Journal of Intelligent and Robotic Systems **27:** 263–273, 2000.
© 2000 *Kluwer Academic Publishers. Printed in the Netherlands.*

Managing Complexity in Large Learning Robotic Systems

KYNAN ENG, ALEC P. ROBERTSON and DEANE R. BLACKMAN
Department of Mechanical Engineering, Monash University, Wellington Road, Clayton VIC 3168, Australia; e-mail: deane@mec-blackman.eng.monash.edu.au

(Received: 30 December 1997; in final form: 26 May 1999)

Abstract. Autonomous learning systems of significant complexity often consist of several interacting modules or agents. These modules collaborate to produce a system which, when viewed as a whole, exhibit behaviour that can be interpreted in some way as learning. In designing these systems, the complexity of the interactions of large numbers of modules can become overwhelming, making debugging difficult and obscuring the workings of the system when viewed from an architectural level. A way of controlling system complexity called the Layered Learning System architecture (LLS) has been developed, which offers the advantages of incremental development and testing, easier debugging and progressive upgrading facilitation. A hexapod robot has been implemented using LLS principles, with the main learning task being that of learning to walk as fast as possible without falling over.

Key words: distributed control system, learning system architecture, layered systems, mobile robots, walking robots, robotic learning.

1. Introduction and Related Work

Much research has gone into the many different aspects of the cognitive design of autonomous systems. Many specific task-oriented learning problems have been defined and investigated in great detail. Each of these problems has had many algorithms and strategies developed to address the particular challenges of that problem. Very broadly speaking, the research in machine learning can be said to fall into three overlapping categories:

High level, for example: concept learning, language acquisition and goal-directed decision making. Knowledge at this level tends to be very abstract in nature and is often encoded into symbolic systems. Decision rule trees built up by trial and error are one simple example of this.

Medium level, for example: route planning, visual object classification and navigation. Problems in this class tend to be addressed by a moderate degree of abstraction of the environment, for example, a sonar-based navigation system may encode a flat terrain as tuples of landmarks and bounding polygons.

Low level, for example: intelligent controllers and any sort of system that deals with sensor data in a mostly unfiltered form. Brooks (1990) asserts that “physical grounding” is essential for any truly autonomous robotic system; ie. all system functionality must relate directly to sensor data. A softer form of this view is taken in the use of analogical data representations (Steels, 1990), where some degree of abstraction of the data is allowed as long as the internal storage “looks” like the raw sensor data in some way.

In developing practical fully autonomous systems, it is very difficult for any single classical algorithm or strategy to cover all required system functionality. Often, multiple strategies are used to achieve a desired overall behaviour. For example, the navigation system developed by Elfes (1995) uses a stochastic model with multi-objective optimisation methods to enable a robot to explore and map its environment using sonar. Here, two overlapping goals (exploration and mapping) are achieved using two approaches to environmental modelling (stochastic and multi-objective optimisation), which are carefully designed to work in an integrated system. Other researchers believe that task-fulfilling systems can evolve on their own through complex interactions of simple components. Vogt (1998) outlines a system of robotic designed to perceptually ground and distinguish objects in their environment, using feature generation and self-organisation. He also shows how robots may evolve a kind of simple language, using previous work on the origins of language done by Steels (e.g., Steels, 1997).

As learning systems increase in their range of potential functionality and behaviour, the task of integrating many different components working at different levels of abstraction becomes more difficult. Three broad theories on how to approach the integration task exist:

Non-Localised: Any particular aspect of system knowledge cannot be easily said to reside in any particular part of the cognitive structure. The subsumption architecture as used in the walking robot *Genghis* (Brooks, 1990) and neural networks are two well-known examples.

Localised: Each particular learning task of the system is contained in a separate functional block, with each block being mostly self-contained. All current conventional software engineering is done this way. Learning systems which work by task decomposition are also likely to be designed in this way.

Semi-Localised: Learning functionality is shared between two or more components of the system in such a way that removal of one impairs the learning function, but does not necessarily totally remove it. The human brain is an example that comes easily to mind.

This paper deals only with learning systems designed using the localist and semi-localist paradigms.

Many learning robots have an *ad hoc* architectural design, usually because a non-localist paradigm was followed or because the robot was sufficiently simple for architectural considerations to be ignored. More recently, however, it has been recognised that learning robots functioning at different levels of learning abstraction are usually too complex for *ad hoc* implementation. Borrowing the software engineering ideas of system design through functionality decomposition, researchers have been able to understand and handle increasingly complex systems. Arkin (1990) describes an autonomous robot architecture which decomposes system functionality into blocks, specifically designed for mobile robot applications. In a more architecturally systematic vein, Crowley (1996) describes a parallel hierarchical architecture for vehicle control that features a layered system of control. The motors and sensors are at the lowest level, with a vehicle control level and navigation level above that. At the top sits a supervisor and a human interface.

For future development of very complex learning robots that will appear in the future, methods will be required to facilitate the design and verification processes. One important step towards achieving this will be the definition of a general-purpose architecture for the development of such systems.

2. Layered Learning Architectures

2.1. INSPIRATIONS

Consider first the well-known OSI reference model for computer networking as shown in Figure 1 (Stallings, 1994).

Some interesting characteristics of this model are:

1. Only the physical layer has any access to the real data-transmission world.
2. The higher the level, the greater the degree of abstraction of concepts.
3. Communication is only allowed between adjacent layers; i.e., layer 5 cannot communicate directly with layer 2, for example.

One way of mapping these features on to aspects of a learning system architecture proceeds as follows:

1. Only the physical layer has any access to sensors and actuators.
2. Higher level learning functions occupy higher layers.
3. Complexity control is provided by forcing learning system components to interact only with adjacent layers.

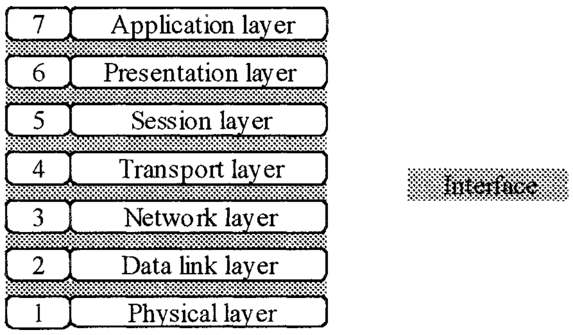

Figure 1. OSI networking reference model.

2.2. LAYERED LEARNING SYSTEM ARCHITECTURE DEFINITION

The LLS definition seeks to make the different levels of thinking and learning explicit, and in doing so to make the design of learning systems more straightforward. Using the ISO model shown in Section 2.1 as a basis, a model can be devised for a layered learning system architecture (see Figure 2):

The general characteristics of each of these layers can be summarised as follows:

Physical: The real world of sensors and actuators – motors, cameras, switches, etc.

Sensory: Filtering of low-level real-world data – converting signals into data structures.

Reflexive: Automatic reactions to input – responding to direct threats.

Action: Implementation and sequencing of simple operations.

Planning: Route setting, obstacle avoidance, etc.

Motivational: Controls the initiation and termination of planning functions.

Reasoning: Moderates the motivational behaviours of the system for long-term goal achievement.

Metaphysical: Control of reasoning: religion, imagination, fantasies.

Communication between layers can be via any mechanism, for example, via message queues, polled outputs, interrupts, etc. The circumstances of a particular system implementation may suggest one interface solution over another, or even several different interface types for different layers within the one system.

2.3. OTHER NATURAL AND ARTIFICIAL LEARNING SYSTEMS

It is interesting to rank some existing learning systems in terms of how they fit into the LLS framework described in the previous section (see Table I).

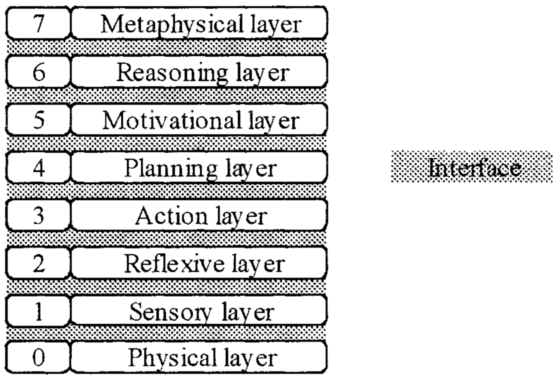

Figure 2. Layered Learning System (LLS).

Table I

System	**Level**
Bacteria	Physical-reflexive
Map computer	Planning *only*
Navigation robot	Physical-planning
Cat	Physical-motivational
Chimpanzee	Physical-reasoning
Human	Physical-metaphysical

There may be arguments over which levels each learning system fits into but that is unimportant for the purposes of this discussion. The main point to note is that the LLS appears to provide a way to categorise all known learning systems into a contiguous block of LLS levels, or "learning competence levels".

In most real robotic systems, the higher layers are implemented implicitly in the system design or are under human control. In most cases this is the most practical thing to do: the vast majority of robots are designed with a specific purpose in mind and it would be a waste of effort to bother trying to model the robot's inclination to carry out its designated task. Even more of a waste of time (at current technological levels) would be trying to model a robot's ruminations about philosophical questions. By placing higher cognitive levels under human control, either explicitly through a user interface or implicitly in the design of an autonomous robot, robots are kept in a subservient role by virtue of the lower cognitive level of the robot. An organic version of intellectual control can be seen in the widespread historical use of animals for labour. Equal intellects are also able to cooperate with, or subjugate, each other: humans have long known how to do this through the appropriate understanding and manipulation of cognitive functions.

2.4. LLS ADVANTAGES AND DISADVANTAGES

The main advantages of the LLS architecture come from an implementation perspective:

- High complexity is limited to the internal structure of each layer, making it more likely that humans will be able to understand the system architecture.
- The forced modularity imposed by the architecture allows for the possibility of incremental development. In other words, any component of the system can be implemented in a basic way at first and upgraded later.
- Incremental development shortens the time to a working prototype because all components can be implemented in a very basic fashion at first (perhaps, even without any learning components) to check that the interfaces work.

- The layered architecture limits the amount of rework required when a system component is changed since at most two inter-layer interfaces need to be dealt with.
- The graduated multi-filtering process for transferring data from low to high levels (and vice versa) makes it more likely that very low-level data will be able to be translated into high-level concepts, even if the means for doing so is not known at the beginning of system implementation.
- The LLS naturally lends itself to multiprocessing and distributed implementations, since each layer could feasibly be implemented on a separate processor.

There are also some potential disadvantages of the LLS:

- It is not known how much the layered architecture limits the potential for interesting and useful interactions between very high-level and very low-level modules. It is up to the intermediate modules to pass on information to higher/lower levels as they deem appropriate.
- Latency delays caused by data transfer across many layers may make system performance unacceptable for real-time applications. This can be avoided by careful system design – events requiring an immediate response should be handled by lower levels of the system. In any case, at least the lowest level must be able to run in real-time for any useful autonomous system.
- Poor design of any one layer could effectively block meaningful data transmission in the whole system.
- Care needs to be taken to avoid deadlock or starvation situations for individual layers. An intuitively obvious way to do this is to implement each layer so that it never blocks waiting for input, but continues processing with best approximations that are updated as new data becomes available.

3. Robbie – the Running Robot

3.1. ARCHITECTURAL ROBOT DESIGN

To test the theory of learning system implementation using LLS principles, a hexapod walking robot called Robbie was built. A basic schematic of the robot configuration is shown in Figure 3.

Robbie has six legs arranged symmetrically about a rectangular torso, each with two degrees of freedom. The six foot sensors indicate which feet are on the ground and the four belly sensors indicate when any corner of the torso of the robot is touching the ground.

For this study only the physical, sensory, reflexive and action layers of the LLS were implemented. The robot was designed with one *prime directive* and two *secondary directives*:

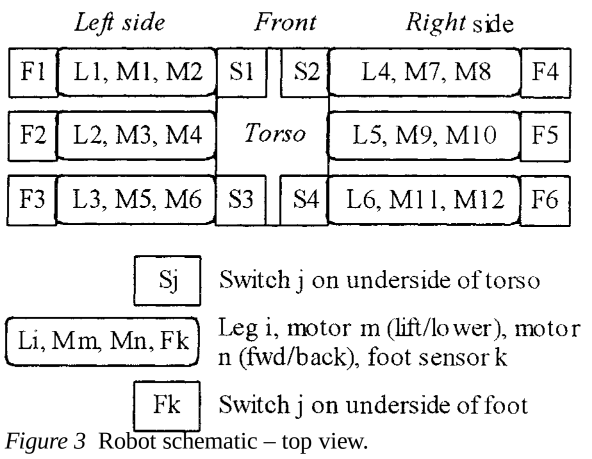

Figure 3 Robot schematic – top view.

Prime directive	• Walk: move forwards in a straight line
Secondary directives	• Avoid pain: stand up without falling over
	• Compete: walk as fast as possible in a straight line

These directives were each captured in a vertical integration of the layers in the LLS, with the prime directive forming the main trunk of the system. Some degree of horizontal integration between directives in different layers was also provided. Since the competing directives generated inconsistent data, *conflict resolution blocks* were included in the interfaces between different levels. An overview of the system architecture can be seen in Figure 3.

This design places all higher cognitive levels under implicit human control. In particular, the use of the term "directives" indicates the predefined nature of the robot's motivations. Real-time explicit human control is provided by a user interface, not shown in the architectural diagram. The interfaces between the modules are typed, but the system could have been designed with untyped interfaces as well.

3.2. LEARNING ALGORITHMS

To test the theory of the LLS allowing incremental development, all modules were initially implemented as simple I/O filters. At the same time, simple software was also implemented to show the state of each module and interface in real-time. This enabled all of the interfaces to be verified in a very short time – in other words, the robot did not do any learning, but the user was able to make it walk like a "zombie" by directly feeding data into the system. Over time, modules were replaced one by one with increasingly sophisticated learning algorithms. Testing and evaluation continued at each stage, and modules were written so that different learning algorithms could be called into use as desired.

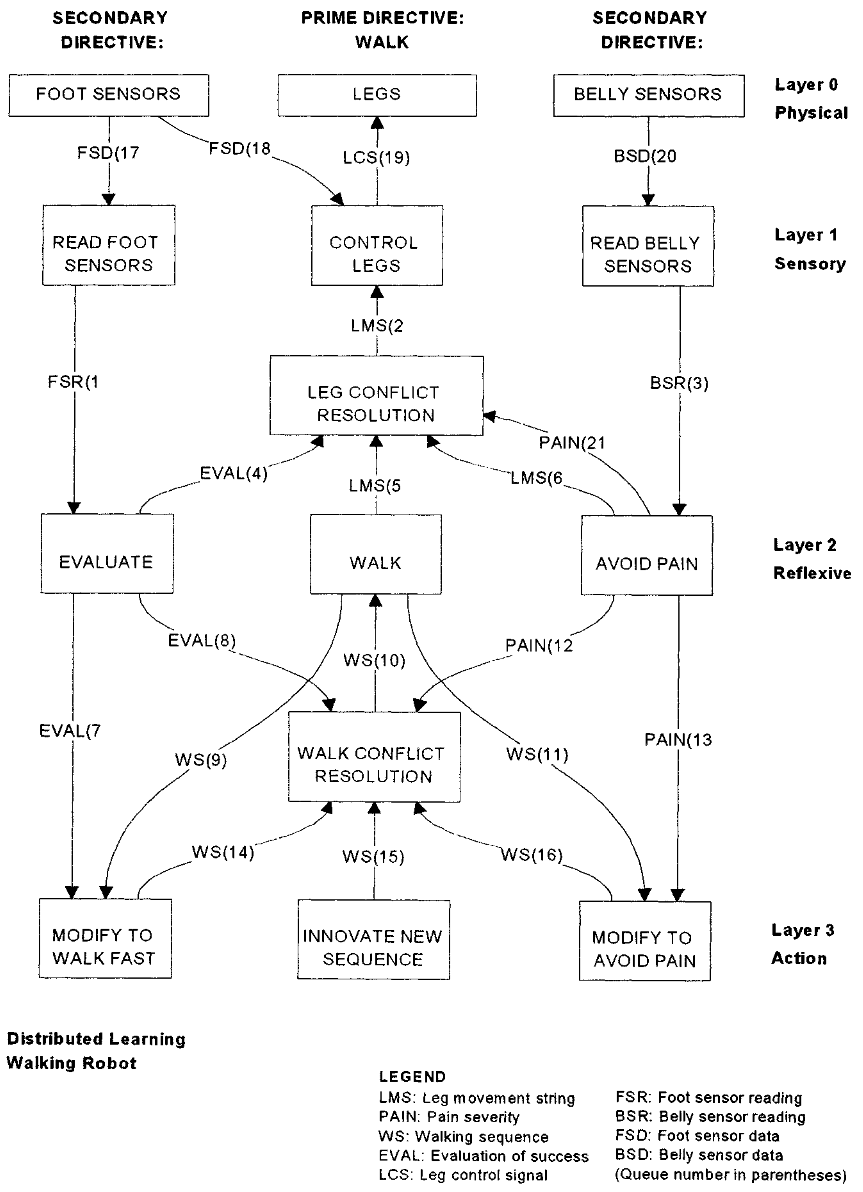

Figure 4. System architecture showing primary and secondary directives.

The development of the actual algorithms implemented can be summarised as follows:

Avoid Pain (Level 2): prevent the robot from falling over, or if it has fallen over then recover to a standing position.

1. Simple pass-through interface test.
2. List of leg positions which cause the robot to fall over.

3. Dynamically generated rule-tree containing information about leg configurations which cause the robot to fall over.
4. Augmented rule-tree incorporating the notion of the robot's own symmetry to speed up rule generation.

Modify to Avoid Pain (Level 3): change the robot's walking gait so that it does not fall over.
1. Simple pass-through interface test.
2. Dynamically generated and updated rule-tree.
3. Augmented rule-tree incorporating symmetry to speed up rule generation.

Evaluate (Level 2) and Modify to Walk Fast (Level 3): change the robot's walking gait to walk as fast as possible.
1. Simple pass.through interface test.
2. Rule-of-thumb algorithms aimed at minimising the amount of time without any leg movements. (Other algorithms were proposed but not implemented due to resource limitations.)

3.3. PHYSICAL CONSTRUCTION

The robot is about 300 mm long and has a total mass of 1 kg. It is constructed mainly of brightly coloured 3 mm translucent acrylic sheets. Normal radio-control servo motors are used to actuate each of the two degree-of-freedom legs. The six legs each have a microswitch mounted in the feet, and microswitches in each corner of the robot's torso tell it when it has fallen down. A photograph of the robot in action can be seen in Figure 5.

Control of the robot is effected by a PC running MS-DOS. This is connected to the robot via a long umbilical cord. All signals to the servos are supplied using two parallel ports in the PC. The learning software is run on a separate computer, connected by a RS-232 serial cable running at 9600 baud.

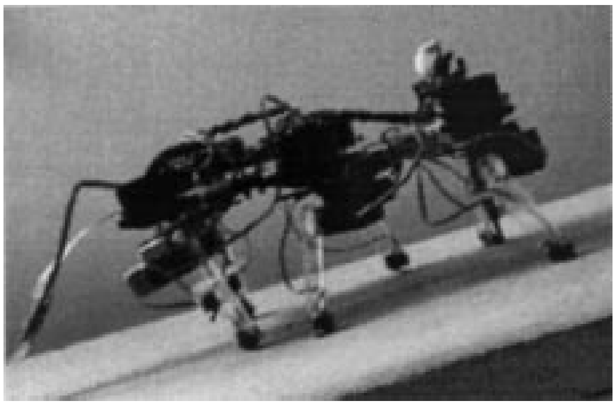
Figure 5. Robbie climbing an inclined plane.

Before the robot was built, a simulation was developed to check that its overall learning architecture was likely to work. This was a simply done by plugging in a "world" simulation module at level 0, and removing it when the robot was ready. The simulated world was based in part on work by Bajd et al. (1994) on unstable states in legged locomotion.

3.4. ROBOT PERFORMANCE

The actual performance of the learning algorithms will not be discussed in depth here. It is sufficient to note that the more sophisticated learning algorithms gave better performance, both in the speed of learning (time to successful walking without falling over) and in the maximum walking speed attained. In particular, it was found that the algorithms which included the concept of the robot's own symmetry performed significantly better than those which did not. This was particularly true at the start of the learning process, when the robot needed as much information as possible to learn quickly – even if that information was not reliable.

Over time, the performance of the robot during development progressed as follows:

1. No learning; "zombie" walking only.
2. Simple learning modules installed: poor performance.
3. More sophisticated learning modules installed: better performance as each improved module was added.

Because the robot was verified as working even before the first learning algorithms were devised, a degree of concurrent engineering was made possible. Implementation of the physical robot began before the simulation was completed, and the learning algorithms were developed incrementally alongside the physical robot. The critical element for providing the confidence to proceed with parallel development was the early definition and verification of the layered learning architecture, even without any idea of how each individual module was to be implemented.

4. Conclusion

A generalised way of designing large learning systems using layers has been developed which provides a framework for controlling the complexity of the system. A robot developed according to the principles of decomposition into layers was found to have several advantages in development, including incremental upgradability, easy design verification, rapid development and concurrent development. Further work should include the building of a system incorporating all of the cognitive levels proposed, at least in a very crude fashion, to further test the viability of the architecture.

Acknowledgements

Thanks go to Dr. Bijan Shirinzadeh of Monash University, for his valuable guidance in the course of the design and construction of Robbie the Running Robot. Other people who have provided assistance include Keith Hoy, Peter Koch and André Spierings.

References

1. Arkin, R. C.: Integrating behavioural, perceptual, and world knowledge in reactive navigation, *Robotics Autonom. Systems* 6, North-Holland, Amsterdam, 1990.
2. Bajd, T., Kralj, A., and Karcnik, T.: Unstable states in four-legged locomotion, in: *Internat. Conf. on Intelligent Robotics and Systems*, IEEE Conf. Proc., 1994, pp. 1019–1025.
3. Brooks, R. A.: *A Robust Layered Control System for a Mobile Robot*, AI Memo 864, MIT Press, Boston, 1985.
4. Brooks, R. A.: Elephants don't play chess, *Robotics Autonom. Systems* 6, North-Holland, Amsterdam, 1990.
5. Crowley, J. L.: Mathematical foundations of navigation and perception for an autonomous mobile robot, in: *Reasoning with Uncertainty in Robotics* (*Conf. Proceedings*), Springer, Berlin, 1995.
6. Elfes, A.: Robot navigation: Integrating preception, environmental constraints and task execution within a probabilistic framework, in: *Reasoning with Uncertainty in Robotics* (*Conf. Proceedings*), Springer, Berlin, 1995.
7. Eng, K., Robertson, A. P., and Blackman, D. R.: Robbie the running robot: A distributed learning system, mechatronics and machine vision in practice, in: *IEEE Conf. Proc.*, 1997.
8. Stallings, W.: *Data and Computer Communications*, Macmillan, New York, 1994.
9. Steels, L.: Exploiting analogical representations, *Robotics Autonom. Systems* 6, North-Holland, Amsterdam, 1990.
10. Steels, L.: Synthesising the origins of language and meaning using co-evolution, self-organisation and level formation, in: J. Hurford et al. (eds), *Evolution of Human Language*, Edinburgh University Press, Edinburgh, 1997.
11. Vogt, P.: Perceptual grounding in robots, in: *6th European Workshop on Learning Robots*, Springer, Berlin, 1998.