Available online at www.sciencedirect.com

ScienceDirect

Procedia Engineering 69 (2014) 216 – 221

Procedia Engineering

www.elsevier.com/locate/procedia

24th DAAAM International Symposium on Intelligent Manufacturing and Automation, 2013

The Architecture of Robotics Control Software for Heterogeneous Mobile Robots Network

Kirsanov Kirill*

Laboratory "Sensorika", Miusskaya sq., 4, 125047, Moscow, Russia
Moscow state technological University "Stankin", Vadkovsky per. 3a, 101472, Moscow, Russia

Abstract

This paper dwells on the control software architecture of mobile robots from a programmer's perspective. Several approaches to the construction of such systems were considered in the case of popular systems and the author's own designs.
The need for such systems has even become increasingly obvious due to the heterogeneous nature of robotics network. The authors had to work with different types of robots, namely Sensorika AMUR 1-7, Brokk-400 and Festo Robotino XT. Here, heterogeneity refers not only to the network architecture but also to the robots themselves. Particular attention was given to programming theory, organization of a two-level control instruction pipeline, using Turing-complete protocols, and virtualization of input/output ports.

Keywords: robtics software; robotics control systems; turing-complite protocol; python; dynamic language

1. Control software architecture for mobile robots

Before we talk about the specific character of the control software (CS) architecture for mobile robots, it is necessary to clarify how the term "mobile robot" is understood by the authors. We will call a mobile robot a machine (composition of mechatronic devices and on-board computers), capable of moving independently in space. Moreover, each mechatronic device that makes up the robot is abstracted as a finite state machine (FSM).

* Corresponding author. Tel.: +7-903-745-46-77.
E-mail address: kkirsanov@gmail.com

doi:10.1016/j.proeng.2014.02.224

Thus, with relation to the control software, the control task is implemented by changing the state of these state machines, even if this is not declared by the authors of the control software. The term “Mobility” implies the need for networking. Besides, the need for a network component arises not only and not just because of the "mobility" of robot, but also in the development of control software, when the developer had to work from remote control devices such as a laptop.

From the foregoing it follows that the key features of the control software of a mobile robot should depend:

- on the way in which the rules for changing the states of FSMs that make up the mobile robot are set
- on how these changes are transmitted via the network to the control software and mobile robot
- on how identification and addressing of mechatronic devices are carried out

Additionally, having considered the most popular CS build system for MR, such as:

- ROS (Robot Operating System) - http://wiki.ros.org/
- Robotino view - http://www.festo-didactic.com/int-en/services/robotino/
- Microsoft Robotics Studio - http://msdn.microsoft.com/en-us/robotics/default.aspx
- Player Project - http://playerstage.sourceforge.net
- ORCA2 - http://orca-robotics.sourceforge.net/orca/index.html
- LAAS/GenoM - https://softs.laas.fr/openrobots/wiki/genom
- Marie - http://marie.sourceforge.net
- URBI - http://www.urbiforge.com
- Webots - http://www.cyberbotics.com
- RoboJRE - http://www.ridgesoft.com/robojde/robojde.htm
- OROCOS - http://www.orocos.org
- iRobot - http://www.irobot.com/

It is possible to classify of CS development systems user interfaces for:

- Structure: The library or framework
- The process for creating programs: visual or classical programming
- The characteristics of programming languages used for development: interpreted and compiled, static and dynamic, strict and not strict, functional and imperative, etc.
- By the networking nature, they can be divided into those using or not using Remote Procedure Call (RPC), such as XML-RPC or .NET Remoting (Microsoft Robotics Studio) and those using their own high-level shared open network protocols.

When developing the control software for Autonomous Mobile University Robot (AMUR), the authors had to develop their own software, which both had similarities with existing systems and a number of fundamental differences. The reason for this was the heterogeneity of the network of mobile robots is not fully supported by the above software.

1.1. CS architecture. Framework versus Library

The framework differs from the library based on how the programmer treats his programs. In the case of framework, the programmer builds his program in the framework, while for libraries; he embeds his library code into his program. In the case of the control software of a mobile robot, the choice is defined by the person creating the main control loop – the control software developer or the person who developed the framework used. Thus, the choice of a particular structure depends on the nature and complexity of the system being developed. When it comes to students’ learning task, it would be better to avoid unnecessary framework development details. This also applies to mass industrial robots. However, in the case of R&D, framework restrictions would hamper effective work. For example [6’]. It is good architecture but it is hard to implement. Besides, when it comes to integrating heterogeneous