JS Gero (ed), *Design Computing and Cognition'04,* 439-457

DESCRIBING SITUATED DESIGN AGENTS

GREGORY J SMITH, JOHN S GERO
University of Sydney, Australia

Abstract. Situated design agents are agents built using concepts from situated cognition. As situated design agents are constructive and interactive, we desire a formalism that starts with interaction and works backwards to what representations of structure, behaviour and function such interaction requires. This paper begins the process of providing a formal underpinning to agency that better corresponds to existing informal descriptions of designing as interactive, situated reflection.

1. Introduction

Dissatisfaction in the design science community with designing being cast by the AI community as problem solving has led to some researchers recasting designing in terms of situated reflection. A situated approach to agency generally holds that agents are social, embodied, concrete, located, engaged and specific (Wilson and Keil 1999). They are social in the sense of being located in a society of agents. Embodied means that actions by the agent are part of a dynamic with the world and results in immediate sensory feedback (Brooks 1991). Concrete, located and specific mean that actions by the agent constrain its behaviour and provide a context within which it reasons and acts. Autonomy is taken to mean that each agent decides by itself what actions to take. A crucial difference between an agent and an object is that an object encapsulates state and behaviour realisation, but not behaviour activation or action choice (Jennings 2000). Engaged means that the agent has an ongoing interaction with the environment; that planning and acting are not separated in time. So when Schön describes designing as reflection-in-action (Schön and Wiggins 1992) he is describing a dialectic view of designing by a situated agent.

For the most part, the recasting of designing into situated reflection and interaction has been informal. Attempts to describe agents less informally tend to start by representing the "mental attitudes" of agents. The FIPA agent communication language (FIPA 2002), for example, has a semantics

based on an underlying realism and BDI modalities of belief and desire. Multi-agent phenomena are then defined in terms of agent beliefs.

Contrary to this, we believe that design agents should be situated, constructive and interactive. We therefore desire a formalism that starts with interaction and works backwards to what representations of structure, behaviour and function such interaction requires. The motivation behind this paper, then, is to start with interactions between situated design agents and work backwards to a formal model of distributed design agents that is suited to design. The research described in this paper is founded on two concepts. First, the situated version of Gero's FBS paradigm (Gero and Kannengiesser 2002), here called sFBS. The sFBS paradigm approaches situatedness by introducing three different kinds of worlds that interact with one another: the external world, the interpreted world, and the expected world. The external world is the world that is outside of the agent. The interpreted world is constructed inside the design agent in terms of sensory experiences, percepts and concepts. The expected world is that which the agent imagines its actions will produce. Second, a model of the computational processes that has been the basis of much of our recent work (Maher, Smith and Gero 2003, Smith, Maher and Gero 2004). The long term aim of this work is to formalise notions such as reflection and common ground. This paper is a start along that path.

Figure 1 shows a world as viewed by an agent called Agent1. Reasoning consists of five processes: sensation, interpretation, hypothesiser, action activation, and effection. Interpretation uses sense-data and expectations to interpret what the agent believes it's world to be. Throughout this paper we use the word “world” to mean that part of the system that the agent is aware of, and “environment” to mean the whole system. The hypothesiser monitors the interpretations of the world, and asserts goals 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.

Three levels of action are possible: reflexive, reactive and reflective. Reflexive action is where sense-data triggers action activation directly. Reflexive actions do not involve beliefs. Reactive actions are where interpretations trigger action activation. These do not involve explicit reasoning with goals and expectations. Reflective action involves the hypothesiser explicitly reasoning about expectations and alternative goals.

To avoid confusion between mathematical function and FBS function, in this paper the word “function” on its own means “mathematical function” and the phrase “FBS function” refers to the ascribed function of an artifact or agent. The theory described here is interactive rather than algorithmic. Agents achieve their goals through situated interaction rather than algorithmic planning. The attitude is one of “everything is allowed that is

not forbidden" rather than the algorithmic attitude of "everything is forbidden that is not allowed" (Wegner and Goldin 1997).

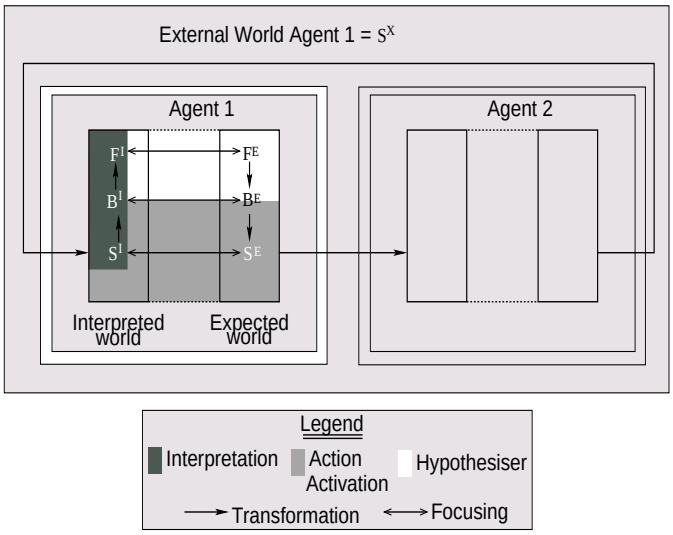


Figure 1. Example world as viewed by an agent Agent1

In this paper we present a formal model of the agents introduced above. In Section 2.1 we describe a system that to an external omnipotent observer appears as a structured set of objects, where the objects are or belong to agents. Section 2.2 then describes the environment and some possible views of it. Section 2.3 reviews situated FBS in this context, with Section 2.4 describing situated action. We then describe an example system in Section 3. We begin by developing symbolic representations of objects and agents. We then represent the environment in similar terms. This representation is used to represent situated FBS before concluding with a representation for situated action.

2. A World of Agents

2.1. OBJECTS AND AGENTS

In this paper we consider a system that to an external omnipotent observer appears as a structured set of objects, where the objects belong to agents. Figure 2 is an example. We call the entire system the *environment*, we call the environment as is viewed by an agent the *external world* of that agent. External world may be shortened to *world* in such cases as will not cause confusion with *interpreted world* or *expected world*. Agents sense the environment and construct an interpretation of that environment as their interpreted world. The way that they wish the environment to be is their

expected world. Agents take actions through effectors by which they attempt to change their external world to make it like their expected world.

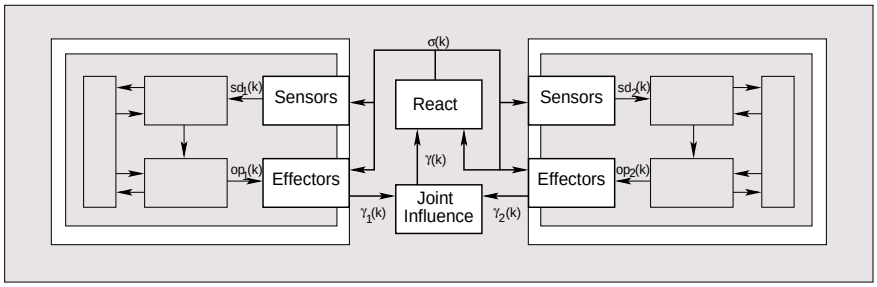


Figure 2. The two agent environment of Figure 1. The unlabelled agent processes are interpretation, hypothesiser and action activation.

We begin here with an *Object* as a set of *exogenous*, or externally visible, properties. The generic types *Value* and *Symbol* are given. All that is assumed of *Symbol* is that elements can be compared for identity. A single property is a *Value* corresponding to an identifying *Symbol*. An *Object* is an entity that has a set of identifiable properties. To facilitate this identification we define *Object* as a partial function from *Symbol*s to *Value*s.

$$Object: Symbol \rightarrow\!\!\!\!\!+\ Value$$

An *Object* may or may not have concealed, autogenous properties. The observable structure of a table, for example, is determined by its atomic structure but that atomic structure is not observable: the observable structure is exogenous, the atomic structure is autogenous. The same distinction applies to humans and to artificial agents. This definition does not require that the set of properties be fixed or even finite. Together with the generic types *Value* and *Symbol*, this allows for *Object* of arbitrary complexity. It can denote an instance of an object oriented language, or it can denote an object sketched by a human designer: the former has a finite number of properties, the latter does not.

The environments we consider here are structured as a distributed set of communicating agents. Each object is constructed by an agent and contains exogenous properties that can be read by any agent that is aware of it's identity. For uniformity we regard agents as accessing the properties of *Object*s with message passing.

An agent is an object that autonomously senses the environment and acts so as to achieve its goals. The environment[1] contains a set of agents $Agents = \{ a_i{:}Agent | i=1..N \}$ where *Agent* is the following tuple:

[1] Some of the following is based on the formalisations in (Wooldridge and Lomuscio 2001, Fagin et al. 1997, Ferber 1999).

$Agent: Exog \times Autog \times Sensor \times Reason \times Effector$
$Exog: Object$
$Autog: \wp Object$
$Sd:$ seq $Object$
$Sensor_i: \sum_0 \times \sum_i \rightarrowtail Sd$
$Reason_i: Sd_i \times \sum_i \rightarrow Op_i$
$Effector_i: Op_i \times \sum_i \rightarrow \Gamma_i$

In this paper we use $\wp X$ to denote the powerset of X and seq X to denote a sequence of X. Each agent a_i contains one object that corresponds exogenous properties constructed from *Exog*. Autogenous properties of the agent are internally constructed and cannot be accessed by other agents. For convenience we let $ex(a_i)$ be the exogenous object of agent a_i and let $au(a_i)$ be the set of autogenous objects of agent a_i.

$ex: Agent \rightarrow Exog$
$au: Agent \rightarrow Autog$

$\forall a_i : Agent \bullet$
$ex(a_i) = first(a_i)$
$au(a_i) = second(a_i)$

Function *first* finds the first element of an n-tuple, *second* finds the second element of an n-tuple, and $x \mapsto y$ denotes a 2-tuple. Functions $Sensor_i$, $Reason_i$ and $Effector_i$ sense the environment, handle inference and memory, and effect the environment respectively. Sd_i are sense-data of agent i and is a sequence of objects on which functions *push* and *pop* are defined. Each effector can push one operator to the environment at a time and so is not a sequence.

Op_i are operations that can be effected by agent i. Γ is the set of influences on the environment by the agents. $\sum$ is the set of global system states and $\sum_i$ is the set of local states of agent i. As $Sensor_i$, $Reason_i$ and $Effector_i$ are the program of the agent, current local state of an agent i is a point in a k dimensional *Value* space:

$$\sum_i = Value^k$$

where k is the cardinality of the sets of exogenous and autogenous properties, or $k = \#ex(a_i) + \#(\#au(a_i))$, where $\#X$ is the cardinality of a set X.

This formalism is simplified if all objects are considered to be contained within an agent. For this reason, all objects not a part of $\{a_i\}$ are assumed to be a part of an imaginary, predefined agent a_0 that represents the causality of

the environment. Causality is discussed in Section 2.3. An example is a message sent but not yet received. Agent a_0 is a mathematical construct only, hence the use of a subscript from outside the *1..N* range. No claim that the environment is an agent is being made. If a message is sent from a_i to a_j, then a message object originates in $\sum_i$, becomes a part of $\sum_0$ in-transit, and finishes in $\sum_j$. The global system state $\sum$ therefore depends on the states of each of the agents.

$$\sum = \sum_0 \times \sum_i \times \ldots \times \sum_N$$

$Sensor_i$ constructs sense-data as a function of the environment and the current local state of the agent. This definition allows for perception that may be incomplete, in error, noisy, or that biases the sensors such as by changing the focus of attention. An external observer may regard the environment as being only partly visible to an agent because some $\sum$ states are treated as being equivalent and so are not viewed as distinct by that agent. The agent's internal view, though, is that perception abstracts away from sense-data to patterns of invariance over interactive experiences.

Agents change properties of external world objects by executing a *push* operation from Op_i.

$$Op_i \supseteq \{a_j : Agent \mid a_i \neq a_j \bullet push(a_i,a_j,o)\}$$

where *o* is a message object. The messages can be of two kinds: a_i *inform*s a_j that a property has a value, and a_i *request*s of a_j that a property has some value. Agent a_i can push a message to a_j an *inform* of content φ if (FIPA 2002):

1. a_i believes that $\varphi \subset \wp Object$ is true:

$$inform(a_i, a_j, a\_j, \varphi) \Rightarrow (\forall s:Symbol; v:Value \mid (s \mapsto v) \in \varphi \bullet ((s \mapsto v) \in ex(a_i) \vee (\exists x:Object \mid x \in au(a_i) \wedge (s \mapsto v) \in x)))$$

2. a_i has a goal that a_j believe φ
3. a_i does not believe that a_j believes φ

A *request* of content φ pushed from a_i to a_j is defined similarly. $Effector_i$ is discussed in the next section.

2.2. ENVIRONMENT

An environment is a tuple $\mathfrak{E} : Objects \times \Gamma \times Op \times React$ where

$$Objects: \wp Object$$

$map: \mathfrak{E} \times Object \rightarrowtail Agent \cup \{\text{null}\}$
$ob: \mathfrak{E} \longrightarrow Objects$

The initial state of the environment is $\sigma(0) \in \sum$, with $\sum$ the global state as defined above. Function *map* is a partial function on an environment that identifies which agent contains an object, or null otherwise. Function *ob* finds the set of *Object*s in an environment. An agent a_i's effector pushes a message object to the environment as a *push* operator. At time k an agent a_i's influence on the environment is $\Gamma(k) \in \Gamma_i$. Now, just as $Sensor_i$ senses the environment differently in different local $\sum_i$ states, so the same operator effected by $Effector_i$ may influence the environment differently in different environmental $\sum_0$ states. The influences of each agent acting simultaneously are combined, and the environment reacts to the joint influence.

$React: \Gamma \times \sum_0 \longrightarrow \sum_0$

The new environment state, therefore, will be (Ferber 1999)

$$\sigma(k+1) = React\left(\bigcup_{i=1}^{N} Effector_i(Reason_i(sd_i(k), \sigma_i(k)))\right)$$

This equation describes the causality in the environment, and is "computed" by "agent" a_0. Each agent senses and acts on a subset of the environment. We call a subset of the environment that is visible or constructed for some purpose a view. The obvious view is the omnipotent one. In the following let $\mathfrak{e} \in \mathfrak{E}$ be a particular environment.

$$omnipotent(\mathfrak{e}) = \{ o{:}Object \mid o \in ob(\mathfrak{e}) \} \cup \{ a_i{:}Agent \mid (\exists\, o{:}Object \bullet map(\mathfrak{e},o) = a_i) \}$$

The multi-agent system (MAS) view is of the environment viewed solely as a multi-agent system. It therefore consists of agents and exogenous objects.

$$MAS(\mathfrak{e}) = \{ a_i{:}Agent \mid ((\exists\, o{:}Object \bullet map(\mathfrak{e},o) = a_i \} \cup \{ o{:}Object \mid \exists\, a{:}Agent \bullet map(\mathfrak{e},o) \wedge ex(a)=o \}$$

The *world* is everything between the effectors and sensors of an agent. It is the external world of that agent, or S_i^X from Figure 1.

$externalworld(\mathfrak{e},a_i) \subset$

$$\{o:Object \mid (\exists a_j:Agent \mid a_j \in MAS(e) \wedge o\ ex(a_j)) \}$$

The view of the environment by a particular agent is of its world plus its autogenous properties.

$$agentview(e,a_i) = world(e,a_i) \cup au(a_i)$$

The view of a person such as a designer interacting with the environment is of a set of objects.

$$personview(e) \subset \{ o:Object \mid (\exists a_i:Agent \mid a_j \in MAS(e) \wedge o = ex(a_j) \}$$

2.3. SITUATED FBS

Agents represent their environment using the situated FBS (sFBS) formalism (Gero and Kannengiesser 2002; Maher, Smith and Gero 2003a) as beliefs of the structure, behaviour and function of objects. Some object properties are obviously structural, such as location. Other properties must be interpreted by an agent. Structure, then, is an interpretation by an agent of what it believes a sensed object is. It is an interpretation of sense-data; sense-data themselves are uninterpreted inputs to the process of interpretation. In Figure 1, S_i^X is the actual external world structure that is visible by a_i (the subscripts *i* are not shown in Figure 1). As the only access to S_i^X by a_i is via sensors, it includes all exogenous objects other than the agent's own. An agent need not sense its own exogenous properties. S_i^I is the interpretation by a_i of what it believes the set of all structures of objects in the environment to be, and S_i^E is the set of expectations of structure.

$$\forall a_i:Agent \bullet$$
$$S_i^X = world(e,a_i) - ex(a_i)$$
$$S_i^I \subset (au(a_i) \cup ex(a_i))$$
$$S_i^E \subset au(a_i)$$

Structure properties can contain values such as location, or can contain relations to other objects. S_i^I can be viewed as a graph of what a_i believes that the world currently is and S_i^E can be viewed as a graph of what a_i believes that the world will be or should be.

In general, behaviour is determined from structure according to some causation. Causation is the relation between two things where the first is

thought of as somehow bringing about the second (Lacey 1996). In the natural world we regard nature as doing the bringing about in the form of the laws of physics. With an artificial world the bringing about is from computations by the agents, so behaviour is determined by whatever the "virtual physics" are. Regardless of whether the causation is natural or artificial, 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.

Behaviour is an interpretation by an agent of what it believes an object does, and FBS function is what the agent believes that an object is for. B_i^I is the set of interpretations of behaviours of objects by a_i, and F_i^I are interpretations by a_i of FBS functions that may have been ascribed to other objects. B_i^E is the set of expectations of behaviours by a_i and F_i^E are expectations of function.

$$\forall a_i{:}Agent \bullet$$
$$interpretedworld(a_i) = S_i^I \cup B_i^I \cup F_i^I$$
$$expectedworld(a_i) = S_i^E \cup B_i^E \cup F_i^E$$
$$au(a_i) = (interpretedworld(a_i) \cup expectedworld(a_i)) - ex(a_i)$$

The process *interpretation* re-computes interpreted structure and behaviour whenever either new sense-data arrive or expectations change. The triggering mechanism can be event-driven, polled, or use a combination of both. For polled interpretation a time sense triggers the pulling of sense-data from the environment. Event-driven interpretation occurs when the environment pushes sense-data into sensors. Regardless of the mechanism, interpretation consists of three partial functions. The first computes interpreted structure from sense-data but biased expectations of behaviour. The second computes interpreted behaviour from interpreted structure and expectations of behaviour. The third computes interpreted FBS function from *request*s from other agents and from chat with persons such as designers.

$$strInterp: Sd_i \times B_i^E \rightarrowtail S_i^I$$
$$behInterp: S_i^I \times B_i^I \rightarrowtail B_i^I$$
$$funInterp: Sd_i \times F_i^E \rightarrowtail F_i^I$$

Computing interpreted structure and behaviour are sequential; computing interpreted FBS function can be in parallel.

$$interpretation = (strInterp \circ behInterp) \,||\, funInterp$$

Figure 3 shows reflexive, reactive and reflective reasoning as a Petri net of an agent a_i, including *interpretation*. Function *Reason* for a_i is everything between *Sensor* and *Effector*.

There are four types of FBS function that can be ascribed by an agent (Qian and Gero 1996): FBS functions that map to static behaviours, those that map to dynamic behaviours, those that map to a set of concurrent behaviours, and those that map to sequential behaviours. Reflective agents explicitly reason over FBS functions; reactive agents have FBS functionality implicit in action rules. Reflective agents use the *hypothesiser* process to compute expectations of behaviour from expected FBS function, and then compare those expectations against the interpreted world. Newly detected differences between expected and interpreted behaviour are asserted as goals for action activation to satisfy. The hypothesiser consists of two sequential processes. The first formulates expectations of behaviour. The second compares expected and interpreted behaviour.

$$Goals \subset Autog$$
$$formulation: \; F_i^E \nrightarrow B_i^E$$
$$evaluation: \; B_i^E \times B_i^I \nrightarrow Goals$$
$$hypothesiser = formulation \circ evaluation$$

Action activation synthesises changes in expected structure for goals asserted from differences between expected and interpreted behaviour.

$$action: Goals \times S_i^I \longrightarrow S_i^E$$
$$effection: \; S_i^E \longrightarrow Op \cup Exog$$

2.4. SITUATED ACTION

For a situated agent, deciding when and how to act is of primary importance. Differences between situations should cause the application of the same knowledge to result in differing behaviour, and the result of such behaviour should allow the agent to learn.

A reflective agent explicitly represents the *interpretedworld*(a_i), the *expectedworld*(a_i), and encoded knowledge of all of the known effects on *interpretedworld*(a_i) of each change in structure that the agent can make.

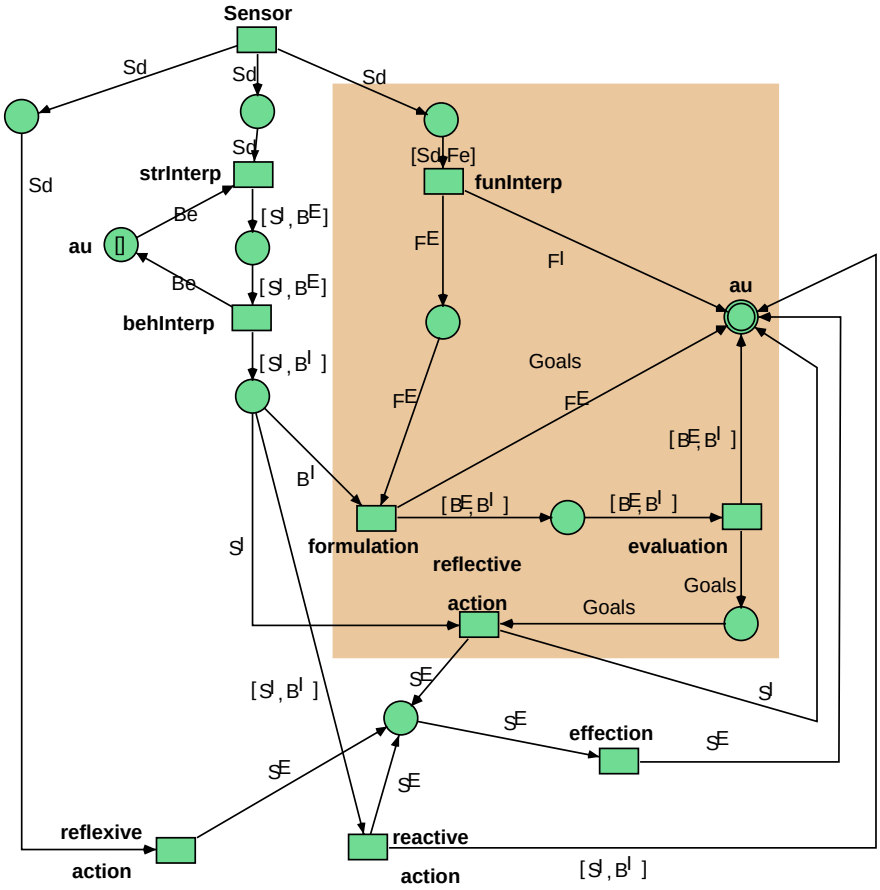


Figure 3. Petri net of reflexive, reactive and reflective reasoning of an agent a_i. It is drawn for a single agent and so all data and function symbols should be read as being subscripted with i. Each transition corresponds to the execution of a function. The double circles are virtual places: the second *au* virtual place is actually the same place as the single *au* place, being drawn that way only to simplify the diagram. The shaded background shows functions only executed during reflective reasoning and not during reflexive or reactive reasoning.

B_i^E is determined from F_i^E by *formulation* using techniques such as constraint satisfaction and abduction, and goals are then determined by comparing B_i^E against B_i^I. These goals, current interpreted structure S_i^I and the encoded knowledge enable reflective *action* to determine a partial order of changes to structure that a_i believes will remove the differences

between B_i^E and B_i^I. This is a partial order on B_i^E, and so deciding on reflective action is equivalent to a search through some solution or plan space. Planning as used here may involve retrieving a pre-compiled plan, explicit runtime planning, or use some other transformation from goals to action sequences.

Reflexive processes use hardwired stimulus-response rules, and reactive processes have no long term memory. Both therefore take actions only according to the current state. They can, however, be very task specific; reflective processes would not need to be so task specific if the computational limitations of planning did not lead to it anyway. One solution is to adopt a hybrid of reactivity and reflectivity. The solution advocated by Horswill (1998) extends that of Agre (1997) and others, using reactive rules on deictic references. Deictic references are indexicals that signify FBS function. The idea is to keep the reactive rules but to change what they signify. Horswill calls his deictic references roles. Each role is a symbol that is bound to a set of properties and is maintained by low level processes. This separates the perceptual processes of identification and localisation. 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 localisation process. As an example, gaze control in robot vision is a localisation process.

Such reactive agents would minimise the explicit inference performed by *hypothesiser*. Instead, a set of *roles* are bound to autogenous structure and behaviour *Objects* (where ran *R* denotes the range of a relation *R*):

$role_i$: *Symbol* $\leftrightarrow$ *Object*
ran $role_i \subseteq S_i^I$

Example roles may be a door agent with a role `the-person-that-needs-clearance` that is bound to an avatar object in a virtual world, or a wall object in a CAD system that binds a role `the-object-that-needs-moving` to picture objects whenever the wall needs to move. Structure interpretation *strInterp* simply observes whatever object the role is bound to and maintains those properties. Because roles have functional meaning, explicit *formulation* and *evaluation* are not needed. Instead, knowledge of FBS function is implicit in reactive rules. Similarly, reactive action rules need not plan a partial order of structure changes from knowledge of the object being acted on. Instead, it just manipulates the object bound to the role.

For such reactive actions to work, though, requires either that the designer of the agent encode the reactive rules such that FBS functions of the agent are maintained with changing structure, or we allow the agent to learn from reflections such that in the future it can react in similar situations. That is, we either implement agents such that they react appropriately in different situations, or we implement learning such that it can recognise how to react appropriately in different situations. Analytical learners such as explanation based learning allows for action sequences constructed from reflection to be used to learn new reactive actions as macro actions. But learning new reactive actions from successful reflective ones may not be enough: to communicate requires common ground (Gero and Kannengiesser 2003). What if an agent is added dynamically to a system at runtime, and that new agent communicates content that is not understood. For simple communicated properties the receiver could respond with a `not understood` message and the initiator could describe the space of that variable. In general, though, to learn common ground in this way requires complex language learning that is beyond the scope of this work.

3. An Example

Figure 4 shows a view of a world on the Active Worlds platform[2]. In this section we describe an example using agents running on this platform. The use here of virtual worlds as a test environment is for convenience; we do not require or intend the theory only apply to virtual worlds. Interested readers should, therefore, refer to Maher, Smith and Gero (2003) and Smith, Maher and Gero (2004) for details of the AWAgent package and its use with the Active Worlds platform.

Often when citizens enter a virtual world for a meeting they all arrive at the same specified location and then stand “on top of each other”. Chair agents self-organise so as to relocate the avatars appropriately. When slides are being shown on a wall or the whiteboard being used they will reorganise, teleporting their assigned avatars with them, around the slide display or whiteboard. Afterwards they may reorganise around a central table in the meeting room.

The environment contains the agents listed in Table 1. Chair agents are constructed dynamically as required. Consider the chair agent a_1, labelled as "chair0" in the table (the others are identical). The FBS function of a_1 is to maintain the equilibrium location of the chair and of a citizen's avatar that is

[2] The virtual world platform Active Worlds, http://www.activeworlds.com, is one that we use for agent testing.

allocated to that chair. Chairs are implemented reactively here, though, and so do not explicitly represent or reason about FBS function.

Figure 4. Meeting room in Active Worlds

TABLE 1. Agents in the example environment.

Agent No.	Symbol	Type of agent	Categories
1	"chair0"	Chair	*obstacle*
2	"chair1"	Chair	*obstacle*
3	"chair2"	Chair	*obstacle*
4	"chair3"	Chair	*obstacle*
5	"wall0"	Wall	*wall, goal*
6	"wall1"	Wall	*wall, goal*
7	"wall2"	Wall	*wall, goal*
8	"wall3"	Wall	*wall, goal*
9	"table"	Table	*obstacle, goal*

Expected structure is a set of roles and a function from sensed object properties to an object category. For this implementation the categories are determined from the Active Worlds *Model*s.

$$\textit{Category: } \{ \textit{ obstacle, wall, goal, null } \}$$

$$S_1^E = \{ \textit{ classify: Model} \leftrightarrow \textit{Category} \mid (\forall x: \text{dom } \textit{role} \bullet \exists_1 y \bullet \textit{classify}(x,y)) \}$$

Chair reactivity is implemented as subsumption, encoded as expectations of behaviour. They are a partial order on the priority of each subsumption

behaviour contribution to resulting action vector, and so each expected behaviour applies to a set of sensed objects. Each expected behaviour is a function of the following type:

$$\textit{Subsumption: Vector} \times \textit{Gain} \times \textit{Gain} \longrightarrow \textit{Vector}$$

where the first *Gain* is a linear gain and the second is a radial gain. The linear and radial *Gain* are tunable parametrically, and so can be adapted should equilibrium not be found.

$$\begin{aligned} B_i^E = \; < & \{calcs \mapsto obstacle,\ linear \mapsto 4,\ radial \mapsto 2, \\ & \quad inhibitedBy \mapsto \{\ \},\ \} \\ & \{calcs \mapsto goal,\ linear \mapsto 3,\ radial \mapsto 100, \\ & \quad inhibitedBy \mapsto \{obstacle\}\ \}, \\ & \ldots > \end{aligned}$$

Behaviour *obstacle* computes an exponentially decreasing (with distance) vector of repulsion from a sensed 3D object such as another chair. Behaviour *wall* and *goal* are similar, except that wall avoidance is like a ball bouncing off a hard surface. Goal attraction computes an exponentially increasing (with distance) vector of attraction a 3D object (the goal, such as a table, wall or whiteboard) towards the chair. Behaviour *random* computes a vector that is a random vector step. Behaviour *anger* maintains the anger of the chair. Every time that the chair is forced to move (the action vector, described below, is above a threshold), the chair gets a little more angry; every time it does not move it gets a little less angry. Obstacles compete and the strongest repulsion vector wins. Some vectors are then subsumed by others if their magnitude is large enough and superposition is used to arrive at a single movement vector for an agent.

Sensor$_1$ is a pseudo-sonar sensor: it senses properties that are interpreted by the agent as an object category and vector from the agent to the closest point on each sensed object. Interpreted structure and behaviour compute properties of each sensed object. For example,

$$\begin{aligned} &\text{Ran } role_1 = S_1^I \\ &S_1^I = <\ldots, \{vector \mapsto (50,0),\ category \mapsto goa\},\ \ldots> \\ &B_1^I = \{\ \{anger \mapsto 0\}, \\ &\qquad <\ldots, \{vector \mapsto (0,0),\ threshold \mapsto 8,\ inhibits \mapsto false\ \}, \\ &\qquad \ldots > \} \end{aligned}$$

S_1^I is a sequence of structure objects, with each object being represented as a set of object properties. The particular sequence element shown is for the goal object. B_1^I contains a behaviour object for this agent, holding the *anger* property, plus a sequence of behaviour objects.

Structure interpretation computes $Sd_1 \times B_1^E \rightarrowtail S_1^I$ such that the following holds.

$$pop(Sd_1) = o \wedge c=classify(o(model)) \wedge c \neq null \wedge$$
$$(|ex(a_1)(location) - o(location)| < |role(c)(vector)|)$$
$$\Rightarrow O(o=role(c) \wedge c=o(category) \wedge$$
$$o(vector)=ex(a_1)(location) - o(location))$$

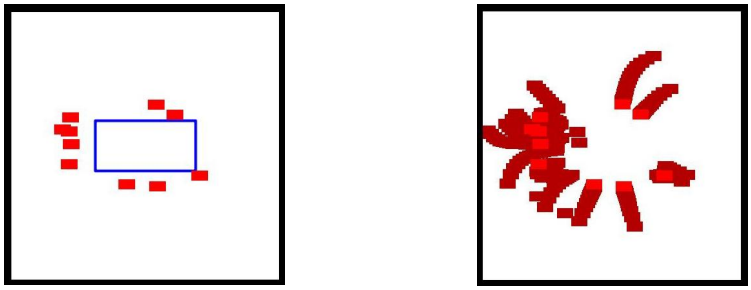
where *O* is the temporal "next" operator and | | is the Euclidean distance metric. Structure interpretation therefore maintains vector and category properties for the nearest sensed object of each category. Behaviour interpretation similarly maintains B_1^I as a behaviour vector for each S_1^I object corresponding to *calcs* from B_1^E, and maintains *inhibits* from the behaviour vector and threshold. So interpreted structure is inferred from sense-data, and interpreted behaviour is interpreted from interpreted structure and expected behaviour. Reactive action then performs the subsumption inference, ensuring that the following holds.

$$\forall n,m:1..\#\, B_1^I;\ \forall c_n,c_m:Category \mid$$
$$n \neq m \wedge c_n = S_1^I(n)(category) \wedge c_m = S_1^I(m)(category) \bullet$$
$$c_m \in B_1^E(n)(inhibitedBy) \wedge B_1^I(n)(inhibits)$$
$$\Rightarrow B_1^I(m)(inbibits) = true$$

The action vector is set to the sum of $B_1^I(n)(vector)$ over all B_1^I for which $B_1^I(n)(inhibits)=false$. Effection changes $ex(a_i)$ to relocate chair according to action vector.

Figure 5 shows one trial run of a set of chair agents around a table object that is the goal. This output is from a simulation written to capture and so better understand and illustrate the behaviour of the agents. In Figure 5 boundaries correspond to walls, the small rectangles correspond to the chairs, and the large rectangle corresponds to the table. In this trial the table is the goal and the chairs are assigned random initial locations. As can be seen, the chairs move towards the goal until an equilibrium is reached

between attraction to the goal, repulsion from the object that is the goal, and repulsion from other chairs.



(a) Snapshot view. (b) Timelapse view.

Figure 5. Trial 1 simulation of the subsumption model of chair self-organisation: view after 40 iterations. (a) is a snapshot view, (b) is a timelapse view. The black rectangle around the boundary are the four walls listed in Table 1, the large polygon in the centre shows where the goal is (*table*), and the small polygons show the positions of the chairs. The darker small polygons on the timelag view shows a timelag view of the chairs, indicating their movement towards the goal.

Notice however that they do not line up evenly around the goal. One price paid for a flexible, situated system of computationally efficient, independent agents is that their behaviour is not globally constrained. That is, some agents reach what they believe to be a minima when in fact it is only locally so. Whenever *anger* reaches a threshold the agent gets "upset" and moves off a random amount in a random direction. So some chairs quickly reach an equilibrium position around the goal, at which time their anger reduces. Some, however, will be prevented from reaching equilibrium by others. The slower ones get repelled by the faster ones, get angry, and take a random step. There are a number of ways of handling this, such as by decaying the behaviour parameters, or by having walls adapt to anger so as to change room geometry.

The subsumption implementation is reactive and situated but uses no concepts. A chair does not reason about what it is attracted to or repelled from and does not reason about the nature of the space it occupies. We could consider a reflective version as a planning problem using constraint satisfaction, with the constraint properties being the spatial locations of chairs and the constraints being both the geometry and on the set of citizens to be seated. One difference would be that it would now be a non-distributed task rather than a distributed reactive one. Here it must first pre-allocate spaces, whereas the reactive version treats the space as continuous. On the other hand we can extend the reflection of the agent by adding constraints. For example, that adjacent chairs should not chair citizens that do not like

each other. A similar effect could be achieved by the reactive implementation by adding more subsumption modules, but it is well known from robotics work that there comes a point beyond which such reactive architectures do not scale up (Murphy 2000).

Within the bounds of what they perceive and reason over, these agents are robust to new situations. The chairs adapt to changing goal objects, changing numbers of chairs and changing room size without needing to do any planning. Their situation directs what behaviours should activate. Such reactive reasoning couples perception tightly to action, avoiding the frame problem by eliminating the need to model the environment (Murphy 2000). On the other hand, the chairs do not achieve any kind of optimal distribution because the view of each agent is local.

The reactive example here can be extended to facilitate designing from within the design. Consider, for instance, a set of room agents together with wall, floor, ceiling, furniture agents and so on. If the designer decides to change the shape of a room then adjoining walls, floors, ceiling, and furniture would automatically shift to new equilibrium locations. Combining it with explicit communication[3]. would allow adjoining rooms to negotiate to decide which wall(s) should move.

4. Conclusion

We have defined an environment as a distributed system of agents that communicate via message passing. Agents need not be distributed and need not explicitly use message passing. We defined multiple views of an environment because the world viewed by one agent need not be the same as another. This allows for situated agents to not only sense different subsets of objects but to then construct their own interpretations. We defined reasoning and communication by agents in sFBS terms so as to facilitate the description of such reasoning and interaction as it applies to designing without prescribing what that reasoning should necessarily look like. That is, instead of starting with representations of mental entities and then describing communication in those terms, we start with interaction and describe what needs to be represented so as to facilitate it.

This paper marks the beginning of a larger enterprise: to provide a formal underpinning to agency that better corresponds to the informal casting of designing as interactive, situated reflection. Future work needs to further constrain relations between structure, behaviour and function of agents with respect to situated agents and this framework, to refine the semantics of

[3] See (Maher, Smith and Gero 2003, Smith, Maher and Gero 2004) for further discussions of agent communication and Active Worlds.

inform and *request* messages in sFBS terms, and to describe what common ground between agents means in these terms.

Acknowledgements

This work was supported in part by an Australian Postgraduate Award at the University of Sydney, Australia and by a University of Sydney Sesqui R&D grant.

References

Agre, PE: 1997, *Computation and Human Experience*, Cambridge University Press, Cambridge, UK.

Brooks, RA: 1991, Intelligence without reason, *Proceedings of the 12th International Conference on Artificial Intelligence*, pp.569-595.

Fagin, R, Halpern, JY, Moses, Y and Vardi, MY: 1997, Knowledge-based programs, *Distributed Computing* **10**: 199-225.

Ferber, J: 1999, *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*, Addison Wesley, Harlow.

FIPA: 2002, *Agent Communication Language Specifications*. Version H.
http://www.fipa.org/repository/aclspecs.html

Gero, JS and Kannengiesser, U: 2002, The situated function-behaviour-structure framework, *in* JS Gero (ed), *Artificial Intelligence in Design '02*, Kluwer, Dordrecht, pp. 89-102.

Gero, JS and Kannengiesser, U: 2003, Towards a framework for agent-based product modelling, *in* K Gralen and U Sellgren (eds), *International Conference on Engineering Design, ICED 03, Stockholm, Sweden*, The Design Society, August 2003, pp. 1621-1622. Abstract - full paper on CD-ROM, ISSN/ISBN: 1-904670-00-8.

Horswill, ID: 1998, Grounding mundane inference in perception, *Autonomous Robotics* **5**(1): 63-77.

Jennings, NR: 2000, On agent-based software engineering, *Artificial Intelligence* **117**: 277-296.

Lacey, AR: 1996, *A Dictionary of Philosophy*, Routledge, London.

Maher, ML, Smith, GJ and Gero, JS: 2003, Design agents in 3D virtual worlds, *IJCAI Workshop on Cognitive Modeling of Agents and Multi-Agent Interactions*, pp. 92-100.

Maher, ML, Smith, GJ and Gero, JS: 2004, Situated agents in virtual worlds, *Working Paper*, Key Centre of Design Computing and Cognition, University of Sydney (in preparation).

Murphy, RR: 2000, *An Introduction to AI Robotics*, MIT Press, Cambridge, MA and London.

Qian, L and Gero, JS: 1996, Function-behaviour-structure and their roles in analogy-based design, *Artificial Intelligence in Engineering Design, Analysis and Manufacture* **10**: 289-312.

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

Wegner, P and Goldin, D: 1997, Interaction as a framework for modeling, *in* PPS Chen (ed), *Conceptual Modeling: Current Issues and Future Directions*, Springer, Berlin and New York, pp. 243-257.

Wilson, RA and Keil, FC: 1999, *The MIT Encyclopedia of the Cognitive Sciences,* MIT Press, Cambridge, MA.

Wooldridge, M and Lomuscio, A: 2001, A computationally grounded logic of visibility, perception, and knowledge, *Logic Journal of the IGPL* **9**(2): 273-288.