Collaborative Learning Agents

Cherif Branki * **Alan Bridges **** **Andrew Wallis***

** Department of Computer Studies, Glasgow Caledonian University, UK.*

*** Department of Architecture and Building Science, University of Strathclyde, UK*

Email: cbr@uk.ac.gcal

ABSTRACT: In this position paper we will explore the intersection between machine learning in design and agent technology by discussing a framework for a collaborative agents learning group which may provide new insights into machine learning in design. Agents are very complex and hard to specify in both dynamics and behaviour and it is therefore commonly agreed they should be able to learn, in order to improve their performance.

1. Machine Learning

In multi-agent systems, there are two important reasons to introduce machine learning [GRECU 95]: to adapt the system to the domain (and to the problems it has to solve), and to adapt the agents to one another, and this is the main issue of this paper. Learning in multi-agent systems must therefore target both domain and interaction knowledge in agents. It is generally accepted that students learn more efficiently if they are able to discuss what they have learnt individually amongst themselves, similarly it is to be hoped that integrating the learning of multiple agents in a system would produce similar benefits. Various coordination strategies have been implemented to address this issue, such as:

- all agents collect data, but all data is passed to a central learning agent [SCHLIMMER et al 88] .
- several agents learn, but do not communicate during the learning process; the contrasting knowledge of these agents is then taken into account during problem solving. [GAMS 89] shows this model achieves superior performance compared to single knowledge bases.
- each agent learns individually but solicits the opinions of other agents on the hypotheses it forms, with the objective of achieving consensus [SIAN 91]. This has the advantage of allowing early testing of hypotheses.
- agents learn individually, then integrate that knowledge ('distributed re-learning'). This integration gives significant improvements in performance over individualised learning [BRAZDIL et al 91]. This is the method used within the CAL Group (see section 3.2).

2. Agent Technology

The idea of an agent can be attributed to the work of both [MCCARTHY and SELFRIDGE 59]. The term “agent” is not used by all researchers in the same way. Some apply the term entity, human or automatic, that is capable of taking some initiative, whereas others restrict its use to computer based processes that act on behalf of human users or software processes. Despite terminology variations, however, the

core notion of an intelligent system possessed of the ability to carry out useful tasks without lavish prompting by an other is common to most concepts of agents [CONNOLY 94]. However, attempts to implement agents are still relatively new, and combine two fields: Artificial Intelligence and Human Computer Interaction.

The AI approach views agents as entities within a larger community of agents. This position forms the basis of Distributed AI (DAI) where agents are viewed as entities with different skills and knowledge, but essentially part of a community or a society [SMYTH 94]. The aim of DAI is to construct intelligent agents that make decisions for themselves allowing them to achieve their goals in a world populated by other agents with their own goals [CHAIB-DRAA, 1993].

In the Human Computer Interaction approach, the study of agents is focused on the necessity of agents to interact with the users at the level of the interface. They understand the users' needs and goals. Goals can be derived either implicitly, by monitoring the current activity state and its associated history, or explicitly, based on specific instruction by the user. Agents translate those goals into a set of actions, which will be undertaken independently of the user and will then deliver results in a form that is amenable to the user and pertinent to the current task.

Agent based technologies will adopt a different approach to traditional CAD tools. For example, upon editing a construction file, using your favourite CAD tool, and in order to change construction materials requested by your client to cut down on the overall project cost, your agent will notify you that it has already started consulting a number of construction materials databases available on networked information sources provided by suppliers. A short time later, you are presented with the summary of the cheapest construction materials. Once a decision is made (with your client) on a type of material, your agent will modify your construction file and run a quick environmental simulation available on your CAD tool, to present you with a summary of heat gains or losses as a consequence of choosing that particular material.

3. **An Example in Machine Learning**

The CAL Group framework is a subset of a multi-agent framework [BRANKI 93] that was constructed to support collaborative work by enabling designers to work effectively as a team whilst also using existing CAD applications to assist their individual tasks. Given that architectural design is a multidisciplinary, multicriteria, multiperson activity the CAL Group uses the concept of shared learning [DUFFY 95] not only between a designer and a machine but also between designers themselves, and between agents themselves. The multi-agents framework (Figure 1) models design information and customises it to specific designers' needs.

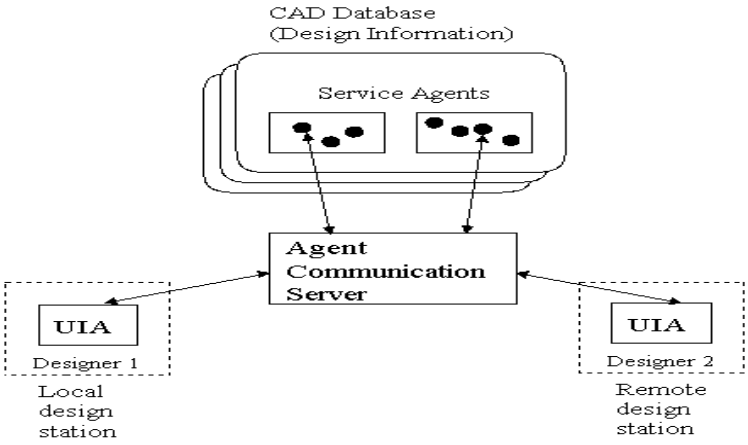

Figure 1: The CAL Group

3.1 User Interface Agent (UIA)

The Interface agent is responsible for managing and monitoring all user interactions and implementing the level of communication as necessary for a given user and task combination. Unlike conventional systems where the interface remains static throughout a session, by keeping track of the user's progress, the interface agent can, using its internal model of users' profiles, tailor the dialogue to the user's level of expertise and performance history.

To perform this function, the interface agent has several mechanisms at its disposal to achieve interaction with the user. The primary mechanism is a generic set of dialogue objects which are presented to the user depending on the type of user and level of expertise. In general, the interface agent is essentially an intelligent communication switch and protocol converter, sitting between a database of interaction objects and the agent communication server. Upon receiving information from the service agent, the interface agent uses the dialogue object to present the designer with examples of past design, or any relevant information that it deems necessary to the designer. The UIAs learn autonomously, i.e. no integration of learning, or learnt knowledge, occurs.

3.2 Service Agents (SA)

The CAD database that the SAs use contain design information which can be re-used in new designs or consulted in order to obtain, for instance, the rationale behind certain design decisions. While such databases contain rich design information which is well structured it is often not easily understood. These agents provide services such as acquiring specific patterns, concept designs requested by the UIAs or when ordered by the designers. SAs use a form of 'redistributed learning' [BRAZDIL et al 90], sharing learned knowledge where appropriate.

3.3 Agent Communication Server (ACS)

An ACS is used by the collaborative agents to share data and knowledge. It supports reliable interaction of agents with shared resources, which include shared knowledge and data items, co-operative objectives and shared computer and communication resources.

The ACS uses a global naming service, peer-peer transaction services protocol, a message queuing protocol and one or more data transfer protocols to provide a set of services to agent frontends that support distributed co-operative tasks and that help control the interactions among the co-operating agents. Communication protocols are selected from international standard and industry standard sources. It supports a set of transparency services providing many of the benefits of distributed computing, including server replication, service migration and concurrency, location independence, and a degree of failure transparency. All the information describing a co-operative design task is held in a persistent message queue, for example. After a malfunction, it can be recovered by the appropriate server, and the task restarted.

The ACS provides both asynchronous and synchronous message delivery services to agents that are distributed among the designers' workstations. Messages flow over logical conversation paths, where path connections may be established between co-operating agents or between queue servers located at designers workstations. In the latter case, connectionless communication service is provided, where agents' messages are enqueued to queues that may reside anywhere within the agents' network environment. Messages are dequeued from queues local to a designer's workstation, either upon agent request or upon receipt of message types that trigger the scheduling of co-operative agents.

Agents may select service options that determine whether message exchange occurs synchronously, and at which priority level; whether explicit responses are required to specific messages; message types that are persistent across system malfunctions; and message context values associated with origin and identity.

Task-oriented dialogues are supported in part through a conversation mapping service, which acts on either or both ends of the conversation path. The mapping function allows additional information about task objectives and resources to be attached to each message that flows over a specific path. Mapping information may be changed at any time by authorised agents, to reflect progress and status of the planning process. The intent of conversation mapping is to allow flexibility in definition of formal mechanisms for communicating ontologies affecting designers' behaviour.

Reliable communication of data and knowledge is supported through alternate path selection, peer-to-peer transactions and message persistence. Alternate paths may be necessary when physical links between geographically dispersed workstations are interrupted or taken out of service, or a new designer wishes to join the co-operative planning session through a communication link that is not in use by the active players. In these cases, the ACS calls upon Network Interface Servers (NIS) for new path connections. Each NIS serves path connections over a single type of logical medium. After the connection is established, NI servers appear to agents as the ACS. This decoupling of logical conversation path use from the conversation services interface presented to the agent means that:

- agents need not be aware of the logical/physical location of player workstations, and
- new networking technologies can be added to the prototype without modification to agent software or to the ACS.

Communication interface decoupling occurs through use of an intermediate Name Server, which steps aside after it brokers connection of an agent's conversation request with the appropriate co-operating agent or message delivery queue.

Message persistence is provided on demand, or preconfigured, by maintaining a copy of the desired message in a backup/recovery message queue. After a system interruption or malfunction is corrected and the workstation is restarted, the message queue can be accessed by agent-cores, just like any other queue, to retrieve and act on interrupted messages.

Peer-to-peer transactions are used between store/forward message queues and between those co-operating agents that require reliable, synchronised communication service. In these cases, each message that is delivered over a conversation path receives confirmation of receipt back to the ACS, together with the standard error recovery and session services that are characteristic of the SNA/LU6.2 transaction services protocol.

Agents may employ different representations of their local knowledge. So that agents can exchange information easily, the ACS is being extended to provide access to shared information objects through a context-free specification and constraint language which is designed to support shared ontologies.

4. Related work

The CAL Group architecture, and specifically the ACS, is similar to that described in [GENESERETH 94], which features a facilitator agent which coordinates the activities of the other agents in the architecture. Facilitators provide such services as agent identification by name or capability, content-based routing, problem decomposition and allocation, and knowledge translation and monitoring. Content-based routing is not provided within the CAL Group architecture, as it is likely to be implemented in (relatively) stable environments, where it can be assumed that providing agents with reliable knowledge about their peers' abilities is straightfoward. We also feel that facilitators, as described in [GENESERETH 94], constitute a significant communications bottleneck; while this may be counter-balanced by greater flexibility in dynamic environments, it is unwelcome in the stable environment in which we see the CAL Group being useful.

5. Summary

The Cal Group is a multi-agent framework supporting collaborative design. A central Agent Communication server coordinates the interaction of Service Agents and User Interface Agents, allowing them to share data and knowledge. This is being extended with a context-free specification and constraint language to support shared ontologies.

6. References

Branki, N.E., Edmonds, EA., Jones, R.M. *A Study of Socially Shared Cognition in Design.* Environment and Planning B, Vol.20, 1993.

Brazdil, P, Gams, M., Sian, S., Torgo, L. and van de Velde, W. *Learning in Distributed Systems and Multi-Agent Environments.* In Proceedings of the 6th European Working Session on Learning (EWSL-91), Kodratoff, Y. (Ed.), Morgan Kaufmann, 1991.

Chaib-Draa, B., Moulin, B., Mandieu, R., *Trends in Distributed Artificial Intelligence*. Artificial Intelligence Review, Volume 6, pp 35-66, 1992.

Connoly, J.H., Edmonds, E.A. *CSCW and Artificial Intelligence: Preface*. Springer Verlag (ed), 1994.

Duffy, S.M., Duffy, A.H.B., MacCallum, K.J. *A Design Reuse Model.* International Conference on Engineering Design ICED'95 Praha, 1995

Gams, M. *The Measurement Highlight the Importance of Redundant Knowledge.* In Proceedings of the 4th European Working Session on Learning (EWSL-89), Morik, K. (Ed.), Pitman - Morgan Kaufmann, 1989.

Genesereth M.R., Singh N.P., Syed M.A., *A Distributed and Anonymous Knowledge Sharing Approach to Software Interoperation* In Proceedings of the Workshop on Intelligent Hypertext, Mayfield J. and Nicholas C. (Eds.), National Institute of Standards, Gaithersburg, MD, 1994.

Grecu, D., Brown, D.C. *Design Agents That Learn*, Artificial Intelligence for Engineering Design, Analysis and Manufacturing, special issue on "Machine Learning in Design", A.H.B. Duffy, M.L.Maher & D.C. Brown Eds, Cambridge University Press, pp. 149-150, 1996

McCarthy, J. *Programs with Common Sense*. In: Sutherland GBBM9ed) proceedings of the Teddington Conference on Mechanisation of Thought Processes. HMSO, London, pp 77-84, 1959.

Schlimmer, J.C. and Fisher, D. *A Case Study of Incremental Concept Induction.* In Proceedings of the Fifth National Conference on Artificial Intelligence, Morgan Kaufmann, 1986.

Selfridge, O.G. *Pandemonium: A Paradigm for Learning*. In: Proceedings of the Teddington Conference on Mechanisation of Thought Processes. HMSO, London, pp 513-526, 1959.

Sian, S. *Extending Learning to Multiple Agents: Issues and a Model for Multi-Agent Machine Learning (MA-ML).* In Proceedings of the 6th European Working Session on Learning (EWSL-91), Kodratoff, Y. (Ed.), Morgan Kaufmann, 1991b.

Smyth, M. *Towards a Cooperative Software Agent*. In CSCW and Artificial Intelligence. Springler Verlag (ed), pp 1-12, 1994.