TOWARDS MASS CUSTOMIZED INTEROPERABILITY

UDO KANNENGIESSER AND JOHN S. GERO
Key Centre of Design Computing and Cognition
University of Sydney
Australia

Abstract. This paper describes an approach to interoperability in design projects that is based on computational agents customizing the representation of product data to individual design tools. The agents can augment their translator capabilities autonomously at runtime. This has the potential to lessen the need to manually pre-define an appropriate set of tool translators. The process of generating tailored product models uses an approach to organizing agent knowledge that limits computational complexity when large numbers of tools are involved. We report the implementation and evaluation of an agent-based system that demonstrates fundamental features of mass customized interoperability. Results include an average reduction of manual work by 88% using this approach.

1. Introduction

Mass customization is a term from business management to describe a strategy that involves providing products for a relatively large market while tailoring the products to individual customers (Pine 1993). The need for mass customization has increasingly been recognized to cater for strongly heterogenous markets with specialized or fluctuating customer demands. This poses the challenge for manufacturing firms to minimize the resulting tradeoffs in terms of cost, time and quality.

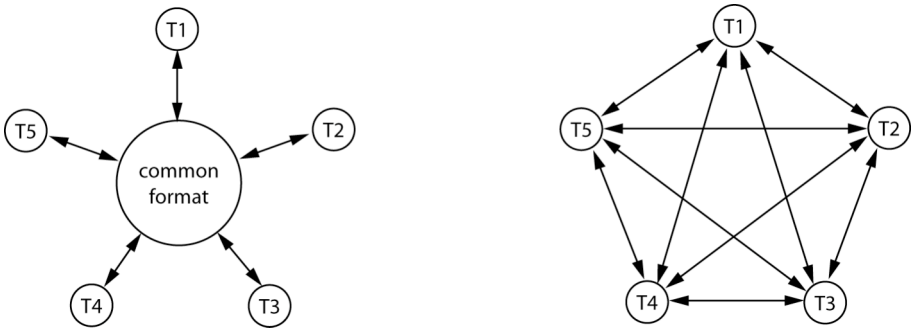
Supplying product models to a large number of different design tools can be seen as an activity that requires something similar to mass customization. Here the "market" consists of different design tools that demand different formats in which the product is to be represented. This heterogeneity is a consequence of the stand-alone manner in which today's computational tools have been implemented to perform highly specialized analysis and documentation tasks. The need to integrate these tools in a single design project has created the issue of interoperability, i.e. the need to move data from one representation of a product to another to allow other computational

processes to operate on it. The annual cost resulting from poor interoperability in the American automotive industry has been estimated to be one billion US dollars (NIST 1999).

1.1. APPROACHES BASED ON MARKET REGULATION

Most approaches to interoperability are based on "market regulation": They impose interfaces on every tool to translate all native data into a standard format. This circumvents the need for mass customization as all the tools now share a common format that allows for mass production of product models. Figure 1(a) shows this approach schematically. The relationship between the number of design tools and the number of interfaces required to translate the tools' data into or from the common format is linear, as N design tools need 2·N translators.

A customized approach based on direct translations between individual tools as shown in Figure 1(b) has generally been rejected due to issues of combinatorial complexity. Specifically, there is a quadratic relationship between tools and translators, as N tools would need N·(N-1) translators. As writing and maintaining translators is an arduous task done by human software developers, this architecture leads to prohibitive costs when large numbers of tools are to be made interoperable.



(a) Standardized data transfer using 2·N translators (2·5=10)

(b) Customized data transfer using N·(N-1) translators (5·4=20)

Figure 1. The number of translators (represented as arrowheads) required for a system of N=5 tools using (a) standardized or (b) customized data transfer.

One of the best-known interoperability efforts based on standardized data transfer is the ISO 10303 standard, informally known as STEP (STandard for the Exchange of Product model data) (ISO 2005). STEP aims to standardize product models used in the automotive, aerospace and shipbuilding industry. A similar 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). There have been some reports on the use of STEP and IFCs in various individual design projects. However, these approaches have had great difficulties to completely regulate the "market" of design tools, and for most projects the standardized solution is only partially available. This is due to a number of reasons.

Product model standardization is an on-going 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 very slow and always lags behind practice 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 in 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 competitors' software (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). 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 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 organizational changes over the duration of a project that requires the integration of new tools or new exchanges among present tools.

1.2. APPROACHES TOWARDS MORE MARKET ORIENTATION

The limited success of market regulation has led to a number of more market-oriented ways of exchanging product data. These approaches require

only a minimal core set of pre-defined, abstract data types. These data types can then be extended by designers rather than software developers. This moves the generation of interoperability closer to the design project, making it more responsive to the particular needs of the tools.

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. 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 standardized 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 at the possibility of specializing 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 in subsequent design projects. However, there is no mechanism to automate retrieving or generating suitable Perspectors or organizing them into the dependency network.

All these approaches involve the human designer as the generator of a common format and of the appropriate translators. This process has only been automated partially and relies mainly on user interfaces to facilitate the remaining manual tasks. Our approach aims to model the process of

generating interoperability more explicitly to lessen the need for human intervention. It is based on a view of interoperability as the emergent result of interaction rather than as a pre-defined state of affairs. Here interaction occurs between generators, product models and the context for which the product models are generated. This allows interoperability to become more flexible and adapted to the specific needs of the interaction and the actors involved. Such an interaction-based approach is consistent with existing work on situated cognition in general (Clancey 1997), and, as we will show in Section 2, human communication in particular. Computational constructs that are capable of dealing with interactions are agents. Agents are software systems that are able to act on their environment autonomously according to their individual goals and beliefs. In turn, their goals and beliefs are adapted to their environment. This bi-directional relationship between agents and their environment can be used for a computational model of adaptive interoperability that has the potential to mass customize product models, in the way depicted in Figure 1(b).

This paper is structured as follows. Section 2 develops the conceptual basis of our approach using a cognitive science view of human communication and a knowledge representation schema. Section 3 describes the implementation of an agent-based system for mass customizing interoperability. Section 4 presents the experimental setup and the case study used to evaluate our approach. The results of our experiments included an average reduction of manual work by 88% when using these automated updates of translators. Section 5 presents some conclusions.

2. A Model of Mass Customized Interoperability

This Section commences with an outline of cognitive science research in communication between human agents that is known as the common ground approach, which provides one of the foundations of our work. A representation schema for agents is then presented and used as a basis for a model of common ground. Finally, a hypothesis is developed about the potential of this model to support mass customized interoperability.

2.1. THE NOTION OF 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. Common ground among two agent A and B on a proposition p has 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? Clark and Marshall (1981) have advocated using a heuristic to avoid recursion. Their co-presence heuristic 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.

Communicative actions are not 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.

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 the 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.

We can extract a set of fundamental concepts of the common ground approach and map them onto an approach to interoperability. This approach is described in terms of a number of desired features:

1. *Customized communication*: Messages are formed in a way that is tailored to the receiver of that message. Different agents can thus be provided with different representations according to their individual needs.
2. *Distributed translation*: As both the sender and receiver use their knowledge of each other to reach common ground, communication is based on the union rather than only the intersection of the agents' translator capabilities. This increases the chances of successful communication.
3. *Robustness*: The ability of agents to provide feedback and to engage in repair for resolving communicative problems collaboratively increases robustness. Most interoperability approaches, in contrast, do not provide similar mechanisms for automated recovery from format incompatibilities.
4. *Autonomous, runtime increase of translator capabilities*: The agents increase their common ground by learning from their interactions.

This is the basis for increasing efficacy and efficiency in the agents' future interactions. No conventional design tool has been developed to date that can similarly update its own translator capabilities.

Feature 1 describes the fundamental difference between common ground and interoperability by market regulation. Features 2 and 3 relate to the overall success or quality of this approach. Feature 4 is a basis for time- and cost-efficiency, which are necessary requirements for mass-customized interoperability. Further research is needed to bridge the gap between all these capabilities and the current state-of-the-art in interoperability. Our earlier work (Kannengiesser and Gero 2006) demonstrated a way to implement aspects of features 2 and 3. The present paper describes the basics of an approach to realising feature 4.

2.2. THE FUNCTION-BEHAVIOUR-STRUCTURE SCHEMA

A key to integrating common ground in computational agents is their ability to model themselves as well as other agents (Gero and Kannengiesser 2003). Gero's (1990) function-behaviour-structure (FBS) ontology provides a schema that supports these models. In particular, function (F), behaviour (B) and structure (S) provide a general framework for representing knowledge about any existing (physical) or imaginary object. Their definitions are as follows:

- *Function* (F) describes the teleology of an object, i.e. "what the object is for".
- *Behaviour* (B) describes the attributes that are derived or expected to be derived from the structure (S) of an object, i.e. "what the object does".
- *Structure* (S) describes the components of an object and their relationships, i.e. "what the object consists of".

The FBS schema has originally been developed to model design artefacts. For example, the function (F) of a car is usually to transport people from one place to another, its behaviour (B) may include fuel consumption, and its structure (S) can be viewed as the composition of a number of mechanical parts. Gero and Kannengiesser (2003) have shown that the FBS schema can also be applied to agents. For example, the function (F) of a design agent may be to perform stress analyses, its behaviour (B) may be the speed with which it produces results, and its structure (S) may include its physical (or virtual) embodiment as well as its knowledge and goals.

An agent may use the FBS schema to model its individual experiences with a specific object (or agent). We will refer to the resulting models as specialized FBS models. Once a number of experiences with different objects has been gained and represented as specialized FBS models, the agent is able to generalize by clustering sets of like experiences. The

resulting cluster of FBS models is called a generalized FBS model. Generalization reduces computational complexity as it eliminates redundant information.

When the agent needs to recall a particular object, it can use specialized FBS models to directly access information that is unique to that object as well as generalized FBS models to derive information that is typical for that object's class. Gaining and recalling experiences often requires merging information drawn from different models as well as transforming one model into another. These processes are readily supported by the FBS schema as it provides a set of constructs to represent all objects uniformly, irrespective of their level of generality.

The capacity provided by the FBS schema to use both specialized and generalized knowledge in an integrated way has been employed to model how designers are able to start designing when only incomplete information is available about the function, behaviour and structure of the object to be designed. Here generalized FBS models called design prototypes (Gero 1990) are used to instantiate specific FBS models that represent preliminary design solutions. The information derived from design prototypes can be regarded as default information about the hypothesized design founded on generalized knowledge constructed from similar previous designs.

2.3. A MODEL OF COMMON GROUND

The FBS knowledge representation can now be used to model Clark and Marshall's (1981) co-presence heuristics for inferring common ground. Here all knowledge that is considered part of common ground is co-present either directly in the current interaction (as personal common ground) or indirectly via generalized experience with previous interactions with the same or similar agents (as communal common ground). Personal common ground uses a specialized FBS model of the other agent constructed from the current interaction with that agent, while communal common ground uses a generalized FBS model that is retrieved using cues provided by the specialized FBS model.

Figure 2(a) shows the (specialized) FBS models that agent 0 has constructed of itself as well as of agents it has previously interacted with (1 – 4). Figure 2 represents the relationship between FBS_0 and $FBS_{1\text{-}4}$ with the latter being nested in the former. The FBS models are represented as circles of different sizes to indicate the different amounts of knowledge 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 agent 0 itself.

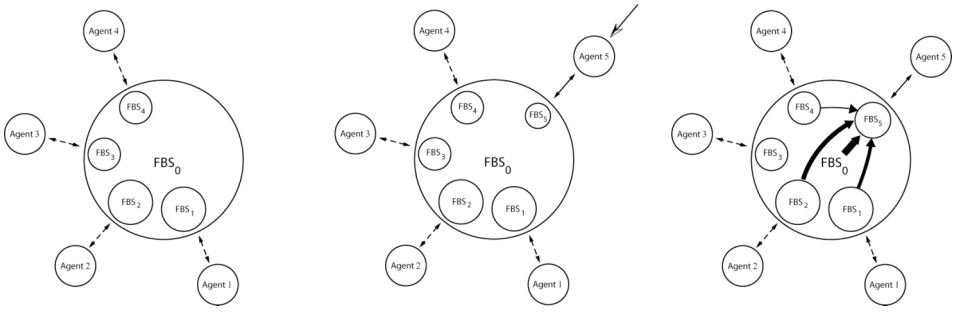


Figure 2. New FBS models are constructed using generalizations of previously constructed FBS models. The size of the circle for each FBS model is an indication of the amount of grounding of this FBS model of the other agent. The width of the arrows is an indication of the confidence of the potential applicability of the originating FBS model in constructing or supporting the FBS model of a new agent. Double-headed arrows represent agent interactions (previous interactions in dashes). (a) The set of FBS models at the outset; (b) a new FBS model created in response to the interaction with a new agent (agent 5); (c) the knowledge sources used to add generalized information to the new FBS model.

Let us now assume that agent 5 comes into play and starts interacting with agent 0, Figure 2(b). Agent 0 has little knowledge about that agent. Its FBS model of agent 5 might only be founded on the very first moments of that agent's appearance if it has never interacted with it before. In order to complement this FBS model (personal common ground), agent 0 has to derive additional knowledge from experiences generalized from interactions with agents that are similar to agent 5 (communal common ground). Figure 2(c) depicts the provenance of this generalized 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. This way of using generalized knowledge as a supplement for insufficient specialized knowledge is similar to deriving default assumptions from design prototypes as described in Section 2.2.

Although generalized knowledge is a useful source for constructing default assumptions that inform intended interaction, some of it may turn out to be incorrect. This becomes evident usually via negative feedback provided by the other agent. The agents will then engage in repair to collaboratively establish new common ground that is integrated in the agents' experience via modifications (usually involving specializations) of their FBS models of each other.

2.4. MASS CUSTOMIZED INTEROPERABILITY

Using a set of FBS models that are as generalized as possible and as specialized as needed is not only useful to deal with dynamic, ill-defined agent interactions, but also to manage computational complexity. This issue would arise for agents that are not able to generalize as they would have to use a new specialized FBS model of every individual agent in the system, Figure 3. It can be seen that this scenario resembles the approach based on pre-defined, specialized translators shown in Figure 1(b). Although the agents' FBS models of each other fulfil a slightly different role than conventional tool translators, they would share a similar combinatorial problem. As the ability to construct generalized FBS models reduces the number of specialized FBS models, our approach has the potential to handle this issue.

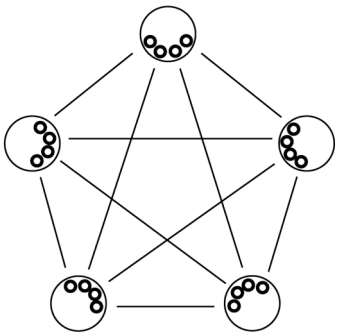


Figure 3. Worst-case scenario: N·(N-1) FBS models within a system of N agents (here N = 5).

The applicability of generalization ultimately depends on the degree of heterogeneity of the environment. In a group of interacting agents that are largely similar, it can be expected that the number of specialized FBS models to be stored would be low. A group of very dissimilar agents, in contrast, will require a high number of specialized FBS models. We expect that since most domains are structured, this allows for generalization.

The structure of a domain is independent of its size. As an agent usually knows only part of a domain, there is a relationship between the structure that the agent perceives and the extent to which the agent has explored the domain by interacting with it. Let us assume an agent that has only little knowledge about its environment or domain. When the agent starts exploring this environment, it will encounter many things that are novel with respect to its limited previous experience. As a result, the number of specialized FBS models will grow. As the agent continues exploring and learning about the domain, it will find fewer things that it has not come

across before. This allows integrating most new experiences into generalized FBS models. Figure 4 shows qualitatively how the rate at which new (specialized) FBS models are created by an agent is expected to decrease with its increasing knowledge about the domain (here measured as the number of agents it has interacted with). The number of specialized FBS models eventually converges to a value that can be seen as an indicator for the system's heterogeneity. This adaptiveness is not available in systems that do not have the capacity to generalize such as those using pre-defined, direct translators. Here the number of specialized FBS models (or statically customized translators) will always rise at a constant rate.

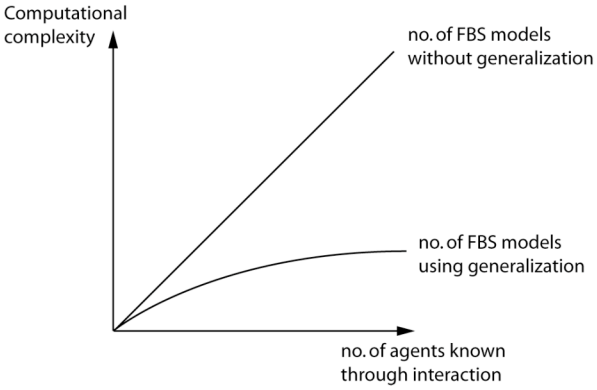


Figure 4. Comparison of the expected increase in computational complexity in an agent and a static tool when the number of agents in the system rises.

The learning path of an agent exploring its domain is rarely completely linear. The agent may have clustered experiences at one point in time that turn out to be in conflict with its interactions at a later point in time. The domain may also be dynamic, with new agents entering or existing agents changing their properties or preferences. In all these cases, an agent must adapt its knowledge to the current situation. This involves specializing and eventually re-generalizing FBS models. The uniform way in which all its models are represented readily allows for these modifications.

3. Implementation

This Section outlines the implementation of an agent-based system that we have used to evaluate our approach. Parts of this implementation have been reported in Kannengiesser and Gero (2006). In this paper we will give an overview of the overall system and emphasize the way in which agents organize their memory in terms of FBS models.

Our implementation is limited to the representational aspect of interoperability, i.e. the individual tools differ only in the vocabulary of a

format used to represent the same ontological concepts, without polysemy (multiple meanings). While this does not address a range of other interoperability issues such as those based on differences in the tools' internal models (e.g. how particular geometric shapes are defined), it focuses on a class of translation issues that serves as a sufficient vehicle for the conceptual ideas of our approach.

3.1. OVERALL SYSTEM

Our implemented system uses agents to exchange product data among a set of computational design tools, each of which modifies or adds to that product data. Specifically, every tool is modelled as being "wrapped" by an agent. Our implementation uses simulations of tools as simple mappings from an input to an output model. This allows us to focus on interoperability issues among the tools rather than on issues of internal data transformations within the tools. An agent can feed its encapsulated tool with input data it receives from other tools (agents) and can transfer the resulting output data to other tools (agents). All product data is conveyed as the content part of FIPA ACL[1] messages (FIPA 2004). ACL messages are transferred in a point-to-point fashion using a communication infrastructure to which all agents are connected via their sensors and effectors. The communication infrastructure and the sensors and effectors are written in Java.

One of the wrapper agents encapsulates a project database system as a special kind of "tool". This system fulfils two tasks. First, it keeps track of the current state of the product as the other design tools add or modify subsets of the product data. Second, it monitors the current state of the design process with respect to a given project plan. The project database system can fulfil these tasks only if it is in some way involved in every data exchange occurring in the project. This can be established by assigning to the agent encapsulating that database system the role of a mediator (Wiederhold 1992) that is interposed in the data transfers between all other agents. This agent will from now on be referred to as the "mediator agent", while all the other wrapper agents will be referred to as "tool agents".

Using a mediator agent centralizes data exchange in the agent-based system on a physical level. On a conceptual level, this arrangement may be viewed as equivalent to the existence of a neutral, intermediary format among the tool agents, mapping directly onto the mediator agent's native format. However, the initial translator capabilities (in terms of the common ground among the agents given at the outset of the experiments) will be limited to allow only a subset of required data transfers to be interoperable. For the remaining data transfers, the mediator agent's format will not be

[1] Agent Communication Language

available to be used as a "neutral" format unless the agents use their ability to self-update the initial set of mappings between their formats. This distinguishes our system from neutral-format approaches.

The experiments are conducted using two different types of agents. The first type is what we call a "reactive" agent. This agent has only minimal autonomy and differs from a simple tool just in its ability to communicate ACL messages. This allows the agent to engage in repair (see Section 2.1), which includes the ability to provide feedback, answer queries about format mappings or request re-representation of a product model in a particular format. Reactive agents do not have the ability to learn new formats and can understand and send product models only in one fixed format. The second type of agent is called a "reflective" agent. This agent has more autonomy and capabilities than a reactive agent. Its architecture includes a memory system that represents FBS models of other agents and mappings between different formats. Besides its ability to communicate ACL messages, it can construct specialized and generalized FBS models of other agents. Reflective agents can also learn new formats and integrate them into their FBS models.

3.2. PRODUCT MODEL GENERATION

The knowledge used by a reflective agent to generate product models is partitioned into application knowledge and social knowledge, Figure 5.

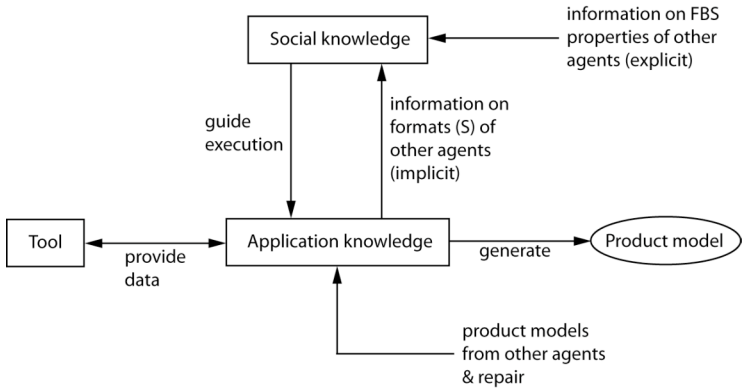


Figure 5. Using application knowledge and social knowledge to generate customized product models.

Application knowledge contains mappings between different formats and uses them to translate between different product model formats. We will use the term "product model" here only in the sense of an external product representation produced by an agent. This distinguishes the term from the internal product representation used by the tool and constrains the meaning of the agent's "product model generation" to include only the process of

creating rather than interpreting (parsing) external product models. Parsing product models generated by other agents provides an internal data model for the tool to work on and potentially allows the agent to augment its application knowledge by learning new formats or new parts of known formats.

Social knowledge is linked to application knowledge to guide the selection of format mappings that suit the needs of the agent's current interaction. This customizes product models to the individual formats required by the other agents. New social knowledge can be gained in two ways. First, other agents may introduce themselves by directly communicating explicit information about their FBS properties. Second, information about the format of another agent (S property), which is implicitly available in the product models generated by that agent, may be extracted by the application knowledge and then passed on to the agent's social knowledge.

Both kinds of knowledge are derived from the agent's interactions with one another. An initial amount of knowledge, however, will be necessary to be given in advance when spawning the agent.

3.2.1. Application Knowledge

Application knowledge is implemented in the rule-based language Jess[2]. It consists of two parts: a dictionary represented by a collection of Jess facts and a reasoning system represented by a set of Jess rules.

The dictionary contains a list of all the variables for a particular product and maps each of them onto a set of alternative names (synonyms). In addition, variables are grouped into categories described by a super-name (hypernym). For example, the variables "inner diameter" and "outer diameter" share the hypernym "diameter". The approach using synonyms and hypernyms has been inspired by the WordNet database (Miller 1995). Every variable name is linked to one or more descriptors denoting specific formats. The dictionary can be regarded as the agent's total translator capabilities. Rather than being a set of separate, fixed translators, this dictionary is organized as a flexible network that can readily add new variable names as well as new mappings to format descriptors. This allows the agent to autonomously increase its translator capabilities at runtime.

The reasoning system uses the dictionary and (format) expectations from the agent's social knowledge to perform product model translations. Let us start presenting how product models generated by another agent are parsed. Specifically, we are interested in those product models for which the current set of mappings in the dictionary is insufficient for complete parsing. In this

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

case, the reasoning system has a number of generative methods available to infer the required mappings:

- 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".

If these methods fail to generate the missing mappings, the other agent can be asked to provide a set of alternative (synonymous) or supplemental (hypernymous) representations for any unknown variable names. This is repeated until no more alternatives are available or until the product model is finally understood. In the former case, human intervention is required to manually translate the product model. In the latter case, the agent adds to the dictionary all confirmed mappings that originated from its own generative methods or that were communicated by the other agent.

Besides adding individual variable names to its dictionary, the agent's reasoning system categorises the complete set of variable names used in the product model as a particular format. If this set is consistent with any existing format, it is subsumed into that format. If the set is not consistent with any of the existing formats, a new format is created for the variable names in that set. The (existing or new) format into which the variable names have been categorised is then compared to the initially expected format. If the two formats are not identical, the agent's social knowledge will be revised as described in Section 3.2.2.

The process of generating a product model for another agent is then straightforward. Guided by expectations from the agent's social knowledge, the reasoning system selects and executes the appropriate set of mappings to represent a product model that is likely to be understood by the other agent. If such mappings are not available for parts of the product model, the agent simply uses its own native format to represent those parts. While this strategy will fail to establish interoperability with a reactive agent, it may succeed when communicating with a reflective agent capable of using generative parsing methods and repair.

3.2.2. Social Knowledge

The core element of the agent's memory system is a neural network, more specifically an Interactive Activation and Competition (IAC) network (McClelland 1981) written in Java. It is used as a mechanism to represent the agent's FBS models as it has the capacity to retrieve both generalized and specialized information based on stored knowledge of specifics. IAC networks organize a set of neurons into different pools (cohorts). Neurons within the same cohort are connected to one another via mutually inhibitory connections, i.e. the neurons in this cohort compete. In contrast, connections between neurons in different cohorts are mutually excitatory. Every neuron computes its activation level as a function of the total of all excitatory and inhibitory input it receives from the neurons it is connected to. The neuron then spreads this activation back to the other neurons, which, in turn, is used as their excitatory or inhibitory input depending on whether the connection strengths are positive or negative. All neurons update their activations simultaneously in time units called cycles. After a number of cycles the network eventually reaches a state of equilibrium in which no more significant changes of activation occur.

Figure 6 shows an example of an IAC network. It represents memories of agents as FBS models, where each agent is represented by a neuron ("instance neuron") in the central cohort in Figure 6, and its FBS properties are represented by neurons ("property neurons") in the other cohorts. The implementation uses five different cohorts to represent the following five classes of FBS properties of an agent.

- role in the project (F)
- task output (B)
- knowledge of an input format (S)
- knowledge of an output format (S)
- vendor (objectified relationship between different FBS models)

To retrieve an FBS model of a specific agent, one or more neurons are activated to represent "cues" for the memory of that agent and the network is then cycled until equilibrium is reached. The final activations within every cohort are then compared to determine the "winning" neurons that represent the constructed FBS memories. For example in Figure 6, if we want to retrieve the FBS model of agent 2, we can activate the corresponding instance neuron (i.e. neuron 2). The activation spreads to property neurons in the other cohorts and is then sent back to the instance neurons, which in turn affects the activation of neurons in other cohorts, and so on. When the network settles into equilibrium, it is likely that the neurons for the values "F1", "B3", "IF2", "OF3" and "V3" will be the winners within their respective cohorts. This would instantiate an FBS memory that is exactly the same as the initially encoded experience. As some neurons are also used by other experiences whose properties then receive part of the activations during the cycles of the network, different FBS properties may blend in and

change the memory with respect to the initial experience. This is called an over-generalization of the IAC network. The effects of over-generalization can be limited by choosing appropriate parameters for the activation function and appropriate connection strengths.

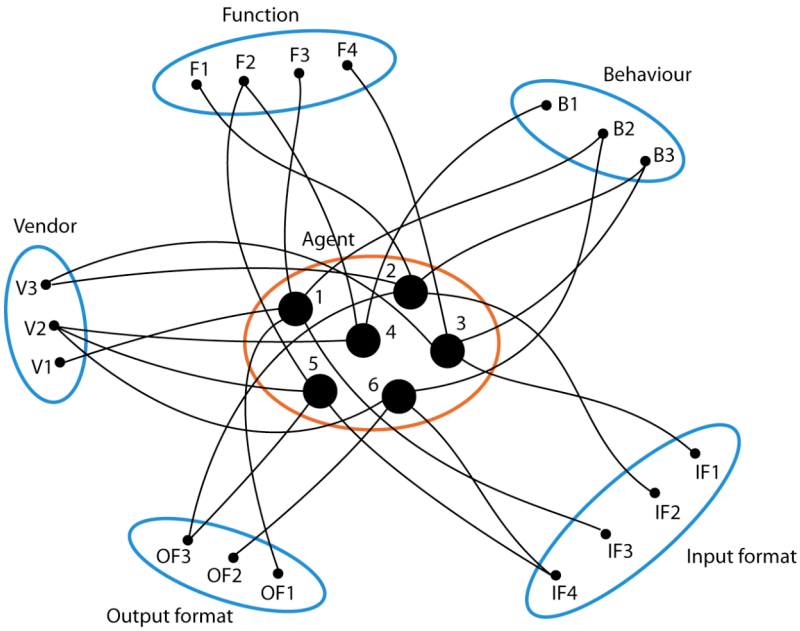


Figure 6. An IAC network (after McClelland 1981) representing FBS models of agents. The larger dots are instance neurons representing agents, while the smaller dots are property neurons representing agent properties. Only the excitatory connections among the neurons are depicted; the inhibitory connections (within the cohorts) have been omitted for simplicity.

The ability of the IAC network to spread activation across different individual experiences implements the use of generalized FBS models to complement incomplete memories. Figure 6 shows that the instance neurons representing agents 3 and 4 are not connected to any of the property neurons representing formats, except for the connection between the agent "3" neuron and the input format "IF1" neuron. This can be interpreted as a situation in which only incomplete information has been grounded initially about agents 3 and 4. As instance neuron "3" shares the properties "B3" and "V3" with instance neuron "2" and as instance neuron "4" shares the properties "F2" and "V2" with instance neuron "5", it can be expected that the network will generalize from these similar experiences and activate the formats associated with agents 2 and 4 and agent 5, respectively. These assumptions are likely to be correct in most cases, especially since particular

vendors (here "V3" and "V2") often use the same formats across different tools (or agents).

The kind of generalization described so far is concerned with sharing individual FBS properties among the instances (i.e. instance neurons) in the IAC network. The memory system also includes a control mechanism that allows sharing of individual instances among similar experiences to limit the size of the IAC network. This mechanism is used when a new agent enters the system and starts interacting by conveying information about some of its FBS properties. The mechanism then checks if a memory of another agent with the same properties already exists. This requires running the IAC network using the initial set of FBS properties as cues. If such a memory exists, the new experience can be integrated simply by creating a reference to the instance neuron representing the existing memory. If no such memory exists, a new instance neuron must be instantiated and integrated into the IAC network by a set of new excitatory and inhibitory connections.

The required adaptability of the agent's generalized and specialized FBS models to its current view of the world is realized by a set of methods that support both ungrouping and modification of FBS models. Specifically, revisions of FBS models are achieved by one of the following modifications of the IAC network.

1. Removal of the reference of an experience to the memory in which it had been subsumed and instantiation of a new instance in the network to represent that experience. This requires creating a new instance neuron and new connections in the network.
2. Adding a new connection between an instance neuron and a property neuron. This may involve removing a previous connection to a competing property neuron that is regarded as "obsolete".
3. Increasing the connection strength between an instance neuron and a property neuron to compensate for the impact of over-generalization.

Once a new instance has been added to the network, it cannot be removed at a later point in time. It contributes to generalizing the FBS properties of other instances and new experiences can be subsumed in it. This increases the potential of the IAC network to generalize in future situations.

4. Experiments and Results

4.1. OBJECTIVES

Previous experiments and analyses of the implemented agent-based system reported in Kannengiesser and Gero (2006) have been oriented to the performance of the system as a whole. In this paper we investigate performance at the agent level to evaluate if the requirements for mass

customization of product models are met. Our experiments will concentrate on agents that generate product models for other agents, specifically dealing with two aspects:

1. extensibility of the agents' dictionaries
2. scalability of the agents' memory systems

The first aspect examines the extent to which agents can learn new formats, which corresponds to a reduced need for humans to manually update translators. Here an agent, the test object, will be endowed with an initial set of mappings that can only inadequately cater for all the required formats in the multi-agent system. We have chosen the mediator agent as the test object as it is involved in all interactions. The experiments will be conducted in two stages. The first stage uses only reactive tool agents to examine how much of the learning is done by the mediator agent. The second stage replaces reactive agents by reflective ones, using a reflective (learning) agent at each end of an interaction. This will indicate the potential to improve the results by a more collaborative approach to translation and learning.

The second aspect concerns the potential issue of computational complexity as discussed in Section 2.4 when an agent has to generate product models for large numbers of other agents (tools) in the system. This will give evidence about the ability of the agent to use generalized FBS models in addition to specialized FBS models to represent the other agents. We have again chosen the mediator agent as the test object. We will measure the rate at which new specialized FBS models (represented by IAC network instances) must be constructed for a new set of agents in the system, which is expected to fall. In contrast, an agent without this ability would have to construct the same number of FBS models as new agents in the system, i.e. the gradient of this function is 1. The difference between the two gradients can be seen as a performance improvement (PI) that is defined as:

$$PI = 1 - \frac{no.\ new\ specialised\ FBS\ Models}{no.\ new\ agents}$$

PI is expected to increase with the number of new agents in the system and to converge to a value that is lower than 1. The experiments aim to show this by gradually increasing the number of initial FBS models given to the mediator agent. The FBS properties of each of the initial agent memories (FBS models) as well as of the new agents will be selected randomly to create the worst-case scenario of a missing domain structure. This allows finding the lower bound for the agent's ability to generalize FBS models. All tool agents will be reactive and the mediator agent will be given all the mappings required, as the focus of the experiments here is on memory efficiency rather than on translation efficacy.

4.2. CASE STUDY

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 7. We have modelled this product using 299 variables describing its structure and behaviour. The components of the turbocharger are organized into a 3-layer tree structure, Figure 8.

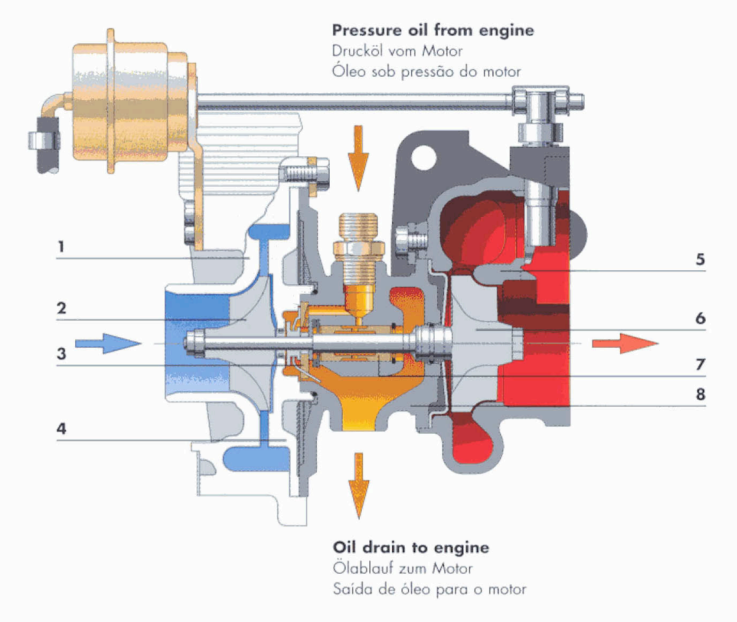


Figure 7. A turbocharger for passenger cars: 1. compressor housing, 2. compressor wheel, 3. thrust bearing, 4. compressor backplate, 5. turbine housing, 6. shaft & turbine wheel assembly, 7. bearing bushing, 8. centre housing (Source: BorgWarner Turbo Systems)

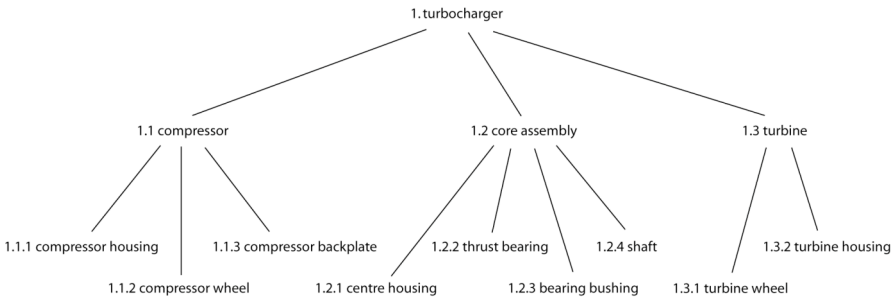


Figure 8. Assembly structure of the turbocharger.

All product models are represented by content, Figure 9. 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 "PART compressor, AF5500 0.176 m^3/s", "PART compressor, AF_5500 0.176 m^3/s" or "PART compressor, V_RED_5500 0.176 m^3/s". We have defined 7 different formats covering each of the 299 product variables.

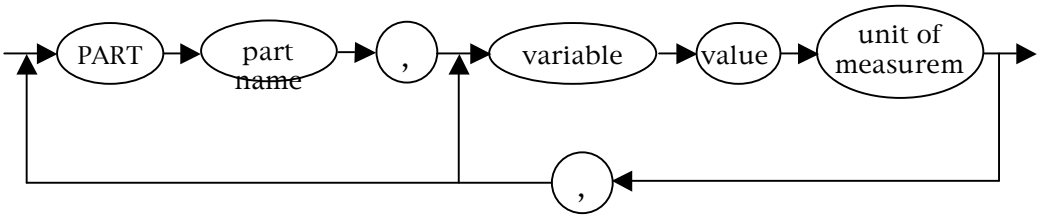


Figure 9. Syntax diagram for a product model.

Figure 10 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 optimizing the product's performance after evaluating the prototype. Tables A.1 and A.2 in the Appendix show the classes of input data required for every tool agent to perform its individual task as well as the resulting classes of output data.

Figure 11 depicts a set of exemplary communications between agents A-0 and A-10 to transfer product data required for a thermal stress analysis of the turbine wheel (performed by A-10). The individual messages are represented as Jess facts. The process is started by agent A-0 sending a product model to agent A-10 describing parts of the geometry of the turbine wheel and the inflow temperature of the turbine for a range of engine speeds (message 1 labelled in Figure 11). In the example, A-10 does not know some of the format but succeeds in constructing an assumption using its generative methods. A-10 then initiates repair by seeking reconfirmation from A-0 for the conjectured synonymous relationship between the two variable names "LO1" and "LA_1" (message 2). After receiving a positive response (message 3), A-10 can parse the product model and perform the thermal

stress analysis. The results of this analysis are returned to A-0 as a new product model (message 4).

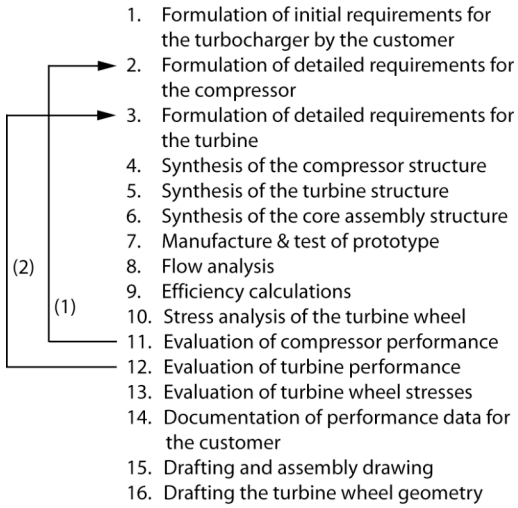


Figure 10. A project plan for simulating the design of a turbocharger with 2 iterations occurring during the process.

4.3. RESULTS

4.3.1. Automated Updates of Translators

The first set of experiments is run with a domain-typical distribution[3] of FBS properties among the agents (see Table A.3 in the Appendix). In some of the experiments, a set of agents will replace their input or output formats against others, which they will do after the first iteration of the design scenario (see Figure 10). This simulates the integration of new tools during the project and allows testing the mediator agents' ability to adapt to these changes.

[3] based on the tools' vendors, functions and behaviours

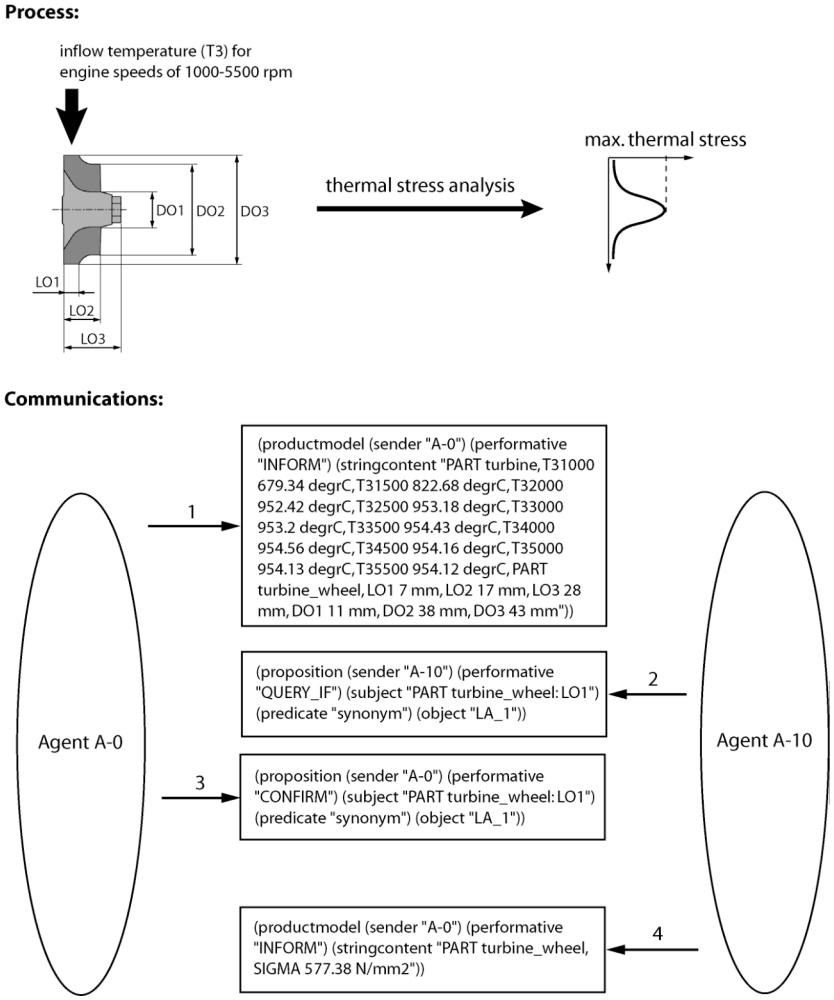


Figure 11. A set of communications between two agents (A-0 and A-10) required for thermal stress analysis.

Six different experiments have been run with the implemented system, each of which uses different sets of initial format mappings in the mediator agent's application knowledge. Table 1 shows parts of the format knowledge initially given to the mediator agent in the first of the six experimental runs. It can be seen that most gaps in that knowledge (indicated by the shaded areas in Table 1) could not be removed during the run. Exceptions are parts of formats F_2 and F_3 that were learned when applying generative parsing methods. In one case, this new knowledge was sufficient to be reused for transferring product data to a tool agent (A-15) that would not have been possible without such runtime increase of knowledge.

TABLE 1. Gaps, indicated by the shaded areas, in the mediator agent's initial format knowledge in experiment 1.

= knowledge gaps that could be removed autonomously
= knowledge gaps that could not be removed

Product part	Meaning	Variable name							Class
		Format F_1	Format F_2	Format F_3	Format F_4	Format F_5	Format F_6	Format F_7	(Hypernym)
generic	number	ANZ	NOM	N	NO	ANZ	NO	N	organisational
	efficiency_1000	WGRAD1000	EFF1000	ETA1000	EFF1000	WGRAD_1000	EFF_1000	ETA_1000	ratio
	efficiency_1500	WGRAD1500	EFF1500	ETA1500	EFF1500	WGRAD_1500	EFF_1500	ETA_1500	
	efficiency_2000	WGRAD2000	EFF2000	ETA2000	EFF2000	WGRAD_2000	EFF_2000	ETA_2000	
	efficiency_2500	WGRAD2500	EFF2500	ETA2500	EFF2500	WGRAD_2500	EFF_2500	ETA_2500	
	efficiency_3000	WGRAD3000	EFF3000	ETA3000	EFF3000	WGRAD_3000	EFF_3000	ETA_3000	
	efficiency_3500	WGRAD3500	EFF3500	ETA3500	EFF3500	WGRAD_3500	EFF_3500	ETA_3500	
	efficiency_4000	WGRAD4000	EFF4000	ETA4000	EFF4000	WGRAD_4000	EFF_4000	ETA_4000	
	efficiency_4500	WGRAD4500	EFF4500	ETA4500	EFF4500	WGRAD_4500	EFF_4500	ETA_4500	
	efficiency_5000	WGRAD5000	EFF5000	ETA5000	EFF5000	WGRAD_5000	EFF_5000	ETA_5000	
	efficiency_5500	WGRAD5500	EFF5500	ETA5500	EFF5500	WGRAD_5500	EFF_5500	ETA_5500	
turbine	inflowpress_1000	DRUCKZU1000	INPRESS1000	P31000	P31000	DRUCKZU_1000	INPRESS_1000	P3_1000	inpressure
	inflowpress_1500	DRUCKZU1500	INPRESS1500	P31500	P31500	DRUCKZU_1500	INPRESS_1500	P3_1500	
	inflowpress_2000	DRUCKZU2000	INPRESS2000	P32000	P32000	DRUCKZU_2000	INPRESS_2000	P3_2000	
	inflowpress_2500	DRUCKZU2500	INPRESS2500	P32500	P32500	DRUCKZU_2500	INPRESS_2500	P3_2500	
	inflowpress_3000	DRUCKZU3000	INPRESS3000	P33000	P33000	DRUCKZU_3000	INPRESS_3000	P3_3000	
	inflowpress_3500	DRUCKZU3500	INPRESS3500	P33500	P33500	DRUCKZU_3500	INPRESS_3500	P3_3500	
	inflowpress_4000	DRUCKZU4000	INPRESS4000	P34000	P34000	DRUCKZU_4000	INPRESS_4000	P3_4000	
	inflowpress_4500	DRUCKZU4500	INPRESS4500	P34500	P34500	DRUCKZU_4500	INPRESS_4500	P3_4500	
	inflowpress_5000	DRUCKZU5000	INPRESS5000	P35000	P35000	DRUCKZU_5000	INPRESS_5000	P3_5000	
	inflowpress_5500	DRUCKZU5500	INPRESS5500	P35500	P35500	DRUCKZU_5500	INPRESS_5500	P3_5500	
turbine_housing	flangeinnerdiam	DI1	DI1	DI1	DI1	DI_1	DI_1	DI_1	inner_diameter
	outletdiam	DI2	DI2	DI2	DI2	DI_2	DI_2	DI_2	
	exhaustdiam	DI3	DI3	DI3	DI3	DI_3	DI_3	DI_3	
	volutediam	DI4	DI4	DI4	DI4	DI_4	DI_4	DI_4	
	fittingdiam	DI5	DI5	DI5	DI5	DI_5	DI_5	DI_5	
	flangesidelength	LA1	LO1	LA1	LO1	LA_1	LO_1	LA_1	outer_height
	maxheight	LA2	LO2	LA2	LO2	LA_2	LO_2	LA_2	
	valvelength	LA3	LO3	LA3	LO3	LA_3	LO_3	LA_3	
	flangeheight	LA4	LO4	LA4	LO4	LA_4	LO_4	LA_4	
	valveheight	LA5	LO5	LA5	LO5	LA_5	LO_5	LA_5	
	fittingheight	LI1	LI1	LI1	LI1	LI_1	LI_1	LI_1	inner_height
	voluteheight	LI2	LI2	LI2	LI2	LI_2	LI_2	LI_2	
	diffusorheight	LI3	LI3	LI3	LI3	LI_3	LI_3	LI_3	
	outletheight	LI4	LI4	LI4	LI4	LI_4	LI_4	LI_4	
turbine_wheel	rotordiam	DA1	DO1	DA1	DO1	DA_1	DO_1	DA_1	outer_diameter
	outletdiam	DA2	DO2	DA2	DO2	DA_2	DO_2	DA_2	
	inletdiam	DA3	DO3	DA3	DO3	DA_3	DO_3	DA_3	
	inletheight	LA1	LO1	LA1	LO1	LA_1	LO_1	LA_1	outer_height
	outletheight	LA2	LO2	LA2	LO2	LA_2	LO_2	LA_2	
	rotorheight	LA3	LO3	LA3	LO3	LA_3	LO_3	LA_3	
shaft	turblength	LA1	LO1	LA1	LO1	LA_1	LO_1	LA_1	outer_height
	ctrhlength	LA2	LO2	LA2	LO2	LA_2	LO_2	LA_2	
	maxlength	LA3	LO3	LA3	LO3	LA_3	LO_3	LA_3	
	turbdiam	DA1	DO1	DA1	DO1	DA_1	DO_1	DA_1	outer_diameter
	ctrhdiam	DA2	DO2	DA2	DO2	DA_2	DO_2	DA_2	
	maxdiam	DA3	DO3	DA3	DO3	DA_3	DO_3	DA_3	
centre_housing	comprbpfittingdiam	DA1	DO1	DA1	DO1	DA_1	DO_1	DA_1	outer_diameter
	centrediam	DA2	DO2	DA2	DO2	DA_2	DO_2	DA_2	
	maxdiam	DA3	DO3	DA3	DO3	DA_3	DO_3	DA_3	
	turbhfittingdiam	DA4	DO4	DA4	DO4	DA_4	DO_4	DA_4	
	bearbushdiam	DI1	DI1	DI1	DI1	DI_1	DI_1	DI_1	inner_diameter
	thrustbeardiam	DI2	DI2	DI2	DI2	DI_2	DI_2	DI_2	
	sealdiam	DI3	DI3	DI3	DI3	DI_3	DI_3	DI_3	
	oilinletdiam	DI4	DI4	DI4	DI4	DI_4	DI_4	DI_4	
	comprhfittingheight	LA1	LO1	LA1	LO1	LA_1	LO_1	LA_1	outer_height
	oilcxheight	LA2	LO2	LA2	LO2	LA_2	LO_2	LA_2	
	turbhfittingheight	LA3	LO3	LA3	LO3	LA_3	LO_3	LA_3	
	maxheight	LA4	LO4	LA4	LO4	LA_4	LO_4	LA_4	
	thrustbearheight	LI1	LI1	LI1	LI1	LI_1	LI_1	LI_1	inner_height
	bearbushheight	LI2	LI2	LI2	LI2	LI_2	LI_2	LI_2	

In stage 1 (only reactive tool agents), the remaining knowledge gaps resulted in failure to produce product models in the required formats. This concerned data transfers to five tool agents (A-3, A-5, A-7, A-8 and A-16).

As these agents were all reactive, the mediator agent's strategy to default to its own native format (F_1) when coming across missing format knowledge (see Section 3.2.1) was never successful. As a result, the experiment ended with only one out of six required knowledge increases (or translator updates) being autonomously performed by the system.

However, when we turned these tool agents into reflective agents (stage 2), all data transfers succeeded. This is based on the ability of reflective agents to use generative parsing methods. While the mediator agent's knowledge gaps persisted, the tool agents were able to increase their format knowledge to parse product models that were completely or partially represented in the mediator agent's native format F_1. For example, Table 2 shows that agent A-7 could increase its knowledge from initially encompassing only format F_4 to comprise format F_1, which was completely unknown to that agent previously. This illustrates the idea of distributed translation that is a feature of the common ground approach (see Section 2.1). The system using stage 2 was able to achieve all six required knowledge increases (translator updates) autonomously.

TABLE 2. Gaps, indicated by the shaded areas, in the reflective A-7's initial format knowledge in experiment 1. All of these gaps could be removed autonomously.

Product part	Meaning	Variable name		Class
		Format F_1	Format F_4	(Hypernym)
turbine_housing	flangeinnerdiam	DI1	DI1	inner_diameter
	outletdiam	DI2	DI2	
	exhaustdiam	DI3	DI3	
	volutediam	DI4	DI4	
	fittingdiam	DI5	DI5	
	flangesidelength	LA1	LO1	outer_height
	maxheight	LA2	LO2	
	valvelength	LA3	LO3	
	flangeheight	LA4	LO4	
	valveheight	LA5	LO5	
	fittingheight	LI1	LI1	inner_height
	voluteheight	LI2	LI2	
	diffusorheight	LI3	LI3	
	outletheight	LI4	LI4	
turbine_wheel	rotordiam	DA1	DO1	outer_diameter
	outletdiam	DA2	DO2	
	inletdiam	DA3	DO3	
	inletheight	LA1	LO1	outer_height
	outletheight	LA2	LO2	
	rotorheight	LA3	LO3	
shaft	turblength	LA1	LO1	outer_height
	ctrhlength	LA2	LO2	
	maxlength	LA3	LO3	
	turbdiam	DA1	DO1	outer_diameter
	ctrhdiam	DA2	DO2	
	maxdiam	DA3	DO3	

Tables 3 and 4 show the overall results of all experiments conducted using reactive (stage 1) and reflective (stage 2) tool agents. The difference between the numbers of required and actually effectuated updates can be

interpreted as the number of translators that would have to be updated manually. It can be seen from Table 3 that the remaining manual work could be reduced by an average of 31% with respect to conventional approaches based on direct translation. The use of reflective tool agents could clearly improve this result to an average reduction of 88%, Table 4. Figure 12 compares the two stages graphically.

TABLE 3. The degree of automation in translator updates of stage 1.

Stage 1 (reactive tool agents)				
Experiment	**Total required updates**	**Learned updates**	**Manual updates**	**Reduction of manual updates**
1	6	1	5	17%
2	8	4	4	50%
3	6	2	4	33%
4	10	3	7	30%
5	8	2	6	25%
6	9	3	6	33%
Average	**7.8**	**2.5**	**5.3**	**31%**

TABLE 4. The degree of automation in translator updates of stage 2.

Stage 2 (reflective tool agents)				
Experiment	**Total required updates**	**Learned updates**	**Manual updates**	**Reduction of manual updates**
1	6	6	0	100%
2	8	7	1	88%
3	6	5	1	83%
4	10	7	3	70%
5	8	8	0	100%
6	9	8	1	89%
Average	**7.8**	**6.8**	**1.0**	**88%**

4.3.2. Scalability of the Memory System

A subsequent set of experiments analysed the performance improvement (PI) with increasing FBS models in the system. In addition, we completely abandoned the domain structure used in the previous experiments. Specifically, all values for all FBS properties for the 16 tool agents in Table A.3 in the Appendix were distributed randomly among the tool agents. With 5 different functions (F), 8 different behaviours (B), 6 different vendors and 7 different input and output formats (S), the number of possible combinations (FBS models) is $5\times8\times6\times7\times7 = 11{,}760$. This provides sufficient complexity to create a worst-case scenario for the mediator agent's memory system.

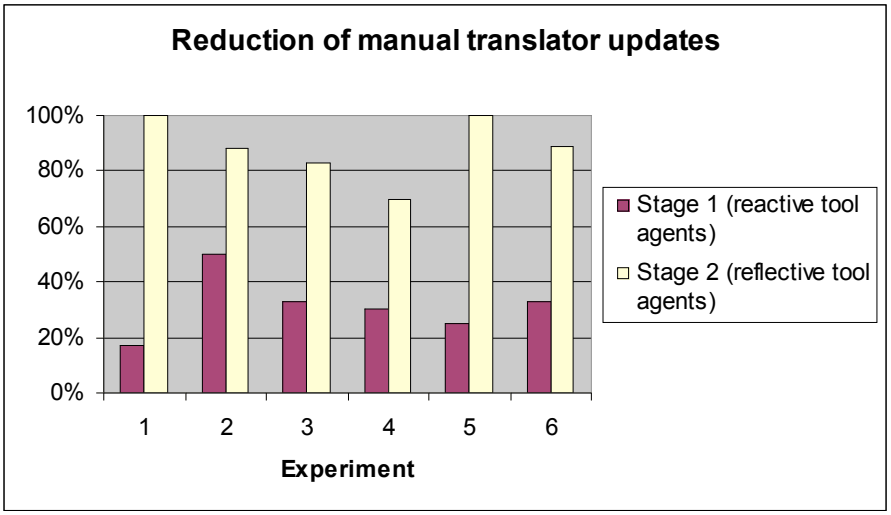


Figure 12. The reduction of manual updates of translators in the two stages.

Table 5 shows the performance improvement (PI) of the mediator agent's memory system when the number of initial FBS models is increased from 10 to 60. Every performance value in the Table is the result of a Monte Carlo simulation and is the average of 30 different runs that use the same number of initial FBS models. Figure 13 presents the results graphically. It can be seen that PI increases with the number of pre-established FBS models, but tends to an asymptote at a rate not far above 50%. This may seem as a no significant improvement; however, it is important to remember that this value is only a lower bound. Using the agent under more realistic conditions, i.e. in more structured environments, is likely to significantly enhance this result.

TABLE 5. Results of experiments with the mediator agent's memory system.

New agents	10	10	10	10	10	10
Initial FBS models (IAC instances)	10	20	30	40	50	60
New FBS models (IAC instances) after being introduced to the new agents	2.7	2.9	2.4	1.8	2.3	2.5
New FBS models (IAC instances) after interacting with the new agents	7.5	6.4	5.4	4.4	4.5	4.4
Performance improvement (PI) after interacting	25%	36%	46%	56%	55%	56%
Standard deviation (performance improvement)	0.1	0.2	0.2	0.2	0.2	0.2
Incorrect generalisation	48%	35%	30%	26%	22%	19%
Standard deviation (incorrect generalisation)	1.6	1.3	1.3	1.5	1.2	1.3

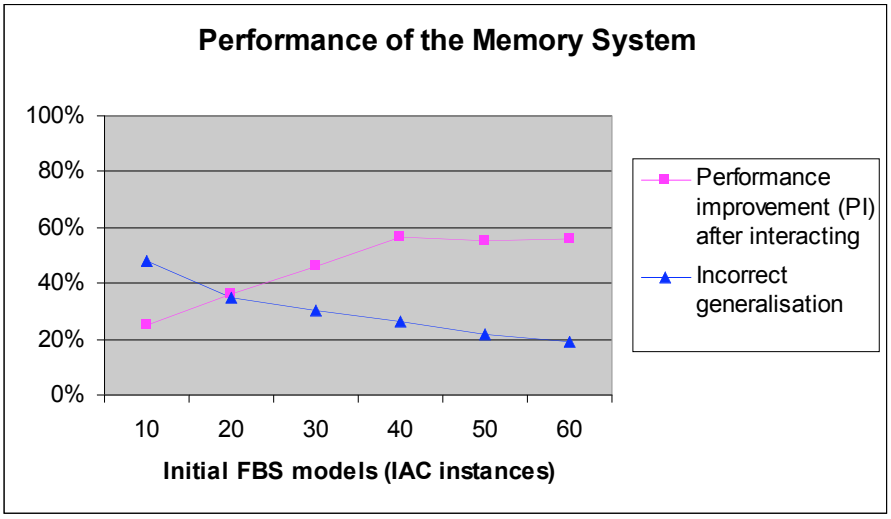


Figure 13. Performance of the mediator agent's memory system.

The memory system, in its current state of implementation, is not capable of performing a re-generalization of FBS models once they have been separated. As a result, the system is likely to have missed out on a number of potential opportunities to improve its performance by re-clustering instances in the light of new experiences. An additional factor is that the parameters of the IAC network have not been modified when the number of initially stored instances was increased. It can be assumed that this has been the reason for some over-generalizations that have led to wrong classifications of agents and finally to a higher number of specialized FBS models.

Another interesting outcome of the experiments is related to the increase of the number of new FBS models of the mediator agent from the moment after its first introduction with the new tool agents to the moment after its interactions with these agents, Table 5. This increase is due to the dynamic adaptation of FBS models in response to the detection of incorrect initial generalizations. It can be noticed from Table 5 and Figure 13 that the more FBS models the mediator agent has available in its memory, the more accurate become its initial categorizations of the new agents and the less adaptation of FBS models is required over the course of its product data exchanges.

5. Conclusion

Mass customization requires high degrees of automation and flexibility of the production line (Pine 1993). Industrial companies have implemented this strategy using flexible manufacturing systems (FMSs), computer-integrated

manufacturing (CIM), and modular product platforms. To date there has been no production line for interoperable product models that includes methods or mechanisms with similar degrees of automation and flexibility. As a result, mass customization has generally not been considered as a feasible alternative to the current approach based on market regulation.

This paper has proposed an agent-based approach that has the potential to fulfil the conditions for mass customization of interoperability: It automates the production of new mappings between product model formats and flexibly connects these mappings to their use in individual interactions. This creates a highly automated and flexible production line for product models that can adapt to possibly large markets with heterogenous or fluctuating customer demands. Specifically, our approach draws from recent progress in two areas of research.

1. *Situated agents*: This research area aims to develop agents that are autonomous and interactive. Agents need to be endowed with initial knowledge of the domain but can then operate autonomously within that domain. This includes the activity of learning new knowledge from interacting with their environment in addition to encoded knowledge. We have shown that this capacity allows automating the process of generating mappings between different product model formats.
2. *Models of social knowledge*: Internal models of agents are used in multi-agent systems to connect an individual agent's domain knowledge to the social context in which it operates. This aims to achieve flexible cooperation, as the agent can now reason about and adapt to individual interactions with other agents and establish a common ground. We have introduced the FBS schema as a basis for internal models that include both specialized and generalized information. Generalized FBS models can be seen as reusable facilities to generate families of product models, thus reducing combinatorial complexity.

Our approach may bring the idea of mass customizing product models (back) on the research agenda for interoperable design systems. Mass customization may then be considered to be applied in those domains where the market for tools is hard to regulate by a standardization approach. This will augment rather than replace standardization efforts. A direct link may be established between existing standards such as STEP or IFCs and the agent-based approach. For example, 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 turn, developers of new agents could use the emerging standard for engineering the initial knowledge of new agents.

We still have a long way to go to realize mass customized interoperability in practice. For example, a number of well-known, technical issues in product data exchange remain unaddressed by our approach. Some of these issues arise from discrepancies in the modelling kernels of different design tools, such as geometric primitives, transformations and tolerances. This produces the need for dealing with more complex mappings than those used in our example system. These complexities are likely to represent an obstacle to the rapid implementation of a fully mass customised approach to interoperability. Therefore, initial deployments of our approach may focus on more manageable sets of product data, such as the meta-data handled by product data management (PDM) systems. This could support current ISO activities in the area of interoperability between different PDM systems.

References

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.

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.

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.

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. 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.

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

ISO (2005) *International Organization for Standardization*, http://www.iso.ch/

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.

Kannengiesser U. and Gero J.S. (2006) Agent-based interoperability without product model standards, *Computer-Aided Civil and Infrastructure Engineering* (to appear)

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.

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 Analysis Office, Gaithersburg, MD.

Pine B.J.I. (1993) *Mass Customization: The New Frontier in Business Competition*, Harvard Business School Press, Cambridge.

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

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.

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.

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.

Wiederhold G. (1992) Mediators in the architecture of future information systems, *IEEE Computer* **25**(3): 38-49.

Appendix

TABLE A.1. The input and output data of stages (tool agents) 1 to 6 of the design of the turbocharger.

Task	Input	Output
1	(request to start formulation of initial requirements)	**ATL**: required air mass flow and efficiency (1000 - 5500 rpm), max. total dimensions **Turbine**: inflow pressure and inflow temperature (1000 - 5500 rpm) **Compressor**: inflow pressure and outflow pressure (1000 - 5500 rpm) **Core assembly**: oil pressure and temperature **Turbine housing**: required location of connections to engine **Compressor housing**: required location of connections to engine **Centre housing**: required location of connections to engine
2	**ATL**: required air mass flow and efficiency (1000 - 5500 rpm), max. total dimensions **Compressor**: inflow pressure and outflow pressure (1000 - 5500 rpm)	**Compressor**: required air flow, efficiency and pressure ratio (1000 - 5500 rpm), inlet and outlet diameter **Shaft**: max. speed
3	**ATL**: required air mass flow and efficiency (1000 - 5500 rpm), max. total dimensions **Compressor**: efficiency (1000 - 5500 rpm) **Turbine**: inflow pressure and temperature (1000 - 5500 rpm)	**Turbine**: mass flow, mass flow parameter, efficiency and pressure ratio (1000 - 5500 rpm), inlet and outlet diameter
4	**ATL**: max. total dimensions **Compressor**: inflow and outflow pressure (1000 - 5500 rpm), inlet and outlet diameter **Compressor housing**: required location of connections to engine	**Compressor housing**: geometry, material **Compressor wheel**: geometry, material **Compressor backplate**: geometry, material
5	**ATL**: max. total dimensions **Turbine**: inflow pressure and temperature (1000 - 5500 rpm) **Turbine housing**: required location of connections to engine	**Turbine housing**: geometry, material **Turbine wheel**: geometry, material
6	**ATL**: max. total dimensions **Core assembly**: oil pressure and temperature **Compressor backplate**: connections to centre housing **Compressor wheel**: connections to shaft **Centre housing**: required location of connections to engine **Turbine housing**: connections to centre housing **Shaft**: max. speed	**Centre housing**: geometry, material **Shaft**: geometry, material **Thrust bearing**: geometry, material **Bearing bushing**: geometry, material

TABLE A.2. The input and output data of stages (tool agents) 7 to 16 of the design of the turbocharger.

Task	Input	Output
7	**Centre housing**: geometry, material **Shaft**: geometry, material **Thrust bearing**: geometry, material **Bearing bushing**: geometry, material **Turbine housing**: geometry, material **Turbine wheel**: geometry, material **Compressor housing**: geometry, material **Compressor wheel**: geometry, material **Compressor backplate**: geometry, material	**ATL**: actual air mass flow (1000 - 5500 rpm) **Compressor**: actual inflow and outflow pressure and temperature and pressure ratio (1000 - 5500 rpm) **Turbine**: actual inflow and outflow pressure and temperature and pressure ratio (1000 - 5500 rpm)
8	**ATL**: actual air mass flow (1000 - 5500 rpm) **Compressor**: actual inflow pressure and temperature (1000 - 5500 rpm) **Turbine**: actual inflow pressure and temperature (1000 - 5500 rpm)	**Compressor**: actual air flow (1000 - 5500 rpm) **Turbine**: actual mass flow and mass flow parameter (1000 - 5500 rpm)
9	**Compressor**: actual inflow and outflow temperature and pressure ratio (1000 - 5500 rpm) **Turbine**: actual inflow temperature and pressure ratio (1000 - 5500 rpm)	**ATL**: actual efficiency (1000 - 5500 rpm) **Compressor**: actual efficiency (1000 - 5500 rpm) **Turbine**: actual efficiency (1000 - 5500 rpm)
10	**Turbine**: actual inflow temperature (1000 - 5500 rpm) **Turbine wheel**: geometry	**Turbine wheel**: actual thermal stress
11	**ATL**: actual air mass flow and efficiency (1000 - 5500 rpm) **Compressor**: actual air flow, pressure ratio and efficiency (1000 - 5500 rpm)	**Compressor**: evaluation result: - OK - new pressure ratio (1000 - 5500 rpm)
12	**Turbine**: actual mass flow parameter, pressure ratio and efficiency (1000 – 5500 rpm)	**Turbine**: evaluation result: - OK - new pressure ratio (1000 - 5500 rpm)
13	**Turbine wheel**: material, actual thermal stress	**Turbine wheel**: evaluation result: OK
14	**ATL**: actual air mass flow and efficiency (1000 - 5500 rpm) **Compressor**: actual pressure ratio (1000 - 5500 rpm)	(acknowledge comprehension)
15	**Centre housing**: connections to engine **Compressor housing**: connections to engine **Compressor wheel**: connections to shaft **Compressor backplate**: connections to centre housing **Turbine housing**: connections to engine and centre housing	(acknowledge comprehension)
16	**Turbine wheel**: geometry, material	(acknowledge comprehension)

TABLE A.3. A domain structure for the FBS properties of the 17 agents. Brackets indicate formats that replace an agent's initial format in some of the experiments.

Agent	F	B	S (input)	S (output)	Vendor
A-0	Data & workflow management	INFORM (input data)	F_1	F_1	V_1
A-1	Formulation	INFORM (overall requirements)	---	F_1	V_1
A-2	Formulation	INFORM (component requirements)	F_2	F_2	V_2
A-3	Formulation	INFORM (component requirements)	F_3	F_4 (F_5)	V_3
A-4	Synthesis	INFORM (component structure)	F_2 (F_6)	F_4	V_2
A-5	Synthesis	INFORM (component structure)	F_4	F_4 (F_7)	V_3
A-6	Synthesis	INFORM (component structure)	F_4 (F_5)	F_2	V_3
A-7	Analysis	INFORM (results of prototype test)	F_5 (F_6)	F_5	V_4
A-8	Analysis	INFORM (results of simulations)	F_4	F_6 (F_7)	V_5
A-9	Analysis	INFORM (results of simulations)	F_2	F_3 (F_4)	V_2
A-10	Analysis	INFORM (results of simulations)	F_3	F_6 (F_7)	V_6
A-11	Evaluation	INFORM (evaluation results)	F_1 (F_2)	F_1	V_1
A-12	Evaluation	INFORM (evaluation results)	F_2	F_2	V_2
A-13	Evaluation	INFORM (evaluation results)	F_6	F_7	V_6
A-14	Documentation	INFORM (performance description)	F_1	---	V_1
A-15	Documentation	INFORM (description for manufacture)	F_2	---	V_2
A-16	Documentation	INFORM (description for manufacture)	F_7	---	V_6