Classifying Agent Systems

Brian Logan
School of Computer Science, University of Birmingham
Birmingham B15 2TT UK
b.s.logan@cs.bham.ac.uk

Abstract

To select an appropriate tool or tools to build an agent-based system we need to map from features of agent systems to implementation technologies. In this paper we propose a simple scheme for classifying agent systems. Starting from the notion of an agent as a cluster concept, we motivate an approach to classification based on the identification of features of agent systems, and use this to generate a high level taxonomy. We illustrate how the scheme can be applied by means of some simple examples, and argue that our approach can form the first step in developing a methodology for the selection of implementation technologies.

Introduction

The conventional approach to developing an agent system involves defining a theory of the agent(s) which comprise the system, devising an architecture which realises the theory and finally implementing this architecture, possibly using a toolkit which supports the theory or architecture (see, for example, (Fisher *et al.* 1997)). In this view, agent theories are essentially formal specifications of the behaviour of an agent system in terms of the properties of an agent and how to represent and reason about these properties. The agent architecture is a specification of a set of software and/or hardware components which together satisfy the properties specified by the agent theory. The architecture is in turn implemented in real hardware and/or software developed using languages which may directly embody principles of the theory or tool(kit)s which provide direct support for the architecture (Wooldridge & Jennings 1995).

While it has much to commend it, this approach is often difficult or impossible to apply in practice, either because the theory makes assumptions which cannot be met by any finite agent (e.g., logical omniscience) or no theory exists for the domain in question, or our aim in building the agent system is to inform the process of theory construction. A more realistic scenario would begin with a developer who wishes to build an agent-based system to meet a particular need (e.g., an air traffic control system or a meeting scheduling system), or a researcher who wishes to explore the implications of different architectures, e.g., (Sloman & Logan 1998). The developer or researcher is often presented with a range of different tool(kit)s and must choose the most appropriate for the current problem or decide that no appropriate tool exists and a new one must be developed.

At present, there is very little guidance on how to go about choosing a toolkit. Usually we can't even determine what is possible in principle with a given toolkit. The theory which the toolkit actually implements is rarely made explicit, and even if the theory were available, it may offer few useful generalisations or be at too low a level to be useful. What *can* be developed depends largely on the ingenuity of the programmer. In any event, such 'in principle' questions are of limited relevance to the development of practical systems. What is required is a heuristic mapping from features of the agent system to be developed to toolkits, which suggests tools that are likely to be appropriate to a given development task. At the moment, we have no normative theory of how to choose a good toolkit for a task. We do however have a growing body of published and anecdotal experience of successful and not so successful agent systems developed with different tools. Unfortunately it is hard to draw on this experience when faced with a new problem, unless the system being developed is similar *in all relevant respects* to one which has already been built. Moreover the fact that one can build a system with characteristics X using tool(s) Y doesn't mean that it is a good idea to do so; there may be other tools, Z, which are better suited to the problem.

Such a mapping presupposes a classification of agent systems. Without a stable, task-neutral classification of agent systems, we can't state empirical generalisations of what works and what doesn't. Such generalisations would be useful in their own right and as a first step in understanding why a toolkit works for a particular problem.

In this paper we propose a simple scheme for classifying agent systems. As such, the paper makes no falsifiable claims. Rather we propose an approach to classification which we believe may be useful in comparing and evaluating agent systems.[1] Its ultimate utility will depend on whether it

[1] Our aim is not to compare two agents with, e.g., the same architecture but differing action selection functions or goal utilities. Although difficult, this is conceptually unproblematic. Rather we address the problem of how to develop an abstract characterisation of an agent system capable of supporting general statements about and comparisons between classes of systems and as an aid in identifying appropriate implementation strategies. Such a classification scheme can't be used to decide if one agent system is *better* than another; at best we can say that one agent A subsumes another

captures at least some of the important ways in which agent systems can differ, and whether the resulting classifications can serve as a basis for empirical generalisations. In the next section, we briefly discuss previous attempts to define 'agent', and suggest that it may be more fruitful to view the notion of agent as a cluster concept. In section 3, we use this view to motivate an approach to classification based on the identification of features of agent systems, and outline a high level taxonomy based on these features. In section 4, we illustrate how the scheme can be applied by means of some simple examples. In the conclusion, we identify some open problems and discuss some issues in the design of toolkits.

'Agent' as a cluster concept

Despite a number of attempts, there is no widely agreed definition of what constitutes an agent (see, for example, (Wooldridge & Jennings 1995; Nwana 1996; Franklin & Graesser 1997)). It has been argued that this is not necessarily a problem: if many people are successfully developing and implementing interesting and useful applications, then it does not matter whether they agree on terminological details (Wooldridge & Jennings 1995). However, if we are to compare agent systems, we need some means of characterising them.

Wooldridge and Jennings (Wooldridge & Jennings 1995) identify four key characteristics of a hardware or software based system which together define a 'weak notion of agency':

autonomy: agents operate without the direct intervention of humans or others, and have some kind of control over their actions and internal state;

social ability: agents interact with other agents (and possibly humans) via some kind of agent-communication language;

reactivity: agents perceive their environment, (which may be the physical world, a user via a graphical user interface, a collection of other agents, the INTERNET, or perhaps all of these combined), and respond in a timely fashion to changes that occur in it;

pro-activeness: agents do not simply act in response to their environment, they are able to exhibit goal-directed behaviour by taking the initiative.

In this view, an agent is a something like a UNIX software process, i.e., a self-contained, concurrently executing thread of control that encapsulates some state and communicates with other agents via some sort of message passing (Wooldridge & Jennings 1995).

However, this similarity with existing software notions has led some to claim that this definition does not distinguish an interesting *new* class of system. For example, Franklin has argued that Wooldridge and Jennings' 'weak notion of agency' could apply to "a payroll program with a graphical interface and a decidedly primitive communication language" (Franklin & Graesser 1997). Franklin goes on to propose an alternative definition based on the notion of an 'autonomous agent':

> "An autonomous agent is a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future."

Further restrictions are then added to give a hierarchical classification based on set inclusion which defines more particular classes of systems, for example 'mobile agent', 'learning agent', 'mobile, learning agent' and so on. However Franklin's definition itself is also open to criticism; for example, it could be applied to a payroll program which pays bonuses or commission on the basis of profits or sales.

Other approaches which are frequently adopted are to characterise agent systems using features of their implementation or the tasks they are intended to perform. However, this makes it difficult to identify useful generalisations which apply to agents implemented using different tools, or which perform tasks which are superficially different but share important characteristics.

In this paper we take the view that there is no single definition of 'agent'; the metaphor has several different senses which are appropriate to varying degrees in different systems. Rather we view agent as a *cluster concept*. By a cluster concept we mean a concept C for which there is a some (not necessarily well defined) set of features or properties $F_1, F_2, \ldots, F_n$ such that possession of various subsets of those features is taken as adequate justification for ascribing the concept C to an individual, without there being any precise specification of *which* combinations of the features are necessary and which are sufficient, i.e., C is not equivalent to some disjunction of conjunctions of the features. Hence indeterminate cases can arise (Sloman 1995).

We therefore adopt a different approach, which is to devise a series of more or less orthogonal properties or 'dimensions' which can be used to describe an agent system in a systematic way . The dimensions could be ratio scales, or ordinal scales, but in the simplest case they are binary predicates which are either true or false of the system. This is the technique of conceptual analysis familiar from philosophy. It is similar in spirit to the approach advocated by Franklin, except that we dispense with the notion of 'autonomous agent' *per se* and instead simply attempt to characterise software and hardware systems using a number of attributes which are associated with the notion of an agent. It would be an error to assume that any subset of these dimensions or positions on these dimensions 'really' defines an agent, though some regions of the space may be more interesting than others. All of the dimensions are contingent properties of an agent system and there is no subset of properties which are either necessary or sufficient for a system to be an agent.[2] In principle, these dimensions are applicable to all software and hardware systems including payroll programs, word processors etc. The fact that such systems *can* be viewed as agent-like in some sense is not a flaw, they

agent B if the features of A are a superset of the features of B.

[2] Although the list of properties outlined in the following section covers many of the features typically associated with intelligent agents in the agent literature, it is entirely conceivable that a system could have none of these properties and still be an agent system.

are just not very interesting cases.[3] Which dimensions are relevant depends on which aspects of agency we are interested in. In the next section we outline a set of dimensions which we have found useful in characterising intelligent autonomous agents.

Classification dimensions

In this section, we identify a number of properties or attributes which attempt to capture the main dimensions of variation in agent systems. The properties are grouped into four main categories: the features of the environment in which the agent system is embedded, the types of actions it can perform and the kinds of goals and beliefs that can be attributed to it. Each property defines a broad class of system, e.g., the class of systems which are mobile or which are capable of autonomous goal generation, and can be decomposed into sub-categories to allow finer distinctions to be made. (In a number of cases we have indicated how the classification could be refined to allow greater discrimination between cases.) In so far as possible, we have attempted to define the properties such that the dimensions of variation are orthogonal, i.e., the fact that one property is true of an agent system should not imply that any other property is true of the system.

In an attempt to avoid conflating the properties of the agent's 'environment' and its architecture, the classification distinguishes between external and internal views of an agent system: discussion of the environment requires no knowledge of the agent or its capabilities; discussion of actions is relative to a particular agent but requires no knowledge of the internals of the agent; goals are partly internal, and representations are wholly internal (unless manifest in messages etc.). Different viewpoints may be appropriate depending on the purposes of the analysis or the amount we know about the agent system.

We shall use the term 'agent system' to refer to the system being analysed, as this allows us to finesse the distinction between an agent and a multi-agent system. An agent system consists of one or more agents. An agent is either a primitive agent (for the purposes of the analysis), or an agent system. A primitive agent is not further defined; in principle it could be anything (see, for example, (Luck & d'Inverno 1995)), but more usually it is some sort of software or hardware system. An agent system therefore includes the usual sense of a multi-agent system in which the agents cooperate to achieve a common goal or goals, and e.g., an artificial life environment, in which the agents compete with one another.

In what follows, we develop a classification of agent systems. For example we consider the environment of the agent system as a whole rather than the environment(s) of the agents which comprise the system (if there is more than one), though each of these can be viewed as an agent system in its own right with its own environment (which may extend beyond the agents comprising the system of which it is a part), by changing the level of analysis. Similarly, the actions, goals and beliefs are those of the agent system as a whole.

[3] This is similar to viewing a thermostat as a limiting case of an intentional system, it is just not a very interesting one (Dennett 1987).

Properties of the agent's environment

The following set of properties is offered as a first attempt to characterise the environment of an agent system. The environment of an agent system is that part of the world or computational system 'inhabited' by the agent.[4]. The environment may contain other agents whose environments only partially overlap with the environment of the agent under study. The classification is based on that proposed by Russell and Norvig (1995, pages 45–46), except that it makes no reference to the capabilities of a particular agent, i.e., it attempts to describe what the agent's environment is 'really like'.[5] This allows us to describe important aspects of an agent's task(s) in an agent-independent way, without reference to the design of the agent. Such an approach is essential if we are to determine whether two agents inhabit the same environment.

How an agent system is embedded in its environment is an important aspect of its design and the environment as perceived by an agent may differ from 'reality', for example if the agent has no, or only limited sensors, or lacks the ability to represent or reason about the environment (see below). Different agents may employ different sets of concepts relating to the same environment: a fly may be able to detect an object moving rapidly towards it but unable to interpret its visual input in terms of concepts like 'wall', 'floor', or 'ceiling'. In what follows we adopt the position of an omniscient observer, but the same set of properties could be used to classify the environment as it appears to the agent.

Observable vs partially observable environments The environment is observable if it is possible in principle to determine the complete state of the environment at each time point (given reasonable assumptions about sensors and finite processing capacity). Otherwise it is only partially observable. An environment may be observable if it is specified as part of the task, for example an artificial or synthetic environment such as a simulation; naturally occurring environments are usually only observable to some specified level of resolution or abstraction. For example, a chess board is observable, whereas the environment of a mobile robot usually is not. Certain aspects of the agent may themselves form part of the agent's environment, in that they are observable in some sense by the agent. For example, an agent may be able to monitor certain aspects of its internal state, such as the resources remaining to complete its tasks.

Static vs dynamic environments If the environment only changes as a result of the agent's actions, it is static; otherwise it is dynamic. This may be as a result of natural processes, such as the diurnal cycle or the weather, or the

[4] It is specifically *not* the output of the agent's sensors

[5] For example, Russell and Norvig argue that it is "often better to think of an environment as deterministic or nondeterministic from the point of view of the agent"; however this conflates partial observability with nondeterminism.

actions of other agents or both. An environment which is in principle static may appear dynamic to a resource limited agent which is unable to anticipate all the consequences of its actions, even if the actions are infallible (see below).

Deterministic vs nondeterministic environments An environment is deterministic if its future state can be predicted *in principle* by an omniscient observer from the current state of the environment and the set of actions which can be performed in the environment (given finite processing capacity), whether or not the agent system actually makes use of such predictions. Otherwise it is nondeterministic. For example, an environment where there is some means of making a random choice, such as tossing a coin, is nondeterministic. An action is nondeterministic if it can have have several different outcomes when performed in a given state. Whether the environment is predictable in practice depends on how well the agents in the environment implement the actions. If an agent makes errors in executing a deterministic action, the environment will be nondeterministic in practice in the sense that it is not possible to predict the future state of the environment given the current state and the action 'performed' by the agent.

Discrete vs continuous environments If there are a limited number of distinct, clearly defined, percepts and actions, we say the environment is discrete, otherwise it is continuous (Russell & Norvig 1995). For example, the environment of an INTERNET agent is typically discrete, whereas the environment of a mobile robot is usually continuous.

Single vs multiple agents The agent's environment may contain other agents. If it does, these may be of the same kind as the agent, or they may be of different kinds. Whether these agents form part of a single larger agent system may depend partly on the purpose of the analysis, for example if we are considering the environment of one component of a multi-agent system. Even if the other agents are not part of the same multi-agent system, the environment may still be deterministic if the other agents behave in predictable ways, e.g., if they are under the control of the agent being analysed.

Properties of the agent's actions

In this section, we outline some general properties of actions. As with environments, we adopt the perspective of an omniscient observer to facilitate comparisons, however in this case our discussion is constrained to a particular set of idealised actions performed by one or more agent(s). In some cases, it may make sense to classify the agent relative to some subset of the actions it can perform, for example if different types of actions have different properties. In the limit, we may wish to classify each of the agent's actions individually, if this is relevant to the analysis.

We define an action in terms of its preconditions and effects and any constraints which specify the way in which the intended state is achieved, e.g., the time taken to perform the action, or the requirement that the agent should not bump into anything while performing the action. An agent may make errors in performing an action, either by failing to achieve the intended effect of the action, or by violating one of the constraints. We therefore distinguish between the properties of an action (considered abstractly) and the properties of the agent's behaviour which realises or implements the action. For example, whether the agent can reliably perform the same behaviour in the same environment, and whether the agent is equally reliable in all environments (in which the action is possible in principle). The action of 'moving to the next room', may always be reliably executed by an agent if the door is of a given width, say 1m. With a narrower door, the execution of the action may become unreliable, even though the action is still possible in principle (i.e., the doorway is wide enough to allow the agent to pass through).

This allows us to distinguish between the 'true nature' of an action and the action as it appears to an agent (for example, different agents may find the 'same' action more or less difficult or costly to execute. In what follows, we consider only idealised actions independent of the behaviour of the agent (with the exception of action costs which are necessarily agent specific) so that we can say whether two agents are 'performing the same action'. However, a similar set of properties could be used to classify the behaviour of the agent.

Infallible vs fallible actions An action is infallible if it is guaranteed to produce its intended effects when correctly executed in an environment which satisfies the preconditions of the action. Otherwise it is fallible. Note that fallibility is related to the *intent* of an action and only indirectly related to nondeterminism. A nondeterministic action is one which can have several different outcomes when performed in a given state. An infallible action is one in which the possible outcomes of the action are a subset of the intended outcomes of the action. For example, the action of tossing a coin may be an infallible means of choosing between two alternatives (if the coin never lands on its edge), whereas the action of 'rolling a six' with a fair die is fallible. If the agent's actions are fallible, then even a static environment will be nondeterministic.

Utility of actions The utility of an action for a given goal and environment is the utility of the state which results from the action. Different actions may have different utilities. If there are several actions the agent could perform to achieve a goal in the current environment and there is no way of choosing between them, we say the actions have equal utility. If there is a well defined notion of action quality, we say actions have different utilities and the action with maximum utility is correct.

Action costs Different actions may have different resource implications or costs. For example, consumption of energy or money, consumption of time that could be spent doing other things, or abstract costs related to the agent's value system: e.g. doing something one disapproves of would be a cost. Costs may be incurred immediately or they may be deferred until some future time. In some cases, all actions for a given agent may have zero or unit cost, but in general different actions will have different costs, which may or may not be a function of the current state of the environment.[6] An action is optimal if it is correct and there is no other correct action with lower cost. If the environment is nondeterministic, the cost of an action may not be known before it is performed. If the environment is only partially observable, the cost of an action may not be known even after it is performed. In general, different agents will incur different costs in performing the same action. For example, the action 'take the ball from A to B' will be easier for an agent to perform if it can pick up the ball. Note that the agent may not consider all the possible costs of its actions, either because it is unable to represent them (see below) or because it has no way of predicting the cost of the action. For example, an agent which ignores the cost of computation will behave in a manner similar to pure deliberative systems, e.g., classical planning systems.

Types of Actions

In this section, we identify a number of action types which are often associated with the notion of 'agent' (and which may or may not have the properties outlined above).

Sensing actions The ability to perceive its environment is necessary if the agent is to be *reactive*, i.e., able to respond to changes in the environment in a timely manner (Wooldridge & Jennings 1995; Franklin & Graesser 1997). The agent may have several different ways of sensing the environment, which may be more or less fallible, and more or less expensive.

Moving actions If there is no action an agent can perform which has the effect of changing its view of its environment in some way it is said to be immobile, otherwise it is said to be mobile. For example, an agent which receives all its sensory data in the form of broadcast messages is effectively immobile. Conversely, if an agent can move within its environment in such a way as to change what it senses or the actions it can perform (or reduce the costs of performing an action), then it is mobile in the conventional sense. This includes the case of an agent which changes its position in a partially observable environment and that of an agent which moves its execution state to a different machine.

Communicating actions The ability to communicate with other agents is often taken to be one of the defining characteristics of an agent (Wooldridge & Jennings 1995). An agent can be said to communicate with other agents in a meaningful way (other than through its non-communicative actions in the environment), if it interacts with them via some kind of agent communication language (i.e., in a language with a semantics and pragmatics). If there are different kinds of agent in its environment, the agent may have to communicate in several different languages,

[6] In simple cases there may be a single measure of utility/cost. For organisms, and certainly for humans, there does not seem to be. In particular, some costs and benefits may be incommensurable.

Properties of the agent's goals

In this section and the next, we consider properties of the agent's goals and beliefs. In attempting to characterise the beliefs and goals of an agent, we are of course assuming that the agent *explicitly* represents its goals and beliefs, i.e., we can know what the agent's goals and beliefs are by looking inside the agent. Not all agents represent beliefs and goals explicitly, even though they act in a goal-directed manner. For example, the behaviour of an agent may be controlled by a collection of decision rules or reactive behaviours which simply respond to the agent's current environment.

In cases where the agent has no explicit representations of goals or beliefs we assume that it is possible to view the agent as an *intentional system*, that is we ascribe to it the beliefs and goals it *ought* to have, given what we know of its environment, sensors and (putative) desires (Dennett 1987; 1996).[7] For example, an agent which has an 'avoid obstacles' behaviour, can be said to have a goal of 'avoiding collisions', even though this goal is not explicitly represented in the agent. This approach, which Dennett calls "adopting the intentional stance", allows us to ascribe propositional attitudes to agent systems which do not explicitly represent beliefs and goals, without having to know anything about the agent's state or architecture.

In many cases this works reasonably well; for example, the predictions we can make by attributing a goal of avoiding collisions to a behaviour-based agent with an 'avoid obstacles' behaviour will be similar to the behaviour exhibited by the system. In other cases it is more problematic, largely due to the arbitrary nature of intentional attribution to such minimal intentional systems. Given only the agent's desires and its environment, we must assume some sort of design for the agent—some reasonable way of achieving its desires—and work backwards to what sorts of events in the environment are significant, and hence the sorts of percepts and beliefs it 'ought' to have. The more we know about the design of an agent, e.g., what sorts of sensors it has, the easier it is to choose between alternative competing designs, and the sorts of beliefs the agent 'ought' to have. However, in general, viewing an agent as an intentional system seems more likely to yield useful insights than, e.g., a description of the topology and weights of a neural net.

In this section we will focus on the intrinsic goals of an agent. An *intrinsic* goal is one which is not a subgoal of an already intended end (Georgeff & Lansky 1987). Such goals may be thought of as the top-level goals of an agent. They often originate outside the agent system (unless the agent can generate its own goals, see below), with users or other

[7] We may be able to infer some of the agent's current goals from its actions in the environment, but a purely 'behaviouristic' approach can't tell us about other goals which are not currently affecting behaviour, but which, e.g., may have influenced the choice of the current goal.

agent systems. The treatment of sub-goals, goals generated in the process of achieving an intrinsic goal, is similar, but the properties of such goals may be different. For example, an agent may be less committed to a sub-goal than to the intrinsic goal that gave rise to it, if there are other ways of achieving the intrinsic goal. Whereas abandoning a sub-goal may require a change of strategy, abandoning an intrinsic goal may involve negotiation with other agents or reporting failure to the user.

Autonomous generation of goals The ability to generate its own goals is often taken to a defining characteristic of an 'autonomous agent'. The autonomous generation of goals implies that the agent has in built desires or preferences determined by the developer of the agent system. Such desires are often sensitive to both the current state of the environment and the agent system; situations which give rise to a new goal when the agent is in one state may not give rise to goals when the agent is in another state, e.g., when it is attending to a higher priority goal.[8]

Achievement vs maintenance goals A goal to achieve a particular state in the environment once is termed an achievement goal; a goal to maintain or preserve a state in the environment is termed a maintenance goal. For example, a thermostat has a single maintenance goal whereas a goal to find the lowest price for a product or service is an achievement goal.

Single vs multiple goals If an agent is capable of representing (explicitly or implicitly) more than one goal, we say the agent has multiple goals, otherwise it has a single goal. If an agent has only a single achievement goal and this goal is achieved, then the agent typically either terminates or becomes quiescent awaiting further goals (unless it is capable of autonomous goal generation, see above). If the agent has at least one maintenance goal, then the agent's 'lifetime' is potentially unbounded. If the agent has multiple goals, they may be all of one type or a mixture. For example, a delivery robot may have several achievement goals to deliver packages and a maintenance goal not to let its batteries run down. An agent with multiple goals may process only one goal at a time or it may process some subset of its goals in parallel. If there is a single world state, reachable from the current state, in which all the agent's goals are (simultaneously) true we say the agent's goals are consistent. If there is no sequence of states reachable from the current state which satisfies the agent's goals, we say the goals are inconsistent. Note that the agent may not know whether its goals are consistent or not.

Commitment to goals If an agent only abandons a goal when it is achieved we say the agent is strongly committed to its goal(s). If the agent will also abandon a goal in other circumstances, e.g., if it can prove that it cannot be achieved or after expending a given amount of effort, we say the agent is weakly committed to its goals. For example, an agent may ignore a new autonomously generated goal if it already has a goal, or the new goal may always replace the current goal, or the agent may use the relative utilities of the goals to decide which goal to pursue. Alternatively, if at least some of the agent's goals originate outside the agent (e.g., in the form of requests from other agents or users), the agent may either be inherently cooperative if the goal always takes precedence over its own desires and goals, or more or less autonomous in the sense that it only adopts the goal if it is in its own interest to do so (as judged by its own desires) or if the external goal does not conflict with any of the agent's existing goals. If the agent is capable of representing multiple goals, then it can add the goal to its list of pending tasks.

Utilities of goals If the agent's goals are all equally important, in the sense that the reward for achieving them is the same, we say the agent's goals have equal utility. If the agent's goals have different utilities, the utilities may be incommensurable, e.g., if the utilities define only a partial order over goals, or commensurable, e.g., the goals are ordered on a ratio scale, allowing the agent to determine that achieving goal A has greater utility than the combined utility of achieving goals B and C. If the agent's goal(s) have to be achieved before some deadline or the utility of achieving the goal varies with time, then the goals are time dependent or real-time. If the utility of achieving one goal is independent of the utility of achieving future goals, we say the agent's goals are episodic, otherwise we say they are non-episodic. For example, the goals of an information retrieval agent may be episodic in that its success in achieving subsequent goals (requests for information) is not dependent on how well it achieves its current goals. In contrast, the goals of a robot building a tower of blocks may be non-episodic, if inability to achieve the current goal prevents future goals being achieved (or causes the tower to collapse).[9]

Meta-goals Some conditions on the attainment of goals such as the time or resources available to achieve a goal or the reliability with which the goals are achieved cannot themselves be modelled as goals. A meta-goal is a constraint on another goal or more generally on the internal states and processes of of the agent. For example an agent

[8] It seems less natural to view such autonomously generated goals as simply sub-goals of a higher-level maintenance goal. Consider a delivery robot which, when it knocks something over, picks it up again. It is not clear how a top level maintenance goal to pick up objects which have been knocked over would be formulated ('keeping X upright' or 'keeping X on top of Y, for any relevant X and Y'?). Even if such a goal could be formulated, the agent would still require some low level processing to detect when this goal had been violated. Although autonomous goal generation does effectively the same thing, it does not require the explicit representation of the universally quantified goal state, since we can write a goal generator which responds to particular instances of things which have been knocked over.

[9] In (Russell & Norvig 1995) this is seen as a property of environments rather than goals.

which responds to requests for information may be constrained to spend no more than t seconds processing each request. (Note that this is different from the case in which we require that, e.g., a plan to achieve a goal should take no more than s seconds to execute.)

Properties of the agent's beliefs

In this section we consider the characteristics of the agent's beliefs about its environment, i.e., what the agent believes its environment to be like as opposed to what its environment is 'really like' (see above). The agent's representation of its environment may diverge from reality in a number of ways, for example the agent may simplify or abstract from the true nature of the environment in an attempt to reduce the effort required to construct and maintain the representation by focusing on those aspects which are critical to the attainment of the agent's goals, or the agent may simply be mistaken.

A complete characterisation of the agent's beliefs would require some way of classifying the ontology used by the agent. We focus on the general characteristics of the agent's beliefs rather than the particulars of what it believes in any given situation or how these beliefs are represented. In some cases, an agent may use several different representations of beliefs, for example it may use different kinds of representations for different kinds of information or it may use different representations of the 'same' belief, e.g., in the hierarchical processing of sensory data. At one extreme the agent may simply store the percepts produced by its sensors, while at the other constructing a representation may require considerable effort in the form of parsing or image understanding.

Not all agents build and maintain an explicit representation of their environment. As with goals, we shall assume that it is possible to determine the general characteristics of the agent's implicit beliefs by viewing the agent as an intentional system.

Consistent An agent's beliefs are consistent, if for any proposition p, the agent does not believe both p and $\neg p$, otherwise they are inconsistent. For example, it is inconsistent for an agent for an agent to believe both that it has and has not sufficient resources to achieve its goals. Whether an agent's beliefs are currently consistent is a contingent fact; what we are concerned with is whether the agent's beliefs can ever be inconsistent. Inconsistencies can arise if there are several different ways of deriving the 'same' belief which give different results; for example, a mail filtering agent which classifies a message as 'interesting' based on its author and 'uninteresting' based on its subject. Some designs guarantee consistency, for example, where the agent's beliefs are limited to the current values of its sensors. However, in general, determining whether an agent's beliefs are consistent is is only possible if the agent's belief representation corresponds to a very weak logic.

Certain An agent's beliefs are certain if the representation of its beliefs does not admit degrees of belief, i.e., for any proposition p that the agent entertains, it either believes that p or it believes that $\neg p$, otherwise it is uncertain about p. In the latter case, the agent may represent it degree of belief in a proposition in the form of confidence factors or probabilities, or in some other form. For example, an agent may believe that it can obtain the information for which it is searching on a given web page with certainty of 0.1. Note that an agent's beliefs can be certain, even if the agent's environment is only partially observable.

Nested propositional attitudes The agent's beliefs contain nested propositional attitudes if the agent can represent its own beliefs and goals and/or the beliefs and goals of other agents. For example, an agent may believe that it has a goal to deliver a package to an office on the first floor, or that another agent knows whether there are free seats on the next flight to Madison.

Some examples

In this section we briefly illustrate the application of the classification scheme outlined above in two prototypical examples, a mail filtering agent and a multi-agent system for transporting large objects. The mail filtering agent is based on that described by Maes et al (Maes 1994; Lashkari, Metral, & Maes 1994), whereas the multi-agent system has many similarities to that described by Barnes et al (Coddington & Aylett 1996; Barnes *et al.* 1997; Coddington & Aylett 1997). In each case, we illustrate how the classification helps to clarify the level of the analysis and how different interpretations of the description of the system results in a different classification of the system.[10]

A mail-filtering agent

Mail filtering is a frequently cited application for agent-based systems. For example Maes has described an electronic mail agent which learns to prioritise, delete, forward, sort and archive mail messages on behalf of the user (Maes 1994; Lashkari, Metral, & Maes 1994). In this section we illustrate how one might go about classifying a hypothetical mail filtering program, which is broadly similar to that developed by Maes et al. and highlight those aspects which seem most sensitive to the level of analysis and implementation strategy adopted.

We assume that the environment of the agent is one or more mail spool files which the agent monitors for incoming mail messages. Such an environment is observable, dynamic and nondeterministic. It is discrete at one level of abstraction, in that all messages consist of one or more fields some of which are always present. However, in general, the contents of the fields can be anything (unless the messages are completely stereotyped) and so is more appropriate to consider the agent's environment as continuous. The environment may or may not contain other agents, for example,

[10] In many ways it would have been preferable to classify implemented systems, however in the small number of systems we have studied to date, there is insufficient information in the published literature to allow the agent system to be classified on each of the dimensions identified above.

the mail delivery system may be an agent, or the mail filter itself may be part of a larger multi-agent system, e.g., a personal digital assistant which notifies the user of important messages or responds to routine requests for meetings etc.. If the mail filter program is not persistent, for example, if a new instance of the agent is created by another agent to classify each newly arrived mail message as interesting or uninteresting, then we could be justified in describing its 'environment' as both static and deterministic, since the environment is a single mail message.

The actions the agent can perform in such an environment are typically infallible. (The action may be incorrect given the goal(s) of the agent, e.g., it may incorrectly delete a message as uninteresting, but the action of deleting the message is usually guaranteed to succeed.) Whether the agent's actions have different utilities or costs depends on whether it has different way of classifying a message as interesting. For example, checking for keywords and performing a semantic analysis of the message are two different way of determining if the message is interesting, but they typically have very different utilities and costs. An agent which monitors a mail spool file must be able to sense when the file is updated, and may also require additional sensing actions to extract information from the message. (Conceivably, an agent which is simply given the goal of determining if a message with a given set of keys is interesting would not need to sense its environment, since all the information it requires to perform its task is part of the goal.) The agent may be able to communicate with other agents, for example to ask for assistance with the semantic analysis of a message or to request information about the sender of a message. Alternatively, it may simply delete uninteresting messages from the spool file or move them to a different mail folder. It is unlikely that a mail filtering agent would be mobile.

The goals of the agent will typically take the form of a stream of autonomously generated achievement goals. However if a new instance of the agent is created to classify each message, then the agent only ever processes a single achievement goal (and is not autonomous). The agent may be strongly committed to its goals, e.g., if the agent always classifies one message before moving on to the next or is its architecture guarantees that the time taken to classify a message is bounded. All goals have equal utility and are episodic.[11] The agent may have meta-goals or it may be possible to ascribe meta-goals to the agent, for example, if the time available to classify a message is a function of load on the system or the rate of incoming messages.

The beliefs of the agent depend on the information it can extract from its environment, e.g., the content of messages, the number of messages waiting to be classified, the rate of incoming messages etc., and the ways in which its beliefs can be manipulated, e.g., whether it can infer new beliefs from its current beliefs. In addition, the agent will typically have certain innate beliefs, e.g., that messages from certain people are always important. Its beliefs may be inconsistent and uncertain, for example it may expresses its belief in whether a given message is interesting in the form of a certainty factor, and may contain nested propositional attitudes, for example if it maintains an explicit or implicit model of the user's preferences. If the MAXIMS agent (Maes 1994) does not have enough confidence in its prediction of whether an action is appropriate, it asks for help from other mail filtering agents. Over time, it learns which other agents are trustworthy sources of information for different classes of problems.

A multi-agent transportation system

In contrast, in this section we illustrate how one might classify a multi-agent transportation system based on physical robots. The system we describe is loosely based on that developed by Barnes, Aylett and Coddington (Coddington & Aylett 1996; Barnes *et al.* 1997; Coddington & Aylett 1997), in which two or more physical robots with potentially different characteristics cooperate to transport an object. The system contains both behaviour-based and deliberative components and must confront all the problems of real agents operating in an uncertain physical environment. In doing so, we attempt to illustrate how the use of our classification scheme can help to characterise the similarities and differences in systems with very different architectures and implementations.

In what follows, we consider the case of an agent system consisting of two or more physical robots, which cooperate to move objects around a warehouse. The environment of the system is only partially observable, static (since it is assumed that nothing moves unless one of the robots move it), nondeterministic (since actions may have several outcomes) and continuous. In the experiments described by Barnes et al., the environment contains no other agent systems, though the system being analysed itself consists of several agents.

The actions the agent can perform include various sorts of sensing, grasping and motion, however the agents do not communicate directly with one another. Rather when both robots are holding an object the relative motion of one robot is transmitted to the other robot and vice versa. Actions are fallible (since they may fail to have their intended effects) and have differing utility and cost, for example there are many different routes between any two points in the environment, only some subset of which have maximum utility and minimum cost.

The system has both achievement and maintenance goals, for example to transport an object to a particular location while avoiding collisions. However, the generation of goals is not autonomous; goals are generated externally to the system and either form part of the system's definition or are input by a human user. At any one time the system has at most one achievement goal, to which it is strongly committed. For example, if the object is dropped, the agents will attempt to pick it up again. Goals have differing utilities, in that maintenance goals, such as the avoidance of collisions, can modify the actions generated to satisfy the achievement goal. Each goal may or may not be episodic, for example if the achievement of a transportation goal is dependent on achieving a goal to pick up the object, or if the goal is collect

[11] This if often true of filters, where the aim is to generate some information which can be used to determine the utility of other goals.

several objects together at the same location. The system appears to have no meta-goals.

The beliefs of the system are dependent on the information it can obtain from the sensors of the physical robots and on the innate beliefs of the 'Reflective Agent' which contains an abstract world model representing the large scale static properties of the environment and abstract representations of the actions which can be performed by the behavioural agents. Beliefs are not represented explicitly in the behaviour based architecture of the physical robots, but certain beliefs can be ascribed to the system at this level, for example that an agent is holding the object, or that it is about to collide with a wall, and such beliefs serve to modulate the behaviour of the agents, e.g., by causing the agents to slow down when they approach an object.

Conclusions and further work

To select an appropriate tool or tools to build an agent based system we need to map from features of agent systems to implementation technologies. In this paper we have presented a classification scheme for agent systems as a first step in developing a methodology for the selection of implementation technologies. We have said almost nothing about implementation technologies themselves, largely because we do not understand which of their many features are significant. However, while such classification is desirable, and ultimately necessary, in the short term is seems more likely that we can make progress by focusing on the relationship between the features of successful and unsuccessful agent systems and the toolkit employed in developing the system where this is considered to be a factor in the success or otherwise of the system.[12]

To date, relatively few agent systems have actually been implemented and in the majority of cases little use has been made of previous work. However this situation is beginning to change, and we are starting to see the reuse of tools in the development to systems for more than one problem or domain. As such data become available, it should be possible to classify the types of systems which have been built with each toolkit, and use this information to develop a mapping from problems to tools. It should also be possible to explore the differences between agent systems in a more systematic way, to determine if one is a special case of another, or to discover why an approach which was useful in one domain turned out not to be useful in an apparently similar domain. By focusing on such anomalies, it may be possible to refine our classification of agent systems.

The classification scheme outlined above can be seen as a first step towards an ontology for agent systems. It is based on an analysis of what seem to us to be the most important features of the cluster concept 'agent'. However, it is very much a first attempt and will undoubtedly evolve in response to attempts to apply it to new systems. Further work is required to develop and refine the set of classification dimensions and to develop a classification scheme for toolkits (which we suspect is a much harder problem). Further work is also required to apply the classification scheme developed above to a wider range of agent systems. However, we believe this approach will lead to a clearer idea of the similarities and differences between systems by allowing us to abstract away from the details of the tasks the systems perform. If we are fortunate, such a classification may also suggest future lines of research.

[12]This assumes that the details of the task and task specific knowledge is not essential to the analysis, but we have to assume this to support any sort of inductive generalisation.

Acknowledgements

We wish to thank Aaron Sloman, Jeremy Baxter, Natasha Alechina and the participants in the 'Research Oven' seminar at the School of Computer Science, University of Birmingham for many useful discussions and comments. This research is partially supported by a grant from the Defence Evaluation and Research Agency (DERA Malvern).

References

Barnes, D. P.; Ghanea-Hercock, R. A.; Aylett, R. S.; and Coddington, A. M. 1997. Many hands make light work? an investigation into behaviourally controlled co-operant autonomous mobile robots. In Johnson, W. L., ed., *Proceedings of the First International Conference on Autonomous Agents*, 413–420. ACM SIGART.

Coddington, A. M., and Aylett, R. S. 1996. Plan generation for multiple autonomous agents: an evaluation. In *Proceedings of the 15th Workshop of the UK Planning and Scheduling Special Interest Group*, 126–137. Liverpool John Moores University.

Coddington, A. M., and Aylett, R. S. 1997. Interfacing UCPOP to multiple execution agents. In *Proceedings of the 16th Workshop of the UK Planning and Scheduling Special Interest Group*, 19–30. University of Durham.

Dennett, D. C. 1987. *The Intentional Stance*. MIT Press.

Dennett, D. C. 1996. *Kinds of Minds: Towards an understanding of consciousness*. Basic Books.

Fisher, M.; Müller, J.; Schroeder, M.; Staniford, G.; and Wagner, G. 1997. Methodological foundations for agent-based systems. *Knowledge Engineering Review* 12(3):323–329.

Franklin, S., and Graesser, A. 1997. Is it an agent, or just a program?: A taxonomy for autonomous agents. In *Intelligent Agents III: Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages*, volume 1193 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag. 21–35.

Georgeff, M. P., and Lansky, A. L. 1987. Reactive reasoning and planning. In *Proceedings of the Sixth National Conference on Artificial Intelligence, AAAI-87*, 677–682.

Lashkari, Y.; Metral, M.; and Maes, P. 1994. Collaborative interface agents. In *Proceedings of the Twelfth National Conference on Artificial Intelligence, AAAI-94*, volume 1. MIT Press.

Luck, M., and d'Inverno, M. 1995. A formal framework for agency and autonomy. In *Proceedings of the First International Conference on Multi-Agent Systems*, 254–260. AAAI Press/MIT Press.

Maes, P. 1994. Agents that reduce work and information overload. *Communications of the ACM* 37(7):30–41.

Nwana, H. S. 1996. Software agents: an overview. *Knowledge Engineering Review* 11(3):205–244.

Russell, S., and Norvig, P. 1995. *Artificial Intelligence: a modern approach*. Prentice Hall.

Sloman, A., and Logan, B. 1998. Architectures and tools for human-like agents. In *Proceedigs of the Second European Conference on Cognitive Modelling, ECCM-98*. (to appear).

Sloman, A. 1995. A philosophical encounter. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, IJCAI-95*, 2037–2040. Morgan Kaufman. (Also Birmingham University Cognitive Science technical report: CSRP-95-06).

Wooldridge, M., and Jennings, N. R. 1995. Intelligent agents: Theory and practice. *Knowledge Engineering Review* 10(2):115–152.