CONTROL ARCHITECTURE CHARACTERISTICS FOR INTELLIGENCE IN AUTONOMOUS MOBILE CONSTRUCTION ROBOTS

Rahee Agate*
Engineering Dept.
Lancaster University
UK, LA1 4YW
r.agate@lancaster.ac.uk

C. Pace
Manufacturing Dept
University of Malta

Derek W. Seward
Engineering Dept.
Lancaster University
UK, LA1 4YW

Mohmed.J. Bakari
Engineering Dept.
Lancaster University
UK, LA1 4YW

Abstract:
This paper details the implementation of a hierarchical control architecture based on the Real-time Control System-Reference Model Architecture (RCS-RMA)[1][2], for autonomous operation of mobile construction robots. The RCS-RMA framework offers a structure for developing intelligent autonomous behaviour. This paper extends the use of this framework by suggesting an approach to embedding different types of control models within the framework, so as to achieve the desired operational characteristics at different decision-making levels and under diverse operational circumstances. Various control methods can be used as tools for reaching a decision at a given level. The paper presents a typical multi-level architecture for a mobile robot which can be used for wide range of applications, including an autonomous construction robot. It is suggested that the use of different control methods is more appropriate than using a single approach at all levels of the hierarchy.

Keywords: RCS -RMA, Mobile robot, POMDP, PID

1. INTRODUCTION

The idea behind this paper is to suggest an approach for designing the robotic systems so as to make them more flexible, efficient and reliable. It is argued that one control methods at all levels of a hierarchical architectural scheme may not be the best. Use of a particular control method appropriate at a given level may make the system better and effective. Real time control system- reference model architecture (RCS-RMA) [2] is used as a hierarchical architectural framework.

2.REAL-TIME CONTROL SYSTEM REFERENCE MODEL ARCHITECTURE (RCS-RMA)

2.1 Robot Architectures

Architectural structure can be defined as the method by which a system is divided into subsystems, and how those subsystems interact with each other and with the external world to reach the set goal. A system 'architecture' primarily refers to the software and hardware framework for controlling the system. Architectural style refers to the computational concepts that underlie a given system. In the last decade, a number of system architectures have evolved [3].

Architectural styles described in the technical literature can be classified into three categories: deliberative, behavioural, and hybrid. A deliberative architecture tends to adopt a top-down hierarchical command structure often using reasoning based on extensive models or maps of the world. A behavioural architecture, on the other hand, tends to be based on simple low-level sense/act loops. The hybrid style combines both reactive and deliberative control in a heterogeneous architecture. It facilitates the design of efficient low-level control with a connection to high-level reasoning. RCS[2] (Real-time Control System) is a hybrid architecture in that it combines deliberative with reactive components. The approach was developed at the N.I.S.T. (National Institute of Standards and Technology) laboratory in the USA and has been applied to a wide range of intelligent systems.

RCS partitions the control problem into four basic Functional Elements (see below) along with sensors and actuators. RCS clusters these elements into computational nodes that have responsibility for specific subsystems and arrange these nodes in hierarchical layers such that each layer has characteristic functionality and timing. Each layer provides a rich and dynamic world model and a sensory processing hierarchy to keep the world model up to date.

The RCS reference model architecture has a systematic regularity and recursive structure expressed in a canonical form that provides a basis for an engineering methodology [2]

Functional Elements[1][2]: are the fundamental computational processes from which the system is composed.
"The functional elements of RCS reference model architecture are sensory processing (SP), World modelling (WM), value judgment (VJ), and behaviour generation (BG).

Value Judgment (VJ)

It is a process that
a) Computes cost, risk, and benefit of actions and plans.
b) Estimates the importance and value of objects, events, and situations.
c) Assesses the reliability of information.
d) Calculates the rewarding or punishing effects of perceived states and events.

World Modelling (WM)

A functional process that constructs, maintains, and uses a world model knowledge database (KD) in support of behaviour generation and sensory processing.
World modelling performs four principle functions:
a) It predicts (possibly with several hypotheses) sensory observations based on the estimated state of the world. Predicted signals can be used by sensory processing to configure filters, masks, windows, and schema for correlation, model matching, recursive estimation, and focusing attention.
b) It generates and maintains a best estimate of the state of the world that can be used for controlling current actions and planning future behaviour. This best estimate resides in a knowledge database describing the state and attributes of objects, events, classes, agents, situations, and relationships. This knowledge database has both iconic and symbolic structures and both short and long-term components.
c) It acts as a database server in response to queries for information stored in the knowledge database.
d) It simulates results of possible future plans based on the estimated state of the world and planned actions. Simulated results are evaluated by the value judgment system to select the best plan for execution.

Behaviour Generation (BG)
The planning and control of actions intended to achieve or maintain behavioural goals.
Behavioural goal: a desired result that a behaviour is intended to achieve or maintain.
Desired result: a result that value judgment evaluates as desirable or beneficial.

Command: a name, a commanded action, and a command goal. Both commanded action and command goal may include parameters.

Sensory processing (SP)
A set of processes by which sensory data interact with prior knowledge to detect and recognize useful information about the world.

The RCS Computational Node:[2]
A RCS_NODE is an organizational unit of a RCS system that processes sensory information, computes values, maintains a world model, generates predictions, formulates plans, and executes tasks.

Figure 1 illustrates the relationships within a single RCS_NODE of the RCS architecture. Each RCS_NODE contains BG, WM, SP, and VJ processes, plus a knowledge database (KD). Any or all of the processes within a node may communicate with an operator interface. The interconnections between sensory processing, world modelling, and behaviour generation close a reactive feedback control loop between sensory measurements and commanded action. The interconnections between behaviour generation, world modelling, and value judgment enable deliberative planning and reasoning about future actions. The interconnections between sensory processing, world modelling, and value judgment enable knowledge acquisition, situation evaluation, and learning.
Within sensory processing, observed input from sensors and lower level nodes is compared with predictions generated by world modelling. Differences between observations and predictions are used by world modelling to update the knowledge database.

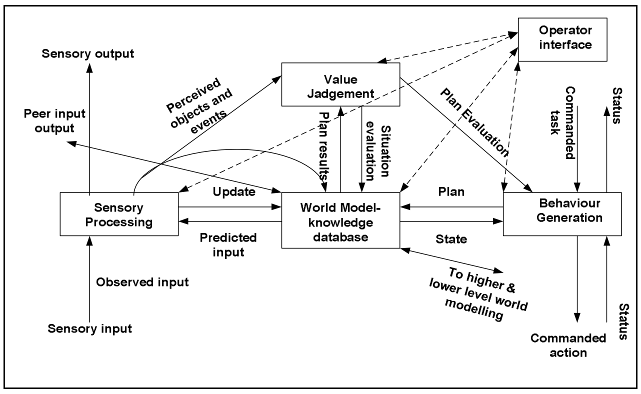


Figure 1 RCS_NODE [2]

Each RCS_NODE looks upward to a higher level node from which it takes commands, for which it provides sensory information, and to which it reports status. Each RCS_NODE also looks downward to one or more lower level nodes to which it issues commands, and from which it accepts sensory information and status. Each RCS_NODE may also communicate with peer nodes with which it exchanges information. A collection of RCS computational nodes such as illustrated in figure 1 can be used to construct a distributed hierarchical reference model architecture.

Each RCS_NODE acts as an operational unit in an intelligent system. Depending on where a particular RCS_NODE resides in the hierarchy, it might serve as a controller for one or more actuators, a subsystem, an individual machine, a group of machines comprising a manufacturing workstation, a group of workstations comprising a manufacturing cell, or a group of cells comprising a manufacturing shop. The functionality of each RCS_NODE can be implemented by a set of software processes or by a person or group of persons.

3. IMPLEMENTATION OF DIFFERENT CONTROL METHODS WITHIN HIERARCHICAL ARCHITECTURE

3.1 **Need for different control methods**

In hierarchical architectures, the application is decomposed into subtasks and subtasks to further sub-subtasks. The whole task is represented as a hierarchical architecture.
Every level has to deal with different task or subtask. Every level has different subsystem/subsystems. The top level has to deal with planning and as such it needs lower resolution, but wider scope. It's perception about the world is on a larger scale and as such not too much detail. In other words, we can say that the world model at this level is world-centric.

As one goes down the hierarchy, levels become more and more reactive and less and less deliberative. Lower levels thus need to have an ego-centric world model representation. Scope of the lower levels is narrower but the resolution is higher. They don't participate in the decision making process as much as the higher levels. They are specifically responsible for the actual physical action generation. In this regard they need to deal with the robot's immediate surroundings and any obstacle which is within its peripheral range. They have to make sure that the system actually produces behaviours precisely and smoothly. Whereas, the higher levels are responsible for decision making and planning. They do not generate the actual physical action but rather plan and command the lower level to perform this planned action .

Each level has different task/subtask to complete. Higher levels plan while lower levels perform. As such the control methods required at each level can be different. It can be argued that each control method is not the best suitable for every task/subtask at all the levels. All features of any method may not be utilised at all the levels due to the task the level has to complete. On the other hand it is possible that some methods do not offer sufficient features. E.g. the PID control method is not suitable for higher levels. But it is suitable as the lower level controller where feedback is required to ensure smooth operation. But it does not meet the demands of higher levels where planning and decision making takes place. In summary the hierarchical nature of the architecture can be visualised by considering the upper level WM to offer a world-centric representation, low in resolution but large in scope, providing a general picture of the construction site within which the robot moves. At the lower levels, the WM will take the form of an egocentric representation allowing the system to evaluate its current state and interaction with its immediate environment such that the robotic system can be considered stationary within a dynamic world. Given the variation in control requirements and scope within the architectural hierarchy, it is believed that the mode of control at each RCS level (and also at intra-level nodes) must adapt according to the characteristics that define that specific level of decision-making.

3.2 Role of RCS-RMA for implementation of different control methods [4]

In the past few decades various techniques, based on, for example, lead-lag compensation, PID control, optimal control, feedback linearization, adaptive control, robust control, fuzzy logic, neural networks, and so on, have been developed for control of linear and nonlinear plants. However these methods are typically not sufficient to achieve the levels and types of automation that are currently desired for more sophisticated and complex systems. There is often a need for more sophisticated algorithms, or 'hybrid' combinations of the foregoing algorithms, which can also, for instance, predict and prevent or compensate for faults, control the 'discrete event' part of the plant, and generally coordinate complex sequences of behaviours of a system. If the system is physically distributed, there is often a need for distributed controllers and hence communications between, and coordination of, the subsystem controllers to achieve overall performance objectives. Also, in many applications it is often convenient to break complex tasks down into simpler subtasks which can be implemented using low-level numerical algorithms (such as some of those listed above), provided that there is a ' supervisor' to coordinate the activities so that the overall task is achieved. The need for supervision of controller functions, the distributed nature of many systems, and the need in many systems for an interface to a human user (e.g. for monitoring and specifying system goals) leads to a hierarchical structure for a controller. An example of a hierarchical structure is shown in figure 2 (this is sometimes called the *organizational hierarchy* of the controller). In this diagram the boxes represent different modules of the controller, each assigned a different task (or subtask), and the lines represent communication links. The diagram is organized into *layers (levels)* where the *upper level* holds the operator interface if needed, the *lower level* typically holds traditional control functions that interface to different parts of the system via sensors and actuators , and the *middle level* typically holds modules that coordinate the actions of the low-level algorithms and carry out the actions planned by the upper level. The middle level is the one which has the decision making capacity.

RCS is a generalized architecture for intelligent systems, providing a hierarchical breakdown of tasks and control activities. Rising up the control hierarchy from low-level servo-control of individual joints to strategic planning of the whole task, the spatial resolution reduces and the time horizon increases.

4. MOBILE ROBOTS IN CONSTRUCTION

The application under consideration here is a mobile robot which can be employed for various purposes, one of them being as an excavator on a construction site. As the focus of this research is on the development of the mobile robot , navigation and safety issues are primarily considered. Other issues such as digging, moving objects etc are not considered at this point. A 4-layer robot architecture (figure 2) employing different control methods is described in greater detail in this section.

4.1 RCS design Methodology

The RCS design methodology [4] is employed in order to structure the organizational hierarchy. Task decomposition is performed so as to decide the splitting of tasks at the higher level into subtasks at lower levels. It also helps the designer to identify the tasks or operations that the whole system performs and which task is performed by which actuator(s) or subsystem(s).

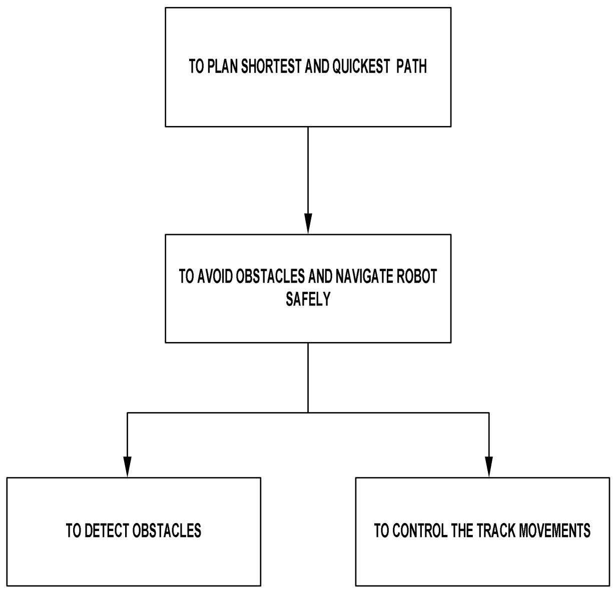

TO PLAN SHORTEST AND QUICKEST PATH
TO AVOID OBSTACLES AND NAVIGATE ROBOT SAFELY
TO DETECT OBSTACLES
TO CONTROL THE TRACK MOVEMENTS

Figure 2: Task Decomposition Analysis.

Once the tasks and subtasks are assigned to various levels, the controller architecture is defined. For defining the hierarchical architecture[4], the first and most important thing to consider is the layout of the physical subsystems and all the actuators and sensors of the system to be controlled. Each subsystem will have its own sensors and actuators (although these won't be unique to a specific subsystem). Then, based on the physical layout of the subsystems, the connections between them, the information flow, and the task decomposition analysis performed in the previous step, we define the controller architecture. This typically starts with assigning a control module to each actuator and sensor on the bottom of the hierarchy.

Consequently diverse control methods are needed to be used within RCS to achieve the desired operational goal and to provide the required level of autonomy. Furthermore, the control model applied at each control level must suit the representation and management of uncertainty, given the control resolution and scope of that level.

The following control and decision-making strategies are thus proposed for integration within an RCS framework (figure 3):

- A PIP/PID controller is considered to be the most appropriate at the lowest level consisting of sensors, actuators and their servo-feedback loops, providing a minimum level for handling uncertainty.
- At an intermediate level of control, it is argued that Partially Observable Markov Decision Processes (POMDP) offers a suitable structure for developing desired control strategies allowing the mapping of the functional elements, viz, SP, WM, VJ and BG, onto specific components of the POMDP itself. This mapping provides the means of ensuring the appropriate interaction between these elements of the control level, providing a coherent controller behaviour and decision-making process which succeeds in managing the uncertainty present at such a level of control.
- Other techniques based on downhill algorithm based path planning[5] algorithms are more appropriate at the higher planning levels.

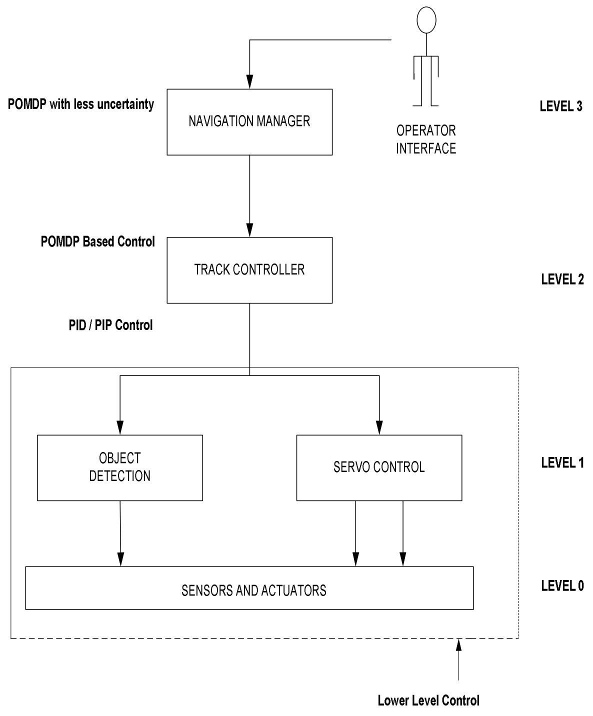


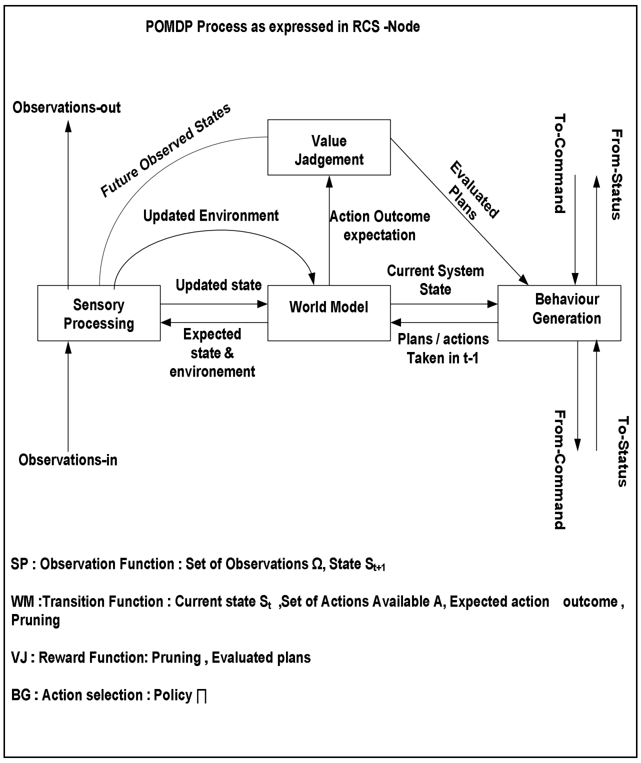
Figure 3: Control Hierarchy of Mobile Robot Architecture

Figure 3 is explained in greater details in this section:

Level 3: Nav_Manager Module.
The task of this level is to plan the path. This level can be said to be responsible for 'strategic planning'. It has most deliberative in nature and least reactive in all the layers .
It consists of a grid-based dynamic path planning algorithm. This level ensures that the task gets completed in the most cost-effective and time-effective manner. The algorithm plans the path which is optimally safe .It avoids the obstacles but does not take into account the terrain conditions while planning. It uses down-hill planning algorithm based methods[5] for path planning.
The WM(world model) at this level is world centric in nature. It can be imagined that the robot is stationary in a constantly moving world around it. The time horizon at this level is in minutes.
PID or such low level methods can not be used at this level as they can't handle the planning part .
POMDP is not required as it does not have to make decision based on various available actions/choices available.

Level 2:

Track Controller Module
This level deals with decision making to ensure the safety of operation and precise movement of the robot. This level receives the command from level above (Nav_Manager module). IT can be said that this level makes the decision and plans 'tactically' within the limits of 'strategic planning' made by level 3(Nav-manager module).
It is less reactive than level 1 and level 0, but more reactive than level 3. It is also less deliberative in nature than level 3, but more deliberative than levels below.
Ensures safety of the operation by choosing the action within the commanded actions it receives from level 3.
The WM at this level is said to be less world centric in nature and more ego-centric. It has detailed information regarding its surroundings, but it lacks the information outside its sensitive range. The information is centred on the robot.
It has time horizon in seconds.
The control method used at this level is POMDP (Partially observable Markov Decision Process)[6]. POMDP deals effectively with uncertainty and hence help reduce the unsafe behaviour. This level is responsible for safe decision making within the 'commanded action' from level 3.Other reason for using POMDP is that it can be mapped very well within the fundamental building block of the RCS-RMA, i.e. node. Figure 4 shows POMDP process within RCS node.



SP : Observation Function : Set of Observations Ω, State S_{t+1}

WM :Transition Function : Current state S_t ,Set of Actions Available A, Expected action outcome , Pruning

VJ : Reward Function: Pruning , Evaluated plans

BG : Action selection : Policy ∏

Figure 4: POMDP process development in RCS_node

Level 1:

As can be seen from figure 3, level 1 consists of 2 modules, viz object detection and servo control. Each of these modules is explained here.

- Object Detection Module

This module deals with obstacle detection. The world model contains mainly expected obstacle data. It can be highly reactive when an obstacle comes within safe operating peremeter. This module is also capable of halting the system if unexpected and sudden obstacle comes in a close vicinity of the robot. Time horizon is in ms.

- Servo Control Module

This module ensures accurate movement of robot towards the target position. Ensures smooth operation of tracks
This is also capable of halting the system if required. WM contains robot data, such as position, speed, slope, tilt angles etc.

It is important to note that this level does not take part in decision making or planning.
The control method used at this level is PID.
Both these modules at level 1 help keep system safe by reducing the uncertainty about the

1. current state of robot, e.g. position, orientation, speed, tilt angles etc
2. current state of the world, e.g. obstacle presence, speed, nature etc.

5. CONCLUSION AND FUTUTE WORK

It is believed that using varying methods at different levels of hierarchical architecture facilitates reliable, effective and the best suitable system operation.
This architecture is in the development stages at present. The next step is to implement it on the simulated mobile robot and then the future objective is to extend it to the actual mobile robot excavator and test it on the real construction site.

6. REFERENCES

[1]Albus, J.S. (1991). "Outline for a Theory of Intelligence", IEEE Transactions on Systems, Man and Cybernetics, 21 (3): p. 473-509

[2]Albus, J.S.; Meystel, A.M., 'Engineering of mind: an introduction to the science of intelligent systems' Chichester; Wiley, 2001

[3]Arkin, R.C., (1998) Behaviour-Based Robotics, MIT Press, Cambridge, MA.

[4]Balch, T.R., Grid-based navigation for mobile robots. 'The Robotics Practitioner', 2(1), 1996

[5]Gazi, V., (et al eds), The RCS handbook - tools for real-time control systems software development, Wiley, 2001

[6] Kaelbling L.P., (1998), Littman M.L., Cassandra A.R., "Planning and Acting in Partially Observable Stochastic Domains", Artificial Intelligence Vol 101, pp99-134.