A Representation Language for a Prototype CAD Tool for Intelligent Rooms

Daniel Barker[1], Andy Dong[2]
[1] University of Sydney
[2] University of Sydney

Abstract

Intelligent rooms are a type of intelligent environment which enhance ordinary activities within the confines of a room by responding to human interaction using pervasive and ubiquitous computing. In the design of intelligent rooms, the specification of how the intelligent room enacts intelligent behavior through computational means is as integral as the geometric description. The self-aware and context-aware capabilities of intelligent rooms extend the requirements for computer-aided design tools beyond 3D modeling of objects. This article presents a Hardware as Agents Description Language for Intelligent Rooms (HADLIR) to model hardware in an intelligent room as "hardware agents" having sensor and/or effector modalities with rules and goals. End-users describe intelligent room hardware as agents based on the HADLIR representation and write agent rules and goals in Jess for each hardware component. This HADLIR agent description and the requisite software sensors/effectors constitute "hardware agents" which are instantiated into a multi-agent society software environment. The society is then bridged to either a virtual environment to prototype the intelligent room or to microelectronic controllers to implement a physical intelligent room. The integration illustrates how the HADLIR representation assists in the design, simulation and implementation of an intelligent room and provides a foundation technology for CAD tools for the creation of intelligent rooms.

Introduction

The publication of Norbert Wiener's *Cybernetics* in 1948 on the reciprocal nature of man-machine symbiosis and the thesis of a (mathematically) functional relation between human brains and machines influenced architectural theory at least since Gordon Pask's (Pask, 1969) publication of *The architectural relevance of cybernetics*. The notion that built environments and artificial objects within that environment should perform as embodied extensions of human minds and bodies offered tantalizing potentials to conceptualize new types of experiences within those spaces. While the range of actual experiences humans could conceive is possibly too multifarious to be designed *a priori*, the proposition has been that those experiences can be designed for and manifested during operation of these "intelligent" built environments. However, architectural designers needed to wait until the technologies caught up with their visions for intelligent architecture.

Today, intelligent built environments appear in places ranging from research labs such as the MIT 835 Intelligent Workspaces project to residential homes such as The Gates Estate. One type of intelligent environment, the **intelligent room**, is described by Hirsh (Hirsh, Coen, Mozer, Hasha, & Flanagan, 1999) and Coen (Coen, 1998) as an integration of sensor and effector modalities to enhance ordinary activities within the physical boundary of a room. Intelligent rooms incorporate hardware for sensing and effecting such as wireless sensors (e.g., "motes"), overhead projectors, array microphones, smart whiteboards and virtual reality equipment. Applications of intelligent rooms include intelligent conference rooms (Coen, 1998), automated homes (Intille, 2002) and kids' play rooms (Bobick et al., 2000).

One hurdle in the design of intelligent rooms is the lack of computer-aided design (CAD) and simulation tools, particularly those accessible to non-computer scientists and electrical engineers. Physical rapid-prototyping components such as Smart-Its (Gellersen, Kortuem, Schmidt, & Beigl, 2004) simplify the physical implementation of computational objects into intelligent rooms, but do not offer a means to develop and test a virtual prototype of an entire intelligent room plus its associated computational objects prior to implementation. For the most part, an intelligent room is designed and constructed by a pervasive computing expert as a custom integrated hardware and software system *in situ*.

A CAD package which contains pre-modeled drafting elements such as cameras and infrared sensors is probably insufficient for an intelligent room (architectural) designer. The computational elements drawn into the background of intelligent rooms suggest that the type of computer-aided design tool necessary to design intelligent rooms is more than the 3D modeling and drafting currently available in software packages such as AutoCAD. While a complete discussion of the requirements for a CAD system for intelligent rooms is beyond the scope of this article, we believe that such a CAD system should enable the designer to model and simulate the following three aspects of an intelligent room, the geometric modeling aspects notwithstanding:

Sensing:
- to detect human activities including speech, touch, presence, and movement
- to be self-aware within a finite bounded space

Thinking:
- to behave as an intelligent system with a knowledge representation
- to coordinate sensors and effectors

Effecting:
- to express audio and visual effects

Thus, a CAD tool for intelligent room should satisfy at minimum the following requirements:

1. A means to model arbitrary computational (e.g., pervasive computing components) hardware in the intelligent room
2. A means to specify, model and simulate the function and behavior of the computational hardware in a virtual environment including interoperation
3. A means to implement the software functionality of the intelligent room in a physical environment
4. A means to model the geometry (i.e., 3D model) of the intelligent room and associated hardware

A 3D physical (structural) model of an intelligent room (Requirement 4) could be readily modeled in a persistent, virtual environment such as Active

Worlds [www.activeworlds.com] or Second Life [www.secondlife.com]. However, these 3D virtual environments contain no native capability to specify and model the function and behavior of pervasive computing elements in the intelligent room or to implement the intelligent behavior of the intelligent room in an actual physical environment.

To address these deficiencies, and partially satisfy the first and second requirements of an intelligent room CAD tool, we developed the HADLIR representation. Our eventual aim is to create a set of CAD tools which would allow a designer to specify the function, structure and behavior of the computing hardware constituting an intelligent room and the intelligent, goal-oriented behavior of the room itself. Another research project at the Key Centre of Design Computing and Cognition is investigating the design of an intelligent room as a type of curious agent, that is, an affective agent whose behavior is partially determined by the cognitive-behavioral emotion of curiosity. In support of the curious agents for intelligent rooms research project, it is first necessary to create a CAD tool to specify, model and simulate the behavior of an intelligent room. The CAD tool should make possible the specification, modeling and simulation of the behavior of the intelligent room with the pervasive computing hardware operational. Likewise, the specification and modeling should permit the designer to program intelligent behavior, that is, the "thinking" aspect of the hardware. For this research project, these objectives are addressed in part by modeling hardware in intelligent rooms as (software) agents having sensor and/or effector modalities with a descriptive language called the Hardware as Agents Description Language for Intelligent Rooms (HADLIR).

Deploying software tools for the design of intelligent rooms as comprised of sensing and effecting hardware modeled as agents follows in the ideas by architects before us who proposed agents as a way to implement "intelligent architecture." For example, Krueger (Kreuger, 1996) proposed a biological model for intelligent and interactive architecture implemented as a goal-directed agent (Kreuger, 1998). Goal-oriented behavior is an integral characteristic of the HADLIR representation because intelligent systems must call upon knowledge (i.e., operate at a knowledge level) to formulate a set of actions (plan) that may achieve a given goal or solve a problem. An intelligent room must also review the results of the plan and compare those results to the desired goal. If the plan failed to achieve the goal, alternative plans must be formulated and may involve a request to other systems. As Hunt (Hunt, 1998) argued, intelligent buildings should do more than sense and control the environment; they should also "anticipate the behavior of its users through knowledge based on experience." Believing that intelligent architecture consists of more than the built environment, Gage (Gage, 1998) suggested deploying intelligent objects throughout these spaces, objects which communicate and converse with each other and the occupants. Because the HADLIR representation can model these intelligent objects as software agents, designers may prototype and simulate in a virtual environment the technological means by which these objects act together with occupants before implementation.

This paper focuses on the development of the HADLIR representation as a part of foundational technology for intelligent room CAD tools. The next section details the HADLIR representation and an interactive tool for describing hardware in intelligent rooms based on the HADLIR representation. Thereafter, a demonstration of the HADLIR representation in modeling and simulating a pressure mat in a virtual and physical intelligent room is provided. The paper concludes with a discussion on the current limitations of the system and future work.

The HADLIR Representation

Background

Recently, ontologies have been implemented in various intelligent rooms and pervasive computing environments as a potential solution to describe the computational hardware constituting an intelligent room and the intelligent goal-oriented behavior of the intelligent room. Recent implementations include the ontology and Semantic Web technology integrated into a CORBA-based infrastructure of an intelligent environment named GAIA (Ranganathan, McGrath, Campbell, & Mickunas, 2004) and the COBRA-ONT ontology developed for context aware intelligent environments (Chen, Finin, & Joshi, 2004). The GAIA intelligent environment primarily addresses the hardware configuration

problems; the GAIA technologies incorporate ontologies to augment system-level services such as configuration management and interoperation of computational components. The COBRA-ONT ontology, on the other hand, deals with the description of the state and behavior of objects in intelligent environments. The COBRA-ONT ontology describes four distinctively related categories that include: ontologies about physical places, ontologies about agents (both human and software agents), ontologies about the location context of the agents, and ontologies about the activity context of the agents. While formal ontology systems such as these offer the highest levels of specificity and interoperability, they also require the most up-front work and "ontological commitments" from the users and are rather inflexible to subsequent changes.

For this research, we do not attempt to create a complete ontology for intelligent environments for the reasons pointed out by Chen (Chen et al., 2004) and also due to the overhead in ontology development, the federation of ontologies, ontological commitments that make knowledge sharing possible, and the requirement to update the ontology as new hardware devices are added. Instead, we propose to describe pervasive computing hardware in intelligent rooms from the standpoint of (software) agents.

Representation of Hardware as Agents in HADLIR

Multi-agent systems appear to be the dominant approach for creating intelligent rooms. Multi-agent systems such as the ScatterBrain system (Coen, 1997) are already found in operational implementations of intelligent rooms. Because agents are simple entities, agent based systems relinquish the need for complex centralized control (Coen, 1998). As such, the key philosophy of the HADLIR representation is to minimize the descriptive overhead for each component of hardware in the intelligent room to what is sufficient to characterize the piece of hardware as an agent.

Because of the predominance of the agent model in intelligent room software architectures, the idea behind the HADLIR representation is to model hardware devices as (software) agents having sensor and/or effector modalities with rules and goals. The HADLIR representation describes the pervasive computing hardware constituting an intelligent room as (software) agents ("hardware as agent") having sensor and/or effector modalities with rules and goals written in Jess. Many hardware devices can be configured for intelligent rooms; some examples include Web cameras, pressure mats, lights, speakers, microphones, and display panels. Regardless, each type of device in its interaction with the intelligent room environment and occupants could be classified as either a sensor, an effector, or a sensor and an effector. Figure 1 describes the sensor/effector modality for some typical hardware devices found in intelligent rooms. Consequently, the function of the device as a sensor or an effector could be characterized by one or more behaviors where each behavior would include associated actions.

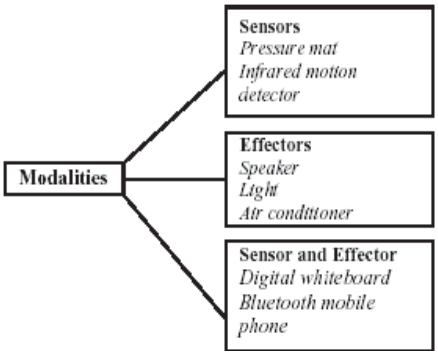

Figure 1. *Example of hardware devices as sensors, effectors and sensors and effectors*

We define the term "HADLIR agent description" as the hardware as agent description plus the rule base. The components of a "hardware agent" are the HADLIR agent description and the agent software sensors and effectors written in a programming language such as Java. Software sensors and/or effectors allow the hardware agent to interact with the agent environment. As illustrated in Figure 2, the HADLIR agent description is combined with requisite Environment Descriptors and a Communication Bridge in order for the agent environment to communicate with the intelligent room environment.

Because all practical intelligent environments would consist of multiple components of hardware, once each piece of hardware has been represented as a hardware agent, the hardware agent is then added into a device library and instantiated into a multi-agent society (MAS).

Within the MAS, the hardware agent may maintain awareness of itself and of other hardware agents and the environment (Huhns & Seshadri, 2000). For integration with a CAD tool, the MAS is connected to either a virtual environment for simulation or a physical room for implementation as shown in Figure 3.

The hardware as agent description in the HADLIR representation is comprised of a taxonomic framework derived from an entity-relation model of the potential hardware devices to their behavior(s) and action(s). The framework classifies behaviors and actions associated with the hardware in a taxonomic hierarchy. The framework includes a set of representational terms to describe a specific behavior and action. Ranganathan (Ranganathan et al., 2004) implements a SVO (Subject–Verb–Object) format in their ontological framework. Consequently the structure of their predicates is Context Type (<Subject>, <Verb>, <Object>). For instance, the ontology declares that the location predicate must have a subject which belongs to the set of persons or things, a verb or preposition like ''inside'' or ''entering'' and a location, which may be a room or a building.

We take an entity-relation approach. For clarity in the following discussion, we provide the entity-relation diagram in Figure 4 to illustrate the relationships and organizational structure of the HADLIR representation. The behavior-type and action child elements are used to describe the actions a behavior performs. The behavior-type child element type describes the form of data, for example, force, distance, and time, that the hardware agent senses and/or effects. The action child element name is used in conjunction with behavior-type child elements name and type to describe the specific sensors and/or effectors the hardware agent needs to be configured with. For example, a pressure mat would have a behavior to sense if an object were present on the pressure mat. For this behavior, the behavior-type name is sensor, the behavior-type type is force and the action name is measure. To a certain extent, because the HADLIR agent descriptions adhere to a taxonomic hierarchy and set of representational terms, one could classify the HADLIR representation as an ontology (Gruber, 1993). According to this definition of an ontology, the HADLIR representation is an ontology of hardware as agents in intelligent rooms rather than an ontology for intelligent rooms. The latter is the aim of the GAIA and COBRA-ONT ontologies.

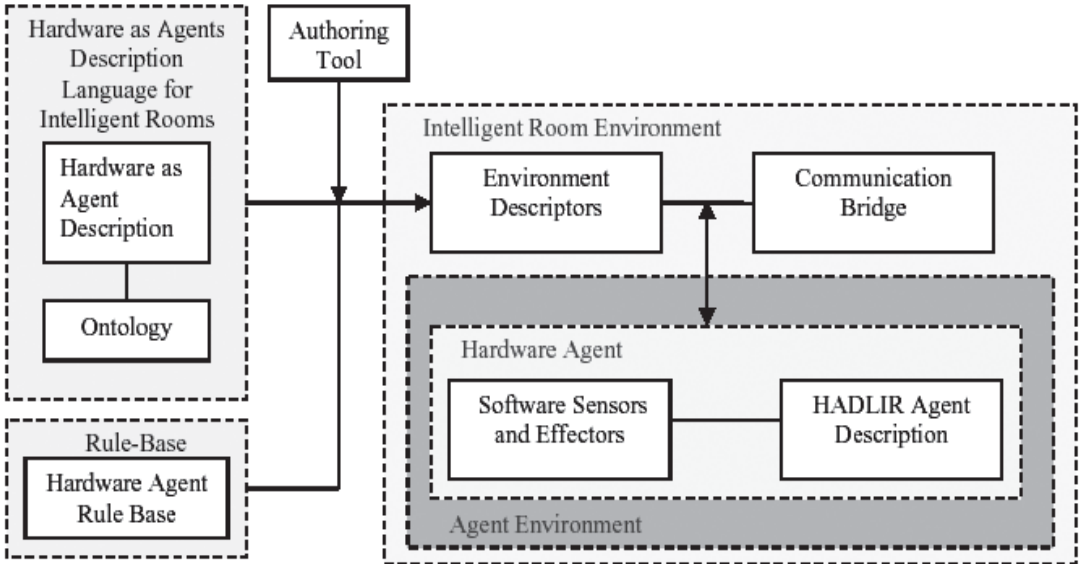

Figure 2. *Dynamic creation of a Hardware Agent*

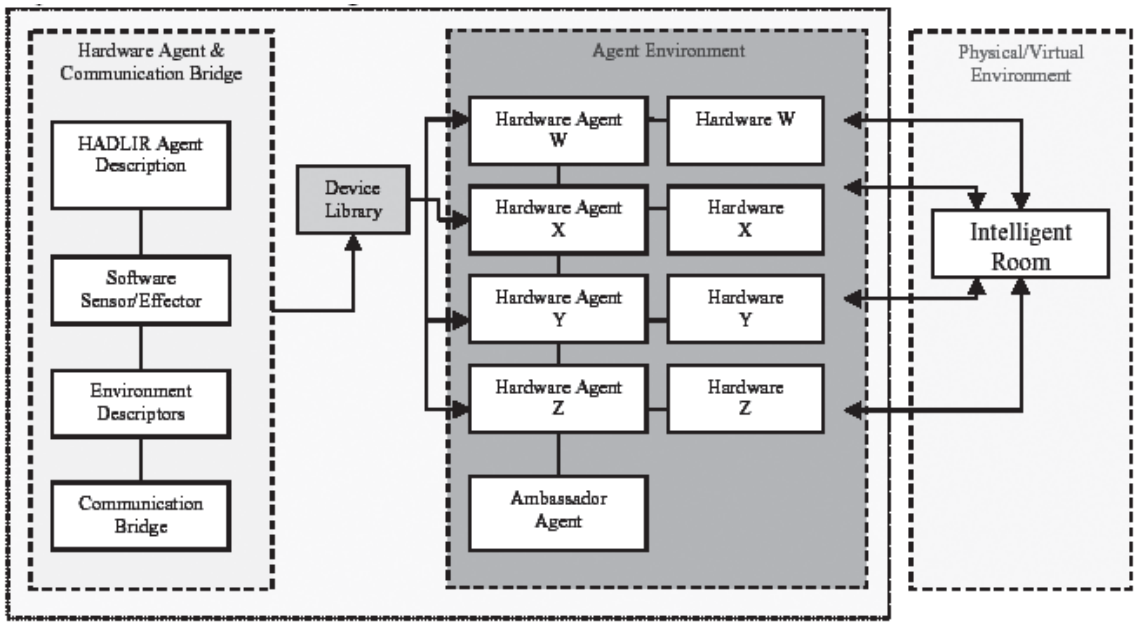

Figure 3. *Agent Architecture of a CAD Tool for Intelligent Rooms*

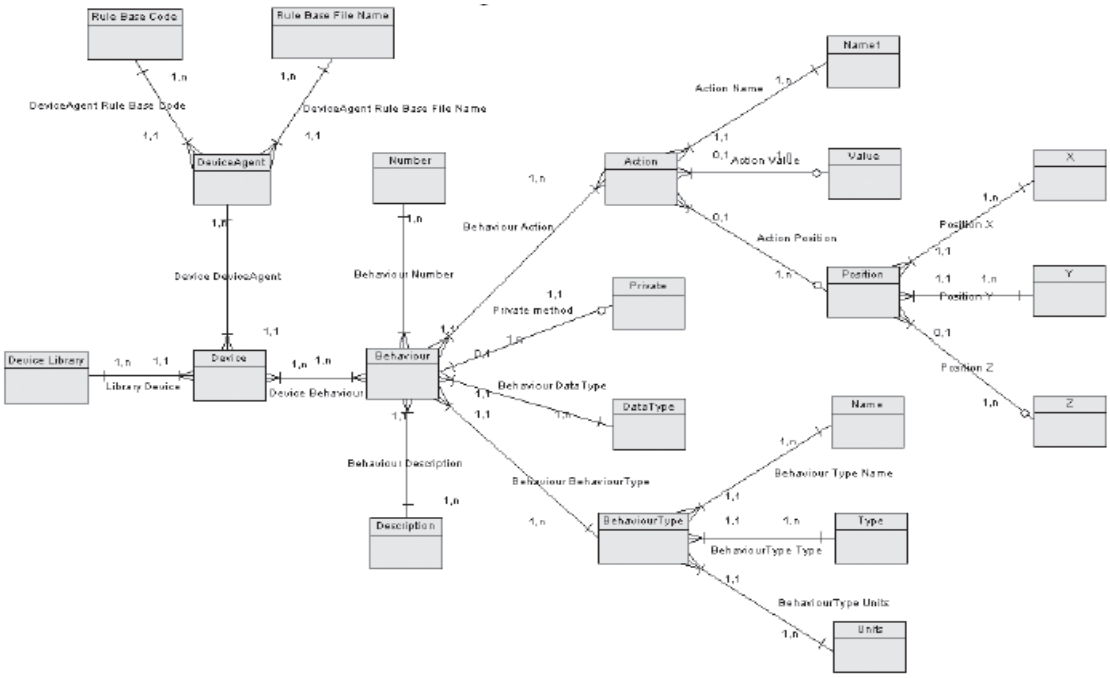

Figure 4. *Entity relationships for hardware devices as agent descriptions*

Furthermore, the HADLIR representation is intended to describe situated agents (Smith & Gero, 2004). Situated agents are agents built using concepts from situated cognition. They achieve goals through situated interaction rather than algorithmic planning. The situated agent reasoning model consists of five processes:

1. sensation: senses the environment
2. interpretation: uses sensor data and expectations to interpret what the agent believes the environment to be
3. hypothesizer: observes interpretations of the environment, and asserts goals based on the agent's view of itself in the environment
4. action activation: reasons about the steps to achieve a goal and trigger the effector(s) to make changes to the environment
5. effection: effects (alters) the environment

Situated agents move away from traditional agent communication languages such as CORBA because they are constructive and interactive. Moreover situated agents are capable of reflexive, reactive and reflective action. Situated agents facilitate three types of actions:

1. Reflexive action: directly initiated from sensor data
2. Reactive action: initiated by interpretations of the agent, no explicit reasoning with goals and expectations occurs
3. Reflective action: hypothesizer explicitly reasons about expectations and alternative goals.

Jess Rule Base

Situated hardware agents instantiated from the HADLIR representation also require rule-bases to be able to sense, interpret, hypothesize, take action, and effect the intelligent room. The second part of the HADLIR representation is the rule-base. The implication of situated agents on the HADLIR representation is that each device must also have a rule-base to encode the device's rules and goals. Jess (Friedman-Hill, 2003) is a rule-based language extension to Java that supports the creation of software agents capable of reflexive, reactive and reflective action. We integrated Jess with an agent environment package (Smith, 2003) to create an operational multi-agent society. This multi-agent society enables the agents to communicate amongst each other and with the environment. The rules control the agent reasoning. Facts are obtained from the environment using sensors. Rules use the facts obtained to determine the state of the environment. Jess rules have the following structure:

```
(LHS          (IF
    =>            =>
RHS)          THEN)
```

The LHS of the rule contains the conditions that need to be satisfied to fire the RHS of the rule. The RHS of the rule contains the action of the rule. Hence the Jess rules are structured much like IF THEN statements. The rule base has MAIN, STRUCTURE, BEHAVIOR, and FUNCTION modules for facts. These four categories describe *a priori* facts about the hardware agent. The rule base has INTERPRETATION, HYPOTHESIZER and ACTION modules for rules. The INTERPRETATION module uses sense-data and expectations to interpret what the agent believes the environment to be. The HYPOTHESIZER module observes interpretations of the environment, and asserts goals based on the agent's view of itself in the environment. The ACTION module reasons about the steps to achieve a goal and triggers the effector(s) to make changes to the environment.

Figure 5 describes the basic components of the situated agent architecture underlying the HADLIR representation. One principal requirement of the agent architecture is for the HADLIR representation to have the capability to configure an agent as a set of sensors/effectors and a rule base. The agents in the agent package (Smith, 2003) are reteagents. The reteagent is an implementation of an agent in Jess excluding the sensors and effectors (Smith, 2003). The sensors and effectors are instead implemented in Java. Sensors are Java objects that sense an agent's environment and effectors act on that environment. Sensor data is stored in the Jess working memory. The Java-based sensors send their messages to Jess, which stores the messages in its Working Memory. Messages from other agents are also recorded in the Jess working memory by the sensors. That is, each time new sensor data is received, a new fact (actually, a Java Bean) is asserted into Jess working memory. The Rete

Engine implements its goals through the Java-based sensors. Both the sensors and effectors are realized in the agent environment.

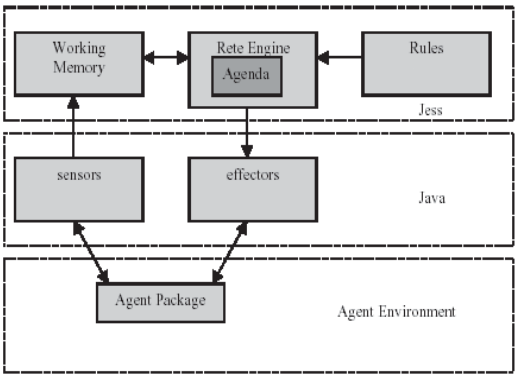

Figure 5. *Architecture of ReteAgent*

HADLIR Authoring Tool

The designer creates HADLIR agent descriptions for all of the intelligent room hardware as agents having sensor and/or effector modalities using the HADLIR authoring tool. The HADLIR authoring tool produces XML. Using XML permits the semantics and structure of the HADLIR hardware as agent description to be enforced with XML Schema. The XML Schema ensures that:

- The element relationships in the device descriptions are maintained
- All necessary elements are present in each hardware as agents description
- The field content of elements is appropriate.
- The elements and attributes are in the correct order.

The JavaScript-based HADLIR authoring tool shown in Figure 6 applies dynamic tables, inputs, select fields and buttons to generate a HADLIR hardware as agent description. Any number of behaviors can be added to the hardware as agent description. The HADLIR authoring tool produces an XML file similar to the one shown in Figure 7 which was produced for the pressure mat example described later in the paper. Once generated, the hardware as agent description is combined with the rule base and inserted into the collection of HADLIR agent descriptions for intelligent room hardware devices and organized into a device library.

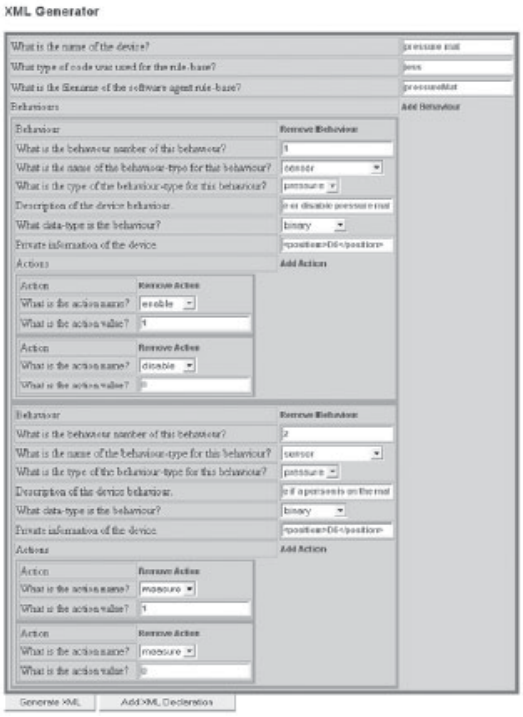

Figure 6. *HADLIR Generator*

```
<device name='pressure mat'>
 <deviceAgent>
  <ruleBaseCode>jess</ruleBaseCode>
  <file>file:pressureMat</file>
 </deviceAgent>
 <behavior>
  <number>1</number>
  <behaviorType>
    <name>sensor</name>
    <type>pressure </type>
    <units>Pa</units>
  </behaviorType>
  <description>enable or disable pressure
mat</description>
  <dataType>binary</dataType>
  <private>
   <position>D6</position>
  </private>
  <action>
    <name>enable</name>
    <value>1</value>
  </action>
  <action>
    <name>disable</name>
    <value>0</value>
  </action>
 </behavior>
</device>
```

Figure 7. *XML Description for a Pressure Mat*

Demonstration of the HADLIR Representation

Using HADLIR to Describe a Pressure Mat

We will illustrate the implementation of the HADLIR representation with a simple sensor, a pressure mat. A pressure mat is a pressure sensor which has no dependencies, that is, does not rely on other hardware devices to operate. As will be described in the Implementation in a Physical Environment section, the pressure mats provide an array of sensors for the production of generative digital media. Occupants walk on top of the pressure mats (hidden underneath the carpeting), thereby producing a sensation. The speed and direction of their movements affects the L-system that generates the digital media. To the occupants, the intelligent room appears to react to their movements without the occupants explicitly "interacting" with the room.

The prototype CAD tool for the demonstration consists of the following components:

1. The HADLIR representation and authoring tool
2. A multi-agent society software package (Smith, 2003)
3. A virtual environment modeling system (Active Worlds)
4. A microelectronics control system (Teleo and Max)

First, the designer creates the HADLIR hardware as agent description using the authoring tool. Next, the designer creates a HADLIR agent description by combining the previous HADLIR hardware as agent description with an appropriate Jess rule base. The Jess rule set is dependent only on the hardware, that is, it is constant across the virtual and physical intelligent room environment. The pressure mat INTERPRETATION module has rules that do the following:

- Locate the pressure mat in the environment and load the rule modules.
- Collect local sensor data.

The pressure mat HYPOTHESIZER module has rules that do the following:

- Tell the pressure mat object to activate whenever an object is on the pressure mat.
- Tell the pressure mat object to deactivate when there is not an object on the pressure mat.
- If the pressure mat has been told to deactivate and we don't currently have a goal to deactivate the pressure mat, then assert one.

The pressure mat ACTION module has rules that do the following:

- Satisfy the goal to activate the pressure mat by changing the pressure mat state in the world such that it is active.
- Satisfy the activate pressure mat goal when the pressure mat is already active by doing nothing.
- Satisfy the goal to deactivate the pressure mat by changing the pressure mat state in the world such that it is not active.

An effector, such as a video camera, would have similar modules except that the HYPOTHESIZER module's rules would request activation/ deactivation upon request rather than an environmental stimulus. Then, the HADLIR agent description is combined with appropriate Environment Descriptors and transformed into an agent definition file specific to our multi-agent society software package using XSLT rules. The agent definition file contains the properties necessary to instantiate the agent into the multi-agent society software package. The XSLT rules include two elements: the specification of the software sensor(s)/effector(s) associated with the HADLIR agent description and the Environment Descriptors. For the pressure mat, the software sensor (the hardware agent's sensor) detects location and pressure. The Environment Descriptors express how the agent is to be presented in the (virtual or physical) intelligent room environment. For example, in the virtual environment, the Environment Descriptors would prescribe the appropriate 3D model of a device in a VRML file. In the physical environment, the Environment Descriptors would prescribe the required device drivers and the communication port on the microelectronics board.

Virtual Environment Simulation

In the virtual environment simulation, the XSLT transformations additionally produce an object

definition for the pressure mat in Active Worlds. The object definition is read by Active Worlds to instantiate the pressure mat as a 3D object with avatar capabilities in Active Worlds. Once instantiated into Active Worlds, the designer can move and modify the geometric properties of the 3D object as in a standard 3D modeling package. The hardware agent running in the agent environment communicates with the pressure mat in Active Worlds via a Communication Bridge. The Communication Bridge is essentially a "hook" into the event notification engine in Active Worlds. The hardware agent's software-based sensors, written in Java, pass and receive messages from the Communication Bridge as a Java Bean to allow the hardware agent to sense and interact with the Active Worlds virtual environment. Active Worlds sends events as messages to the hardware agent through the Communication Bridge; the Communication Bridge parses these messages for relevant messages, that is, messages of interest to the software sensors. In response, the agent sends actions through the Communication Bridge to Active Worlds. Unfortunately, the physics of pressure is not defined in the Active Worlds. Consequently proximity was used to model pressure. Simulations of other hardware devices in the virtual environment would require the use of different software-based sensors and/or effectors. We are currently working towards the development of a generic class of sensors and effectors that would be suitable for a range of hardware devices.

For purposes of illustration, a wall object was used as a 3D representation of a pressure mat. The activation of the mat is visualized in the virtual environment through a color change. As illustrated in Figure 8, when an avatar (virtual representation of a person) walks towards the wall object representing the pressure mat, and the avatar is inside the proximity of the object, the activation of the pressure mat is signaled by a color change.

Implementation in a Physical Environment

The primary difference between the virtual environment simulation and the physical environment is the requirement of additional hardware and software to process the electronic data signals from the pressure mat. In other words, the hardware agent continues to operate within the multi-agent software package as before with the same HADLIR agent description. However, the Communication Bridge is configured to send location and object presence messages to and from the physical environment. For the physical environment, we instrumented Teleo [www.makingthings.com] microelectronic control modules and Max [www.cycling74.com] to process data signals from the pressure mats as schematically illustrated in Figure 9.

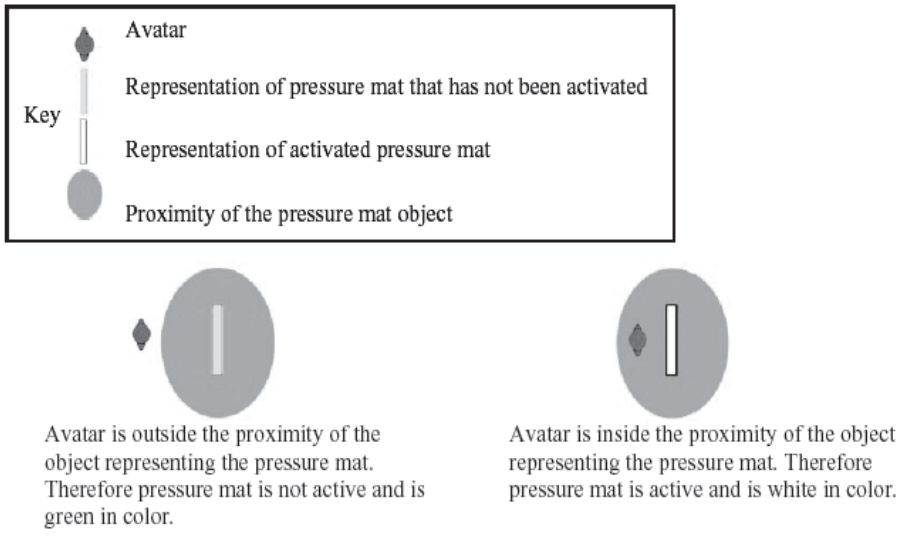

Figure 8. *Simulation of the pressure mat hardware agent in the KCDCC AW Agent Package*

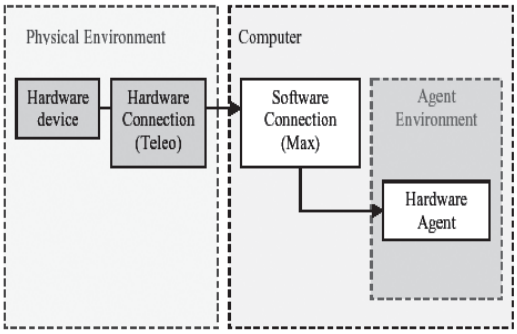

Figure 9. *Version that uses a physical environment sensor to produce an action*

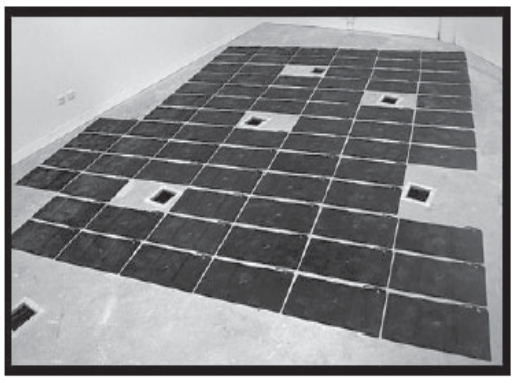

Figure 10. *Pressure Mats in the Sentient*

The HADLIR representation was implemented for a physical intelligent room called The Sentient. The Sentient is an intelligent room project being developed in the Faculty of Architecture at the University of Sydney. The Sentient provides a physical test bed for the HADLIR and the HADLIR-based CAD tool. Underneath The Sentient lies a series of pressure mats as shown in Figure 10. Each of the pressure mats is connected to a Teleo module (Figure 11); several Teleo modules are connected to create a society of pressure mat hardware agents.

Figure 12 illustrates the data flows for the Max application developed to send a binary number from the pressure mat to the hardware agent to indicate the state of the pressure mat. When an object is placed on a pressure mat, the signal is passed through the Teleo module to the hardware agent through the following data flow. The Teleo module receives a digital signal from the physical pressure mat. If the binary value received is 1, then there is an object on the pressure mat; if it is 0, then there is no object on the pressure mat.

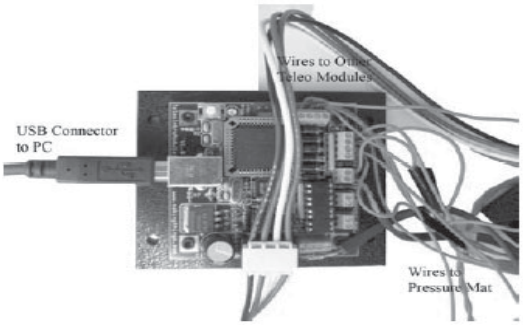

Figure 11. *Teleo Module*

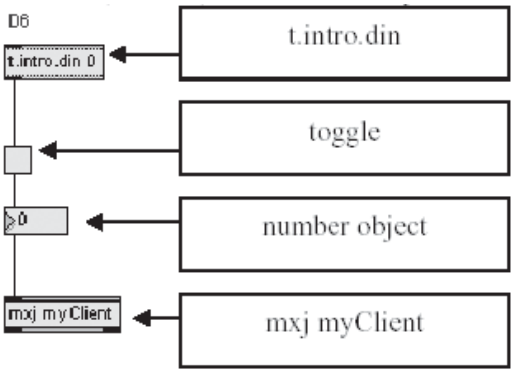

Figure 12. *Max application for a pressure mat at loc. D6*

The t.intro.din retrieves the data signal from the pressure mat and sends it as a message into the toggle. The toggle sends the binary number as a message to the number object, which displays the data for debugging purposes. The data is then sent from the number object to the mxj myClient object.

The mxj myClient object is a Java-based network socket that sends the binary value to the Connection Bridge. The Connection Bridge is simply a network socket which listens for messages from Max, sent from myClient, and then passes the message as a Java Bean to the pressure mat's hardware agent. Recall that in the virtual environment version, the Communication Bridge handled message passing between Active Worlds and the hardware agent's software sensors. The pressure mat's software sensor is the same as before.

Results of Implementation

Figure 14 illustrates the output from the CAD tool during operation in a virtual and a physical

intelligent room. First, the CAD tool loads the HADLIR agent description (pressuremat.properties and pressuremat.xml). Then, the pressure mat hardware agent instantiates itself into the multi-agent society called "Office."

Figure 13 illustrates what is seen in the simulation in Active Worlds. In the images of Figure 13, the perspective shown is from a first-person (avatar) view. Thus, the left hand image is "from a distance" whereas the right hand image is "up close." The hardware agent performs the color changes based on the sensor data it receives. When the avatar is distant, the pressure mat is inactive (color green). When an avatar moves close to the pressure mat, the pressure mat hardware agent, based on the rule base, activates. For demonstration purposes, we visualized the activation by having the hardware agent send the message "action create color white" to Active Worlds. If the avatar moves out of the proximity then the color of the wall object representing the pressure mat is changed back to its original color.

In the physical intelligent room, an object is placed on the pressure mat. At that instance, the message is passed through the Teleo module to the hardware agent. As it is received and parsed, the hardware agent displays the message "input value 1" in the output screen. The rule triggers and the message "goal ON asserted" is displayed. When the object is removed, the sensor displays the message "input value 0" and the rule triggers; the resulting message is "goal OFF asserted."

Conclusions

The HADLIR representation is founded around the concept of modeling hardware in an intelligent room as agents having sensor and/or effector modalities with rules and goals. This high level representation of the hardware in intelligent rooms captures the attributes and relationships that determine hardware behavior and actions while the Jess rule-base forms a basis for the degree of intelligence that can be embedded into the hardware. The HADLIR representation of intelligent room hardware as a hardware agent, comprised of its description as an agent, a rule base, and Java-based software sensors/effectors, is suitable for prototyping intelligent rooms in virtual environments and implementation in a physical intelligent room. The physical implementation is the same as the virtual environment simulation apart from the fact that the hardware agent accesses sensor data from a physical hardware device. We were able to demonstrate that a CAD tool could allow a designer to prototype and simulate an intelligent room in a virtual environment and then implement the room using the same description. The demonstration offers evidence that the basic idea of describing hardware as agents in intelligent rooms is a workable model for building a CAD tool for intelligent rooms.

At present, the HADLIR representation and authoring tool caters for the description of devices without dependencies. An example of a "device"

Figure 13. *Images of the pressure mat simulation in Active Worlds. The pressure mat hardware agent changes the color of the wall object representing the pressure mat*

with dependencies would be augmented reality. The hardware for augmented reality includes a printed pattern, a digital video camera, and a raster display device. We are currently experimenting with the relative merits of including these dependencies into the HADLIR representation itself or into the rule-base. Others have defined these dependencies in the ontology itself. For example, the Semantic Space ontology (Wang, Dong, Chin, Hettiarachchi, & Zhang, 2004) includes friendOf and colleagueOf in describing relationships between people in a context. This idea could be extended into the HADLIR representation. On the other hand, these relationships may also be included in the rule set as goals to allow each hardware agent to partner with any number of devices which could satisfy the goal, rather than a pre-defined partner. The ontology could encapsulate functional dependencies which could be satisfied by a class of devices – for example, a digital video camera, rather than a light, could satisfy the functional need for illumination by projecting a white image. We believe these issues warrant closer scrutiny.

The creation of intelligent environments presents a new level of complexity in architectural design because, in addition to the description of the state, the architect-designer must also describe the function and behavior of the computational objects that enact the environment's intelligent behavior. Thus, a new type of CAD tool is necessary in which the specification of *how* the intelligent environment operates is incorporated into its geometric description. The HADLIR representation presented in this paper describes a way to incorporate "machine instructions" into the geometric description and to use the description for both virtual prototyping and physical implementation.

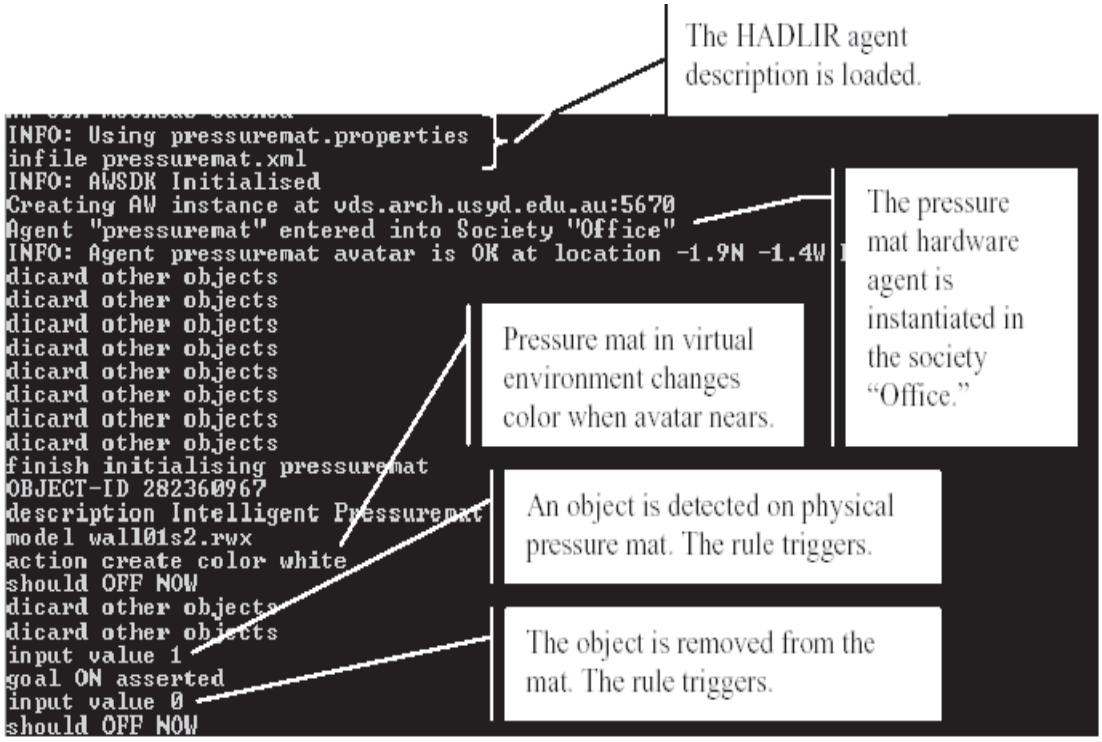

Figure 14. *Output from HADLIR implementation of a pressure mat*

References

Bobick, A. F., Intille, S. S., Davis, J. W., Baird, F., Pinhanez, C. S., Campbell, L. W., Ivanov, Y. A., Schütte, A., & Wilson, A. (2000). Perceptual user interfaces: the KidsRoom. *Communications of the ACM, 43*(3), 60-61.

Chen, H., Finin, T., & Joshi, A. (2004). An ontology for context-aware pervasive computing environments. *The Knowledge Engineering Review, 18*(3), 197-207.

Coen, M. H. (1997). Building Brains for Rooms: Designing Distributed Software Agents. *The Ninth Conference on Innovative Applications of Artificial Intelligence (IAAI-97)*, (pp. 971-977), Providence, Rhode Island: AAAI Press.

Coen, M. H. (1998). Design principles for intelligent environments. *The Tenth Conference on Innovative Applications of Artificial Intelligence (IAAI-98)*, (pp. 547-554), Madison, Wisconsin: AAAI Press.

Friedman-Hill, E. (2003). *Jess in action: rule-based systems in Java*. Greenwich, Connecticut: Manning Publications Company.

Gage, S. (1998). Intelligent interactive architecture. *Architectural Design, 68*(11-12), 80-85.

Gellersen, H., Kortuem, G., Schmidt, A., & Beigl, M. (2004, July-September 2004). Physical Prototyping with Smart-Its. *IEEE Pervasive Computing, 3,* 74-82.

Gruber, T. R. (1993). A translation approach to portable ontologies. *Knowledge Acquisition, 5*(2), 199-220.

Hirsh, H., Coen, M. H., Mozer, M. C., Hasha, R., & Flanagan, J. L. (1999). Room service, AI-style. *IEEE intelligent systems, 14*(2), 8-19.

Huhns, M. N., & Seshadri, S. (2000). Sensors + Agents + Networks = Aware Agents. *IEEE Internet Computing, 4*(3), 84-86.

Hunt, G. (1998). Architecture in the 'cybernetic age'. *Architectural Design, 68*(11-12), 53-55.

Intille, S. S. (2002). Designing a Home of the Future. *IEEE Pervasive Computing, 1*(2), 80-86.

Kreuger, T. (1996). Like a second skin, living machines. *Architectural Design, 66*(9-10), 29-32.

Kreuger, T. (1998). Autonomous Architecture. *Digital Creativity, 9*(1), 43-47.

Pask, G. (1969). The architectural relevance of cybernetics. *Architectural Design, 7*(6), 494-496.

Ranganathan, A., McGrath, R. E., Campbell, R. H., & Mickunas, M. D. (2004). Use of ontologies in a pervasive computing environment. *The Knowledge Engineering Review, 18*(3), 209-220.

Smith, G. (2003). KCDCC AW Package. Sydney: Key Centre of Design Computing and Cognition.

Smith, G., & Gero, J. (2004). Describing situated design agents. In J. S. Gero (Ed.), *Design Computing and Cognition '04* (pp. 439-457). Dordrecht: Kluwer Academic Publishers.

Wang, X., Dong, J. S., Chin, C., Hettiarachchi, S. R., & Zhang, D. (2004, July-September 2004). Semantic Space: An Infrastructure for Smart Spaces. *IEEE Pervasive Computing, 3,* 32-39.