Cyberoos'2001: "Deep Behaviour Projection" Agent Architecture

Mikhail Prokopenko[1], Peter Wang[1], and Thomas Howard[2]

[1] Intelligent Interactive Technology
CSIRO Mathematical and Information Sciences
Locked Bag 17, North Ryde, NSW 1670, Australia
{mikhail.prokopenko, peter.wang}@cmis.csiro.au
[2] London, U.K.,
thoward@tibco.com

The RoboCup Simulation League presents developers with a variety of challenges. Arguably, the following list includes the most critical topics:

- distributed client/server system running on a network;
- concurrent communication with a medium-sized (25) number of agents;
- fragmented, localised and imprecise (noisy and latent) information about the environment (field);
- heterogeneous sensory data (visual, auditory, kinetic);
- asynchronous perception-action activity;
- limited range of basic commands/effectors (turn, kick, dash, ...);
- limited window of opportunity to perform an action;
- autonomous decision-making under constraints enforced by teamwork (collaboration) and opponent (competition);
- conflicts between reactivity and deliberation;
- no centralised controllers (no global vision, etc.);
- evolving standards, rules and parameters of the Simulation.

Clearly, these challenges are not uniform and may require a complex agent architecture, covering not only simple situated behaviour but also more involved strategic skills. Pure behavioural approaches to artificial intelligence often feature situatedness of agents reacting to changes in environment and exhibiting emergent behaviour, instead of reliance on abstract representation and inferential reasoning [2,3]. Tactical and strategic reasoning, however, would seem to require domain knowledge and a degree of multi-agent cooperation beyond the reach of situated behaviour-based agents. Over the last few years, it has become apparent that a unifying architecture, combining cognitive (top-down) and reactive (bottom-up) approaches, cannot be achieved by simply connecting higher and lower layers. It has been suggested in recent literature that a "middle-out" architecture [1] is required. The approach adopted in [1] follows the *behavioural programming* principle (and the situated automata framework [3]) in "taking symbolic descriptions of tasks and predictably translating them into dynamic descriptions that can be composed out of lower-level controllers".

Our principal target is a systematic description of increasing levels of agent reasoning abilities, where a given behaviour can "grow" into an expanded level, while every new level can be projected onto a deeper (more basic) behaviour. More precisely, we develop a scalable *Deep Behaviour Projection* (DBP) agent architecture, and enhance Cyberoos agents incrementally. The DBP framework introduced in our previous work [9] achieves:

– realisation of the behavioural programming principle (translating symbolic descriptions into the low-level specifications or reactive behaviours);
– encapsulation of an emergent behaviour (a feedback between the emergent behaviour and symbolic descriptions);
– behaviour depth or duplication (both embedded and emergent levels), unlike "shallow" behaviour subsumption;
– behaviour projection and resultant functional interchangeability, mediated internally within the agent.

The feedback connection from an emergent behaviour to a (meta-)action theory, and then to a derived behaviour on a higher level, extends the behavioural programming principle. Thus, a behaviour can be present in the architecture in two forms: implicit (emergent) and explicit (embedded). This duplication (or depth) allows the agent to "mediate" among related behaviours. The results reported in [6,7,8,9] formalise the DBP approach at the situated and tactical levels and introduce new systematic models and formal correctness results.

The developed hierarchy can be briefly summarised as follows:

$$
\begin{array}{rlll}
\langle C, S, E, & \mathit{sense} : C \rightarrow S, & & \mathit{response} : E \rightarrow C, \\
 & \mathit{timer} : C \rightarrow S, & & \mathit{command} : S \rightarrow E, \\
 & \mathit{tropistic\_behaviour} : S \rightarrow E, & & \\
I, & \mathit{hysteretic\_behaviour} : I \times S \rightarrow E, & & \mathit{update} : I \times S \rightarrow I, \\
D, & \mathit{domain\_update} : I \times S \times D \rightarrow D, & & \\
 & \mathit{domain\_revision} : I \times S \times D \rightarrow D, & & \\
 & \mathit{domain\_projection} : I \times S \times D \rightarrow I, & & \\
T, & \mathit{decision} : I \times S \times T \rightarrow T, & & \mathit{combination} : T \rightarrow 2^H, \\
P, & \mathit{engage} : I \times S \times T \times P \rightarrow P, & & \mathit{tactics} : P \rightarrow 2^T \ \rangle
\end{array}
$$

where C is a communication channel type, S is a set of agent sensory states, E is a set of agent effectors, I is a set of internal agent states, D is a domain model, T is a set of agent task states, P is a set of agent process states, and H denotes the set of instantiations $\{(i, s, e) : e = \mathit{hysteretic\_behaviour}(i, s)\}$.

The following list illustrates the hierarchy with informal examples from soccer, while highlighting some obvious biological parallels:

– tropistic behaviour: Sensors $\rightarrow$ Effectors (obstacle avoidance, chase, goalkeeper catch; *plants following sunlight, spiders catching prey*);
– hysteretic behaviour: Sensors & Memory $\rightarrow$ Effectors (intercept, resultant pass, dribble; *movement of a school of fish, ants tracing pheromones*);
– task-oriented behaviour: Sensors & Memory & Task $\rightarrow$ Effectors (defensive blocks, backing-up, maintaining triangles; *lions hunting, patrolling territory*);
– domain-oriented memory projection: Sensors & Memory & Domain $\rightarrow$ Memory, followed by behaviour-based actions ("blind" pass, off-side check; *whales communication during a hunt*).

The Deep Behaviour Projection framework underlies this hierarchy and ensures that more advanced levels capture relevant behaviour more concisely than their deeper projections. Our primary application domain is the RoboCup Simulation — an artificial multi-agent world [4], where the DBP framework provided a systematic support for design and full implementation of Cyberoos [6,8]. Previous generations of Cyberoos developed under the DBP approach, captured certain types of situated behaviour (Cyberoos'98) and some basic classes of tactical behaviour (Cyberoos'99). Cyberoos'2000 exhibited emergent tactical teamwork. Cyberoos'2001 is the fourth "generation" designed in line with this framework.

Cyberoos'2001 agents incorporate a domain model into the architecture. The idea of having a "world model" directly represented in the architecture is intuitively very appealing. However, we believe that "world model" should grow incrementally instead of being inserted and glued to other elements. In other words, our preference is to observe and encapsulate instances of the emergent behaviour which potentially make use of the domain model. Most definitely, the RoboCup Simulation domain is rich enough to produce such examples — one considerable motivation among others is sensory stalls, when an agent does not receive a visual information during a simulation cycle. In particular, Cyberoos'2001 agents extrapolate their domain model each simulation cycle with the *domain_update* function, and revise it with the *domain_revise* function whenever new information becomes available (new visual inputs or team-mate communications). The partition between update and revision operators corresponds to the well-known distinction between belief update and belief revision [5].

Previous generations of Cyberoos did not use world models and inter-agent communication — they relied entirely on deep reactive behaviour and emergent tactics. For the first time in our four years long experiment, we can now directly observe and rigorously compare behaviours produced by agents using domain models and purely reactive agents. *This comparative analysis is the primary topic of our research.*

The described hierarchical framework has enabled a systematic incremental implementation and testing of software agents and their modules. In particular, the framework allowed us to correlate enhancements in agent architecture with tangible improvements in team performance. The Cyberoos'98 team took 3^{rd} place in the 1998 Pacific Rim RoboCup competition. Cyberoos'99 finished in the top half of the RoboCup-1999, while Cyberoos'2000 were 4^{th} in the Open European RoboCup-2000 ($\sim$ 16 teams) and shared 9^{th} place at the RoboCup-2000 ($\sim$ 40 teams). Cyberoos'2001 shared 9^{th} place at the RoboCup-2001 as well — this time among 44 teams. In addition, the DBP framework was used for development of an intelligent multi-agent system for *networked multimedia appliances.* The system is aimed at providing customisable user support in operating and controlling a flexible and changing network of multimedia appliances, while satisfying several (possibly conflicting) users' preferences [10].

It has been illustrated elsewhere [6,7,8,9] that an agent's complexity (be it *Tropistic* or *Hysteretic* or *Domain-Oriented* agent) is relative: for any elaborate agent type, it is possible to define more *concisely* another agent type with the same external behaviour. Therefore, an agent based on the DBP architecture will always have a choice as to which one of related hierarchical levels should assume control to better suit external environment. This selection process serves, in fact, as the "middle-out" layer between any two levels of the DBP agent architecture — replacing the need for a generic hub connecting "reactive behaviour" and "cognitive skills". Although, in general, it does take time to implement and experiment with the DBP-based agents, the resulting depth and flexibility in the agents behaviour appears to be worthwhile. The main lesson of the Cyberoos chronicle is, hence, the identification and study of the dialectic relation and necessary feedbacks between emergent behaviour and the agent architecture. More precisely, an emergence of essentially new behavioural patterns always indicates a need for new elements in the agent architecture.

Acknowledgements
The authors are grateful to Marc Butler and Ryszard Kowalczyk for their comments on various aspects of the RoboCup Simulation and multi-agent modelling, and for their contributions to previous generations of Cyberoos.

References

1. Michael S. Branicky. Behavioural Programming. In AAAI Tech. Report SS-99-05, the AAAI-99 Spring Symp. on Hybrid Systems and AI, 29–34, Stanford, 1999.
2. Rodney A. Brooks. Intelligence Without Reason. In Proceedings of the 12th Int'l Joint Conference on Artificial Intelligence, 569–595 Morgan Kaufmann, 1991.
3. Leslie P. Kaelbling and Stanley J. Rosenschein. Action and planning in embedded agents. In Maes, P. (ed) Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back, 35–48, 1990.
4. Hiroaki Kitano, Milind Tambe, Peter Stone, Manuela M. Veloso, Silvia Coradeschi, Eiichi Osawa, Hitoshi Matsubara, Itsuki Noda and Minoru Asada. The RoboCup Synthetic Agent Challenge. In Proceedings of the 15th International Joint Conference on Artificial Intelligence, 1997.
5. Pavlos Peppas, Abhaya Nayak, Maurice Pagnucco, Norman Foo, Rex Kwok and Mikhail Prokopenko. Revision vs. Update: Taking a Closer Look. In Proceedings of the 12th European Conference on Artificial Intelligence, 1996.
6. Mikhail Prokopenko, Ryszard Kowalczyk, Maria Lee and Wai-Yat Wong. Designing and Modelling Situated Agents Systematically: Cyberoos'98. In Proceedings of the PRICAI-98 RoboCup Workshop, 75–89, Singapore, 1998.
7. Mikhail Prokopenko. Situated Reasoning in Multi-Agent Systems. In AAAI Technichal Report SS-99-05, the AAAI-99 Spring Symposium on Hybrid Systems and AI, 158–163, Stanford, 1999.
8. Mikhail Prokopenko and Marc Butler. Tactical Reasoning in Synthetic Multi-Agent Systems: a Case Study. In Proceedings of the IJCAI-99 Workshop on Nonmonotonic Reasoning, Action and Change, 57–64, Stockholm, 1999.
9. Mikhail Prokopenko, Marc Butler and Thomas Howard, On Emergence of Scalable Tactical and Strategic Behaviour, In Proceedings of the RoboCup-2000 Workshop, Melbourne 2000.
10. `http://www.cmis.csiro.au/iit/Projects/RoboCup/rc_network.htm`, 2001.