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

mobile robots into a single system (in this case, “AMUR” and Festo “Robotino” XT robots), the library structure is much simpler than framework.

1.2. A method for creating programs

In the last few years, there has been growing popularity in the use of visual programming systems (VP systems) [1] in robotics problems, such as Microsoft Visual Programming Language from Microsoft Robotics Studio, Labviw and Robotino View. Visual programming is a method of creating programs by manipulating program elements graphically rather than by specifying them textually. This approach enables you improve the clearness of presentation of the program, but it is only for a very narrow class of problems. In solving general-purpose problems, such languages as C or Java become more benefic.

1.3. Characteristics of programming languages.

The vast majority of modern control software for mobile robots are created using general-purpose high-level static strict languages, such as C, C#, and Java. Programming languages are divided primarily into general-purpose languages (languages used for a large variety of tasks) and domain-specific languages (DSL), which are specialized to a particular application domain, providing higher efficiency but less versatility. These same languages are divided into compiled and interpreted languages. When compiling, the source code is converted into a machine code and stored in a file. For interpreted programming languages, the program is executed line by line. Compiled languages provide faster speed (in computational tasks, C is 60 times faster than Python) while the resources of the on-board computer are saved more. Interpreted (scripting) languages are faster in developing and debugging. The performance comparison of different languages on the same tasks can be viewed at http://benchmarksgame.alioth.debian.org.

The type system of a language describes how a language regulates work with its data. Type systems may be variously strong or weak typing, static or dynamic typing. Unlike weak typing, strong typing requires you to define and strictly follow the types declared by the programmer when writing the program. Dynamic typing involves assigning types directly in the computing process, while static typing – in the programming phase. Selection of a particular type system determines the distribution of the programmer’s efforts between debugging and developing. A weak dynamic typing allows you to quickly write a program (PHP, which is a weak dynamic type system, is the most popular language among websites) but requires a fairly long debugging. On the other hand, strict strong typing requires longer design process but eliminates simple errors at the testing and debugging phases.

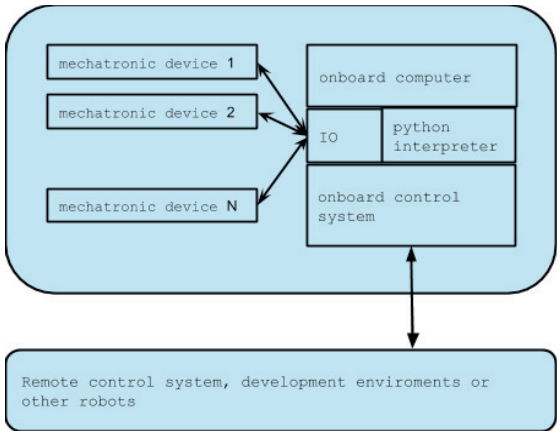

Fig. 1. CS concept.

Therefore, a particular approach to typing should be chosen based on actual needs. The aerospace sector and other relevant industries usually use strict strong compiled languages (C, C #, Java or own DSLs with the same properties), whereas in education and R&D, dynamic interpreted languages (Python, Matlab) are preferred. Functional languages could not be useful outside the academic sphere [3].

After analyzing all these features, the authors selected Python as their main development language and proposed the following concept: Via hardware input/output ports (such as COM, CAN, I2C or USB), mechatronic devices are connected to the onboard computer where the onboard software code is executed. The onboard software secures networking with the remote control, other robots and the developer console (Fig 1).

This network interaction can be implemented in several ways. This paper considers interaction technologies based on the application layer of technology stack TCP\IP.

2. Network

As shown earlier, networking is used to request or change the states of the mechatronic devices that make up a mobile robot. New states can be formed in all kinds of ways, ranging from user interfaces (joystick) and ending with calling from the program text. Regardless of the method by which the state is formed, it should be transmitted. Known packets offer programmers with two approaches:

- Own protocols when the developers of the framework or library used describe the necessary command sequences to be transmitted to onboard control software in text form. For example, "the engine speed is controlled by the command SETSPEED X Y, where X is the engine number and Y is the speed range from 0 to 255"
- A remote procedure call (RPC), when the robot commands are abstracted using the language constructs of a particular programming language. This requires additional components, modules and libraries that may not be available to the development operating system and language. For example

```
import robot
r=robot.connect ("192.168.1.10")
r.setSpeed(10)
r.whait(1)
r.stop()
```

The developed control software combines both approaches – RPC library is implemented for Python language and at the same time, JSON encoded commands are accepted. So, to interact with the system, you can use any programming language without additional technology stacks as in the case of Microsoft Robotics Studio or ROS.

3. IO ports virtualization

The problem of integrating heterogeneous mobile robots into a single network also equally requires a separate attention. It is often not possible to provide a uniform access method to all mechatronic devices because this requires creating the appropriate drivers and loading them into the onboard computer. In this case, the authors used an I/O port network visualization system (fig 2):

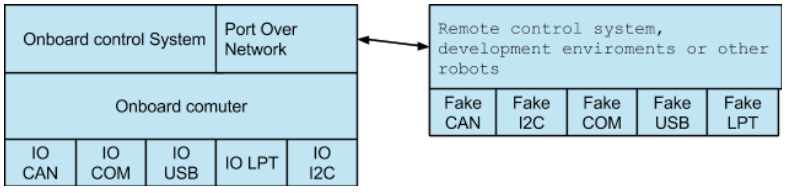

Fig. 2. IO port virtualization.

In this case, "mirrors" of input/output ports are programmatically created on the remote computer (such as the developer's computer) with the same features and timeouts. This allows you to use proprietary control software without any networking intention, or to easily integrate it into your own designs. However, this requires additional work when creating software drivers for the mechatronic devices since it is necessary to support a transition from a port virtualization mode to a command mode and back. This approach resembles the approach to virtualization in virtual machines. For example, XEN [3].

4. Two-level instruction pipeline

When using a turing-complete protocol approach [5] to manage multiple heterogeneous [6] robots (Sensorika AMUR 1-7, Brokk-400 and Festo Robotino XT) was necessary to create the instruction pipeline on CS (Fig 3):

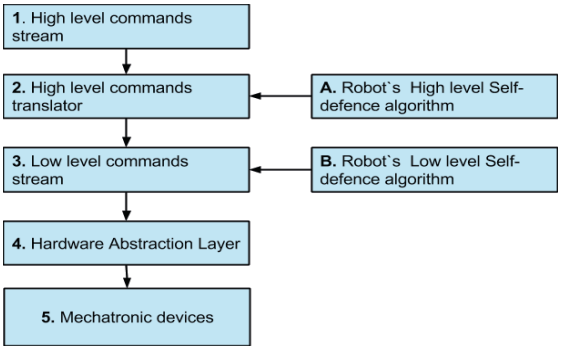

Fig. 3. Two-level instruction pipeline.

A high-level commands stream **1** is received at the input. This stream is translated into low-level commands stream **2**, which is transformed taking into account the robot's high-level self-defense constraints **A**. Low-level commands stream 3 is similarly transformed taking into account the robot's low-level self-defense constraints **B** and enters the transformation lower level **4**

5. Conclusion and results

There are many ways and techniques of building control software in robotics. From a programmer's perspective, each of them has its advantages and disadvantages. However, in research and development, the most optimal path is the one proposed in this paper and in [5]. Software listed in Part 1.1 perfectly solves the problem, whether it be industry or education but has trouble with heterogeneous robots network. Some of it are closed in one ecosystem, whether the manufacturer or programming language. Due to the widespread popularization of cloud robotics it is necessary to use more open than earlier solutions.

Python language, dynamic typing, turing-complete protocols, two-level pipeline, language independent network protocol - all these greatly speeds up the development of control software for mobile robots. In addition, they enable the developer quickly move from the design stage to the implementation of control software.

Nomenclature

CS	control system
MR	mobile robot
MD	mechatronic devices
FSM	finite state machine
RPC	remote procedure call
DSL	domain specific language

References

[1] T. R. G. Green , M. Petre «Usability Analysis of Visual Programming Environments: a `cognitive dimensions' framework» Journal of Visual Languages & Computing Volume 7, Issue 2, June 1996, Published by Elsevier Ltd. pp 131–174.
[2] Nenad Medvidovic, Hossein Tajalli, Joshua Garcia, Yuriy Brun, Ivo Krka, and George Edwards, «Engineering heterogeneous robotics systems: A software architecture-based approach», IEEE Computer, vol. 44, no. 5, May 2011, pp. 61–71.
[3] J. Peterson, P. Hudak, C. Elliott, «Lambda in Motion: Controlling Robots with Haskel» Proceeding PADL '99 Proceedings of the First International Workshop on Practical Aspects of Declarative Languages Springer-Verlag London pp 91-105.
[4] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, A. Warfield Xen and the Art of Virtualization Proceeding SOSP '03 Proceedings of the nineteenth ACM symposium on Operating systems principles ACM New York, NY, USA ©2003 pp 164-177.
[5] K. Kirsanov,»Development and debugging of information measurement and control systems for mobile robots with python dynamic languages». Annals of DAAAM & Proceedings 2010 DAAAM International Vienna ISSN: 1726-9679, pp 1835-1836.
[6] Andreev V., Pryanichnikov V., Prysev E. «Multi-access control of distributed mobile robotic systems on the base of networking technologies» // Annals of DAAAM for 2010 &Proceedings of the 21st International DAAAM Symposium «Intelligent Manufacturing & Automation: Focus on Interdisciplinary Solutions». Zadar, Croatia. 20-23rd October 2010, pp .15-16.