Decentralized Autonomous Organization of the Intelligent Home According to the Principle of the Immune System

Werner Dilger
Computer Science Department
Chemnitz University of Technology
D-09107 Chemnitz, Germany
email: dilger@informatik.tu-chemnitz.de

Abstract

The basic principles of the Intelligent Home technology are presented and it is described how it can be modeled as a multi-agent system. Because of the complexity of the system it is argued that it should be generated by an evolutionary process and maintained according to the principles of the immune system.

1. Introduction

The Intelligent Home (IH) is a new technology that is on the way to change the way how to live in a home drastically. It is one of the areas where the application of information technology step by step replaces conventional technology and opens the possibility for realizing new functions that make living easier and more comfortable. However, it also is more complex than conventional technology such that it becomes more and more difficult to realize IHs and to maintain them.

Since a few years the principle of evolution has been applied to the development and in particular to the optimization of technical system. A well known form of algorithms of this kind are the genetic algorithms. However the generation and optimization is just one side. For systems that are used in variable environments it is required that they can adapt themselves to these environments. For this purpose the principles of immunity-based systems can be applied. In this paper it is shown how both steps can be applied for the generation and maintenance of the IH.

Section 2 presents the principles of the IH and how it can be modeled as a multi-agent system. In section 3 the evolution process for generating the IH is described in detail. Some of the operations of the evolution process can also be used for the maintenance of the IH in a changing environment, and section 4 shows how this can be viewed in analogy to the immune system. Section 5 gives final remarks.

2. Intelligent Home technology

Basic principles

The principle of Intelligent Home technology is to consider the IH as an information processing system. The units of this system are the devices that are needed to realize the functions that are expected from the IH. There are two classes of devices, namely sensors and actuators. Sensors are e.g. motion detectors, noise sensors, brightness sensors, thermometers and switches. Actuators are e.g. lights, blinds, buzzers, heaters, and alarms.

In conventional technology the devices of a home are operated mechanically or by electrical circuits. In IH technology this is done by the means of information technology. That means that the desired functions like turning on or off lights, keeping a certain temperature, or opening and closing doors are realized by communication between the devices involved. For instance for keeping a certain temperature in a room, the temperature is regularly measured by a thermometer and the value is sent to a control unit. The control unit matches this value against the set value. If it records a significant difference, it produces a message to the heater to change the heat flow up or down as required.

Technology of this type is already in use. Up to now it is almost everywhere only applied to separate subsystems of buildings, such as the lighting system, the air conditioning or heating system, or the security system. However, the IH technology has the potential to realize a lot of functions that are not yet in existence. This shall be illustrated by the following scenario. It admittedly sounds futuristically but some of the functions mentioned do already exist, some are in reach, and only a few are still to be developed.

You are driving home in the evening in your car and you send a message to your home by phone saying that you will probably arrive at 6 p.m. When you arrive the garage door is opened automatically because the home recognizes you car. The temperature is adjusted as you like it, the cleaning robot has cleaned up, the radio plays

0-7803-4053-1/97/$10.00 © 1997 IEEE

the music of your favorite radio station, your electronic assistant gives you a summary of what happened this day and what should be done immediately, e.g. answering a telephone call, and the bathtub is prepared for a refreshing bath. By the phone call you may have ordered a cup of coffee but nothing to eat because you want to go out for dinner, so after your bath the coffee is served by the kitchen robot.

The scenario illustrates that the IH is more than a house with some technical installations, rather it is a *habitable stationary robot*, consisting of different parts that can to some degree act autonomously but have to cooperate to perform the requested functions. It is in fact a very complex system. Therefore it is

- hard to realize in full thinkable complexity, and
- hard to control.

For this reason, it is a good idea to think of a realization in the manner of an evolutionary process and of self-maintenance in the manner of the immune system. It seems natural to realize it as multi-agent system.

Modeling the Intelligent Home as a multi-agent system

All devices of the IH, sensors as well as actuators, are conceived as autonomously acting agents. As an agent, a device has to control its hardware and to cooperate with other agents by sending messages and interpreting incoming messages such that the whole system fulfils the task it is designed for. Many of the agents have to do rather simple jobs, therefore their behavior can be adequately described by means of rules. Also for more complex agents it is assumed that their communicative behavior can be defined by rules. This approach is often made when the aspect of cooperation between agents to achieve a certain function is stressed, for instance by [6], for an overview on other works cf. [7]. The rules that are used here refer to situations in which an agent may be. Such a situation is defined by the parameter values and by the incoming messages. Because of this special feature of the rules they are called *situative rules*. Thus the essential ingredients for defining an agent of the IH are

- a number of parameters together with their value domains,
- a list of situative rules.

Most value domains are discrete sets, some are even Boolean. Value domains that originally are continuous such as the real numbers are discretizised. The agent can change the parameter values using the actions of the rules. For each parameter a default value (some element of the value domain) is defined, and if there is nothing known about the actual value of the parameter, it is assumed to be the default value. Thus each parameter has a value at any time.

A *situative rule* is a pair consisting of a situation specification and an action list. By the term *situation specification* the aspect of time comes in, which is essential for agents acting in a dynamic environment. A *situation specification* is a pair

(condition, list of message specifications)

The condition may be empty which means that it is equivalent to the value true, and also the list of message specifications may be empty. The condition is a Boolean expression over the set of the agent's parameters which comprises all parameters. A triple of the form

(sender, content, receiver)

is called a *message pattern*. If all components are specified it is called a *message*. It is supposed that there is only a finite, even a small set of symbols denoting the possible contents. A triple of the form

((sender, content, receiver), t, [m, n])

is called a *message specification*. The first component is a message pattern. Usually some of the components of the message pattern are left unspecified. Obviously, at least the receiver should be specified, either by the name of the agent or a local variable denoting the agent or the name of an agent group to which the agent belongs and to which the message is broadcasted. For example, if A is an agent and event the content of a message informing the agent about a certain event then

(?sender, event, A)

is a message pattern with unspecified sender. t denotes a time and [m, n] an interval of positive integers. The following triple is a message specification.

((?sender, event, A), 15, [3, 5])

The Boolean expressions of situations are evaluated as usual. A message specification of a situation is evaluated by matching its message pattern against incoming messages. If the message pattern can be successfully matched against some of the messages and if these messages were received not later than t time steps ago and if the number of the messages is 3, 4, or 5, then the message specification yields the value true.

The meaning of a situation specification is that it matches against different actual situations. For instance in the condition part a parameter p may be specified by

$$c_1 \leq p \leq c_2$$

but in an actual situation p has a well defined value, say d. If $d \in [c_1, c_2]$, the condition is satisfied. Thus the condition covers several situations, namely those where $val(p) \in [c_1, c_2]$ (notice that discrete values are presupposed). Similarly, a message specification describes a number of different messages. Therefore a situation specification represents a whole class of situations.

An action in the action list of a situative rule is either an assignment statement by which the value of a parameter is changed or a send statement. A send statement consists of a send command and a message, i.e. a fully specified message pattern. The action list describes the reaction of the agent to any of the situations that are covered by the situation specification of the rule. As an example of an agent consider the definition of the motion detector.

name	motion detector
type	sensor
parameters	motion-impulse {0, 1} 0
variables	?b is motion detector
if motion-impulse = 1 then	send (?b, on, receiver), motion-impulse ← 0

This is a basic definition of the motion detector. It has one parameter motion-impulse with the value domain {0, 1} and the default value 0. The variable ?b is a local variable and is used to specify the rule in different ways for different instances of this agent type, cf. [1]

3. Evolution of the Intelligent Home

The evolution process starts with agents having some predefined functions. In particular the internal functions of an agent, e.g. control of the agent's hardware, is supposed to be specified in detail and will not be subject to the evolution. For instance, the basic task of a motion detector is clearly defined: It has to record motions in some area and to inform other devices about them. For this purpose it makes use of the parameter *motion-impulse* (see above). When a motion is recorded the value changes to 1, a message is produced, and the value switches back to 0. By this way a repeated motion can be detected.

In general it is supposed that some of the parameters of each device agent and some basic behavior described by situative rules are fixed. The evolution process mainly concerns the communication and cooperation between agents. This means that first of all the message specifications of the situations within situative rules and the send statements in the action list of situative rules may be modified. But the modification of rules is not restricted to these parts, because an incoming message may cause the change of a parameter value, therefore the values of the agent's parameters are influenced by other agents and thus are involved in the communication process. Also the set of parameters may be changed by addition or deletion of parameters.

It is supposed that the behavior of the agents that are considered here can be defined by situative rules as described in section 2. Therefore the behavior description of an agent can be modified in the following ways:

- Changing the situation of a rule,
- changing the action list of a rule,
- generating a new rule,
- deleting an existing rule, or
- introducing a new parameter.

The operations of the evolution process are not as arbitrarily applied as it is usually done in genetic algorithms, rather their application is controlled by the environment, which in the case of the IH is defined by the situations that are recorded. Thus, at least for the evolution process it is required that each agent maintains a history of situations.

The history of situations

The history of situations is a list of situations. A situation is defined as a pair

(value-list, message-box)

The value-list has the form

$$((param_1, val_1), ..., (param_k, val_k), t)$$

where $param_1, ..., param_k$ are the agent's parameters and $val_1, ..., val_k$ their actual values and t is the time when this value-list was recorded. It is important to note this time because it can be used to form clusters of situations as described below. The message-box is a list of the form

$$(message_1\ (t_{11}, ..., t_{1,n1}), ..., message_m\ (t_{m1}, ..., t_{m,nm}))$$

where t_{i1} is the time of the first incoming of $message_i$ and $t_{i,ni}$ that of the last incoming. The following list is a history of a particular motion detector agent named M. M receives messages from a number of other agents, namely A1, ..., A6. The empty list is denoted by ().

(((motion-impulse, 1), 5), ())
(((motion-impulse, 0), 6), ())
(((motion-impulse, 1), 11), ())
(((motion-impulse, 0), 12), ())
(((motion-impulse, 1), 14), ())
(((motion-impulse, 0), 15), ())
(((motion-impulse, 1), 16), ())

```
(((motion-impulse, 0), 17), ())
(((motion-impulse, 1), 24), ())
(((motion-impulse, 0), 25), ())
( ((motion-impulse, 0), 28),
  ((A1, event, M) (28)))
(((motion-impulse, 0), 30),
  ((A1, event, M) (28), (A3, event, M) (30)))
(((motion-impulse, 0), 31),
  ((A1, event, M) (28, 31), (A3, event, M) (30)))
(((motion-impulse, 1), 33),
  ((A1, event, M) (28, 31), (A3, event, M) (30)))
(((motion-impulse, 0), 34),
  ((A1, event, M) (28, 31), (A3, event, M) (30, 34),
  (A4, event, M) (34)))
(((motion-impulse, 1), 42),
  ((A1, event, M) (28, 31), (A3, event, M) (30, 34),
  (A4, event, M) (34)))
(((motion-impulse, 0), 43),
  ((A1, event, M) (28, 31), (A3, event, M) (30, 34),
  (A4, event, M) (34)))
```

The building of the history is event driven. The events that cause the agent to record a new situation are

- the change of a parameter value,
- the incoming of a message, or
- the deletion of a message.

The first two events are illustrated by the above example. The last event happens when a message becomes too old to be interesting, i.e. when the difference between the actual time and the last incoming time of the message becomes greater than some predefined value. In this way, the lengths of the lists of incoming times are restricted.

The history of situations can be used for different purposes. The simplest case is the deletion of a rule. If over some period of time none of the recorded situations satisfies the situation specification of the rule, then this indicates that the rule is of no use or, though it has been used before, it has become superfluous because the environment has changed, and it is deleted.

Generating new situative rules

Generating a new situative rule requires the definition of a situation specification and of an action list. It does not make sense to define a situation specification arbitrarily rather it should be oriented to the situations in which an agent really may be, i.e. which are recorded in the history of situations.

In a real application it can be expected that the situations contained in the history are not equally distributed over time, over the sets of parameter values, and over the set of possible messages, rather there will be clusters with respect to the parameter values, to the number of messages that were received within certain time intervals, and to the possible messages themselves, i.e. sender and content. These clusters define subclasses of the history of situations. These classes can be described on the one hand by certain relations between the parameters and any constant values, and on the other hand either by the sender and content of the messages or only by the content when the sender is irrelevant.

In the history example above one can identify at least two clusters with respect to time, namely the sequence of changes of the motion-impulse parameter between the times 11 and 17 and the message receptions between times 28 and 34. Whereas there is no concept at hand for describing the first cluster as a class (possibly the concept of oscillation would be appropriate), the second cluster can be decribed by the following situation specification:

```
(motion-impulse = 0, ((?sender, event, M), 7 [5, 5]))
```

Abstracting from the sender yields the message pattern. The time t was computed by 34 - 28 +1. The interval [m, n] denotes just one number, namely the sum of received messages with the same content. This class definition may be modified when further situations are evaluated.

The detection of the clusters clearly can not be done manually. Instead an algorithm from the field of neural networks is used which follows the principle of the *Growing Cell Structures* cf. [2]. The Growing Cell Structures are a self-organizing type of neural networks, which means that they need not to be trained. The examples from the training set, in this case the history of situations, are presented to the algorithm. Each example must be described by a certain number of attributes, as it is usually required by learning algorithms. Here the parameters of the agent and the message contents or the combination of sender and content are taken as parameters.

The generation of situation specifications from the clusters proceeds in two steps. In the first step single clusters are considered and a first version of the condition and the message specifications are produced. For the condition, for each parameter p the set of its values val(p) is considered. If val(p) contains only one element, say d, $p = d$ is taken as one part of the condition. If val(p) contains more than one element and $d_1 = \min(val(p))$ and $d_2 = \min(val(p))$, then $d_1 \le p \le d_2$ is taken as one part of the condition. All the parts are connected by conjunction to form the whole condition. The message specifications are generated as follows: For each message pattern determine the first incoming time t_1 and the last incoming time t_2 and compute the time t as $t_2 - t_1 + 1$. If k is the number of occurrences of the message pattern, set [m, n] = [k, k].

In the second step, situation specifications that are similar to each other are combined. This step is only made if the situation specifications differ only in a few parameter specifications or message specifications. If a parameter is specified in one rule by $d_1 \le p \le d_2$ and in another rule by $d_3 \le p \le d_4$, these restrictions can be combined by disjunction. If the restrictions overlap, e.g. $d_1 < d_3 < d_2 < d_4$, they can be combined like $d_1 \le p \le d_4$. If a message specification with the same message pattern occurs in two rules with times t_1 and t_2 and intervals $[m_1, n_1]$ and $[m_2, n_2]$, they can be combined by defining $t = \max(t_1, t_2)$ and $[m, n]$ by $m = \min(m_1, m_2)$ and $n = \max(n_1, n_2)$.

When a situation specification has been generated, a corresponding action list has to be produced. This requires the definition of value assignments to parameters and of send statements. Although the generation of actions is more experimental than that of situation specifications, there are some restrictions on doing it. For instance, the possible value assignments are restricted by the set of parameters and their value domains. A value assignments also may depend on the content of a message specification. If e.g. the content is *higher*, then the assigned value must be higher than the actual one. The messages that can be sent are restricted by a set of possible contents, which is specific for a certain agent type, and by the set of possible receivers. In general, a single agent can only send to other agents that are located in some surroundings of the agent. This results in a local propagation of values by message passing which corresponds to the fact that there is no central control of the IH.

Changing the situation specification and the action list of a situative rule

Like with the generation of new rules, the changing of a situation specification of an existing rule is guided by the history of situations. If in the history some situations that are similar to a situation specification of the rule occur several times, the specification may be modified such that these situations are covered by the specification. This means that the new situations are considered as modifications of the cluster from which the situation specification originally was created, caused by changes in the environment. Therefore the situation specification must be adjusted to these changes. For instance, if the situation specification of rule is

(motion-impulse = 0, ((?sender, event, M), 7 [5, 7])))

and the history contains several sequences of situations with the same condition and the same message pattern but such that there are only 4 situations of this kind in the sequence within the time restriction 7, the specification can be adjusted to

(motion-impulse = 0, ((?sender, event, M), 7 [4, 7])))

Changes in the action list of an existing rule obey the same restrictions as the generation of actions of a new rule described above, thus this step will not be described in more detail.

Introducing a new parameter

A new parameter is introduced to further classify the behavior of an agent. Firstly, the rules are grouped together according to their action lists such that the rules in a group have the same or almost the same action list. Next, if there exists a quantitatve difference between the situation specifications of the rules in different groups, a new parameter is introduced which represents that difference. A quantitative difference may be given by message specifications with identical message patterns but strongly different intervals of incomings or by sequences of changes of parameter values within certain time intervals where the length of the sequences differs (i.e. many changes or only a few).

The newly introduced parameter essentially is a counter. However, it is evaluated only with respect to certain landmarks that are derived from the groups of rules. Typically there is just one landmark that can be regarded as a threshold but in general there are several. The rules of a group are now modified according to the following prescription: If q is the new parameter and l_1 and l_2 any landmarks of q, then the expression $l_1 < q < l_2$ is conjunctively added to the condition. In addition, a new rule for the manipulation of q has to be generated. In the situation specification of this rule the event or one of the events that cause q to be increased or decreased has to be recorded. In the action part q is increased or decreased. Because q registers the frequency of an event it is decreased in each time step, which is controlled by the function next-time.

The process of introducing a new parameter will be illustrated by the example of the motion detector, cf. section 2. The name of the parameter motion-impulse is abbreviated to mi. Assume, the following rules have been added to the rule set of the motion detector:

if mi = 0 or mi = 1, ((?sender, event, M), 6, [7, 20])
then send (M, event, {A1, A3, A4, A6})

if mi = 0 or mi = 1, ((?sender, event, M), 5, [6, 20])
then send (M, event, {A1, A3, A4, A6})

if mi = 0 or mi = 1, ((?sender, event, M), 7, [7, 20])
then send (M, event, {A1, A3, A4, A6})

The rules indicate that a message with content event has been received rather often. We introduce a new parameter called excitement together with the threshold value 10 (because this is the least number of incomings of the

message within 10 time steps according to the rules) and a parameter time. The value domain of excitement clearly are the nonnegative integers with default value 0 and the value domain of time is a predefined discrete set T of time steps with default value actual-time. The above rules are replaced with the following ones:

```
if ((?sender, event, M), t, [1, 1])
then excitement ← excitement + 1

if excitement > 10
then send (M, event, {A1, A3, A4, A6})

if next-time
then excitement ← excitement ÷ 1
```

The operation ÷1 yields 0 if the parameter already has the value 0.

An evolutionary process needs a fitness function to test the result of the operations applied to the elements of some population. Because the IH is a very complex system it is impossible to define a single fitness function for the whole system, rather this is done for the subsystems of the IH. We have developed such a function for the security subsystem, a kind of an artificial burglar, i.e. a reactively planning simulated robot that tries to get into the home. Other functions are in development.

4. Aspects of immunity-based systems in the Intelligent Home

Ishida [3] points out, refering to Metchinikoff, that the main job of the immune system is „identifying, defining, and maintaining the self". Self/non-self discrimination and eliminating the non-self are subgoals that are derived from the main goal. I adopt this point of view because it fits well to the IH. Also Ishida's assumptions about agents in an immunity-based approach are fulfilled by the IH agents.

In particular, the IH and its evolution as described in the above sections is a self-defining and self-maintaining system. The operations applied in the evolution process, namely modifying the rule set and modifying single rules can also be used to adapt the IH to a changing environment during its existence. The application of the rules can be viewed in the same manner as in [4]. The situations can be taken as antigens, the situation specifications in the rules behave like antibodies that are activated by certain situations. Because the situations are partly determined by incoming messages, the agents are connected in a network that reflects Jerne's network hypothesis [5]. The self/non-self discrimination is realized insofar as the IH can distinguish between the requests of different subsystems. If the influences from outside (burglars, weather) are viewed as non-self entities, then the special reactions of the IH on them by special subsystems are the elimination of the non-self.

5. Conclusion

In this paper I argued that the IH is a too complex system to be realized and maintained in a conventional manner. This is certainly one reason why up to now there does not exist an IH that contains all those functions that in principle could already be realized. I think that people dealing with IH technology should try to take another point of view of the whole technology to exploit all its possibilities. This paper describes a concept how to do it: Generating the IH by an evolutionary process and maintaining it according to the principles of the immune system.

6. References

[1] W. Dilger, „The Immune System of the Smart Home", *Workshop Notes, Workshop 4: Immunity-Based Systems*, ICMAS 96, Kyoto 1996, pp. 72 - 81.

[2] B. Fritzke, „Let it grow - Self-Organizing Feature Maps with Problem-Dependent Cell structure", *Procedings of the ICANN-91*, Helsinki 1991, Elsevier Science Publ.

[3] Y. Ishida, „The Immune System as a Self-Identification Process: A Survey and Proposal", *Workshop Notes, Workshop 4: Immunity-Based Systems*, ICMAS 96, Kyoto 1996, pp. 2 - 12.

[4] A. Ishiguro, Y. Watanabe, T. Kondo, Y. Shirai, Y. Uchikawa, „Immunoid: A Robot with a Decentralized Behavior Arbitration Mechanism Based on the Immune System", *Workshop Notes, Workshop 4: Immunity-Based Systems*, ICMAS 96, Kyoto 1996, pp. 82 - 92.

[5] N.K. Jerne, „The Immune System", *Scientific American*, vol. 229, No.1, 1973, pp. 52 - 60.

[6] L. Steels, „Cooperation between Distributed Agents through Self Organization", Y. Demazeau, J.-P. Müller (eds.), *Decentralized AI, Proceedings of the First European Workshop on Modeling Autonomous Agents in Multi-Agent Worlds*, Elsevier Science Publ. B.V., 1990, pp. 175 - 196.

[7] M. Wooldridge, N.R. Jennings, „Agent Theories, Architectures, and Languages: A survey", M. Wooldridge, N.R. Jennings (eds.), *Intelligent Agents. ECAI-94 Workshop on Agent Theories, Architectures, and Languages*, Springer Berlin, 1995, pp. 1 - 39.