Middleware for Robotics: A Survey

Nader Mohamed, Jameela Al-Jaroodi, and Imad Jawhar
The College of Information Technology
United Arab Emirates University
Al Ain, P.O. Box 17551, UAE, {nader.m, j.aljaroodi, ijawhar}@uaeu.ac.ae

Abstract—**The field of robotics relies heavily on various technologies such as mechatronics, computing systems, and wireless communication. Given the fast growing technological progress in these fields, robots can offer a wide range of applications. However real world integration and application development for such a distributed system composed of many robotic modules and networked robotic devices is very difficult. Therefore, middleware services provide a novel approach offering many possibilities and drastically enhancing the application development for robots. This paper surveys the current state of middleware approaches in this domain. It discusses middleware challenges in these systems and presents some representative middleware solutions specifically designed for robots. The selection of the studied methods tries to cover most of the middleware platforms, objectives and approaches that have been proposed by researchers in this field.**

Keywords—**robots, middleware, robot system integration**

I. INTRODUCTION

The advances of technology in computing, wireless communication, mechatronics, and sensor technologies is pioneering an emerging field of robotics, and offering an unprecedented opportunity for a wide array of real time applications. Examples of these new applications are Search-and-rescue (SAR) missions in dangerous environments or inaccessible terrains, human-assistance for the elderly or physically challenged, and medical/surgical robots. Future robots for upcoming applications should be modular for easy and rapid implementation, flexible, maintainable, customizable, self configuring, self-optimizing, and able to interact with other systems like senor networks and enterprise information systems.

Modern robots are considered complex distributed systems consisting of a number of integrated hardware and software modules. The robot's modules cooperate together to achieve specific tasks. These modules are sensors, actuators, and controllers. Due to their tight integration to the physical world and unique characteristics, robots in general pose considerable impediments and make the development of robotic applications non-trivial. There must be new software services that glue all of the components together in an efficient manner, supporting concurrency-intensive operations and insuring robustness and modularity. A friendly user programming interface that executes applications and marshals the high level constructs of the programming language to the low level constructs understandable to the operating system should be provided. The middleware should be customizable to different scenarios, applications and environments it should also be self-configuring, self-adapting, and self-optimizing. Indeed the need for a middleware layer that fully meets the design and implementation of different challenges of robot technologies is a novel approach to resolve many of the open issues and drastically enhance the development of applications on such systems.

Some research efforts have been done on surveying different aspect of robotics. [1] surveyed space robotics. [2][3] focused more on robot programming environments characteristics and evaluations. [4] focused on surveying vision for mobile robot navigation, while [5] presented different robotic mapping techniques. In addition, some research efforts were conducted on surveying different middleware approaches for emerging technologies such as ad hoc networks [6] and wireless sensor networks [7]. However, none of the existing work investigated the current state of research on the design and development of middleware for robotics. In this paper, we explore different important middleware projects for robotics, and provide a discussion of these approaches.

The remainder of the paper is structured as follows. Section II provides a short overview of robotic applications and outlines the most relevant challenges that face a middleware design for robotics. In Section III, we describe several research projects in middleware for robotics. Section IV, is a discussion of the different approaches used and some open research issues. Section V concludes the paper.

II. MIDDLEWARE CHALLENGES FOR ROBOTICS

While the older generations of robots were designed to achieve specific tasks and manufactured as one unit, the new generations of robots are usually ubiquitous and autonomous. This is achieved by using modular design and implementation. New robots are composed of heterogeneous interconnected hardware components. These components are usually controlled by software modules developed by different manufactures using different programming languages. These components may also use different communication mechanisms. Software modules are also needed to process sensor information and control actuators for performing computational, vision and cognitive tasks like planning, navigation, and user interaction.

Although modular design has many advantages in engineering, it raises some integration issues such as communication, interoperability, and configuration. These issues could be solved by using a middle layer, middleware. In general, middleware systems are used in distributed systems to reduce development time and cost. This is achieved by providing well-structured and well-tested services for often-needed functionalities. In addition, it provides some value added functions that can not be added to the operating systems such as

reliability, security, and abstraction. However, the design and development of a successful middleware for robots is not trivial. It needs to deal with many challenges dictated by robots characteristics on one hand and the applications requirements on the other hand. These challenges are the following:

- **Simplifying the development process:** application development is not easy for the robotic environment. Middleware should simplify the development process by providing higher-level abstractions with simplified interfaces that can be used by developers and the middleware should also enhance software integration and reuse.

- **Support communications and interoperability:** robotic modules can be designed and implemented by different manufactures. Efficient communication and simple interoperability mechanisms are needed among these modules. Therefore, robotic middleware should provide these functions.

- **Providing efficient utilization of available resources:** robots usually need to execute processing- and communication-intensive tasks in real-time. Examples are vision processing, mapping, and navigation. Therefore, efficient utilization of robot components and resources is needed. The robots may have single or multiple microprocessors, one or more interconnection networks and several other resources. Middleware should help in efficiently utilizing these resources for different application requirements.

- **Providing heterogeneity abstractions:** any robotic system contains many heterogeneous hardware and software components. Communication and cooperation among these components is an important aspect. Commonly the abstraction of this role is played by a middleware which acts as a collaboration software layer among all involved modules, hiding the complexity of the low-level communication and the heterogeneity of the modules.

- **Supporting integration with other systems:** New types of robots such as ubiquitous robots need to interact with other systems such as other robots, wireless sensor networks, and high-end servers. Most of these interactions should be done in an abstract way and in real-time. Hence, middleware should provide real-time interaction services with other systems.

- **Offering often-needed robot services:** A great deal of effort is spent writing new implementations of existing algorithms and control services for robots multiple times. The same algorithm/service may be rewritten several times due to changes in the robot's hardware, the development of new applications, changes in the operating systems, changes of technical staff, or just for adding new functionalities. These often-needed robot services should be provided by robotic middleware which allows for reuse of the modules offering these functionalities.

- **Providing automatic recourse discovery and configuration:** robots are considered dynamic systems due to their modularity and mobility. For example external devices can be dynamically available/unavailable for a robot's use. Hence, automatic and dynamic resource discovery and configuration are needed. In addition, it should support mechanisms for the robots to be self-adapting, self-configuring, and self-optimizing.

- **Supporting embedded components and low-resource-devices:** robots in many situations use or interact with embedded devices that may have several limitations such as limited power, small memory, limited operating system functionalities and limited connectivity. Handling such resources is usually different from handling other regular resources; therefore, the middleware should provide special functionalities to manage the resources as needed.

III. Different Middleware for Robotics

As mentioned above, several researchers and research groups are working on middleware solutions for robotics. Some design principles and research projects have already been proposed and implemented, while others offer conceptual models and frameworks for the proposed middleware. In this section, which is the focus of our paper, different approaches and projects will be presented. However, due to the size restrictions, we will limit the detailed descriptions to some projects that we view as more representative of the issues in discussion.

A. Miro

Miro [8][9] is an object-oriented middleware for robots developed by University of Ulm, Germany. The main motivation of using object-oriented middleware is to improve the software development process for mobile robots and to enable the interaction between the robots and enterprise information systems. Miro is designed and implemented by applying object-oriented design and implementation approaches using the common object request broker architecture (CORBA) standard. This allows inter-process and cross-platform interoperability for distributed robot control. Miro is constructed using three layers: the device, the service, and the class framework. The device layer provides object-oriented interface abstraction for all sensor and actuator devices. This layer is platform-dependant. The service layer provides abstractions for devices via CORBA interface definition language (IDL). The class framework provides a number of often-needed services such as mapping, self localization, behavior generation, path planning, logging, and visualization facilities. The layered architecture and object-oriented approach make Miro very flexible and expandable to support new devices and new services for new robot applications [10]. Miro was implemented using multiplatform libraries for easy portability. Examples of these libraries are the CORBA-based adaptive communication environment (ACE) [11], used for providing object-oriented abstraction layers for many operating systems and communication primitives and the CORBA Notification Services [12] used for providing the event-based communication functionality.

B. Orca

Orca [13] is a middleware framework for developing component-based robotics. It is designed to target applications from single vehicles to distributed sensor networks. The main goal of Orca is to enable software reuse in robotics. Orca enables implementing a distributed component-based robotic system by allowing the user to define interfaces and communication mechanisms. Orca was implemented using CORBA. However, due to the complexity problems faced with CORBA, Ice [14] was used. Ice is a new approach to object-oriented middleware that offers a much smaller and more consistent API, lighter implementations, advanced services, and good performance. Orca is an open-source product.

C. UPnP Robot Middleware:

UPnP middleware [15][16] was developed by Korea Institute of Science and Technology to utilize the Universal Plug and Play (UPnP) architecture for dynamic robot internal and external software integrations and for ubiquitous robot control. UPnP was developed to offer peer-to-peer network connectivity among PCs, wireless pervasive devices, and intelligent appliances [17]. The

UPnP has automatic discovery and configuration mechanisms. These mechanisms are utilized to configure robot components and to allow ubiquitous robots to discover and interact with other devices around them such as cameras, sensor networks, and electromechanical devices. The new trend for implementing new robots is to assemble a number of robot components that form the final robotic system. These components are usually created by different manufactures using various hardware and software technologies. A modular robot can be manufactured by connecting multiple components through an internal network. If each component supports UPnP, then the process of linking and configuring multiple robotic components together is simplified. This approach provides a simple scheme for building intelligent robots with a lot of hardware and software components. It solves some implementation issues currently facing the robotic field.

The automatic discovery and configuration mechanisms are also appropriate for dynamic computing environments such as ubiquitous robots. Mobile robots can discover the existence of external devices and can configure themselves to interact with them. These devices can be cameras, sensor networks, and controllable electromechanical devices. Using UPnP mechanics robots are able to configure their internal components to interact with other external devices based on the specific goals or services they should provide. This is an essential feature for the implementation of intelligence in robotics. The intelligence component can be internal or external since software components can cooperate with each other regardless of their location.

D. RT–Middleware

RT (Robot-Technology) - Middleware [18] was developed in collaboration between The Japanese Ministry of Economy, Trade and Industry (METI), The Japan Robot Association (JARA), and National Institute of Advanced Industrial Science Technology (AIST). The RT-Middleware is an infrastructure software based on CORBA implemented using a number of specifications at the distributed middleware interface level and a prototype implementation named OpenRTM-aist was produced. The main goals of this middleware are to build robots and their functional parts in a modular structure at the software level and to simplify the process of building robots by simply combining selected modules. These goals are to allow system designers or integrators to build customized robots for a variety of applications in cost effective and efficient manners. The components used to construct robotic systems are called RT-Components. There are some efforts to standardize the architecture of RT-Components in the Object Management Group (OMG) [19]. These efforts will enable fast integration among robotic components implemented by different manufacturers.

Another important goal is to make robots more intelligent by distributing their necessary resources over a network. RT-Middleware provides the necessary services to enable implementing robotic applications that need these types of distributed systems. One example of these applications is a network distributed monitoring system for the human assistance robot system [20]. This application was developed to improve the interaction among the users and local robotic systems. In addition, it enables a remote user to better monitor the local human and the environment. Another application is the development of home integration systems [21]. In this project, multiple home devices and appliances can interact with the robot system.

E. ASEBA

ASEBA [22] is an event-based middleware that supports distributed control and efficient resource utilization of multiprocessor robots. This middleware is designed for robots with several processors that communicate through a shard bus. Some robots, in addition to the main processor, have several microcontrollers that are part of or close to the sensors and actuators to control them. Microcontrollers can communicate among themselves by asynchronous messages called events. Messages are only transmitted when relevant events occur. For example when a specific observation was noticed by a sensor, an event about that observation will be sent by the corresponding microcontroller to another microcontroller or to the main processor. This reduces the load on the bus as less data is transmitted compared to regular robot systems in which periodic sensor readings and actuator commands are generated from the main processor. ASEBA improves the modularity and efficiency of robots by distributing some of the processing tasks to all the microcontrollers and communicating only the relevant data to the main processor. It allows dedicating the main processor for CPU-intensive tasks such as vision processes and higher-level controls. Lightweight virtual machines are developed to run on microcontrollers. These virtual machines support the implementation of the policies for sending events. Policies are described by a simple scripting language called AESL (ASEBA Event Scripting Language). In addition, ASEBA provides an integrated development environment with an editor, compiler and debugger for implementing distributed controls for robots.

F. Player/Stage System

Player/Stage system [23] is a middleware platform that provides infrastructure, drivers and some algorithms for mobile robotic applications. This middleware has two major components: Player and Stage. Player is a device repository server for actuators, sensors, and robots. Each device in Player is composed of a driver and an interface. Interfaces are the part used by the client to write new applications that get information from a sensor or control an actuator. Drivers can also implement algorithms that receive data from other devices, process it, and then send it back. Stage is a graphical simulator that models devices in a user defined environment. Driver can also generate arbitrary data when needed. The Player/Stage system is implemented as a three-tier architecture in which the clients are software developed for specific robot applications, the middle tier is Player which provides common interfaces for different robot devices and services, and the third tier is the actual robots, sensors, and actuators. Various client side libraries exist in the form of proxy objects for different programming languages to access the services provided by the Player platform. Clients can connect to the Player platform to access data, send commands, or request configuration changes to an existing device in the repository. Examples of client programming languages supported are C, C++, Java, and Python.

Player serves as an interface to many different types of robotic devices and provides drivers for many hardware modules. Some of the main features of this middleware are the platform-, programming language-, and transport protocol-independence; open source; and modularity. Player's modular architecture makes it flexible to support new hardware. Players can run on both regular and embedded Linux, Solaris, and BSD Unix.

Player/Stage system was started at the University of Southern California in the late nineties and moved to Source Forge in 2001.

G. The PEIS Kernel

The PEIS Kernel [24] is based on a collaborative research project between the Electronics and Telecommunications Research Institute (ETRI), Korea, and The Centre for Applied Autonomous Sensor Systems, Sweden. This middleware is designed toward the concept of Ecology of Physically Embedded Intelligent Systems, or PEIS-Ecology, in which many robotic devices, pervasively embedded in everyday environments such as our homes or offices, cooperate in performing some tasks in the service of people. In this approach, complex robotic functionalities are not achieved via the implementation of extremely advanced robots, but rather through the cooperation of many simple robotic components. The main aim of this middleware is to provide a common communication and cooperation model that can be shared among robotic devices such as mobile robots, static sensors or actuators, and automated home appliances. With this middleware, any robotic device with software control in the environment is defined as PEIS. Each PEIS is a set of inter-connected software components developed to control sensors or actuators. All PEIS devices are connected by a uniform communication model, which allows the exchange of information among PEIS devices and allows dynamic joining and leaving of PEISs. All PEIS devices can cooperate using a uniform cooperation model. In this model, each participating PEIS can use functionalities from other PEISs in the ecology in order to complement its own. For example, in a home environment, an autonomous vacuum cleaner (PEIS device) can use a localization function provided by an overhead tracking system (PEIS device) to know its position. The PEIS Kernel provides a shared memory model, a simple dynamic model for self-configuration and introspection, and supports heterogeneous devices. Both tiny embedded devices and complex large robots are supported [25].

H. ORiN

ORiN (Open Robot Interface for the Network) [26] is an interface developed to provide standard methods for accessing and controlling robotic systems from windows-based PCs. ORiN is based on HTTP and other web technologies such as XML and SOAP. It was developed to target industrial robots. This interface provides separation between the specifications and the implementations and enables third parties to develop robotic applications that are controlled by PCs. Therefore low cost mutli-vendor systems can be easily developed. The interface is based on the distributed object model which provides network transparency and language independence. In this system, various types of robotic specifications are allowed and vendors can define unique options using XML.

I. MARIE

MARIE (Mobile and Autonomous Robotics Integration Environment) [27] is a middleware framework created for developing and integrating new and existing software for robotic systems. MARIE aims to create a flexible distributed components system that allows robotics developers to share, reuse, and integrate robotic software programs for rapid robotic application development. MARIE is implemented in three layers: Core, Component, and Application. The Core layer consists of services for communication, low-level operating functions, and distributed computing functions. The Component layer is used to add components for often-used services and to support domain-specific concepts. The Application layer contains useful services and tools to build and manage integrated applications. MARIE middleware provides some services that allow the adaptation of different communication protocols and applications which make it very flexible. The integration aspect of MARIE uses the Adaptive Communication Environment (ACE) communication framework [11]. A variety of software components can be connected in MARIE using a centralized component. In addition, there are four functional components: application adapters, communication adapters, communication managers, and application managers. Application adapters act as proxies between the central component and the applications. The data exchanged among application adapters is translated by communication adapters, while communication managers create and manage the connections. Finally, application managers instantiate and manage components locally or across distributed processing nodes. MARIE follows the mediator design pattern in which it provides mediator interoperability layers among adapters and managers. The key features of MARIE are the interoperability and reusability of robotic application components.

J. RSCA

The RSCA (Robot Software Communication Architecture) [28] is a QoS (Quality of Service) -Aware middleware for networked service robots developed by Seoul National University. The key strength of RSCA is the real-time support. RSCA provides a standard operating environment and development framework for robot applications. The operating environment consists of a Real-Time Operating System, communication middleware, and deployment middleware. The operating system is compliant with the PSE52 in IEEE POSIX.13. It provides an abstraction layer that makes robotic applications both portable and reusable on different hardware. The communication middleware is compliant to minimum CORBA and RT-CORBA v.1.1 [29] and provides mechanisms for distributed heterogeneous components to communicate in real-time. The deployment middleware provides frameworks for program execution in distributed environments, dynamic program deployment, real-time support, QoS, and a management capability for limited resources and heterogeneous hardware.

K. The Middleware of AWARE

This platform [30] is a data-centric middleware for the integration of wireless senor networks and mobile robots developed by the University of Seville, Spain and the University of Stuttgart, Germany. The main aim of this middleware is to provide simplified mechanisms for integrating information gathered by various types of sensors including wireless sensor networks (WSN) and mobile robots. This type of integration is needed for applications where robots are used to obtain and process data from its environment through a WSN. This data can be temperature, light level, or humidity for example. Another application is to allow a robot to locate itself in an environment where GPD (Geographic Positioning Device) service is not available. This middleware provides data-centric capabilities in which users can access data in an abstract way. Any user of the network can make references to objects that exist in the environment, such as a fire, a car, or an animal. The user has to provide the conditions that define the targeted object. These

conditions can be, for example, high temperature for a fire object. Then, the user can address any specific object in the environment in order to obtain data from it. In this platform, the middleware components are executed on all sensor and robot nodes. Sensor nodes use TinyOS operating system designed for devices with limited resources.

L.   Sensory Data Processing Middleware

This middleware [31] is developed at The University of Tsukuba in Japan to provide abstracted services for accessing sensor information to support service mobile robots. Two types of services were implemented to provide obstacle information and to localize the robot position using landmark observations from multiple external sensors. This middleware provides a unified model for different configurations of external sensors on a service mobile robot. The unified model abstracted from sensors can be used in any service mobile robot application independent of the sensors configuration. The developed services can be reused in multiple applications without dealing with individual sensors.

M.   Distributed Humanoid Robots Middleware

This communication middleware [32] is developed by Honda Research Institute with other organizations to facilitate communication among the modules of distributed humanoid robots. In humanoid robots, a number of modules are needed such as sensors, actuators, planning modules, decision making modules, movement controllers, etc. The performance quality of humanoid robots is completely dependant on the collaboration and communication performance among these modules. The developed communication middleware serves various functional roles using three different communication subsystems: the Cognitive Map (CogMap), Distributed Operation via Discrete Events (DiODE), and Multimodal Communication (MC). The CogMap allows for sharing and transforming information streams dynamically among modules. DiODE provides direct connection between two modules for direct and fast communication. Finally, MC provides service to stream raw sensory data to other modules.

N.   A Layer for Incorporations among Ubiquitous robots

This layer [33] is developed by Korea Advanced Institute of Science and Technology to enable communication among ubiquitous robots which are usually of different types. These types can be software robots, mobile robots, and embedded robots. Software robots are similar to mobile agents while mobile robots are usually hardware robots controlled by software. This middle layer is mainly designed to allow software robots and mobile robots to communicate even when they use different communication mechanisms. The middle layer consists of two mappers: sensor mapper and behavior mapper. The senor mapper helps software robots get physical sensor information from mobile robots; while the behavior mapper helps software robots make physical behavior using the actuators of the mobile robots.

O.   WURDE

The WURDE (Washington University Robotics Development Environment) Middleware [34] provides a set of utilities to simplify interfacing with robotic components and software development. The main goal of WURDE is to allow rapid development of robotic applications by having clean levels of abstraction. WURDE enables modular robotic applications to be implemented in which robot software can be written as a number of small, interconnected components. WURDE utilizes a message passing protocol as its distributed computation mechanism. In addition, WURDE uses asynchronous communication and wraps the communication mechanism to simplify the process of supporting new communication protocols. However, WURDE dose not provide any security for controlling access to the modules. Another module built with WURDE is the RIDE interface that provides multi-robot tasking and control mechanisms. It has interactive display environments.

IV. Discussion and Open Issues

In the previous section we surveyed different existing middleware approaches for robotics. The general observation is that all the projects target some form of enhancement to the robotics systems both at the development and the utilization levels. In addition, it is also clear that we cannot provide a clear set of distinct classification criteria that distinguish between the different projects and provide a solid basis for comparisons. However, we define here a set of main objectives each of which match a few of the projects listed above. These objectives are:

1. Enhancing the development process by providing some form of modular design mechanisms, high level abstractions, and component-based development. Many projects have this goal in sight, for example, Micro, Ocra, RT-Middleware and WURDE.
2. Reusability of existing components where developers and robot designers are provided with ready made components and software modules that can be put together to create new robot devices and applications such as Ocra, UPnP Robot Middleware and MARIE.
3. Better utilization of resources and real-time support, where robots are equipped with the capabilities to optimize resource utilization and support real—time functions such as UPnP Robot Middleware, RT-Middleware, ASEBA, RSCA and Distributed Humanoid Robots Middleware.
4. Integration with external systems where robots become capable of communicating and utilizing external resources like sensors, devices controllers and GPS systems. Some examples are: PEIS Kernel, ORIN, the Middleware of AWARE, Sensory Data Processing Middleware and Middleware Layer for cooperation among Ubiquitous Robots.
5. Flexible enhancements and expansion of functionalities, where it is easy to enhance functionalities and incorporate new ones in existing robot systems such as Micro, Player/Stage and MARIE.

Although we listed examples for each objective, many projects cover several of these objectives with varying degrees. Consequently, the examples given are mostly based on the main objective of each project. As for the technologies or standards used in these projects we observe that many tried to follow well defines and common technologies like CORBA, ICE, XML, and virtual machines. Table 1 includes a list of all the projects surveyed in the paper with a brief list of their objectives and technologies/standards used.

As mentioned earlier, it is obvious that there is no clear definition or common understanding of what middleware for robotics should or should not provide, and how to provide them. It is apparent that different researchers and practitioners view the issues in different ways. As a result, the target of reaching a single middleware infrastructure that will solve all the problems of robotics systems is not realistic. There are many issues, technical

limitations and difficulties that need to be addressed to achieve a good middleware-based solution. Some of these issues are:

1. Current middleware systems provide very limited often-needed advanced services that can be used to simplify the development process and to enhance resource utilization. In addition, many of the available services are not standardized, which make it difficult to achieve interoperability between different robotic systems.
2. The availability of self-adaptation and self-configuration mechanisms is very limited. Since the target is to develop autonomous and ubiquitous robots, these mechanisms are very important to enhance the performance of the robotic applications.
3. The security mechanisms within the middleware solutions for robotics are inadequately investigated. As the use of multiple robots and distributed robotic components increases, the need to secure their communication and collaboration becomes essential for their operations. However, researchers seem to steer away from this issue.
4. There is very limited work towards providing high level abstractions for coordination and collaboration among multiple robotic applications. Therefore, application developers working on collaborative robotic systems must start from lower, more primitive levels.
5. There is very limited work towards providing automatic mechanisms for efficient utilization of the availability and the heterogeneity of multiple robots working on the same task. In some cases, tasks need to be distributed among the robots to be completed in parallel rather than being done by individual robots. While in other cases, the existence of multiple robots that have heterogeneous resources provides a great opportunity to redirect tasks to the robot with the most suitable resources.

As the author of [35] tries to answer the question: "Is a Common Middleware for Robotics Possible?" He lists the issues and difficulties of robotic systems such as high levels of heterogeneity, limited resources, and high probabilities of failures that make the development of a common middleware a very complicated task. Here we also identified some lacking features and open issues that current middleware approaches did not address. However, we view the difficulties and the open issues as the motivations for working harder to design flexible and efficient middleware solutions for robotic systems. This may be one common middleware or several middleware components which address different issues. In the end, the goal is to reach a useful solution, which we view as a hard, but achievable task.

V. Conclusion

In this paper, we surveyed several projects for middleware in robotics and discussed some of the main issues that face the design and the development of such middleware solutions. Many projects have different objectives such as simplifying the development process, reusability, integration and flexibility. In addition different projects used different technologies like CORBA, ACE, Virtual machines, XML, and message passing to achieve their objectives. Furthermore, we examined the current projects to determine what were the limitations and open issues that were not addressed well. As a result we identified several open issues that need to be addressed to be able to design a comprehensive middleware solution for robotics systems. We arrived at a general conclusion that it is very hard to have one middleware platform that will solve all the problems and address all the issues because that will basically result in a very complex system. In addition, many robotics systems do not need all the functionalities and features together. Therefore, we envision a modular or component-based middleware that provides customizable solutions based on the integration of the needed components to design and develop the required robotic system.

TABLE 1. A SUMMARY OF THE OBJECTIVES AND TECHNOLOGIES OF THE DISCUSSED PROJECTS.

Middleware	Main Objectives	Standards and Technologies Used
Miro	To improve the software development process for mobile robots and enable interaction between robots and enterprise systems using the distributed object paradigm.	CORBA
Orca	To enable software reuse in robotics using component-based development.	Ice
UPnP Robot Middleware	To enable automatic discovery, configuration, and integration for robot components in both modular and ubiquitous robots.	UPnP
RT–Middleware	To make robots and their functional parts in a modular structure at the software level and to simplify the process of building robots by simply combining selected modules.	CORBA
ASEBA	To allow distributed control and efficient resources utilization of robots with multiprocessors.	Event-based middleware, Virtual machines
Player / Stage System	To provides a development platform that supports different robotic hardware and provides common services needed by different robotic applications.	Three-Tier Architecture Proxy Objects
The PEIS Kernel	To provide a common communication and cooperation model that can be shared among multiple robotic devices.	A shared memory model
ORiN	To provide an interface for accessing and controlling robotic systems from PCs.	HTTP, XML , SOAP
MARIE	To create flexible distributed components that allows developers to share, reuse, and integrate new or existing software programs for rapid robotic application development.	Mediator Interoperability Technology, ACE
RSCA	To provide real-time support for robotic applications and to provide abstractions that makes robotic applications both portable and reusable on different hardware platforms.	PSE52 in IEEE POSIX. 13, CORBA, RT-CORBA v1.1
The Middleware of AWARE	To provide data-centric capabilities for the integration of wireless senor networks and mobile robots.	TinyOS, TinySchema, Publish/subscribe
Sensory Data Processing Middleware	To provide abstracted services for accessing external sensor networks information to support service mobile robots.	N/A
Distributed Humanoid Robots Middleware	To facilitate communication among the modules of distributed humanoid robots.	Publish/subscribe, TCP & shared memory. Stream communications
Layer for Incorporation	To enable communication among ubiquitous robots which are usually of different types.	Sensor and behavior mappings
WURDE	To allow rapid development of robotic applications by having clean levels of abstraction and modular development.	Communication wrapping, Massage passing,

REFERENCES

[1] L. Pedersen, D. Kortenkamp, D. Wettergreen and I. Nourbakshk, " A Survey of Space Robotics," in The 7th International Symposium on Artificial Intelligence, Robotics and Automation in space (i-SAIRAS-03), 2003.

[2] G. Biggs and B. MacDonald, "A Survey of Robot Programming Systems," In Proc. of Australasian Conference on Robotics and Automation (CSIRO), Dec. 2003.

[3] J. Kramer, M. Scheutz, "Development Environments for Autonomous Mobile Robots: A Survey," Autonomous Robots, Volume 22, No. 2, pp. 101-132, Feb. 2007.

[4] G. N. DeSouza and A. C. Kak, "Vision for Mobile Robot Navigation: A Survey," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 24, No. 2, pp. 237-267, Feb. 2002.

[5] S. Thrun "Robotic Mapping: A Survey," Technical Report CMU-CS-02-111, School of Computer Science, Carnegie Mellon University, Feb. 2002.

[6] S. Hadim, J. Al-Jaroodi, N. Mohamed, "Trends in Middleware for Mobile Ad Hoc Networks," The Journal of Communications, Vol 1, No. 4, pp. 11-21, July 2006.

[7] S. Hadim and N. Mohamed, "Middleware Challenges and Approaches for Wireless Sensor Networks," in IEEE Distributed Systems Online, Vol. 7, No. 3, art. no. 0603-o3001, March 2006.

[8] S. Enderle, H. Utz, S. Sablatng, S. Simon, G. Kraetzschmar, and G. Palm, "Miro: Middleware for autonomous mobile robots," IFAC Conference on Telematics Applications in Automation and Robotics, 2001.

[9] H. Utz, S. Sablatng, S. Enderle, G. Kraetzschmar, "Miro – Middleware for Mobile Robot Applications," IEEE Transactions on Robotics and Automation, 18(4):493-497, Aug. 2002.

[10] D. Krüger, I. Lil, N. Sünderhauf, R. Baumgartl, P. Protzel, "Using and Extending the Miro Middleware for Autonomous Robots," Towards Autonomous Robotic Systems (TAROS), Guildford, September 2006.

[11] D. C. Schmidt, "ACE: An Object-Oriented Framework for Developing Distributed Applications," Proceedings of the 6th USENIX C++ Technical Conference, April 1994.

[12] T. H. Harrison, D. L. Levine, and D. C. Schmidt, "The Design and Performance of a Real-Time CORBA Event Service," in Proc. OOPSLA'97, pp. 184-200, Oct. 1997.

[13] A. Makarenko, A. Brooks, and T. Kaupp, "Orca: Components for Robotics," In International Conference on Intelligent Robots and Systems (IROS), pp. 163-168, Oct. 2006.

[14] M. Henning, "A New Approach to Object-Oriented Middleware," IEEE Internet Computing, pp. 66-75, Jan.-Feb. 2004.

[15] S. Ahn, J. Lee, K. Lim, H. Ko, Y. Kwon, and H. Kim, "Requirements to UPnP for Robot Middleware," in Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Oct. 2006.

[16] S. Ahn, K. Lim, J. Lee, H. Ko, Y. Kwon and H. Kim, "UPnP Robot Middleware for Ubiquitous Robot Control," The 3rd International Conference on Ubiquitous Robots and Ambient Intelligence (URAI 2006), Oct. 2006.

[17] UPnP, www.upnp.org

[18] N. Ando, T. Suehiro, K. Kitagaki, T. Kotoku, W. Yoon, "RT-Middleware: Distributed Component Middleware for RT (Robot Technology), IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3555-3560, Aug. 2006.

[19] N. Ando, T. Suehiro, K. Kitagaki, T. Kotoku, "RT(Robot Technology)-Component and its Standardization - Towards Component Based Networked Robot Systems Development -", SICE-ICASE International Joint Conference 2006 (SICE-ICCAS 2006), pp.2633-2638, Oct. 2006.

[20] S. Jia and K. Takase, "Network Distributed Monitoring System Based on Robot Technology Middleware," International Journal of Advanced Robotic Systems, Vol. 4, No. 1, pp. 69-72, 2007.

[21] Y. Hada, S. Jia, K. Takase, H. Gakuhari, and T. Ohnishi, "Deveopment of Home Robot Integration System Based on Robot Technology Middleware," The 36th International Symposium on Robotics (IRS), Japan , 2005.

[22] S. Magnenat, V. Longchamp, F. Mondada, "ASEBA, an event-based middleware for distributed robot control" Workshops DVD of International Conference on Intelligent Robots and Systems (IROS), Oct.-Nov. 2007.

[23] M. Kranz, R. Rusu, A. Maldonado, M. Beetz, A. Schmidth, "A Player/Stage System for Context-Aware Intelligent Environments," in Proc. of the System Support for Ubiquitous Computing Workshop (UbiSys), Sep. 2006.

[24] M. Broxvall, B.S. Seo, W.Y. Kwon, "The PEIS Kernel: A Middleware for Ubiquitous Robotics," in Proc. of the IROS-07 Workshop on Ubiquitous Robotic Space Design and Applications, Oct. 2007.

[25] M. Bordignon, J. Rashid, M. Broxvall, A. Saffiotti, "Seamless Integration of Robots and Tiny Embedded Devices in a PEIS-Ecology," in Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Oct. 2007.

[26] M. Mizukawa, H. Mastsuka, T. Koyama, T. Inukai, A. Nodad, H. Tezuka, Y. Noguch, and N. Otera, "ORiN: Open robot interface for the network," In SICE, pp. 925-928, 2002.

[27] C. Côté, Y. Brosseau; D. Létourneau; C. Raïevsky, F. Michaud, "Robotic Software Integration Using MARIE," International Journal of Advanced Robotic Systems, Vol. 3, No. 1, pp. 55-60, March 2006.

[28] J. Yoo, S. Kim, and S. Hong, "The Robot Software Communications Architecture (RSCA) QoS-Aware Middleware for Networked Service Robots," in Proc. International Join Conference SICE-ICASE, pp. 330-335, Oct. 2006.

[29] D. C. Schmidth, A. Gokhale, T. Harrison, and Parulkar, "A Higher-Performance Endsystem Architecture for Realtime CORBA," IEEE Communication Mag., Vol. 14, Feb 1997.

[30] P. Gil, I. Maza, A. Ollero, P. Marrón, "Data Centric Middleware for the Integration of Wireless Sensor Networks and Mobile Robots," in Proc. 7th Conference on Mobile Robots and Competitions, ROBOTICA 2007. April 2007.

[31] E. Takcuchi and T. Tsubouchi, "Sensory Data Processing Middlewares for Service Mobile Robot Applications", in Proc. International Join Conference SICE-ICASE, Oct. 2006.

[32] V. Ng-Thow-Hing, T. List, K.R. Thórisson, J. Lim, J. Wormer, "Design and Evaluation of Communication Middleware in a Distributed Humanoid Robot Architecture," IROS '07 Workshop Measures and Procedures for the Evaluation of Robot Architectures and Middleware, 29 Oct. - 2 Nov. 2007.

[33] T. Kim, S. Choi, and J. Lim, "Incorporation of a Software Robot and a Mobile Robot Using a Middle Layer," IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, Vol. 37, No. 6, pp. 1342-1348, November 2007.

[34] F. Heckel, T. Blakely, M. Dixon, C. Wilson, and W. D. Smart, "The WURDE Robotics Middleware and RIDE Multi-Robot Tele-Operation Interface," AAAI Mobile Robotics Workshop, 2006.

[35] W. Smart, "Is a Common Middleware for Robotics Possible?," In Proceedings of the IROS 2007 workshop on Measures and Procedures for the Evaluation of Robot Architectures and Middleware, Oct. 2007.