A Real-Time Software Framework for Indoor Navigation

Frederic Pont and Roland Siegwart
Ecole Polytechnique Fédérale de Lausanne (EPFL)
Autonomous Systems Laboratory (ASL)
CH-1015 Lausanne, Switzerland
{*frederic.pont, roland.siegwart*}*@epfl.ch*

***Abstract*— We introduce an initial implementation of a real-time component-based software framework for autonomous mobile robots. We argue that real-world autonomous mobile robots shall be controlled by self-contained software systems able to meet hard timing constraints. The proposed solution empowers specialized roboticists to contribute software components that can be integrated into complete real-time systems. The framework also facilitates robotic software components reuse and portability across hardware platforms. Based on an indoor navigation case study we evaluate the advantages and the limitations of the framework in terms of ease of use, modularity and real-time capabilities.**

***Index Terms*— Real-time, software framework, component-based software, indoor navigation.**

I. Introduction

With the increasing complexity of the missions to be performed by autonomous mobile robots (e.g. space exploration), a growing number of disciplines become involved in the conception of a complete system, from mechanical and electronics engineering, to computer and even cognitive sciences. This complexity results in an augmenting amount of software to be produced to control robots, with contributions from a number of specialized roboticists pursuing different goals and with varying software engineering skills.

Architectures for autonomous robots have been studied for many years, describing how software systems could be organized [1], [2]. Nowadays, a hybrid solution combining reactive behaviors with limited knowledge of the world and higher level reasoning is widely accepted [3]–[5]. However, there is no standardized way of implementing this architecture to build a complete software system. As a result, software systems for autonomous robots are usually written from scratch and not built from existing pieces of software, leading to a waste of time and resources. Indeed, students, researchers and developers spend a large amount of their time solving software implementation and integration problems instead of focusing on their specific areas of interest to make valuable contributions. Therefore, new solutions and frameworks shall be provided, so that implemented software components can be reused for building complete systems. Moreover, components that comply to a specific model can easily be replaced by other implementations providing similar services in a system, and performance comparison is made possible.

The majority of current robotic research platforms rely on an off-board infrastructure, using autonomous robots as mobile sensors. If this approach has many advantages for carrying out efficient research in individual areas like localization, mapping or path planning, we argue that real-world autonomous mobile robots must be self-contained and able to meet timing constraints. Indeed, autonomy and mobility ask for the ability to perform missions in unknown environments where supporting infrastructure for off-board processing is not always available. Moreover, the limited processing power usually available on embedded systems and the critical safety requirements that apply to autonomous mobile robots performing missions in the real-world, for example when interaction with humans is involved, call for the ability to meet strict timing constraints.

A. Related Work

Recently, many research projects started tackling these software reuse, integration and implementation problems. Among them, PlayerStage [6], [7] is widely used. PlayerStage provides a network server for robot control over IP. It is designed for off-board control and is therefore not suitable for self-contained autonomous systems, but the proposed hardware abstraction based on devices, drivers and interfaces is promising [7]. ORCA-Robotics [8]–[10] promotes a component-based software engineering [11] approach for robotics, where software systems can be composed by mixing existing components with custom in-house developed components. A number of open-source components are already available.

Moreover, a number of other projects [12]–[14] also investigate how robotic software systems can be improved, and other focus on how modern software engineering techniques can be applied to embedded systems in general [15].

B. Approach

In order to be widely accepted by the robotic community, a solution for coping with system complexity, integration problems and software reuse for autonomous mobile robots should at least fulfil the following requirements:

- Embeddable (self-contained) for autonomy

- Modular for independent development of components
- Portable across robotic platforms
- Real-time for reliability and safety
- Open-source for easy component sharing and reuse

All existing solutions and frameworks tackle some of the above listed requirements, but most lack support for hard real-time constraints and focus on distributed systems or off-board processing. As listed above and mentioned in [16], a robust, reliable and safe software system for autonomous mobile robots must be able to meet timing constraints.

In this paper, we present an initial implementation of a real-time component-based software framework for self-contained autonomous mobile robots. Based on an indoor navigation case study, we evaluate the advantages and limitations of this framework. The initial implementation focuses on specialized software components generation and integration into a complete real-time embedded system. Ultimately, the framework will also provide real-time constraints checking and adaptation to events such as missed deadlines.

This paper is organized as follows. Section II introduces the real-time software framework and section III describes a case study, in which the framework is used to build an embedded software system for indoor navigation with real-time constraints. In section IV, we evaluate the advantages and the limitations of the framework and finally, we outline future research orientations and present some concluding remarks in section V.

II. A Real-Time Software Framework

The proposed real-time capable component-based software framework for autonomous mobile robots is composed of a real-time operating system (RTOS), a robotic hardware abstraction layer (rHAL) and a component-based software system, as represented on figure 1. It targets self-contained, embedded and real-time software systems for controlling autonomous mobile robots. It aims at empowering roboticists to implement specialized software components that can be used to build complete software systems able to meet timing constraints. It also provides for easier integration of components and takes care of inter-component communication.

The current implementation of the framework is based on RTAI Linux [17] as its underlying RTOS. RTAI Linux is a real-time extension to the regular Linux kernel, and has been selected because it provides the usual advantages of the Linux operating system, along with hard-real time capabilities. This choice limits the programming language to C, to allow for component execution in kernel space when hard real-time is required, but extensions to other programming languages are possible for non real-time components. The robotic hardware abstraction layer is a thin real-time capable software layer ensuring that higher level software components are portable across different robotic platforms. The current implementation supports the tour-guiding robot RoboX [18] and provides

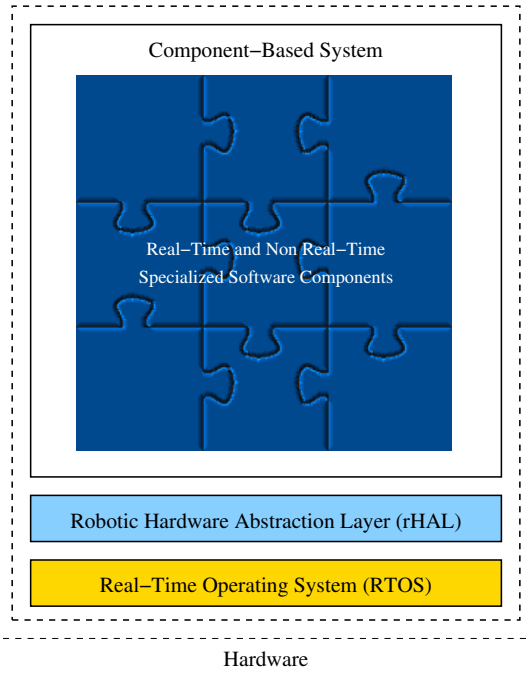

Fig. 1. A hard real-time capable component-based software framework for autonomous mobile robots composed of a real-time operating system (RTOS), a robotic hardware abstraction layer (rHAL), and a component-based software system.

standardized abstractions for the following hardware devices:

- analog inputs and outputs
- digital inputs and outputs
- counters (e.g. encoders)
- serial interfaces

The robotic hardware abstraction layer is implemented as a set of kernel modules for interfacing with real-time components, and as a user-space library for non real-time components, as represented on figure 2. Both user space library and kernel space modules provide the exact same interface to ensure that components accessing hardware through the abstraction layer can be executed transparently in user space or kernel space. For more details about rHAL implementation, see [19]. The component model selected for the proposed software framework is GenoM (**gen**erator **o**f **m**odules) [20], [21], which is an environment for description and implementation of robotic software components that provides the following:

- Component Model: GenoM defines specific interaction between components and composition standards.
- Component Model Implementation: GenoM provides the dedicated set of executable software elements required to support the execution of software components that conform to the model.
- Component Architecture: GenoM defines the internal architecture of software components, and their structure

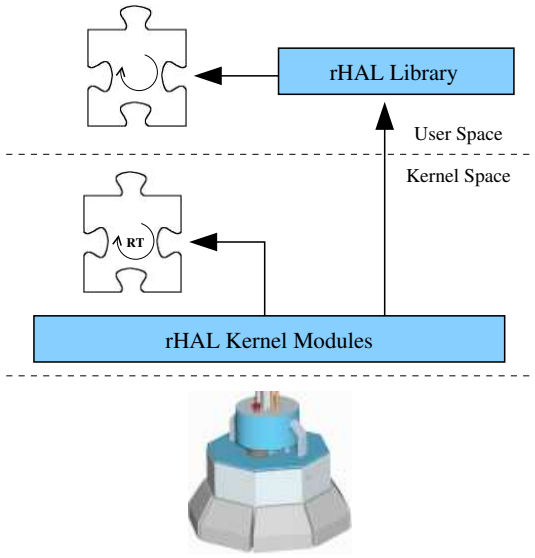

Fig. 2. Software components accessing RoboX hardware (sensors or actuators) through the robotic Hardware Abstraction Layer (rHAL). In RTAI Linux, real-time components (RT) are kernel modules, and therefore access hardware through rHAL kernel modules. Non real-time components use the user-space rHAL library, which acts as a proxy for rHAL kernel modules.

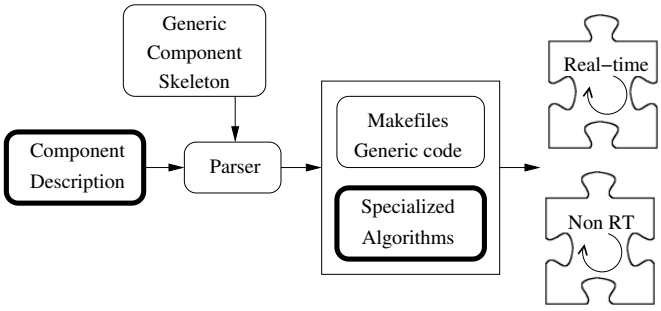

Fig. 3. GenoM-based software component generation: from the component description file to real-time and non real-time executables.

and functioning.

- Component Generation Tools: GenoM provides a set of tools for describing software components and for generating templates.

GenoM-based software component generation is represented on figure 3. The generation process is based on a component description file that is parsed by the tool. Then, templates are generated and can by used by specialized roboticists for algorithms implementation. Finally, binary components are generated, in both real-time and non real-time forms, along with test programs.

III. Indoor Navigation Case Study

In this section, we present an indoor navigation case study to describe and evaluate how a complete embedded and real-time software system for a mobile robot can be developed using the introduced software framework. For the software system developed in this case study, the following requirements have been defined:

1) The complete software system shall be embedded on the autonomous mobile robot, and all processing must be carried out on-board and on-line.
2) Localization of the robot shall be performed using both encoders data and laser range finders.
3) Localization relies on a known map of the environment.
4) Navigation in the environment shall be performed without bumping into humans or fixed obstacles.
5) For safety reasons, obstacle avoidance shall be performed in a periodic hard real-time task.
6) Requests from a human operator will be in the form of: go to x, y, θ. No path planning is required.

In the remainder of this section, we describe the software layers and components that have been used to build the navigation system represented on figure 4.

A. RTOS and rHAL

As introduced in section II, the current implementation of the software framework relies on RTAI Linux, and on a robotic hardware abstraction layer that provides for hardware (sensors and actuators) access through well defined interfaces (see figure 2). For this case study, the robot RoboX has been selected. A PowerPC 750 clocked at 400 MHz is included, as well as two differentially driven wheels, bumpers and two laser range finders.

B. Software Components

The indoor navigation software system is made of a mix of reused and newly developed components. Existing components have been developed at LAAS (Laboratory for Analysis and Architecture of Systems) in Toulouse, France, [22], and have been used for many years on LAAS robots. New software components have been developed in collaboration with specialized roboticists from the Autonomous System Lab at EPFL. The complete system with data flow between components is shown on figure 4.

1) SICK Laser Range Finder (real-time): The SICK component is responsible for managing the two SICK laser range finders placed back to back on RoboX. To ensure no data loss on the two high speed serial interfaces RS422 and to meet the hard real-time obstacle avoidance requirement, this component is a periodic real-time task.

- Input: Data from rHAL serial interfaces.
- Output: Scans in polar and cartesian coordinates.

2) Obstacle Avoidance (real-time): The *obstacle avoidance* component is based on nearness diagram navigation (ND) [23], which performs a high level information extraction and interpretation of the environment, producing appropriate linear angular speeds. To ensure safe indoor navigation in a human crowded environment, this component is a periodic real-time task. Note that the real-time requirement

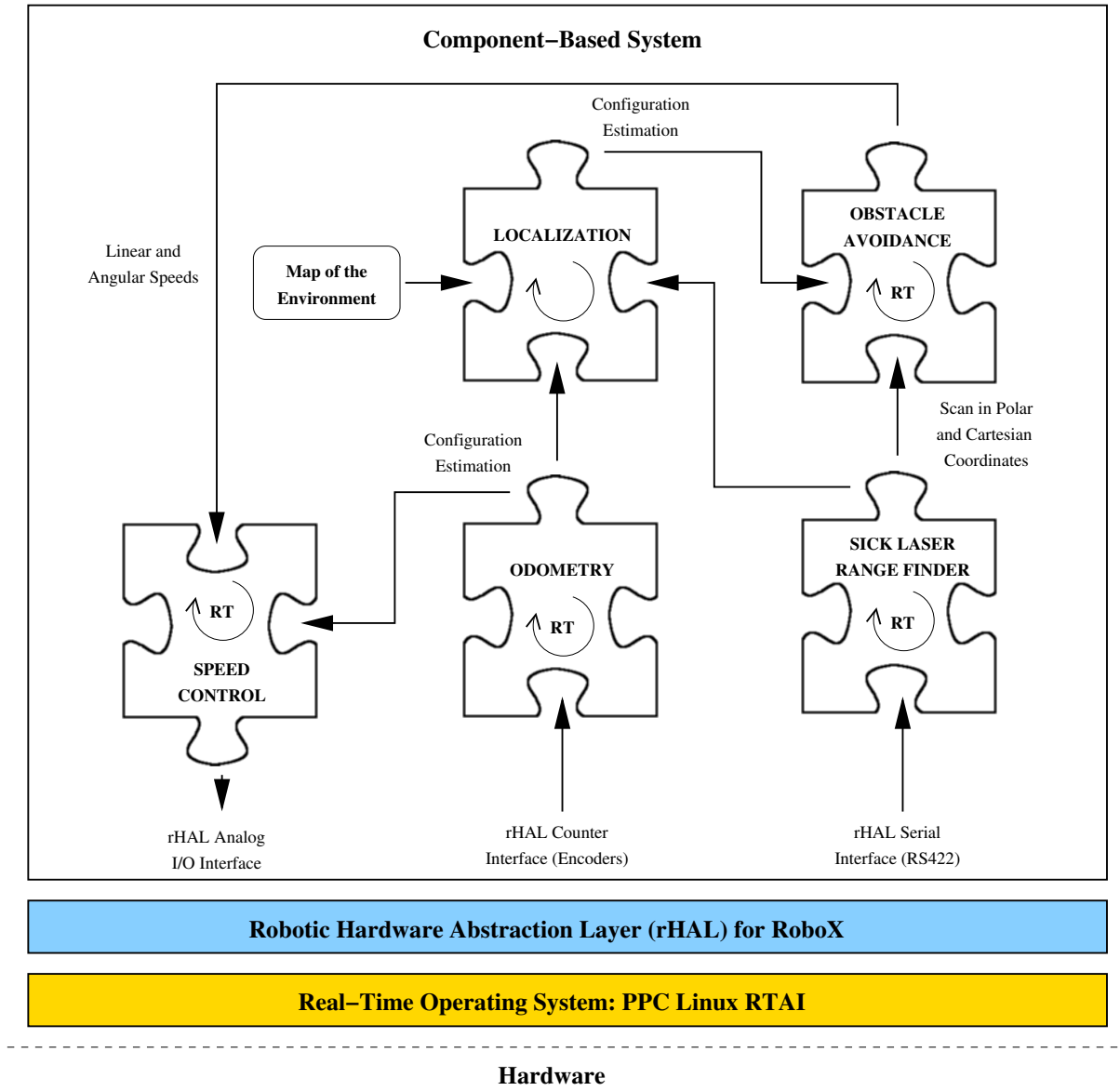

Fig. 4. Real-time software system for indoor navigation. Software components that are executed as real-time tasks are mentioned with RT. Arrows represent data flow between components.

on this component implies that all components producing data used by it must also be hard real-time. In the case study, the suggested linear and angular speed produced by this component are read by the *speed control* component.

- Input: Scan in cartesian coordinates from the *SICK* component.
- Output: Suggested linear and angular speeds.

3) Speed Control (real-time): This component is responsible for translating angular and linear speed requests into a specific speed for each wheel. Maximum speeds and acceleration can be configured. This component is a real-time task to ensure smooth movements of the robot, and to comply with the real-time obstacle avoidance requirement. In the indoor navigation scenario, this component reads linear and angular speeds from the *obstacle avoidance* component. However, when the robot is remote controlled or when obstacle avoidance disabled, linear and angular speeds can be set manually.

- Input: Linear speed, angular speed from *obstacle avoidance*.
- Output: Left and right motor values (rHAL analog outputs).

4) Odometry (real-time): The *odometry* component estimates the current robot configuration (x, y, θ) using only

wheel sensors. It is well known that position estimation using only odometry is rough and that the error grows with time. Therefore, other kind of position estimation must be introduced for advanced navigation. Odometry configuration estimations are used by the *localization* and *speed control* components.

- Input: Left and right encoder values (rHAL counter).
- Output: Robot configuration (x, y, θ, C).

5) Localization: The *localization* component relies on the two SICK laser range finders and on an a-priori known map of the environment to evaluate the current configuration of the mobile robot. This component also relies on configuration estimations provided by the *odometry*. The line extraction method used is based on the split and merge approach [24] and the fusion with the configuration estimation produced by the *odometry* is performed using an Extended Kalman Filter. The configuration estimation produced by this component is used by the *obstacle avoidance* component in the indoor navigation scenario. Note that if this component is disabled or is not able to produce an acceptable estimation, the indoor navigation system can rely on the *odometry* component for a rough configuration estimate. As this component is not critical for security, it is executed as a non real-time task.

- Input: Map of the environment, cartesian scan from *SICK*, configuration from *odometry*.
- Output: Robot configuration (x, y, θ, C).

IV. Evaluation

The indoor navigation system developed in this case study has been tested on the robot RoboX in the Autonomous System Laboratory at EPFL during office hours, and an exemplary navigation path is shown on figure 5, where configuration estimations from the *odometry* and *localization* components are depicted. As path planning was not listed as a requirement, the following intermediate targets have been set manually by a human operator:

1) go to x=79 y=99
2) go to x=73 y=99
3) go to x=112 y=99

Relying only on its embedded software system, the robot was able to navigate autonomously and safely in its environment and to localize itself using laser range finders, correcting the imprecise configuration estimations from odometry. The obstacle avoidance implemented as a hard real-time periodic task ensured that intermediate targets set by the human operator were reached safely. As the goal of the experiment was not to prove the quality or reliability of the navigation system and as all requirements were fulfilled, the case study was considered a success.

In the remainder of this section, we discuss key requirements for a software framework for autonomous mobile robots and we evaluate the proposed solution based on the indoor navigation case study experience.

A. Ease of use

Components generation and development require basic knowledge of the GenoM tools. The learning process is quick for roboticists with basic Unix knowledge. The obligatory usage of the C programming language can be seen as a limitation. However, the templates provided by the framework define the structure of software components and most of the coding is limited to algorithm implementation in simple functions, where the power of object oriented programming languages is not required. The ability to carry out the specialized algorithms debugging phase with non real-time components is very valuable. Real-time and scheduling issues can then be tackled separately, with the help of a system integrator.

B. Modularity

The component-based approach provides for independent component development, and for easier components integration into a complete embedded system. As software components conform to a predefined model, testing, evaluation and integration of externally developed components is possible. Moreover, the complexity of the whole system can be broken down into manageable pieces, as specialized roboticists need only minimal knowledge about other components.

C. Real-time capabilities

The framework ensures that software components can be executed with timing constraints when necessary. Moreover, the framework provides real-time capabilities with minimum impact on specialized roboticists, who can develop and debug their software components without additional constraints. Later, real-time versions of specialized components can be used and real-time testing can be carried out.

V. Conclusion and Outlook

We presented an autonomous indoor navigation system based on an initial implementation of a hard real-time capable component-based software framework for autonomous mobile robots. The navigation case study demonstrated the possibility to develop a complete embedded software system using a mix of existing components and newly developed components. The case study also highlighted the possibility for specialized roboticists to develop software components without having to deal with the overall complexity of the system, even in the case of timing constraints being set on some components. Therefore, the proposed software framework helps roboticists to make valuable contributions in their specialized areas of interest.

The initial implementation of the framework will be extended to include mandatory tools and mechanisms for precise performance monitoring and timing constraints verifications in order to ensure safety and compliance to requirements. Adaptation capabilities to events such as missed

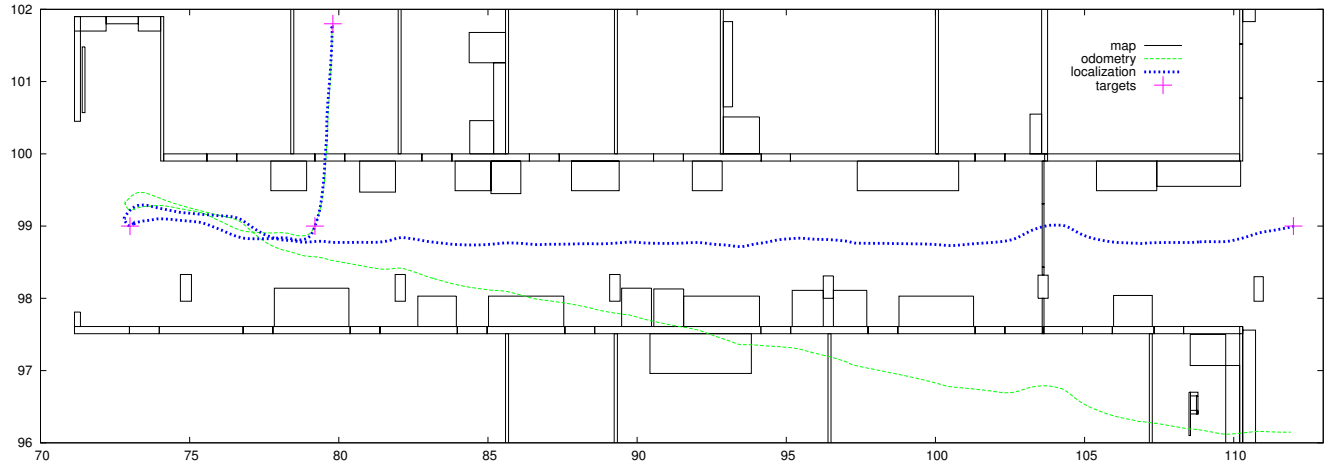

Fig. 5. Successful indoor navigation in the Autonomous System Lab at EPFL with position estimations from the *odometry* (thin line, imprecise estimations) and *localization* (thick line, precise estimations) software components. Initial location and intermediate targets points are also represented.

deadlines will also be investigated. For non real-time components, interfaces with other programming languages and other existing robotic software frameworks are also planned.

ACKNOWLEDGMENT

The work reported in this paper has been supported in part by the IST Project RECSYS (IST-2001-32515). Moreover, the authors would like to thank Agostino Martinelli, Viet Nguyen and Anthony Mallet for their precious contributions to the framework and to the indoor navigation case study.

REFERENCES

[1] R. A. Brooks, "A robust layered control system for a mobile robot," *IEEE Journal of Robotics and Automation*, vol. 2, no. 1, pp. 14– 23, March 1986.

[2] R. C. Arkin, *Behaviour Based Robotics. Intelligent Robots and Autonomous Agents*, MIT Press, 1998.

[3] E. Gat, "On three-layer architectures," *Artificial Intelligence and Mobile Robots. MIT/AAAI Press*, 1997.

[4] K. Konolige and K. Myers, "The saphira architecture for autonomous mmobile robots," Tech. Rep., Artificial Intelligence Center, SRI International, 1996.

[5] R. Volpe, I. Nesnas, T. Estlin, D. Mutz, R. Petras, and H. Das, "The CLARAty architecture for robotic autonomy," in *IEEE Aerospace Conference Proceedings*, Big Sky, Montana, March 2001, pp. 121–132.

[6] B. Gerkey, R. Vaughan, K. Sty, A. Howard, G. Sukhatme, and M. Mataric, "Most valuable player: A robot device server for distributed control," in *Proceedings of the International Conference on Intelligent Robots and Systems*, Wailea, Hawaii, October 2001, pp. 1226–1231.

[7] R. T. Vaughan, B. P. Gerkey, and A. Howard, "On device abstractions for portable, reusable robot code," in *Proceedings of the International Conference on Intelligent Robots and Systems*, Las Vegs, Nevada, October 2003, pp. 2121–2427.

[8] "ORCA-Robotics," http://orca-robotics.sourceforge.net/.

[9] W. Li, H. I. Christensen, A. Oreback, and D. Chen, "An architecture for indoor navigation," in *Proceedings of the International Conference on Robotics and Automation (ICRA)*, April 2004.

[10] A. Oreback, *A Component Framework for Autonomous Mobile Robots*, Ph.D. thesis, KTH Stockholm, 2004.

[11] G. T. Heineman and W. T. Councill, *Component-Based Software Engineering: Putting the Pieces Together*, Addison Wesley Professional, 2001.

[12] "Orocos," http://www.orocos.org/.

[13] C. Cote, D. Letourneau, F. Michaud, J.-M. Valin, Y. Brosseau, C. Raievsky, M. Lemay, and V. Tran, "Code reusability tools for programming mobile robots," in *Proceedings of the Conference on Intelligent Robots and Systems (IROS)*, 2004.

[14] M. Montemerlo, N. Roy, and S. Thrun, "Perspectives on standardization in mobile robot programming: The carnegie mellon navigation (CARMEN) toolkit," in *Proceedings of the Conference on Intelligent Robots and Systems (IROS)*, 2003.

[15] A. Pasetti, *"Software Frameworks and embedded Control Systems"*, vol. 2231 of *Lecture Notes in Computer Science*, Springer-Verlag, 2002.

[16] R. Brega, N. Tomatis, and K.O. Arras, "The need for autonomy and real-time in mobile robotics: A case study of xo/2 and pygmalion," in *Proceedings of the Conference on Intelligent Robots and Systems (IROS)*, 2000.

[17] "RTAI: Real-time application interface," http://www.rtai.org.

[18] "RoboX," http://robotics.epfl.ch.

[19] F. Pont and R. Siegwart, "Towards improving robotic software reusability without losing real-time capabilities.," in *ICINCO (2)*, 2004, pp. 291–294.

[20] S. Fleury, M. Herrb, and R. Chatila, "GenoM: a tool for the specification and the implementation of operating modules in a distributed robot architecture," in *Proceedings of the International Conference on Intelligent Robots and Systems*, Genoble, France, September 1997, pp. 842–848.

[21] A. Mallet, S. Fleury, and H. Bruyninckx, "A specification of generic robotics software components: future evolutions of genom in the orocos context," in *International Conference on Intelligent Robotics and Systems*, Lausanne (Switzerland), Oct. 2002, IEEE.

[22] "Laas open software for autonomous systems," http://softs.laas.fr/openrobots/.

[23] J. Minguez and L. Montano, "Nearness diagram navigation (nd): A new real time collision avoidance approach for holonomic and no holonomic mobile robots," 2000.

[24] T. Pavlidis and S. Horowitz, "Segmentation of planar curves," *IEEE Transactions on Computers*, vol. C-23, no. 8, pp. 860–870, 1974.