Hindawi Publishing Corporation
Journal of Robotics
Volume 2012, Article ID 959013, 15 pages
doi:10.1155/2012/959013

Review Article

Robotics Middleware: A Comprehensive Literature Survey and Attribute-Based Bibliography

Ayssam Elkady and Tarek Sobh

School of Engineering, University of Bridgeport, Bridgeport, CT 06604, USA

Correspondence should be addressed to Ayssam Elkady, ayssam.elkady@gmail.com

Received 21 August 2011; Revised 15 January 2012; Accepted 29 January 2012

Academic Editor: Yangmin Li

Autonomous robots are complex systems that require the interaction between numerous heterogeneous components (software and hardware). Because of the increase in complexity of robotic applications and the diverse range of hardware, robotic middleware is designed to manage the complexity and heterogeneity of the hardware and applications, promote the integration of new technologies, simplify software design, hide the complexity of low-level communication and the sensor heterogeneity of the sensors, improve software quality, reuse robotic software infrastructure across multiple research efforts, and to reduce production costs. This paper presents a literature survey and attribute-based bibliography of the current state of the art in robotic middleware design. The main aim of the survey is to assist robotic middleware researchers in evaluating the strengths and weaknesses of current approaches and their appropriateness for their applications. Furthermore, we provide a comprehensive set of appropriate bibliographic references that are classified based on middleware attributes.

1. Introduction

Robot middleware is an abstraction layer that resides between the operating system and software applications (as shown in Figure 1). It is designed to manage the heterogeneity of the hardware, improve software application quality, simplify software design, and reduce development costs. A developer needs only to build the logic or algorithm as a component, after which the component can be combined and integrated with other existing components. Furthermore, if he wants to modify and improve his component, he needs only to replace the old one with the new one. Therefore, experiment efficiency will improve. In [1], the authors outline some of the problems faced in the development of some robotics middleware. A survey of robot development environments (RDEs) by Kramer and Scheutz [2] described nine open source, freely available RDEs for mobile robots, evaluated and compared them from various points of view with suggestions of how to use the results of the survey, and concluded with a brief discussion of future trends in RDE design. Mohamed et al. [3] provide a short overview of several research projects in middleware for robotics and their main objective. Mohamed et al. [4] provide an overview study of networked robot middleware and different criteria for evaluating networked robot middleware. Furthermore, in [5], some freely available middleware frameworks for robotics are addressed, including their technologies within the field of multirobot systems.

This paper is structured as follows. Section 2 presents an overview and the objectives of current middleware solutions. Some attributes, focusing on the architecture, simulation environment, standards and technologies, support for a distributed environment, security for accessing modules, fault detection and recovery, real-time and behavior coordination capabilities, and open-source and dynamic wiring for the most of the existing robotic middleware frameworks, are then discussed in the following sections. Each section describing an attribute for different middleware structures includes an embedded set of the appropriate bibliographic references to provide researchers with easy access to the current state of the art research in the area. The final section summarizes the survey findings.

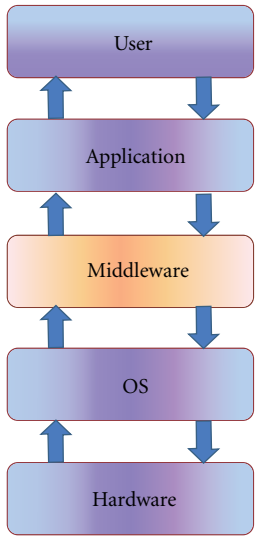

Figure 1: Middleware layers.

The primary advantages of the robotics middleware are *Software Modularity*, *Hardware Architecture Abstraction* to hide the low-level device-specific details of the device in order to give the developers more convenient, standardized hardware APIs, *Platform Independence,* and *Portability* to be able to run on any platform with only changing in the system's configuration. Furthermore, the core of the middleware should not depend on the specific device or software algorithm. The middleware should be scalable and upgradable with the growing of its components. The middleware should be easy-to-use, robust, reliable, easy to maintain, efficient flexible, and support for parallelism and distribution systems. It should also allow for changing in the configuration of control flow and the data flow at runtime. It is favorable to be real-time system and provides some security aspects such as authentication, authorization, and secure communication.

2. Robot Software

2.1. Middleware. Bakken et al. [6] defined middleware as follows: "a class of software technologies designed to help manage the complexity and heterogeneity inherent in distributed systems. It is defined as a layer of software above the operating system but below the application program that provides a common programming abstraction across a distributed system." In this section, several robotics middleware such as Player, CLARAty, Player, Miro, and OpenRTM-aist and their attributes are described. For each attribute, a short description of the relevant research attribute is described and several representative references are provided.

The Player project started at the University of Southern California [8–13]. Player is a "distributed device repository server" [13] for robots, sensors, and actuators. A client program communicates with Player, running on the robot, using a separate TCP socket connection for data transfer. Coupled Layer Architecture for Robotic Autonomy (CLARAty) [7, 14–26] is an attempt by NASA, through collaboration with the California Institute of Technology's Jet Propulsion Laboratory, Ames Research Center, Carnegie Mellon University, and the University of Minnesota. OpenRTMaist [27–32] is a software platform developed by the National Institute of Advanced Industrial Science and Technology-Intelligent Systems Research Institute.

Miro [33, 34] is an object-oriented robotics middleware developed by the University of Ulm, Germany. Microsoft Robotics Developer Studio (MRDS) [35–37] is a Windows-based environment for robot control and simulation. Marie (Mobile and Autonomous Robotics Integration Environment) [38–41] was developed by the Mobile Robotics and Intelligent Systems Laboratory, University of Sherbrooke, Canada. Orca [42–44], and OPRoS (Open Platform for Robotic Services) [45–47], designed by the IT R&D program of the Ministry of Knowledge Economy of Korea, are component-based software frameworks. The ERSP Software Development Kit [48] provides technologies for vision, navigation, and system development. Webots [49, 50] is a commercial robot simulation package for fast prototyping and simulation of mobile robots. Robot Operating System (ROS) [51] is a "thin, message-based, peer-to-peer" [52], robotics middleware designed for mobile manipulators.

The *O*pen *Ro*bot *Co*ntrol *S*oftware (Orocos) [53–59] is a Real-Time Toolkit (RTT) that helps the developers to build C++ robotics applications. RSCA (Robot Software Communication Architecture) [60], developed in Seoul National University, is a robot middleware for networked home service robots. Skilligent Robot Behavior Learning System [61], RoboFrame [62–65], SmartSoft [66–69], iRobot AWARE [70], developed by iRobot, and ASEBA [71, 72] are some examples of robotic middleware platforms. RoboFrame [62–65] is an OO middleware developed using C++. Table 1 summarizes the objectives of some commonly used robotics middleware.

2.2. Robot Toolkit. A robot toolkit is a set of software tools used by developers that provides them with the facilities to create their own robot applications, such as CARMEN and Pyro. Carnegie Mellon Navigation (CARMEN) Toolkit [74–76] is an open-source collection of robot control software. Pyro [77–81], written in Python, stands for Python Robotics which provides "a set of abstractions that allows students to write platform-independent robot programs." [81].

3. Architectural Approach

In this section, the architecture of the previously discussed middleware platforms is described. CLARAty's architecture has two distinct layers (as shown in Figure 2): the Functional Layer and the Decision Layer. The Functional Layer includes a number of generic components such as "digital and analog I/O, motion control and coordination, locomotion, manipulation, vision, navigation, mapping, terrain evaluation, path planning, estimation, simulation, and system behavior" [7]. Furthermore, control algorithms are implemented in this

Table 1: Objectives of several robotics middleware frameworks.

Name	Objectives
Orocos	"Develops a general purpose modular framework for robot and machine control" [53]
Pyro	"Provides a programming environment for easily exploring advanced topics in artificial intelligence and robotics without having to worry about the lowlevel details of the underlying hardware" [73]
Player	Provides a development framework supporting different hardware devices and common services needed by different robotic applications and transfers a controller from simulation to real robots with as little effort as possible
Orca	"Enables software reuse in robotics using component-based development" [3]
Miro	"Improves the software development process for mobile robots and enable interaction between robots and enterprise systems using the distributed object paradigm" [3]
OpenRTMaist	Provides efficient development for robotic systems by proposing a modular software structure platform and "simplifies the process of building robots by simply combining selected modules" [3]
ASEBA	"Allows distributed control and efficient resources utilization of robots with multiprocessors" [3]
MARIE	"Creates flexible distributed components that allows developers to share, reuse, and integrate new or existing software programs for rapid robotic application development" [3] and integrates other middleware in a single robot
RSCA	"Provides real-time support for robotic applications and to provide abstractions that makes robotic applications both portable and reusable on different hardware platforms" [3]
MRDS	Provides a robotic software platform supporting a wide variety of hardware devices and a set of useful tools that facilities the programming and debugging
OPROS	"Establishes a component based standard software platform for the robot which enables complicated functions to be developed easily by using the standardized components in the heterogeneous communication network" [46]
CLARAty	A reusable robotic framework to enable integration, maturation, and demonstration of advanced robotic technologies, from multiple institutions on NASA's rover platforms in support of its technology programs (Mars and Intelligent Systems)
ROS	"Provides the operating system's services such as "hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management" [51]
SmartSoft	Implements sensorimotor systems based on communication patterns as central mean to achieve decoupling at various levels and supports model-driven software development
ERSP	"Provides cutting edge technologies for vision, navigation, and system development" [48]
Webots	"Provides a rapid prototyping environment for modeling, programming and simulating mobile robots" [50]
RoboFrame	Covers the special needs of autonomous lightweight robots such as dynamical locomotion and stability

layer, such as sensor-based manipulation, visual target tracking, and vision-based navigation. "The Decision Layer is a global engine that reasons about system resources, the state of the system and its environment, and mission constraints. It includes general planners, executives, schedulers, activity databases, and rover and planner-specific heuristics. The Decision Layer plans, schedules, and executes activity plans. It also monitors the execution modifying the sequence of activities dynamically, when necessary" [23]. The relation between the Decision and Functional Layers is a client-server model through a publish/subscribe communication scheme.

Webots [82] is organized as an OOP interface for robot control. Objects correspond to robot devices such as differential wheels, camera, distance sensor, laser range-finder, and accelerometer. As described in [64], two mechanisms are implemented in RoboFrame to exchange data between the modules: a blackboard architecture and mutual exclusion mechanisms for large data structure, and a message-based system for smaller data structures.

MIRO [33] consists of three layers (as shown in Figure 3): the MIRO device, MIRO service, and MIRO framework layers. The MIRO device layer, which is a platform dependent, provides interface abstractions for the low-level hardware details (all sensor and actuator devices). MIRO service layer provides service abstractions for sensors and actuators by means of CORBA interface definition language (IDL). "The Miro Class Framework provides a number of often-used functional modules for mobile robot control, like modules for mapping, self-localization, behavior generation, path planning, logging and visualization facilities" [33].

Marie consists of three layers [40]: Application, Component, and Core Layers. Application/Component layers provide some tools for building applications/components. The Core Layer consists of low-level tools for communication, data handling, and I/O control. Furthermore, Marie uses a centralized control unit called the Mediator Design Pattern (MDP) to interact with each application independently, as shown in Figure 4. As described in [38], Marie uses four functional components for interaction and communication between the applications through a centralized control unit (CCU) based on the MDP: Application Adapter (AA), Communication Adapters (CAs), Communication Manager (CM), and Application manager (AM). The AA is responsible for communication between the CCU and applications.

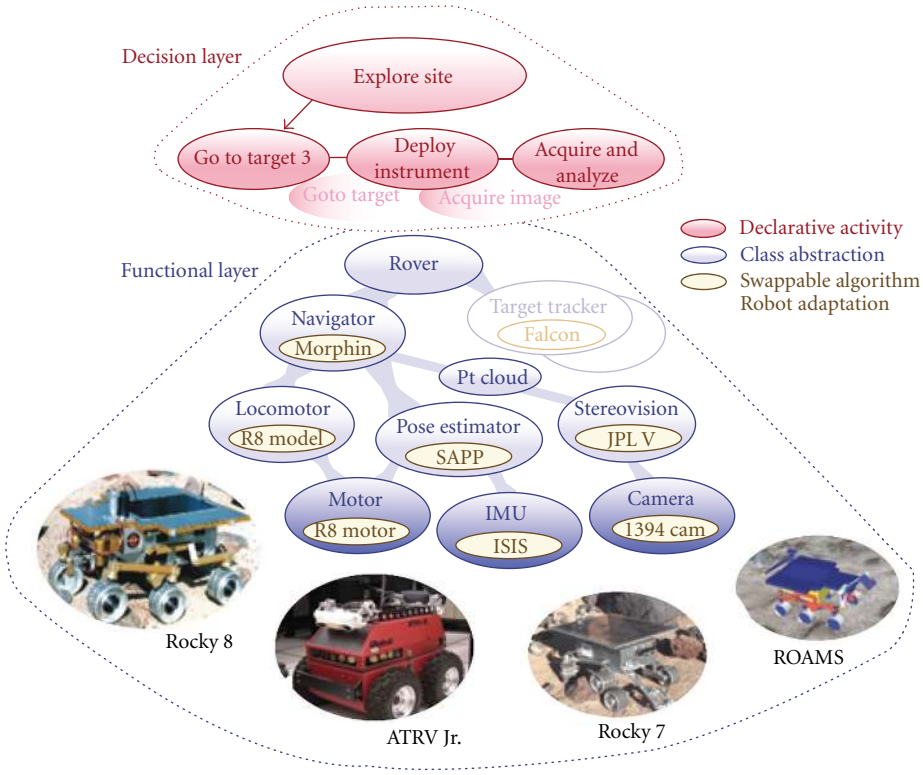

Figure 2: The CLARAty Architecture (© (2006) International Journal of Advanced Robotic Systems) (reproduced from [7] with permission of Dr. Issa Nesnas).

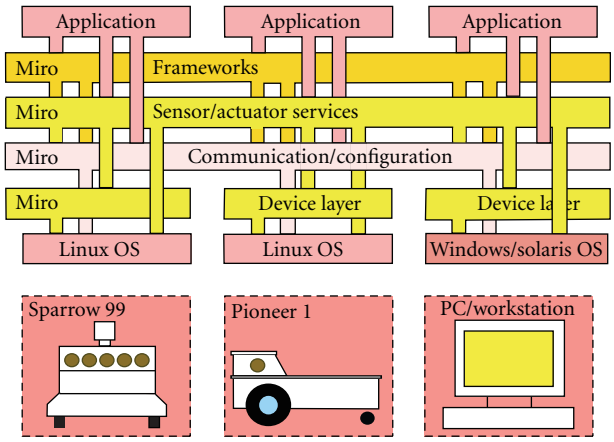

Figure 3: MIRO abstraction layers (© (2002) IEEE) (reproduced from [33]).

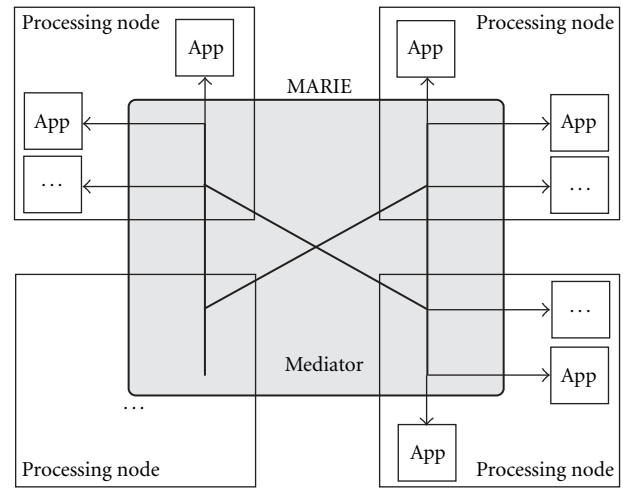

Figure 4: MARIE's adaptation of the mediator pattern for distributed system. (© (2004) IEEE) (reproduced from [38] with permission of Dr. Carle Côté and Dr. François Michaud).

For integration with the system, "each application should have its own AA in order to encapsulate communication mechanisms, services, and configurations. The CA is responsible for translating information between different communication protocols and mechanisms. It allows different applications to exchange data correctly" [40]. The common types of CA are Mailbox ("data buffer between asynchronous components" [40]), SharedMap ("It forwards incoming data to multiple outputs" [40]), Splitter ("a push-in/pull-out data structure used to store system states that can be accessed by many components" [40]), and Switch ("It sends only one of its inputs to the output port" [40]). "The CM is responsible for creating and managing communication links between Application Adapters. The AM is responsible for control and management of the whole system, by coordinating system states. It also configures and controls all components available in the system." [40].

SmartSoft consists of three layers [67], as shown in Figure 5, Skill, Sequencing, and Deliberation Layers. A module of the Skill Layer, such as the path planner and the mapper, works simultaneously as client and server. The Sequencing Layer selects and synchronizes the required behaviors, according to the actual situation, to be executed. The Deliberation Layer contains time-consuming algorithms such as symbolic action planning.

ERSP [48] consists of the three primary layers: Hardware Abstraction Layer (HAL), Behavior Execution Layer (BEL), and Task Execution Layer. The HAL hides the low-level details of the robot's hardware and operating system. The BEL contains sensing, decision-making, and autonomous action modules. "The TEL provides a high-level, goal-oriented method of programming and an interface to the BEL" [48]. Pyro consists of several modules, such as direct control, reactive control, behavior-based control, fuzzy logic, and finite state machine. Orca [43] is a component-based middleware. Communication between its components is handled by the Internet Communication Engine (Ice) which is a general-purpose communication middleware [83], as shown in Figure 6.

OpenRTM-aist [28] is a component-based framework. The RT Component may be a device unit, like a servomotor, and a sensor, or a combination of device units, such as a mobile platform and arm or software modules, such as control algorithm or image processing algorithm. The architecture of the RT Component is shown in Figure 7 [28]. Each RT Component has the interfaces, called Ports, to communicate with other components or exchange data. The RT system is constructed by connecting the ports of RT-components.

ORoCoS is a Real-Time Toolkit that provides the components to be able to run on a real-time operating system. As described in [53], it consists of the following libraries: the Orocos Components Library (OCL) that provides some ready-to-use control components, the Orocos Kinematics and Dynamics Library (KDL) that provides the real-time calculation of kinematic chains, and the Orocos Bayesian Filtering Library (BFL), such as Kalman Filters and Particle Filters.

Skilligent [61] is a modular-structured framework. The core of the Skilligent Robot Learning and Behavior Control System consists of various modules, such as Audio Recognition and Robot Learning, and Behavior Execution and Coordination. Furthermore, various external software modules can be attached to the core, as shown in Figure 8.

OPRoS is a component-based platform supporting a client/server scheme for control flow and a publisher/subscriber scheme for data/event flow. ROS is a "metaoperating system" [51] for robot software consists of many small tools designed to work together. "The ROS system is a computation graph consisting of a set of nodes communicating with one another over edges" [51]. It consists of nodes (software modules), messages (passed peer to peer), topics, and services (analogous to web services). Nodes communicate together by passing messages through publish/subscribe model. Messages are not based on a specific programming language. "A node sends a message by publishing it to a given topic, which is simply a string. A node that is interested in a certain kind of data will subscribe to the appropriate topic." [51].

There are four main components in MRDS [36]: Concurrency and Coordination Runtime (CCR), Decentralized Software Services (DSSs), Visual Programming Language (VPL), and Visual Simulation Environment (VSE). The CCR is a .NET-based concurrent library for coordinating between the multiple sensors and robot actuator and managing asynchronous, parallel tasks through messages. DSS is a service-oriented runtime allowing multiple services to work together in order to achieve certain tasks and behaviors. VPL is integrated with Visual Studio to give the developers the facilities to implement a program through drag and drop blocks (activities or services) onto the design surface. A program, implemented with VPL, can collect and process data from the various sensors and create autonomous behavior for any robot that has a distance-measuring device and differential drive system. Finally, VSE is a simulation environment (Figure 9).

Player [13] is a device server (application server) with a collection of dynamically loadable device-shared libraries. It is a queue-based message passing system, consisting of two parts: the Player Core and the Transport Layer. "The core system includes the device and driver classes, the dynamic library loading code, configuration file parsing and the driver registry" [13]. Each driver has a single incoming message queue and can publish messages to the broadcast data and to all other drivers' subscribed client queues. The core library defines message syntax and coordinates the passing of these messages. The Transport Layer is independent of device drivers and is based on TCP communication protocol using sockets (and message queues).

Irobot Aware is implemented as a data messaging system designed through a publisher/subscriber scheme. As described in [70], Irobotware consists of the following: base tools and component, robotics, and operator control unit (OCU) frameworks. The base tools are open-source software packages dealing with the low-level details of the OS and hardware. Component/Robotics/OCU frameworks provide the software applications with infrastructure for autonomous networked systems/mobile robot-specific applications/real-time OCU controls and displays.

RSCA "consists of a real-time operating system, a communication middleware, and a deployment middleware called core framework" [60]. CARMEN [84] consists of three layers: the base layer to hide the low-level details of the hardware, the navigation layer to provide navigation primitives, such as motion planning, localization, and dynamic object tracking, and "the third tier is reserved for user-level tasks employing primitives from the second tier".

4. Simulation Environment

The simulation environment is very important for fast prototyping and educational purposes, although it may have some drawbacks and limitations, such as lack of noisy data existing in the realistic world and simulated models that may

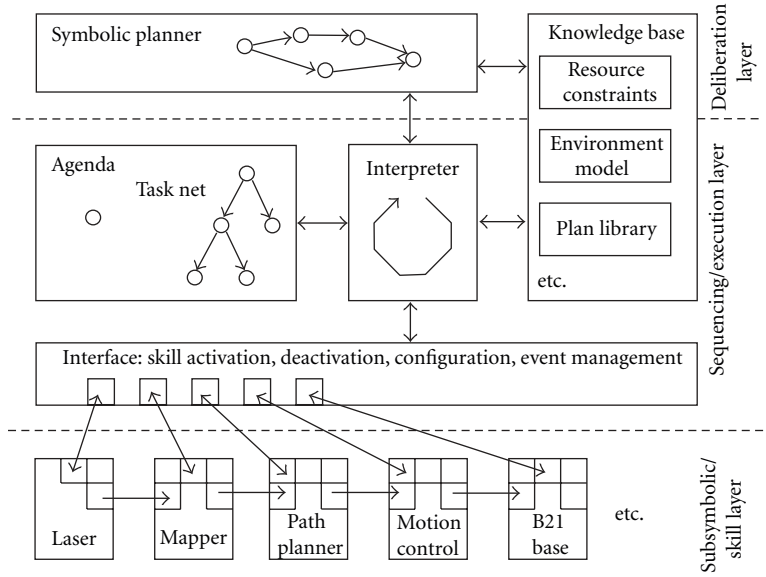

Figure 5: Smartsoft system architecture. (© (1999) IEEE) (reproduced from [67] with permission of Dr. Christian Schlegel).

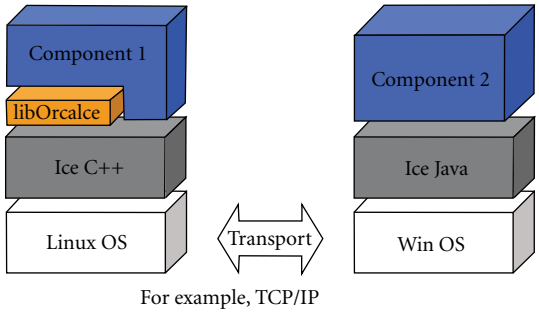

Figure 6: Two Orca components acting as a server or a client. (© (2006) IEEE), (reproduced from [43] with permission of Dr. Alexei Makarenko).

be incomplete or inaccurate. Table 2 shows whether or not each middleware supports a simulation environment.

Most of the middleware, such as iRobotAWARE [70], CLARAty [16], Orca, Miro [2], ASEBA, MRDS [35], ROS [51], RoboFrame [63], OpRoS [47], and CARMEN [76], provides simulation environment, although Orocos, Skilligent, RSCA, and ERSP do not come with a simulation environment. The Player [9] project provides a graphical, two-dimensional device simulator called Stage, which supports research in multirobot systems by using socket-based communication and a high-fidelity, three-dimensional simulator called Gazebo. In the Player [50], it is easy to simulate nonexistent devices for research in device design. Marie [38] provides interfaces to Stage, Gazebo, the ARIA, and CARMEN simulators. OpenHRP3 is a dynamics simulator based on OpenRTM-aist. Webots is a simulation environment. MuRoSimF is a simulation framework developed in RoboFrame. SmartSoft uses Player/Stage 2D simulator and also Gazebo 3D simulator. Pyro [80] is integrated with several existing robot simulators, including Robocup Soccer, Player/Stage, Gazebo, and the Khepera simulator.

5. Standards and Technologies Used

Table 2 summarizes the standards and technologies used in each middleware. Miro [33] and Orocos [85] are implemented using the Common Object Request Broker architecture (CORBA) standard. CORBA allows interprocess and cross-platform interpretability for distributed robot control. Although Orca [43] uses ICE, CORBA and ICE provide the basic functionality for component interaction. Smartsoft [68] provides two reference implementations, one based on CORBA (ACE/TAO) and one based on ACE only. Webots [50] uses Open Dynamics Engine (ODE) for detecting collisions and simulating rigid body dynamics. In fact, OpenRTM-aist and OPRoS implement the same standard, OMG's Robot Technology Component Standard.

6. Distributed Environment

The different software modules of an application should be able to exchange data and be able to run in different machines, from which each one is able to obtain its maximum efficiency. Table 2 summarizes whether or not each middleware supports distributed environment. In CLARAty, there are modules that support distributed processing: ACE/TAO (CORBA), sockets, published subscribed mechanisms, and so forth, but CLARAty modules cannot be readily distributed without either existing ACE/TAO wrappers or additional software development (clarified via correspondences with the author Dr. Issa Nesnas).

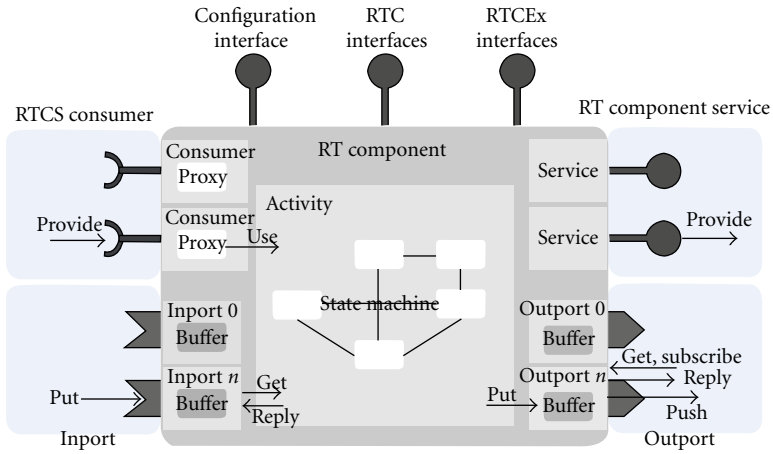

Figure 7: RT-Component Architecture (© (2006) IEEE) (reproduced from [28] with permission of Dr. Noriaki Ando).

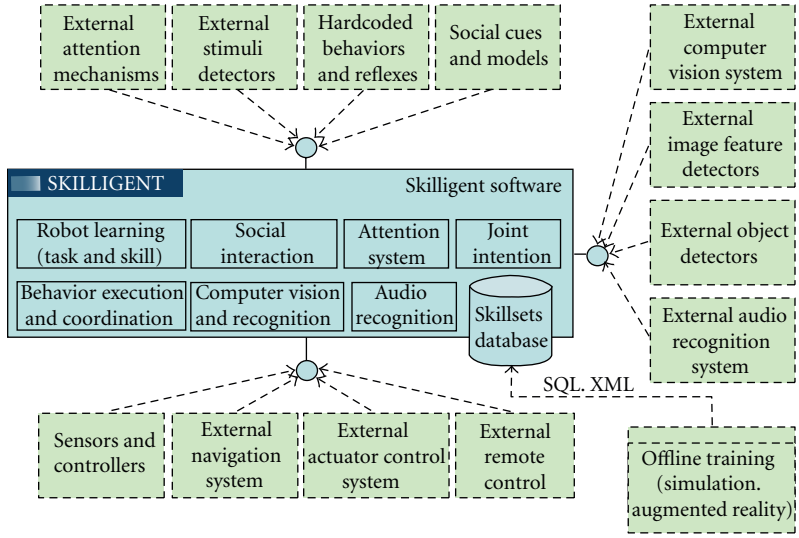

Figure 8: Various external software modules attached to the Skilligent Control System (reproduced from [61] with permission of Skilligent).

7. Security for Controlling Access

Data transportation and user access should be secure so that no one can control the robots other than the user. Some Middleware platforms, such as Orocos [53] and Miro [33] (based on CORBA), do not support security for control access, but CORBA [86] itself supports SSL for its communication. Table 2 summarizes whether each middleware supports security capabilities for controlling access.

8. Fault Detection and Recovery

Fault detection and recovery capabilities are necessary to provide the framework with the ability to be used in real, critical situations. A failure in one module should not damage the whole system. There is always the possibility of a fault at runtime. The faults in the robot's framework should be detected and localized, and also, the robot should be able to complete its mission or at least to proceed to a safe mode. Table 2 summarizes the fault detection and recovery capabilities for each middleware. ORCA, MIRO, OpenRTM-aist, and Player do not provide any explicit fault tolerant capabilities on the system level apart from the exception list, which may indicate service failures. CLARAty [87] offers a broad variety of low- and high-level means for fault tolerant and robust system performance (state monitoring and recovery from some faults, resources checking, state estimation, verification, test, simulation classes, (clarified via correspondences with the author Dr. Issa Nesnas). CARMEN programs are "robust to a variety of failures" [75] (in communications and of other programs). In ROS [51], there is additional fault tolerance as crashes are isolated to individual nodes.

9. Real-Time Capability

Reat-time capability of a robot middleware means that the reactiveness of a robot is guaranteed by the real-time

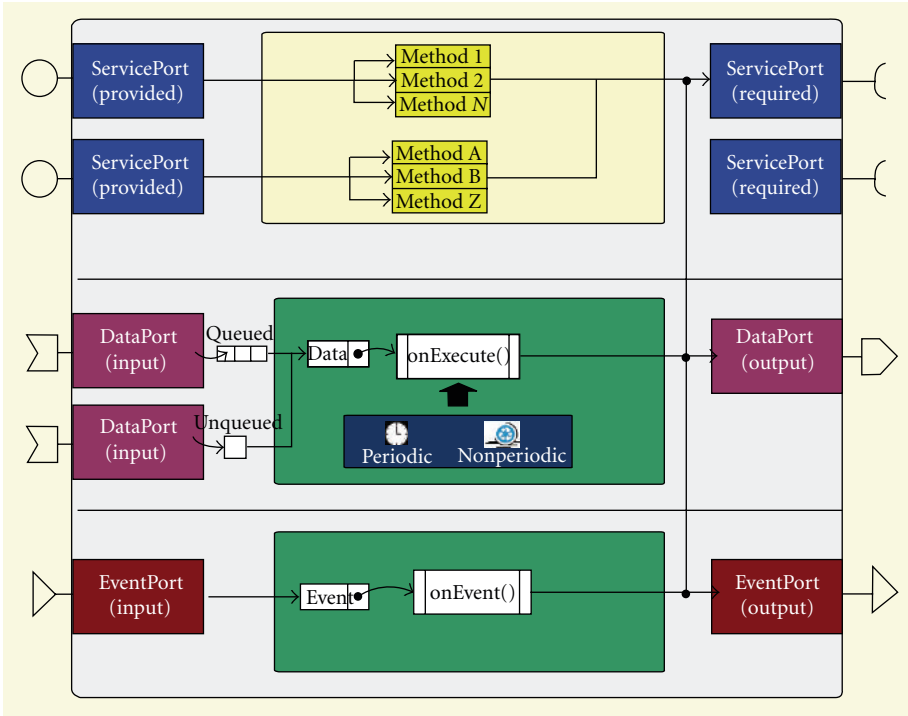

Figure 9: OPRoS component model (© (2010) ETRI) (reproduced from [47] with permission of ETRI Journal).

system by providing real-time capabilities for the component communication and process in the framework. Table 2 summarizes the real-time capability for each middleware. RSCA [60] provides real-time capability, if the operating environments support RT-CORBA and POSIX RT profile (PSE52) (clarified via correspondences with the author Dr. Seongsoo Hong). In CLARAty, most modules are real time and operate under VxWorks and QNX RTOS (clarified via correspondences with the author Dr. Issa Nesnas).

10. Behavior Coordination

Behavior Coordination is not part of RoboFrame (but robot coordination can be provided by integrating XABSL as a component, running on top of RoboFrame) (clarified via correspondences with the author Dr. Dirk Thomas). In RSCA, the behavior coordination should be implemented as an RSCA application or a set of RSCA components (clarified via correspondences with the author Dr. Seongsoo Hong). Table 2 summarizes the behavior coordination capability of the middleware.

11. Open Source

OPRoS, OROCOS, ROS, OpenRTM-aist, MARIE, ORCA, ASEBA, RSCA, SmartSoft, Pyro, MIRO, and Player are open-source software, although ERSP, Skilligent, Webots, and iRobotAWARE are commercial software products. MSRDS is commercial but is free of charge for research and hobbies. In CLARAty, most modules have been approved for public release. Only about 10% has been released due to funding constraints (clarified via correspondences with the author Dr. Issa Nesnas). RoboFrame is not open-source but available for research use for free (clarified via correspondences with the author Dr. Dirk Thomas).

12. Dynamic Wiring

This feature allows dynamic configuration of connections between services of components at runtime, making both control flow and the data flow configurable. As described in [88], "the automatic configuration facility in Marie is not available so communication setup should be static. It definitely is not well suited for all the requirements of dynamic scenarios." Autoconfiguration is not supported in ORCA [42], although ORCA relies on the ICE [83] Naming Service for delivering location transparency. But the dynamic wiring is supported in Orocos and SmartSoft [66] through scripting, XML, and run-time parameter setting. In SmartSoft [69], the wiring pattern provides a consistent mechanism for dynamic wiring of client parts of communication patterns from outside a component. CLARAty partially supports the dynamic wiring, Only some components that required this feature support runtime reconfiguration. But that is not a general rule that applies to all components in CLARAty (clarified via correspondences with the author Dr. Issa Nesnas).

13. Other Platforms

There are several other robotics software platforms available, such as Yet Another Robot Platform (YARP) [89, 90],

Table 2: Comparisons of the attributes of the main robotics middleware.

Name	System model	Control model	Fault tolerance	Simulator	Linux	Windows	Standards and technologies used	Open source	Behavior coordination	Real time	Distributed environment	Dynamic wiring	Security
CLARAty	2-layer AM; decentralized data; client server; platform independent abstractions and interfaces for various robotic components for motion control, coordination, mobility, manipulation, perception, estimation, navigation and planning.	Supports adaptation for centralized and distributed control; event driven	Yes	Yes	Yes	Only cygwin	OO design patterns, generic programming (C++ STL), ACE/TAO, CP-PUnit, Qt, TCP, UDP, Doxygen	Partially	supported	Most modules are real time	Yes	Partially	Yes
Player	No particular architectural constraint, client-server, decentralized data	Not applicable; but on module level can be considered centralized, since it relies on polling model	No explicit fault handling capabilities	Stage is a 2D simulator, Gazebo is a 3D simulator	Yes	Yes	3-Tier Architecture Proxy Objects	Yes	No	No	Yes	Yes	Yes
ORCA	No particular architectural constraint, peer to peer, decentralized data, component-based robotic systems	Not applicable; not clear on component level	No explicit fault handling capabilities	Yes	Yes	Yes	Ice	Yes	No	No	Yes	No	
MIRO	3 layers, client-server, decentralized data	On object level there is an emphasis on event–driven control	No explicit fault handling capabilities	Yes	Yes	Yes	TAO Middleware C++ imple-mentation of the CORBA standard	Yes	Yes	No	Yes	Yes	No

Table 2: Continued.

Name	System model	Control model	Fault tolerance	Simulator	Linux	Windows	Standards and technologies used	Open source	Behavior coordination	Real time	Distributed environment	Dynamic wiring	Security
OpenRTM-aist	Component-based frame work, model-driven architecture, platform-independent model (PIM), Platform-specific model (PSM)	Component-based	Supported by RT-component model	OpenHRP3 is a dynamic simulator	Yes	Yes	CORBA	Yes	No	Yes	Yes	Yes	No
ASEBA	Event-driven distributed control	Event-based control	No	Yes	Yes	No	Event-based middleware, Virtual machines	Yes	Yes	Yes	Yes	Yes	No
MARIE	Component-oriented engineering approach, 3 layers	Centralized control unit	No	Yes	Yes	No	Mediator Interoperability technology, ACE	Yes	Yes	No	Yes	Yes	No
RSCA	Consists of real-time OS, communication middleware, and deployment middleware called core frame-work	Event-based control	No	No	Yes	Yes	POSIX. 13. CORBA. RT-CORBA vl. 1	Yes	No	Yes	Yes	Yes	No
OPRoS	Component-based frame-works, validation/test tools	Client/server mechanism for control flow and the publisher/subcriber mechanism for data/event flow.	Being developed	Yes	Yes	Yes	event-driven	Yes	No	Being developed	Yes	Yes	No
ROS	Component-based frame-work, publisher/subscriber	Message oriented frame-works	Not Explicit	Yes	Yes	Partial functions	Message oriented, RPC services	Yes	Yes	Yes	Yes	Yes	No
MRDS	Component-based, REST	Distributed messaging		Yes	No	Yes	.NET/SOA	Comm.	Yes	No	Yes	Yes	Yes
OROCOS	C++ libraries: OCL, KDL, BFL,	Event-based control		No, but it has Orocos Simulik Toolbox	Yes	Yes	ACE/TAO, CORBA	Open source	No	Yes	No	Yes	No

Table 2: Continued.

Name	System model	Control model	Fault tolerance	Simulator	Linux	Windows	Standards and technologies used	Open source	Behavior coordination	Real time	Distributed environment	Dynamic wiring	Security
SmartSoft	Service-oriented, component-based software, model-driven architecture, platform-independent model and platform-specific model	Client/server, publish/subscribe, master/slave, arbitrary control models within component hull	Being Developed	Yes	Yes	Yes	Two reference Implementations, one based on CORBA (ACE/TAO) and one based on ACE only	Yes	Yes	yes, runs with RTAI-Linux and QNX	Yes	Yes	Yes
ERSP	3 layers		No		No	Yes	Yes	Comm	Yes	No	No	No	Yes
Skilligent			Yes	No	Yes	Yes		Comm.	Yes	No	Yes		
Webots	Multiprocess architecture		Can simulate dynamical device failure: (physical destruction, noise increase, etc.)	Yes	Yes	Yes	TCP Socket interface, Open Dynamics Engine, Ogre 3D	Comm.	Yes	Yes	No	No	
Irobotaware	Layered architecture	Publish/subscribe, Messaging		Yes	Yes			Comm.	Yes	No	Yes	Yes	Yes
Carmen	3T hybrid architecture			2D simulator	Yes	No	Socket based, using TCP protocol, IPC	Yes	No	No	Yes	Yes	Yes
RoboFrame	Message-oriented publish/subscribe and shared memory communication mechanisms	Message-based	No	Yes	Yes	Yes	Socket based	No	No	No	Yes	Yes	No
Pyro	Architecture independent		No	Yes	Yes	Yes	Socket based using TCP protocol, XML, SOAP, OpenGL, HTTP	Yes	Yes	No	No	Yes	Yes

SPICA [88, 91, 92], BABEL [93, 94], Dave's Robotic Operating System (DROS) [95], Intelligent Robot Software Platform (IRSP) [96], K-MIDDLEWARE [97], the Washington University Robotics Development Environment (WURDE) [98], OpenRDK [99, 100], OpenJAUS [101], Open Robot Controller Computer Aided Design (ORCCAD) [102–104], Pyro [77–81], Robot Intelligence Kernel (RIK) [105, 106], MissionLab [107–110], and Mobile Robot Programming Toolkit (MRPT) [111].

14. Conclusions and Bibliography Access Information

In this survey, we outlined the architecture and some important attributes, with the appropriate bibliographic references for most of the existing robotic middleware, such as Player, CLARAty, ORCA, MIRO, UPNP, RT-Middleware, ASEBA, MARIE, RSCA, OPRoS, ROS, MRDS, OROCOS, SmartSoft, ERSP, Skilligent, Webots, Irobotaware, Pyro, Carmen, and RoboFrame. All references listed in this paper can be found in http://www1bpt.bridgeport.edu/~aelkady/Survey.bib. They are stored in a BIBTEX format file: *survey.bib*.

Acknowledgments

The authors would like to sincerely thank Oskar von Stryk (ROBOFrame), Stphane Magnenat (ASEBA), Reid Simmons and Nesnas, Issa (CLARAty), Francois Michaud and Carle Cot (Marie), Ando Noriaki (OpenRTM-aist), Alex Makarenko (ORCA), Richard Vaughan (Player), Jean-Christophe Baillie (URBI), Sang C. Ahn (Upnp), Douglas.Few (RIK), and Byoundyoul Song (OPROS), for their help and support while compiling this survey. They also would like to express their gratitude to Christian Schlegel (SmartSoft), Alex Makarenko (ORCA), Kasper Stoy (Player), Olivier Michel(Webots), Ronald (Arkin MissionLab), Byoundyoul Song (OPROS), and Dirk Thomas (RoboFrame) who reviewed and provided them with very useful comments regarding Table 2, which summarizes a comparison of the attributes of main robotic middeware designs.

References

[1] W. D. Smart, "Is a common middleware for robotics possible?" in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems Workshop on Measures and Procedures for the Evaluation of Robot Architectures and Middleware (IROS '07)*, E. Prassler, K. Nilsson, and A. Shakhimardanov, Eds., 2007.

[2] J. Kramer and M. Scheutz, "Development environments for autonomous mobile robots: a survey," *Autonomous Robots*, vol. 22, no. 2, pp. 101–132, 2007.

[3] N. Mohamed, J. Al-Jaroodi, and I. Jawhar, "Middleware for robotics: a survey," in *Proceedings of the IEEE International Conference on Robotics, Automation and Mechatronics (RAM '08)*, pp. 736–742, September 2008.

[4] N. Mohamed, J. Al-Jaroodi, and I. Jawhar, "A review of middleware for networked robots," *International Journal of Computer Science and Network Security*, vol. 9, no. 5, pp. 139–148, 2009.

[5] M. Namoshe, N. Tlale, C. Kumile, and G. Bright, "Open middleware for robotics," in *Proceedings of the 15th International Conference on Mechatronics and Machine Vision in Practice (M2VIP '08)*, pp. 189–194, Auckland, New Zealand, December 2008.

[6] D. Bakken, "Middleware," in *Encyclopedia of Distributed Computing*, J. Urban and P. Dasgupta, Eds., Kluwer Academic, Dodrecht, The Netherlands, 2001.

[7] I. A. D. Nesnas, R. Simmons, D. Gaines et al., "claraty: challenges and steps toward reusable robotic software," *International Journal of Advanced Robotic Systems*, vol. 3, no. 1, pp. 023–030, 2006.

[8] B. P. Gerkey and M. J. Mataric, "Sold!: auction methods for multi-robot coordination," in *Proceedings of the IEEE Transactions on Robotics and Automation, Special Issue on Multi-robot Systems*, 2001.

[9] B. Gerkey, R. Vaughan, and A. Howard, "Howard, the player/stage project: tools for multi-robot and distributed sensor systems," in *Proceedings of the 11th International Conference on Advanced Robotics (ICAR '03)*, Coimbra, Portugal, 2003.

[10] B. P. Gerkey, R. T. Vaughan, K. Støy, A. Howard, G. S. Sukhatme, and M. J. Matarić, "Most valuable player: a robot device server for distributed control," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1226–1231, Wailea, Hawaii, USA, November 2001.

[11] R. T. Vaughan, B. P. Gerkey, and A. Howard, "Howard, on device abstractions for portable, reusable robot code," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '03)*, pp. 2121–2427, Las Vegas, Nev, USA, 2003.

[12] M. Kranz, R. B. Rusu, A. Maldonado, M. Beetz, and A. Schmidt, "A player/stage system for context-aware intelligent environments," in *Proceedings of the System Support for Ubiquitous Computing Workshop, at the 8th Annual Conference on Ubiquitous Computing (Ubicomp '06)*, Orange, Calif,USA, September 2006.

[13] T. H. Collett, B. A. MacDonald, and B. P. Gerkey, "Player 2.0: toward a practical robot programming framework," in *Proceedings of the Australasian Conference on Robotics and Automation (ACRA '05)*, Sydney, Australia, 2005.

[14] T. Estlin, R. Volpe, I. A. D. Nesnas et al., "Decision-making in a robotic architecture for autonomy," in *Proceedings of the International Symposium on Artificial Intelligence, Robotics, and Automation in Space (iSAIRAS '01)*, pp. 92152–97383, 2001.

[15] I. A. D. Nesnas, R. Volpe, T. Estlin, H. Das, R. Petras, and D. Mutz, "Toward developing reusable software components for robotic applications," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '01)*, pp. 2375–2383, November 2001.

[16] R. Volpe, I. A. D. Nesnas, T. Estlin, D. Mutz, R. Petras, and H. Das, "The claraty architecture for robotic autonomy," in *Proceedings of the IEEE Aerospace Conference*, vol. 1, pp. 1121–1132, Big Sky, Mont, USA, March 2001.

[17] I. A. D. Nesnas, "The claraty project: coping with hardware and software heterogeneity," in *Springer Tracts in Advanced Robotics*, vol. 30, pp. 31–70, Springer, Berlin, Germany, 2007.

[18] A. Diaz-Calderon, I. A. D. Nesnas, H. D. Nayar, and W. S. Kim, "Towards a unified representation of mechanisms for

robotic control software," *International Journal of Advanced Robotic Systems*, vol. 3, no. 1, pp. 061–066, 2006.

[19] T. Estlin, D. Gaines, C. Chouinard et al., "Enabling autonomous rover science through dynamic planning and scheduling," in *Proceedings of the IEEE Aerospace Conference*, pp. 385–396, March 2005.

[20] M. Bualat, C. Kunz, A. Wright, and I. A. D. Nesnas, "Developing an autonomy infusion infrastructure for Robotic exploration," in *Proceedings of the IEEE Aerospace Conference*, vol. 2, pp. 849–860, March 2004.

[21] M. Bualat, C. Kunz, A. Wright, and I. A. D. Nesnas, "Developing an autonomy infusion infrastructurefor Robotic exploration," in *Proceedings of the IEEE Aerospace Conference*, vol. 2, pp. 849–860, March 2004.

[22] R. Volpe, "Rover functional autonomy development for the mars mobile science laboratory," in *Proceedings of the IEEE Aerospace Conference*, vol. 2, pp. 643–652, 2003.

[23] I. A. D. Nesnas, A. Wright, M. Bajracharya, R. Simmons, T. Estlin, and W. S. Kim, "Claraty: an architecture for reusable robotic software," in *Space Robots*, vol. 5083 of *Proceedings of SPIE*, pp. 253–264, April 2003.

[24] I. A. D. Nesnas, A. Wright, M. Bajracharya, R. Simmons, and T. Estlin, "Claraty and challenges of developing interoperable robotic software," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '03)*, pp. 2428–2435, October 2003.

[25] C. Urmson, R. Simmons, and I. A. D. Nesnas, "A generic framework for robotic navigation," in *Proceedings of the IEEE Aerospace Conference*, vol. 5, pp. 2463–2470, 2003.

[26] C. Chouinard, F. Fisher, D. Gaines, T. Estlin, and S. Schaffer, "An approach to autonomous operations for remote mobile robotic exploration," in *Proceedings of the IEEE Aerospace Conference*, vol. 1, pp. 1–322, 2003.

[27] Y. Tsuchiya, M. Mizukawa, T. Suehiro, N. Ando, H. Nakamoto, and A. Ikezoe, "Development of light-weight RT-component (LwRTC) on embedded processor-application to crawler control subsystem in the physical agent system," in *Proceedings of the International Joint Conference (SICE-ICASE '06)*, pp. 2618–2622, October 2006.

[28] K. Ohara, T. Suzuki, N. Ando, B. Kim, K. Ohba, and K. Tanie, "Distributed control of robot functions using RT middleware," in *Proceedings of the International Joint Conference (SICE-ICASE '06)*, pp. 2629–2632, October 2006.

[29] N. Ando, T. Suehiro, K. Kitagaki, T. Kotoku, and W. K. Yoon, "RT-component object model in RT-middleware—distributed component middleware for RT (Robot Technology)," in *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA '05)*, pp. 457–462, June 2005.

[30] N. Ando, T. Suehiro, K. Kitagaki, T. Kotoku, and W. K. Yoon, "RT-middleware: distributed component middleware for RT (Robot Technology)," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '05)*, pp. 3933–3938, 2005.

[31] N. Ando, T. Suehiro, K. Kitagaki, T. Kotoku, and W. K. Yoon, "Composite component framework for RT-Middleware (Robot technology middleware)," in *Proceedings of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM '05)*, pp. 1330–1335, Monterey, Calif, USA, July 2005.

[32] H. Chishiro, Y. Fujita, A. Takeda et al., "Extended RT-component framework for RT-middleware," in *Proceedings of the IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC '09)*, pp. 161–168, Tokyo, Japan, March 2009.

[33] H. Utz, S. Sablatnog, S. Enderle, and G. Kraetzschmar, "Miro—middleware for mobile robot applications," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 4, pp. 493–497, 2002.

[34] S. Enderle, H. Utz, S. Sablatnög, S. Simon, G. Kraetzschmar, and G. Palm, "Miro: middleware for autonomous mobile robots," in *Telematics Applications in Automation and Robotics*, 2001.

[35] K. Johns and T. Taylor, *Professional Microsoft Robotics Developer Studio*, Wrox Press, Birmingham, UK, 2008.

[36] J. Jackson, "Microsoft robotics studio: a technical introduction," *IEEE Robotics and Automation Magazine*, vol. 14, no. 4, pp. 82–87, 2007.

[37] S. Morgan, *Programming Microsoft Robotics Studio*, Microsoft Press, Redmond, Wash, USA, 2008.

[38] C. Côté, D. Létourneau, F. Michaud et al., "Code reusability tools for programming mobile robots," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '04)*, pp. 1820–1825, October 2004.

[39] É. Beaudry, Y. Brosseau, C. Ct et al., "Reactive planning in a motivated behavioral architecture," in *Proceedings of the National Conference on Artificial Intelligence (AAAI '05)*, vol. 3, pp. 1242–1247, 2005.

[40] C. Côté, Y. Brosseau, D. Létourneau, C. Raïevsky, and F. Michaud, "Robotic software integration using MARIE," *International Journal of Advanced Robotic Systems*, vol. 3, no. 1, pp. 55–60, 2006.

[41] C. Côté, D. Létourneau, C. Raïevsky, Y. Brosseau, and F. Michaud, "Using marie for mobile robot component development and integration," *Software Engineering for Experimental Robotics Book Series*, vol. 30 of *Springer Tracts in Advanced Robotics*, Springer, Berlin, Germany, 2007.

[42] A. B. Alexei Makarenko and T. Kaupp, "On the benefits of making robotic software frameworks thin," in *Proceedings of the Benefits of Making Robotic Software Frameworks Thin IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '07)*, San Diego, Calif, USA, October-November 2007.

[43] T. K. Alexei Makarenko and A. Brooks, "Orca: components for robotics," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems Workshop on Robotic Standardization (IROS '06)*, Beijing, China, October 2006.

[44] A. Brooks, T. Kaupp, A. Makarenko, S. Williams, and A. Orebäck, "Towards component-based robotics," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '05)*, pp. 3567–3572, Edmonton, Canada, August 2005.

[45] C. Jang, B. Song, S. Jung et al., "A development of software component framework for robotic services," in *Proceedings of the 4th International Conference on Computer Sciences and Convergence Information Technology (ICCIT '09)*, pp. 1–6, Seoul, Korea, November 2009.

[46] B. Song, S. Jung, C. Jang, and S. Kim, "An introduction to robot component model for opros(open platform for robotic services)," in *Proceedings of the International Conference Simulation, Modeling Programming for Autonomous Robots Workshop*, pp. 592–603, 2008.

[47] C. Jang, S. I. Lee, S. W. Jung et al., "Opros: a new component-based robot software platform," *ETRI Journal*, vol. 32, no. 5, pp. 646–656, 2010.

[48] Ersp 3.1 software development kit, 2010, http://www.evolution.com/products/ersp/.

[49] Webots, 2009, http://www.cyberbotics.com.

[50] O. Michel, "Cyberbotics ltd. webots professional mobile robot simulation," *International Journal of Advanced Robotics Systems*, vol. 1, pp. 39–42, 2004.

[51] Robot operating system (ros), 2011, http://www.ros.org.

[52] M. Quigley, K. Conley, B. Gerkey et al., "Ros: an open-source robot operating system," in *Proceedings of the Workshop on Open Source Software (ICRA '09)*, 2009.

[53] P. Soetens, RTT: Real-Time Toolkit, 2010, http://www.Orocos.org/rtt.

[54] H. Bruyninckx, P. Soetens, and B. Koninckx, "The real-time motion control core of the Orocos project," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 2766–2771, September 2003.

[55] P. Soetens and H. Bruyninckx, "Realtime hybrid task-based control for robots and machine tools," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 260–265, April 2005.

[56] P. Soetens, *A software framework for real-time and distributed robot and machine control*, Ph.D. thesis, Department of Mechanical Engineering, Katholieke Universiteit Leuven, Heverlee, Belgium, 2006, http://www.mech.kuleuven.be/dept/resources/docs/soetens.pdf.

[57] K. Gadeyne, T. Lefebvre, and H. Bruyninckx, "Bayesian hybrid model-state estimation applied to simultaneous contact formation recognition and geometrical parameter estimation," *The International Journal of Robotics Research*, vol. 24, no. 8, pp. 615–630, 2005.

[58] K. Gadeyne, *Sequential monte carlo methods for rigorous bayesian modeling of autonomous compliant motion*, Ph.D. thesis, Department of Mechanical Engineering, Katholieke Universiteit Leuven, 2005.

[59] H. Bruyninckx, J. De Schutter, T. Lefebvre et al., "Building blocks for slam in autonomous compliant motion," in *Proceedings of the International Symposium on Robotics Research (ISRR '03)*, pp. 432–441, 2003.

[60] J. Yoo, S. Kim, and S. Hong, "The robot software communications architecture (RSCA): QoS-aware middleware for networked service robots," in *Proceedings of the International Joint Conference (SICE-ICASE '06)*, pp. 330–335, October 2006.

[61] Skilligent, 2010, http://www.skilligent.com/index.shtml.

[62] M. Friedmann, J. Kiener, S. Petters, D. Thomas, and O. von Stryk, "Modular software architecture for teams of cooperating, heterogeneous robots," in *Proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO '06)*, pp. 613–618, Kunming, China, December 2006.

[63] M. Friedmann, J. Kiener, S. Petters, D. Thomas, and O. von Stryk, "Reusable architecture and tools for teams of lightweight heterogeneous robots," in *Proceedings of the 1st IFAC Workshop on Multivehicle Systems (IFAC '06)*, pp. 51–56, Salvador, Brazil, 2006.

[64] S. Petters, D. Thomas, and O. von Stryk, "Roboframe—a modular software framework for lightweight autonomous robots," in *Proceedings of the Workshop on Measures and Procedures for the Evaluation of Robot Architectures and Middleware of the International Conference on Intelligent Robots and Systems (IEEE/RSJ '07)*, San Diego, Calif, USA, 2007.

[65] D. Thomas and O. von Stryk, "Efficient communication in autonomous robot software," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '10)*, pp. 1006–1011, Taipei, Taiwan, 2010.

[66] C. Schlegel and R. Woerz, "Software framework smartsoft for implementing sensorimotor systems," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '99)*, vol. 3, pp. 1610–1616, October 1999.

[67] C. Schlegel and R. Worz, "Interfacing different layers of a multilayer architecture for sensorimotor systems using the object-oriented framework smartsoft," in *Proceedings of the 3rd European Workshop on Advanced Mobile Robots (Eurobot '99)*, pp. 195–202, 1999.

[68] C. Schlegel, T. Hassler, A. Lotz, and A. Steck, "Robotic software systems: from code-driven to model-driven designs," in *Proceedings of the International Conference on Advanced Robotics (ICAR '09)*, pp. 1–8, Munich, Germany, June 2009.

[69] C. Schlegel, "Communication patterns as key towards component-based robotics," *International Journal of Advanced Robotic Systems*, vol. 3, no. 1, pp. 49–54, 2006.

[70] Aware 2 robot intelligient software, 2010, http://www.irobot.com/gi/developers/Aware/.

[71] S. Magnenat, P. Retornaz, M. Bonani, V. Longchamp, and F. Mondada, "ASEBA: a modular architecture for event-based control of complex robots," *IEEE/ASME Transactions on Mechatronics*, pp. 1–9, 2010.

[72] S. Magnenat, V. Longchamp, and F. Mondada, "Aseba, an event-based middleware for distributed robot control," in *Proceedings of the Workshops DVD of International Conference on Intelligent Robots and Systems (IROS '07)*, 2007.

[73] Python robotics website, 2011, http://www.pyrorobotics.org.

[74] Carnegie mellon robot navigation toolkit, 2008, http://carmen.sourceforge.net.

[75] M. Montemerlo, N. Roy, and S. Thrun, "Perspectives on standardization in mobile robot programming: the carnegie mellon navigation (carmen) toolkit," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '03)*, pp. 2436–2441, October 2003.

[76] Carmen, the carnegie mellon robot navigation toolkit, 2008, http://carmen.sourceforge.net.

[77] D. Blank, D. Kumar, and Bryn Mawr College, "Pyro: a python-based versatile programming environment for teaching robotics," *ACM Journal on Educational Resources in Computing*, vol. 3, no. 4, pp. 1–15, 2003.

[78] D. Blank, L. Meeden, and D. Kumar, "Python robotics: an environment for exploring robotics beyond LEGOs," in *Proceedings of the 34th Technical Symposium on Computer Science Education (SIGCSE '03)*, pp. 317–321, ACM Press, February 2003.

[79] D. S. Blank, D. Kumar, L. Meeden, and H. A. Yanco, "The pyro toolkit for AI and robotics," *AI Magazine*, vol. 27, no. 1, pp. 39–50, 2006.

[80] D. Blank, D. Kumar, L. Meeden, and H. Yanco, "Pyro: an integrated environment for Robotics education," in *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI '05)*, pp. 1718–1719, July 2005.

[81] D. Blank, H. Yanco, D. Kumar, and L. Meeden, "Avoiding the Karel-the-robot paradox: a framework for making sophisticated robotics accessible," in *Proceedings of the Spring Symposium on Accessible, Hands-on AI and Robotics Education (AAAI '04)*, 2004.

[82] O. Michel, "Webots: professional mobile robot simulation," *Journal of Advanced Robotics Systems*, vol. 1, pp. 39–42, 2004.

[83] M. S. Michi Henning, Distributed programming with ice, 2010, http://www.zeroc.com/doc/Ice-3.4.0/manual/.

[84] R. P. Bonasso, R. J. Firby, E. Gat, D. Kortenkamp, D. P. Miller, and M. G. Slack, "Experiences with an architecture for intelligent, reactive agents," *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 9, no. 2-3, pp. 237–256, 1997.

[85] H. Bruyninckx, P. Soetens, and B. Koninckx, "The real-time motion control core of the Orocos project," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 2, pp. 2766–2771, September 2003.

[86] Common object request broker architecture (corba), 2008, http://www.omg.org/spec/CORBA.

[87] R. Volpe, I. A. D. Nesnas, D. Mutz, R. Petras, and H. Das, "Claraty: coupled layer architecture for robotic autonomy," Tech. Rep., 2000, NASA Jet Propulsion Laboratory.

[88] P. A. Baer, *Platform-independent development of robot communication software*, Ph.D. thesis, University of Kassel, Munich, Germany, 2008.

[89] P. Fitzpatrick, G. Metta, and L. Natale, "Towards long-lived robot genes," *Robotics and Autonomous Systems*, vol. 56, no. 1, pp. 29–45, 2008.

[90] G. Metta, P. Fitzpatrick, and L. Natale, "YARP: yet another robot platform," *International Journal of Advanced Robotic Systems*, vol. 3, no. 1, pp. 43–48, 2006.

[91] P. A. Baer, R. Reichle, and K. Geihs, "The spica development framework—model-driven software development for autonomous mobile robots," in *Proceedings of the 10th International Conference on Intelligent Autonomous Systems (IAS-10 '08)*, W. Burgard, R. Dillmann, C. Plagemann, and N. Vahrenkamp, Eds., pp. 211–220, IAS Society, 2008.

[92] U. Kaufmann, R. Reichle, C. Hoppe, and P. A. Baer, "An unsupervised approach for adaptive color segmentation," in *Proceedings of the 1st International Workshop on Robot Vision (VISAPP '07)*, March 2007.

[93] J. A. Fernández-Madrigal, "The BABEL development system for integrating heterogeneous robotic software," Tech. Rep., System Engineering and Automation Department, University of Málaga, Málaga, Spain, 2003.

[94] J. Fernandez-Madrigal, C. Galindo, and J. Gonzalez, "Integrating heterogeneous robotic software," in *Proceedings of the IEEE Mediterranean Electrotechnical Conference (MELECON '06)*, pp. 433–436, Málaga, Spain, May 2006.

[95] Dave's robotic operating system, 2009, http://dros.org/.

[96] J. Y. Kwak, J. Y. Yoon, and R. H. Shinn, "An intelligent robot architecture based on robot mark-up languages," in *Proceedings of the IEEE International Conference on Engineering of Intelligent Systems (ICEIS '06)*, pp. 1–6, April 2006.

[97] D.-H. Choi, S.-H. Kim, K.-K. Lee, B.-H. Beak, and H.-S. Park, "Middleware architecture for module-based robot," in *Proceedings of the International Joint Conference (SICE-ICASE '06)*, pp. 4202–4205, Busan, South Korea, October 2006.

[98] F. Heckel, T. Blakely, M. Dixon, C. Wilson, and W. D. Smart, "The wurde robotics middleware and ride multi-robot tele-operation interface," in *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI '06)*, July 2006.

[99] D. Calisi, A. Censi, L. Iocchi, and D. Nardi, "Openrdk: a modular framework for robotic software development," in *Proceedings of the International Conference on Intelligent Robots and Systems (IROS '08)*, pp. 1872–1877, Nice, France, September 2008.

[100] D. Calisi, A. Censi, L. Iocchi, and D. Nardi, "Openrdk: a modular framework for robotic software development," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (SECESA '08)*, pp. 1872–1877, 2008.

[101] Openjaus, 2010, http://www.openjaus.com/.

[102] D. Simon, B. Espiau, K. Kapellos, and R. Pissard-Gibollet, "Orccad: software engineering for real-time robotics a technical insight," *Robotica*, vol. 15, no. 1, pp. 111–115, 1997.

[103] D. Simon, R. Pissard-Gibollet, and S. Arias, "Orccad, a framework for safe robot control design and implementation," in *Proceedings of the 1st National Workshop on Control Architectures of Robots: Software Approaches and Issues(CAR '06)*, Montpellier, France, 2006.

[104] D. Simon, F. Boudin, R. Pissard-Gibollet, and S. Arias, "Orccad, robot controller model and its support using eclipse modeling tools," in *Proceedings of the 5th National Conference on "Control Architecture of Robots" (CAR '10)*, 2010.

[105] D. J. Bruemmer, D. A. Few, M. C. Walton, and C. W. Nielsen, "The robot intelligence kernel," in *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI '06)*, pp. 1960–1961, Boston, Mass, USA, July 2006.

[106] Robot intelligence kernel, 2010, https://inlportal.inl.gov/portal/server.pt/community/robot_intelligence_kernel/457.

[107] D. C. Mackenzie, R. C. Arkin, and J. M. Cameron, "Multiagent mission specification and execution," *Autonomous Robots*, vol. 4, no. 1, pp. 29–52, 1997.

[108] D. C. MacKenzie and R. C. Arkin, "Evaluating the usability of robot programming toolsets," *International Journal of Robotics Research*, vol. 17, no. 4, pp. 381–401, 1998.

[109] Y. Endo, D. MacKenzie, and R. C. Arkin, "Usability evaluation of high-level user assistance for robot mission specification," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 34, no. 2, pp. 168–180, 2004.

[110] G. T. M. R. Laboratory, User manual for missionlab version 7.0, 2006, http://www.cc.gatech.edu/aimosaic/robot-lab/research//MissionLab/mlab_manual-7.0.pdf.

[111] The mobile robot programming toolkit, 2010, http://www.mrpt.org/.