

Towards Java-based Intelligent Control Architecture

Konrad Kułakowski*

Institute of Automatics, AGH University of Science and Technology
Al. Mickiewicza 30, 30-059 Cracow, Poland

Abstract

The concept of intelligent control system architecture is not a new idea. The reference architectures of such systems were proposed and broadly discussed in the late 80s and 90s of the twentieth century. Unfortunately, many authors either focus on the general scheme of this class of systems without a clear implementation perspective, or implement some solutions without precisely defined designs.

The aim of this article is to propose a design for intelligent control system. Theoretical discussion is supplemented by implementation remarks. The current state of our prototype implementation of intelligent control system is also presented.

In order to propose some design principles of such systems, existing models and approaches are briefly reviewed and discussed.

1 Introduction

In the last decade, we have observed the increasing popularity and accessibility of autonomous control systems. Some of them are almost as easy to build and control as playing with a child's building blocks. An example of such out-of-the-box systems is Lego Mindstorms [22]. Additionally, a programming language for writing a robot's control procedures evolved from the error-prone and hard-to-understand assembler or C to more advanced languages as Java, Prolog, or visual languages such as LabView Toolkit for Lego Mindstorms NXT.

Along with the growing availability of software toolkits facilitating the creation of a mobile robots' control software, the whole software development process has been shortened. Development acceleration, however good from a marketing perspective, usually negatively impacts on the design phase and, as a result, the number of under-designed or chaotic-design projects is increasing. This fact reminds everybody that good programming tools have to be accompanied by well-designed software.

The primary goal of this article is to propose a general model of intelligent control architecture. The presented general model aims to create grounds for developing Java-based intelligent control systems, which would be suitable for different hardware platforms. The prototype implementation presented here has the status of *work-in-progress*, so it cannot be ultimately evaluated at the moment.

* Author was partially supported by MNiSW under grants N516 228735 and N516 024 32/2878

2 Intelligent Control System Architectures

As pioneer works in the field of intelligent control systems we might consider HILARE - a mobile robot constructed in LAAS (*Laboratoire d'Architecture et d'Analyse des Systèmes*) [16] or SHAKEY the robot constructed by Nilsson [27] at *Stanford Research Institute*. Later on they evolved into three major approaches to intelligent control architectures [4]:

- Hierarchical Planning and Control Architecture
- Reactive/Behaviour Based Control Architecture
- Hybrid Architecture

One of the most well-known representatives of the first approach is "A Reference Model Architecture of Intelligent Control" proposed by J. S. Albus and A.M. Meystel [3]. It is based on early work [8] describing *RCS (Real-time Intelligent Control System)*. *RCS* partitions the control problem into four components: behaviour generation (in some systems it is just called a task decomposition), world modelling, sensory processing and value judgment. Each component has its own computational nodes, which are organized into hierarchical layers. Every layer has its own, well defined area of responsibility and specific timing. In this approach the world model plays an important role. All the decisions taken by the system are verified against it and all the incoming events may change the systems knowledge about the world.

On the opposite side, there is another approach. According to the principle "The world is its own best model" [10] reactive/behaviour based control systems focus on immediate data sensing and behaviour generation rather than spending time on possibly time-consuming world model exploration. They are particularly useful for quickly changing and unstructured environments, where the precise world model might be hard to construct [4]. A good representative of this approach might be *Subsumption Architecture* proposed by Brookes [9].

A solution that tries to take benefits from both hierarchical planning and reactive control techniques is a hybrid approach. It is based on the observation that in the actual environment the autonomous intelligent system sometimes has to behave in a reactive manner, and at other times to perform careful knowledge-based hierarchical planning. In fact both, reactive control and hierarchical planning frequently address different parts of the same problem. Finding the optimal path is usually associated with hierarchical, deliberative processing, whilst reaction to an obstacle which suddenly appears on the path is better handled by reactive control procedures. *AuRA* [5] or *PRS* [15] may serve as examples of this class of architecture.

3 Related works

A proposition of the intelligent control architecture is presented by Zeigler and Chi in their work [39]. Besides the outline of an autonomous system's control architecture, they propose a systematic approach to design, generation and diagnosis of that kind of systems. A more formal approach for 'embedding intelligence

in control' has been proposed by Acar and Üzgüner [29]. In their work the authors propose a formally defined structure-based hierarchy of computational nodes, which are mapped via appropriate structural coordination to a piece of hardware. Every node has its own set of accomplishable tasks and methods, or procedures to accomplish them. The top node in the hierarchy represents the whole of the system. The subsequent levels in the hierarchy gather nodes responsible for more and more detailed parts of the system.

Intelligent control is also implemented as component-oriented and object-oriented architectures. An example of the component-based approach is *OSCAR* [1]. In this system all the communication is handled by the *CORBA (Common Object Request Broker Architecture)* infrastructure [28] specified by *OMG (Object Management Group)*. From the architectonic point of view it is an interesting idea to introduce several layers of signal processing (e.g. *Integration Layer*) upon the sensor layers. Another object-oriented architecture is *BERRA* [21]. The authors define three conceptual layers of a system: *Deliberate Layer*, *Reactive Layer* and *Task Execution Layer*. All the layers consist of object-oriented components, covering certain areas of responsibilities. Examples of the object oriented approach are also mentioned in [35] and [25]. The second work focus on modeling control architecture entirely in *UML*.

The choice of implementation platform strongly depends on designed usage and available hardware components. Very often it is a *C++* based component application running under the control of a unix-like operating system [21, 20].

The Java platform along with its growing popularity has been proposed as an environment for robotics control systems [24, 32]. Additionally, a *Real Time Java* [12] has been used as a software platform for robot control [33].

Lego Mindstorms [22] is a very interesting platform combining software and hardware that might be used for building intelligent control systems. Besides the original firmware provided by *Lego* there are several others software platforms that might be uploaded into the robot. These are: *LeJOS* [7] – *Java Virtual Machine* running on *RCX (Programmable Lego Brick)*, *BrickOS* [37] – an alternative *RCX C/C++* based real time operating system and the *Programmable Brick Forth* [17] – runtime environment supporting language *Forth*. There are many examples of using *Lego Mindstorms* for building and programming robots. Many of these works are not autonomous control systems (since they are controlled remotely by human), or it might be classified as a simple reactive/behaviour based control architecture [4].

4 Architecture of the Intelligent Control System

4.1 Single Robot Perspective

A great number of system architectures for intelligent systems have been conceived and implemented. The architecture for intelligent systems that will be proposed as a starting point is *the architecture for intelligent systems* proposed by James Albus [2]. However, in contrast to the Albus model we assume that the system sometimes has to perform actions in a responsive manner i.e. without prior deliberation based

on specific knowledge. This assumption, according to the taxonomy proposed by Arkin [4], locates our architecture in the class of hybrid systems.

The proposed system architecture for a single robot has a hierarchical nature. It is composed of two main layers (figure 1):

- automatic control layer,
- deliberative control layer

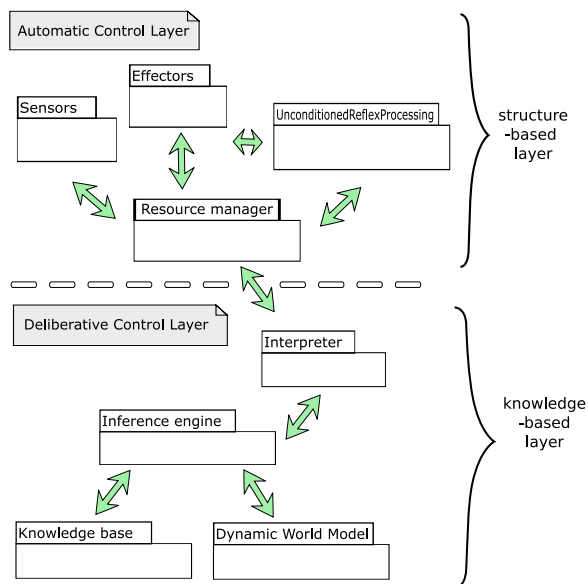


FIGURE 1: General system architecture, modules and inter-modules communication

The *automatic control layer* contains a *sensor module* (SM) including all the objects and procedures responsible for gathering and preprocessing events coming from the environment, and an *effectors module* (EM) including all the objects and control routines managing the system's effectors on the level of groups of servo mechanisms. Thus, on this level of hierarchy some actual stimuli coming from outside the system are fetched by hardware sensors, next they are grouped and preprocessed. Then they are re-raised in the form of software events to the rest of the system. Similarly this level encapsulates single hardware actuators and makes them available to the rest of the system in the form of more abstract structures like robot legs, arms etc. An *unconditioned reflex processing module* (URPM) is a set of independent procedures that are responsible for quick but very simple system reactions. This module is responsible for short feedback between sensors and effectors. There is no deliberative, knowledge-based processing, thus it does not need access to the knowledge base and world model. Operations performed by all the automatic control layer modules are coordinated by the *resource manager module* (RMM). This module is responsible for resolving software/hardware conflicts between control requests coming from the *interpreter module* (IM) and

URPM. Because all the modules in this layer depend directly or indirectly on physical organisation of the system, this layer is also called *structure-driven*.

The *deliberative control layer* is responsible for proper situation judgement and behaviour generation. It consist of four modules: *knowledge base module* (KBM), *dynamic world model module* (DWMM), *inference engine* (IEM) and *interpreter module* (IM). Almost all the modules of this layer store some forms of knowledge. The KBM module stores declarative knowledge. DWMM stores descriptive knowledge of the world and declarative knowledge about its behaviour. Also IM stores some descriptive and procedural knowledge. It is indispensable to properly translate the conclusions formulated by IEM to action requests understood by the resource manager. IE is responsible for processing the knowledge coming from KBM and DWMM.

Inter-modules communication is shown on figure 1 in the form of bidirectional arrows.

4.2 Multi-robot Perspective

The field of distributed robotics is rapidly growing, and a vast amount of research deals with multi-robot systems [36, 31, 11]. There are several reasons why such systems might be interesting [11, 36]:

- some problems might be solved more effectively using many robots working in parallel,
- systems based on many robots are more robust and fault tolerant,
- building several simple, universal robots might be cheaper than constructing a single, powerful robot solving one specific problem,
- the theory of multi-robot systems may benefit from other scientific disciplines such as multi-agent systems [26], social sciences and live sciences, and therefore it may contribute to the development of many theories going beyond the generally understood idea of computer science.

Because of all these reasons we also conduct research into distributed robotics. Verret proposed two main characteristics of multi-robot system architectures. The first one, depending on the coordination scheme, divides systems into centralized and decentralized, whilst the second one distinguishes between homogeneous and heterogeneous architectures. Centralized systems have one machine, agent or process controlling all of the agents (robots) in the system. Decentralized systems [5, 30, 6, 14] do not have a central agent that monitors all the agents. There are also hybrid architectures in which there is a common planning super-agent that manages the decentralized teams of agents. Although centralized architectures seem to be easier to implement than decentralized ones, in many practical applications they are not so efficient and robust as decentralized architectures. For instance, this kind of problem might be observed in [18]. Thus, in our research we focus on distributed and hybrid solutions. Our aim is to propose distributed deliberative robot team architecture equipped with a well-defined motivation mechanism [34].

5 Implementation in Java

5.1 Basis of Prototype Implementation

The platform of choice for the proposed architecture is Java Virtual Machine. As hardware for the prototype implementation we choose NXT Mindstorm Intelligent Brick and a PC class computer. Almost all components of the automatic control layer (except the *resource manager*) are designed to be implemented directly on the hardware i.e. NXT Brick (see figure 2). Besides the components SM, EM and URPM, there are also additional system procedures responsible mainly for system communication. All the code running on the robot's hardware has to be written carefully because of the platform limitations: limited ability to track errors, small amount of operation memory, limited computational power of the ARM based processor. Almost all the parts of the deliberative control layer are designed to be deployed on a PC computer, where the platform limitations are not so strong. In particular, modules connected with knowledge processing e.g.: KBM, DWMM IEM and IM, require a significant amount of RAM and substantial computational power.

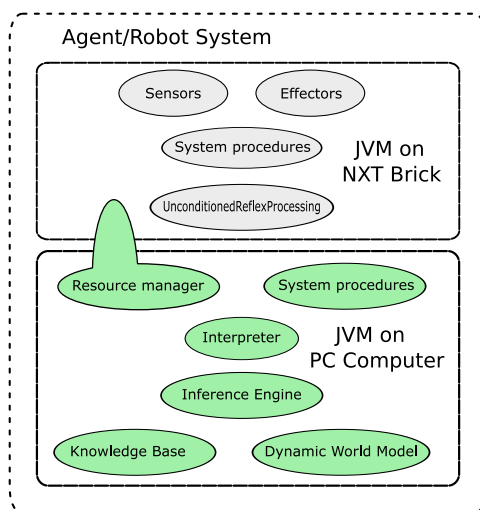


FIGURE 2: Hardware deployment of ICS components

There is also one module that crosses the hardware NXT/PC boundary. That is the *resource manager module*. Because some parts of this module have to intensively communicate with SM and EM, whilst the other parts have to intensively interact with the *interpreter module*, RMM has been split into two complementary parts running on NXT and PC.

Taking into account the fact that the NXT brick provides some native remote control interface, it may be asked why the *sensor module*, *effector module* and

unconditioned reflex processing module are not deployed on a PC computer. The tests carried out showed that placing SM, EM and URPM directly on a PC significantly increases communication traffic between NXT and the PC. As a result, a degradation of the reactivity and lengthening of the response time of the system has been observed.

Communication between the NXT Brick and PC computer is realized via a bluetooth link. A pair: JVM on NXT Brick and JVM on PC Computer make up the intelligent control system of a single robot. Different robots (agents) may communicate among themselves via the Internet (figure 3).

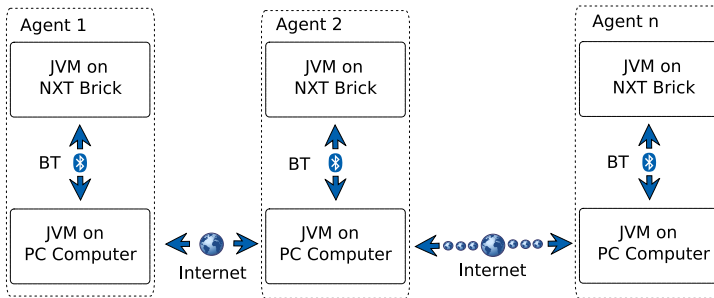


FIGURE 3: System communication

5.2 Implementation status

The presented intelligent architecture has been partially implemented. The preliminary NXT implementation of the *automatic control layer* is ready including handling of some 3rd party sensors such as Vision Subsystem v2 for NXT (NXTCam) or Magnetic compass for NXT (CMPS-Nx). Implementation of the *deliberative control layer* has started, and initial versions of some module interfaces (KBM, IEM, IM, DWMM) have been written. As an implementation of KBM different Java based rule systems are tested. A cellular automaton of the ECAL class [13] as a dynamic world model will be used.

Currently, the system is able to perform, simple tasks like detect and react to colour objects appearing in the view of the NXTCam, move smoothly along a given sequence of primitive moves and react to a simple stimulus coming from light, touch and sonar sensors (figure 4).

6 Conclusion and Further Works

The prospect of intelligent robots helping people to repair cars or cut hedges is very attractive and promising. Thus, despite its long and rich history, the building of intelligent robotics control systems is still popular and worth further effort. An

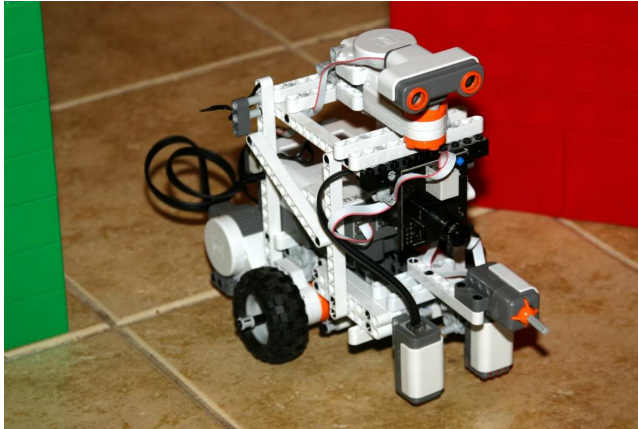


FIGURE 4: NXT based robot

important element of a successful intelligent control system is properly designed architecture. It determines many important factors of the whole construction, such as efficiency, stability, reliability and robustness. For this reason we believe that further research into intelligent control architectures is important and necessary.

This paper includes a brief review of different approaches to intelligent control architectures. Next, the proposition of a new ICS architecture, together with prototype implementation of such architectures are discussed. Both architecture and implementation need further effort and research. The most urgent matter is to complete the implementation of the deliberative control layer, and perform more complex experiments providing further data, which help to improve the system. After preliminary refinement, the system is planned to move to other hardware platforms, such as Hexor [38], Talrik IV [23] or Pioneer 3-DX [19].

References

- [1] Blum S. A. *From a CORBA-Based Software Framework to a Component-Based System Architecture for Controlling a Mobile Robot*, pages 333–344. Springer Berlin, Heidelberg, 2003.
- [2] J. S. Albus. Outline for a theory of intelligence. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(3):473–509, May–June 1991.
- [3] J. S. Albus. RCS: A Reference Model Architecture for Intelligent Control. *IEEE Computer*, 25(5):56–59, 1992.
- [4] R. C. Arkin. *Intelligent Control of Robot Mobility*, chapter 16. Wiley, 2007.
- [5] R. C. Arkin and T. Balch. Cooperative multiagent robotic systems, September 1998.
- [6] H. Asama. Operation of cooperative multiple robots using communication in a decentralized robotic system. In *Proceedings of the Conference From Perception to Action*, 1994.
- [7] B. Bagnall. *Maximum LEGO NXT: Building Robots with Java Brains*. Variant Press, 2009.

- [8] A. J. Barbera, M. L. Fitzgerald, J. S. Albus, and L. S. Haynes. RCS: The NBS real-time control system, 1984.
- [9] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE J. Robot. and Auto.*, 2(3):14–23, 1986.
- [10] R. A. Brooks. Intelligence without Representation. *Artificial Intelligence*, 47:139–159, 1991.
- [11] Y. U Cao, A. S. Fukunaga, and A. B. Kahng. Cooperative Mobile Robotics: Antecedents and Directions. *Auton. Robots*, 4(1):7–27, 1997.
- [12] P.C. Dibble. *Real-Time Java Platform Programming*. Prentice Hall PTR, 2002.
- [13] E. Dudek-Dyduch and J. Waś. Knowledge representation of Pedestrian Dynamics in Crowd. Formalism of Cellular Automata. In *Lecture Notes in Artificial Intelligence*, volume 4029, 2006.
- [14] C. Schoenwald D.A. Feddema, J.T. Lewis. Decentralized control of cooperative robotic vehicles: theory and application. *IEEE Transactions on Robotics and Automation*, 2002.
- [15] M. P. Georgeff and A. L. Lansky. Reactive Reasoning and Planning. In *Proceedings of the 6th National Conference on Artificial Intelligence*, 1987.
- [16] G. Giralt, R. Sobek, and R. Chatila. A multi-level planning and navigation system for a mobile robot; A first approach to HILARE. In *International Joint Conference on Artificial Intelligence*, pages 335–337, 1979.
- [17] Hempel Design Group. Programmable Brick Forth.
- [18] A. Howard, G. S. Sukhatme, and M. J. Matarić. Multi-Robot Mapping using Manifold Representations. *Proceedings of the IEEE - Special Issue on Multi-robot Systems*, 94(9):1360 – 1369, Jul 2006.
- [19] Mobile Robots Inc. Pioneer 3-DX - technical specification. www.activrobots.com, 2009.
- [20] K. Konolige, K. Myers, E. Ruspini, and A. Saffiotti. The saphira architecture: A design for autonomy, July 14 1997.
- [21] Matthias Lindström, Anders Orebäck, and Henrik I. Christensen. BERRA: A research architecture for service robots. In *ICRA*, pages 3278–3283. IEEE, 2000.
- [22] Giulio Ferrari Mario Ferrari and David Astolfo. *Building Robots with LEGO Mindstorms NXT*. Syngress Media, 2007.
- [23] Mekatronix. Talrik IV Manual. www.mekatronix.com, 2009.
- [24] S. Meloan. Futurama: Using Java Technology to Build Robots That Can See, Hear Speak, and Move. Technical report, Sun Microsystems, 2003.
- [25] E. Mumolo, S. Moratto, M. Nolich, and G. Vercelli. Object Oriented Design of a Mbile Robot using UML. In *Proceedings of the 23rd International Conference on Information Technology Interfaces*, 2001.
- [26] R. R. Murphy. *Introduction to AI Robotics*. MIT Press, 2000.
- [27] N. Nilsson. Shakey the Robot. Tech Note 323, AI Center, SRI International, 1984.
- [28] OMG. The Comon Object Request Broker: Architecture and Specification. Technical Report PTC/96-03-04, Object Management Group, 1996. Version 2.0.
- [29] Ü. Özgüner and L. Acar. *Design of Structure-Based Hierarchies for Distributed Intelligent Control*, chapter 4. Kluwer Academic Publishers, 1993.

- [30] L. E. Parker. ALLIANCE: An architecture for fault tolerant cooperative control of heterogeneous mobile robots. In *In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1994.
- [31] L. E. Parker. Current State of the Art in Distributed Autonomous Mobile Robotics. Technical report, Oak Ridge National Laboratory, 2000.
- [32] S. Presston. *The Definitive Guide to Building Java Robots*. APress, 2005.
- [33] S. G. Robertz, R. Henriksson, K. Nilsson, A. Blomdell, and I. Tarasov. Using Real-Time Java for industrial robot control. In *JTRES '07: Proceedings of the 5th international workshop on Java technologies for real-time and embedded systems*, pages 104–110, New York, NY, USA, 2007. ACM Press.
- [34] J. Starzyk. *Motivation in Embodied Intelligence*. I-Tech Education and Publishing, 2008.
- [35] F. Vazquez, R. Marn, J. L. Trillo, and J. Garrido. Object oriented modeling, design simulation of industrial autonomous mobile robots, February 03 1998.
- [36] S. Verret. Current state of the art in multirobot systems. Technical report, Defence R&D Canada - Suffield, 2005.
- [37] Luis Villa. brickOS HOWTO.
- [38] Stenzel Sp. z o.o. Robot HEXOR.
- [39] B. P. Zeigler and S. Chi. *Model-Based Architecture Concepts for Autonomous Systems Design and Simulation*, chapter 3. Kluwer Academic Publishers, 1993.