EXPLOITING AMBIENT INFORMATION INTO REACTIVE AGENT ARCHITECTURES

P. N. Stamatis[1], I. D. Zaharakis[1], A. D. Kameas[1,2]

[1] Research Academic Computer Technology Institute, Hellas
[2] Hellenic Open University, Hellas
{stamatis,jzaharak,kameas}@cti.gr

ABSTRACT

This work presents an implementation of the subsumption architecture to agent individuals summing up a multi-agent system approach for the scale formation problem in secondary oil production. This approach is evaluated through a 3D simulator and some known issues concerning the architecture limitations are revealed and discussed. Then, the paper proposes a new hybrid architecture in order to minimize or even overcome these drawbacks.

INTRODUCTION

Future Ambient Intelligence (AmI) spaces will be populated by a large number of autonomous service providing elements, each of which will be capable of peer-to-peer communication as well as local sensing, processing and storage of digital data. In certain aspects, AmI spaces resemble distributed multi-agent systems, where agents collaborate towards the achievement of collective goals. Two important aspects of these systems are the limited resource available to each agent and the need to adapt to emerging context, such as unavailability of peer agents, uncertain information, network failure, etc.

In this paper we focus on multiagent systems consisting of a large number of small-sized, reactive, homogenous robotic agents with limited processing power and capabilities (energy, storage capacity etc), situated in a real-world environment. Limited capabilities force each individual agent to have sensing capability of its immediate environment only. Thus an individual agent is incapable to accomplish complex tasks by itself, and can only participate in task accomplishment by a group of agents. However, no centralized control mechanism is used in order to guide, direct or control the agents. In order to achieve reactiveness and simplicity, we implement the subsumption architecture model in every individual agent.

The paper is structured in three parts. In the first part, an extended description of the problem area is given, and the reasons that led to the multi-agent approach are discussed. Next, a brief survey on multi-agent systems is provided and the features of the subsumption architecture are discussed. In the second part, the intelligent environment is described and the implementation of the multiagent system is detailed in a bottom up fashion, from individual agent components to system versatility discussing some known or emerging issues. In the third part, an agent architecture design is proposed that overcomes some of the disadvantages of the subsumption model.

Target Application

The architecture presented here has been implemented in a multi-agent system, which has been used to deal with the scale formation problem in secondary oil production. This problem varies in severity depending on the composition of the make-up water used in water flooding. Scale deposits consisting either of Calcium carbonate or Calcium and Barium sulphate cause clogging in the pipes and damage to the pumping systems because of the formation of tenacious scale deposits. Calcium carbonate deposits tend to form around various nuclei of foreign material. Bulky and tenaciously adhering calcium carbonate deposits are formed in riser pipes used to control water level in secondary oil recovery processes. Moreover these deposits are encountered in heater-treatment units, i.e. storage tanks in which water is heated to be used for raising the temperature of the produced fluids facilitating the breakdown of water and oil emulsions in order to achieve separation of the two fluids. The process of steam-flooding of high-velocity oil reservoirs involves also heating water into tanks. Hardness leakage through the ion exchangers usually employed, in combination with the bicarbonates in water result in the formation of calcium carbonate scale that result in dramatic reduction of the heat transfer coefficient. Investigations on the formation of scale deposits in industrial systems indicate that the slower the fluid velocity the more intense the scaling problems due to the formation of insoluble salts like calcium carbonate, Cowan *et al* (15). Quasi static conditions imply no flow or very slow flow velocities.

However, the architecture is general enough to cover other application areas too, as it is based on the local representation of context, the absence of local knowledge and the context-dependent synthesis of basic behaviours. Thus the architecture can be applied to other environments with similar characteristics. For example, a nano-scale application would be the cleaning of medical equipment used for haemodialysis: the multi-agent system is embedded to the medical equipment

giving to it the ability of self-cleaning. Another robotic application would be garbage collection, by spreading robots in a designated area. Finally, a software application of this multi-agent approach could be developed for dynamic add hoc routing in communication networks.

Multi-agent Systems

In general, a multi-agent system consists of a (usually large) number of homogenous or heterogeneous, autonomous, decentralized agents. Each agent interacts with the environment, other agents or even people. Jennings *et al* (10) give a definition of a multi-agent system as a system with the following characteristics: i) each agent has incomplete information, or capabilities for solving a problem, thus each agent has a limited viewpoint, ii) there is no global system control, iii) data is decentralized, and iv) computation is asynchronous
Taking under consideration the target application environment, we find the multi-agent system approach application more suitable than the traditional artificial intelligence approach that would imply the development of a single sophisticated agent. By using numbers of small and process-limited agents we gain major advantages in contrast to using a single intelligent agent:
Space: an environment such as the abovementioned has limited space available for an agent to wander. Also, an intelligent agent capable of solving the problem by itself has greater size than an unsophisticated one, as it needs more computational resources and consequently more hardware components, energy power etc.
Cost: The total cost of developing/maintaining an intelligent agent is greater than creating tiny agents with limited capabilities: the multiagent system consists of low unit cost subsystems. Also, as we are more interested in agent societies than the individuals, the cost of losing a number of small agents isn't important compared to the loss of an intelligent agent. Also, a great number of low cost agents can be produced without getting concerned about long term maintenance.
Robustness and reliability: A multi-agent system is more stable than a single intelligent agent entity. Even if a number of agents are broken/destroyed/failed, the system will continue to operate, because other agents already available in the system may take over their part (16). If a single entity-processor or agent controls everything, then the entire system could crash if there is a single failure.
Distribution: The problem is distributed; groups of inter-operating agents can work separately in multiple targets while other groups search for new ones. A single entity cannot handle this distribution at the same time.
Scalability: Since multi-agent systems are inherently modular, it should be easier to add new agents to a multi-agent system than it is to add new capabilities to a monolithic system. Systems whose capabilities and parameters are likely to need to change over time or across agents can also benefit from this advantage of multiagent systems, Stone *et al* (17).
Complexity and reactiveness: Taking under consideration the target application area, why create a complicated agent that wastes time and resources making complicated computations when a multi-agent system consisting of unsophisticated agents operating asynchronously and in parallel, reacting immediately to environmental changes may be fitted to accomplish the same?
Development and Reusability (16). Individual agents can be developed separately; the overall system can be tested and maintained more easily, and it may be possible to reconfigure and reuse agents in different application scenarios.
Apparently, the multi-agent system approach is the best fitted approach in problem areas where the problem occurrence is distributed in a large area and there are serious cost and space circumscriptions. In the following, we focus on the individual agent of the distributed system, dealing with the issues of reactiveness and simplicity.

The Subsumption Architecture

In the latest years Artificial Life (ALife), a new AI trend emerged which differs in major concepts from the traditional AI scheme. Assuming that high level intelligence is too complex to analyze, model and manipulate, the ALife scheme uses a bottom-up approach assuming that intelligence arises from low level component interaction. Reactive behaviour systems are the most known systems that have emerged from this scheme. These systems regard behaviour as a building block that relies on sensory input for determining a command; a control mechanism interferes between behaviours, excluding or including behaviours.
The arguably best-known reactive architecture is the subsumption architecture, developed by Brooks (1). According to this behaviour based architecture, each individual robotic agent consists of a sensor set, an actuator set and finally a set of simple behaviours; a simple behaviour may be thought as an individual action function which continually takes perceptual input (by direct connection to the sensors) and maps it to an action to perform.

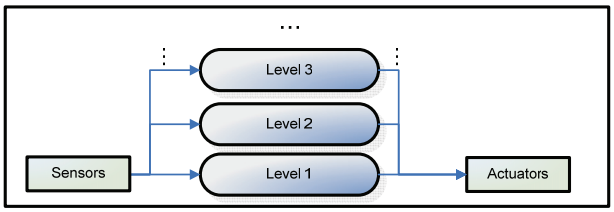

Fig 1. A typical subsumption model, as depicted in (1).

The set of simple behaviours is placed in different layers creating the subsumption hierarchy. Each behaviour module is based on the *situation→action* schema. A major defining characteristic is that more than one behaviours may be triggered simultaneously. Still, through the subsumption hierarchy, only one takes control of the actuators. This means that a) there exists a hierarchy between the behaviours, giving the privilege to high priority behaviours to suppress lower importance behaviours and b) each behaviour is autonomous, meaning that it can take control of the agent actuators if permitted. Moreover, each behaviour is represented by a finite state machine (FSM), arranged in the layered structure and use the inhibition / suppression mechanisms to influence other behaviours and to control output. The control system may be a simple switching control circuit, giving great simplicity in implementation. Also, the energy consumption of such a control system is minimal, while no memory or processing power is needed. To summarize, there are some obvious advantages of the subsumption architecture (and consequently in pure reactive architectures): simplicity, economy, computational tractability, robustness against failure and elegance, which make this architecture appealing, Wooldridge (5). A typical subsumption model is presented in figure 1.

Still, there are some important disadvantages in the subsumption architecture model. First, the system designer has to specify the suppression architecture manually, having in mind the tasks the agent is built to execute and satisfy. This disadvantage is brought out by the use of the term "emergent behaviours", meaning the overall behaviour that emerges from interaction of the component behaviours: the very term "emergent" suggests that the relationship between individual behaviours, environment and overall behaviour are not understandable (5). As there is no principled methodology for creating the behaviour architecture, the designer has to use a laborious process of experimentation, trial and error to engineer the agent, in order to achieve an "eligible" system performance ,Rosenblatt and Payton (6). As an example, Behaviour Analysis and Training (BAT) methodology, Colombetti *et al* (14) assumes that behaviours are allocated to behaviour al modules by design, but the function of each module is developed by machine learning techniques. Although, it is possible to speed up training through the use of simulators, still the training phase can be an extremely expensive task. Additionally, after the time the agents are created, the architecture is concrete and cannot be changed. In other words, the system cannot learn from any form of experience in order to improve its performance due to dynamic environmental changes. Brooks specializes this problem on the automatic construction/modification of the individual behaviour generating modules Brooks (12). Another major drawback is the unpredictable expandability of the system. As new simple behaviours are asserted to the agent individual, the interactions between the behaviour set become more complex, making it very difficult for someone to predict the result. Any hierarchical perturbation may result in dramatic changes in the system's balance. The asserted behaviour's level of suppression must be tested by the designer before the assertion is applied to the behaviour set, in order to obtain each separate module being productive and cooperative (12). An important drawback found not only in the subsumption architecture but in all reactive architectures is the "short-term" agent view: An agent does not employ models of its environment, and any action is determined based on local information (5). This can trap the agent to deadlock situations that the agent is unable to escape from. Finally, this approach cannot deal effectively with the situation when more than one behaviours are triggered simultaneously. This is possible when a set of behaviours coexists at the same time without affecting each other. As the suppression scheme allows only one behaviour to act every time, simultaneous actions are prohibited. This one-can-pass restriction places serious limits that cannot be overcome without creating a major complexity burden, as the particular behaviour cannot stand by itself. Rosenblatt and Payton refer to the above problem as the "Inadequate Command Fusion" (6).

Some of the abovementioned problems may be treated either by creating task-specialized alternatives of the subsumption architecture (6), or by using complementary architectures/techniques. Another alternative is to create hybrid architectures, e.g. Flakey robot by Saffiotti (9) applies hierarchy in a fuzzy behaviour architecture or TOURINGMACHINES, Ferguson (3), a hybrid architecture system containing subsumption).

SYSTEM DESCRIPTION

The following section focuses on the illustration of the intelligent environment, and on the analysis of some arising issues. Moreover, it describes the adopted architecture and analyzes the sensing, behaviour and actuator modules. The oil pipe environment is used as an example, although the approach can be generalized and applied to other environments as discussed in the introduction.

The intelligent environment was simulated through a Java built application using the J3D API. The application provides 3D observation and CSV accumulation data. Accumulation data concern global phenomena, e.g. fault creation (time, position, range etc), fault repair and local phenomena like collisions (agent with agent and agent with other permanent obstacles), total transitions, etc., 3D observation is needed to observe emerging group behaviours like chain following and flocking.

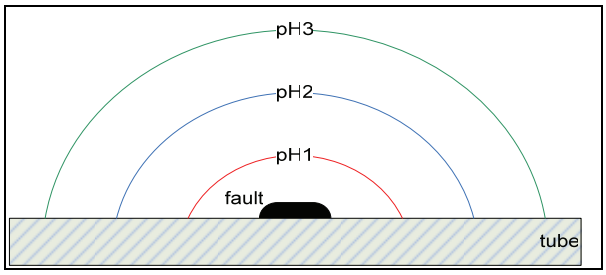

Fig 2. Representation of the damage incipient on calcification. As a consequence of the formation of scale, local pH drops and the signal is transmitted to the pH sensors that record pH values depending on their spatial distribution with respect to the damage (8).

The intelligent environment

Intelligence is diffused to the oil tank environment by spreading a large number of small-sized agents inside one compartment of the fluidic environment. Agents circulate in the compartment continuously sensing for "fault signals". The "fault signal" is an environmental stimulus that signals the presence of a faulty element. In that case, it corresponds to an altered chemical/physical property e.g. pH/conductance. A representation of a damaged area as the consequence of the damage in the environment is given by Fig 2.

When an agent perceives a "fault signal", it moves towards its source (i.e. the fault area) and simultaneously releases a "pherormone" signal, which ultimately "attracts" other agents: when any of the other agents perceive the signal, they move towards its source (i.e. the agent having perceived the fault signal) and also start emitting the pherormone signal themselves. The effect of pherormone signal on each agent is cumulative. As pherormones are summed more agents are eventually attracted towards the fault area; when a signal threshold is exceeded, a spatio-temporally ordered community is formed, which deals with the fault (i.e. by releasing a proper chemical substance). When agents repair the fault the strength of the original environmental stimulus drops, pherormones diminish and hence the community disperses.

The large number of agents spread over the compartment reinforces the overall sensing capability of the system: Each agent has a local sensing area but the overall system sensing depends on the distribution of the agents. In normal conditions, each agent tends to avoid social gathering, so as to minimize the chances of collision and, maximize the overall system sensing area at the same time. Collision with any moving or static "obstacle" is a non-eligible situation that can cause stultification of the agent.

Also, due to limited capability, an agent is not capable of dealing with the fault by itself. The total effort increases (proportionally or even exponentially) depending on the agent number density in the particular area of the fault. Thus, in such a situation, the need of agent coiling is necessary.

An important resurgent issue is the form of communication. We chose indirect communication instead of direct communication as the latter is undesirable for certain reasons. First, the sensing capability of each agent is local and not global. This implies that a listening agent must compute a series of transformations in order to use that information. Besides, limited capabilities and energy consumption restrict large packet transfer. The form of the indirect communication is active stigmergy: an agent alters the environment so as to affect the sensory input of another agent. In this case, alteration occurs with the broadcasting of the pherormone signal, which is used as a "quorum sense" signal. This form of stigmergy covers the lack of memory of the agent in a simple manner: The information is received and placed in the environment. The agent doesn't have to remember the position of the fault. It just follows the path to it. To rephrase Brooks, we do not model any individual agent neither the environment; instead, the "world is the model itself".

Another issue is agent size. Taking under consideration the environment's fixed capacity, system performance is related to agent size and agent total number: system performance is growing relatively with the number of agents. Still, as agent density grows, agent circulation is constrained as each agent has to avoid many "moving" obstacles. Reducing agent capacity without reducing agent acting power is a feasible (if not the only) option. In this paper, we do not intend to study ways of reducing sensor-actuator size. Sensor and actuator modules are provided with fixed capacity and energy consumption specifications On the other hand, the requiring energy storage module holds a very significant percentage of the total capacity. Considering the current battery miniaturization technology, we focus on minimizing energy consumption. This inevitably leads to the agent processing module. In order to preserve energy and space, the processing module must be as small as it can be. Considering that circuit size is related to processing power and complexity, we conclude that the only way of reducing agent capacity without affecting its sensor/actuator capabilities is implementing a simple, non sophisticated processing unit.

The final issue is system effectiveness. The intelligent environment must deal with the fault as soon as possible as any delay may worsen the fault area situation. This means that the system must react immediately from the time a fault is created to the time the fault is repaired. Three critical time lapses are discriminated, making the difference between desirable and undesirable outcomes:

(i) fault creation to fault detection lapse
(ii) fault detection to agent rallying lapse
(iii) agent rallying to fault repair lapse.

It is obvious that overall system reactivity is depended on

a) the distribution of the agents; this in turn depends on 1) the "social repulsion" of the agent which

depends on behaviour design and implementation and 2) the number of the agent individuals involved. The (i) fault creation to fault detection timing lapse improves by sparse distributions of as many as possible agents.

b) the sensor/actuator power of the agent individual, e.g. how fast can it move, what is the strength of the repairing actuator etc. This affects both (ii) and (iii) lapses.

c) the individual agent reactiveness, regarding react agent reaction speed to sensor stimuli perceived. This is a crucial issue because through the procedure of mapping the sensor stimuli to the actuators the whole performance of the agent is affected. Delayed reaction wastes/degrades sensing/ acting capability affecting the overall system capability. In (ii) lapse for example, if agents don't exploit stigmergic information in a productive way, valuable time will be lost until a group/swarm is formed to repair the fault.

The above mentioned agent individual requirements lead to the subsumption architecture schema implementation which is described in the next paragraph.

The Subsumption Architecture schema implementation

The subsumption architecture is implemented in each agent individual. Each agent is equipped with sensor devices (the sensor module), actuator devices (motors pumps etc) and finally the behaviour module in which the Subsumption hierarchy is applied.

Sensor module: This module consists of three kinds of sensing elements: a set of infrared sensors placed on the body of agent so that they can sense a short-range spherical area around the agent. Depending on the values returned by the set of these sensors, the agent is able to separate large from small objects. Also, it can calculate the relative distance between the objects and the agent. A pH sensor is used in order to make pH measurements. Any pH disturbance (drop) indicates a fault area. Finally, the agent is equipped with a "pherormone" sensor, a virtual sensing element which traces pherormone signals and returns its intensity.

Actuator module: This module consists of four actuators. The first actuator is the `steering` actuator. This actuator controls the orientation of the agent in space. The second actuator is `speed` actuator, which controls the speed of the agent move. The `release pherormone` actuator is a virtual on/ off actuator that releases a pherormone signal when enabled. Finally the `release` chemical substance actuator is an on/off actuator that releases a fixed quantity of the proper chemical substance in order to clean the fault area from calcium carbonate.

Behaviour module: This module consists of seven simple behaviours. A high level description of these behaviours is given below.

`avoid_obstacles.` This simple behaviour is enabled when the agent is close to a mass object. The behaviour fires a new `steering` direction that is reversal to the direction towards this object. The `speed` command depends on the distance of the object.

`avoid_kin.` This simple behaviour ensures the agent won't collide with other agents. In this point it is remarked that the simple behaviour name doesn't imply that the agent can conceptualise or recognize other agents; it only senses "small moving objects". The operation of this behaviour is similar to the `avoid_obstacles` behaviour. The behaviour is enabled when an agent is close enough and fires a new `steering` direction that is reversal to the direction between the two agents. The `speed` command depends on the distance of the object. If more than one agents are close enough, the behaviour converges to avoid the closest one.

`chase.` A simple behaviour that makes the agent chase another agent that is close enough. This behaviour is enabled when a "moving obstacle" is traced from the front infrared sensors ("front" in 3-dimentions is the direction of move). The behaviour fires a new `steering` direction directly to the traced object. `Speed` command is relative to the distance between the two agents. In case more than one agent are in the zone of follow enabling, the behaviour chooses randomly.

`aggregate.` A simple behaviour that makes the agent move towards a centroid point. This centroid is the centre of the area defined by the sensed objects around the agent. This behaviour is responsible for agent rallying. The behaviour fires a new `steering` direction directly to the centroid point. `Speed` command is relative to the distance between the agent and the centroid. It also enables the `pheromone` actuator.

`disperse`. This simple behaviour is contradictory to `aggregate` behaviour. It is used to spread dense formations of agents. The behaviour is enabled when the density of "moving obstacles" around the agent is big enough. The behaviour fires a new `steering` direction directly reverse to the centroid point. `Speed` command is relative to the distance between the agent and the centroid.

`wander.` The basic behaviour that enables the agent to circulate in the tank. This behaviour fires random `steering` and a particular `speed` command to the actuators. This behaviour is always enabled.

`repair.` This is the final simple behaviour. This behaviour is triggered when the pH value returned from the pH sensor has exceeded a threshold value. This results a steering command towards the sensed large

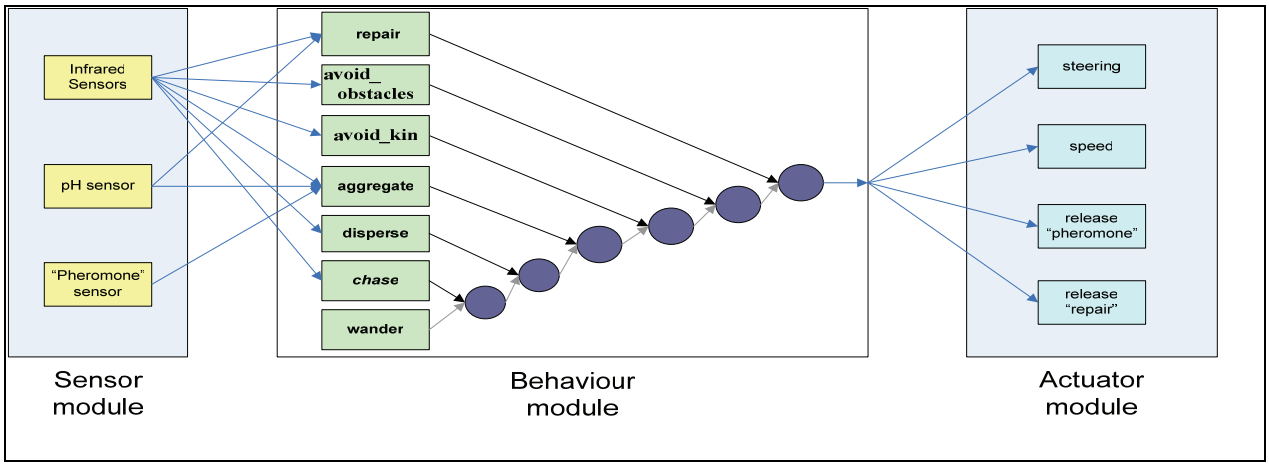

Fig 3. A detailed description of the three agent components, focusing on the connection between sensors and simple behaviours and the subsumption hierarchy.

object with a minimal speed command. If it touches the object, the behaviour sends the `release` chemical substance command, making the agent release a fixed quantity of the chemical substance.

Subsumption hierarchy

Fig 3 depicts a detailed description of the agent components, showing the sensory input for each simple behaviour and the suppression hierarchy between the seven simple behaviours. Suppression is represented by a two input/ one output link switch node: the node propagates data arriving from the weak (grey) link only if no data has arrived through the strong (black) link. High importance behaviours are placed on top, suppressing behaviours of lower importance, placed below them. The most important behaviour is `repair`. If the `repair` behaviour is enabled, all other behaviours are suppressed. The next behaviour in order of importance is `avoid_obstacles`, which secures the agent from bumping into other things. The suppression hierarchy continues with the `avoid_kin`, `aggregate`, `disperse`, `chase` and finally the `wander` behaviour. The least important `wander` behaviour controls the actuators only if no other behaviour is enabled. The particular hierarchy came up after several simulations concerning different hierarchies. Each hierarchy was tested according to the total number of collisions and the fault handling time. Although other hierarchies dealt also well with the fault and collisions, the particular hierarchy was chosen because of the most evident occurrence of some complicated behavior discussed later.

The test case scenarios

The simulation scenario considers a cubic water tank and a cylindrical tube (oil pipe) crossing in the middle of the tank. Agents (conic objects) are floating inside the tank. The first test case scenario involved an overall observation of the agents during a bounded interval. The second test case scenario involved the manual creation of a major fault on the tube and agents overall behaviour observation. This test set was repeated for a small (30), a medium (100) and a large (200) number of agents.
The first result is that in normal situations where no fault area was formed, no collision happened. On the other side, multiple collisions occurred when agents where gathering around a fault area. The number of collisions was relative to the total number of agents (Fig 4). A possible reason for that could be the restriction of movement as the agent gets closer to other agents, taking under consideration that the `avoid_kin` behaviour does not perform well when it has to choose one obstacle to avoid. A possible solution would be the use of `dispersion` but unfortunately this would slow if not restrict agent group formations, as the particular behaviour is contradictory to `aggregation` behaviour. Another important result is that the number of `repair` behaviors is not growing relative to the agent total number (Fig 5). The number of repairs has a meaning regarding that a released chemical compound is added to the chemical compound released already. This is probably because of the relative to the agent total number increasing of collisions (collisions/repair: 0,39%, 2,16% and 7,47% for 30agent, 100agent and 200agent system respectively). Finally, chain following was intense (formation and duration of its consistency) in small and average numbers of agents. In large numbers of agents, this phenomenon was sensibly weaker. In Fig 6 we give a snapshot from the simulator, showing agent rallying around a "fault" area.

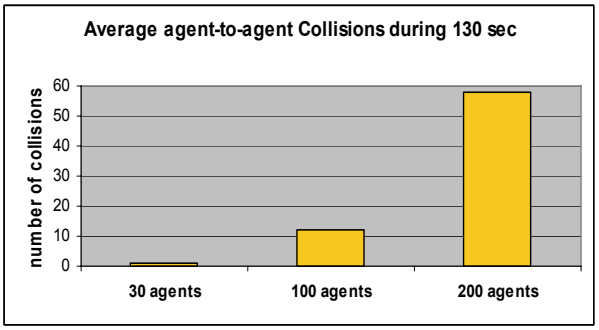

Fig 4. The total number of collisions during the 130 seconds of simulation.

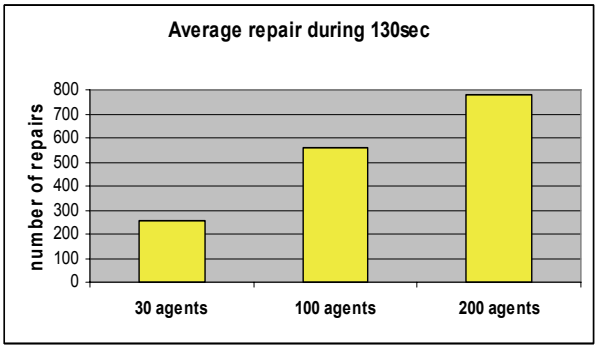

Fig 5. The total number of repairs during the 130 seconds of simulation.

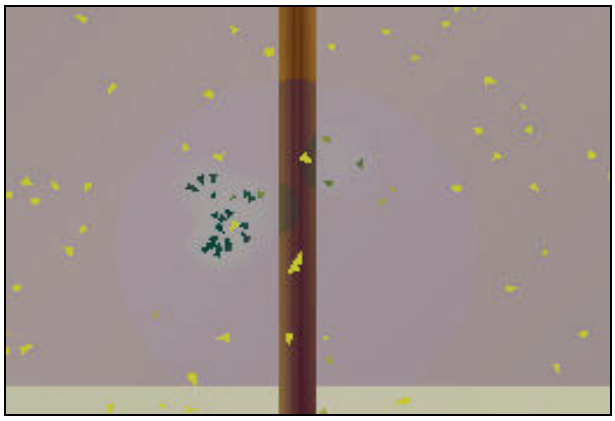

Fig 6. A snapshot showing the creation of a group (flock) of agents near the fault area.

Discussion

The subsumption architecture was applied to the scale formation problem, where simulation showed that a multi agent system consisting of agents applying the purely reactive subsumption model is well fitted. It came out that more complex behaviours emerge even when no indirect communication is applied. If the area of the pH distribution is large, a satisfactory number of agents will sense the pH disturbance. The pherormone distribution and attenuation do not "enlarge" the already large fault area while inside the pH disturbance, aggregation is enabled no mater whether a pherormone exists or not. The importance of pherormone becomes bigger if the area of pH disturbance is small; the pheromone "adds" important space to the fault area, making more agents enable the aggregation behaviour.

During the simulation observation, the system revealed an overall complicated collective behaviour that emerges from the interactions of the agent individuals. In Fig 7, a bottom-up diagram is given presenting the way simple behaviours in a single agent level are affecting global system behaviour, making it exhibit an overall collective behaviour. An emerging behavior arises from simple behavior interaction in such an intra-agent (suppresion) as to an inter-agent (behavior sequences over time) level. The **safe wandering** overall behaviour emerges from the combination of `wander`, `avoid_obstacles`, and `avoid_kin` simple behaviours between a set of agents. Agents circulate in the environment while avoiding bumping into the walls, the tube or other agents. This behaviour can also be observed by the overall combination of the three component behaviours in a particular individual. The **safe following** behaviour is an emerging behaviour from the sequential combination of `avoid_kin` and `chase`. An agent tries to chase another agent that is in front of it. When the agent gets very close, the higher importance `avoid_kin` behaviour is enabled. This behaviour is observed in pairs of agents. The **chain group safe wandering** high level emerging behaviour is observed in large groups of agents. This behaviour describes dynamic chains of 3-10 agents circulating in the environment. The duration of a chain formation depends on the non-chained agent distribution on the area of the chain formation. Chains seem to diminish when chained agents are forced to change dramatically their steering direction in order to avoid an agent side-approaching or the chain is lead to a wall or tube. In the second case, the leading agent is forced to make a large and continuous turn "confusing" the chain followers.

The **flock** emerging behaviour is observed when agents enable the simple `aggregate` behaviour. The overall result is a group of agents converging to the centre of the group area, creating a slow moving flock. As the resulting flock is converging, it is possible that some agents get closer enough the fault, enabling the `repair` behaviour. This implies that the centre of the flock is shifting towards the fault area. On the other hand, pheromone release attracts more agents (or chains, according to the relative emerging behaviour). The result is that agents closer to the fault are heading towards the tube releasing the chemical substance while others try to keep a relative position to the flock, always trying to avoid each other. This leads to an overall intelligent self-healing attribute of the system.

The **safe diminishment** emerging behaviour is observed in areas that agent distribution is dense, possibly after a flock creation. The `disperse` behaviour, combined with the higher importance `avoid_kin` and

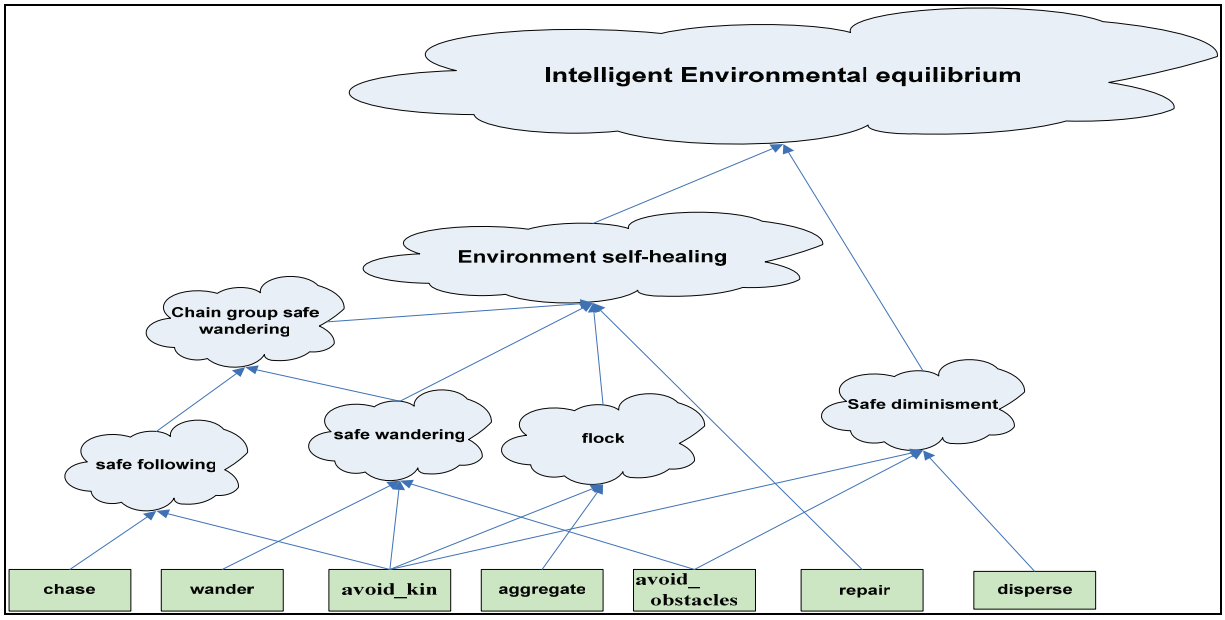

Fig 7. Interpretation of System emerging behaviours

`avoid_obstacles` behaviours is enabled leading to a gradient diminishment of the flock

Finally, when the fault is cleaned and thus no pH disturbance is sensed, the safe diminishment emergent behaviour occurs spreading the dense flock formation. We claim that concatenation of the environment self-healing and safe diminishment results to an overall system high level property, the **intelligent environmental equilibrium**.

During the simulation, we exposed the very important issue of each behaviour triggering conditions. The subsumption architecture is a very simple architecture that is based on behavioural hierarchy. If more than one behaviours are triggered, the higher importance behaviour suppresses the others. Also, different importance behaviours get the same sensor data. It is obvious that the triggering conditions of a behaviour are very important and affect not only the behaviour rate of triggering, but a) the prevail of lower importance behaviours and b) the occurrence of global phenomena /emerging behaviours. A very prominent example is the safe following emerging behaviour: The `chase` and the higher-importance `avoid_kin` simple behaviours use the same sensory input. Each behaviour has an akin triggering condition, which is the distance between the agent and a "moving obstacle". In each case, the behaviour is triggered if this distance is less than a distance threshold, let d_1 and d_2 for the `chase` and `avoid_kin` behaviour respectively. If $d_1 < d_2$, the `chase` behaviour will never prevail, leading to safe following extinction which finally affect the whole environmental equilibrium. On the other hand, $d_1 > d_2$ condition doesn't ensure the system stability. If the difference Δd is short enough, a behavioural switching from `chase` to `avoid_kin` may not have the expected results, e.g. the chasing agent cannot slow down/stop in time causing a collision. Collisions are not expected (especially between high speed moving agents) from the system, which in every case looses its stability collisions reduce safe following, which in turn reduces chain sustention which leads to an untuned overall appearance. In this point we cite that the emerging behaviours were observed for a particular tuning of every simple behaviour triggering conditions. Changing these conditions, we surely untune or even diminish the emergent behaviours, it is possible on the other hand, other complicated behaviours to occur. This leads to the claim that intelligence is an attribute that is given by the beholder.

Future Work

The previous section exposed (agreed with the) several problems and limitations of the subsumption architecture discussed in beginning of this paper. In this section, a hybrid architecture is proposed that combines the subsumption architecture and fuzzy behaviour univalued based system approaches and methods in a different way in order to create simple, reactive, and scalable robotic agents, overriding these limitations at the same time.

Each agent individual consists of four separate modules. The first one is the **Sensor module**. This module contains various input devices (sensor devices) that the agent is equipped with. The **Behaviour module** essentially is a collection of simple behaviours which are hardware components implemented to execute specific tasks. The **Selection module** is actually a controller that gathers every simple behaviour output and selects a final output, which is passed to the **Actuator module**. The actuator module contains the several actuators the agent is equipped with in figure 8

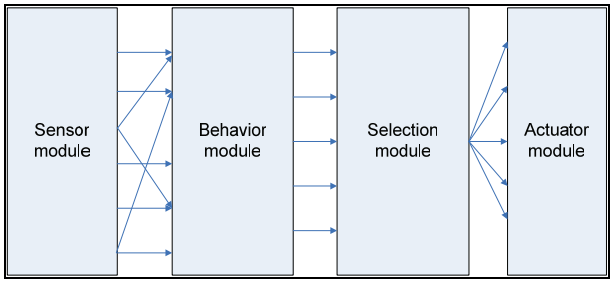

Fig 8 A general representation of the proposing hybrid system.

we give a representation of the modules and their connections.

Behaviour module: This module contains each simple behaviour. A simple behaviour is a separate hardware module which is connected to the Sensor module which provides the behaviour with sensory input. This type of connection may differ between different simple behaviours as each simple behaviour is fed by different sensory input. Each simple behaviour uses the sensory input to react immediately producing a message that is propagated to the Selection module. The operation of each simple behaviour is alike to the operation of the behaviour of the reactive (subsumption) model. The difference is that the simple behaviour is not directly connected to the actuators but it is reacting in a behavioural stage, producing a data stream (*packet*) that is propagated to the next module. This packet contains a header and separate blocks of data, corresponding to each actuator.

Selection module: The Selection module receives the behavioural outputs as packets. These packets are propagated to a network of nodes, each of comparing two neighbouring packets. Depending on the result, each node propagates the "best" of the packets (suppression) or an integration of these packets into one. In general, the Selection module for n simple behaviours can be represented as a grid of nodes, separated in n layers. The first layer contains n nodes while the last layer contains 1 node. Each middle layer contains 2-input/2-output nodes, where the input links are outputs of two adjacent nodes of the previous layer while the two outputs are linked with the nodes of the next layer. Figure 9 represents the produced grid for four simple behaviours.

Each simple behaviour fires when all of its triggering conditions are satisfied. A triggering condition may be a signal from, for example, an infrared sensor, a particular value of the pH sensor etc. We claim that no matter the assigned importance of the behaviour, the grade of behaviour triggering must be taken under consideration. Thus, each sensor signal is passed through a fuzzy controller that computes a response factor independently of the simple behaviour output. The controller output is added to the simple behaviour output and the data packet is then propagated to the selection module.

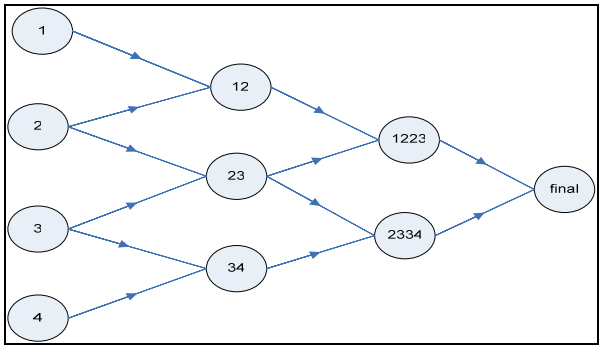

Fig 9. A 10-node grid correspond to four simple behaviours.

A total weight w is computed regarding both the assigned importance (behavioural importance $\underline{s}$) and the grade of behaviour triggering p:

$$w=f(s,p)$$

The node behavior is to compare the packets arriving, let $packet_1$ and $packet_2$, with w_1, w_2 total weights respectively;

$$\Delta w = w_1 - w_2$$

Next, according to a threshold $\theta \in \mathbb{R}^+$, the node propagates either one of the packets (suppression) or an integration of the two packets. After the final packet is propagated by the final node, it is rationed out and propagated to the Actuator module.

User defined suppression factor and system-defined sensor sensitivity factor are both considered into a total weight. According to this new factor, the behaviour output packets are emulating each other in a triadic logic. This logic makes the agent behaviour changes smoother, depending on the grading of sensory input. Also it keeps the reactive character of the involved approaches.

This design can support training in three different aspects of data propagation from the moment a stimulus is perceived to the moment an actuator is ordered to act. The first aspect is Simple behaviour triggering. Can we find the "best" set of triggering thresholds in order to maximize the Simple behaviour efficiency taking under consideration the possible interaction with other triggered Simple Behaviours? Also, what is the "best" relationship between the response factor and sensory input? Another aspect of training is the behaviour integration/suppression procedure in the Selection module. First, the integration rate is relative to the threshold value; Second, the formula $w=f(s,p)$ can be modified; What is the relationship between user-defined suppression and system/sensor defined response factor? Finally, training can occur regarding packet integration. In that case, crucial answers must be taken about packet manipulation and System tolerance to erroneous packet integrations.

Conclusions

The simulated environment confirmed that a multi-agent system consisting of agents applying the purely reactive subsumption model is well fitted to the scale formation problem. During the simulation, new collective behaviours emerge even when no indirect communication is applied. Although these collective behaviours (which represent the "intelligence" of the environment) are very sensitive not only to changes in the behavioural hierarchy in the agent level but also to tiny modifications of the simple behaviour triggering conditions in the simple behaviour level. The way simple behaviours of different agents interact through time can be estimated and not predicted before the simulation. This estimation is validated through the resulting collective behaviors.
Finally, a new hybrid architecture is proposed that combines the subsumption architecture and fuzzy behaviour univalued based system approaches. This new architecture increases the acting potential of each agent and may support training; still this issue is premature and must be studied in future work.

ACKNOWLEDGMENT

This work was supported in part by a grant from the European Community under the "Information Society Technologies" Programme (01/01/2003-31/12/2005), Project SOCIAL IST-2001-38911, http://www.socialspike.net.

REFERENCES

1 Brooks, R., 1985, "A robust layered control system for a mobile robot", Tech. Rep. A.I. Memo 864, Massachusetts Institute of Technology Artificial Intelligence Laboratory

2 Butler, G, Gantchev, A, Grogono, P., 1999, "Reusable Strategies for Software Agents via the Subsumption Architecture", apsec, p. 326, Sixth Asia-Pacific Software Engineering Conference

3 Ferguson, I., 1992, "TouringMachines: An Architecture for Dynamic, Rational, Mobile Agents", PhD thesis, Clare Hall, University of Cambridge, UK (1992) (Technical Report No. 273, University of Cambridge Computer Laboratory)

4 Müller, J., 1997, "A cooperation model for autonomous agents", Intelligent Agents III (LNAI Volume 1193). Springer-Verlag: Berlin, Germany, 245–260

5 Wooldridge, M., 1999, "Intelligent Agents", in G. Weiss (ed.) Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence, MIT Press

6 Rosenblatt, K, Payton, D., 1989, "A fine grained alternative to the subsumption architecture for mobile robot control", In Proceedings of the IEEE/INNS International Joint Conference on Neural Networks, Washington, D.C.

7 Nilsson, N., 2000, "Learning Strategies for Mid-Level Robot Control: Some Preliminary Considerations and Results" (unpublished note available at http://ai.stanford.edu/users/nilsson/publications.htm)

8 SOCIAL Project, IST-2001-38911, http://www.socialspike.net/

9 Saffiotti, 1998, "A. Fuzzy Logic in Autonomous Robot Navigation: a case study", Chapter G6 ``Autonomous Robot Navigation" , Ruspini, E, Bonissone, P, Pedrycz W.: Handbook of Fuzzy Computation, Eds. Oxford Univ. Press and IOP Press

10 Jennings, N, Sycara, K, Wooldridge, M., 1998, "A roadmap of agent research and development", Journal of Autonomous Agents and Multi-Agent Systems 1, 275-306

11 Wooldridge, M, Jennings, N., 1995, "Intelligent agents: Theory and practice", Knowledge Engineering Review 10(2)

12 Brooks R., 1990, "Elephants Don't Play Chess", Robotics and Autonomous Systems, Vol. 6, 3-15

13 Brooks R., 1991, "Intelligence without representation," Artificial Intelligence, vol.47, no.1-3, 139–159

14 Colombetti M., Dorigo M., Borghi G., 1996, "Behavior Analysis and Training: A Methodology for Behavior Engineering", IEEE Transactions on Systems, Man, and Cybernetics-Part B, 26, 3, 365-380

15 Cowan, J, Weintritt, D., 1975, Water Formed Scale Deposits, Gulf Publ. Co. Houston Texas, 223-226

16 Weiss G., (ed.), 1999, "Multiagent Systems: A Modern Approach to Distributed Artificial

Intelligence ", The MIT Press, Cambridge, Massachusetts

17 Stone P.and Veloso M., 1997, “Multiagent systems: A survey from a machine learning perspective”. Submitted to Journal of Artificial Intelligence Research (JAIR)

18 Craig Reynolds, BOIDS, http://www.red3d.com/cwr/boids/