Presented at the 6h International Conference on Information Systems Analysis and Synthesis (ISAS 2000), Orlando, Florida, July 2000

A Distributed Intelligent Building Agent Language (DIBAL)

Filiz Cayci, Victor Callaghan, Graham Clarke

Department of Computer Science,
University of Essex
Wivenhoe Park, Colchester CO4 3SQ
United Kingdom
Email: fcayci@essex.ac.uk

ABSTRACT

From the multi-agent system perspective Intelligent Buildings (IB) are a new research area and this domain poses several interesting challenges. Our IB model comprises a building containing room based embedded-agents which control environmental variables and devices within rooms that communicate with each other via a network. Inter-agent communication is central to such multi-agent systems. In this paper we explain why existing agent communication languages (ACLs) are unsuitable for IB embedded-agents and present a specification for a hierarchical Distributed Intelligent Building Agent Langauge (DIBAL) that overcomes the problems in applying ACLs to IB based embedded-agents. The paper begins by reviewing existing Agent Communication Languages (ACL) and discussing their unsuitability for IB applications before preenting our IB embedded-agent communition language, DIBAL. We then illustrate the ways in which DIBAL would facilitate the functionality we require by looking at a few IB scenarios in some detail.

1. INTRODUCTION

An Intelligent Building (IB) has been described as one "*that utilises computer technology to autonomously govern the building environment so as to optimise user comfort, energy-consumption, safety and monitoring-functions*" [2].

The Intelligent Building domain poses several interesting challenges to computer science and AI in particular.

These challenges arise because of (a) the existence of the many different tasks and interactions involved to achieve an IB's goals, (b) a requirement that actions be taken in real-time and (c) because IB's comprise large numbers of connected and interacting components. It is intrinsically difficult to design aspects of individual embedded-agents that, when connected together, will cause the desired over-all system functionality to be implemented successfully which is why the agents have to be adaptive and intelligent.

The way the system should behave as a whole depends upon the way the agents communicate and interact, and the information they exchange with one another. Because of this the communication language has to be supportable by the agents.

The embedded agents developed at Essex use the behaviour-based paradigm first proposed by Brooks [1]. In this approach the interaction of behaviours provides the required pseudo reasoning and planning whilst a deliberative layer provides learning [2, 3, 13].

These embedded agents based on a microcomputer or embedded Internet chip built into a device have a small computational footprint with potentially high levels of input and output. The form of collaboration being addressed here allows different embedded agents, possibly employing different strategies, to communicate and produce the best functionality to the benefit of the buildings stakeholders (owners and occupiers).

The architectural paradigm we have developed, in order for our multi-embedded-agent system to function, is a high-level building and occupant related information exchange [3]. There is an underlying network infrastructure to support the technical needs, such as sending and receiving packets and dealing with changing agent configuration that is managed via an IP network.

In a typical agent based IB systems, control goals are split into four related functions: *economy*, for energy efficiency, *emergency* for dealing with unexpected situations, *safety* to ensure the environmental variables remain at safe levels and *comfort*, which are particular preferences relating to an individual occupant.

The communication load will also entail messaging to and from sensors, between room agents, to and from personal agents, and with agents from the outside world. This information will be in a wide variety of forms, for example room status and occupant information, readings from a variety of sensors, information about the times of particular activities etc.

In section two we are going to discuss research relating to standard ACLs like KQML or FIPA and frameworks that have been developed with a similar functionality. We will look at these critically in the light of our needs for a computationally light solution because of the constraints on IB agents. We will argue that the type of agent used in IB will not need, or be able to support, a complex communication language like KQML or FIPA.

Section three will introduce and specify our proposed communication model and language and the hierarchical message passing technique between agents in our IB model. We propose a JAVA based Distributed Intelligent Building Agent Language (DIBAL), which will be flexible and efficient and does not produce too much of an overhead computationally. We specify a generic, compact and hierarchical data message packets structure with two main categories of message primitives: an agent must be able to both, send and request information from others.

In order to show that the DIBAL specification is adequate to the IB domain, in section four we will look in some detail at characteristic IB scenarios and show how IB goals could be achieved using the language specified.

2. CRITICAL REVIEW OF EXISTING ACLs

The Knowledge Query and Manipulation Language (KQML) is a high-level agent communication language, based on speech act theory, which provides twelve reserved performatives, falling into seven categories. KQML, conceived in the early 90's, gradually developed the concept of an ACL. According to Finin [5,6,12], KQML is independent of the transport mechanism, independent of the content language and independent of the ontology assumed by the content. It is divided into three layers: the content, the message and the communication layer. The content layer bears the actual content of the message, the communication level encodes a set of features to the message which describe the lower level communication parameters and the message layer that is used to encode any message that one application would like to transmit to another. KQML has a fixed context partly because the language has too many constraints and partly because it is inflexible [4]. For example, by imposing the pragmatic requirement to be co-operative, which may not be acceptable in certain contexts because of violating a key prerequisite.

Foundation for Intelligent Physical Agents (FIPA) is also a high-level ACL, which, like KQML, is based on speech act theory. Messages are actions or communicative acts, as they are intended to perform some actions by virtue of being sent.

KQML and FIPA are almost identical with respect to their basic concepts and the principles they observe and differ primarily in the details of their semantic frameworks [5]. Their syntax is identical except for the different names for some reserved primitives.

Labrou [6] defines agent communication languages (ACL) as a collection of message types each with a reserved meaning. An ACL is not concerned with the physical exchange, over the network, of an expression in some language, but rather with stating an attitude about the content of this exchange, like making a promise or a commitment to perform a future action.

Because of the small computational footprint and the structure of our embedded agents, such highly developed, heavy weight and functionally complex ACLs like KQML and FIPA are not suited to our IB domain. For example the number of primitives and the complex layered structure of these ACLs could not be easily accommodated within the constraints of our agents limited resources.

Beside this, FIPA does not support what is a very important component for IB needs, that is, the problem of acknowledging receipt of messages. This is a conceptual problem related to agent interaction. An agent waiting for an answer does not know whether the agent it is communicating with is busy (and will reply later) or does not want to answer.

This has a negative impact on protocols which imply synchronisation between the involved agents and whose termination is subordinate to the reception of all the communication acts. To overcome this problem, FIPA allows time specifications to be placed in a message to constraint the reply; but this in turn introduces new problems when the peer agent needs time to compute the answer [7]. In our case, it is important that a sender (room) agent knows whether a message has been executed or received and understood by the receiver agent.

Standard ACL languages do not have convenient and flexible mechanisms for registering new agents presence on the network since this has not been the focus of the standardisation efforts [5].

As well as fully-fledged ACLs like KQML and FIPA there are a number of frameworks, which aim to achieve the same functionality. We have looked at these also with a view to using them if they suit our IB specific problem.

There has also been the development of agent communication languages for the domain of mobile robots that we will consider first.

Wang and Premvuti [8] argue that, the general principles of the Distributed Robotic Systems model (mobile agents) for instance does not allow for any co-ordination mechanism such as a centralised CPU, a centralised and shared memory, or a synchronised clock. Moreover, the system only consists of multiple, autonomous mobile robots. They also state that, there is no centralised ‘ground support’ such as a communication server. Autonomous mobile agents in distributed robotic systems communicate through either localised broadcasting (point-to-point communication) or other types of ground support. The communication in these systems takes place mostly through radio communication network and is mostly sensor based. The lack of a co-ordination mechanism makes this unsuitable for IB applications.

Over the past few years, a multitude of applications and systems have appeared that are built around agent communication languages. Either they are applications, i.e. multi-agent systems, that use an ACL for inter-agent communication, or APIs that facilitate the incorporation of ACL capabilities into an application.

The Java-based Agent Framework for Multi-Agent Systems (JAFMAS), developed at the University of Cincinnati, provides a generic methodology for developing speech-act (e.g. KQML) based multi-agent systems (MAS), an agent architecture, and a set of classes to support implementing these agents in Java [14]. JAFMAS also supports direct (point-to-point) communication as well as subject-based broadcast communications.

Jackal, developed at University of Maryland Baltimore Country, is another Java package that allows applications written in Java to communicate via an ACL (KQML is currently the ACL of choice). Jackal strongly emphasizes conversations between agents and provides a flexible framework for designing agents around conversations [9].

Both Jackal and JAFMAS are extensions to JAVA. They both aim to implement the functionality of an ACL like KQML. It seems likely that this would require even more of the limited resources of an embedded agent than KQML and is therefore also unsuitable for our agents.

Another application, Java Agent Template, Lite (JATLite) is a package of Java programs, developed at Stanford, that allows users to quickly create communication agents. Agents are running as applets actuated from a browser and because of that all agents register with an Agent Message Router facilitator (AMR) that handles message delivery.

JATLite is centralized, and from this it follows, that it does not allow a peer-to-peer communication between the agents [10]. Intelligent Building based embedded agents have to communicate in a distributed environment. In many cases occupant related information have to be sent directly to an appropriate agent, instead of broadcasting to all agents. This limitation would make it unsuitable for IB related applications.

We have made a few different enquires to the people who produced the systems mentioned above concerning the size of the system and the possibilities of reducing this in order to fit our embedded agents needs, but unfortunately we have yet to received any positive response.

Martin [11], describes the Open Agent Architecture (OAA) Interagent Communication Language (ICL), for instance, as the interface, communication and task co-ordination language. One of the fundamental elements of ICL is the "event". The ICL includes a layer of conversational protocol, similar in spirit to that provided by KQML, and a content layer, analogous to that provided by Knowledge Interchange Format (KIF). The conversational layer of ICL is defined by the event types. The content layer consists of the specific goals, triggers, and data elements.

This approach offers greater expressiveness than an approach based solely on a fixed selection of speech acts, such as embodied in KQML [11]. We assume, that this expressiveness will in all probability make the ACL larger and therefore not fulfil our special interest in keeping the ACL's size as limited as possible.

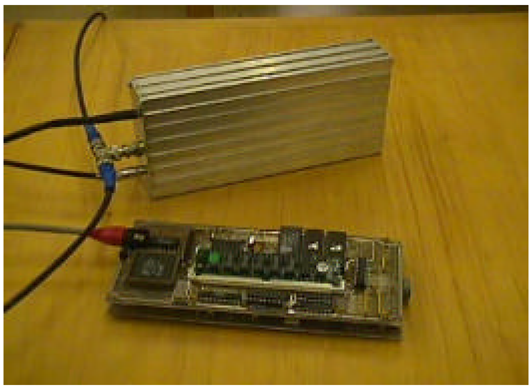
Each implementation introduces its own variety of supporting agents and services. Agreement is needed on the assumptions of these services so that such sevices can be provided as a standard suite of tools.

ACLs also present problems when their semantics are considered: (1) they are tied to a specific agent theory that might not be applicable to all agents that want to use the ACL, and (2) they introduce complex formalisms that have no bearing on the implementation of agent systems [12].

In the next section we will introduce our proposed agent communication language – DIBAL - and specify the language primitives and the structure of the messages needed for our Intelligent Building domain. Our communication language allows us to reduce the amount of infrastructure by providing for both communication and control with a much lighter-weight system.

3. THE DIBAL MODEL

Our Intelligent Building model consists of different rooms; each controlled with an embedded agent and equipped with different sensors for temperature, light, door and window status and so on. Our Intelligent Building model uses a combination of Ethernet (agent-to-agent) and LONWorks (agent-to-sensors) networking technology.



Picture 1: Essex Embedded Agent Prototype

The embedded agents [Picture 1], e.g. microcomputer or embedded Internet chip built on a device[1], have limited computational resources - processor power and memory. To be effective agents have to handle all of the following processes - learning occupant behaviours, controlling environment variables, monitoring sensors and other devices and

[1] http://www.ibutton.com/TINI/index.html

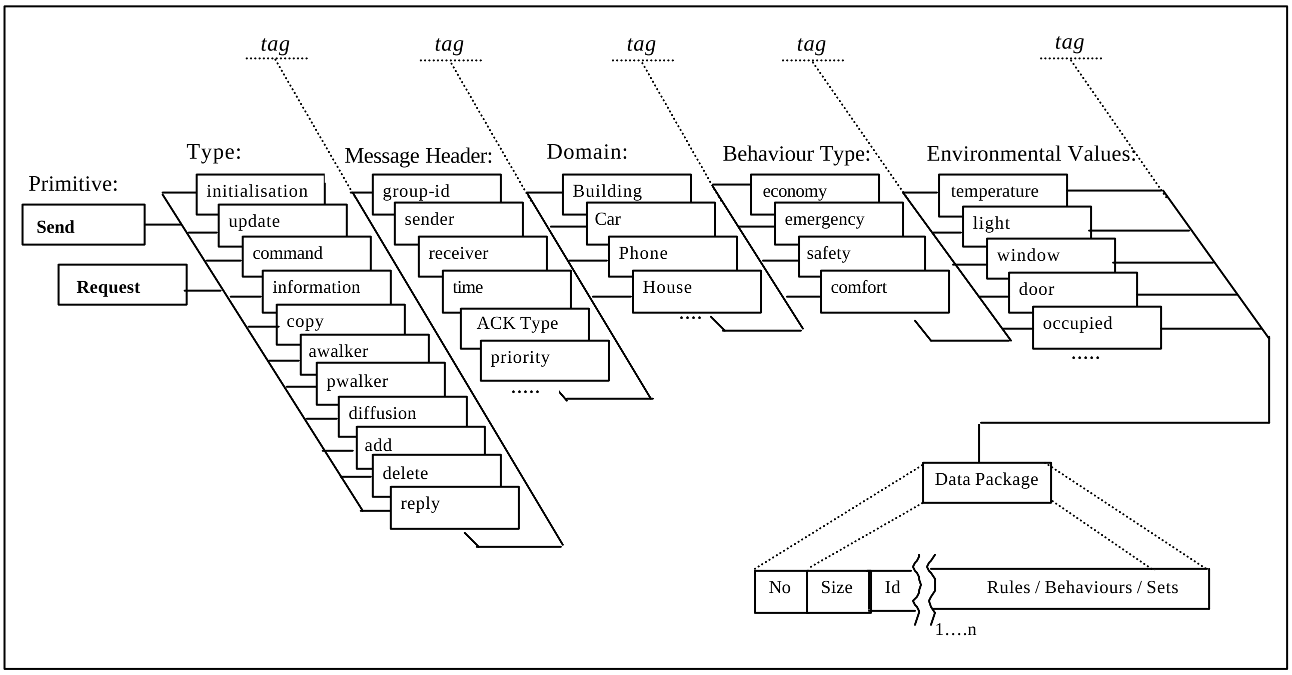


Figure 1: Message Format

communicating with other agents whilst operating within these resource limitations.

This is one of the overriding reasons for making the communication language as compact and flexible as possible. For this reason we are using a hierarchical and tagged form of messaging, and have a limited number of main primitives [Figure 1].

Each agent will have the two main primitives at its disposal: send and request. All messages will contain a message type, which describes the content of the message. Any embedded agent will know, after receiving a message, what kind of data is going to be sent and what kind of action is expected. As the figure shows there are eleven message types. Each of these types will help to enable a specific scenario characteristic of Intelligent Buildings. An agent receiving a "command" message for instance will automatically know that the data package contains actions, which have to be executed immediately. On the other hand an agent receiving an "information" message can decide whether this information package is useful for it or not.

The next "tag" contains lower-level information such as the sender and receiver of the message, its priority, time and whether an acknowledgement, such as 'message successfully performed', or, 'error occurred', is required from the receiver.

Our proposed agent communication language is not concerned exclusively with the domain of Intelligent Building since we can envisage the development of mobile personal agents and the spread of embedded agents into all areas of life. This will mean that a distributed agent language will be required for a whole range of environments including the home, the car, other forms of transport and eventually space vehicles and extraterrestrial habitats.

As mentioned before the domain of Intelligent Buildings is generally composed of cooperating control functions. In our case we are utilising behaviour-based agents composed of *economy*, *emergency*, *safety* and *comfort* behaviours (effectively, sets of rules) [2]. These are individual to the occupant. In our message structure each of these behaviour types relates to its own environmental values such as temperature, light or window status etc.

The transmitted message packages, excluding the data package, have the same fixed message size. The variable data package associated with environmental variables starts with a flag specifying the number of records and their total size. N.B. it is here that the identity of the user, for whom this behaviour is important, is stored. Even before "editing" the high-level data content, the agent has to know if its existing limited resources are enough to process the message data.

Each behaviour – rule is also specified with its own size, to allow the agent to execute just the first two rules for instance (Figure 1: n=2).

send (initialisation
(group-flag, sender, receiver, time of msg,
priority of msg., ACK, Mode,
(house, comfort, temp,
[Data-type, Number of Items, Total Size, person- id,
(RuleId
(size/length of field, usefulness, history(10),rule(s)),
RuleId
(size/length of field, usefulness, history(10),rule(s)))])))

The tagged and hierarchical structure of our message formats allows the embedded agents to communicate flexibly and quickly in real time environments, which is appropriate to their limited resources. An example of a transmitted message is shown above.

An agent receiving an initialisation message enters new behaviours or rules, in this case, personal temperature preferences, in its list of current behaviours. This is useful in cases where a new person moves into a new room and the room agent needs to set up appropriate comfort behaviours for its new occupant.

The next section looks at such scenarios in greater detail and demonstrates the appropriateness of the proposed ACL primitives and message types for different characteristic Intelligent Building scenarios.

4. THE DIBAL SCENARIOS

Different scenarios for the operation of an Intelligent Building can best be understood using a 3-tier-model.

The actual scenario, as some completed function of a room agent for instance, is made up of a series of messages between itself and other agents, each of which is built out of the primitive message types specified in the DIBAL.

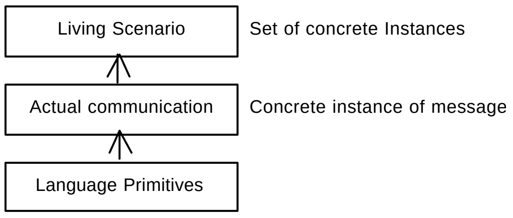


Figure 2: 3 tier-model for Scenarios

This process runs in parallel with the operation of the reasoning and learning capabilities of the agent, which may, or may not themselves require messages to and from other agents to be completed successfully.

In all the scenarios that follow we assume that each person in the building can be uniquely identified by the system.

Buildings contain all sorts of different activities such as, people entering and leaving rooms and/or the building, gathering together for meetings or social intercourse in rooms that might be general purpose or specifically tailored to particular uses. Similarly at any point in time rooms that were occupied might become empty and remain empty or emergency situations of one sort or another might develop like the outbreak of fire or an attempted burglary.

Each person in the building will have his or her own preferences regarding environmental variable that it is the agents' task to learn and make part of its repertoire of behaviours.

These behaviours (sets of rules related to environmental variables, devices and people) are split into economy, emergency, safety and comfort types, and are stored in a form that is consistent with the communication languages data packages.

The building consists of a network of room-based agents covering the entire building. The location and function of the room for each agent can be used to group agents into useful categories. For instance all corridor agents might be part of a group or all agents on one floor of a building etc. There is no reason why an agent might not be part of several different groups.

One significant scenario involves the case of an attempted break-in.

Imagine that someone tries to enter a room on the first floor by breaking a window. The agent will be able to work out that there is movement from an unidentified person who has entered the room without opening the door. The room agent then has to make sure that the doors of rooms next to it, or all doors on the same floor, are locked immediately. To realise this, it should be possible to send each appropriate agent (members of a particular group for instance) a "command" message, as follows:

send** (**command
(group-flag, sender, receiver, time of msg, priority of msg., ACK, Mode,
(house, emergency, door, "lock door")))

If the group-flag is set, than the receiver is going to be a group of agents, like all agents in the same floor or office agents etc otherwise the receiver will be one specific agent.

We assume that empty rooms in the buildings are going to have default temperature setting to ensure that the building functions economically. This temperature will possibly be dependent on the time of year - the value in summer times is likely to be different than the one in winter. To update and reconfigure an existing standard value for a group or all agents it is possible to send the following message:

send (update
(group-flag, sender, receiver, time of msg, priority of msg., type, ACK, Mode,
(house, economy, temp,
[Data-type, Number of Items, Total Size,
(RuleId
(size/length of field, usefulness, history(10),rule(s)),)])))

Next, we would like to show more complex scenarios including several messages. Suppose in a normal everyday situation, a person enters a building. The entrance hall or door agent recognises this person, and sends a message throughout the building, to inform other agents that a specific person has entered the building.

send (information
(group-flag, sender, receiver, time of msg, priority of msg., (ACK), Mode,
(house, economy, occupied, "person X entered building")))

If this person is already known, this message could be a direct message to his room agent, if not, it will be a broadcast. Let us imagine that this person is a regular occupant of a room – his office say, but before he goes to his office he goes to a laboratory. The laboratory agent will therefore also request this persons preferred comfort behaviours, though this time from the agent of the space adjacent to the laboratory:

request (copy
(group-flag, sender, receiver, time of msg, priority of msg.,
Reply_With, Mode,
(house, comfort, temp, person-id(RuleId)))

This sort of scenario will be repeated often as a person moves about the building. This walking through different rooms allows us 'behaviour migration' for each person throughout a building.

Suppose however that the person who entered is new to the building. Naturally the standard economy, emergency and safety behaviours will be available in advance. To save time and agent resources on learning from scratch the agent can for instance request a default set of comfort behaviours.

request (diffusion
(group-flag, sender, receiver, agent-type, time of msg,
priority of msg., Reply_With, Mode,
(house, comfort, temp)))

Because of the "group-flag" mentioned earlier, the "agent-type" specifies the agent in more detail for instance agents located in rooms, elevators, kitchen etc.

Each of our message types is related to important activities occurring on a regular basis in Intelligent Buildings.

5. CONCLUSION AND FUTURE WORK

Agent communication languages have been used for about ten years in proprietary multi-agent systems, and KQML was among the first such ACL to be developed and used. KQML has also played a large and important rule in defining what an Agent Communication Language is and what the issues are when it comes to integrating communication into multi- agent system.

In this paper, we have outlined the reasons why an agent communication language for embedded agents in the domain of Intelligent Buildings is required. Within the space limitations of this paper we have specified the DIBAL language and described the operational model. DIBAL is a compact, flexible and hierarchical way of messaging with consideration of the limited computational resources of embedded-agents.

The next step in this work will be to implement the DIBAL model in Java and evaluate it using a representative Intelligent Building environment.

REFERENCES

[1] Brooks R.A, Intelligent Room Project, MIT Artificial Intelligence Lab, Proceedings of the Second International Cognitive Technology Conference (CT'97), Aizu, Japan, August 1997

[2] Sharples S, Callaghan V, Clarke, G "A Multi-Agent Architecture For Intelligent Building Sensing and Control", Int'l Sensor Review Journal, Vol 19. No. 2. May 1999

[3] Callaghan V, Clarke G, Pounds-Cornish A, Sharples S, "Buildings As Intelligent Autonomous Systems: A Model for Integrating Personal & Building Agents", The 6th International Conference on Intelligent Autonomous Systems, Venice, Italy; July 25 - 27

[4] Singh M.P, North Carolina State University, "Agent Communication Languages: Rethinking the Principles", December 1998

[5] Labrou Y, Finin T, Peng Y, "The current Landscape of Agent Communication Languages", 1999

[6] Labrou Y, Finin T, Peng Y, "The Interoperability Problem: Bringing together Mobile Agents and Agent Communication Languages, 1999

[7] Mamdani E, Charlon P, Cattoni R, Potrich A, "Evaluating the FIPA Standards and Its Role in Achieving Cooperation in Multi-agent Systems", Submitted to Multiagent Systems, Internet and Applications (HICSS-33 Minitrack), January 2000

[8] Wang J, Premvuti S, "Resource sharing in distributed robotic systems based on a wireless medium access protocol", Robotics and Autonomous System 19 (1996), 33-56

[9] Cost R.S, Finin T, Labrou Y, Luan X., Peng Y, Soboroff I, Mayfield J and Boughannam A, "Jackal: A Java Based Tool for Agent Development", " Working Notes of the Workshop on Tools for Developing Agents (AAAI Technical Report), AAAI 98, 1998

[10] Jeon H, Petrie C, Cutkosky M.R, "JATLite: A Java Agent Infrastructure with Message Routing", University of Stanford, IEEE Internet Computing, March/April 2000

[11] Martin D.L, Cheyer A.J., Moran D.B., "The Open Agent Architecture: A Framework for Building Distributed Software Systems", January-March, 1999

[12] Labrou Y and Finin T, "Towards a standard for an Agent, Communication Language", 1997

[13] Callaghan V, Clarke, G., Colley, M., Hagras, H. "A Soft-Computing DAI Architecture for Intelligent Buildings", Journal of Studies in Fuzziness and Soft Computing on Soft Computing Agents, Physica-Verlag-Springer, July, 2000

[14] Galan A, "JAFMAS: A Java-based Agent Framework for Multiagent Systems Development and Implementation", Deepika Chauhan, ECECS Department, University of Cincinnati, 1997