knowledge depends on the class of agent:

- Reactive agents: Every reactive agent has hard-coded knowledge about at most 2 of our 7 formats (cf. Section 4.1): one corresponds to the native input format and the other to the native output format of the encapsulated tool. These formats can also be identical. Reactive agents cannot swap or mix their input and output formats, and every product model must exactly match the required format.
- Reflective agents: Similar to reactive agents, reflective agents have 2 preferred (native) formats; however they can use both of them (as well as any additional format that they are given or that they learn) to comprehend or produce product models. The mediator agent has been given more initial formats than the tool agents. Reflective agents integrate new formats or new parts of a format into their generalised FBS models. The FBS models include expectations about other agents'
    - role in the project (F)
    - task output (B)
    - knowledge of an input format (S)
    - knowledge of an output format (S)
    - vendor (generalised relationship between different FBS models).

Reflective agents use their generalised FBS models as a bias for the correct parsing template to interpret product models. If the knowledge about a particular format is not sufficient, the agent can use a combination of several methods to construct the format:

- Syntactic similarity: The agent can, for instance, use generative rules to parse the unknown descriptor "AF_5500" based on similarity with the known descriptor "AF5500".
- Semantic knowledge: The agent can make assumptions based on pre-coded qualitative relationships between some of the product variables, such as "nozzle inner diameter < nozzle outer diameter".
- Expectations: If the number of unknown descriptors matches the number of expected product variables that could not be parsed, the search space can be reduced to an extent that may allow constructing specific assumptions about the unknown descriptors.
- Type of values and units of measurement: The agent can conclude, for example, that a numeric value with a unit of measure of "N/mm^2" possibly indicates "stress".

Whenever an agent has to use one of the above methods to interpret a product model, it seeks reconfirmation from the agent that produced this product model to ensure its assumptions are correct and can be grounded. If the agent fails to interpret a product model, it notifies the other agent that then tries to repair the model. If all attempts to communicate product data between two agents fail, the receiving agent will be given the data manually from the human

user (our system simulates this user intervention to allow continuity of the experiments).

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 (repair) their meanings when an agent fails to understand a product model. The latter type is based on synonyms and hypernyms (terms for abstract superclasses) as a means to explain unknown data. This has been inspired by the conceptual foundations of WordNet (Miller 1995).

5 Experiments

This Section presents the methodology we have used to test our implementation. We have set 2 objectives:

1. to test the efficacy of our approach by examining its ability to provide higher interoperability rates than a pre-defined system
2. to test the increasing efficiency of our approach by examining its ability to reduce the occurrence of repair over time

The following question arises: How can we conceive the architecture of a static system so that we can compare it with our agent-based system? In other words, which assumptions would we have to make for our agent-based system that it has the same capacities as a static system?

Our agent-based system has integrated the selection and execution of translations into the situated knowledge of the mediator agent. In contrast, a static system pre-defines the mappings between the tools and their translators as well as between the native formats and a common standard. Therefore, if we want to model a static system using our agent-based one, we would only have to fix the knowledge of the mediator agent (and all other agents) by removing its ability to construct new formats and integrate them into its FBS models of the agents.

Lack of a common standard can then be modelled as gaps in the mediator agent's fixed knowledge. Any gap in the knowledge required to exchange particular data with a particular tool necessarily leads to failure in establishing interoperability. This model has the advantage that the interoperability of a static system can be predicted directly from the initial knowledge of the agents without having to run any actual simulations.

We have set up a series of experiments that gradually reduce potential interoperability by systematically decreasing the initial knowledge of the mediator agent. Figure 10 shows 3 different possibilities where FBS knowledge can be reduced to impair communication with another agent.

- X1: The agent knows a particular variable name (from a different experience or format) but does not know that this variable name (also) belongs to format f.
- X2: The agent does not know a particular variable name at all.
- X3: The agent does not know which format the other agent uses.

Seven cases can be defined using different combinations of all 3 possibilities that are expected to gradually increase the degree of difficulty for agent A-0, Table 1. Accordingly, we have specified 7 different experimental settings, each of which tries to include instances of one of the 7 cases. Due to the complex interrelationships between the product data, the formats and the agents, some overlap could not be avoided, Table 2. However, we see these overlaps more as a benefit than a shortcoming, as they distribute the different cases over several experiments thus providing a broader and more solid basis for testing the system.

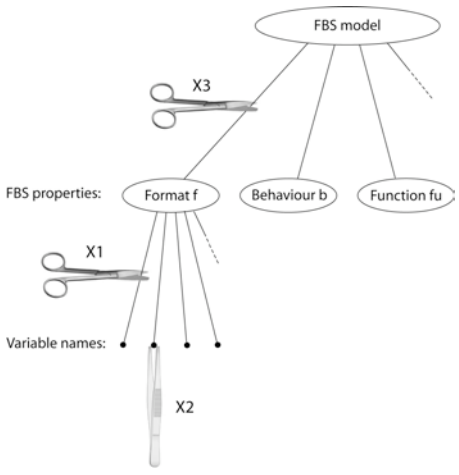


Figure 10. 3 different possibilities (X1, X2 and X3) to cut off FBS knowledge about a tool agent.

Table 1. Possible combinations of the different kinds of initial knowledge gaps (Figure 10).

	Initial knowledge gaps for a particular exchange		
Case no.	X1	X2	X3
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

Table 2. Distribution of the occurrence of the 7 cases from Table 1 in every experiment.

Experiment	Number of occurrences							Sum
	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7	
1	18	0	0	0	0	0	0	18
2	0	30	0	0	0	0	0	30
3	0	8	18	0	0	0	0	26
4	0	0	0	34	0	0	0	34
5	10	0	0	21	8	0	0	39
6	0	16	0	4	0	14	0	34
7	0	4	6	1	0	4	12	27

Table 2 also shows for every experiment the sum of all potentially problematic data exchanges. This sum equals the number of times that a static system based on standard translators would fail. If our agent-based system is to increase interoperability, we would expect that the number of failures per experiment is reduced. In a first series of experiments, we have set up our system (agent-based system I) to include only one reflective agent – the mediator agent – while all the other agents are reactive. In a second series of experiments (agent-based system II), we have made reflective all those reactive tool agents from the first experiments that had been involved in data exchanges that either failed or had to be repaired. This is expected to further improve the results in terms of efficacy and efficiency.

6 Results

6.1 Efficacy

Tables 3 and 4 summarise the experimental results related to the interoperability of the 3 systems (1 static, 2 agent-based systems). We have defined the interoperability rate (IR) as

$$IR = \frac{no.\ successful\ \ product\ data\ transfers}{no.\ total\ \ product\ data\ transfers}$$

The chart in Figure 11 plots the interoperability rates of each of them for the 7 experiments. It can be seen that the agent-based systems I and II could increase the average interoperability compared to the static system by 54% and 79%, respectively.

Figures 12 and 13 show how interoperability rates differ depending on the direction of the data transfer. Transferring data from the tools to the mediator agent was always successful in the two agent-based systems, Figure 12. This is a result of the variety of different possibilities accessible for the mediator agent to construct new formats, which is also facilitated by its rather large pool of knowledge about different formats. In contrast, transferring data from the mediator agent to the tool agents was obviously a harder task, Figure 13. The relatively low performance of the first agent-based system (using only reactive tool agents) can be explained by the inabil-

ity of the tool agents to construct new formats. However, it still achieved higher interoperability rates than the static system because some knowledge gaps could be filled by knowledge that the mediator agent had learned from previous interactions. The second agent-based system (using some reflective tool agents) has performed much better, as the tool agents were now able to construct new formats. However, their pre-existing knowledge about formats was limited to a maximum of 2 different (native) formats, which is the reason why not every attempt to construct a format was successful.

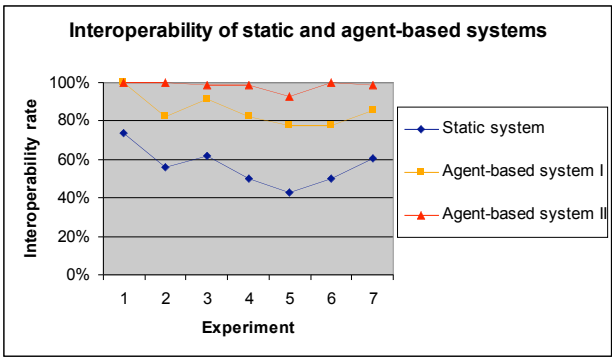


Figure 11. Comparison of interoperability rates.

Table 3. Interoperability characteristics of agent-based system I compared to a static system.

Experiment	Total number of data exchanges	Static system: Number of interoperable exchanges	Static system: Number of non-interoperable exchanges	Static system: Interoperability rate (IR)	Agent-based system I: Number of interoperable exchanges	Agent-based system I: Number of non-interoperable exchanges	Agent-based system I: Interoperability rate (IR)	Agent-based system I: Increase in interoperability compared to static system
1	68	50	18	74%	68	0	100%	36%
2	68	38	30	56%	56	12	82%	47%
3	68	42	26	62%	62	6	91%	48%
4	68	34	34	50%	56	12	82%	65%
5	68	29	39	43%	53	15	78%	83%
6	68	34	34	50%	53	15	78%	56%
7	68	41	27	60%	58	10	85%	41%
Average		38	30	56%	58	10	85%	54%

Table 4. Interoperability characteristics of agent-based system II compared to a static system.

Experiment	Total number of data exchanges	Static system: Number of interoperable exchanges	Static system: Number of non-interoperable exchanges	Static system: Interoperability rate (IR)	Agent-based system II: Number of interoperable exchanges	Agent-based system II: Number of non-interoperable exchanges	Agent-based system II: Interoperability rate (IR)	Agent-based system II: Increase in interoperability compared to static system
1	68	50	18	74%	68	0	100%	36%
2	68	38	30	56%	68	0	100%	79%
3	68	42	26	62%	67	1	99%	60%
4	68	34	34	50%	67	1	99%	97%
5	68	29	39	43%	63	5	93%	117%
6	68	34	34	50%	68	0	100%	100%
7	68	41	27	60%	67	1	99%	63%
Average		38	30	56%	67	1	98%	79%

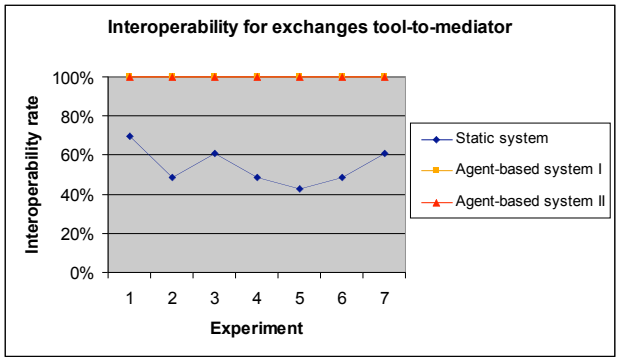


Figure 12. Interoperability rates for data exchanges from the tool agents to the mediator agent.

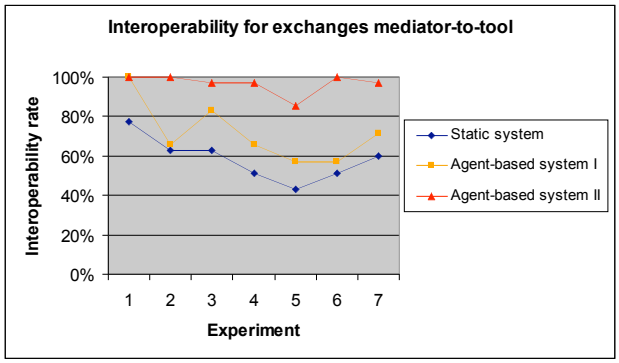


Figure 13. Interoperability rates for data exchanges from the mediator agent to the tool agents.

6.2 Efficiency

As the agents could establish increasing amounts of common ground during the course of every experiment, they were often able to exchange product models without having to engage in repair when interacting a second or third time. Table 5 summarises how the number of repair attempts was reduced for all repeated interactions occurring in every experiment. Both successful and failed attempts are included. All reductions of their amount are a result of increasing (mutual) knowledge from successful repair attempts. A mechanism for learning from failed repair attempts, however, has not been implemented. This kind of learning would include a notion of common ground about what is "un-common" and could avoid unnecessary repair attempts when the prospects of success are seemingly unrealistic. This would further improve the efficiency of the system.

In a few cases overgeneralisations of the mediator agent's neural network occurred, which led to incorrect FBS models of tool agents and consequently to product models represented in an incorrect format. Reactive tool agents were hard-coded to include in their negative feedback an explicit proposition about which input format they wish to receive. In this case, the mediator agent could easily repair its product model and re-structure or re-adjust its neural network to avoid future overgeneralisations. In contrast, reflective tool agents were not programmed to explicitly tell the mediator agent their input format. Given their own capacity to construct new formats instead of relying solely on the mediator agent to provide the appropriate format, such an explicit feedback was rarely needed. However, this has also prevented the mediator agent from adjusting its neural network to provide more stable and correct FBS models in future interactions with the same tool agent. This was the reason why the first agent-based system using only reactive tool agents happened to be more efficient in experiment 1 than the second agent-based system using some reflective tool agents.

7 Conclusion

Our experiments have shown that an agent-based approach to product data exchange can provide interoperability without pre-defined translators. It can generate common ground when it is needed – at runtime rather than prior to the process. This allows a better adaptation to the specificities of the individual tools in terms of the kind and the form of the product models needed. However, the generation of the "common standard" is not done *ab initio*, as agents can build on their memory of previous interactions represented as FBS models of other agents. Changes in the tool environment can be accommodated by updating this memory. Subsequent interactions can then be carried out in a more effective and efficient way.

The experiments suggest that the best results can be achieved when interactions involve reflective tool agents. In this way the driving mechanism of constructing the needed common ground – namely, situated cognition – can be used no matter if the mediator agent or the tool

agent is at the receiving end of the data exchange

Table 5. Reduction of the number of repair attempts for repeated interactions.

Experiment	Agent-based system I			Agent-based system II		
	no. repairs in first interaction	no. repairs in second interaction	reduction of repair attempts	no. repairs in first interaction	no. repairs in second interaction	reduction of repair attempts
1	4	0	100%	2	1	50%
2	11	4	64%	11	0	100%
3	9	2	78%	7	1	86%
4	6	3	50%	6	0	100%
5	7	4	43%	6	1	83%
6	12	5	58%	12	0	100%
7	9	4	56%	8	1	88%
Average	8.29	3.14	64%	7.43	0.57	87%

Endowing tool agents with more capabilities can also eliminate the role of a central mediator agent. A system architecture could thus be conceived consisting of a network of tool agents that autonomously establish virtual connections as learned patterns of their direct interactions.The experiments suggest that the best results can be achieved when interactions involve reflective tool agents. In this way the driving mechanism of constructing the needed common ground – namely, situated cognition – can be used no matter if the mediator agent or the tool agent is at the receiving end of the data exchange. Endowing tool agents with more capabilities can also eliminate the role of a central mediator agent. A system architecture could thus be conceived consisting of a network of tool agents that autonomously establish virtual connections as learned patterns of their direct interactions.

The common ground within our agent-based testbed includes only the representational aspect of product data ontologies, i.e. the vocabulary used to describe the design concepts. The concepts themselves and their relationships are fixed in a pre-defined class hierarchy. However, many interoperability problems are due to conceptual differences between the ontologies used by different tools. Future research should therefore find ways for agents to reach common ground on a conceptual level. It may use some of the work on automated discovery of mappings between different ontologies in the area of the Semantic Web (Noy 2004). We expect that such an extended form of interoperability will be based on the same dual approach as proposed in this paper, i.e. both on generative mechanisms used by individual agents and on their capacity to collaborate on the construction of a shared understanding.

Automating the generation of interoperability can potentially 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.

Letting situated agents negotiate a shared product model on the fly can potentially constitute a method for pushing future standardisation by ISO or IAI. After a product model has been agreed upon and successfully used by a set of agents, this model could 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. In addition, developers of new agents could use the emerging standard for engineering the agents' initial knowledge.

Acknowledgments

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

References

Bartlett F.C. (1932 reprinted in 1977) *Remembering: A Study in Experimental and Social Psychology*, Cambridge University Press, Cambridge.

Bickhard M.H. and Campbell R.L. (1996) Topologies of learning, *New Ideas in Psychology* **14**(2): 111-156.

Brennan S.E. and Clark H.H. (1996) Conceptual pacts and lexical choice in conversation, *Journal of Experimental Psychology: Learning, Memory, and Cognition* **22**(6): 1482-1493.

Cherniak C. (1986) *Minimal Rationality*, MIT Press, Cambridge, MA.

Clancey W.J. (1997) *Situated Cognition: On Human Knowledge and Computer Representations*, Cambridge University Press, Cambridge.

Clark H.H. (1996) *Using Language*, Cambridge University Press, Cambridge.

Clark H.H. and Marshall C.R. (1981) Definite reference and mutual knowledge, *in* A.K. Joshi and B. Webber (eds) *Linguistics Structure and Discourse Setting*, Cambridge University Press, Cambridge, pp. 10-63.

Clark H.H. and Murphy G.L. (1982) Audience design in meaning and reference, *in* J.F. Le Ny and W. Kintsch (eds) *Language and Comprehension*, North-Holland Publishing Company, Amsterdam, pp. 287-299.

Clayton M.J., Teicholz P., Fischer M. and Kunz J. (1999) Virtual components consisting of form, function and behavior, *Automation in Construction* **8**: 351-367.

Cohen P.R. and Levesque H.J. (1994) Preliminaries to a collaborative model of dialogue, *Speech Communication* **15**: 265-274.

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.

Dartigues C. and Ghodous P. (2002) Product data exchange using ontologies, *in* J.S. Gero (ed.) *Artificial Intelligence in Design'02*, Kluwer Academic, Dordrecht, pp. 617-637.

De Kleer J. and Brown J.S. (1984) A qualitative physics based on confluences, *Artificial Intelligence* **24**: 7-83.

Dennett D.C. (1987) *The Intentional Stance*, MIT Press, Cambridge, MA.

Dewey J. (1896 reprinted in 1981) The reflex arc concept in psychology, *Psychological Review* **3**: 357-370.

Eastman C.M. (1999) *Building Product Models: Computer Environments Supporting Design and Construction*, CRC Press, Boca Raton, FL.

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*, http://www.fipa.org/specs/fipa00061/SC00061G.pdf.

Fussell S.R. and Krauss R.M. (1989) The effects of intended audience on message production and comprehension: Reference in a common ground framework, *Journal of Experimental Social Psychology* **25**: 203-219.

Gero J.S. (1990) Design prototypes: A knowledge representation schema for design. *AI Magazine*, **11**(4): 26-36.

Gero J.S. (1999) Constructive memory in design thinking, *in* G. Goldschmidt and W. Porter (eds) *Design Thinking Research Symposium: Design Representation,* MIT, Cambridge, MA, pp. 29-35.

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.

Gero J.S. and Kannengiesser U. (2003) Function-Behaviour-Structure: A Model for Social Situated Agents, *in* R. Sun (ed.) *Workshop on Cognitive Modeling of Agents and Multi-Agent Interactions,* International Joint Conference on Artificial Intelligence 2003, Acapulco, Mexico, pp. 101-107.

Grosz B.J. and Sidner C.L. (1990) Plans for discourse, *in* P.R. Cohen, J. Morgan and M.E. Pollack (eds) *Intentions in Communication*, MIT Press, Cambridge, MA, pp. 419-444.

Haymaker J., Kunz J., Suter B. and Fischer M. (2003) Perspectors: Composable, reusable reasoning modules to automatically construct a geometric engineering view from other geometric engineering views, *Working Paper #WP082*, Center for Integrated Facility Engineering, Stanford University, Stanford, CA.

Horton W.S. and Keysar B. (1996) When do speakers take into account common ground? *Cognition* **59**: 91-117.

IAI (2003) *Industry Foundation Classes,* http://www.iai-international.org/iai_international/Technical_Documents/R2x2_final/index.html, International Alliance for Interoperability.

Karttunen L. and Peters S. (1975) Conventional implicature of Montague grammar, *in* C. Cogen, H. Thompson, G. Thurgood, K. Whistler and J. Wright (eds) *Proceedings of the First Annual Meeting of the Berkeley Linguistic Society*, University of California, Berkeley, CA, pp. 266-278.

Kuipers B. (1984) Commonsense reasoning about causality: Deriving behaviour from structure, *Artificial Intelligence* **24**: 169-203.

Maher M.L. and Gero J.S. (2002) Agent models of 3D virtual worlds, *in* G. Proctor (ed.) *ACADIA 2002*, California State Polytechnic University, Pomona, CA, pp. 127-138.

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.

Newell A. (1982) The knowledge level, *Artificial Intelligence* **18**(1): 87-127.

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

Norman D.A. (1983) Some observations on mental models, *in* D. Gentner and A.L. Stevens (eds) *Mental Models*, Lawrence Erlbaum Associates, Hillsdale, NJ, pp. 7-14.

Noy N.F. (2004) Semantic integration: A survey of ontology-based approaches, *SIGMOD Record* **33**(4): 65-70.

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

Russell S. and Norvig P. (1995) *Artificial Intelligence: A Modern Approach*, Prentice Hall, Englewood Cliffs, NJ.

Schegloff E.A., Jefferson G. and Sacks H. (1977) The preference for self-correction in the organization of repair in conversation, *Language* **53**: 361-382.

Schön D. and Wiggins G. (1992) Kinds of seeing and their functions in designing, *Design Studies* **13**(2): 135-156.

Stouffs R. and Krishnamurti R. (2002) Representational flexibility for design, *in* J.S. Gero (ed.) *Artificial Intelligence in Design'02*, Kluwer, Dordrecht, pp. 105-128.

Strandh S. (1979) *A History of the Machine*, A & W Publishers, New York.

Suwa M., Gero J.S. and Purcell T. (1999) Unexpected discoveries and s-inventions of design requirements: A key to creative designs, *in* J.S. Gero and M.L. Maher (eds) *Computational Models of Creative Design IV,* Key Centre of Design Computing and Cognition, University of Sydney, Sydney, Australia, pp. 297-320.

Szykman S., Fenves S.J., Keirouz W. and Shooter S.B. (2001) A foundation for interoperability in next-generation product development systems, *Computer-Aided Design* **33**(7): 545-559.

Tolman F.P. (1999) Product modeling standards for the building and construction industry: Past, present and future, *Automation in Construction* **8**: 227-235.

Wooldridge M.J. (2000) *Reasoning about Rational Agents*, MIT Press, Cambridge, MA.

Wooldridge M.J. and Jennings N.R. (1995) Intelligent agents: Theory and practice, *Knowledge Engineering Review* **10**(2): 115-152.

This is a copy of the paper: Kannengiesser, U and Gero, JS (2005) Agent-based interoperability without product model standards, *Computer-Aided Civil and Infrastructure Engineering* (to appear)