Computer science Education

ISSN: 0899-3408 (Print) 1744-5175 (Online) Journal homepage: https://www.tandfonline.com/loi/ncse20

Teaching Powerful Ideas With Autonomous Mobile Robots

Rolf Pfeifer

To cite this article: Rolf Pfeifer (1996) Teaching Powerful Ideas With Autonomous Mobile Robots, Computer science Education, 7:2, 161-186, DOI: 10.1080/0899340960070203

To link to this article: https://doi.org/10.1080/0899340960070203

Published online: 03 Aug 2006.

COMPUTER SCIENCE EDUCATION 7, 2 (1996)

Teaching Powerful Ideas With Autonomous Mobile Robots

Rolf Pfeifer
University of Zurich

Core computer science is about virtual machines, whereas computer applications and robotics are about the real world. The hypothesis that we will pursue in this paper is that real-world autonomous robots provide a tool that enables us to learn about how to design intelligent computer systems for the real world. They can teach us many powerful ideas about intelligence and computer systems in general. We will show a number of case studies illustrating how complex behaviors can be achieved by very simple mechanisms (e.g. how robots can clean up without knowing about it), and how robots learn to form concepts about the real world depending on how they are built. We also show how intelligence does not require central control and how the real world is not something to fight against, but something to live with and to exploit beneficially. We discuss how real-world autonomous agents can be used to teach an interdisciplinary audience. Finally, we discuss potential ways in which ideas like the ones developed in this paper might eventually change computer science.

1. INTRODUCTION

The business of computer science is information processing. Obviously, computers process information[1]: there is input, the input is somehow processed, and finally some kind of output is produced. This way of thinking, which we will call the "information processing metaphor", is ubiquitous and has shaped our views of the world. In particular it has strongly influenced our notion of intelligence. Humans are information processing systems; the brain is a computer. Terms like cognition, memory storage and retrieval, levels of processing, deferred updating, neural computation, perceptual processing, and sensory buffers abound in the literature in psychology, artificial intelligence, and the neurosciences. Human experts like medical doctors, service engineers, investment advisors, loan experts, and insurance consultants are seen as problem solvers, and problem solvers in turn are, in essence, information processors: the problem is presented to the expert, the expert performs some information processing on the problem like

I would like to thank Andreas Aebi, Clemens Cap, and Ralf Salomon for many stimulating discussions and suggestions concerning the manuscript. Thanks goes also to the anonymous reviewers whose comments have greatly helped improve the text.

Correspondence and requests for reprints should be sent to Rolf Pfeifer, AI Lab, Computer Science Dept., University of Zurich, Winterthurerstrasse 190, CH-8057 Zurich, Switzerland

drawing logical inferences, applying domain knowledge to the current case by using an interpreter, evaluating alternatives, and he finally comes up with a solution to the problem which is the output.

Many people endorse this perspective, especially in computer science, but also in other scientific disciplines like psychology and the neurosciences. If you share this intuition, the next logical step is to try and automate this problem solving or information processing behavior of humans. One of the outcomes was the idea of expert systems, which at the time, in the heyday of classical artificial intelligence, seemed very natural and generated a lot of enthusiasm. Alas, it never really worked out. The successes were very limited. Expert systems only worked in very restricted domains.

Why? The essential point that we will elaborate in detail later on, was that computer applications deal with the *real world*, whereas core computer science is about *virtual worlds*. The theoretical underpinning of computer science is the *virtual machine*, an enormously powerful concept, precisely because it does not refer to the real world, to a real machine, but to an abstract one. We owe to this concept, that we do not have to worry about the kind of computer our programs will run on—the algorithms are independent of their physical realization. And the virtual machines are universal, equivalent to Turing machines.

In order to work out the distinguishing characteristics between real and virtual worlds let us look at the game of chess as a prototypical example of a virtual world. Chess is a formal game. There are precisely defined states, namely the board positions. It is also a game with complete information: if you know the board position you know all there is to know about the current game. Given a certain board position, the possible moves are precisely defined. Even though you may not know the particular move the opponent will make, you do know that he will make a legal move, otherwise he would not be playing chess any longer. The game is also static in the sense that if no-one makes a move, nothing changes. Moreover, the types of possible moves, given a particular board position, do not change over time.

Quite in contrast is the real world. There are no "states" or precisely defined situations. What is the "state" of a soccer game? Perhaps we want to say it is the current score, but that would be only a very crude characterization of the game. The interesting part is the physical manipulation of the ball, an aspect that is not captured by the score. One can make a model of the real world, and that model may have states, but the real world as such does not. The real world is also indefinitely rich. Thus, the available information is always "incomplete." In fact, it is not even defined what "complete" information would mean. Expressed differently, the real world is only partially knowable. This implies that it is only partially predictable. Moreover, it is not defined what the possible moves are. In addition, the real world changes continuously, even if you don't do anything: other people move around, there is wind, light conditions change, leaves fall from the trees, etc.

One of the reasons why this distinction between real and virtual worlds is hard to see in traditional computer applications is the fact that there is always a human in the loop. And it is this human who does all the work of relating to the real world. The human acts like a kind of "bio-buffer" which is the reason why

even many poorly designed computer application still work—poor meaning that they do not automate the right sorts of tasks.

And here is where autonomous mobile robots, or autonomous agents, as we will call them, come in. Anyone who has worked with them immediately realizes that real and virtual worlds are not only different worlds, but different universes. The two also require completely different ways of working and programming. We feel that by teaching real-world autonomous agents we can achieve several goals. On the one hand, we can promote an understanding of the distinction between real worlds and virtual worlds, and elucidate some of the fundamental problems in developing computer applications in general. More specifically, we hope to further the understanding of embedded systems. Embedded systems will become ever more important: through miniaturization virtually all consumer products will be equipped with microprocessors. Developing software for embedded systems requires different skills and a different kind of thinking.

On the other hand, we can acquire a different and deeper understanding of the notion of intelligence, since intelligence always has to manifest itself in the real world. Real-world autonomous agents, by definition, act in the real world. Their application forces the student of intelligence to deal with these issues.

We will proceed as follows. First, we will give an overview of the fields of computer science, as they relate to this paper. Then we will start our discussion of how we use real-world autonomous agents for teaching. Since our background is in AI and cognitive science, we will start from there and discuss what we can learn from real-world autonomous agents about intelligence. We will discuss the contributions to various disciplines. Concretely we discuss a class, entitled "New Artificial Intelligence", where we have strongly included autonomous agents for classwork exercises. Then we present three case studies illustrating what we mean by "powerful ideas" and we point out the benefits of autonomous agents for teaching an interdisciplinary audience. We will then briefly introduce the Didabots, the platform we have been mainly using for classwork exercises. We then move to computer science and discuss what we think computer science students could learn from autonomous agents. As we will see, it is a fundamentally different way of viewing the world and of thinking about computer science. We conclude by discussing the "Zen of robot programming."

2. COMPUTER SCIENCE

Figure 1 depicts a certain way of carving up the field of computer science that we will use for our discussion throughout the paper. There is a division between core and applied computer science. Core computer science deals with the foundations, with computers themselves, operating systems, compilers, etc. The basis of core computer science is, in essence, the virtual machine. There is no or only little need to worry about the outside world. The virtual machine is also at the center of theoretical computer science. By contrast, applied computer science deals with applications, with putting computers to work in the real world. Software engineering,

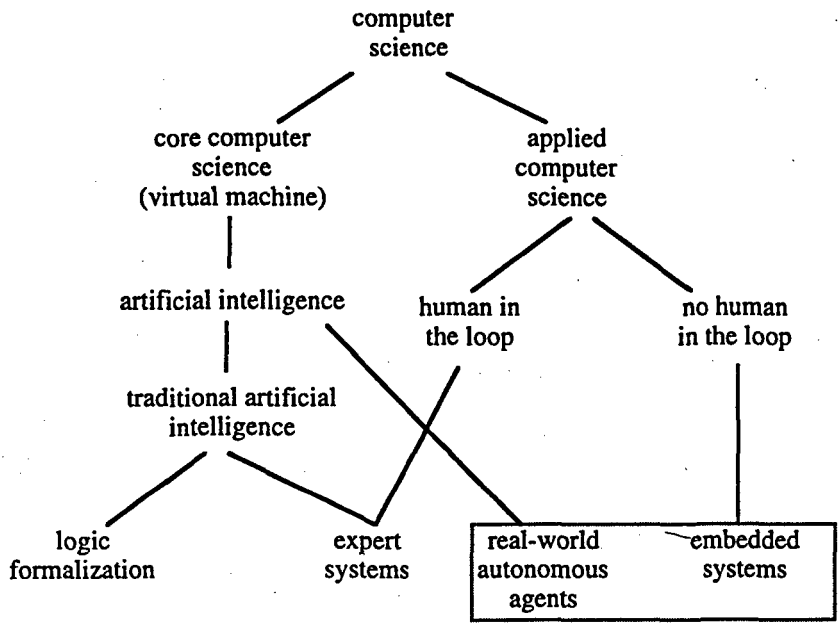

Figure 1. Categorization of the field of computer science.

applications development, human factors engineering, work and organizational psychology are the disciplines concerned with applications.

Applications can be further subdivided into applications with a human in the loop and those without a human in the loop. Of course, ultimately, in all applications there will at some point be a human, otherwise, why develop the application in the first place? But process control applications, for example, do not require human intervention during routine operation. The kinds of systems with no human in the loop, i.e. the systems interacting directly with the real world, are sometimes called *embedded systems*. While currently most applications still have a human in the loop, embedded applications are rapidly making their way into the large markets.

On the left in figure 1 we put traditional artificial intelligence, also called GOFAI, for "good old-fashioned artificial intelligence" [1]. Only recently it became clear that one of the fundamental problems of GOFAI was its focus on formal tasks such as chess, theorem proving, cryptarithmetic puzzles, Tower of Hanoi, or missionaries and cannibals. Formal tasks are, of course, optimally suited for information processing, for the virtual world. As soon as AI moved to the real world, as exemplified by the field of expert systems, serious problems emerged. Eventually, towards the end of the eighties, it became clear that the problems were insurmountable. But the reasons for the failure of expert systems were not obvious. They remained hidden for a long time because the standard expert system always had a human interpreting the results.

In parallel to the field of expert systems a new approach evolved in AI, namely the one of real-world autonomous agents. Rodney Brooks of the MIT Artificial

Intelligence Laboratory argued that intelligence is not so much a matter of internal representations and information processing, but an emergent property of the interaction of a physically embodied agent with the real world [2]. This approach, called "behavior-based robotics", "New Artificial Intelligence", or "autonomous agents", represents a radical departure from the classical approach. It requires a completely different way of thinking. It is the spirit of "New AI" that we would like to teach to students of various disciplines.

Traditional AI has strongly moved into logic, and much of the work has become very formal. This is suited for computer programming, but does no longer contribute much to our understanding of intelligence.

In figure 1 real-world autonomous agents and embedded systems end up next to each other. Although the goals of the two fields are very different, there are many similarities. We have discussed them in the context of computer science. But we would like to stress that the productive and creative potential of the two fields can only be exploited if they are not merely considered to be an extension of the virtual world of computer science; we have to let the real world intrude into our thinking.

3. REAL-WORLD AUTONOMOUS AGENTS AND ROBOTICS

The title of this Special Issue is "Robotics in Computer Science and Engineering." Robotics is a very large field. In this paper, we focus on real-world autonomous agents for a number of reasons. One of the main areas of robotics is, of course, industrial robotics. Industrial robots are physical systems, they manipulate objects in the real world, they are subject to forces, to wear, they can break, etc. The main goal of industrial robotics is to make efficient, precise, and fast machines that can be easily controlled. In the field of real-world autonomous agents, the main goal is to understand intelligence, which is entirely different. In the former case, the goal is, in a sense, to engineer the robots in a way that the programmer can neglect the problems of the real world and write his programs as if he were dealing with a virtual world. By contrast, in the latter case the goal is adaptivity. Autonomy, i.e., independence of human control, is an important characteristic of intelligent systems, but it is to be avoided by all means in an industrial context for obvious reasons. Because of the different goals and the different basic philosophies, there is relatively little overlap between the two fields. Both are relevant. We will only discuss real-world autonomous agents. We feel we can learn more from them since the underlying ideas differ more radically from classical computer science.

4. TEACHING REAL-WORLD AUTONOMOUS AGENTS TO AN INTERDISCIPLINARY AUDIENCE

The experiences that we report here are from a class entitled "New Artificial Intelligence" (for details, see [3]). It is a strongly interdisciplinary undergraduate course. There are computer scientists, psychologists, biologists, neurobiologists, mechanical engineers, and philosophers. It has grown out of roughly ten years of

teaching various fields of artificial intelligence and cognitive science such as expert systems, psychological modeling, neural networks, foundations of AI, and cognitive science.

The goal of the class is to encourage students to think differently about issues in the study of intelligence. The question about the nature of intelligence, is replaced by the following one, which is much more productive: "Given a certain kind of behavior that we are interested in, what are the underlying mechanisms that are responsible for it?" The question we have to answer, is what we mean by "underlying mechanisms."

The response depends on the kinds of explanations we are interested in. At least three levels can be distinguished, namely functional, learning or developmental, and evolutionary. Functional means that we are trying to relate the current internal state to the behavior, learning or developmental that we are giving explanations where we include some of the history of the individual, and evolutionary that we try to understand how the individual has evolved over many generations. For the purpose of this paper let us focus on the first two, functional and learning or development.

In order to illustrate what we mean by "powerful ideas" (see title) let us look at three case studies, Braitenberg vehicles, Didabots cleaning up an arena, and garbage collecting robots.

4.1 Case Study 1: Braitenberg Vehicles

Assuming that most people are familiar with the basic idea of Braitenberg vehicles, we only give a very short introduction and discuss one type of vehicle. The reader interested in more detail is referred to the excellent little book by Braitenberg [4].

Let us look at the vehicle shown in figure 2. It is called of type 3c. There are two types of sensors, light sensors and proximity sensors. Light sensors signal high activation when it is bright; proximity sensors signal high activation when they are close to an object. The sensors are connected to the two motors in a contralateral way as shown in the figure. The light sensors have positive connections meaning that the brighter it is, the more the respective motor is driven. If the light source is to the left, the left light sensor will get more activation; thus, the right wheel is driven more strongly and the vehicle will turn towards the light source. The contrary is the case for the proximity sensor. If the object (the light source in our example) is to the left, the left proximity sensor will get more activation, meaning that the right motor will be slowed down because of the negative connection. The vehicle will turn away from the light source. The behavior of the vehicle is determined by the quantitative relationships of the various strengths of the connections and intensity signals of the sensors.

There is another consequence of this particular wiring scheme for the behavior of the vehicle. The brighter it is the faster it will go. In the dark it will go slowly. We might be inclined to say that it prefers the dark and dislikes the light. Note that when we say this, it is in our own heads. There is no representation of preferences

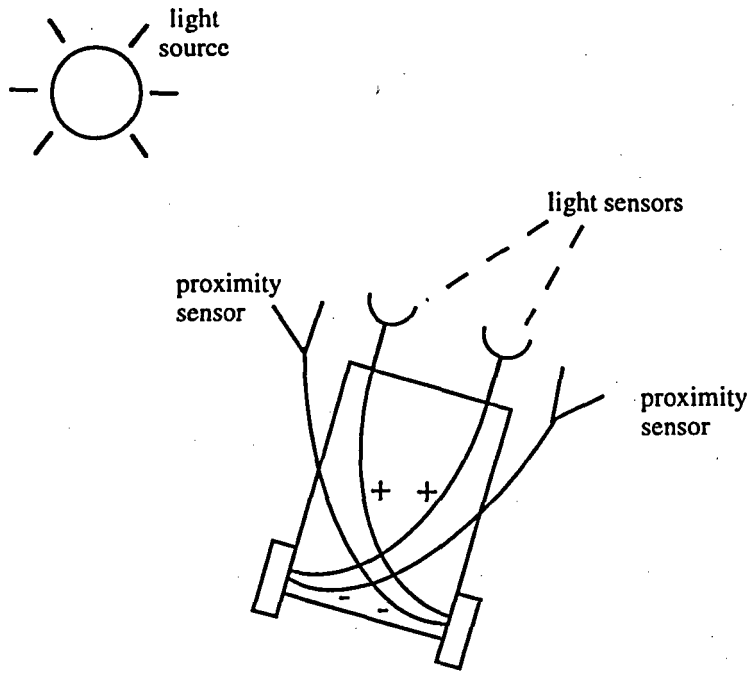

Figure 2. A Braitenberg vehicle of type 3c.

within the vehicle whatsoever. This also implies that we might just as well give a different interpretation.

Try to make exact predictions about what the vehicle will do. You see immediately that this is not possible—it depends on so many factors. Imagine now, that there are other vehicles of the same kind present. Imagine further that they have a light source on top of themselves. And try now to predict the behavior of several vehicles. Impossible!

Originally Braitenberg vehicles were meant to be thought experiments. In the meantime they have been simulated by many people, and many have been implemented in real robot hardware. They are very easy to implement on the Didabots (see below). Their infrared (IR) sensors can be used to measure proximity as well as ambient (normal) light. IR sensors send out an IR signal and measure the reflected signal. Thus IRs actually measure intensity of reflection rather than proximity. The reflection depends on the type of surface and its color. Moreover, on the Didabots, the sensors cannot distinguish their own signals from those of the other sensors or other robots for that matter. So we see, that in the real world, we typically have no neat, precise distance measure, but rather a messy, unsystematic, extremely noisy signal. Under these circumstances it is even more difficult to predict what the vehicles will do.

What can we learn from Braitenberg vehicles? We have summarized a number of important points that we should remember.

- Behavior has to be distinguished from internal mechanisms. This is not so obvious for a computer scientist, since most of the time there is only a computer program and no behaving organism.
- Attributions by an observer have to be clearly distinguished from the mechanisms responsible for the behavior.
- Seemingly sophisticated behaviors can result from very simple mechanisms, which is due to the interaction of the agent with its environment. The sophistication could not be understood by looking at the internal mechanisms only.
- The mechanisms implemented on Braitenberg vehicles are entirely deterministic. Nevertheless, it is virtually impossible to predict their behavior. This property of unpredictability is typical for systems displaying chaotic behavior.
- The signals that we get from the sensors in the real world are often very noisy and correlate only to a certain degree with the quantity we want to measure. An example is the use of IR sensors to measure proximity.

4.2 Case Study 2: Didabots are Cleaning Up

Look at figure 3. There is an arena with a number of Didabots. The arena is initially cluttered with randomly distributed styphor cubes of roughly $10cm^3$. The Didabots are programmed as simple Braitenberg vehicles with only one type of sensor for proximity. All they can do is avoid obstacles. Now look at the sequence of pictures shown in figure 4.

Initially the cubes are randomly distributed. Over time a number of clusters are forming. At the end there are only two clusters and a number of cubes along the walls of the arena. We ran this experiments many times. The result is absolutely consistent. What would you say the robots are doing? "They are cleaning up." "They are trying to get the cubes into clusters." "They are making free space." These are answers that we often hear. These answers are fine if we are aware of the fact that they represent an observer perspective. They describe the behavior. The second one also attributes an intention by using the word "trying." Because we are the designers, we can say very clearly what the robots were programmed to do: to avoid obstacles!

The Didabots only use the two sensors which are marked in black, namely front left and front right. Normally they move forward. If they get near an obstacle within reach of one of the sensors (about 20cm) they simply turn toward the other side. If they encounter a cube head on, neither the left nor the right sensor measure any reflection and the Didabot simply continues moving forward. At the same time it pushes the cube. But it pushes the cube because it doesn't "see" it, not because it was programmed to push it.

For how long does it push the cube? Until the cube either moves to the side and the Didabot loses it, or until it encounters another cube to the left or the right. It then turns away, thus leaving both cubes together. Now there are already two cubes together, and the chance that another cube will be deposited near them is

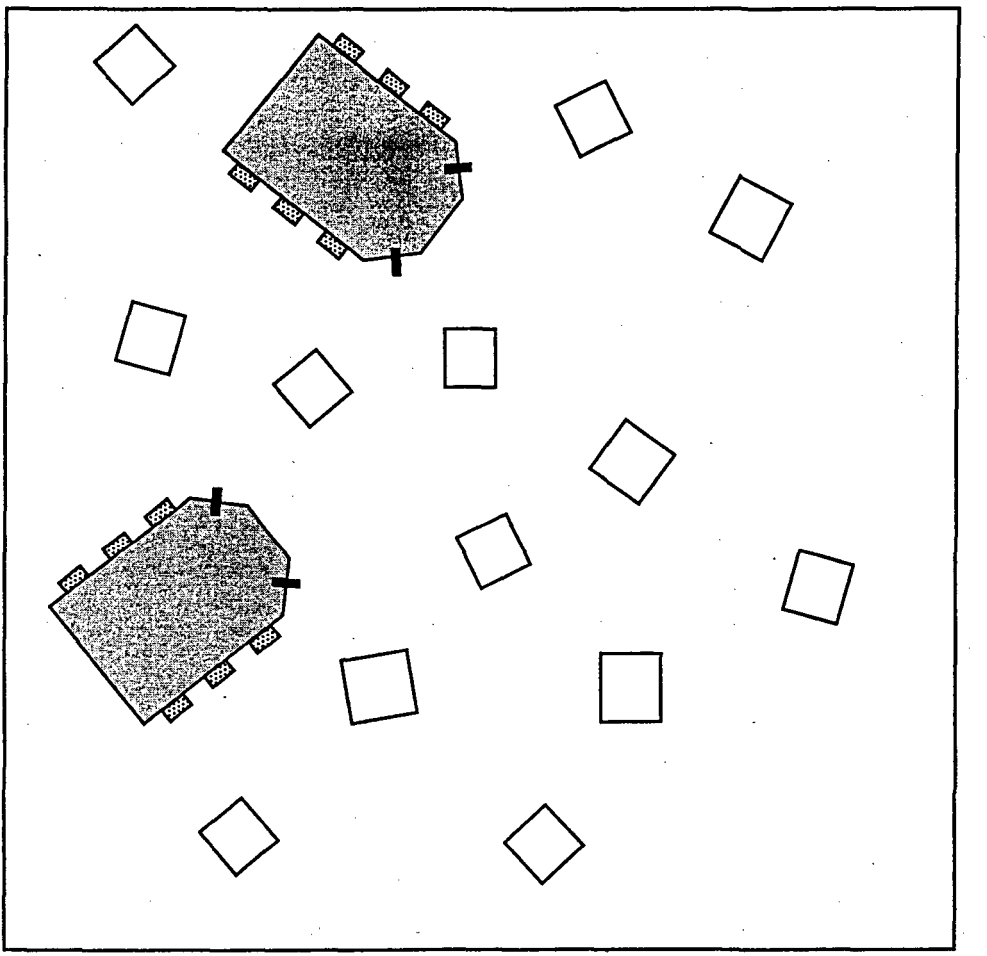

Figure 3. Arena for the Didabots.

increased. Thus, the robots have changed their environment which in turn influences their behavior.

While it is not possible to predict where exactly the clusters will be formed, we can predict with high certainty that only a few clusters will be formed in environments with the geometrical proportions used in the experiment (for details, see [5]). So, we can make predictions, but they are of a different nature than what we are normally used to in physics.

The kind of phenomenon that we have seen in this experiment is also called self-organization. The behavior of the individual leads to a global change, namely

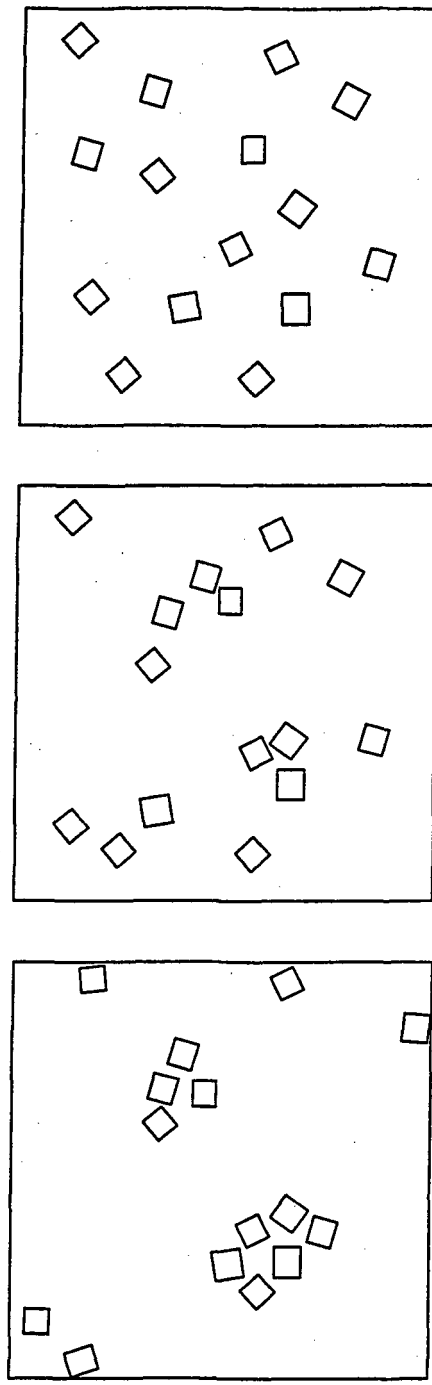

Figure 4. Sequence of situations after a few minutes each as the Didabots are put to work.[2]

the arrangement of the cubes, which in turn influences the behavior of the individuals. As is well-known, self-organization is a ubiquitous phenomenon in the world around us, in biological, social, economic, engineering, and inanimate physical systems. In computer science the concept of self-organization is to date virtually non-existent. We will show later, that at least in some areas of computer science, the notion of self-organization could be usefully employed.

Given the right conditions and geometrical proportions, this phenomenon of cluster formation is stable and consistently occurs. Therefore, we can in fact design a cleaning robot without explicitly programming a representation of cleaning into the robots. The task of cleaning is in our minds as designers, it does not have to be in the "minds" of the robots. To use another buzzword, the behavior of such an agent is sometimes called *emergent*. Emergence can be utilized for design [6]. What can we learn from our cleaning robots? Here are a number of points:

- What agents do, i.e. how they behave, cannot be directly derived from the program alone. Metaphorically speaking, we could say that what they are doing, namely cleaning up, is not what they "think" they are doing, namely avoiding obstacles.
- More generally speaking, we can say that behavior is by definition happening in the interaction of an agent with its environment. It cannot be reduced to internal mechanism. Although this is often done, reducing behavior to mechanism constitutes a category error.
- Self-organization implies that the behavior of the individuals leads to global patterns which in turn influence the behavior of the individuals. This principle is very common. It can be exploited for engineering purposes.
- If agents interact with a real physical environment, the results are sometimes surprising and very different from what one would expect based on an analysis of the internal mechanisms alone.

4.3 Case Study 3: A Garbage Collecting Robot

4.3.1 Introduction. Assume that we want to build a robot for collecting garbage. The first thing is to define the sorts of things it should be able to do and the environment in which it is to function. The specification of task and environment defines the agent's ecological niche. The ecological niche also includes other agents which typically compete for the same resources. Every real-world agent is made for a particular ecological niche. There is no universal robot. If the robot functions and competes well in one type of environment, it may not do so in others. A robot on wheels might perform very well in well-paved environments, but it will not be effective in environments with stairs and on rough terrain.

A garbage-collecting robot has to be able to pick up garbage and bring it either to a garbage truck or to an incinerator plant. Picking up garbage implies that the agent somehow has to be able to distinguish between garbage and non-garbage. If the robot is to be used in different cities, where the garbage bags are of different color and shape, and if we want our robot to be used in several cities, it may be a good idea to equip the robot with learning capabilities.

Our garbage-collecting robot, like any mobile robot in the real world, always has several things to do. If it is to collect garbage it has to worry about not bumping into things, not being run over by cars, not getting damaged, finding garbage it should collect, finding its way back to the garbage truck or incinerator plant, etc. Thus, the robot will at any point in time have to decide what to do.

4.3.2 The experimental set-up. In order to come to grips with the extremely complex task of collecting garbage, we have started with a simplified version. Figure 5 shows an arena for a simple garbage collecting robot. The robot should learn to distinguish the small objects from the large ones. The reason it should learn the distinction is that it should be able to perform its task in environments where the objects can be of different sizes and shapes. Moreover, we want the robot to form its own categories in order to guarantee that they suit its own needs. This is done by means of a value system. Whenever we have a learning agent, and we do not want to supervise all of its steps, we have to equip it with a so-called value system. We have to define what we think is "good" for the robot. If the agent does something right the behavior should be reinforced. For example, if an animal eats the right kind of food, a reinforcement signal should be generated. If our garbage col-

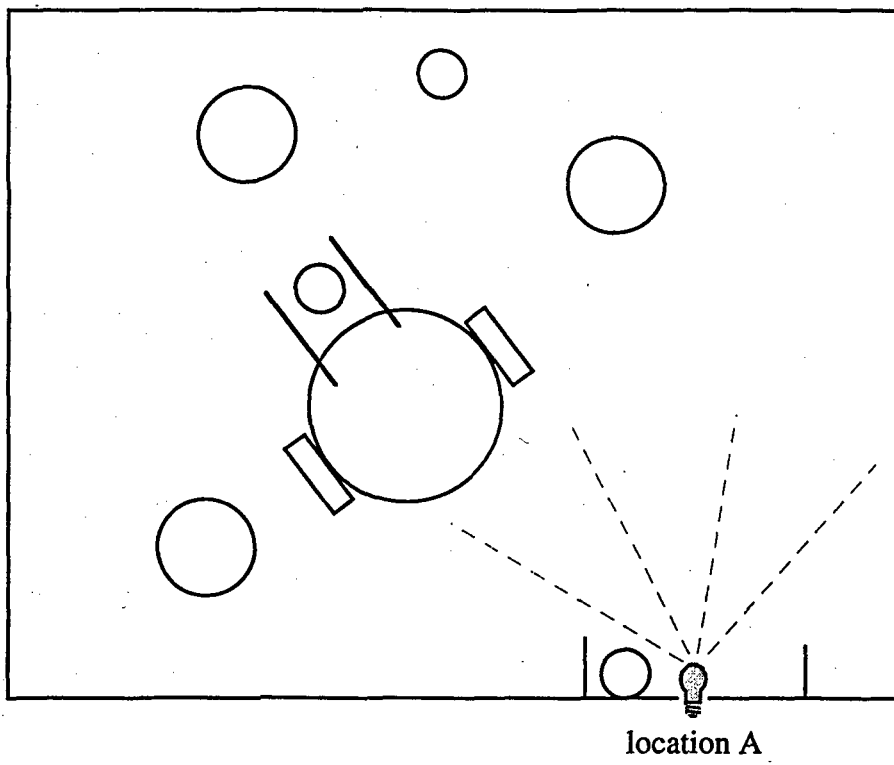

Figure 5. Simplified environment for the garbage-collecting robot.[3]

lecting robot manages to pick up an object, it has done something right: a reinforcement signal is generated whenever the agent successfully picks up an object. This kind of learning is called self-supervised, meaning that learning is guided by the agent's own value system.

4.3.3 Learning and categorization—definition. First, let us define what we mean by learning and categorization. We want to define it in purely behavioral terms. If an agent consistently picks up small objects but ignores large ones, we say that it has two categories at its disposition. If initially the robot reacts similarly to small and large objects, but after some time picks up only small ones and ignores large ones, we say that it has learned to categorize small and large objects. Note that we are talking about the robot's behavior, not about its internal mechanisms.

There are many complex issues involved in this kind of self-supervised learning. We also talk about situated learning, because the agent learns to acquire categories from its very own perspective. We only point out some aspects that are particularly relevant to this paper. The reader interested in more detail is referred to [7] and [8].

Perception in mobile robots: One of the fundamental problems of mobile robots is that the sensory stimulation of one and the same object changes dramatically, depending on orientation and distance. Traditional computer vision has tried to solve this problem by means of highly sophisticated algorithms for mapping sensory stimulation onto some kind of internal representation. Perception, in this approach, is seen as a problem of information processing. But real-world agents cannot only process information, they can move around and act upon the world.

Our agents have been designed such that as they encounter an object they start exploring it by moving along it. This exploratory process leads to additional sensory stimulation. What had to be done entirely by means of algorithms in classical computer vision has been partly replaced by an action in the real world. Thus, the increased difficulties with perception introduced by mobility (i.e. the strongly varying stimulation of the sensors from one and the same object), are compensated by the ability of the robots to move around and directly affect their environment. This is an option no computer program has. It turns out that perception and thus category learning is made much simpler in this way. This requires some explanation.

In addition to the IR-sensors, the robot has sensors to measure how fast each wheel is turning, the so-called wheel encoders. In a sense, they provide information about what the agent is *doing*, rather than what it is "seeing." As the agent moves along an object, there will be stable activation in a lateral IR sensor (i.e. an IR sensor on the left or on the right). At the same time—if the object is more or less round—the angular velocity will be constant, i.e. the motor speeds will have a constant ratio (ignoring noise). These angular velocities can be used to distinguish objects: large angular velocities correspond to small objects and vice versa. We say that objects can be distinguished on the basis of a sensory-motor coordination rather than sensory stimulation alone. In other words, the object distinctions depend on what the agent is doing, not only on what it is "seeing"!

If the agent moves around in the open, it will get quickly varying stimulation of the sensors by the objects and it won't be able to perceive much. The sensory-motor coordination is necessary.

4.3.4 Architecture. The architecture we have developed is shown in figure 6. It consists of a number of parallel processes which more or less directly connect the sensors to the motors, without any substantial amount of central processing in between. This represents a radical departure from traditional thinking where central integration is the main issue. The idea of this architecture is that intelligent behavior is viewed as being emergent from a large number of parallel processes, rather than being based on some central controller or *homunculus.* This architecture is a natural extension of what we have seen in Braitenberg vehicles.

Again, the details need not concern us here. The important idea is having many parallel, largely peripheral processes which are only loosely coupled. It turns out that many problems are taken care of by the physics of the agent itself. If, for example one process "tells" the motor to go forward, another to go backwards by the same amount, the agent will simply stand still. No arbitration scheme is

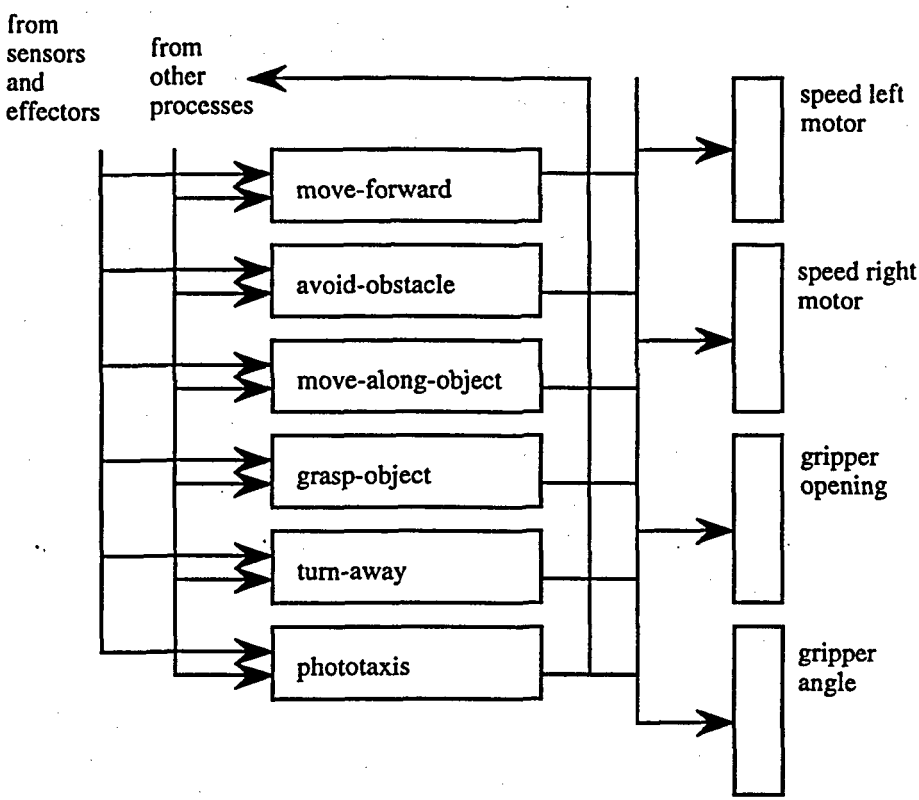

Figure 6. Control achitecture of the robot.[4]

required for this case—the situation does not have to be excluded. A deadlock will not occur because: one, there are other processes that change in intensity over time and continuously influence the motors, and two, the environment changes and thus the influence of the various processes on the motor variables changes. In summary, there are many factors which influence the behavior of the agent. They will eventually take care of the conflicting situation. There is no need to provide an information process for resolving it.

4.3.4 Learning and categorization—results. The agent has a grasp reflex which is automatically triggered if the agent has been moving along an object for some time. The intuition behind this reflex is that if the agent is near an object, there is some chance that grasping will be successful, so the agent should try. If it is successful, there will be reinforcement, if not, there is no reinforcement. Eventually most of the agent's trials at grasping will be successful.

Table 1 shows what happens over time. The main result is:

- The number of steps the agent has to move along an object before it "recognizes" the appropriate category is significantly reduced.
- After some time, the agent hardly grasps any large objects at all.

Based on these observations, we, the observers, say that the agent has learned to categorize the objects. Note that we are only talking about behavior, not about internal mechanisms. The effect of this categorization is that the behavior of the agent gets more efficient over time.

4.3.5 Learning and categorization—mechanism. In order to achieve this learning behavior we have used neural networks. If we want to understand their dynamics, we have to know how they are embedded into the sensory-motor setup. If we exchange a sensor for a different one, the networks—and therefore the representations that will be built—will be different. In other words, categorization strongly depends on the sensory-motor (physical!) properties of the agent and is not merely a matter of information processing.

What can we learn from case study 3? Here is a list of points.

- In the real world, there is no universality, there is no universal robot. This is quite in contrast to general purpose computers which are universal in the sense that they can perform any type of computation. Robots are always

Table 1
Averaged (±Std.Dev.) number of steps the robot moved along the objects and number of objects grasped.

Performance Measure	*Large object (before)*	*Large object (after)*	*Small object (before)*	*Small object (after)*
Steps	41.3±2.5	14.6±1.6	39.1±2.0	12.3±2.5
Objects grasped	18.2±1.3	1.6±.8	18.7±.6	19.2±5

designed for a particular ecological niche in which they have to compete with other agents, with robots and perhaps with humans.

- In order to achieve learning behavior or coherent behavior in general, it is not necessary to postulate central control or a "homunculus." Rather, coherence and learning are emergent from a (potentially large) number of parallel processes which are only loosely coupled.
- For real world agents it is possible to change the environment, as we have already seen in case study 2. An important kind of change is self-motion. As a consequence of this capability of self-motion, not everything has to be represented internally, but the world can be exploited in the sense of being its own best model: the agent can move up to an object, circle around it, grasp it—in short explore it. This is an option no computer program has.
- The kind of intelligence and the nature of the categories the agent will form strongly depend on the kinds of sensors and effectors. Obviously, if it only has a few IR sensors, as in our case study, the categories will be very different from the ones formed if the agent is equipped with a camera. Thus, intelligence cannot be defined at the level of information processing alone—i.e. it is not merely a matter of programs, but of complete, embodied systems.
- The kind of behavior we have discussed is called *adaptive. Autonomous agents are especially suited to study adaptive behavior.*

4.4 Benefits for an Interdisciplinary Audience

Students of various disciplines have strongly non-overlapping backgrounds. Normally, the different backgrounds make it almost impossible to teach a course. What makes it still possible in our case of the "New AI" class is the robot. First, it is a real behaving system, not merely a simulation, or a model of something else. Although it can be taken as a model, say of an ant, it is a behaving system in its own right. Its behavior can be observed by anyone. Ethologists and psychologists can use their own vocabulary to describe its behavior. And it is always clear what they are talking about. Students of all disciplines have exactly the same knowledge about the agents.

Second, we designed the robot and its control architecture ourselves, so we know about the internal mechanisms. By using visualization tools for internal states we can develop a profound understanding of the dependencies between behavior, physical set-put (morphology, sensors, effectors), and internal mechanisms. This is one of the main goals of a theory of intelligence.

Third, we can always make matters concrete. Especially discussions about intelligence tend to become very vague and personal. "This is not *really* intelligent," is a typical utterance. We can always say something like: this robot behaves in this way (we can observe its behavior), and we find it interesting for some reason. Then we can point to the mechanisms that bring the behavior about. Whether someone might want to call that behavior intelligent, or not, is besides the point.

Fourth, we have a continuous source of ideas and surprises. The robots (particularly mobile ones) force us to ask certain kinds of questions which, we think, are the right ones, i.e. questions that are essential to behavior in the real world and

to intelligence. For example, agents in the real world move around, which leads to hard problems concerning perception (as discussed above). An example of a surprise are the robots cleaning up an arena or the Braitenberg vehicles: very simple mechanisms can lead to seemingly sophisticated behaviors in the real world that are very hard to predict. Robots are performing tasks they are not *programmed* for—but they are *designed* for it.

And fifth, the robots provide an excellent metaphor for advanced applications in computer science, in particular where embedded systems, or systems interacting in some ways with the real world are concerned. More details will be given below.

5. THE DIDABOT PLATFORM

For our "New AI" class we have mainly been working with the Didabots, a simple platform developed in our laboratory geared toward educational purposes. In order to provide some idea of its capabilities, we briefly describe its hardware and software. For some of the experiments we have used the miniature robot Khepera™. It can be bought off the shelf and we will briefly describe it at the end of the chapter.

5.1 Didabot Hardware and Software

Didabots are "didactic robots" that enable the student to make simple experiments and to collect hands-on experience with real robots. They have been built in a few weeks in a computer science laboratory that is not particularly equipped with the standard electrical engineering infrastructure. Figure 7 shows a picture of the Didabot.

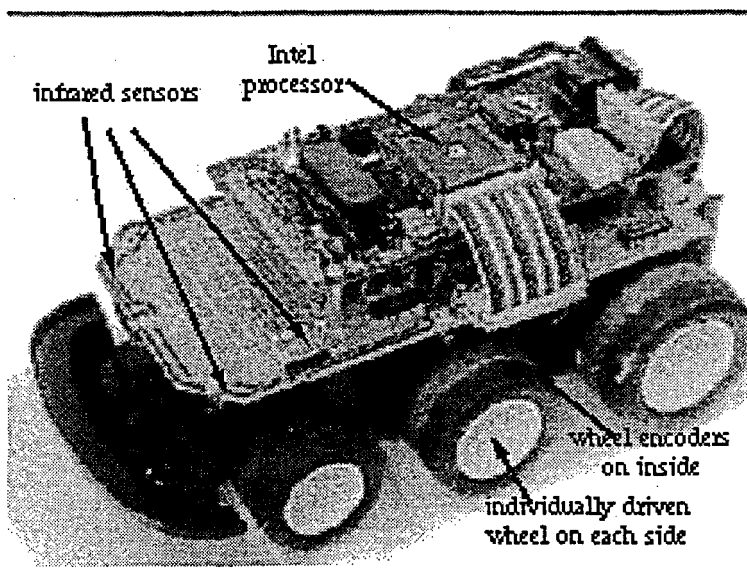

Figure 7. A picture of a Didabot.

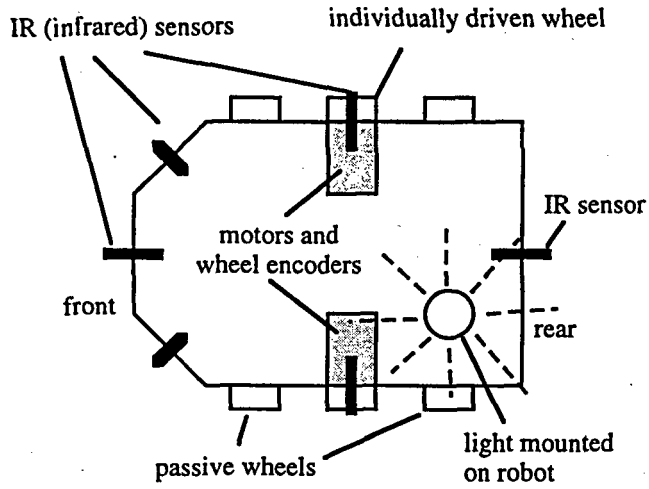

Figure 8. Schematic drawing of a Didabot.[5]

The Didabot platform has been designed by Marinus Maris and René Schaad [9] in our laboratory. The programming environment has been implemented by Daniel Regenass [10]. The chassis has been taken from a standard remote controlled car. The remote control has been removed, a sensory-motor board has been mounted on top, and an Intel microprocessor board provides full autonomy for the robot.

Figure 8 shows a schematic view from the top. As shown in the figure, there are 6 IR sensors. They can also be used for the detection of ambient light, i.e. normal (non-IR) light in the environment. There are two wheel encoders, measuring the speed of rotation of the wheels (a kind of proprioceptive sensor). There are two individually driven wheels which enable the robot to turn on the spot. Also, there is a light source mounted on top of the robot.

The Didabots are ideally suited for experiments such as the ones shown in case studies 1 and 2. To implement the Braitenberg vehicles in case study 1, not only two IR sensors have been used, but the five in the front and on the sides, which gives the robot more smooth behavior. This is done by computing a kind of average over the sensor readings.

The programming environment, DDS (Didabot Development System) [10] enables the users to easily access sensor data, and control the motors. The programs can be written in C on a normal PC, cross-compiled and downloaded onto the Didabots. The Didabots are equipped with a radio modem. It can be used to transmit data about their internal state which is needed to explain their behavior.

5.2 The Miniature Robot Khepera

Figure 9 shows a picture of a Khepera™ robot which is equipped with a gripper. It is only about 5cm in diameter and can be run on a normal office desk. It can be

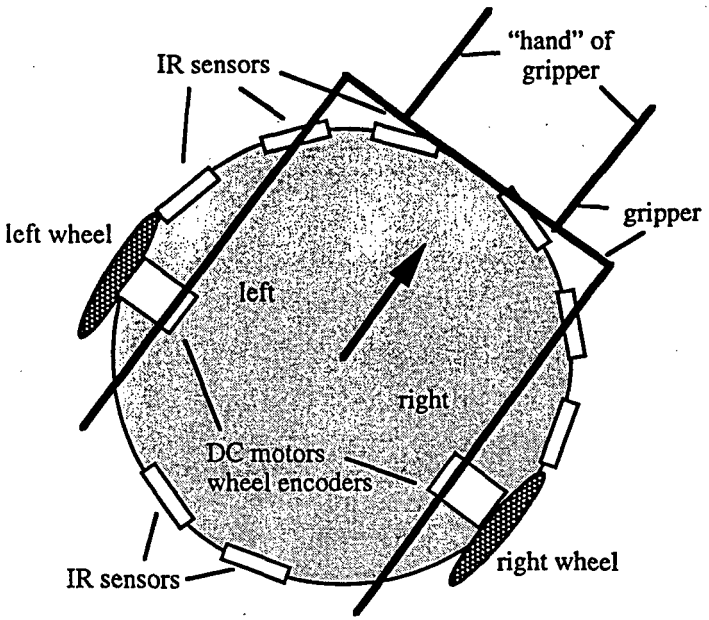

Figure 9. Schematic representation of the miniature robot Khepera™.[6]

operated in a computer science environment in which no technical equipment is available. Conceptually it is very similar to the Didabot. It can be run autonomously from batteries, or from a cable that is connected to a workstation. It comes with a programming environment. The experiments that we described in case study 3 were done with the Khepera™ robot.

6. TEACHING REAL-WORLD AUTONOMOUS AGENTS TO A COMPUTER SCIENCE AUDIENCE

As computer scientists we basically have two options. We can focus on our own territory, and help improve computers in terms of developing better programs. Or we can look towards applications and the effects of computing in the real world. If our sole interest is the former, real-world autonomous agents are not of central interest. If it is the latter, we feel autonomous agents are extremely potent tools not only for artificial intelligence, but for computer science in general. They are effective media for teaching and provide highly productive metaphors. If computer scientists do not want to yield the fascinating and rapidly growing field of embedded systems entirely to the engineers, they had better start worrying about them. Because of their capacity for abstract analytic thinking computer scientists are well-equipped for the task. But they have to open up towards the real world. The Japanese have precisely identified this issue: they set up a large research program

entitled "Real World Computing" (although they may have had something different in mind).

The following ideas are not yet very concrete, not yet at the level where we can exactly say: this is what we have to teach, this is how we should change our software engineering practices, etc. They are more "inspirations", some philosophical, some psychological, some perhaps a bit esoteric. But we feel that we are onto something very important. The importance is also reflected in the fact that there is this special issue of the journal. It is all preliminary, there is no real structure in the arguments. Take it as a collection of thoughts for inspiration.

6.1 Understanding Natural Intelligence and Building Intelligent Machines

The field where researchers are trying to develop intelligent machines is called artificial intelligence. The latter is only a relatively small subfield of computer science. Nevertheless, from its very beginning, computer science has implicitly or explicitly had the goal to make computers more intelligent. We talk about intelligent cameras, intelligent washing machines, intelligent modems, intelligent network communication, etc. As computer scientists, our notion of intelligence is typically restricted to capacity for information processing. As we have seen above in our case studies, intelligence has much to do with interacting with the real world.

Let us take the example of concept acquisition. We have discussed that the robots need to learn from their own perspective and they evolve their own categorization of the world. Another way of saying this is that the robots have acquired a *concept*. To us, as humans, the concepts the robots have acquired may seem primitive (consisting only of "objects to be picked up" and "other things"), but they serve the purpose for the robots: they start collecting the right sorts of objects, namely garbage. If we adopt this perspective, we also come to realize that the world looks very different from the robot's own perspective because its sensors and effectors are so different. If we want to communicate with them, the only "channel" is through their sensory-motor system. If people were more aware of that, they would better realize the limitations in terms of communication. This point has been made very clearly by Lucy Suchman [11] from a sociological perspective. From an autonomous agents perspective, it is quite obvious. The message is that we should not project onto our computer systems properties they don't have and they can't have.

This idea can be generalized. The world of any agent, human, animal, robot, or computer, is limited by its sensory and motor capabilities. Since the concepts an agent can acquire depends strongly on its physical set-up—as we have seen earlier—the nature of intelligence of any robot will also be different from human intelligence. Humans are endowed with very different sensory-motor faculties.

This perspective leads to an entirely different notion of human intelligence and expertise. It becomes immediately clear that we cannot model human expertise by means of information processing only: we must take the system-environment interaction into account which, of course, includes the sensory-motor system of the agent. In the case of humans, this happens to be enormously sophisticated.

The neglect of this system-environment interaction is the deeper reason why expert systems are inappropriate models of human expertise.

Experts are experts not only because of their academic training, but because their knowledge is grounded in their sensory-motor experience with the world. This sensory-motor experience in turn is the foundation for social competence. Based on this view of human expertise we have developed a methodology for designing computer applications, called Situated Design (e.g. [12]). Although this methodology could have been developed without knowing about real-world autonomous agents, the latter has greatly helped our pertinent thinking.

6.2 Behavior and Exploiting the Real World

As we have just seen, human experts *behave*, but so do machines. Behaving means interacting with the real world, i.e. reacting to and affecting the real world. As pointed out in the introduction, in computer applications there is normally a human intermediary to make the connection to the real world. All the computer program does is displaying something on the screen. Thus, behavior is not really an interesting category for standard computer programs. The situation changes radically if the program is to interact directly with the outside world.

The real world is very different from the virtual one. It is there, even if we don't think about it. The styropor cubes are pushed by the Didabots even if they don't "know" about it. This is purely given by the physics and there is no representation of this fact in any of the agents.

Categorization of our garbage collecting robots is another example. In order to "recognize" the objects, the agents have to *re-enact* a sensory-motor sequence. It is sufficient for the agent to engage in this sensory-motor interaction. There is no need for an abstract representation; the interaction with the objects themselves is an integral part of the perceptual process.

An additional illustration is given by the fact that agents stay on the ground because of gravity and they can move forward with wheels because of friction. But neither gravity nor friction have to be represented for the robots to work. In all these cases the robots are, in a sense, exploiting the real world, but without "knowing" it.

But there are even more advantages to being in the real world. Imagine a computer system in which data is lost. If we are within a purely virtual world, the data is gone forever. This is the reason why so much effort is put into making data transfer and storage absolutely reliable. Just think of all the error detection and correction hardware and software. Imagine now that the data item of interest originates from a sensor reading. For one thing, the sensor reading is not arbitrary, the neighboring bits are related to one another, there are correlations between the bits because they originate from the real world. These correlations can be exploited—they represent redundancy. So, typically if we are dealing with real-world data, it is not a disaster if one data element gets lost. There is a good chance that at least part of it can be recovered by taking advantage of the redundancy. Moreover, if a sensor reading is entirely lost, we can simply read the sensor again. If this is done quickly, the sensory reading is very likely to be similar to the previous one. Thus,

our complete system (consisting of the computer plus sensors and effectors) can perhaps be made much cheaper. This idea can be characterized as "The world is its own best model."

It is also powerful as a metaphor. Take a service engineer who has to repair a machine. Obviously, the machine is there even if the expert has no explicit internal model of it. He can look at it, he can perform measurements, he can exchange parts and observe the effect, etc. This is not information processing; this is interaction with the real world. If he doesn't exactly remember a reading on a dial he can look again. And this is precisely what human experts do: they continuously interact with the machine, and exploit it as "its own best model." They look at the machine again and again—there is no need for secure, error-free storage of data in the brain. We have to support this capacity to interact with the environment when designing systems, not automating the information processing aspects of the task.

There is another fascinating thought that emerges out of the autonomous agents perspective. Think of an insect. Insects can walk in coordinated ways. It has been shown that there is no central controller for walking, legs only communicate locally with each other [13]. But how come their walking is still coordinated? The answer is that they do communicate, but via the external world, not via an information process within the agent. As soon as the insect lifts one leg, because of the stiffness of the body, and its weight, the forces on all other legs change instantaneously. This is a physical process, not an information process. It is this kind of communication which is exploited by insects in ingenious ways. It is not only very fast (much faster than neuronal processes), but also cheap: in fact it is free of charge. Where can we use this idea in computer applications?

The answer is not very concrete. It tells us that we always have to look at the entire system which includes the physics of a particular machine, the whole work situation. Here is an interesting example from computer vision. If you have a camera image where it is hard to tell the different objects apart, it might be useful to have a robot arm to simply move into the pile of objects. You change the real world, rather than trying to resolve everything by means of information processing. The task of separating the objects might become trivial.

What can we learn from these examples? On the one hand, we can see that a lot can potentially be gained by exploiting the real world. On the other hand, there is not much systematicity in our discussion. This reflects that fact that our understanding of real-world interactions and of embedded systems is still very restricted. Remember the disaster of the new Denver airport. It was the software of a luggage conveyer system that caused enormous delays in the delivery of the system (e.g. [14]). We can speculate about what went wrong in terms of project organization, management, etc. The fact of the matter is that we do not understand very well how to develop software for systems that directly interact with the real world. Specifications, as we know them from software engineering, typically do not deal with the real world, but focus on the capabilities of the programs, not of the entire system. Real-world autonomous agents can teach us the principles of the relation between physical systems, equipped with sensors and effectors (as a luggage conveyer system), and software.

6.3 Local Interactions and Self-Organization Rather than Central Control

We saw that coherent behavior can come about without centralized control, but rather as a result of a (potentially large) number of parallel processes that only interact weakly with each other. Intelligence is emergent as the agent interacts with its environment. This holds for processes within the agent, as well as between several agents. The Didabots in case study 2 don't send signals directly to each other, but they "communicate" in that they get into each other's ways and that they change the environment. Similarly, ants leave traces of pheromone which in turn changes the behavior of the other agents. Structure emerges—the agents seem to cooperate, but there is no central control. Information processing is no longer sufficient to explain the behavior. Self-organization is a more appropriate term. But do we find self-organization in computer systems and if so, in what form?

As we observed earlier, the concept of self-organization does not exist in core computer science. However, recently there has been a lot of talk about self-organization in the computer science community, namely in the context of the Internet. Let us look at the Internet a bit closer in the framework of what we have said.

Individuals behave in particular ways: they use certain services and they use particular kinds of machines which adhere to certain standards (like IBM PCs with Windows 95). If the number of individuals who use them increases, even more individuals will use them because additional services will be offered for them. This leads to a global phenomenon, namely that particular standards win over others, which in turn influences the behavior of the individuals. We have an example of self-organization. There is an analogy to the Didabots cleaning the arena. Initially it is not clear where the clusters will be formed. This is decided by random factors. But once the "seeds" have been formed, the clusters will continue to grow. Metaphorically speaking, similar fluctuations may be the reason why certain standards win and others lose. Note that there is no central control in the Internet. There is nobody who has some kind of a "master plan" in his head telling the others what to do (perhaps with the exception of Bill Gates).

In the case of the Internet, it is not the computer system itself which is self-organizing, but rather the system plus its about 40 million users. Is it a computer science application? Yes, it most certainly is. If we want to understand its dynamics, information processing concepts are not sufficient. We must resort to principles of self-organization. Although, in principle, we have access to any location on the net, at any one time, we can only "look" at a very limited part of it. Thus, we are dealing with "situated users", i.e. users limited by their sensory and motor capabilities. In spite of the fact that there is no or very little global control, it works extremely well. Moreover, there is a lot of creative potential.

We could speculate that in the Denver airport example, it might have been better to focus on locally self-organizing entities, rather than trying to design a globally optimized system. Exploiting the potential for self-organization is a very powerful metaphor. And this brings us to the last point in this discussion.

6.4 Adaptivity—"Being in the World"—Rather than Exerting Control

The French have an excellent word for the computer, *ordinateur*. Ordinateur comprises two meanings, namely creating order, and giving orders. They are both very characteristic of computer science. It implies control. Control is probably the most essential concept of the field.

Control is also inherent in Western thinking. Classical robotics has the goal to control, say robot arms as much as possible. The Puma™ arm, for example, can simply be controlled by indicating the angles of the joints and the arm will very quickly move into the desired position. The system is extremely efficient and the programmer does not have to worry about anything else. Well, this approach completely ignores that there is the physics. The physics is already taken care of by the control engineers who developed this nearly perfect arm. It seduces robotic engineers to only worry about kinematics, i.e. the geometric problems, rather than the physical ones. But our world is a physical one, not a geometric one. The Puma™ arm is, in a sense, extending the virtual world (where one can simply set the angles) to the real world.

The approach with real-world autonomous agents is very different. The main interest is in adaptivity, how to cope with the real world, how to take advantage of it, how to exploit the physics, how to benefit from the real world's own dynamics, and so on. In short, the approach searches for how we can be in the world, how one can live with it in harmony, rather than fighting against it. This is my own personal interpretation of what Rodney Brooks meant with "the Zen of robot programming." He didn't refer to industrial robotics. Perhaps, at some point, some of these ideas will start influencing the ways we think about computers in general.

7. CONCLUSIONS

Our discussion has been a bit metaphorical, rather than being very concrete. If you look at the performance of what our autonomous agents are currently capable of doing (see our case studies) you may not be terribly impressed. But what you can learn, and the questions that you are asking when you are using them as tools, is what it is all about. I hope that we have succeeded in demonstrating the power of the approach.

NOTES

1. It would be more accurate to say that computers process data, as the old term EDP (electronic data processing) suggests, because the pragmatic aspects are up to the user, not the computer itself. Loosely speaking, the computer does not "know" about the meaning of the data it is manipulating. We still use the term information processing because it has become customary to do so, but we mean data processing.
2. The whole process from the first situation to the last one takes approximately 20m. The last frame shown in the figure is a relatively stable configuration: once it is reached, there is hardly any change at all.

3. There are two different types of objects, small ones which it should pick up and bring to location A, and large ones that should be left alone: they are too heavy to be picked up. For these experiments we have used the miniature robot Khepera™. The details of Khepera™ are shown in figure 9. The robot has a diameter of about 5cm. It has two individually driven wheels, a gripper for picking up small objects, and a number of IR sensors distributed around its periphery. It uses them for obstacle avoidance, for moving along objects, and for detecting ambient light. The light sensors are used to bring the objects to location A. Location A is marked with a light source. The robot also has sensors for measuring how fast its wheels turn, the so-called wheel-encoders.
4. There are many parallel processes that connect the sensors to the effectors (wheel motors, gripper motors) without undergoing central ("high-level") processing in between. The processes receive input from the sensors and from the other processes. The processes are all permanently active and they all continuously influence the motor variables. The outputs of the processes are combined into motor signals which determine the behavior of the robot.
5. The Didabot's length is 23cm. There are 6 IR sensors (shown as thick black lines), one in the front, one in the back, one front left, one front right, and one on each side. IR sensors can also be used for ambient (non-infrared) light detection. The robot has 6 wheels, two of which are individually driven by motors (one on each side). The other wheels are passive. There are two wheel encoders for measuring the effective rate of wheel turning. Also, there is a light source mounted on the robot.
6. Khepera™ was developed at the Swiss Federal Institute of Technology in Lausanne and is now widely used in the autonomous agents community. There are 8 IR sensors, 6 around the front half and two in the back. Each IR sensor consists of an emitter and a receiver. The sensor output is a measure of the intensity of the reflected IR signal. Thus, it yields, very roughly a measure of proximity: the closer, the more reflected IR. The IR sensors are also used to detect normal (non-infrared) light. There are two wheels that are individually driven by DC motors. The wheel encoders measure the rate of rotation of the wheels. In addition to the two wheels there is also a "caster", a third point of support for the robot that simply slides on the ground (not shown in the figure). Finally, there is a gripper that can be used for picking up small objects. There is a sensor in the "hand" of the gripper which can be used to detect whether there is an object between the two "fingers". The robot comes with a programming library for accessing the sensors and controlling the motors.

REFERENCES

[1] J. Haugeland, *Artificial Intelligence — The Very Idea, A Bradford Book* The MIT Press, Cambridge, Mass., 1985.
[2] R.A. Brooks, "Intelligence without representation" *Artificial Intelligence*, Vol. 47, 1991. pp. 139-160.
[3] R. Pfeifer and C. Scheier, *An Introduction to "New Artificial Intelligence,"* Lecture notes, Manuscript in preparation.
[4] V. Braitenberg, *Vehicles: Experiments in Synthetic Psychology*, MIT Press, Cambridge, Mass., 1984.
[5] M. Maris and R. te Boekhorst, "Exploiting Physical Constraints: Heap Formation Through Behavioral Error in a Group of Robots", Proc. IROS'96, IEEE/RSJ International Conference on Intelligent Robots and Systems, in press.
[6] L. Steels, "Towards a Theory of Emergent Functionality". In *From Animals to Animats*, Edited by J.-A. Meyer, and S.W. Wilson, Proc. of the 1st Int. Conference on Simulation of Adaptive Behavior, MIT Press, Cambridge, Mass., 1991, pp. 451-461.

[7] C. Scheier and R. Pfeifer, "Classification as Sensory-Motor Coordination. A case study on autonomous agents", *Proc. of the Third European Conference on Artificial Life*, Springer, Berlin., 1995, pp. 657-667.

[8] R. Pfeifer and C. Scheier, "Sensory-Motor Coordination: the Metaphor and Beyond" *Robotics and Autonomous Systems*, (in press).

[9] M. Maris and R. Schaad, "The Didactic Robots", University of Zurich, Computer Science Department, AI Lab Technical Report, 95.09, 1995.

[10] D. Regenass, "The Didabot Development System", Student Thesis, University of Zurich, Computer Science Department, AI Lab, 1996.

[11] L.A. Suchman, *Plans and Situated Actions—The Problem of Human-Machine Communication*. Cambridge University Press, 1987.

[12] R. Pfeifer and P. Rademakers, "Situated Adaptive Design: A Methodology for Developing Knowledge Systems" *Proc. of the Conference on Distributed Artificial Intelligence*, Munich. Springer, Berlin, 1991.

[13] H. Cruse, "Coordination of Leg Movement in Walking Anamals" *From Animals to Animats* Vol. 1, *Proceedings SAB'90*, Edited by J.A. Meyer, and S.W. Wilson, pp. 105-119.

[14] W. Wayt Gibbs, "Software's chronic crisis." *Scientific American (International Edition)*, Vol. 271, No. 3, Sept. 1994, pp. 72-81.