Design Agents in 3D Virtual Worlds

Mary Lou Maher, Gregory J Smith and John S Gero
Key Centre of Design Computing and Cognition
University of Sydney
mary@arch.usyd.edu.au, g_smith@arch.usyd.edu.au, john@arch.usyd.edu.au

Abstract

Design agents are rational agents that monitor and modify elements of a designed environment. Special characteristics of design agents include the ability to reason about patterns and concepts, and the ability to act autonomously in modifying or changing the design to achieve their own goals. 3D Virtual Worlds are multi-user distributed systems that provide a designed environment and a closed world environment for studying design agents in a multiagent system. We present a model for a design agent reasoning process and a model for constructing a memory of the agent's knowledge and interaction with a virtual world. The reasoning process includes sensation, perception, conception, hypothesizing, and planning a sequence of actions. Each agent has a constructed memory: a dynamic and changing view of the designed world that is determined by the agents sense data and reasoning. This model of design agents is compared to the BDI model of rational agents. We have implemented these models by extending the Active Worlds platform so that each object in the 3D world can have agency. We illustrate the models with a door agent and a multi-agent office that operate within a 3D world.

1 Agent-Based Virtual Worlds

3D Virtual Worlds are multi-user distributed systems that provide a closed world environment for studying design agents in a multiagent system. The 3D virtual worlds that we have been working with allow multiple users to connect to a server through a client, where the processing is distributed between the server and the client. Our early investigations used a virtual campus implemented in LambdaMOO [Maher and Skow, 1997] as the virtual world environment, and we now use a virtual learning environment implemented in Active Worlds [1] (AW) as the software environment for developing our agent societies. In Active Worlds, the users appear as avatars and are able to interact with and change the objects in the world. This environment provides an unbounded closed environment

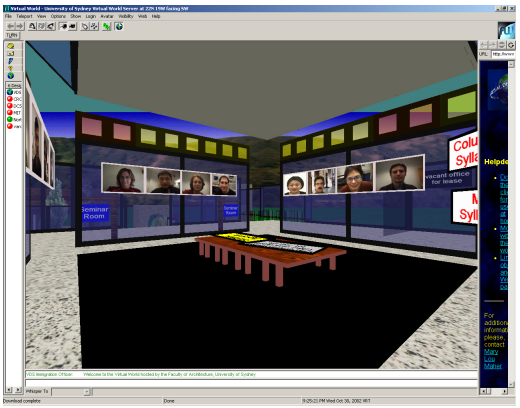

Figure 1: A 3D model of a virtual conference room.

for studying agent reasoning, interaction, and communication. Figure 1 shows a virtual conference room that was designed and used for meetings and seminars. The objects in this room exhibit interactive behaviors preprogrammed in the Active Worlds platform, but they do not have agency. Our research starts with this platform as a basis for developing agent-based virtual worlds.

With the exception of human controlled avatars, each object in a virtual world has both a 3D model that supports the visualisation of the object's function in the world, and is a software object that can have agency to support autonomous behaviour. When each 3D object has agency, the virtual world is composed of design agents in the sense that the world is designed and each object in the world can sense the world and respond by adapting the designed world.

Research in the field of design computation and cognition is increasingly agent based. This ranges from computational models of design processes to agent-based generative systems, see, for example, [Gero and Brazier, 2002]. The combination of design computation research and of agent-oriented virtual worlds provides a theoretical basis for associating agency with the designed physical world [Cohen, 1999] and can lead to agents designing virtual worlds [Maher and Gu, 2002].

3D virtual worlds are networked, multi-user environments that support communication and collaboration in a place-like context. The virtual worlds that we are considering as the ba-

[1] http://www.activeworlds.com

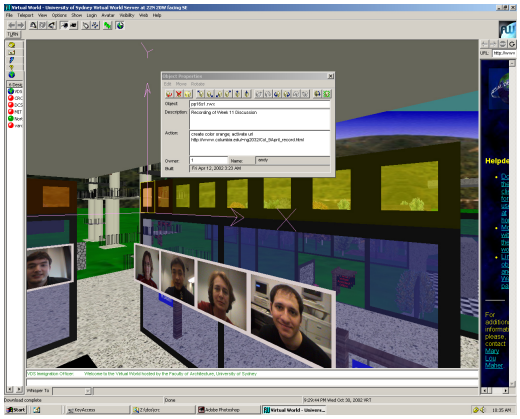

Figure 2: Inserting an object into an AW world.

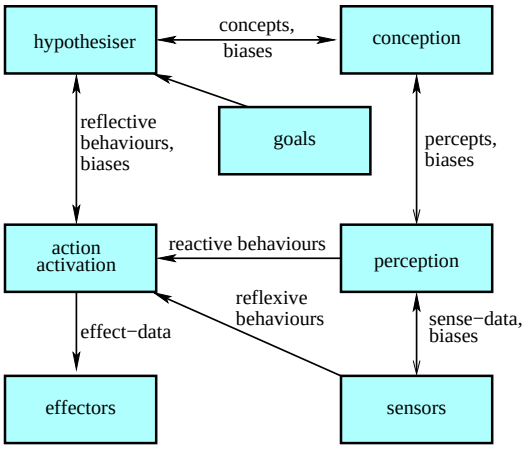

Figure 3: Design Agent Model: each agent has sensors, perceptors, conceptors, a hypothesiser, actions and effectors.

sis for our agent-based virtual world are object-oriented systems that associate a 3D model and a behaviour with each element of the world. Examples of such worlds include: Active Worlds and VirTools[2]. For the remainder of this paper, we will describe our agent-oriented world using an adaptation of the Active Worlds platform.

New 3D objects are added to an AW world by copying an existing object, moving it as required, and editing a dialog box to configure it, as shown in Figure 2. The dialog box allows the world builder to specify a 3D model and a script that describes the behaviour of the object. The 3D models can be taken from a standard library provided by Active Worlds, or can be generated in a 3D modelling package and added to the library. The behaviour of an object is limited to the preprogrammed behaviours allowed by the scripting language. By making the world agent-based, a person designing the world is able to select or generate an agent model in a way that is similar to selecting or generating a 3D model for the object.

Although the focus of this paper is on agent reasoning and agent memory, we intend that these agents communicate and collaborate. In this application, we define a multi-agent system to be an aggregation of agents that share some ontological connection, such as a room agent plus a set of wall agents that collectively comprise a virtual conference room. The multi-agent system facilitates inter-agent communication and manages resources on behalf of the agents. This includes computational resources, such as a thread pool and a connection to a virtual world.

This paper presents a model for a design agent reasoning process and a model for constructing a memory of the agent's knowledge and interaction with the world and other agents. By establishing a common model for reasoning and agent memory, the world can be designed using a consistent representation and therefore expectation of what it means for the objects in the world to have agency.

2 Reasoning Process Model for Design Agents

We have developed a reasoning process model for design agents that is independent of the type of design or environment in which the agent operates. This model, derived from [Maher and Gero, 2002; Gero and Fujii, 1999; Smith and Gero, 2002], allows the agent to reason about the world through sensation, perception, and conception, and to reasoning about its own behaviour by hypothesising and planning a set of actions. Our agent model comprises sensors, effectors and four reasoning processes: perception, conception, hypothesiser, and action activation. This model is illustrated in Figure 3, showing how the processes interact with each other and the sensor and effectors.

The agent is able to sense and have an effect on the virtual world through its sensors and effectors. Perception interprets sense-data and recognises patterns in the data. Conception associates meaning with percepts or patterns. The hypothesiser monitors the percepts and concepts, and identifies and selects goals that are associated with the agent's view of itself in the world. The action activator reasons about the steps to achieve a goal and triggers the effectors to make changes to the environment. Triggers may be directly off sense-data and percepts, or may be the implementation of partial plans.

In our agent model there are three levels of reasoning:

- Reflexive: where sense-data being placed in the agent's memory from the sensors triggers action activation.
- Reactive: where percepts in the agent's memory triggers action activation. These are behaviors that do not involve intentions.
- Reflective: where the agent reasons about concepts and alternative goals before action activation is triggered.

These levels of reasoning allow the agent to act at different levels of knowledge about the world. The reflexive level of reasoning is similar to the pre-programmed behaviours that the Active Worlds scripting language allows. The reactive and reflective levels of reasoning allow the agent to reason about its understanding of itself in the world before acting.

The Active Worlds platform allows the behaviour of the world to be extended using a software development kit (SDK). This SDK can be used to program bots that can enter

[2]http://www.virtools.com

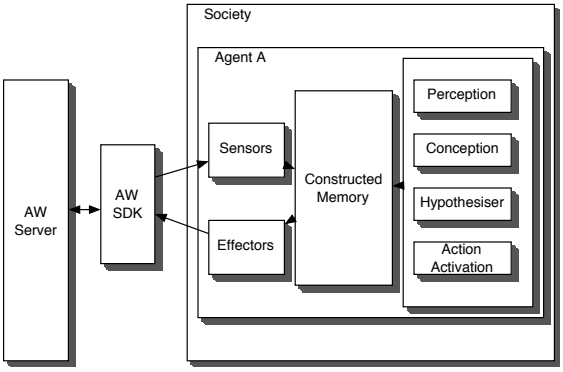

Figure 4: The Agent Model: sensors and effectors are implemented as Java Beans; the constructed memory and agents processes are implemented in Jess.

the world as software controlled avatars and interact with human controlled avatars and objects. We are using the SDK to implement our agent model. Using the SDK directly means writing a C/C++ program to interface to a dynamic link library. Sensors and effectors are implemented in Java using a Java Native Interface. This encapsulates the SDK within a set of standard configurable components. The agent reasoning is implemented using the rule-based language Jess [3]. The rules in the rule-base are grouped into modules according to perception, conception, hypothesiser, and action activation. The fact base is effectively a constructed memory for each agent, which is modified externally by input from the sensors and internally by the rules. This implementation configuration is illustrated in Figure 4.

3 Memory Model for Design Agents

Our agents make use of the Function-Behaviour-Structure (FBS) formalism [Gero, 1990]. In this formalism, design objects are represented by three sets of descriptors: function F, which is the teleology ascribed to the design object; behaviour B, which are attributes either derivable from structure or expected to be derivable from structure before structure exists; and structure S, which are the elements and their relationships that go to make up the design object. Structure, therefore, is what is synthesised in a design process and is what the designer can directly decide. Behaviour can only be derived and, therefore, its values cannot be directly decided upon by the designer. Function is an ascription only, although for humans the relationship between S and F is learned so that it appears there is a direct connection between them.

An Active Worlds world will contain a set of 3D objects where some 3D objects have no agency, some 3D objects are added to the world by an agent, and some 3D objects are recognised by an agent and thereafter can be changed by the agent. Furthermore, some agents will correspond to a single 3D object, whereas some will correspond to a set of 3D objects. Each agent maintains a representation of itself as an object or objects in the world, using FBS to categorise the attributes of itself as an object. Each agent also constructs an FBS model of the other objects in the world that are relevant, as described in [Gero and Kannengiesser, 2003]. In addition, an agent constructs a representation of the avatars in the world, its goals, and its plans as a sequence of actions. Structure will therefore comprise the identification and location of a set of 3D objects, sensed and perceived relations between 3D objects and/or the world, plus other sensed data such as uninterpreted messages from other agents.

A visual scene, as an external context for a design agent, is too complex to be used as pre-supplied percepts. So we distinguish sense-data from percepts. This is an example of a situation where a coupled dynamical systems model of agency [Beer, 1995] applies, and it applies generally for embodiment in a complex environment. Sense-data are uninterpreted data collected by a sensor, and percepts are interpreted patterns in that sense-data. Sense-data provide structure, S, as behaviour is derived and function is ascribed. Sense-data is used to construct the agent's structural representation of objects and people (avatars, citizens) in the world.

Perception reasons about patterns that the agent finds that produce a representation of behaviours of the objects in the world. Behaviour, therefore, is an interpretation by the agent of changing structure. The agent cannot sense behaviour, it must construct it.

An agent can, however, communicate behaviour to another agent providing they share an ontology. For example, consider predicate `on` applied to a wall and roof in a virtual world. If `(on ROOF WALL)` is interpreted as meaning that `ROOF` is located above `WALL`, then `on` is structural. If `(on ROOF WALL)` is interpreted as meaning that `WALL` supports `ROOF`, in the sense of applying a force, then `on` is behavioural. In a virtual world this distinction is important as it is not necessarily the case that a roof must be supported.

There are two possible sets of behaviours: actual behaviours B_s and expected behaviours B_e. Actual behaviours are derived from structure. The transformation $S \rightarrow B_s$ by which the agent interptrets current structure is called analysis. So percepts derived from $s \in S$ are actual behaviours $b_s \in B_s$. Concepts in our model are abstractions over or relations between percepts, so concepts derived from $b_s \in B_s$ are also actual behaviours.

A goal of a design agent is to transform an intended function, $f \in F$, into a plan for changing the structure of the world. Generally no direct transformation $F \rightarrow S$ exists [Gero, 1990]. So instead, design formulation transforms F into expectations of behaviour B_e, and designs are synthesised from B_e and B_s. Formulation, then, is the transformation $F \rightarrow B_e$ that produces concepts that represent expected behaviour. They are concepts, not percepts, as they are derived from required function. Concepts represent behaviours and function. Formulation is one transformation performed by the hypothesiser. The other is evaluation: comparing formulated expectations of behaviour B_e against current actual behaviour B_s.

An agent modifies the structure of a world by adding, deleting or changing the 3D objects that comprise that world. As behaviours are constructed interpretations, they remain internal representations of the agent. The same applies to function. Therefore, effect-data are structure just as sense-data are.

[3] http://herzberg.ca.sandia.gov/jess/

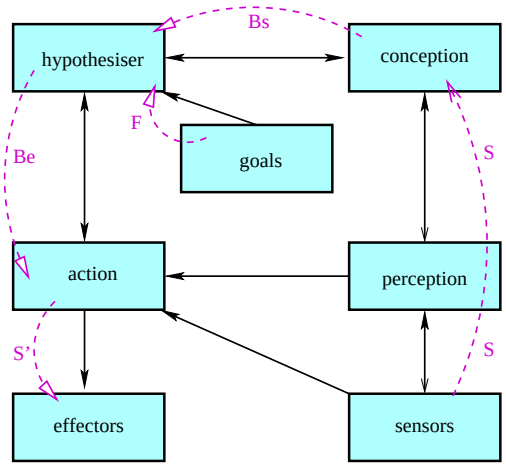

Figure 5: Design synthesis overlayed on Figure 3. To distinguish between current structure from the world and effects to be applied to the world, we denote structure to be effected by S'. Dotted arrows denote FBS transformations.

The principal design process is that of synthesis:

$$\begin{aligned} F &\rightarrow B_e \\ B_e &\rightarrow S(B_s) \end{aligned}$$

This has been illustrated in Figure 5 by overlaying the FBS transformations on the model of Figure 3. Expectations of behaviour are derived from required function ($F \rightarrow B_e$) by the hypothesiser. The action activator then uses the behaviours and knowledge of structure to generate intentions that, if realised, it believes will satisfy the required function ($B_e \rightarrow S(B_s)$). So goals and concepts are used by the hypothesiser to generate a partial plan that the action activator enacts.

Our agents react and reason about objects in the world. Conceptually they *are* objects in the world, only imbued with an agency that they would not otherwise have. Some of the tasks of the agent are simple reflexive or reactive ones. For such tasks a sense-data driven, forward chaining of encoded partial plans is sufficient. On the other hand, many of the reflective tasks of agents that we desire of our multi-agent systems are design ones. Such agents may hypothesise the behaviours that are necessary to satisfy its functions, evaluate those expected behaviours against current actual behaviour, and thus desire changes to the structure of the world. Desiring and acting to bring about changes to the world are design acts that, in our view, are best handled with a design formalism like FBS. Such an approach also allows us to build on prior work done in the field of design computation. Our model has been developed to allow such reflective design reasoning to occur without preventing real time reflexive and reactive behaviour.

4 Illustration of a Door Agent in AW

We demonstrate our design agent model with an example door agent, shown in Figure 6(a). The door is initially a single agent that recognises the 3D representation of itself in an AW world and then maintains itself according to its function.

The agent package currently includes sensors for chat text, avatars, 3D objects (within a configured region of the agent), ACL messages, and virtual reality time. Sense-data are all java beans asserted into Jess working memory. Sensors and effectors are configured from XML, which for the door is as follows:

```
<reteagent name="door">
<parameter name="jess" value="file:officedoor.clp"/>
<parameter name="owner" value="Greg"/>
<parameter name="friendlist" value="Greg Mary"/>
<parameter name="officeXmin" value="-1910"/>
<parameter name="officeXmax" value="-580"/>
<parameter name="officeZmin" value="-5810"/>
<parameter name="officeZmax" value="-4160"/>
<parameter name="radius" value="300"/>
<location x="-1920" y="-250" z="-4065"/>
<behaviours codebase="file:">
<sensor
   class="kcdcc.awa.base.AW3DObjectSensor">
   <parameter name="width" value="500"/>
   <parameter name="height" value="500"/>
</sensor>
<sensor class="kcdcc.awa.base.AWAvatarSensor"/>
<effector class="kcdcc.awa.base.AW3DObjectEffector"/>
</behaviours>
</reteagent>
```

The sensor `kcdcc.awa.base.AW3DObjectSensor`, for example, senses 3D objects from the virtual world. For example, when a 3D object is clicked on from an AW browser by a citizen, a java bean is asserted in working memory that looks from Jess like the following:

```
(kcdcc_awa_base_ClickedObject3DSenseData
   (objectNo ?no)
   (locn ?location)
   (OBJECT ?obj)
   (slot class))
```

Similarly, there are effectors for chat, 3D objects, telegrams, URLs, and avatar movement/teleportation.

In agent memory there is an object corresponding to each object of the 3D world that is of relevance to the agent. These objects have properties that are categorized according to FBS. Perception rules have raw sense data on the LHS and then modify agent memories of structure on the RHS. For the door, perception determines structure to recognise the door 3D object and recognise the presence of citizens. The 3D object of the door must be sensed as the citizen that owns the door can move it indendepently of the agent (using an AW browser). The structure of the door is remembered as Jess fact:

```
(deftemplate door
   (slot state (type LEXEME) (default NONE))
   (slot objectid (type INTEGER) (default 0))
   (slot location (type OBJECT))
   (slot ownerid (type INTEGER) (default  0))
   (slot owner  (type STRING))
   (slot radius (type INTEGER) (default 200))
   (multislot friendlist)
   (slot model (type STRING) default "pp16w3.rwx"))
   (slot description (type STRING)
                     (default "Intelligent Door"))
   (slot action (type STRING)
                (default "create color white"))
)
```

Beliefs of the agent are modular. For example, perceptor rules are in module `PERCEPTION`, so when the avatar for an already recognised citizen moves, the following rule fires:

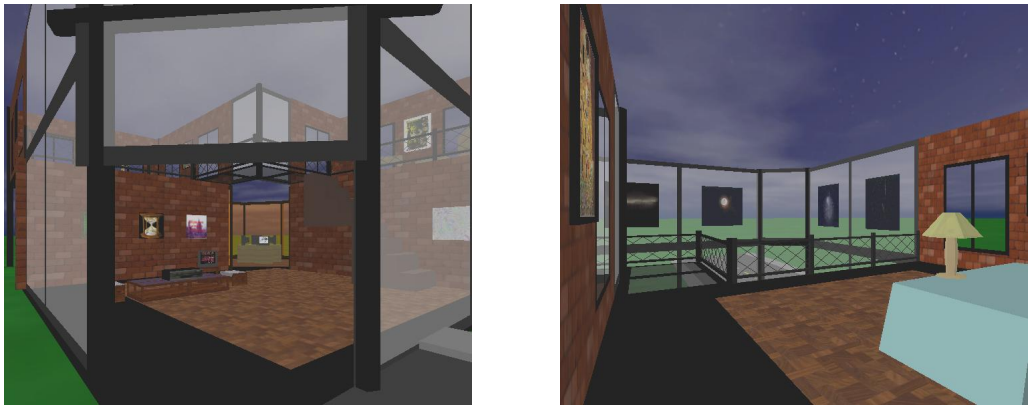

Figure 6: Our virtual office: (a) view from outside, with the door agent opening the door, and (b) the view from upstairs.

```
(defrule PERCEPTION::known-person-perception
?f <- (MAIN::kcdcc_awa_base_LocatedAvatarSenseData
          (name ?n)
           (locn ?l)
           (session ?s))
   ?p <- (MAIN::person (name ?n))
=>
   (modify ?p (location ?l) (area NONE))
   (retract ?f)
)
```

The inference engine is capable of forward and backward chaining but as it is built from java components, any java-implemented reasoning could be used. The focus stack of the inference engine restricts rules to firing from one module at a time. Using the focus stack in this way isolates the control mechanism from any knowledge: all rules on the agenda from the current focus module run, then all rules on the agenda from the next scheduled module run, and so on until the end of the schedule. For this particular agent the schedule is `(REACTIVE CONCEPTION PERCEPTION MAIN)` (in reverse order). Both sensation and the scheduler use module `MAIN`.

Conception rules associate meaningful concepts with the objects in the world that are relevant to the agent. A concept can be a behavior or a function of an object. For example, a person is categorised according to where they are and whether they are recognised as friends. The following rule, then, alters actual behaviour B_s when an avatar is recognised as being within the radius of the doorway:

```
(defrule CONCEPTION::categorise-person-1
   (declare (salience 10))
   ?p <- (MAIN::person (area NONE)
                       (location ?pl))
   (MAIN::door (radius ?rad)
               (location ?dl&:(neq ?dl nil)&:
                         (< (?pl distance ?dl) ?rad)))
=>
   (modify ?p (area DOORWAY))
)
```

This particular agent is reactive; it does not use a hypothesiser. These conceptors therefore categorise avatars and determine as B_e the state that the door controller should be in. Also, as this agent has fixed goals it does not explicitly reason about function.

Door action rules provide an encoded partial plan that allows the door to operate. Each action involves one or more effector activations, and each effector is a java bean. For example, to open the door a 3D object effector is used to change the state of the 3D door object, and the state of the door is updated accordingly:

```
(defrule REACTIVE::open-door-1
   (MAIN::kcdcc_awa_base_ReteAgent (OBJECT ?a))
   ?d <- (MAIN::door (object-id ?id&:(neq ?id 0))
                     (state ?s&:(neq ?s OPEN)))
   ?f <- (MAIN::command (goal OPEN))
   ?eff <- (kcdcc_awa_base_AW3DObjectEffector
              (OBJECT ?reference)
              (objNo ?id&:(neq ?id 0)))
=>
   (modify ?eff (command (get-member
       kcdcc.awa.base.AW3DObjectEffector CHANGE))
      (action "create solid off, visible off"))
   (bind ?no (?reference activate))
   (modify ?f (goal NONE) (sender ""))
   (modify ?d (state OPEN)
              (object-id (?reference getObjNo)))
)
```

5 Communication among Design Agents

Although the agents both reason and communicate in FBS terms, the structure of a communication about about an object is not the same as the structure of an object. So prior to considering a multi-agent example we we need to consider what our agents communicate to each other.

Any multi-agent system must address the issue of communication. At a basic level we have indirect communication since each agent can sense and effect the 3D world server data. Indirect communication does not allow the agents to collaborate with each other, but only be aware of the actions of other agents through sensing the changes in the environment. Direct communication would allow inter-agent communication and collaboration. In developing a model for inter-agent communication, we need to address questions related to the content and intention of the communication. For example, should a room say to a wall “I would like an agent to change so that I am larger”, or “I would like a specific agent to move such that the position of object X which is now outside becomes inside”, or should it say to a particular wall “I would like you to move in direction D”? A crucial difference between an agent and an object is that an object encapsulates state and behaviour realization, but not behaviour activa-

tion or action choice [Jennings, 2000]. So a room agent that dictated to a wall agent what it should do would be a room agent with an aggregation of wall objects. It would not be a room agent collaborating with a wall agent. In our inter-agent communication, agents should communicate their intentions to one another and allow for independent choice of action.

Communication requires a common language, ontology and protocol. The language should be powerful, expressive, not unnecessarily constrain agent communications, and be sensible to all agents. The theory of speech acts [Cohen and Perrault, 1979] is the conceptual basis of agent communication languages (ACL). The ACL from the Foundation for Intelligent Physical Agents (FIPA) and KQML [Huhns and Stephens, 1999] are probably the best known. They are similar, with the FIPA ACL having a well defined semantics [Wooldridge, 2002].

In our multi-agent system we have encapsulated the FIPA ACL within sensors and effectors. Agents that communicate directly use XML and our own simple XML grammar. We adopt the FIPA ACL abstract message structure but not any of the existing implementations because our agents are deliberately not distributed. Indeed, our intention is to design the agent framework such that they can eventually be part of the AW server. XML allows agent developers to extend sensor or effector classes with their own ACL if they so desire, and the text basis of XML minimises functional coupling between sensors and effectors of different agents.

The FIPA ACL describes a set of performatives that constitute the communicative acts [FIPA, 2001]. Examples include the Call-for-Proposal, Inform, Propose, and Request. Each FIPA communicative act should be implemented in accordance with the semantic preconditions defined in terms of belief and intentions. We intend to use this FIPA ACL semantics [FIPA, 2001] only descriptively as a guide for agent designers, not formally.

We do not explicitly represent the ontology an agent uses when communicating. Each XML message contains a public identifier for an associated document type descriptor. This public identifier is taken as identifying the ontology. Agents are encoded to look for particular public identifiers. They assume that they understand any such message. The content of the messages will be reified predicates in order that they have a standard form that other agents can query. Such uninterpreted, reified content that is transmitted is structure and interpreted content is behaviour. These interpreted behaviours are what the receiving agent believes that the sending agent believes.

6 Design of a Multi-Agent Office

Our office is a multi-agent system that includes agents for the zone, walls, a door, a listener, a concierge, and a novelty implementation of the classic "Eliza" chatterbot. The office as it appears in the virtual world is shown in Figure 6. The function of this multi-agent system is to provide an intelligent, virtual meeting place. In this section we shall describe the design of the zone agent and describe functions provided by the other agents.

Function is the result of behaviour from the viewpoint

Figure 7: $ProvideInformationDisplay$ realised using wall graffiti.

of the agent, so each agent desires that its 3D objects achieve their function. If we ascribe function using the terms **ToMake**, **ToMaintain**, **ToPrevent**, **ToControl** [Chandrasekaran, 1994] then the function of a wall in the multi-agent office is threefold:

Wall 1. **Function** $EnsurePrivacy$
ToControl visibility of the room during meetings by adjusting the transparency of the wall and included objects such as windows

Wall 2. **Function** $ProvideVisualBoundary$
ToMaintain an appropriate visual boundary by changing design and appearance of the wall and included objects

Wall 3. **Function** $ProvideInformationDisplay$
ToMake display on wall as graffiti of requested data

A realisation of the $ProvideInformationDisplay$ function is illustrated as Figure 7. It is used to display meeting slides, minutes of meetings and so on using an available wall.

A zone represents agency applied to the space bounded by the office walls, door, and so on. It is not represented in the world by a 3D object but is the entity with which avatars act. What we conceive of as a room is not a 3D object but is a zone. The function of a zone is also threefold:

Zone 1. **Function** $MaintainSpace$
ToMake zone space and ambience

Zone 2. **Function** $MaintainSecurity$
ToMaintain security appropriate to current situation

Zone 3. **Function** $ManageMeeting$
ToMaintain participant seating and chat

The functions of the concierge are to answer questions, schedule meetings, and teleport citizens to selected locations. The functions of the listener are to record chat during meetings, restrict who can chat during a meeting to those sched-

uled to attend, and telegram those recordings as meeting minutes. The functions of the door are to ensure privacy during meetings, and to control access at other times.

Behaviour and structure are represented explicitly by properties constructed during perception, conception and the hypothesiser. They are also represented implicitly by relationships encoded in the rules of these processes. For zone the following are behaviour facts:

```
(deftemplate meeting
   (slot securityState)
   (multislot participants)
   (slot zoneSpace)
   (slot doorSpace)
)
(deftemplate seating
   (slot occupant)
   (multislot neighbourSeats)
)
(deftemplate properties
   "meta-facts of which properties are F, B and S"
   (slot fbs)
   (multislot property)
)
```

Actual behaviour B_s for zone includes how participants are currently seated or not seated. Expected behaviour B_e includes constraints that each participant is currently seated and how they can be seated. All zone sense-data plus the following facts are zone structure:

```
(deftemplate component
   (slot category) ;DOOR,WALL,...
   (slot objectNo)
   (slot model)
   (slot location)
   (slot ownerNo)
   (slot owner)
   (slot timestamp)
   (slot description)
   (slot action)
)

(deftemplate participant
   (slot name)
   (slot session)
   (slot locn)
   (slot citizenNo)
   (slot category) ;from avatar type
                   ;and meeting partipants
   (slot dontLike)
   (slot seat)
)
```

As the current zone space (the 3D region of the world bounded by the zone's component 3D objects) and the current doorway space are dependent variables, they are behaviour not structure.

Perception for zone provides higher level, interpreted structure S. It employs rules to recognise components of this zone (walls, doors and seats), recognise citizens, recognise seats, recognise which seats are neighbours of which other seats, and to interpret inter-agent ACL messages.

Conception for zone performs an FBS analysis transformation $S \rightarrow B_s$. It employs rules to derive current actual behaviour for the zone and door spaces, security state and meeting participants, classification of citizens (participant/not-participant, male/female, doorway/outside/inside), the set of current seats that the seat for a newly arrived citizen is nil, and that set of seats categorised as vacant.

The hypothesiser reasons over zone function by performing an FBS formulation transformation $F \rightarrow B_e$. It recognises currently satisfied functions F, decides on function changes, and which functions to select. The hypothesiser then constructs expected behaviours B_e that include that the set of possible seats for an already seated participant is restricted to their current seat, that the set of possible seats for an unseated participant includes all vacant seats, and constraints such as that no two similarly categorised participants should be next to each other.

Zone includes both reactive and reflective behaviours. Synthesis of a seating plan is reflective and currently uses a constraint satisfaction algorithm. We are investigating an alternative, self-organising method. Other zone actions are reactive, as shown in Figure 8.

The reflective rules synthesise a plan to seat meeting participants. If an action fails to seat a participant and there is at least one free seat, the expected behaviours are reset such that all seats are deemed vacant. If it still fails and there is at least one free seat, the constraints are relaxed. If it still fails then zone will return to the original expected behaviours but with the additional expectation of added seating.

If adding more seating results in crowding, then the zone will communicate with the walls to asked for more room. Functionality required of the room is thus distributed amongst the agents. Some functions are achievable by agents in isolation, others require cooperation. The reason for this is both to simplify the design of the agents and to make them component based. Just as a 3D model of a table can be inserted into a world with minimal influence on objects already in the world, using a multi-agent approach means that we can add or remove agency for objects as we wish. Additionally, for agents which design and alter their world this means that design knowledge of particular virtual artifacts is restricted to agents of that type.

7 The Design Agent Model vs Belief, Desire and Intention

BDI has become a common formalism for rational agents, so comparison with our design agent model is appropriate. We compare our model against BDI because of its increasing popularity, its theoretical underpinning, and its role in the FIPA ACL.

BDI formalisations [Cohen and Levesque, 1990; Rao and Georgeff, 1998; Wooldridge, 2002] commonly use a modal logic with a possible worlds semantics for beliefs, desires and intentions. Amongst the implications of this are that an agent's beliefs are non-contradictory, an agent knows what it knows, is aware of what it does not know, and that agents believe all valid formulae [Wooldridge, 2002]. The reason that BDI requires non-contradictory beliefs is its logical basis; anything can be proven from a contradiction. Logical omniscience presents a problem for design agents with a bounded rationality. Agents that learn by interacting with their environments may be designed to be internally consistent at any given time, but it cannot be guaranteed that future sense-data will not lead to a contradiction of current beliefs. Logical omniscience is also too strong for agents performing design tasks. It is well known from design research that design requirements may change or even initially be incon-

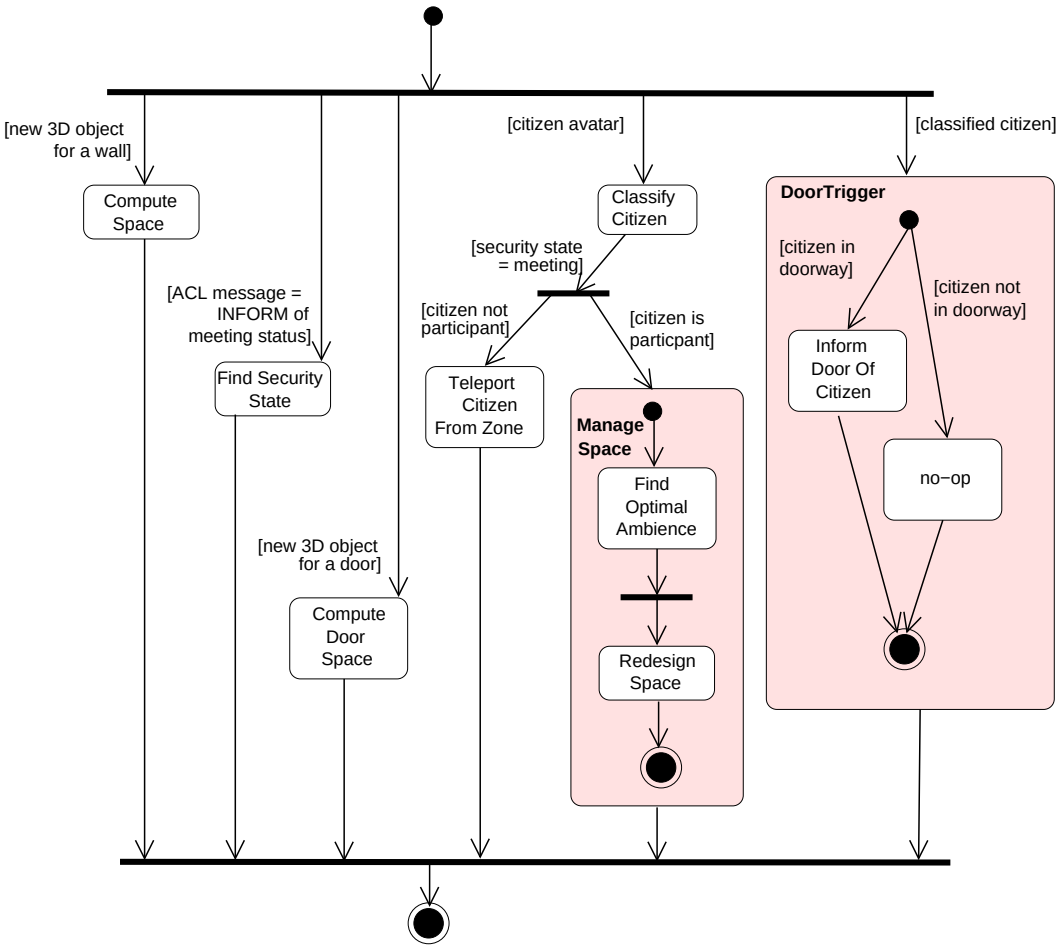

Figure 8: Partial UML activity diagram of zone action.

sistent [Lawson, 1997]. By design tasks here we mean agents that design and build virtual worlds, but we do not exclude other design tasks.

Roughly speaking, we can say there is an equivalence between beliefs in BDI and the percepts and concepts in our design agent model. Further, we can say there is an equivalence between desires in BDI and the goals determined in the hypothesiser and between the intentions in BDI and the actions in our design agent model.

The problem with percepts as beliefs is consistency. No consistent belief should be both believed and disbelieved. Perception determines patterns over sense-data, and sense-data may arise from interaction with an environment that the agent is only be partially aware of. This is especially the case with design agents. An agent that is designing a virtual world will not be aware of all of the implications of design decisions before they are tried, and thus it cannot in advance guarantee consistency of its percepts. A design agent may take an action based on one set of percepts, only later discovering implications of that action that are inconsistent. From a design standpoint this inconsistency is acceptable and occasionally desirable; the inconsistency provides a point from which further designing continues. In any case, having an inconsistency in a belief set should not disqualify an agent from having beliefs [Cherniak, 1986]. It is for this reason that beliefs in our model are modular.

Concepts in our model are abstractions over or relations between percepts. Our conception rules use some form of truth maintenance, explicitly or implicitly, such that its concepts are locally consistent. So concepts count as beliefs.

Desires in BDI are what are often called goals, and intentions are "elements of stable, partial plans of action concerning present and future conduct" [Bratman, 1999]. Rao and Georgeff [Rao and Georgeff, 1998] impose a strong realism on desires by requiring that the agent believes that it can optionally achieve its goals through selected actions. Realism says that an agent perceives objects from its environment whose existence are independent of the agents existence [Honderich, 1995].

For example, consider a light source agent that we shall (for ease of description) call "Sam". Let the level of illumination in the world near Sam be some value `illum`. Sam's constructed memory may include a percept `(dark FALSE)`, in which case we may say that Sam believes that it is not dark. Now "Sam believes that it is not dark" is not the same as "it is not the case that Sam believes that it is dark". Its truth de-

pends on Sam, not only on what the level of illumination is. Sam can rationally believe that it is dark irrespective of what `illum` is.

An alternative, constructive view of desire and belief would allow for objects to be perceived that have little basis in the environment. A cognitive example of this is of perceptual illusions. Strong realism may be acceptable for an agent manipulating an existing virtual world, but it is problematic for a design agent that is designing a new virtual world.

What we call the hypothesiser in our design agent model is in BDI models called deliberation. This is a two part planning process of option generation followed by filtering [Wooldridge, 2002; Bratman *et al.*, 1988]. Intentions as partial plans [Bratman *et al.*, 1988] are the future paths that the agent commits to [Rao and Georgeff, 1998]. Once committed, the BDI agent will not drop an intention unless it is realised or is unachievable. BDI definitions of intention, such as in [FIPA, 2001; Wooldridge, 2002; Rao and Georgeff, 1998], are in terms of persistent goals. This is too strong for our work. An agent may discover something such that a former desire or intention *should not* be satisfied even though it *may* still be satisfied if the agent wanted it to be.

So, we can describe the hypothesiser of our model in terms of deliberation on beliefs, desires and intentions providing we take consistency and realism into account. Loosely adopting BDI terms, we classify the levels of reasoning in our design agent as follows:

- Reflexive behaviours are those that do not involve beliefs.
- Reactive behaviours are those that do not involve intentions.
- Reflective behaviours are those that involve intentions and desires.

There is, therefore, a loose mapping between our model and BDI, and Figure 3 is shown updated as Figure 9 to reflect this.

The FBS mapping onto the model provides an ontological basis for the agent's behaviour. A primary desire of a design agent is to achieve function, and intentions of the agent are partial plans that it believes will satisfy function if activated. Function can be mapped onto concepts, Behaviour can be mapped onto to concepts and percepts and Structure can be mapped onto sense-data and percepts. The reason that percepts appear in both Behaviour and Structure is that it depends on the situation as to which it is. The FBS formalism allows the agent to make use of existing design methods based on it.

8 Conclusion

In this paper we described a design agent model as the basis for all objects in a virtual world. The agent model serves as a formalism for the intelligent behaviour of the objects in the world in a similar manner to the 3D model as a visualisation of the objects in the world. Our agent model for a virtual world is based on the design agent model introduced in [Maher and Gero, 2002; Gero and Fujii, 1999; Smith and Gero, 2002] and is compared to the BDI agent model [Cohen and Levesque, 1990; Rao and Georgeff, 1998; Wooldridge, 2002]. The virtual world provides an opportunity to study the design agent model and communication within a multi-agent system using a multi-user, interactive place that supports professional and educational activities.

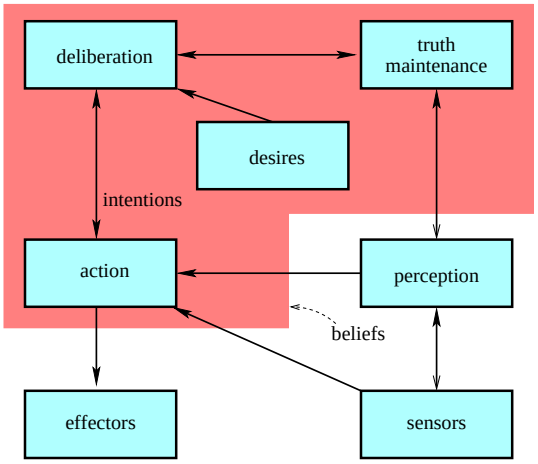

Figure 9: Updated version of Figure 3 to show beliefs, desires and intentions in terms of our model. Beliefs are everything within the large shaded background.

9 Acknowledgements

This work is supported by the Australian Research Council and Active Worlds, Inc.

References

[Beer, 1995] Randall D Beer. A dynamical systems perspective on agent-environment interaction. *Artificial Intelligence*, 72:173–215, 1995.

[Bratman *et al.*, 1988] Michael E Bratman, David J Israel, and Martha E Pollack. Plans and resource-bounded practical reasoning. *Computational intelligence*, 4(4):349–355, 1988.

[Bratman, 1999] Michael E Bratman. Introduction: Planning agents in a social world. In Michael E Bratman, editor, *Faces of intention: selected essays on intention and agency*, pages 1–12. Cambridge, UK; New York: Cambridge University Press, 1999.

[Chandrasekaran, 1994] B. Chandrasekaran. Functional representation and causal processes. *Advances in Computers*, 38:73–143, 1994.

[Cherniak, 1986] Christopher Cherniak. *Minimal Rationality*. MIT Press, Cambridge MA, 1986.

[Cohen and Levesque, 1990] P R Cohen and H J Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42:213–261, 1990.

[Cohen and Perrault, 1979] P R Cohen and C R Perrault. Elements of a plan based theory of speech acts. *Cognitive Science*, 3(3):177–212, 1979.

[Cohen, 1999] Michael H Cohen. The future of human-computer interaction, or how I learned to stop worrying and love my intelligent room. *IEEE Intelligent Systems*, 14(2):8–10, 1999.

[FIPA, 2001] FIPA. FIPA content language library specification communicative act library specification. Document number XC00037H, http://www.fipa.org, 2001.

[Gero and Brazier, 2002] John S Gero and F Brazier, editors. *Agents in Design 2002*. Key Centre of Design Computing and Cognition, University of Sydney, 2002.

[Gero and Fujii, 1999] John S. Gero and Haruyuki Fujii. A computational framework for concept formation for a situated design agent. In K. Hori and L. Candy, editors, *Proc Second International Workshop on Strategic Knowledge and Concept Formation*, pages 59–71. Iwate Prefectural University, Iwate, Japan, 1999.

[Gero and Kannengiesser, 2003] John S Gero and Udo Kannengiesser. A function-behaviour-structure view of social situated agents. In *Proceedings of CAADRIA03*. CAADRIA, 2003.

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

[Honderich, 1995] Ted Honderich. *The Oxford companion to philosophy*. Oxford; New York: Oxford University Press, 1995.

[Huhns and Stephens, 1999] Michael B Huhns and L M Stephens. Multiagent systems and societies of agents. In G. Weiss, editor, *Multiagent Systems: A Modern Approach to Distributed Artifcial Intelligence*, pages 79–120. MIT Press, Cambridge MA, 1999.

[Jennings, 2000] Nicholas R. Jennings. On agent-based software engineering. *Artificial Intelligence*, 117:277–296, 2000.

[Lawson, 1997] Bryan Lawson. *How designers think: the design process demystified*. Oxford; Boston: Architectural Press, 1997.

[Maher and Gero, 2002] Mary Lou Maher and John S Gero. Agent models of 3D virtual worlds. In G Proctor, editor, *ACADIA 2002: Thresholds, Pamona, CA*, pages 127–138. ACADIA, 2002.

[Maher and Gu, 2002] Mary Lou Maher and Ning Gu. Design agents in virtual worlds: A user-centered virtual architecture agent. In J S Gero and F Brazier, editors, *Agents in Design 2002*, pages 23–38. Key Centre of Design Computing and Cognition, University of Sydney, 2002.

[Maher and Skow, 1997] Mary Lou Maher and Bradford Skow. Learning inside the virtual campus, 1997. The Global University: a 21st Century View, RMIT, http://ultibase.rmit.edu.au/Articles/dec97/maher1.htm.

[Rao and Georgeff, 1998] Anand S Rao and Michael P Georgeff. Modeling rational agents within a bdi-architecture. In Michael N Huhns and Munindar P Singh, editors, *Readings in agents*, pages 317–328. San Francisco, CA: Morgan Kaufmann, 1998.

[Smith and Gero, 2002] Gregory J Smith and John S Gero. Interaction and experience: Situated agents and sketching. In J S Gero and F Brazier, editors, *Agents in Design 2002*, pages 115–132. Key Centre of Design Computing and Cognition, University of Sydney, 2002.

[Wooldridge, 2002] M Wooldridge. *An Introduction to MultiAgent Systems*. Chichester, England: John Wiley and Sons, 2002.