Integrating Models and Behaviours in Autonomous Agents: Some Lessons Learned on Action Control

Ferguson, I.A.

Proceedings of the AAAI Spring Symposium on Lessons Learned from Implemented Software Architectures for Physical Agents, 1995

of different behaviors and functionalities which have proven themselves to be highly useful for surviving (and thriving) in dynamic, uncertain, and hostile environments. In so far as such behaviors and capabilities can help to increase the various degrees of autonomy, effectiveness, robustness, and flexibility of one's artificial agent (as defined earlier in this paper), it would seem prudent to mimic such aspects wherever feasible. On the other hand, if the agent's task domain does not demand such sophisticated functionality (which, after all, will be expensive to design, maintain, and operate) then simpler, non-anthropomorphic functionalities may well be more appropriate.

- *Simulation — What, if any, role can advanced simulation technology play in developing and verifying modules and/or systems? Can we have standard virtual components/test environments that everybody trusts and can play a role in comparing systems to each other? How far can development of modules profitably proceed before they should be grounded in a working system? How is the architecture affected by its expected environment and its actual embodiment?*

  Simulators are not a substitute for the real world. On the other hand, if carefully designed, they can provide useful insights into some of the environmental conditions an agent will have to face when ultimately launched in the real world. In addition, because simulators can provide designers with a host of powerful tools for performing controlled, repeatable experiments they can be instrumental in assisting the designer in understanding various aspects of the agent's behavioral ecology: the relationship between the agent's structure or configuration, its environment, and its resulting behaviors (as defined earlier in the paper). In this respect, I think simulation should be regarded as an integral part of the agent design phase, during which the designer is given the opportunity to explore and incrementally test the agent's (potentially vast) design space; in particular, it gives the designer the opportunity to differentiate between, and therefore establish, the agent's fixed and reviewable design decisions, as defined in section 5 above.

- *Learning — How can a given architecture support learning? How can knowledge and skills be moved between different layers of an agent architecture?*

  While a number of different architectures in the literature have been reasonably successful at applying a variety of different learning techniques to the agent control problem (e.g. reinforcement, performance, explanation-based learning), the degree of self-tuning which these agents have demonstrated has been limited primarily, it would appear, due to their reliance on fixed, pre-defined parameter/feature sets, and/or on relatively simplistic performance criteria according to which their behaviors are being optimized. Given the (by now) empirically validated fact that establishing an *ideal* agent parametrization — that is, one which is intended to produce a particular type of desired behavior in an optimal manner — depends very strongly on the agent's particular task constraints and environmental circumstances, more work is required to establish a taxonomy of different task environments which agents could use to self-tune their internal parameters and optimize their behaviors with respect to their own particular resources and task constraints. Wilson (Wilson 1990), among others, has made some inroads into this problem but much work remains in formalizing more complex environments involving global task constraints, real-time events, limited or shared resources, multiple agents, collaborative tasks, hostility, deceit, etc.

how should they be implemented? How much does each level/component of an agent architecture have to know about the other levels/components?

Generally speaking, I think architectures comprising multiple, concurrent layers are the best choice for designing individual agents. A number of reasons can readily be given for using multiple layers: they increase the modularity and scalability of the agent's design; they make the long-term maintenance problem more manageable; they provide an effective framework for integrating a variety of *hybrid* control functions, ranging from purely non-deliberative to deliberative; they can increase the agent's level of robustness through enabling the use of concurrent, redundant behaviors, and through permitting a degree of distribution of the agent's overall control function. The opening statement above, of course, raises (at least) two important questions: What behavior/functionality should be placed in a single layer of an agent? What behavior/functionality should be placed in a single agent? Answers to either of these questions are non-trivial, likely depending on characteristics of the particular task domains of the agent(s) in question. However, some performance criteria might prove useful in answering these questions (see below).

- *Performance — What types of performance goals and metrics can realistically be used for agents operating in dynamic, uncertain, and even actively hostile environments? How can an architecture make guarantees about its performance with respect to the time-critical aspect of the agent's physical environment? What are the performance criteria for deciding what activities take place in each level/ component of the architecture?*

A variety of different performance goals have appeared in the literature. Broadly speaking these can be divided into those which measure some aspect of the internal performance of the agent (e.g. the efficiency of its planning functions, the number of communication acts performed, total computational resources consumed, the utilization of these computational resources); and those which assess performance of the agent with respect to some externally measurable task or goal (e.g. mean time between failure, speed of task completion, accuracy of task completion, total (sub-)tasks completed). As argued earlier in the paper, detailed quantitative comparisons between different architectures are next to impossible given the number and variety of such performance criteria, most of which, judging by existing cases in the literature, can only be interpreted with respect to the particular architecture being measured.

Conventional real-time systems are designed to meet the individual timing requirements of a set of system tasks in such a way that they are not only logically predictable but also temporally predictable. For this to occur, such a system must ensure, in advance, that sufficient resources can be pre-allocated for achieving each and every one of its time-critical tasks. For a resource-bounded agent operating in a dynamic multi-agent world, the conception of real time needs to be revised somewhat. Whereas conventional real-time systems are usually defined in terms of a statically determined control sequence, real-world agents have to deal with both hard and soft aperiodic real-time events which might occur at unexpected times throughout the period of their operation. To cope with such events agents will need to make use of opportunistic control strategies such that they may direct their choice of actions dynamically in response to unanticipated real-time events. While such strategies can virtually guarantee timely responses to a number of localized hard real-time events (e.g. avoiding collisions), it also means that agents' precise action sequences and long-term goal behaviors can no longer be pre-determined with absolute certainty. But this is to be expected of resource-bounded agents: just as the correctness of their behaviors might need to be traded off against their robustness, so too might the long-term predictability of their task-level activities.

Possible performance criteria for deciding what activities should take place in a given level within an agent might include: minimizing inter-level messaging (e.g. by clustering "related" activities in the same level), balancing the number of different world events that each level can respond to, clustering activities according to their space and/or time granularity profiles, and exploiting the "naturalness" with which each activity's corresponding knowledge and skills requirements can be expressed within a given level's representation formalism. Analogous criteria might also be applicable to decide whether the same activities should be carried out by more than one agent.

- *Psychology — Why should we build agents that mimic anthropomorphic functionalities? How far can/should we draw metaphoric similarities to human/animal psychology? How much should memory organization depend on human/animal psychology?*

Humans, by and large, are capable of a vast number

Wilson, S.W. The animat path to AI. In J.A. Meyer and S.W. Wilson, editors, *From Animals to Animats: Proceedings of the First International Conference on the Simulation of Adaptive Behavior.* MIT Press: Cambridge, MA, 1990.

A Appendix: Symposium Questions

Here are my responses to some of the questions raised by the organizers of the symposium:

- *Coordination — How should the agent arbitrate/ coordinate/cooperate its behaviors and actions? Is there a need for central behavior coordination?*

  It would seem to be a requirement that if an agent is comprised of a number of concurrently operating behaviors, some of which conflict for one reason or another (e.g. they propose execution of mutually exclusive actions), then such behaviors will require some form of central coordination. Moreover, in practice, I would suggest that there will always be some behaviors present in an agent which inherently conflict with one another — so behavior arbitration will be a must. Of course "central" in the above sense need not imply that all coordination control functions be localized within one module of the agent; rather, it would only seem to require that global or meta-level knowledge be taken into account when determining which of the agent's multiple, and potentially conflicting, actions should be selected for execution. In either case (TouringMachines, incidentally, localize their meta-level control rules in a single module), it is the agent programmer's responsibility to ensure that the appropriate meta-level control knowledge be appropriately crafted and correctly deployed within the agent. For software engineering purposes, co-location or centralization of the agent's behavior coordination functions seems preferable; for run-time execution purposes this may well not be the case.

- *Interfaces — How can human expertise be easily brought into an agent's decisions? Will the agent need to translate natural language internally before it can interact with the world? How should an agent capture mission intentions or integrate various levels of autonomy or shared control? Can restricted vocabularies be learned and shared by agents operating in the same environment?*

  Answers to most of these questions depend very much on the characteristics, and in particular the richness and complexity, of the agent's task domain. If humans and agents need to interact then it would be desirable, if not essential in some situations, for agents to be capable of natural language recognition and generation (relatedly, speech processing would also be useful); on the other hand, for a number of simpler autonomous or operational tasks — but not excluding those which might require some form of coordination with other (artificial) agents — a less unstructured and more constrained language of interaction would seem preferable, even at the expense of lost expressivity. Human-computer interfaces are, in general, difficult to design well. And given the seemingly conflicting needs for providing, on the one hand, explicit, on-demand interaction between user and agent; and for requiring, on the other, that agents minimize (or at least simplify) the user's interaction with the computer by carrying out delegated tasks as transparently and autonomously as possible, it is almost certainly the case that intuitive and effective human-agent interfaces will be even more challenging to design.

- *Representation — How much internal representation of knowledge and skills is needed? How should the agent organize and represent its internal knowledge and skills? Is more than one representational formalism needed?*

  Answers to these questions, again, depend heavily on the agent's task domain. As many successful agent designs in the literature have shown, some task domains require essentially no representation whatsoever. In general, I think one should use only as much (or as little) internal representation as one needs. How much one needs, however, is likely to depend on one's preference weightings of such criteria as speed of run-time execution, ease of behavior/control programming and long-term maintenance, and ease of incorporating learning mechanisms within the agent (assuming, as I do, that at the very least, agents will require explicit representations of their goals if they are to learn or improve their performance). Because of such diverse and conflicting criteria I think it makes sense to use as many representational formalisms as one needs. In this respect, the TouringMachine architecture demonstrates how several such representations can be fairly easily accommodated in a single agent using multiple hybrid control layers.

•*Structural — How should the computational capabilities of an agent be divided, structured, and interconnected? What is the best decomposition/ granularity of architectural components? What is gained by using a monolithic architecture versus a multi-level, distributed, or massively parallel architecture? Are embodied semantics important and*

References

Agre, P.E. & Chapman, D. Pengi: An implementation of a theory of activity. In *Proceedings Conference of the American Association for Artificial Intelligence*, pages 268—272, 1987.

Bond, A.H. & Gasser, L., editors. *Readings in Distributed Artificial Intelligence*. Morgan Kaufmann: Palo Alto, CA, 1988.

Bratman, M.E., Israel, D.J., & Pollack, M.E. Plans and resource-bounded practical reasoning. *Computational Intelligence*, 4(4):349—355, 1988.

Brooks, R.A. Intelligence without representation. *Artificial Intelligence*, 47:139—159, 1991.

Cohen, P.R., Greenberg, M.L., Hart, D.M., & Howe, A.E. Trial by fire: Understanding the design requirements for agents in complex environments. *AI Magazine*, 10(3):32—48, 1989.

Cohen, P.R. A survey of the Eighth National Conference on Artificial Intelligence: Pulling together or pulling apart. AI Magazine, 12(1):16—41, 1991.

Covrigaru, A.A. & Lindsay, R.K. Deterministic autonomous agents. AI Magazine, 12(3):110—117, 1991.

Davis, E. *Representations of Commonsense Knowledge*. Morgan Kaufmann: San Mateo, CA, 1990.

Davis, R. & Hamscher, W. Model-based reasoning: Troubleshooting. In Howard E. Shrobe and AAAI, editors, *Exploring Artificial Intelligence*, pages 297—346. Morgan Kaufmann: San Mateo, CA, 1988.

Drummond, M.E. & Kaelbling, L.P. Integrated agent architectures: Benchmark tasks and evaluation metrics. In *Proceedings DARPA Workshop on Innovative Approaches to Planning, Scheduling, and Control*, pages 408—411, Morgan Kaufmann: San Mateo, CA, 1990.

Durfee, E.H. & Montgomery, T.A. A hierarchical protocol for coordinating multiagent behaviors. In *Proceedings Conference of the American Association for Artificial Intelligence*, pages 86—93, 1990.

Ferguson, I.A. *TouringMachines: An Architecture for Dynamic, Rational, Mobile Agents*. Ph.D. thesis, Computer Laboratory, University of Cambridge, Cambridge UK, 1992.

Ferguson, I.A. Autonomous Agent Control: a Case for Integrating Models and Behaviors. In *Proc. of the AAAI-94 Fall Symposium on Control of the Physical World by Intelligent Agents*, New Orleans, LA, 1994.

Ferguson, I.A. On the Role of BDI Modeling for Integrated Control and Coordinated Behavior in Autonomous Agents. *Applied Artificial Intelligence*, 9(4), 1995. In press.

Galliers, J.R. The positive role of conflict in cooperative multi-agent systems. In Jean-Pierre Müller and Yves Demazeau, editors, *Decentralized AI*. Elsevier (North-Holland): Amsterdam, NL, 1990.

Georgeff, M.P. Planning. In James Allen, James Hendler, and Austin Tate, editors, *Readings in Planning*, pages 5—25. Morgan Kaufmann: San Mateo, CA, 1990.

Hanks, S. & R. Firby, J. Issues and architectures for planning and execution. In *Proceedings DARPA Workshop on Innovative Approaches to Planning, Scheduling, and Control*, pages 59—70, Morgan Kaufmann: San Mateo, CA, 1990.

Kinny, D.N. & Georgeff, M. Commitment and effectiveness of situated agents. Technical Note 17, Australian Artificial Intelligence Institute, Carleton 3053, Australia, April 1991.

Kirsh, D. Today the earwig, tomorrow man? *Artificial Intelligence*, 47:161—184, 1991.

Langley, P. Machine learning as an experimental science. *Machine Learning*, 3:5—8, 1988.

Maes, P. Situated agents can have goals. *Robotics and Autonomous Systems*, 6(1&2):49—70, 1990.

Maes, P. Modeling Adaptive Autonomous Agents. *Artificial Life*, 1:135—162, 1994.

Minsky, M.L. *The Society of Mind*. Simon and Schuster: New York, NY, 1986.

Pollack, M.E and Ringuette, M. Introducing the Tileworld: Experimentally evaluating agent architectures. In *Proceedings Conference of the American Association for Artificial Intelligence*, pages 183—189, 1990.

Poole, D.L, Goebel, R.G., & Aleliunas, R. Theorist: A logical reasoning system for defaults and diagnosis. Research Report CS-86-06, University of Waterloo, Waterloo, Ont., February 1986.

Shoham, Y. Agent-oriented programming. *Artificial Intelligence*, 60(1):51—92, 1993.

Shoham, Y. and McDermott, D. Problems in formal temporal reasoning. In James Allen, James Hendler, and Austin Tate, editors, *Readings in Planning*, pages 581—587. Morgan Kaufmann: San Mateo, CA, 1990.

Vere, S. and Bickmore, T. A basic agent. *Computational Intelligence*, 6(1):41—60, 1990.

TouringMachine architecture, these design decisions were established upon close examination of the intended TouringWorld domain. For instance, the decision to divide control among multiple, independent concurrent layers was influenced by the fact that TouringMachines would have to deal flexibly and robustly with any number of simultaneous events, each occurring at a potentially different level of space-time granularity. Such differences in event granularity, together with the need for carrying out both long-term, deadline-constrained tasks, as well as short-term reactions to unexpected events, ultimately played a part in the decision to combine both deliberative and non-deliberative control functions into a single hybrid architecture. In turn, the need to ensure that any such (non-real-time) deliberative functions be “suitable" for use in a real-time domain such as the TouringWorld — in other words, that they be efficient and effective on the one hand but flexible and robust on the other — suggested that such deliberative functions be: *(i)* latency-bounded in order to provide guaranteed system responsiveness (this in turn demands fairly strict control over internal computational resource use); *(ii)* that they operate incrementally (in other words, that they be capable of suspending operation and state after regular — and arbitrarily small — periods of processing time); and *(iii)* that they serve merely as *resources* for action rather than as strict *recipes* for overall agent control. This last requirement would also become the main motivating force behind the decision to employ a context-sensitive mediatory control policy for establishing control layer priorities. Other design decisions worth mentioning here include the incorporation of functions for reasoning about — or modelling — other agents' actions and mental states and for identifying and flexibly resolving conflicts within and between agents (this is necessary because the TouringWorld domain is populated by multiple intentional agents with limited computational and informational resources); and mechanisms for constantly sensing and monitoring the external world (which are needed since the TouringWorld domain is both dynamic and unpredictable).

Identification and isolation of the second type of design decisions, reviewable decisions, are those which, as their name suggests, can be “reviewed after they are implemented" (Cohen 1991, page 31). The purpose of differentiating fixed and reviewable design decisions was to enable the basic (fixed) design to be implemented and run as early as possible, and to provide an empirical environment in which to develop iteratively this basic agent model, to test hypotheses about how the model should behave, and then to review, subsequently, particular design decisions in the light of observed performance. Also, by providing — in addition to the TouringMachine agent architecture — a highly parametrized and controllable testbed environment like the TouringWorld, a very effective and productive framework in which to carry out such design activities was thus established. Examples of reviewable TouringMachine design decisions — established with empirical feedback gained through use of the TouringWorld Testbed — include the particular set of reactive rules initially made available to the agent, the contents of its various domain-specific libraries (plan schemas, BDI model templates, conflict-resolution methods, and space-time projection functions), the initial set of heuristics used to program the agent's focus of attention mechanisms, and the precise set of censor and suppressor control rules which are used to mediate the actions of the agent's three control layers.

In conclusion, the evaluation of TouringMachines appeared to support the claim that it is both desirable and feasible to combine non-deliberative and suitably designed and integrated deliberative control functions in a single, hybrid, autonomous agent architecture. As was demonstrated, the resulting architecture, when suitably configured, was capable of effective, robust, and flexible behaviors in a reasonably wide range of complex single- and multi-agent task scenarios. As described above, the behavioral repertoire of TouringMachines is wide and varied, including behaviors which are reactive, goal-oriented, reflective, and also predictive. The evaluation, furthermore, suggested that establishing an appropriate balance between reasoning and acting — that is, between appropriate degrees of deliberative and non-deliberative control — would appear to depend on characteristics of the task environment in which the particular TouringMachine is operating. More generally, and in line with the experiences of both Maes (Maes 1990) and Pollack (Pollack & Ringuette 1990), there is evidence to suggest that environmental factors invariably play an important role in determining which agent configuration or parametrization is the most appropriate for any given situational context. Finally, one cannot underestimate the importance of deploying — from the earliest stages of design — concrete measures for carrying out extensive experimentation. In this respect, the TouringWorld Testbed domain has proved a viable and useful system for evaluating different agent control designs.

traffic lights. Scenarios were selected with the aim of evaluating some of the different capabilities and behaviors which TouringMachines will require if they are to complete their tasks in a competent and effective manner — for example, reacting to unexpected events, effecting of goal-directed actions, reflective and predictive goal monitoring, spatio-temporal reasoning, plan repair, coping with limited computational and informational resources, as well as dealing with real-time environmental change. The scenarios can be considered interesting because they succinctly exercise agents' abilities to carry out time-constrained tasks in complex — partially-structured, dynamic, real-time, multi-agent — environments.

It was not the aim of the evaluation to show that the TouringMachine architecture is in any sense "optimal". As argued elsewhere (Ferguson 1992), optimal rational behavior will in general be impossible if the agent is resource-bounded, has several goals, and is to operate in a real-time multi-agent environment in which events are able to take place at several levels of space-time granularity. As such, one should more realistically expect a TouringMachine to behave satisficingly, but at times — for example, when under extreme real-time pressure — to fail to satisfy every one of its outstanding goals. What was really of interest here was understanding how the different configurations of agents and the different environmental characteristics to which such configurations are subjected affected, positively or negatively, the ability of agents to satisfy their goals.

It was also not the aim of the evaluation to show that TouringMachines were "better"' than other integrated agent architectures at performing their various tasks. Rarely is it the case that the actual and/or intended task domains of different agent architectures are described in sufficient detail so as to permit direct comparisons of agent performance. The lack, at present, of any common benchmark tasks or of any universally agreed upon criteria for assessing agent performance — previous evaluations have relied either on a single performance criterion (for example, the total point score earned for filling holes in specific single-agent Tileworld environments (Pollack & Ringuette 1990; Kinny & Georgeff 1991)), or on a small number of performance criteria which can only be interpreted with respect to the particular architecture being measured (for example, the total number of behaviors communicated between agents in selected MICE environments (Durfee & Montgomery 1990)) — combine to make detailed *quantitative* comparisons with other architectures extremely difficult if not altogether impossible.

Due to the relatively large number of parameters which the TouringWorld testbed provides for specifying different agent configurations, performance evaluation criteria (for example, task completion time, resource utilization), and agent task and environmental characteristics, the evaluation performed was necessarily partial, the main focus being placed on studying selected *qualitative* aspects of TouringMachine behavioral ecology — namely, some of the effects on agent behavior which, in a given task environment, can occur through varying individual agent configuration parameters; and the effects on agent behavior which, for a given agent configuration, can occur through varying certain aspects of the agent's environment. Like with the Tileworld experiments described by Pollack and Ringuette (Pollack & Ringuette 1990, page 187), a number of TouringWorld "knobs" (for example, world clock timeslice size, total per-timeslice resources available to each agent, agent size, agent speed and acceleration/deceleration rate limits, agent sensing algorithm, initial attention focussing heuristics, reactive rule thresholds, plan schema and model template library entries) were set to provide "baseline" environments which would be dynamic, somewhat unpredictable, and moderately paced. In such environments, a competent (suitably configured) agent *should* be able to complete all of its goals, more or less according to schedule; however, under certain environmental conditions and/or agent parametrizations this will not always be the case. Details on the evaluation of TouringMachines can be found elsewhere (Ferguson 1992).

5 Discussion

Apart from matters arising directly from the evaluation process described above, a number of experiential and implementational issues which bear on the applicability and appropriateness of the TouringMachine architecture also merit addressing at this point. As mentioned above, the first stage in designing the TouringMachine architecture involved an analysis of the intended TouringMachine task environment: that is, a characterization of those aspects of the intended environment which would most significantly constrain the TouringMachine agent design. As will now be argued, the main purpose of this analysis was to differentiate between, and therefore establish, what Cohen (Cohen 1991) terms the system's *fixed* and *reviewable* design decisions.

Fixed design decisions are those which "will not be reviewed anytime soon" (Cohen 1991, page 31). In the enabling the user to specify, visualize, measure, and analyze any number of user-customized agents in a variety of single- and multi-agent settings, the testbed provides a powerful platform for the empirical study of autonomous agent behavior.

A number of experiments have been carried out on TouringMachines which illustrate, in particular, that the balance between goal-orientedness (effectiveness) and reactivity (robustness) in agents can be affected by a number of factors including, among other things, the level of detail involved in the predictions agents make about each other, the degree of sensitivity they demonstrate toward unexpected events, and the proportion of total agent resources that are made available for constructing plans or building BDI models of other agents. Other experiments point toward a trade off between the reliability and the efficiency of the predictions an agent can make about the future — this turns out to be an instance of the well-known *extended prediction problem* (Shoham & McDermott 1990). Yet other experiments have been carried out which suggest that predicting future world states through causal BDI modelling of agents' mental states, can, in certain situations, prove useful for promoting effective coordination between agents with conflicting goals.

4.1 Some Methodological Issues

One useful approach toward understanding the reasons for the behaviors exhibited by the TouringMachine agent design — and, more specifically, for identifying the conditions under which one configuration of the architecture performs better than another — is to vary the environment in which it operates. The simplest approach to this issue, Langley (Langley 1988) argues, involves designing a set of benchmark problems, of which some, for the purposes of scientific comparison (that is, for the purposes of enabling independent variation of different task environment attributes), should involve artificial domains. The TouringWorld environment is one such domain; other examples include the Phoenix environment (Cohen *et al.* 1989), the Tileworld (Pollack & Ringuette 1990), and MICE (Durfee & Montgomery 1990).

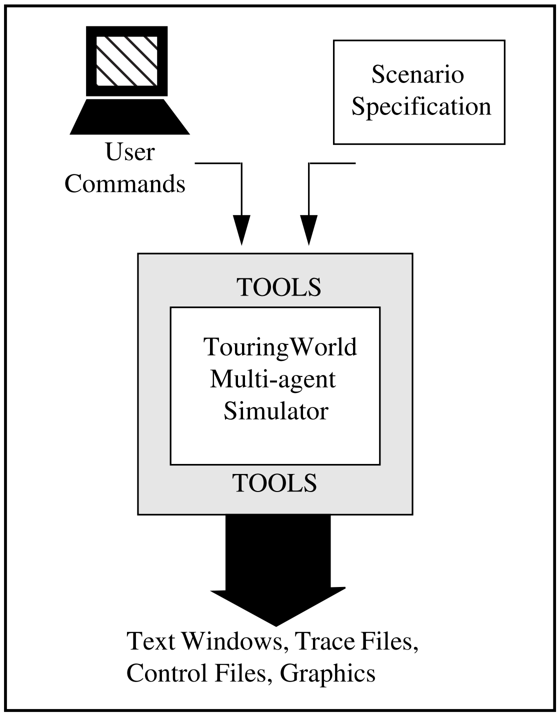
The power of the TouringWorld testbed domain, and of artificial domains in general, arises from the insights it can provide toward the improved understanding of agent — in this case, TouringMachine — behavioral ecology: in other words, the understanding of the functional relationships that exist between the designs of agents (their internal structures and processes), their behaviors (the tasks they solve and the ways in which they solve these tasks), and the environments in which they are ultimately intended to operate (Cohen *et al.* 1989).

The characterization of TouringMachines as a study of agent behavioral ecology exemplifies a research methodology which emphasizes complete, autonomous agents and complex, dynamic task environments. Within this methodological context, the focus of the present evaluation has been centered on two particular research tasks. Cohen *et al.* (Cohen *et al.* 1989) refer to these as *environmental analysis*, in other words, understanding what characteristics of the environment most significantly constrain agent design; and the *design task*, in other words, understanding which agent design or configuration produces the desired behaviors under the expected range of environmental conditions.

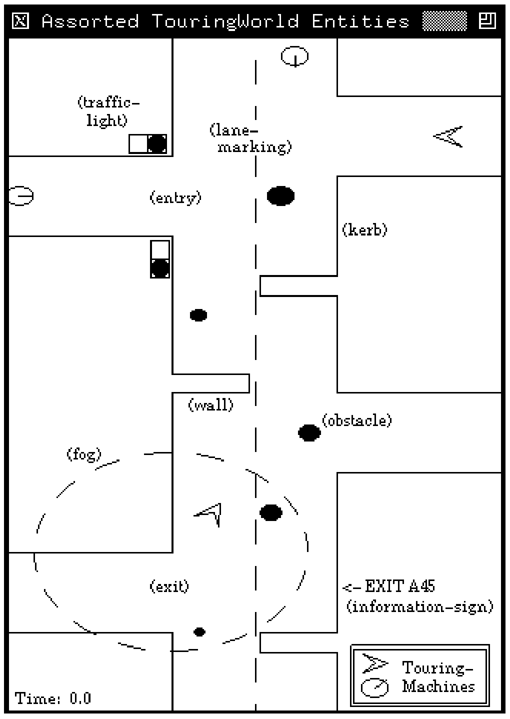
These two tasks, in fact, are the first two stages of a more complete research methodology which Cohen (Cohen 1991) refers to as the *MAD* methodology, for *modelling*, *analysis*, and *design*.[3] This methodology aims to justify system design (and re-design) decisions with the use of predictive models of a system's behaviors and of the environmental factors that affect these system behaviors (more on this below). Like IRMA agents in the Tileworld domain (Pollack & Ringuette 1990), TouringMachine agents can be viewed as having been developed via an incremental version of MAD, in which the (causal) model of TouringMachine behavior is developed incrementally, at the same time as the agent design. In other words, the agent design (or some part of its design) is implemented as early as possible, in order to provide empirical data (or feedback) which flesh out the model, which then become the basis for subsequent redesign (Cohen 1991). The implications of adopting such a design method, as well as the roles played in this method by the environmental and behavioral analyses referred to above, are discussed in detail elsewhere (Ferguson 1992).

The evaluation of TouringMachines was realized through a series of interesting task scenarios involving one or more agents and/or zero or more obstacles or

3. The remaining design activities — *predicting* how the system (agent) will behave in particular situations, *explaining* why the agent behaves as it does, and *generalising* agent designs to different classes of systems, environments, and behaviours — were beyond the scope of this work. See Cohen (Cohen 1991, pages 29—32) for details.



(a)



(b)

Fig. 3. (a) Top-level view of the TouringWorld multi-agent testbed. **(b)** A snapshot of a particular TouringWorld scenario showing various types of (labelled) objects and agents.

deliberative and non-deliberative action control mechanisms.

In addition to addressing the conjecture concerning the suitability of a hybrid control approach, the proposed control architecture was intended to address the importance of and need for extensive empirical evaluation of integrated agent architectures, not merely in terms of the per-agent, task-oriented criteria identified earlier in this section (autonomy, effectiveness, robustness, and flexibility), but also in terms of the controlled agents' behavioral ecology (Cohen *et al.* 1989) — that is, in terms of the functional relationships between agent design (agents' internal structures and processes), agent behavior (the choice of tasks to be solved and the manner in which they are solved), and environmental characteristics. To address these issues, a highly parametrized and instrumented multi-agent simulation testbed was implemented in conjunction with the TouringMachine control architecture. Enabling controlled, repeatable experimentation and facilitating the creation of diverse single- and multi-agent task scenarios, the TouringWorld Testbed is described in more detail elsewhere (Ferguson 1992). For now some comments on the experimental method employed must suffice.

4 Experimenting with TouringMachines

The TouringWorld multi-agent simulation testbed provides the user with a 2-dimensional world which is occupied by, among other things, multiple TouringMachines, obstacles, walls, paths, and assorted information signs. World dynamics are realized by a discrete event simulator which incorporates a plausible world updater for enforcing "realistic" notions of time and motion, and which creates the illusion of concurrent world activity through appropriate action scheduling. Other processes (see Fig. 3.a) handled by the simulator include a facility for tracing agent and environmental parameters, a statistics gathering package for agent performance analysis, a mechanism enabling the testbed user to control the motion of a chosen agent, a declarative specification language for defining the agents to be observed, and several text and graphics windows for displaying output (see Fig. 3.b). By

3.1 Limitations of Pure Non-deliberative Control

The strength of purely non-deliberative architectures lies in their ability to identify and exploit *local* patterns of activity in their current surroundings in order to generate more or less hardwired action responses (using no memory or predictive reasoning, and only minimal state information) for a given set of environmental stimuli. Successful operation using this method of control presupposes: *(i)* that the complete set of environmental stimuli required for unambiguously determining subsequent action sequences is always present and readily identifiable — in other words, that the agent's activity can be strictly *situationally determined*; *(ii)* that the agent has no *global* task constraints — for example, explicit temporal deadlines — which need to be reasoned about at run-time; and *(iii)* that the agent's goal or desire system is capable of being represented *implicitly* in the agent's structure according to a fixed, pre-compiled ranking scheme.

Situationally determined behavior will succeed when there is sufficient local constraint in the agent's environment to determine actions that have no irreversibly detrimental long-term effects. Only then, as Kirsh (Kirsh 1991) argues, will the agent be able to avoid representing alternative courses of actions to determine which ones lead to dead ends, loops, local minima, or generally undesirable outcomes. It follows, then, that if the agent's task requires knowledge about the environment which is not immediately available through perception and which can, therefore, only be obtained through some form of inference or recall, then it cannot truly be considered situationally determined. Kirsh (Kirsh 1991) considers several such tasks, a number of which are pertinent to the TouringWorld domain: activities involving other agents, as these often require making *predictions* of their behavior and reasoning about their plans and goals (Davis 1990, page 395); activities which require responding to events and actions beyond the agent's current sensory limits (such as taking precautions now for the future or when tracking sequences of behaviors that take place over extended periods of time); as well as activities which require some amount of reasoning or problem solving (such as calculating a shortest route for navigation). The common defining feature of these tasks is that, besides requiring reliable and robust *local* control to be carried out, they also possess a non-local or *global* structure which will need to be addressed by the agent. For instance, to carry out a navigation task successfully in the TouringWorld an agent will need to coordinate various locally constrained (re-)actions such as slowing down to avoid an obstacle or slower moving agent with other more globally constrained actions such as arriving at a target destination within some pre-specified deadline.

While non-deliberative control techniques ensure fast responses to changing events in the environment, they do not enable the agent's action choices to be influenced by deliberative reasoning. In most non-deliberative architectures, the agent's goals are represented implicitly — in effect, embedded in the agent's own structure or behavioral rule set. When goals are not represented explicitly, Hanks and Firby (Hanks & Firby 1990) argue, they will not be able to be changed dynamically and there will be no way to reason about alternative plans for carrying them out. Maes (Maes 1990) also argues that without explicit goals it is not clear how agents will be able to learn or improve their performance.

Complex agents will need complex goal or desire systems — in particular, they will need to handle a number of goals, some of which will vary in time, and many of which will have different priorities that will vary according to the agent's situational needs. The implications of this, Kirsh (Kirsh 1991) argues, is that as agents' desire systems increase in size, there will be a need for some form of desire management, such as deliberation, weighing competing benefits and costs, and so on.

There are undoubtedly a number of real-world domains which will be suitable for strictly non-deliberative agent control architectures. It is less likely whether there exist any realistic or non-trivial domains which are equally suited to purely deliberative agents. What is most likely, however, is that the majority of real-world domains will require that intelligent autonomous agents be capable of a wide *range* of behaviors, including some basic non-deliberative ones such as perception-driven reaction, but also including more complex deliberative ones such as flexible task planning, strategic decision-making, complex goal handling, or predictive reasoning about the beliefs and intentions of other agents.

A central goal of the research presented here was to demonstrate that it is both *desirable* and *feasible* to combine suitably designed deliberative and non-deliberative control functions to obtain effective, robust, and flexible behavior from autonomous, task-achieving agents operating in complex environments. The arguments put forward so far have attempted both to outline some of the broader functional and behavioral requirements for intelligent agency in complex task domains like the TouringWorld, and also to justify a hybrid control approach that integrates a number of plete their tasks, be able to *coordinate* their activities with other agents that they might encounter: that is, they should be capable of cooperation.[2]

- A TouringMachine should be **robust** to unexpected events. Successful operation in a real-time dynamic environment will require that TouringMachines be able to identify and handle — in a timely manner — a host of unexpected events at execution-time. For many events (such as the sudden appearance of a path-blocking obstacle) an agent will have little or no time to consider either what the full extent of its predicament might be or what benefits consideration of a number of different evasive maneuvers might bring. In order to cope with such events, TouringMachines will need to operate with guaranteed responsiveness (for example, by using latency-bounded computational and execution techniques) as well as being fairly closely-coupled to their environments at all times. Since the time and location of such events will be unpredictable, TouringMachines will need to monitor their surroundings continually throughout the course of their goals.
- A TouringMachine should be **flexible** in the way it carries out its tasks. Due to the dynamic and unpredictable nature of the TouringWorld environment, and the fact that its multiple inhabitants must operate in real time with limited world knowledge, TouringMachines will inevitably be faced with various belief and/or goal conflict situations arising from unforeseen interactions with other agents. Agents operating cooperatively in complex domains must have an understanding of the nature of cooperation. This, Galliers (Galliers 1990) argues, involves understanding the nature and role of multi-agent conflict. To behave flexibly and to adjust appropriately to changing and unpredicted circumstance, TouringMachines should be designed to recognize and resolve unexpected conflicts rather than to avoid them. Also, for the purposes of control and coordination, TouringMachines must be able to reason about their own and other agents' activities. In this respect, each TouringMachine must have the capacity to *objectify* particular aspects of the world — that is, to construct and deploy internal models of itself and of other agents — to see where it fits in the coordinated process and what the outcomes of its own actions might be (Bond & Gasser 1988, page 25).

2. Following Bond and Gasser (Bond & Gasser 1988, page 19), cooperation in the TouringWorld is viewed simply as a special case of coordination among *non-antagonistic* agents. While TouringMachines are not actually benevolent (they are selfish with respect to their own goals and have the ability to drop or adopt different intentions according to their own preferences and situational needs) they are also not antagonistic since they do not intentionally try to deceive or thwart the efforts of other TouringMachines.

Although much of the above functionality could be described as deliberative (for example, reasoning about the temporal extent of actions, conflict resolution, reflexive modelling), it is unclear whether a strictly deliberative control approach based on traditional planning techniques would be adequate for successful operation in the TouringWorld domain. Most classical planners make a number of important simplifying assumptions about their domains which cannot be made about the TouringWorld: namely, that the environments remain static while their (often arbitrarily long) plans are generated and executed, that all changes in the world are caused by the planner's actions alone, and that their environments are such that they can be represented correctly and in complete detail. Given that the TouringWorld is dynamic and multi-agent and given that TouringMachines also have inherently limited physical and computational means for acquiring information about their surroundings, it seems clear that a strictly traditional planning approach to controlling TouringMachines would be unsuitable. Also, while it is true that planning systems capable of execution monitoring and interleaved planning and execution represent a significant advance on the earlier traditional planners, their usefulness in a highly dynamic and real-time domain like the TouringWorld is questionable, particularly given the reservations expressed by Georgeff (Georgeff 1990) and Bratman *et al.* (Bratman, Israel, & Pollack 1988) concerning their computational efficiency and inability to cope with situationally-varying time constraints.

Similarly, while the inclusion of at least some degree of non-deliberative control in TouringMachines would seem essential — particularly since the agents will need to be closely coupled to their environment, robust to unexpected events, and able to react quickly to unforeseen events and operate with guaranteed levels of responsiveness — it is questionable whether non-deliberative control techniques *alone* will be sufficient for providing TouringMachines with the complete behavioral repertoire necessary for successful operation in the TouringWorld environment. This argument deserves closer consideration.

agent's high-level tasks (e.g. planning, causal modelling, counterfactual reasoning) is sensitive also to its low-level, high-priority behaviors such as avoiding collisions with other agents or obstacles.

3 Hybrid Architectures: a Rationale

An autonomous agent operating in a complex environment is constantly faced with the problem of deciding what action to take next. As Hanks and Firby (Hanks & Firby 1990) point out, formulating this problem precisely can be very difficult since it necessitates consideration of a number of informational categories which are often difficult to ascertain — for example, the benefits and costs to the agent of executing particular actions sequences; or which have been demonstrated from previous research to be problematic to represent — for example, models of agents' beliefs and desires about a world which is complex and unpredictable.

The control problem in an agent is the problem of deciding how to manage these various sources of information in such a way that the agent will act in a competent and effective manner, with respect to its own resources. This problem, Hanks and Firby (Hanks & Firby 1990) suggest, amounts to balancing two "reasonable" approaches to acting in the world: the first, deliberation, involves making as many decisions as possible as far ahead of time as possible; the second approach, reaction, is to delay making decisions as long as possible, acting only at the last possible moment. At a glance, the first approach seems perfectly reasonable since, clearly, an agent which can think ahead will be able to consider more options and thus, with forethought, be more informed when deciding which action to take. On the other hand, since information about the future can be notoriously unreliable and, in many real-world situations, difficult or even impossible to obtain given the agents' changing time constraints, it would also seem reasonable that acting at the last moment should be preferred. In fact, except perhaps for a small number of special-case task domains, it would seem much more reasonable to assume that neither approach — deliberation or reaction — should be carried out to the full exclusion of the other.

TouringMachines are autonomous, mobile agents which are capable of rationally carrying out one or more tasks in dynamic, real-time, multi-agent domains. In particular, TouringMachines have to date been studied in a complex multi-agent traffic navigation domain — the *TouringWorld*. The tasks performed by TouringMachines are prioritized in advance by the agent's designer and include goals like avoiding collisions with other mobile agents and fixed obstacles, obeying a commonly accepted set of traffic regulations, and also, as mentioned above, relocating from some initial location to some target destination within certain time bounds and/or spatial constraints. Besides being limited in terms of its internal computational resources, each TouringMachine will start out with only limited knowledge of its world: in particular, although each TouringMachine possesses a topological map of the various paths and junctions defining all of the navigable routes in the world, it will have no prior knowledge regarding other agents' locations or goals or the obstacles it might encounter en route. In addition, each TouringMachine has limited means for monitoring and acquiring information from its surroundings and will be restricted in its capacity to communicate with other agents: intentions to turn or overtake are communicated via primitive signalling alone, much like a human driver does in a car.

The goal of this research was to produce an integrated control architecture which would enable TouringMachines to carry out tasks and act on their environments autonomously and in accordance with a set of domain-specific evaluation criteria; namely, effectiveness, robustness, and flexibility. These criteria suggest a broad range of behavioral and functional capacities that each TouringMachine might need to possess:

- A TouringMachine should be capable of **autonomous** operation. Operational autonomy requires that the agent have its own *goals* and be able to select among these as and when required. In addition, as Covrigaru and Lindsay (Covrigaru & Lindsay 1991) argue, the agent should, among other things, be capable of interacting with its environment, be able to move (preferably fluidly) around its environment, have selective attention (this is also desirable since TouringMachines have limited computational resources), have a varied behavioral repertoire, and have differential responsiveness to a variety of environmental conditions.
- A TouringMachine should carry out its goals in an **effective** manner. Effective goal achievement requires that the agent be capable of carrying out its multiple tasks in an efficient and timely manner. Since among its various tasks, a TouringMachine must navigate along some route within a pre-specified time limit, the agent should be able to reason predictively about the temporal extent of its own actions. Also, because TouringMachines will operate in a partially-structured multi-agent world, they should, in order to com-

```
censor-rule-1:
    if   entity(obstacle-6) ∈ Perception-Buffer
    then
         remove-sensory-record(layer-ℛ, entity(obstacle-6))

suppressor-rule-3:
    if   action-command(layer-ℛ-rule-6*,
                      change-orientation(_))† ∈ Action-Buffer
         and
         current-intention(start-overtake)
    then
         remove-action-command(layer-ℛ, change-orientation(_))
         and
         remove-action-command(layer-ℳ, _)
```

* `layer-ℛ-rule-6` is the reactive (layer $\mathcal{R}$) rule which is invoked in order to avoid crossing a path lane marking.

† "_" simply denotes a don't-care or anonymous variable.

Fig. 2. Two example control rules: **censor-rule-1** and **suppressor-rule-3**.

agent in front of it.

Inputs to and outputs from layers are generated in a synchronous fashion, with the context-activated control rules being applied to these inputs and outputs at each synchronization point. The rules, thus, act as filters between the agent's sensors and its internal layers (*suppressors*), and between its layers and its action effectors (*censors*) — in a manner very similar to Minsky's suppressor- and censor-agents (Minsky 1986). Both types of rules are of the *if-then* condition-action type. In the case of censor rules, the conditional parts are conjunctions of statements that test for the presence of particular sensory objects recently stored in the agent's Perception Subsystem. Censor rules' action parts consist of operations to prevent particular sensory objects from being fed as input to *selected* control layers. In Fig. 2, for example, the censor rule **`censor-rule-1`** is used to prevent layer $\mathcal{R}$ from perceiving (and therefore, from reacting to) a particular obstacle which, for instance, layer $\mathcal{M}$ might have been better programmed to deal with. In the case of suppressor control rules, conditional parts are conjunctions of statements which, besides testing for the presence of particular outgoing action commands in the agent's Action Subsystem, can also test the truth values of various items of the agent's current internal state — in particular, its current beliefs, desires, and intentions. Suppressor rules' action parts consist of operations to prevent particular action commands from being fed through to the agent's effectors. In Fig. 2, for example, the suppressor control rule **`suppressor-rule-3`** is used to prevent layer $\mathcal{R}$ from reacting to (steering away from) a lane marking object whenever the agent's current intention is to overtake some other agent that is in front of it.

Any number of censor control rules can fire (and remove selected control layer input) when these are applied at the beginning of a synchronization timeslice. Suppressor control rules, on the other hand, are assumed to have been crafted by the agent's programmer in such a way that *(i)* at most one will fire in any given situational context (an agent's situational context is taken to be the combination of its perceptual input set and its current internal BDI state); and *(ii)* at most one action command will remain in the Action Subsystem after the suppressor control rule's action part has been executed. By crafting suppressor control rules in this way, a TouringMachine's effectors can be guaranteed to receive no more than one action command to execute during any given timeslice.

Mediation remains active at all times and is largely "transparent" to the layers: each layer acts as if it alone were controlling the agent, remaining largely unaware of any "interference" — either by other layers or by the rules of the control framework — with its own inputs and outputs. The overall control framework thus embodies a real-time opportunistic scheduling regime which, while striving to service the

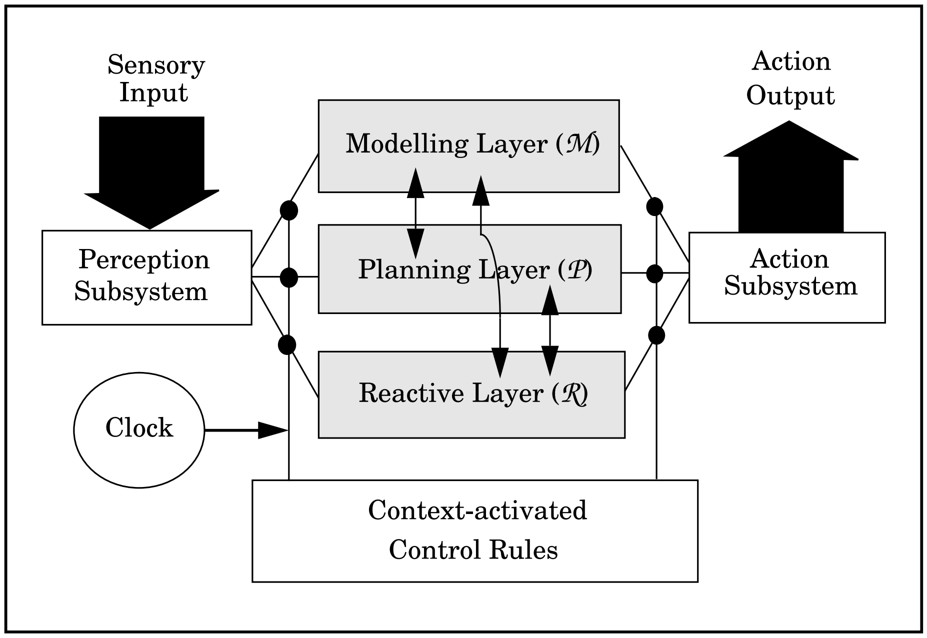

Sensory Input
Action Output
Modelling Layer ($\mathcal{M}$)
Perception Subsystem
Planning Layer ($\mathcal{P}$)
Action Subsystem
Reactive Layer ($\mathcal{R}$)
Clock
Context-activated Control Rules

Fig. 1. A TouringMachine's mediating control framework.

tecture has been designed through integrating a number of reactive and *suitably designed* deliberative control functions. Before addressing aspects of the rationale behind this design, a brief description of the architecture's main features and operation will follow.

2 TouringMachines

Implemented as a number of concurrently-operating, latency-bounded, task-achieving control layers, the TouringMachine architecture is able to produce a number of reactive, goal-directed, reflective, and predictive behaviors — as and when dictated by the agent's internal state and environmental context. In particular, TouringMachines (see Fig. 1) comprise three such independently motivated layers: a *reactive* layer $\mathcal{R}$ for providing the agent with fast, reactive capabilities for coping with events its higher layers have not previously planned for or modelled (a typical event, for example, would be the sudden appearance of some hitherto unseen agent or obstacle); a *planning* layer $\mathcal{P}$ for generating, executing, and dynamically repairing hierarchical partial plans (which are used by the agent, for example, when constructing navigational routes to some target destination); and a reflective-predictive or *modelling* layer $\mathcal{M}$ for constructing behavioral Belief-Desire-Intention (BDI) models of world entities, including the agent itself, which can be used as a platform for explaining observed behaviors and making predictions about possible future behaviors (Ferguson 1995).

Each control layer is designed to model the agent's world at a different level of abstraction and each is endowed with different task-oriented capabilities. Also, because each layer directly connects world perception to action and can independently decide if it should or should not act in a given state, frequently one layer's proposed actions will conflict with those of another; in other words, each layer is an approximate machine and thus its abstracted world model is necessarily incomplete. As a result, layers are mediated by an enveloping control framework so that the agent, as a single whole, may behave appropriately in each different world situation.

Implemented as a combination of inter-layer message passing and context-activated, domain-specific control rules (see Fig. 1), the control framework's mediation enables each layer to examine data from other layers, inject new data into them, or even remove data from the layers. (The term *data* here covers sensed input to and action output from layers, the contents of inter-layer messages, as well as certain rules or plans residing within layers.) This has the effect of altering, when required, the normal flow of data in the affected layer(s). So, in a road driving domain for example, the reactive rule in layer $\mathcal{R}$ to prevent an agent from straying over lane markings can, with the appropriate control rule present, be overridden should the agent embark on a plan to overtake the

Appears in Proc. AAAI Spring Symposium on Lessons Learned from Implemented Software Architectures for Physical Agents, Palo Alto, CA, March, 1995

Integrating Models and Behaviors in Autonomous Agents: Some Lessons Learned on Action Control

Innes A. Ferguson[1]
Knowledge Systems Laboratory
Institute for Information Technology
National Research Council
Ottawa ON, Canada K1A 0R6
innes@ai.iit.nrc.ca

Abstract

This paper describes an architecture for controlling autonomous agents, building on previous work addressing reactive and deliberative control methods. The proposed multi-layered architecture allows a resource-bounded, goal-directed agent to reason predictively about potential conflicts by constructing causal theories which explain other agents' observed behaviors and hypothesize their goals and intentions; at the same time it enables the agent to operate autonomously and to react promptly to changes in its real-time physical environment. A number of criteria which influenced the design and implementation of the architecture, in particular its action control component, are also discussed.

1 Introduction

An integrated agent architecture, Drummond and Kaelbling (Drummond & Kaelbling 1990) suggest, is a theory or paradigm by which one may design and program intelligent agents. Typically targeted for use in dynamic, unpredictable, and often multi-agent environments, an intelligent agent can be regarded as a structured collection of sensors, computers, and effectors; in this structure, the sensors measure conditions in the world, the computers process the sensory information, and the effectors take action in the world. Since changes in the world realized by the agent's effectors will close the loop to the agent's sensors, the agent can be described as being embedded in its environment.

A number of different integrated architectures have been proposed recently, each one aimed at providing agents with a particular level of intelligent, autonomous control. Broadly speaking, the different approaches can be classified according to the mechanism for action control or selection which the agent uses when determining what to do next. In particular, if the agent selects actions by explicitly deliberating upon the various options that are present (for example, with the use of an internal symbolic world model, via a search of its plan space, or by considering the expected utility of available execution methods) the agent can be considered *deliberative* (Durfee & Montgomery 1990; Shoham 1990; Vere & Bickmore 1990). Alternatively, if the agent's choice of action is situationally determined — in other words, pre-programmed or in some way "hardwired" to execute given the occurrence of a particular set of environmental conditions — then they can be described as *non-deliberative* or *reactive* (Agre & Chapman 1987; Brooks 1991; Maes 1994).

Now, while intelligent agents must undoubtedly remain reactive in order to survive, some amount of strategic or predictive decision-making will also be required if agents are to handle complex goals while keeping their long-term options open. On the other hand, agents cannot be expected to model their surroundings in every detail as there will simply be too many events to consider, a large number of which will be of little or no relevance anyway. Not surprisingly, it is becoming widely accepted that neither purely reactive nor purely deliberative control techniques are capable of producing the range of robust, flexible behaviors desired of future intelligent agents. What is required, it would seem, is a *hybrid* architecture that can cope with uncertainty, react to unforeseen incidents, and recover dynamically from poor decisions. All of this, of course, on top of accomplishing whatever tasks it was originally assigned to do.

This paper is concerned with the design and implementation of an integrated agent control architecture, the *TouringMachine* architecture (Ferguson 1992; Ferguson 1994; Ferguson 1995), suitable for controlling and coordinating the actions of autonomous rational agents embedded in a partially-structured, dynamic, multi-agent world. Upon carrying out an analysis of the intended TouringMachine task domain — that is, upon characterizing those aspects of the intended real-time physical navigation domain that would most significantly constrain the TouringMachine agent design — and after due consideration of the requirements for producing autonomous, effective, robust, and flexible behaviors in such a domain, the TouringMachine archi-

1. This research was conducted while the author was a doctoral candidate at the Computer Laboratory, University of Cambridge, Cambridge, UK.