Available online at www.sciencedirect.com

ScienceDirect

Procedia Engineering 100 (2015) 278 – 282

Procedia Engineering

www.elsevier.com/locate/procedia

25th DAAAM International Symposium on Intelligent Manufacturing and Automation, DAAAM 2014

# Software Architecture of Control System for Heterogeneous Group of Mobile Robots

Kirsanov Kirill*

*International Institute of New Educational Technologies, RSUH, Miusskaya sq. 6, Moscow, 125993, Russia*

**Abstract**

This article describes the software architecture, which can significantly reduce development time, provide an effective process of data exchange and control of mobile robots (or a group of mobile robots). This architecture is based on representation of a mobile robot as the composition of mechatronic devices connected in a hierarchical graph of software interactions (also known as middleware systems). This architecture provides high robustness, capacity and transmission frequency of control commands and data. Also, successful results have been achieved in providing dynamic reconfiguration of system components without stopping as well as automatic crash recovery (including complex interaction graphs) and auto-configuration.

Peer-review under responsibility of DAAAM International Vienna

*Keywords:* Control system; mobile robotics; software architecture; software engineering; middleware

## 1. Introdution

For denotation of this article describes the software architecture, which can significantly reduce development time, provide an effective process of data exchange and control of mobile robots (or a group of mobile robots). This architecture is based on representation of a mobile robot as the composition of mechatronic devices connected in a hierarchical graph of software interactions (also known as middleware systems).

This architecture provides high robustness, capacity and transmission frequency of control commands and data. For example, during operation via Wi-Fi in point-to-point mode, control commands and sensor data transmission frequencies of 450-500 Hz (Wi-Fi) were successfully provided.

* Corresponding author. Tel.: +8 903 745 46 77; fax: +8 903 745 46 77.
*E-mail address:* kkirsanov@gmail.com


Peer-review under responsibility of DAAAM International Vienna
doi:10.1016/j.proeng.2015.01.368

Also, successful results have been achieved in providing dynamic reconfiguration of system components without stopping as well as automatic crash recovery (including complex interaction graphs) and auto-configuration. The software designed for interaction of application software with network and providing unification of program-to-program interactions in the context of heterogeneous system platforms [2], the term "Middleware" will be used. Such software serves as a subsystem in larger software. And despite the fact that middleware is comprehensively studied in modern Database, Internet and distributive calculation technologies, the studies have not reached the adequate level as for robotics. Apparently, this is due to the fact that robot software had been quite simple until recently (from the perspective of software engineering) and has not gone beyond the most primitive interactions between objects and subjects of controlling process.

For the last few years, several teams were trying to develop heavy-duty software for programming of mobile robots. Microsoft Robotics Developer Studio (MRDS) [5], Robotics Operating System (ROS) [6] may be referred to as one of the most successful attempts to build such software. Their predecessors, such as Palyer Project (http://playerstage.sourceforge.net), LAAS GenoM (https://softs.laas.fr/openrobots/wiki/genom), URBI (http://www.urbiforge.com), etc., do not make progress any longer. ROS and MRDS are basically frameworks repeating the functionality of operating systems (hardware abstraction, low-level device control, transmission of messages between processes and package management) as applied to robotic tasks; they are based on a graph architecture, in which data processing is performed within the nodes that can receive and transmit messages between each other.

Despite the use of graph architecture, realization of complex, multilink (more than 50) and multiuser systems in them becomes difficult due to architectural limitations.

These frameworks may be nominally referred to as the "large", i.e. including a number of components for SLAM tasks solution, sensor calibration, image processing, etc. However, "small" robotic frameworks that only perform transmission of messages between components also exist, e.g. Concurrency and Coordination Runtime (included in MRDS).

## 2. Related work

At the development of middleware for building a distributed Virtual Laboratory [4], the following requirements were formulated:

- Simultaneous functioning, control and surveillance (acquisition of telemetric information) by the network consisting of hundreds of mechatronic components as well as shared remote access of multiple users to the single mechatronic device.
- Delimitation of rights based on hierarchical representation of components

Additionally, due to heterogeneous nature of the network, the following limitations were applied to the developed architecture:

- Multiplatformity. The possibility of functioning under various platforms and operating systems - x86\arm\windows\Linux\android
- Minimalism. For functioning on Android devices and tiny single-board industrial computers, it is necessary to minimize the size of software components
- Low barrier of entry. As the system is developed for use in educational institutions as well, it is necessary to make it intelligible for people that do not have sufficient knowledge in the area of robotics and/or programming.

The existing systems do not meet these demands to the full extent. In this regard, a custom architecture was developed to meet them:

Table 1. Comparation chart.

| | MSRS | ROS | LAAS, GenoM, Marie, ORCA2, Player, iRobot | OROCOS | URBI | Webots | RoboJRE |
|---|---|---|---|---|---|---|---|
| Windows | + | + | + | + | + | + | + |
| Linux | | + | + | + | + | + | + |
| windows\ARM | | + | | | | | + |
| linux\ARM | | + | | | | | + |
| Development language | *.NET | C, python java | C++, pyton | C++ | URBI | C++ | Java |
| Modeling | + | + | | + | | + | |
| Interpreter | | | - | + | | + | |
| Development environment | + | - | - | - | + | + | - |
| Reconfiguration during operation | partially | + | partially | partially | - | partially | partially |
| Communication protocol | XML/ IP | Own/IP | Own/IP | Own/IP | Own/IP | Own/IP | Own/IP |
| Events | + | + | + | + | + | + | + |
| Runtime environment | .NET | | | | | | Java |
| Shared access | partially | partially | - | - | - | - | - |
| Delimitation of rights | - | - | - | - | - | - | - |
| Size | >500mb | >100mb | >50mb | >50mb | >50mb | >10mb | >100mb |

## 3. Proposed architecture

The multilevel architecture was developed, which includes: transport and logic level, distributed decentralized name service, decentralized directory service, mechanisms of automatic configuration, network crash recovery, caching and verification.

The minimal control object is represented by the mechatronic device, which has a pre-spawned integrating driver. Integrating driver is software integrating the mechatronic device into the developed information system and providing the above-mentioned functions.

The driver provides:

- Naming and identification scheme, e.g. the method for obtaining access to the device by its name. At that, the name may contain IP address and port, user-defined text (for example, “engine”), or the entire hierarchy path (for example, “University->laboratory1->Robottiono->Engines->3->current”.
- Transmission and receipt of commands. The developed architecture adopts ZeroMQ message exchange system (http://zeromq.org/) that is the basis for multiplatformity, since this library is designed for a number of programming languages and processor architectures thus allowing to organize data exchange in heterogeneous environment. Besides 1:1 interaction modes commonly used for network interaction for Berkeley sockets, it provides 1:n and n:1 modes, including those with delivery control, which is extremely convenient for robotic tasks, e.g. data acquisition from multiple sensors.
- Control commands and data are transmitted in JSON format. This is a text data transmission format (e.g. presentable as ASCII text) allowing manual formation of the control commands without using specific libraries, as it is needed by other similar systems. Also, there is a possibility to supplement the transmitted data with additional information without modifying the receiving software.

**Interaction with other drivers.**

Within the scope of this architecture, the mobile robot is represented as the composition of one or several mechatronic devices and their drivers. Each driver on the on-board computer is basically an individual computing process, and the cooperation of these processes must be synchronized and organized. Furthermore, within the scope of the Virtual Laboratory it is necessary to provide cooperation of these processes under different computing platforms. In order to ensure cooperation under such conditions, the following approaches are used:

- Each command or piece of sensor data receives a time tag at passing each driver; this allows to synchronize operation accurately and consider time delays as well as time difference between several laboratories
- Each driver has its own Name Service keeping information about adjacent drivers (with regard to network topology). This service periodically scans the local network and searches for similar services. After finding, it exchanges all available information with the found service by means of the algorithm related to Distributed Hash Tables (DHT). This is what brings in the quality of decentralization. For example, a robotic system debugged within the laboratory without changes may be disconnected from it, and its normal functioning will not be interrupted in such case.

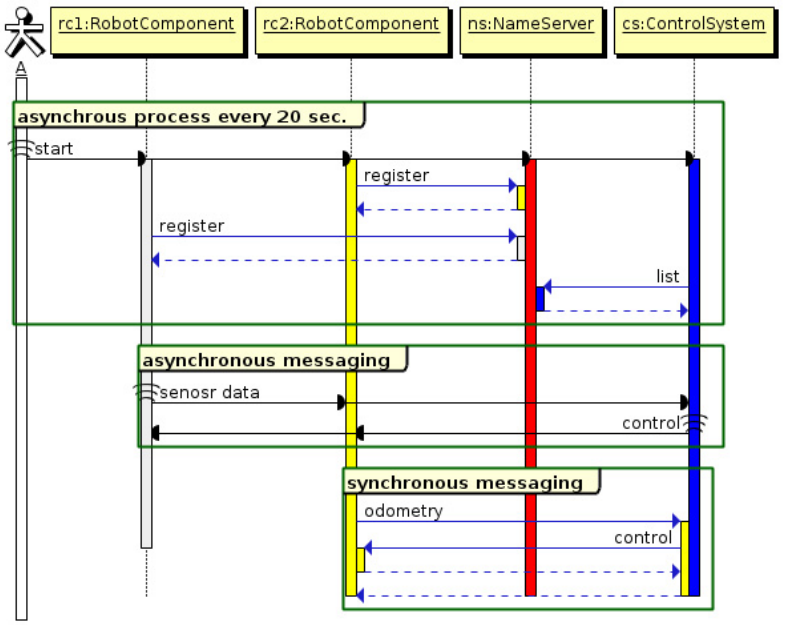


Fig. 1. Sequence diagram of messaging.

The fig.1 figure illustrates a simple interaction in the proposed system: Two mechatronic devices (rc1,rc2) areregistrating on Name service (ns) and controlled by Control system (cs)

**Conclusion**

This architecture provides high robustness, capacity and transmission frequency of control commands and data. For example, during operation via Wi-Fi in point-to-point mode with Festo Robotino[4], control commands and sensor data transmission frequencies of 450-500 Hz (Wi-Fi) were successfully provided.

Also, successful results have been achieved in providing dynamic reconfiguration of system components without stopping as well as automatic crash recovery (including complex interaction graphs) and auto-configuration.
The most significant distinctions from the existing systems are:

- The use of text data exchange format, which allows to simplify debugging and organize interaction with system without using additional libraries, and that's essential for integration purposes. The text format itself (json) is simple enough to be realized (parsed) on low-performance processors, including AVR (Arduiono)
- Decentralization and auto-configuration – the interaction graph is automatically restored after interruption of communications, even in case of minor changes in network topology
- Access delimitation system (based on LDAP) allowing for management of user group access rights.

## Further research

It appears interesting to increase the system's reliability by means of Hindley-Milner algorithm [7] for output and control of types, because untyped JSON is used for interaction, and each component of the system decomposes it independently, orienting on defaults.

The other direction is addition of support of Turing-complete protocols [3]. This, however, is due to a set of difficulties related [8] to the necessity of binding to a certain programming language.

## References

[1] Pieter Hintjens "ZeroMQ: Messaging for Many Applications" O`Relly march 2013 495pp.

[2] Wolfgang Emmerich, Mikio Aoyama, Joe Sventek The impact of research on the development of middleware technology // ACM Transactions on Software Engineering and Methodology. — N. Y.: ACM, 2008. — Т. 17. — № 4. — С. 19-48.

[3] Kirsanov, Kirill. "Development and debugging of information measurement and control systems for mobile robots with python dynamic language." Annals of DAAAM& proceedings (2009).

[4] V.P.Andreev, K.B.Kirsanov, P.F.Pletenev, Yu.V.Poduraev,V.E. Pryanichnikov, E.A.Prysev Virtual Spatially-Distributed Scientific and Educational Laboratory for Remote Control of Mechatronic Devices through the Internet 25th DAAAM International Symposium on Intelligent Manufacturing and Automation, 2014. Procedia Engineering ISSN 1877-7058. IN PRESS.

[5] http://www.microsoft.com/robotics/

[6] http://www.ros.org/

[7] R. Milner A theory of type polymorphism in programming Journal of Computer and System Sciences Volume 17, Issue 3, December 1978, Pages 348–375.

[8] Andreev V.P. Scientific and Educational Spatially-Distributed Virtual Robotics Laboratory / Andreev V.P., Kirsanov K.B., Pryanichnikov V.E. // Extreme robotics. Proceeding of the International Scientific and Technological Conference. – Saint- Petersburg: "Politechnika-service", 2014. - P.239 – 244.