Towards Designing in Adaptive Virtual Worlds

Gregory J. Smith[1], John S. Gero[2] and Mary Lou Maher[3]

[1] University of Sydney, g_smith@arch.usyd.edu.au
[2] University of Sydney, john@arch.usyd.edu.au
[3] University of Sydney, mary@arch.usyd.edu.au

ABSTRACT

We present an approach to designing from within a virtual world in which the objects in the world have agency. In this kind of world, agents are associated with design components rather than specific design processes. By combining adaptive virtual worlds with situated FBS, we have a different kind of design support, illustrated through scenarios of building design. his document outlines the necessary details to prepare a conference paper for the annual CAD conferences. Authors are requested to follow all formatting instructions encoded into this MS Word file. To simplify the task of paper preparation, simply cut and paste the relevant sections into this document.

Keywords: Agent, designing, virtual world.

1. INTRODUCTION

Typically, designing using CAD as a visualization and documentation aid runs as follows. Starting with a set of incomplete and possibly inconsistent requirements, conceptual designing proceeds using pencil and paper or some qualitative computer tools, attempting to resolve those requirements. At some point the designer progresses onto detailed design. Also at some point, but probably not at the same point, pencil and paper give way to computational tools such as CAD packages, numerical analysis tools, word processors, spreadsheets, databases and so on. Current CAD tools tend to focus on detailing, visualizing and documenting a designed structure. Despite AI being useful for specific designing tasks, designing is predominantly performed by one or more designers, either separately or collaboratively.

Virtual worlds as 3D multi-user environments provide the potential for a different kind of support for design processes. In a 3D virtual world, the designers can interact with each other and manipulate the components of the design while being immersed in the virtual design. The concept of adaptive virtual worlds takes this one step further. An adaptive virtual world is one in which each component of the virtual design has an associated rational agent. The agent is able to reason about the design and the design changes, and to interact with the human designers in response to their design decisions.

In this paper we introduce our conception of agent-based virtual worlds, which provides the basis for adaptive virtual worlds. In this kind of world, agents are associated with design components rather than specific design processes. We consider this idea as an extension to the FBS model [1] by using the concept of situated FBS [2]. By combining adaptive virtual worlds with situated FBS, we have a different kind of design support, illustrated through scenarios of building design. We begin with a discussion of agent-based virtual worlds. Situated designing by agents is then described, followed by a review of a package that allows agents to be used with the Active Worlds platform. The notion of designing within the design is considered, and we finish with an example design scenario.

2. AGENT-BASED ADAPTIVE VIRTUAL WORLDS

Work on intelligent buildings has focused on human designers designing real buildings that behave intelligently. There are already existing buildings with computerized communications, energy management and environmental control systems. Recent work in AI is leading to the development of ``enhanced reality'' rooms using embedded sensors and effectors, such as for the provision of embedded teleconferencing facilities [3]. In this paper we reverse this focus, placing intelligence into agents that act at design time. We discuss in this section two motivating examples. one is designing *via* adaptive virtual worlds, the other is the design *of* adaptive virtual worlds.

An example of designing *via* adaptive virtual worlds is collaborative designing using systems of agents. One or more designers log into a virtual world containing a representation of the building that is being designed. The designer expresses changes to requirements as chat using a structured subset of natural language, through changes to structure by directly manipulating objects in the world, or via gesture recognition from a sketching interface. Consider a designer deciding that a room is too small. Let there be zone agents that reason about the spaces delimited by 3D objects, and let wall agents reason about 3D wall objects. The designer uses chat to tell a zone agent that it is too small, the wall reformulates expected behaviours, and communicates with the bounding wall agents so as to achieve the movement of a wall. Agents associated with that zone will act cooperatively but agents associated with adjoining zones may not. For cooperative zone agents, objects connected to the wall will also move (such as paintings and pipes), and floors and ceilings will expand appropriately. An adjoining zone may then reduce its space, and so that zone then negotiates to compensate.

An example of the design *of* adaptive virtual worlds is the intelligent virtual office described in [4]. As an example, one agent in this system is an intelligent door that has functions of allowing access, to restricting access, and providing security. The door agent recognizes the 3D representation of itself in an AW world and then maintains itself according to those functions. The door is informed by other agents of the classification of avatars that are nearby. Depending on these and on the door's interpretation of the world, it chats with avatars so as to determine security clearance and changes its structure so as to allow or restrict access.

These worlds are composed of objects such as walls and floors, and each object can be associated with an agent. When a person interacts with a virtual world via a browser they interact with a set of objects. These objects are 3D models, avatars, sounds, chat and so on. Virtual world browsers are designed to efficiently display these objects. Conceptually we wish that these objects, or at least a subset of these objects, can behave intelligently and so conceptually we consider a world that is a system of interacting agents. But in implementation terms the act of constructing 3D models may be distinct from the act of instantiating them in a world. So some agents represent themselves in a world as one or more objects and some do not. A wall agent would represent itself as a 3D model of a wall; a zone agent would not represent itself as an object in a world. Similarly, some objects may be associated with an agent and some may not be. In the next section we discuss how such agents can represent their world and the objects that constitute it.

3. SITUATED DESIGN

3.1 Situated FBS

The FBS model of design knowledge [1] categorizes the knowledge about a specific design component in terms of its structure (what it is), its behaviour (what it does), and its function (what it is for). In FBS models, behaviour is determined from structure according to some causation. Causation is a relation between two things where the first is thought of as somehow bringing about the second [5]. In the physical world, this causation is modeled based on our understanding of the laws of physics. With virtual worlds the bringing about is from procedures computed by agents or by the server, so behaviour is determined by whatever the "virtual physics"' are in the underlying platform. Regardless of whether the causation is physics or virtual, an agent's representations of interpreted behaviour are computed from its expectations of behaviour and from interpreted structure. These interpretations are computed from either encoded interpretation rules or are learned from experience.

Taking the FBS model to a situated model, the structure, behaviour and function are all interpretations by an agent from sense-data and from the sensed impact of effect-data. The sense data include the information from the virtual world server that describes the state of the components or objects in the world. Structure are beliefs of what objects in the world are, behaviour are beliefs of what objects in the world do, and function are beliefs of what objects in the world are for. Further discussions of interpretation are found in [6].

Figure 1. shows the situated FBS (sFBS) model of Gero [2] overlayed with an agent's process model. The interpreted world $S_i \cup B_i \cup F$ is the agents beliefs of how the environment is now and the expected world $S_e \cup B_e \cup F_e$ are the agent's beliefs of how it expects or desires the environment to be. S_x is the structure of the
environment.

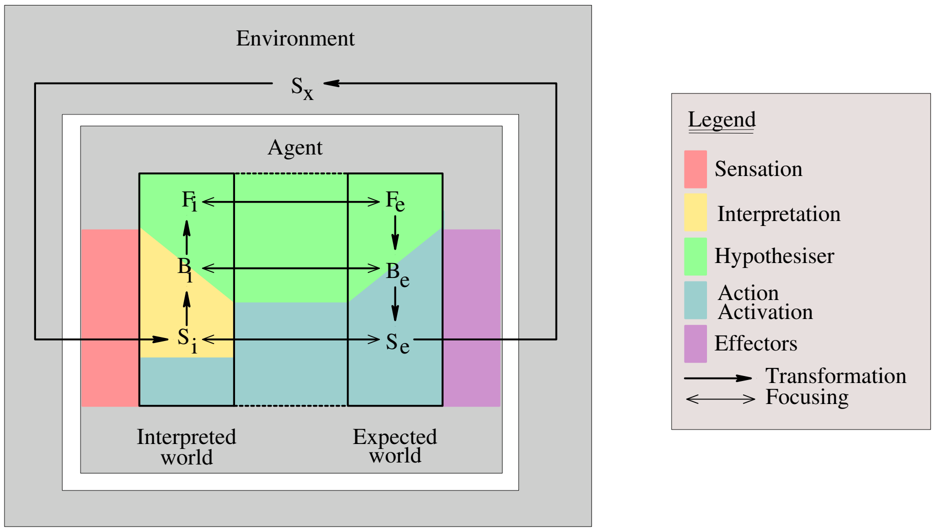

Figure 1. Diagram of sFBS transforms overlyed with agent processes.

S_i are the set of all structures of objects in the environment that are interpreted by the representing agent, and S_e are the set of all expectations of structure. To describe only the structure of a particular object we constrain S_i or S_e to only those objects. So for a wall Wall1, interpreted structure is the set of properties $S_i[\{Wall1\}]$, or in shorthand as $S_i[Wall1]$. Behaviours and functions can be denoted similarly.

Agents in any multi-agent system necessarily communicate. Any agent could communicate indirectly with another since it could sense and effect it's environment, and so changes made by one agent to the environment could be interpreted by another. In this work we define *direct* communication to mean agents intentionally sending each other or persons messages such as chat, and *indirect* to mean changing the environment and having another agent or person interpret those changes.

Agents that communicate directly must believe that they share a common understanding of the contents of messages. A shared understanding is common ground [7]. Common ground here consists of an ontology, a shared language of illocution, and interaction protocols. For our agents, communications to or from person agents are as textual chat, and communications between artificial agents are in an agent communication language (ACL) the locution of which is XML.

Messages, chat or otherwise, sent between agents are objects. A text string representing chat is the structure of a communication. The receiving agent interprets the structure of the communication as concerning the structure, behaviour or function of another object. The structure *of* a communication and structure *in the content* of a communication are therefore different things.

For persons, interaction protocols are learned as a part of learning natural language. For the artificial agents that we have implemented the protocols and ontology used have been encoded. To be situated, though, we should expect experiences of an agent to be reflected in it's beliefs and behaviour. Learning the contents of communications is beyond the scope of this paper; it is a difficult learning task that is being considered by researchers such as Steels [8].

In the case where communications are from a person that is a designer, direct communications will be as chat and so the artificial agents will require some amount of natural language understanding. For the artificial agents that we have implemented, interpretation of chat has been restricted to sentences of the form `[name, ] verb [direct object] [preposition] [indirect object]`. Here `verb` is compulsory and `name`, `direct object`, `preposition` and `indirect object` are optional. Each communication of this form is interpreted as an `inform` or `request` on the structure, behaviour or function of an object known to the agent. With an `inform` a designer communicates a belief of the structure, behaviour or function of that object. With a `request` the designer communicates a desire for change to the structure, behaviour or function of an object. For communications between artificial agents the idea is similar except that natural language is not used. Instead, these inter-agent messages contain XML serialized java beans.

For indirect communications, the structure that is interpreted is not the structure of a communication about an object but is instead the structure of an object in the world. One designer may sketch on a whitebeard, and another designer interprets that changing structure. A designer may decide that a wall is in the wrong place, move that wall, and zone agents interpret changed structure of that wall as a change to the structure of that zone.

3.2 The AWAgent Package Implementation

In this section we review a package that allows for agents to be added to worlds running on the Active Worlds platform. Further details can be found in [4,6].

We consider virtual universes that are object based, where constructing 3D objects to be instantiated in a virtual world is independent of the instantiation of those objects in that world. Similarly, chat, avatars, sounds, inter-agent messages and so on are all discrete objects. We also allow for agents to sense and effect objects that are not a part of a virtual world, such as by sending each other messages in an
agent communication language (ACL).

In general, a virtual universe is an environment and a set of agents. The environment includes virtual worlds plus any objects that exist independently of the worlds, and worlds are a partitioning of virtual universe objects into disjoint sets. That is, objects can exist in one world but agents can exist across worlds. The environment is everything between the effectors and sensors of the set of agents; anything perceived via a virtual world browser is a world.

A virtual world (or universe) may be centralized in a client-server fashion or it may be distributed. In a distributed virtual universe it may be that, unlike with client-server platforms such as Active Worlds, a virtual world is implemented directly as a system of agents. So to describe a universe independently of the implementation platform we describe the server in a client-server implementation as a special : “agent” a_0, and other agents as a_i where i=..N. We can also describe citizens logged into a world as “person agents”. Wall agents represent themselves in a world as a 3D wall object; person agents represent themselves in a world as a 3D avatar.

Some objects are associated with an agent and some are not. An avatar occupies a space bounded by objects - it does not occupy an object. So a 3D model of a door, for example, will be associated with a door agent but a zone agent that provides agency to a virtual space will not correspond to any particular 3D object. By “associated” we mean that the object represents the agent in the world to other agents. Conversely, agents are associated with zero or more objects. Of those agents that are not associated with any 3D object, some represent concrete concepts such as of a zone. Others are abstract, such as to receive high level requirements and decide what existing agent should handle it.

A package called AWAgent was written with the aim of providing a flexible, object oriented framework for providing agency to AW worlds. Instead of being statically linked at compile time, agents are configured using an XML file that is loaded via a validating DOM parser. This component based approach provides a flexibility that allows for the reconfiguration of agents running in a world without having to recompile and restart the server. It is targeted at eventually inserting agents into an AW world in the same way that a 3D object is. Agents can be added or removed dynamically according to the desires of a designer or according to inferences by other agent. The creator of an agent configures it by specifying a set of sensors, effectors, a rule-base and other parameters. These sensors and effectors encapsulate knowledge of how to communicate with a world. An ACL sensor and effector similarly encapsulated knowledge of how to send and receive messages to and from other agents. Agents are written using a combination of Java and Jess. Jess allows for declarative programming of knowledge. Java allows for a Java native interface to the Active Worlds platform, use of imperative Java methods, access to the Internet via HTTP sockets, and access to databases via JDBC. For more details of the AWAgent package, see [4] and [6].

4. DESIGNING FROM WITHIN THE DESIGN

Designing is often a collaborative process in which the rapport between designers is as important as their ideas [9]. The interest in technologies that facilitate collaboration is therefore unsurprising. Virtual design studios (VDS) are one such technology. Maher [10] and others have previously considered VDS and found that they provide a means for sharing representations and that they release some of the restrictions on the physical locations of the collaborators.

Many existing VDS maintain a desktop metaphor. This allows for a loose coupling of existing tools but does not encourage a community of collaborators because designers interact with the desktop rather than each other. The place metaphor of virtual worlds provides a stronger sense of collaboration but existing tools do not easily facilitate their use as design environments. One reason for this is the amount of effort at a structural level that is required to construct these worlds. To build a building in Active Worlds, for example, requires that detailed 3D models of walls, doors and so on be built, followed by their instantiation at specific locations in the world. It is our conjecture that if we can change all of the key objects in the world on which a designer acts to be agents, then a lot of the detailed structural effort can be encapsulated within those agents. This and the ability of agents to communicate and interact should allow designers to focus much more on functional requirements.

Designers from different disciplines often view the same design differently. Further, another problem with virtual worlds as VDS is that they are not good at modeling the "uncertainty, multiple visions and conflicting ideas that exist in the design process" [9]. We believe that this is because constructing a virtual world is a detailed exercise of instantiating specific concrete structures. This forces designers to commit to specific details for the purpose of visualization at a time when they would rather not. The solution of Haymaker et al. [11] is to use filter agents to interpret the same designed structure variously according to the disciplines of the designers.

We propose the opposite approach where agents are constructed with respect to required functions and behaviours, with agents then constructing structure variously. The design methodology proposed, then, is based on designing a world as a set of dynamic behaviours. This is in contrast to using CAD as drafting tools of structure dictated by the designer. We want the designer to concentrate on function and behaviour, and have a system of agents look after changing structure as much as possible. The way that we propose to achieve this is by designing using adaptive virtual worlds. These worlds are adaptive courtesy of multi-agent systems, where design knowledge is embedded into agents of different types that communicate so as to achieve design goals. So, for example, different kinds of design knowledge of what it is "to be a wall" is encoded into generic wall agents that are then instantiated variously and parametrically as required. Further, if the agents learn then they could adapt to both other agents (developing common ground) and to the designer. Agents select and modify their 3D model, and they adapt both collectively to changes in the system (the design) and individually to actions by the designer.

An important consideration is how to partition the functionality required of a design amongst agents. What should the scope of an agent be? A useful principle from software engineering is to assign the scope of agents so that they are maximally cohesive and have minimal coupling with other agents. Decreasing coupling between agents tends to decrease communication but increase agent size, resulting in a larger set of ascribed functions per agent. Having each agent ascribed only a small set of strongly cohesion functions tends to minimize agent size, with a consequent increase in communication.

Consider designing rooms. A wall could be instantiated in a world as a single 3D object, as an aggregation of 3D bricks, or as a part of a single room. So functions and behaviours will, from the designers viewpoint, be of the room as a space and of the wall as an entity. This applies to all objects with an associated agent such as doors, ceilings, floors and so on. Further, some of the objects bound a space (walls, for instance) and some do not (such as tables). A zone agent is an example of agency applied to something that does not correspond to an object in the world. As such it is an example of behaviour that cannot be added to a world without agents. A zone is a region of space, and as such it is delimited by a set of 3D objects in the world but does not represent itself in the world. The zone agent's primary function is to maintain for the multi-agent system a concept of what zones the office has. It computes the zone space from sensed wall geometry, and triggers a redesign of the space according to spatial functions. So a zone can be the largest 3D convex hull of points interior to a set of bounding objects and spatial functions can be ascribed to zone agents. Other functions can be ascribed to agents that represent themselves as bounding and non-bounding objects, with the scope of these agents decided as a compromise between minimizing communication between agents and maintaining that only similar functions be ascribed to one agent.

5. SCENARIO

In previous sections we discussed agent-based virtual worlds, situated designing by agents, and an approach to designing from within the design. In this section we further illustrate the idea through an example design scenario.

The design methodology described here applies to designing any artifact that can be represented in a virtual world where agents represent themselves by changing world structure and designers interpret it in terms of functional and behavioural requirements. Design of architecture and mechanical devices are examples, but the design of electric circuits is not because the layout of circuit components has little effect on the electrical behaviour of the circuit.

In this section we consider by way of example the scenario of the design of a building. The agents do not encode the topology of the building and then search it to satisfy design requirements. Rather, they communicate using interaction protocols to propagate changes to their neighbours, and they act reactively to sensed changes to the environment. The aim is to employ an adaptive system of agents that try to self-organize but under designer control.

An example of self-organization is where one agent makes a small change to the environment that biases the actions of others that are nearby, such as by adding a new chair to a table and having all of the others adjust their locations automatically. There are many cases where such a mechanism could be used in a multi-agent virtual world. Walls and zones could, for example, sense how many avatars are in a space and redesign the space accordingly. Janitor agents could roam a world looking for expired or otherwise undesirable objects or avatars. Lighting agents could move around a world with avatars, adapting to patterns of use. Walls that move in one zone could trigger neighbouring walls to adjust their own spaces by moving other walls.

The minimum design is a single zone that is bounded by the world (that is, unbounded). A new design is started by instantiating a new society of agents containing only a single unbounded zone. For the purpose of this scenario let there be two designers collaborating. Designer 1 is an architect and designer 2 is an interior designer. These two designers could communicate with each other directly with natural language chat or indirectly by changing and observing the structure of the world. Similarly, designers could communicate with agents using the structured chat described in Section 3.1. Designers could also manipulate objects in the world in ways that agents understand, such as by moving a wall and having both wall and zone agents re-interpret their sFBS representations accordingly. A distributed sketching tool could also be used by implementing a distributed sketching system and using gesture recognition to trigger design actions to be sensed by agents. In this scenario we shall focus on communicating functions and behaviours via structured chat.

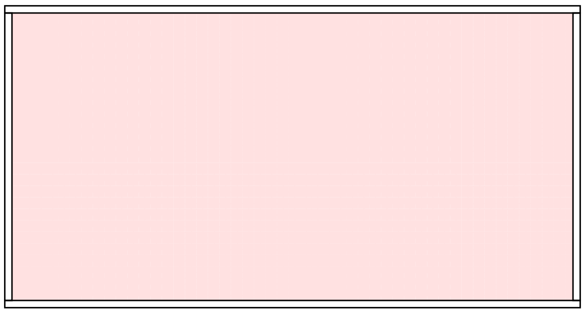

Figure 2. Initial plan view of the multi-agent system design. In this figure the floor agent is shown pink and is named Floor. The wall agents re named, clockwise from the top, Wall1, Wall2, Wall3 and Wall4. The ceiling agent Ceiling is now shown. Note that this particular set of agents were synthesized here for example purposes only.

Let the name of the existing unbounded zone be Zone1. Designer 1 says “Zone1, create visual boundary for Zone1”, which Zone1 receives as chat sense-data. Zone1 interprets this sense-data as the performative verb `request` for subject `create` of a `visual boundary` with respect to agent Zone1. That is, designer 1 is requesting that Zone1 change its function to include a visual boundary. As no other requirements have been communicated, hypothesizer in Zone1 changes it's expected behaviours and action activation satisfies this by synthesizing a new default set of wall, floor and ceiling agents located around the existing central location of the zone. This is illustrated by the floor plan of Figure 2.

Assume that designer 1 then chats with designer 2 saying that there needs to be one room for displaying art and another for holding meetings. Designer 1 then says “Zone1, create partition of Zone1”. Zone1 interprets this, updates it's functions, but hypothesizer decides that it does not have sufficient information to construct new expected behaviours: should the partition split Zone1 into two agents, or is an internal room divider required? Zone1 constructs a communication protocol to interact with the designer and receives an answer. Assume for this example that the answer is to split into two zones.

After the protocol completes, action activation in Zone1 instantiates a new wall called Wall5, changes zones representation $S_i[Zone1]$ to represent a shift of location to one side of Wall5, re-computes it's representation of the bounds of the zone in $S_i[Zone1]$ and avatar classifications in $B_i[Zone1]$ accordingly, and instantiates a new zone Zone2 for the other side of Wall5.

These interaction processes continue, with the designers chatting amongst themselves and then communicating requirements to the agents. Each time that a communication to an agent changes a decision taken by an agent, such as assuming default structure, that agent learns. So agents adapt to the preferences of the designers. Figure 3. shows the design at a later time.

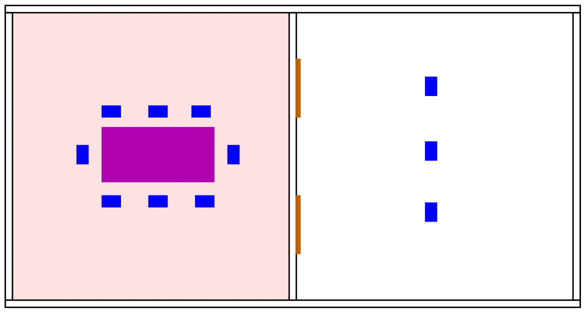

Figure 3. Plan view after the addition of another wall, floor, zone and furniture. Chairs are in blue, a table in magenta, and pictures hanging on Wall5 are brown.

As this is a design environment, we need to be able to add, remove and alter the agents that constitute the system at runtime. We would like for this to happen without all of the agents having to re-compute a world model at each change. They should act appropriately in different situations, where differences in situations cause the application of the same knowledge to result in differing behaviour. To facilitate this we allow the society to add and remove agents dynamically at runtime, and the agents for walls, furniture and so on are all situated and reactive. Reactive processes usually have little or no memory but tend to be very task specific. Reflective (deliberative) processes usually reduce to some form of search and therefore only work tractably if the problem space is restricted [12]. Naturally, therefore, reactive-reflective hybrids have been of interest. Neural network hybrids are fast but lack variable binding ability. Most symbolic hybrids are based on some propositional logic but can be "wildly intractable" [2].

The solution advocated by Chapman [13], Horswill [12] and others is to use deictic references: indexicals of signified function. The idea is to keep the reactive rules but to change what they signify Horwsill calls these deictic references *roles*. Each role is a signifier (a label) that is bound to a set of properties and is maintained by low level processes. For a picture agent, then, at initialization time a role of wall-that-I-hang-on would be bound to the 3D object that it sensed that it touched. The interpretation process consists of simple rules then sense changes in structural properties of whatever object this role is bound to binds to. Reactive action rules then act with respect to the role so as to, in this particular case, maintain the ascribed functions of a picture agent. The effect is to separate perception into processes of identification and localization. Looking at a specific place in the environment and interpreting what is there is an identification processes. Given sense-data from an attended object, identification identifies that object using classifiers that partition the sensory space. Having an expectation of what is in the environment and finding its location is a localization process. Gaze control in robot vision is a localization process.

The furniture added in Figure 3. are situated, reactive agents that also use default values and self-organization to locate themselves. Chairs, for instance, desire that they sit on a floor, attract to tables, repel from other chairs, and they have knowledge of how to change their structure. Pictures have similar goals and knowledge except that they desire to hang on a wall. These agents take the designers to be an oracle from which they learn by making equivalence queries. So if a designer explicitly changes an agent such as by relocating it, then that agent learns such that in later self-organizing actions this new knowledge will subsume the default actions.

Now let designer 2 decide that Zone1 is too crowded for the amount of furniture that has been added. The designers chat, agree that Zone1 needs to be bigger, and designers 1 says "Zone1, enlarge to the east". Zone1 interprets this chat sense-data, updates interpreted function, and hypothesizer asserts a new goal for a wall to move. But wall agents move walls, not zone agents. So Zone1 instantiates a request protocol to Wall5, for example, to request that it move in a direction equivalent to east.

Wall5 receives the request and binds a role for the-zone-to-retreat-from. Action rules in Wall5 recognize this role and act to move Wall5. Now, when the wall moves other agents in the neighbourhood sense the change in structure of the room and react to the new situation, self-organizing to adapt to the change. The wall-that-I-hang-on role of the picture agents interpret this sense-data as no longer in contact with the object bound to the role (Wall5), and so their action rules effect a move of their picture object accordingly. The wall agent does not keep track of room topology for when it moves because other agents look after themselves. The floor and ceiling agents similarly sense that they no longer bound at Wall5, bind roles for the wall, and grow or shrink accordingly. The chair and table agents in the two zones move so as to reach new equilibrium positions with respect to the relocated wall. All of these agents react to the changed situation without building a model of the world a planning the optimal response. This allows the system to adapt to new situations and changes to the agent system.

Sense-data from the change in the location of Wall5 results in Zone2 interpretation re-computing it bounds. Assume that Zone2 has a function of having a minimum area. In this case Zone2 hypothesizer would assert a goal to expand. Not having any knowledge of how to achieve this directly, a contract net protocol is instantiated to ask whether any agent in the system can assist it. The Petri net there was drawn and simulated using Renew, hence all messages are shown as strings so as not to confuse the diagram with other Java bean instantiations. Zone2 sends a call for proposals (CFP) to all agents in the society, or to all nearby agents. This means that Zone2 does not need to maintain a representation of the topology of agents and objects in it's neighbourhood, and so can behave reactively in new situations. Receiving agents either respond with a proposal that it believes will satisfy the initiators requirements, or respond with a refusal. Zone2 selects one of the proposals received within the deadline and sends an accept proposal message to that agent. In this case it will be to a wall, resulting in that wall moving. As before, the effects of that movement propagate to other neighbouring agents.

There is therefore an amount of adaption by system as well as the individual. The system adapts because the effects of changes propagate to neighbouring agents. Indeed we could increase the amount of adaption and self-organization by having zones automatically enlarge according to the amount of furniture present. Individuals adapt by learning from communications with designers.

6. CONCLUSION

Virtual worlds as 3D multi-user environments provide the potential for a different kind of support for design processes. Designers can interact with each other and manipulate the components of the design while being immersed in the virtual design. Further, making the world agent based allows it to reason about the design and the design changes, and to interact with the human designers in response to their design decisions. Combining adaptive virtual worlds with situated FBS leads to a different kind of design support.

7. REFERENCES

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

[2] Gero, J. S. and Kannengiesser, U., The situated function-behaviour-structure framework, in: J. S. Gero (Ed.), *Artificial Intelligence in Design '02*, Dordrecht: Kluwer, 2002, pp. 89-102.

[3] Brooks, R., The intelligent room project, in: *Proceedings of the Second International Cognitive Technology Conference (CT'97),* Aizu, Japan (1997).

[4] Maher, M. L., Smith, G. J. and Gero, J. S., Design agents in 3D virtual worlds, *IJCAI'03 Workshop on Cognitive Modeling of Agents and Multi-Agent Interactions (2003).*

[5] Lacey, A. R., *A Dictionary of Philosophy*, London: Routledge, 1996.

[6] Maher, M. L., G. J. Smith and Gero, J. S., *Situated agents in virtual worlds*,upcoming (2003).

[7] Gero, J. S. and Kannengiesser, U., Towards a framework for agent-based product modelling, in: K. Gralen, U. Sellgren (Eds.), *International Conference on Engineering Design, ICED 03, Stockholm, Sweden*, The Design Society, August 2003.

[8] Steels, L., The origins of syntax in visually grounded robotic agents, *Artificial Intelligence 103* (1998) 1-24.

[9] Lawson, B., *How Designers Think: The Design Process Demystified*, Oxford; Boston: Architectural Press, 1997.

[10] Maher, M. L., Simoff, S. J. and Cicognani, A., *Understanding Virtual Design Studios*, London: Springer, 2000.

[11] Haymaker, J., Ackermann, E. and Fischer, M., Meaning mediating mechanism: Prototype for constructing and negotiating meaning in collaborative design, in: J. S. Gero (Ed.), *Artificial Intelligence in Design'00*, Dordrecht: Kluwer Academic, 2000, pp. 691--715.

[12] Horswill, I. D., Grounding mundane inference in perception, *Autonomous Robotics* 5(1) , 1998), 63-77.

[13] Chapman, D., *Vision, Instruction and Action*, Cambridge, MA: MIT Press, 1991.