ARS ADVANCED ROBOTIC SYSTEMS INTERNATIONAL

Software Systems for Robotics An Applied Research Perspective

Greg Broten; Simon Monckton; Jared Giesbrecht & Jack Collier
DRDC-Suffield, Autonomous Intelligent Systems Section
Firstname.Lastname@drdc-rddc.gc.ca

Abstract: *Over the past 20 years, Defence Research and Development Canada has developed numerous tele-operated unmanned ground vehicles (UGV), many founded on the ANCÆUS command and control system. This paper relates how long experience with tele-operated UGVs influenced DRDC's shift in focus from tele-operated to autonomous unmanned vehicles (UV), the forces that guided DRDC's development approach and DRDC's experience adapting a specific tool set, MIRO, to a UGV implementation.*
Keywords: *software complexity, middleware, components, frameworks, architectures*

1. Introduction

Over the past 20 years, Defence Research and Development Canada has developed numerous tele-operated unmanned ground vehicles (UGV), many founded on the ANCÆUS [Broten et al, 2003] command and control system. First fielded in 1990, ANCÆUS solved difficult wireless networking and vehicle control problems five years or more before similar military systems appeared internationally. With remarkable foresight, ANCÆUS decoupled the vehicle platform from both the commands and communications interface and proved adaptable to other platforms such as the combat CAT, the tele-operated Bobcat with a backhoe and the ILDP landmine detection vehicle depicted in Figure 1.

Hardware and software development eventually overtook ANCÆUS, however. Substituting today's networking technology; ANCÆUS is analogous to the Joint Architecture for Unmanned Systems (JAUS) [JAUS-WG, 2004] command set. Through ANCÆUS, DRDC learned that systems integration becomes more expensive, error prone, and time consuming with:

1. the need to port, extend and maintain customized software across platforms and payloads,
2. platforms and payloads that must evolve continuously to avoid obsolescence,
3. platforms and payloads that are task dependent.

This paper describes DRDC's entry into autonomous systems research and adoption of a methodology and tool chain for managing complex software development. Section 2 explains the hazards of complex software system development and how rapidly evolving technology drove DRDC's decisions. Section 3 reviews current trends in robot architectures, detailing the forces that have guided DRDC's development. Section 4 overviews DRDC's successful application of a specific tool set, MIRO, to an outdoor UGV implementation while Section 5 summarizes DRDCs experiences and current progress.

2. Software Complexity

The relentless increase in circuit density and efficiency is matched only by the growing size, complexity, and pervasiveness of software in personal, industrial, and military affairs. Driven by complexity, programming has evolved from CPU assembly language to network-aware Java and C#, proof that ``the increasing complexity of programs has driven the need for better ways to manage that complexity" [Schildt, 1994].
Military robotics are particularly susceptible to these forces. With sufficient sensors, distributed signal processing, world representation, and control algorithms, UVs can navigate modestly complex environments in real-time. In a military context, such systems challenge development teams to manage the complexity, size, diversity, and future evolution of software and hardware components – a role for Middleware toolkits.

3. Networking Middleware

Middleware toolkits have been identified as a key ingredient in managing software complexity [Gowdy, 2000b]. DRDC further realized that open source software and collaborative research are crucial to capturing and communicating science, the ultimate product of DRDC's UGV research. Therefore, DRDC evaluated all toolkits for open source availability, support and use of open tools and operating systems; support for modular, extensible, flexible and scalable software systems; support for multiple messaging paradigms; usage and maturity; and

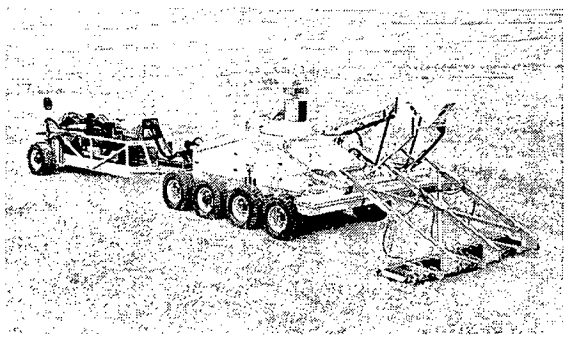

Fig. 1. Improved Landmine Detection Project (ILDP) Vehicle

applicability to robotic architectures. The following overviews some interprocess communication middleware packages and discusses some of their relative advantages and disadvantages.

3.1 CMU Toolkits: IPT, IPC and RTC
IPT [Gowdy, 1996], IPC [Simmons, 1991] and RTC [Pedersen, 1998] are related middleware toolkits emerging from the C.M.U.'s robotic research programs.

3.3 NML - Neutral Messaging Language
Used in NIST's NASREM and 4D/RCS architectures, NML is a uniform application interface (API) to communications functions [J. Michaloski, 2000].

3.4 NDDS - Network Data Distribution System
NDDS is a commercial middleware product by RealTime Innovations that implements the publish-subscribe model [G. Pardo-Castellote and Hamilton, 1999].

3.5 ACE - Adaptive Communications Environment
ACE is a widely-used, open-source, object-oriented middleware that implements core concurrency and networking patterns [Schmidt and Huston, 2002] for communications software [Huston et al., 2004].

Table 1. A subjective comparison of Middleware Toolkits: Ranked on a scale of 1 (low) to 5 (high).

	IPT/IPC	NML	NDDS	ACE	TAO	RTC
Open Source	5	5	1	5	5	2
Open Tools	5	5	1	5	5	5
Component Based	2	3	4	5	5	2
Messaging	2	2	3	3	5	2
Comprehensive	1	3	3	3	5	1
Portable	3	4	3	5	5	3
Reconfiguration	3	2	4	4	4	3
Applicability	3	3	3	5	5	3
Resources	5	5	4	3	1	5
Performance	5	5	5	5	3	5
Useage	1	3	3	5	5	2
Ease of Use	4	3	4	3	2	4

3.6 CORBA
CORBA is an open communications standard for developing portable distributed applications on heterogeneous systems [Henning and Vinoski, 1999], [Bolton, 2002] . TAO (The ACE ORB) based on ACE is open-source and adheres to the RT-CORBA specification [Schmidt and Kuhns, 2000].

3.7 Conclusions
Table 1 summarizes the middleware toolkits reviewed. This table shows that while CORBA is the largest consumer of computing resources it also allows for the most flexible, modular, portable, and extensible implementation.

4. Robot Middleware

Over the last 20 years, robotics has swung from philosophically grounded deliberative AI to reactive control, before settling on pragmatic *hybrid* architectures. This return to pragmatism, captured by Gowdy [Gow00b], resonates with DRDC's long experience with UGVs. Gowdy divides robot architectures into either reference or emergent architecture philosophies. Reference architectures establish idealized guidelines for component design and data flow. Emergent design advocates software frameworks that allow architectures to emerge and evolve.
As with middleware, DRDC evaluated a number of robot-specific middleware systems [Broten et al, 2004] specifically seeking frameworks meeting the basic criteria above and compatibility with an emergent design philosophy. Specifically: Player [Gerkey et al., 2003], DRCS [Albus, 1997], NASREM [Albus et al., 1987], TCA [Simmons, 1994], Dervish [Powers and Birchfield, 1995], CLARAty[Nes, 2003], RAP [Firby, 1989], Xavier [Koenig et al., 1998], Saphira [Konolige and Myers, 1998], 3T [Bonasso et al., 1997], Berra [Oreback and Christensen, 2000], Marie [Cote et al., 2004], Carmen [Roy and Thrun, 2003], OCP [Wills et al., 2001], MIRO [Utz et al., 2002] and Orca [Brooks et al., 2005]. The criteria quickly separated the potential candidates: Player/Stage, Carmen, Marie, MIRO and Orca; from the closed source (such as CLARAty, OCP and Saphira) and reference design (DRCS and NASREM) candidates. Each of the potential candidates are described below.

4.1. Player
Player/Stage/Gazebo is the de facto standard for academic mobile robot control and 2 or 3D robot simulation [Vaughan et al., 2003], but as a monolithic device server, lacks the robustness, extensibility, and network capability of the others.

4.2. MARIE

MARIE (Mobile and Autonomous Robotics Integration Environment) attempts to create an integrated and coherent framework to facilitate application reuse through tools and programming environments. Unfortunately, Marie's adoption is very limited.

4.3. CARMEN

CARMEN (The Carnegie Mellon Navigation Toolkit) is an open source collection of robot control software designed to provide a consistent interface and a basic set. of primitives for robotics research on a wide variety of commercial robot platforms [Roy et al 2003]. CARMEN is a complex toolkit with very limited application to largely indoor vehicles.

4.4. Orca

Orca is an open-source framework for component-based robotic systems with roots in the OROCOS KTH project, mandated to build communications patterns for distributed objects. Orca provides the means for defining, developing and assembling the building-blocks that form arbitrarily complex robotic systems, from single vehicles to distributed sensor networks. Though immature and incomplete in many respects, Orca provides a simple interface to TAO.

4.5. MIRO

Like Orca, MIRO (MIddleware for RObots) is a distributed, object-oriented framework for mobile robot control. MIRO reduces software development times and costs by providing data structures, functions, communications protocols and synchronization mechanisms specific to robots. Used predominantly by the Univ. of Ulm for soccer robots, MIRO provides a powerful but complex TAO toolset.

4.6. Conclusions

Five frameworks were examined for their ability to implement autonomy on DRDC vehicles, as summarized in Table 2. The investigation concluded that CORBA-based middleware offered the most modularity, extensibility, flexibility and scalability. Of the two reviewed frameworks based upon CORBA, only one is sufficiently mature and in current use. Thus MIRO, originally designed for soccer playing robots, proved to possess the flexibility and expandability sufficient for DRDC's unmanned vehicle program.

Table 2. A subjective comparisons between candidate frameworks ranked on a scale of 1 (low) to 3 (high).

	Open	Tools	Frame Works	Middle ware	Modular	Usage
Player	3	3	3	1	3	2
Carmen	3	3	2	2	2	1
MARIE	3	3	3	3	3	1
Miro	3	3	3	3	3	1
Orca	3	3	3	3	3	1

5. Adapting MIRO to Support an Outdoor UGV

DRDC adapted and extended MIRO to support the Kyoker Industries Raptor, an Ackerman steered, hydrostatic, all wheel drive utility vehicle. XJ Designs modified the vehicle, shown in Figure 2, such that throttle, brakes, steering, and other parameters could be monitored and controlled through a single software interface. DRDC added sensors for outdoor, autonomous operations including: forward and backward nodding SICK lasers, forward stereo cameras, differential GPS, inertial measurement unit, wheel encoders, wireless routers and computers.

Fig. 2. The Raptor UGV

While many of Univ. of Ulm's MIRO-based Robocup components were inappropriate for DRDC's Raptors, MIRO can easily address the more complex problems of outdoor multivehicle autonomy with uncertain communications.

5.1. Sensor Drivers

MIRO natively supports some sensors applicable to outdoor UGVs such as the odometry and RangeSensor interfaces. DRDC extended the original SICK laser driver to support a nodding SICK laser generating 3D data over TCP/IP. New drivers were developed to support a Novatel GPS, a Microstrain IMU, a stereo camera (Digiclops), and the Raptor vehicle command set. Each driver published to a device neutral data interface, a CORBA object. Interestingly, a Player device driver thread could be easily ported to a MIRO Device Server, often resulting in superior performance.

5.2. Components

Other than a basic DAMN-like structure, DRDC did not prescribe architecture details at the outset, instead

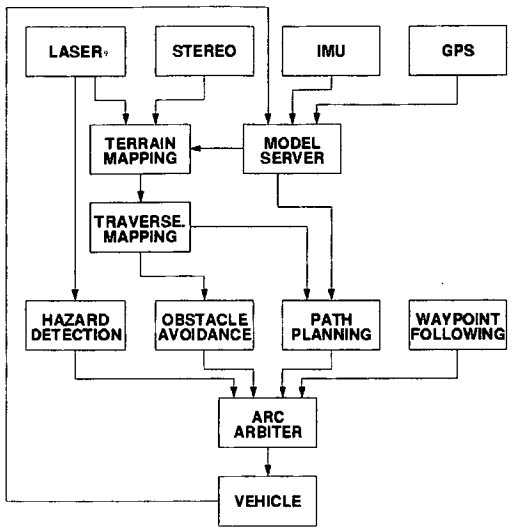

Fig. 3. Raptor Architecture Flow Diagram

allowing the final architecture to emerge and evolve based on sensing and control requirements. Depicted in Figure 3, the resulting services fall into 7 sub-domains:

1. Sensing: Laser ranging, Stereo camera, IMU, GPS and odometry.
2. World Representation: Vehicle Pose; Terrain and Traversibility maps.
3. Global Goal Management (not shown).
4. User Interface (not shown) .
5. Navigation: Pure Pursuit waypoint following, D*-lite Path Planning, Obstacle Avoidance and Hazard Detection.
6. Arbitration: Vote based Selection of steering arc and speed.
7. Low level Vehicle Command and Control.

Together these services formed four parallel "threads" of control: hazard detection, obstacle avoidance, path planning, and waypoint following.

5.3 Effort

Table 3 summarizes the implementation effort, estimated through simple source code line count, for this project. Adapting MIRO to support a UGV application required roughly 20,000 lines of new code implemented through 6 to 8 man years of effort. At project start in the Summer 2004, none of the staff were familiar with MIRO or CORBA. By Summer 2005, the staff were conversant in most aspects of MIRO operations after a steep but manageable learning curve.

Table 3. DRDC's estimated effort to adapt MIRO

Sub-Domain	Lines of Code
Sensing	3429
World Representation	8867
Global Goal Management	220
User Interface and Control	562
Navigation	4438
Arbitration	402
Vehicle Command and Control	3448
Total	**~20,000**

5.4. Results

In fall 2005, the Raptor vehicle performed field trials designed to reveal the system's strengths and weaknesses travelling through obstacle fields and over unimproved grassland prairie. This assessment falls into three categories: hardware assessment, software utility, and algorithmic suitability.

5.4.1 Hardware Assessment

For the moderate grassland prairie environment, the Raptor's sensing proved adequate for low speed maneuvers. The nodding laser and stereo imagery provided sufficient detail for traversing the selected environment at 5-10 km/h. With excellent range, field of view, and resolution, the nodding laser played a crucial role in the system.

5.4.2 Software Utility

Though complex, the ACE/TAO/MIRO toolchain greatly simplified software development. The mature publish/subscribe architecture, arbitrary data logging and playback, and IDL interface discipline, proved pivotal in the system's rapid evolution. With experience, developers could quickly develop and test new modules, often in parallel with earlier versions. MIRO effectively hides much of TAO's internal complexity, though the remaining learning curve is steep. MIRO's publish/subscribe architecture permits multiple processes to opportunistically draw on output data streams. Subscribers can be created or destroyed with little consequence to the publisher and can reside anywhere on the network. Through MIRO's event system, a single developer could quickly log arbitrary CORBA object events in the field, such as device or module output, for instant playback in the lab.

5.4.3 Algorithm Suitability

DRDC implemented a wrappable, variance weighted terrain map on the Raptor UGV under MIRO. Taking *Pose*, *Laser*, and *Stereo* events, the Terrain Map service inserted range data into 0.2x0.2 grid elements within a 20x20m patch centered on the vehicle bumper.

A Traversability Service translated the terrain map into a 20x20m traversability map with 1x1m grid. Proving sufficient for low speed maneuvers over Suffield's grassland environment, the traversability service's sensitivity to pose error effects reduced the map's range to 15m. A resulting traversibility map is shown in Fig. 4.

DRDC's Arbitration services combined discrete behaviour modules in a reactive arbitration system to create

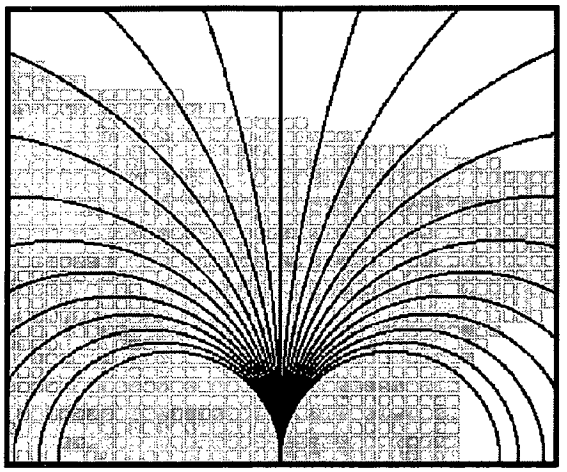

Fig. 4. A Traversibilty Map overlaid with 25 candidate arcs. Obstacles: Blue, Traversable: Red

autonomous behaviour. The pure pursuit controller performed without fault, even using unfiltered low resolution GPS modes. In obstacle avoidance, the system performed reasonably well. However, with only a bumper-forward terrain map, obstacles disappeared as they fell behind the vehicle bumper. Passing within 20cm of an obstacle on the inside of an avoidance turn, the vehicle could collide with obstacles, now off the map, on the return to the path.

5.4.4 Performance

Due to computing power limitations the field of robotics has traditionally favored light weight implementations, such as IPC/IPT, over heavy weight, computationally demanding implementations such as CORBA. Performance tests, conducted on the Raptor, revealed that this processing power axiom no longer applies. Table 4 shows percentage of the CPU consumed by the top 5 Raptor processes.

Table 4. Percent CPU and Percent Memory consumed by top 5 Processes.

#	Process	Update Rate	CPU	Mem
1	Laser	26.6 ms	0.7%	1.2%
2	Stereo	500 ms	4.0%	5.5%
3	Model Service	60 ms	0.3%	1.5%
4	Terrain Map	Events 1,2,3	20.0%	1.5%
5	Traverse Map	500 ms	1.5%	1.1%

Processes 1, 3, 4 and 5 run on a quad Intel Xeon@3.0 GHz with 2GB of RAM while process 2 runs on a Pentium IV@3.2 GHz with 1 GB of RAM. A 100 Mbit ethernet connects the two computers.

5.4.5 Problems and Failure Modes

DRDC encountered one pressing problem with the TAO software underlying MIRO. When software component A published events to consumers B and C, if either B or C's execution interval exceeds A's event period, the event delivery mechanism would hang. With no new events available the vehicle would continue on its last commanded velocity without obstacle avoidance or navigation services. The addition of a watchdog process partially allieviated this problem by monitoring the delivery of events in the system and taking corrective action if required.

6. Conclusions

DRDC required a modular, extensible, flexible and scalable framework to support its unmanned vehicle program. The research and development at DRDC occurs over long time horizons, and this framework must support diverse platforms types in ground, air and marine variations and the activities of numerous labs in their specialized fields of expertise. The framework also must allow the appropriate architecture to emerge from the specific system requirement. Researchers at DRDC-Suffield conducted an in-depth review of middleware toolkits and architectures and concluded the MIRO framework satisfied DRDC's research and development requirements. MIRO uses the capabilities of CORBA to create a framework which implements the set of components where each component has a clearly defined interface that allows it to share data and methods with other components. While the MIRO framework was originally developed for soccer playing robots, its modular, flexible, scalable and extensible design allowed it to be easily and quickly extended. DRDC has extended MIRO to serve the needs of outdoor UGVs. This involved implementing components that encapsulated the specific functionality required by the target UGV and integrating these new components into the existing MIRO structure. DRDC's experiences with MIRO and CORBA has led the DRDC research group to conclude that this framework and middleware pair implements infrastructure services that will meet both our current and future research needs.

7. References

Albus, J. (1997). Drcs: reference model architecture for DEMO III. NISTIR 5994, National Institute of Standards and Technology, Gaithersburg, MD.

Albus J., Lumia R., and McCain H. (1987). Hierarchical control of intelligent machines applied to space station telerobots.

Bolton, F. (2002). *Pure CORBA: A code intensive premium reference.* SAMS.

Bonasso, R., Firby, R., Gat, E., Kortenkamp, D., Miller, D., and Slack, M. (1997). Experiences with and architecture for intelligent, reactive agents. Journal of Experimental and Theoretical Artificial Intelligence, 9(2):237 256.

Brooks, A., Kaupp, T., Makarenko, A., Oreback, A., and Williams, S. (2005). Towards component-based

robotics. In IEEE/RSJ Int. Conf. on Intelligent Robots and Systems.

Broten, G. , Monckton S., Giesbrecht J., Verret S., Collier J., Digney B., Towards Distributed Intelligence, Technical Report DRDC Suffield TR 2003-287, December 2004, Defence R&D Canada – Suffield.

Broten, G. , Erickson D., Giesbrecht J., Monckton S., Verret S., Engineering Review of ANCAEUS/AVATAR, Technical Report DRDC Suffield TR 2003-167, December 2003, Defence R&D Canada – Suffield.

Cote, C., Letourneau, D., Valin, F. M. J.-M., Brosseau, Y., Raievsky, C., Lemay, M., and Tran, V. (2004). Code reusability tools for programming mobile robots. In IROS. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems.

D. Schmidt, D. L. and Mungee, S. (1998). The design and performance of real-time object request brokers. Computer Communications, 21:294 324.

E. Gamma, R. Helm, R. J. and Vlissides, J. (1994). Design Patterns: Elements of resuable Object-Oriented Software. Addison-Wesley.

Enderle, S., Utz, H., Sablatnog, S., Simon, S., Kraetzschmar, G., and Palm, G. (2001). MIRO - middleware for autonomous mobile robots. Int. Federation of Automatic Control.

Firby, R. (1989). Adaptive Execution in Complex Dynamic Worlds. PhD thesis, Yale Univ., Dept. of Computer Science.

G. Pardo-Castellote, S. S. and Hamilton, M. (1999). Ndds: The real-time publish-subscribe middleware. Real-Time Innovations, Inc.

Gerkey, B., Vaughan, R. T., and Howard, A. (2003). The player/stage project: Tools for multi-robot and distributed sensor systems. Proc. of the 11th Int. Conf. on Advanced Robotics, pages 317 323.

Ipt: An object oriented toolkit for interprocess communication. Technical Report CMU-RI-TR-96-07, Robotics Institute, Carnegie Mellon Univ., Pittsburgh, PA.

Gowdy, J. (2000a). Emergent Architectures: A Case Study for Outdoor Mobile Robots. PhD thesis, Carnegie Mellon Univ..

Gowdy, J. (2000b). A qualatative comparision of interprocess communications toolkits for robotics. Technical Report CMU-RI-TR- 00-16, Carnegie Mellon Univ..

Henning, M. and Vinoski, S. (1999). *Advanced CORBA Programming with C++*. Addison-Wesley.

Huston, S., Johnson, J., and Syyid, U. (2004). *The ACE Programme' s Guide*. Addison-Wesley.

J. Michaloski, F. Proctor, W. S. (2000). The neutral message language: A model and method for message passing in heterogeneous environments. In Proc. of the World Automation Conf., Maui, Hawaii.

JAUS-WG, The Joint Architecture for Unmanned Systems, Reference Archictecture Specification RA V3.2 Parts 1-3, August 2004. http://www.jauswg.org/baseline/refarch.html Accessed: 2005-07-15.

Johnson, R. (2004). Frameworks. http://st-www.cs.uiuc.edu/users/johnson/frameworks.html#Papers. Accessed 2005-06-29

Koenig, S., Goodwin, R., and Simmons, R. (1998). Xavier: A robot architecture based on partially observable markov decision process models. Artificial Intelligence and Mobile Robots: Case Studies of Successful Robot Systems.

Konolige, K. and Myers, K. (1998). The saphira architecture for autonomous mobile robots. Artificial Intelligence and Mobile Robots: Case Studies of Successful Robot Systems.

Oreback, M. L. A. and Christensen, H. (2000). Berra: A research architecture for service robots. In Int. Conf. on Robotics and Automation.

Pedersen, J. . (1998). Robust communications for high bandwidth real-time systems. Technical Report CMU-RI-TR-98-13, Carnegie Mellon Univ..

Powers, I. N. R. and Birchfield, S. (1995). Dervish: An office navigation robot. AI Magazine, 16(2):53 60.

Roy, M. M. N. and Thrun, S. (2003). Perspectives on standardization in mobile robot programming: The carnegie mellon navigation (CARMEN) toolkit. In Proc. of the Conf. on Intelligent Robots and Systems (IROS).

Schildt, H. (1994). *C++ from the Ground Up*. Osborne McGraw-Hill, 2600 Tenth Street, Berkeley, California 9470, USA.

Schmidt, D. and Huston, S. (2002). *C++ Network Programming Volume 1*. Addison-Wesley.

Schmidt, D. and Kuhns, F. (2000). An overview of the real-time CORBA specification. IEEE Computer special issue on Object-Oriented Real-time Distrubuted Computing.

Simmons, R. (1991). The interprocess communications (ipc) system. http://www2.cs.cmu.edu/afs/cs/project/TCA/www/ipc/ipc.html. Accessed: 2005-06-29

Utz, H., Sablatnog, S., Enderle, S., and Kraetzschmar, G. (2002). MIRO - middleware for mobile robot applications. IEEE Transactions on Robotics and Automation.

Vaughan, R. T., Gerkey, B., and Howard, A. (2003). On device abstractions for portable, reusable robot code. Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pages 2121 2427.

Wills, L., Kannan, S., Sander, S., Guler, M., Heck, B., Prasad, J., Schrage, D., and Vachtsevanos, G. (2001). An open platform for reconfigurable control. IEEE Control Systems Magazine.