Knowledge-Based Systems 22 (2009) 509–515

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Exploring design space for an integrated intelligent system

Nick Hawes *, Jeremy Wyatt, Aaron Sloman

Intelligent Robotics Lab, School of Computer Science, University of Birmingham, Birmingham B15 2TT, UK

ARTICLE INFO

Article history:
Available online 9 January 2009

Keywords:
Information-processing architectures
Intelligent systems
Robotics

ABSTRACT

Understanding the trade-offs available in the design space of intelligent systems is a major unaddressed element in the study of Artificial Intelligence. In this paper, we approach this problem in two ways. First, we discuss the development of our integrated robotic system in terms of its trajectory through design space. Second, we demonstrate the practical implications of architectural design decisions by using this system as an experimental platform for comparing behaviourally similar yet architecturally different systems. The results of this show that our system occupies a "sweet spot" in design space in terms of the cost of moving information between processing components.

1. Introduction

Intelligent systems (e.g. intelligent service robots) are a product of the many design decisions taken to ensure that the final system meets the requirements necessary to fit in its particular niche [19]. In nature, evolution creates behaviours and bodies that suit an animal's ecological niche. In the field of intelligent artifacts, choices about the design and implementation of hardware and software may be taken by a designer, or enforced by project or resource constraints. Few, if any, of these choices are truly independent; using a particular solution for one part of the system will constrain the space of solutions available for other parts of the system. For example, the number of degrees of freedom of an effector will restrict the design of the control software and behaviours required to use the effector, and the choice of middleware for software components will restrict the communication patterns that components can use. Understanding the *trade-offs available in the design space of intelligent artifacts* is a major open issue in the understanding of integrated intelligent systems, and thus AI.

In this paper, we focus on the design space of architectures for intelligent robots. We discuss the design of, and the trade-offs created by, an *architecture schema* for intelligent agents based on a model of shared working memories. Following this we present a novel exploration of the design space of information sharing models for architectures for integrated intelligent systems based on this schema. This exploration uses an intelligent robot as an experimental platform. The robot's architecture is varied in principled ways to generate quantitative information demonstrating the costs and benefits of the different designs.

* Corresponding author.
E-mail addresses: n.a.hawes@cs.bham.ac.uk (N. Hawes), j.l.wyatt@cs.bham.ac.uk (J. Wyatt), a.sloman@cs.bham.ac.uk (A. Sloman).
URLs: http://www.cs.bham.ac.uk/~nah (N. Hawes), http://www.cs.bham.ac.uk/~jlw (J. Wyatt), http://www.cs.bham.ac.uk/~axs (A. Sloman).

0950-7051/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.knosys.2008.10.010

2. Background

In the field of intelligent systems, the term "architecture" is still used to refer to many different, yet closely related, aspects of a system's design and implementation. Underlying all of these notions is the idea of a collection of units of functionality, information (whether implicitly or explicitly represented) and methods for bringing these together. At this level of description there is no real difference between the study of architectures in AI and software architectures in other branches of computer science. However, differences appear as we specialise this description to produce architectures that integrate various types of functionality to produce intelligent systems. Architectures for intelligent systems typically include elements such as fixed representations, reasoning mechanisms, and functional or behavioural component groupings. Once such elements are introduced, the trade-offs between different designs become important. Such trade-offs include the costs of dividing a system up to fit into a particular architecture design, and the costs of using a particular representation. Such trade-offs have been ignored by previous work on integrated systems, yet these factors are directly related to the efficacy of applying an architecture design to a particular problem. Our research studies architectures for integrated, intelligent systems in order to inform the designers of these systems of the trade-offs available to them.

An important distinction to make when studying the information-processing architectures used in intelligent systems is the distinction between architectures that are entirely-specified in advance (e.g. those used in [16,17]), and architectures that are partially specified. This latter type can then be specialised to produce different *instantiations* of the architecture. We will refer to such partially specificed architectures as architecture *schemas* as they provide outlines from which many different concrete instantiations can be designed. Examples of such schemas include cognitive modelling architectures such as ICARUS [15], ACT-R [1], and Soar

[14]; more general frameworks such as CogAff [19] and APOC [2]; and robotic architectures such as 3T [6]. It is worth noting that the freedom in specialisation available to the designer varies greatly across these schemas.

As architecture schemas can be instantiated in various ways, each one provides a framework for exploring a limited region of design space for possible architecture instantiations: the use of a particular schema restricts the available design space to be those designs that can be created within the schema. Although instantiations produced from a schema may vary considerably, they will all share certain characteristics as a consequence of occupying similar regions of design space. It is difficult to study these characteristics directly, particularly in implemented systems, because it is difficult to separate the effects of the schema (i.e. the aspects of the architecture that will exist in all instantiations of the schema) from the effects of the specialisation (i.e. any additional components and implementation work) in the finished system.

As we wish to study the effects of variations in architecture schema in implemented systems we need an experimental approach that overcomes the problem of separating schema effects from specialisation effects. Our approach involves taking a single task (i.e. a problem requiring a fixed set of known behaviours) and creating instantiations of a number of different architecture schemas to solve it. In this way the task-specific elements of the instantiations are invariant (e.g. the algorithms used to process input and generate output), whilst the schema-level elements change between instantiations (e.g. the nature of the connections between input and output modules). Assuming task-specific invariance exists, comparing instantiations of different schemas on a single task will then provide information about the trade-offs between the different design options offered by the schemas.

Such single-task comparisons could be performed using existing systems. For example, driving tasks have been tackled in ACT-R [18] and ICARUS [5], spatial reasoning tasks by SOAR [20] and ACT-R [13], and natural language understanding has been tackled with almost every architecture schema, e.g. in APOC [3]. The drawback of this approach is that the different instantiations performed by different researchers using different technology will almost certainly introduce variations in behaviour that may mask the underlying effects of the various architectures, making comparisons worthless. To make comparisons between implemented systems informative, the variation in instantiations must be controlled. This is an approach we explore in Section 4. An alternative approach is to perform these comparisons theoretically (cf. [12]), although this risks overlooking the critical aspects that only become apparent when building integrated systems.

3. From requirements to robots

To further explore the idea of design decisions constraining the design space available for a particular intelligent system, it is worth considering an example. Our current research project is studying the problem of building intelligent systems. We are approaching the problem from various perspectives including hardware control, subsystem design (including vision, planning etc.) and architectures. As the project has progressed we have made strong commitments to particular designs for elements of the overall system. These design commitments have constrained the space of solutions available for subsequent developments. Although the following description is anecdotal, it demonstrates one possible type of development trajectory[1] for an integrated intelligent system.

[1] It is arguable that this development trajectory has much in common with a large number of intelligent and integrated system projects.

Prior to any design or development we analysed our target scenarios. From these scenarios we extracted a number of requirements for our integrated systems to satisfy, providing some initial constraints on design space. These requirements are too numerous to explore fully here, but the following proved important for subsequent developments:

- The system must feature concurrently active elements in order to respond to a dynamic world whilst processing.
- The system must represent and reason about hypothetical future states, thus requiring explicit representations.
- The system must support specialised reasoning in its subsystems, requiring support for multiple representations.

Although these requirements do not appear too restrictive, they rule out design approaches that require a single unified representation and that do not support concurrency. This prevents the use of many architectures for modelling human-level intelligence, and logic-based robotics approaches.

Our first design for a system to satisfy the scenario's requirements was constructed using the Open Agent Architecture (OAA) [4]. It featured concurrent components for language interpretation and generation, object category recognition using a modified variant of SIFT, generation of multiple forms of spatial representations, and cross-modal information fusion. The use of OAA constrained us to design the system as a network of exhaustively pair-wise connected components that exchanged information directly. Although the resulting system satisfied our requirements and demonstrated the desired behaviour, the architecture structure had a number of drawbacks. The main drawback was that the direct exchange of information made it difficult to share the same information between more than two components in a system, making it difficult to explore the consequences of component collaboration (e.g. using scene information to incrementally reduce the hypothesis space during parsing [3]). This is a clear of example of a design choice (the connection model enforced by the architecture) limiting the subsequently available design space (the design space of information sharing models). It is worth noting that in theory we could have implemented a different information sharing model on top of OAA, but this would not have been a natural fit with the underlying architecture. These kinds of specialisation costs (i.e. the cost of implementing one system given the constraints of another) are hard to measure, but typically very important in the design and implementation of intelligent systems.

Because of the drawbacks of our OAA-based system, we decided to explore designs for architectures based on commonly accessible information stores. This led to the development of the CoSy Architecture Schema (CAS) [9,7,10], a schema that groups concurrently active processing components into separate *subarchitectures* via limited connections to *shared working memories*. In CAS all communication between components occurs via these working memories (rather than directly between components), enforcing the requirement of data sharing (see Fig. 1a). However, the separation of components into groups around working memories brings its own set of design constraints, and these have become apparent in the systems we have built on top of CAS.

The systems we have built with CAS feature subarchitectures designed around particular representations, e.g. a visual subarchitecture containing 3D representations of objects, and a communication subarchitecture containing logical interpretations of utterances. Although these representations are shared, they are only intelligible by the components in the same subarchitecture. To obtain a complete view of all the information in the system, our instantiations have had to include a *binding* subarchitecture, which abstracts from other subarchitectures to provide a single amodal representation of the current state [11]. It is arguable that