DISTRIBUTED AUTOMATION AND CONTROL SYSTEMS

Eugene M. Burmakin, Boris A. Krassi

Helsinki University of Technology
St. Petersburg State Technical University

Address: Otakaari 1, HUT, FIN-02015, Finland
E-mail: {Eugene.Bourmakine,Boris.Krassi}@hut.fi

Abstract – *Distributed automation and control systems are considered from the implementation point of view. The role of middleware and, in particular, the object-oriented open-standard-based CORBA is discussed and its usefulness for system integration and communication abstraction is shown on a practical example of an experimental robotic system.*

1. INTRODUCTION

Nowadays, the level of integration determines the effectiveness of modern technical systems [1]. With respect to technology, integration has three principal constituents: standardization, automation and rationalization. Application of the system approach and functional and modular decomposition results in that systems become distributed, i.e. consist of a number of physically and/or logically distributed components that constitute the system to reach the common goal. Communication between such components is a crucial issue. The experimental work, which is described in this paper in section 5, has proved the necessity of the utilization of middleware [2], and an emergent CORBA technology [3], that may become the flexible glue for connecting different components of distributed automation systems. This issue is somewhat related to the problem that emergent technologies in the domain of information technologies and communications have been being implemented mostly in e-business applications that have much less inertia to evolve than industrial automation, which utilizes only proved and reliable technologies. Also, it should be mentioned that further development of automation systems depends on the progress in other related applied and fundamental disciplines [4].

2. DISTRIBUTED SYSTEMS

A key property of modern automated systems is *the intensive cross-communication and interaction* between systems' entities and their dynamically changing environment. Consider, for example, a decentralized control of satellite formations [5]. Spacecrafts can *co-operatively* track planned maneuvers while processing only local measurements because centralized real-time control is not effective due to long communication links. The same concept can be exploited for group control of ship, aircraft, cruise missile [4], and other systems that operate in dynamic environments and are highly co-operative, for instance, hardware and software *mobile agents*. In contrast, home automation systems comprise a number of devices from *different vendors* that can be governed by a central controller [6]. Another important application is SCADA systems [7]. A SCADA system consists of a central host, a set of field points for data acquisition and control, and special software. One of the most popular SCADA systems is the one of National Instruments [8].

The main features of the above-mentioned systems are:

- The systems comprise physically and/or logically distributed components (entities);
- The entities are essentially heterogeneous (different architecture, hardware, networking, operating systems and software);
- Cross-communication and co-operation between the entities and their environment are key features;
- The entities act as a unity to achieve a common goal.

A system that has these features is the distributed one. In distributed systems the resources are shared and logic of the system is distributed among its components. Development of distributed systems is a challenging task and the challenges are: heterogeneity, openness, scalability, concurrency, transparency and mobility [9, 10].

3. MIDDLEWARE

Communications between heterogeneous entities of the distributed system play a crucial role. There are different approaches to communications abstracting in the heterogeneous distributed system. On the one hand, the low-level approach requires a developer to be concerned with many details of the interaction between

components of the system, like the packaging of complex data into simple transferable messages. On the other hand, high-level approach is tightly coupled with a particular programming environment and limited to one programming language. The middleware approach lies in between. The idea is introduce a new layer "in the middle", between the application and the network, that hides the complexity of communication and data transfer. From the developer point of view, the invocation of a remote procedure should appear no different than the invocation of a local one.

Middleware can be defined as "software (glue) that connects two otherwise separate applications" [2], and as "a software layer that provides a programming abstraction as well as masking the heterogeneity of the underlying networks, hardware, operating systems and programming languages" [9] that is designed to solve the above-mentioned challenges of distributed systems.

In automation systems, middleware can be considered as an abstracting and uniting layer between entities' hardware, their low-level control, and the high-level control of the entire system. It is an advantage of middleware-based systems that various control architectures, e.g. group, distributed, supervisory control, client-server interaction etc., can be implemented within the same generic middleware-based platform. Middleware can provide interaction of sub-systems and modules within a system (module-based robotics, SCADA), and interoperability among distinguished but co-operative systems (industrial robots and an automated enterprise) [11].

The most challenging issues related to automation and, in particular, robotic systems with middleware are: reliability (fault-tolerance) and (hard) real-time operations. Those properties depend not only on middleware but do on the entire system (hardware and software platform).

4. CORBA-BASED AUTOMATION SYSTEMS

The Common Object Request Broker Architecture (CORBA) is a widely recognized by the control community [12] object-oriented open-standard-based middleware [OMG2002] for distributed systems integration. CORBA supports interoperability, portability and reusability of systems' components. There is a number of CORBA implementations and mappings to various programming languages that are available from different vendors. Also, its real-time and fault-tolerant specifications have been developed recently [3].

Architecture of a typical CORBA-based system [13] is presented in Fig. 1.

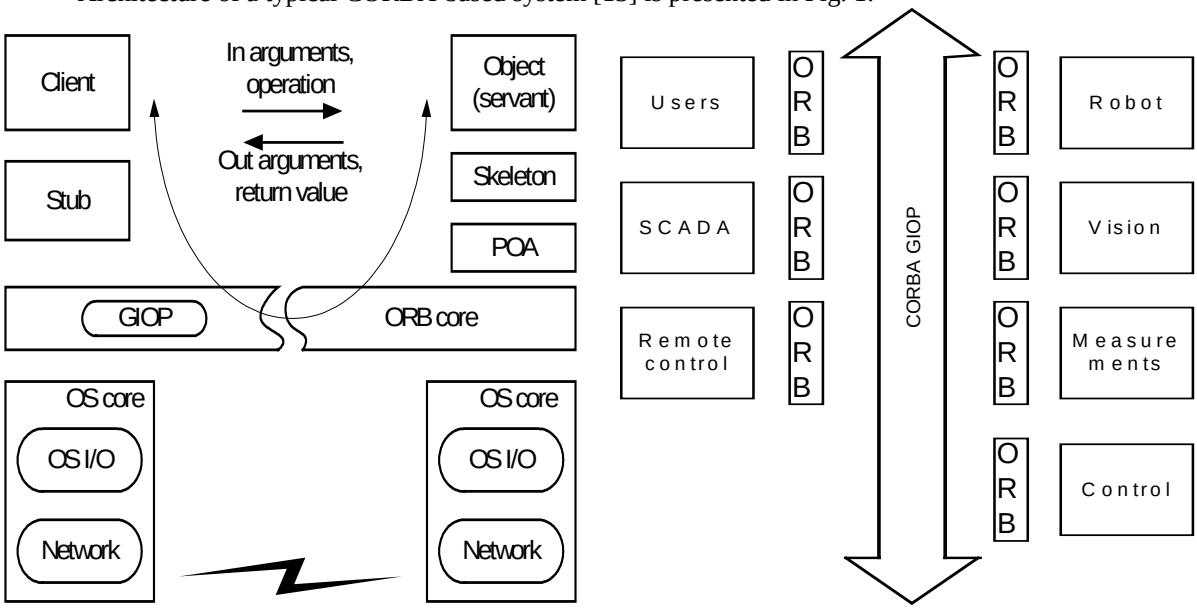

Figure 1. CORBA-based architecture

CORBA component's interface is described in a neutral Interface Definition Language (IDL). CORBA IDL is a declarative non-programming language; it does not provide any implementation details. The methods that are specified in IDL can be implemented with any programming language such as C++ and Java using rules defined by OMG. The Object Request Broker (ORB) is an object bus that provides a transparent communication means for objects to send/receive requests/responds to/from other objects. The General Inter-ORB Protocol (GIOP) provides interoperability (a set of message formats) between ORB implementations from different vendors. The GIOP can be mapped onto different telecommunication protocols for transport protocol independence. Thus, the Internet Inter-ORB Protocol (IIOP) is a mapping of GIOP onto the TCP/IP protocol stack.

5. A PRACTICAL EXAMPLE

A prototype of a robotic system (Fig. 2) has been developed to study most recent innovations in communications and information technology [14]. The purpose was to develop a prototype of an industrial robotic system, which on the one hand would be simple enough to keep full control under the system and its environment, and on the other hand would be sufficiently complex to reflect all the important features and yield the results useful for further development of industrial robotic systems.

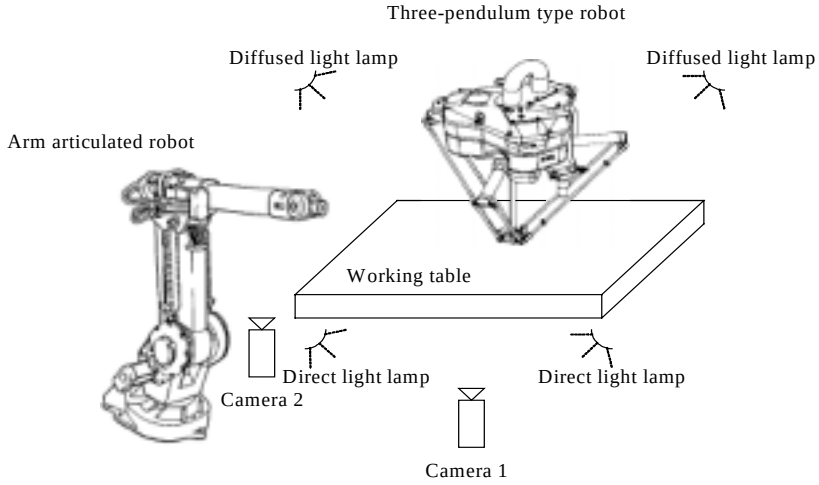

Figure 2. Experimental robotic system

The experimental robotic system consists of a *robotic cell*[1], a *machine vision system*, and a *general controller*. The general controller is a piece of software, which closes the loop "the robotic cell – the machine vision system" and keeps track of user's commands. The machine vision system deals with *model uncertainties* and *external disturbances*. The system is designed to be *flexible* enough to obey *unpredictable commands of a user*. The software subsystem has four levels. The *robotic level* software is a set of simple atomic subroutines executed on robots[2]. The *interface level* software covers all details of the low-level communication and synchronization. The *application level* software utilizes the high-level functionality provided by the interface level software. It implements the logic of action of the system as a whole and, therefore, has a very high level of abstraction. The *user interface level* software provides user's input and introduces into the system some elements of interactivity.

The experimental robotic system is a typical distributed system. Logic and resources are shared among the heterogeneous entities of the system. All components of the system have network capabilities, but operate under different operating systems with different programming languages. Hence, it could be efficient to use CORBA for system integration and communication abstraction. But *currently* the robots do not have any CORBA capabilities. Instead, an obsolete non-object-oriented RPC (Remote Procedure Call) is used. As a result, almost 75% of development time was consumed by the interface level software development and system integration. In other words, the negative experience has proved the concept of usefulness of middleware and CORBA in particular. The main result of the practical work is *the* very *proof* of the concept as well as *the experimental system* itself that can be used for further research and development in the domain of industrial distributed automation and robotic systems.

6. CONCLUSIONS

Distributed automation and control systems are considered from the implementation viewpoint. Theoretical and practical aspects of middleware in distributed automation systems and, in particular, the object-oriented open-standard-based CORBA middleware are discussed. The usefulness of middleware for system integration and communication abstraction is shown on a practical example of the experimental robotic system. The main result is that middleware can significantly reduce development time, increase interoperability, portability and reusability of automation system's components and subsequently increase usability and competitiveness of a middleware-based automation system. Future work is related to the issues of hard real-time

[1] The robotic cell has two ABB industrial robots: arm-articulated IRB 1400 and three-pendulum type IRB 340.
[2] For example, move an object from one position to another etc.

and reliable operations, exploitation of design patterns, context awareness [15], and mobile re-configurable automation distributed systems.

7. ACKNOWLEDGEMENTS

The authors wish to thank all their colleagues at the Industrial IT Lab, Helsinki University of Technology, Finland, and especially their supervisor *prof. Juha O. Tuominen*. Also, attention and support of *prof. Leonid V. Babko*, St. Petersburg State Technical University, Russia, are gratefully acknowledged.

8. REFERENCES

[1] Mäntylä, M., Andersin, H., *Enterprise systems integration*, Helsinki, HUT, 1998, 313 p.
[2] Britton, C., *IT Architectures And Middleware*, Addison-Wesley, 2001, 296 p.
[3] *Object Management Group (OMG), The CORBA Architecture Specification*, <http://www.omg.org>
[4] Iuerevich, E.I., *Robotics*, St.-Petersburg, 2001, 300 p., (in Russian)
[5] Carpenter, J.R., *Decentralized Control Of Satellite Formations*, Int. J. of Robust and Nonlinear Contr., vol. 12, p. 141-161
[6] Wu, Q., Wang, F.-Y., Lin, Y., *A Mobile-Agent Based Distributed Intelligent Control System Architecture For Home Automation*, IEEE, 2001
[7] *An Introduction To SCADA Systems*, <http://members.iinet.net.au/~ianw/primer.html>
[8] *National Instruments*, < http://www.ni.com>
[9] Coulouris, G., Dollimore, J., Kindberg, T., *Distributed Systems: Concepts And Design*, 3rd edition, Addison-Wesley, 2001, 772p.
[10] Burmakin, E., *Developing A Framework For Building Distributed Systems Operating In Mobile Environments*, Master's thesis, HUT, 2002
[11] Jia, S., Takase, K., *An Internet Robotic System Based CORBA*, Proc. IEEE Int. Conf. on Rob. & Automation, 2001, p. 1915-1920
[12] Sanz, R., Clavijo J., Segarra, M., et al., *CORBA-Based Substation Automation Systems*, Proc. IEEE Int. Conf. on Contr. Applications, 2001, p. 773-777
[13] Siegel, J., *CORBA Fundamentals and Programming*, Wiley, OMG 1996, 693 p.
[14] Krassi, B., *Control Of Industrial Systems, Which Are Based On Robots With A Parallel Structure*, Master's thesis, HUT, 2001, 76 p.
[15] Burmakin, E., Krassi, B., *Design Patterns For Development Of Dynamic Distributed Automation Systems*, BOAC 2002