Spartacus Attending the 2005 AAAI Conference

F. Michaud (contact author), D. Létourneau, C. Côté, Y. Brosseau, J.-M. Valin, É. Beaudry, C. Raïevsky, A. Ponchon, P. Moisan, P. Lepage, Y. Morin, F. Gagnon, P. Giguère, M.-A. Roux, S. Caron, P. Frenette, F. Kabanza

LABORIUS - Research Laboratory on Mobile Robotics and Intelligent Systems

Department of Electrical Engineering and Computer Engineering

Université de Sherbrooke, Sherbrooke (Québec, Canada) J1K 2R1

http://www.gel.usherbrooke.ca/laborius

Email: laborius-challenge@listes.USherbrooke.ca

Abstract

Spartacus is the name of our robot entry at the 2005 AAAI Mobile Robot Challenge, which consists of making a robot attend the National Conference on Artificial Intelligence. Designing robots that are capable of interacting with humans in real life settings can be considered the ultimate challenge when it comes to intelligent autonomous systems. One key issue is the integration of multiple modalities (e.g., mobility, physical structure, navigation, vision, audition, dialogue, reasoning) into a coherent implementation. Such integration increases the complexity and the diversity of interactions the robot can have, as of analysis and monitoring of such increased capabilities. This paper reports on our solutions and findings resulting from the hardware, software and computation integration work on Spartacus, along with future perspectives regarding this initiative.

I. INTRODUCTION

Introduced in 1999, the AAAI Mobile Robot Challenge (or simply the AAAI Challenge) consists of having a robot start at the entrance of the conference site, find the registration desk, register, perform volunteer duties (e.g., guard an area) and give a presentation [Maxwell *et al.*2004]. The long-term objective is to have robots participate just like humans attending the conference.

We became interested in taking on this challenge because it deals with the fundamental integration challenge of making an autonomous mobile robot operate in natural settings. Table I presents the characteristics of all the AAAI Challenge entries since the event started in 1999 [Meeden *et al.*2000], [Schultz2001], [Yanco & Balch2003], [Kuipers & Stroupe2003], [Maxwell *et al.*2004], [Smart *et al.*2005]. Progress can be observed in the level of integration demonstrated by the entries. For instance, our 2000 entry, a Pioneer 2 robot named Lolitta, used sonars as proximity sensors, navigated in the environment by reading written letters and symbols using a pan-tilt-zoom (PTZ) camera, interacted with people through a touch-screen interface, displayed a graphical face to express the robot's emotional state, determined what to do next using a finite-state machine (FSM), recharged itself when needed, and generated a HTML report of its experience [Michaud *et al.*2001]. EMIB (Emotion and Motivation for Intentional selection and configuration of Behavior-producing modules) was the computational architecture used to integrate all the capabilities into a fully autonomous system [Michaud2002]. Grace came in 2003 [Simmons *et al.*2003], mounted on a bigger robotic platform, adding vocal commands, map navigation, registration-line detection, natural language (NL) understanding. With such a diverse set of capabilities, software integration became a predominant issue with Grace. The different computational modules had to be manually started at the appropriate time during the different steps of the challenge. Using also a B21 platform, Lewis [Smart *et al.*2003] presented in 2004 an integrated implementation for the Challenge's tasks, using CMU's robot navigation toolkit (CARMEN [Montemerlo, Roy, & Thrun2003]) for path planning and localization. This robot had previously demonstrated the use of a framing algorithm, allowing it to take pictures of people in open settings [Byers *et al.*2003].

Reported difficulties regarding these entries are summarized by the following: [Gockley *et al.*2004], [Smart *et al.*2003], [Maxwell *et al.*2004], [Simmons *et al.*2003]

TABLE I

AAAI CHALLENGE ENTRIES

NAME	HARDWARE	CHARACTERISTICS	DEMONSTRATION
FOOF 1999 USC	Pioneer platform; PTZ color camera; Arrow support (contact switches)	Behavior-based navigation; Direction pointing using a fluorescent orange arrow	Navigation without a map, seeking help from humans
OB2K 1999 CMU	Scout platform, Speech synthesis; Screen-based GUI with NL input	Three-tiered architecture (parser, sequencer, skills); Goal state defined from NL	Navigation with or without a map, seeking help from humans
CEREBUS 2000 Northwestern U.	Magellan platform with camera; Speech synthesis; Screen-based interface	Behavior-based navigation; NL understanding and response	Presentation
Lolitta Hall 2000 U. Sherbrooke	Pioneer platform; PTZ color camera; Touchscreen interface Charging station	Hybrid architecture (EMIB); Navigation by reading symbols; FSM scheduling; Interaction through menus; Animated face display	Integrated version of the entire Challenge; Sign recognition; Presentation using HTML reports; Guard a room
CoWorker 2000, iRobot	CoWorker platform; PT color camera; Bidirectional audio	Internet teleoperation; waypoint navigation	Navigation portion of the entire Challenge
Leo 2000, MIT	B21 platform; Laser range finder	Large-scale concurrent mapping and localization	“Hands-off” autonomous locomotion
Grace & George 2002, 2003 CMU, NRL, Metrica, Northwestern, Swarthmore	B21 platforms, Laser range finder; Flat panel display, PTZ color camera; Stereo vision on PT unit; 1 wireless microphone headset; 2 onboard computers (LINUX); 2 laptop computers (Windows)	Direction using audio input; Map-based navigation; Line detection; Gesture recognition; People/Color object tracking; Message reading; Animated face display; ViaVoice; OpenGL Festival; NAUTILUS NL	Controlled execution of the entire Challenge; Software integration using IPC, Navigation using vocal commands and CARMEN; Error recovery; Natural language interaction
Lewis 2003, 2004 Washington U.	B21 platform; Laser range finder; Directed perception PT; unit; 2 color cameras Touchscreen interface; Keyboard	FSM scheduling, Map-based navigation; Color object tracking; Message reading; Festival	Integrated version of the entire Challenge; Navigation using CARMEN; Sign recognition; Preprogrammed speech scripts
Spartacus 2005 U. Sherbrooke	Custom-designed platform; Laser range finder; PTZ color camera; Touchscreen interface; Electronic display; Microphone array; Business card dispenser; 1 onboard computer; 2 laptop computers; 1 external laptop	Hybrid architecture (MBA); Message reading; Sound localization; tracking and separation; Planning and scheduling; Nuance; Festival	Integrated version of the entire Challenge; Navigation using CARMEN; Mapping using pmap; Mapping of visual and audio messages; Temporal task sequencing

- Robust integration of different software packages and intelligent decision-making capabilities;
- Natural interaction modalities in real life settings (speech, gestures recognition);
- Adaptation to environmental changes for localization;
- Monitoring and reporting decisions made by the robot.

The need to address these problems motivates our work on Spartacus, our 2005 AAAI Challenge entry described in this paper. Section II presents Spartacus' robotic platform and computational architecture. Section III describes software integration tools and capabilities implemented on the robot. Section IV presents results and observations in terms of integrated capabilities and performances, followed by Section V with a discussion on important issues that we want to address in the near future. Section VI concludes the paper.

II. Spartacus the Autonomous Robot

Shown in Figure 1, Spartacus is a custom-built robotic platform designed for high-level interaction with people in real life settings. It is a differential steering wheeled platform with a humanoid-like upper torso. Spartacus is equipped with a SICK LMS200 laser range finder, a Sony SNC-RZ30N 25X PTZ color camera, an array of eight microphones placed on the robot's body, a touchscreen interface, an audio amplifier and speakers, a business card dispenser and a LEDs electronic display. The robot also carries its own chargers so that it can be plugged directly into a regular electric outlet. Low-level interfaces of most of these components are done following a distributed approach, using different microcontroller subsystems that communicate with each other through a shared CAN 2.0B 1 Mbps bus [Michaud *et al.*2005]. This approach facilitates debugging and extensions to the platform. High-level processing is carried out using an embedded Mini-ITX computer (Pentium M 1.7 GHz). The Mini-ITX computer is connected to the low-level microcontrollers through a CAN bus device, to the laser range finder through a serial port and to the serial controller of the PTZ camera. Two laptop computers (Pentium M 1.6 GHz) are also installed on the platform. One is equipped with a RME Hammerfal DSP Multiface sound card using eight analog inputs to simultaneously sample signals coming from the microphone array. It is also connected to the audio amplifier and speakers using the audio output port. The other laptop does video processing and is connected to the camera through a 100Mbps Ethernet link. Communication between the three on-board computers is accomplished

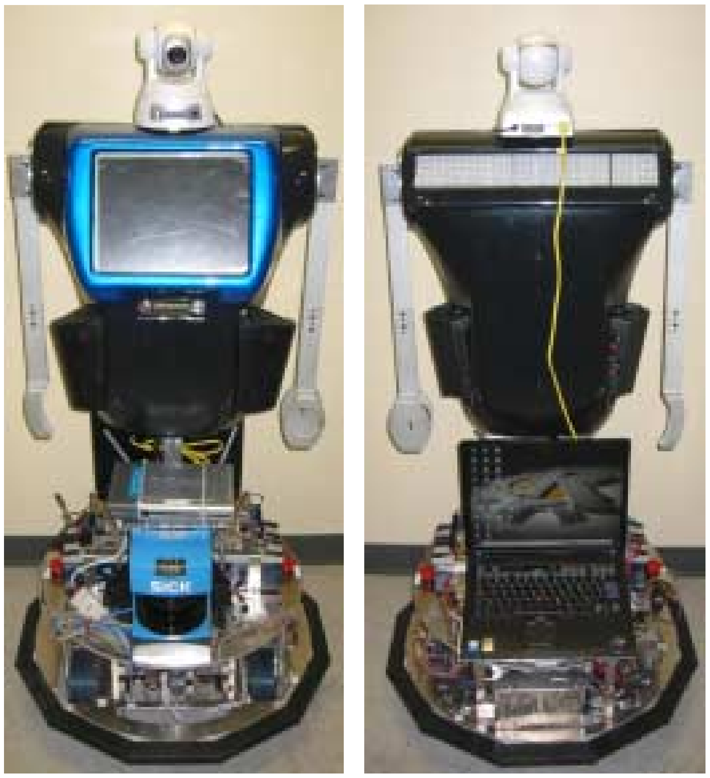


Fig. 1. Spartacus (front view, back view).

with a 100Mbps Ethernet link. Communication with external computers can be achieved using the 802.11g wireless technology, giving the ability to easily add remote processing power or capabilities if required. All computers are running Debian GNU Linux.

The computational architecture we are developing for Spartacus' decision-making capabilities is based on the notion of motivated selection of behavior-producing modules. We refer to it as MBA, for Motivated Behavioral Architecture. The architecture contains different motivational sources derived from perceptual influences, pre-determined scenarios, navigation algorithms, a planning algorithm or other types of reasoning algorithms. Our intention in distinguishing these influences through different motivational sources is to simplify programming of the robot's intentions in accomplishing various tasks.

Figure 2 represents MBA. It is composed of three principal elements:

1) Behavior-producing modules (BPMs) define how particular percepts and conditions influ-

ence the control of the robot's actuators. They can be typical behavior-based reactive controllers with or without internal states, goal-oriented behaviors or other types of behaviors. The actual use of a BPM is determined by the arbitration scheme (realized through BPM Arbitration, which can be priority-based, data fusion, action selection or defuzzification, depending on the implementation) and the BPM's activation conditions, as derived by the BPM Selection module.

2) Motivational sources (or Motivations) recommend the use or the inhibition of tasks to be accomplished by the robot. Motivational sources are categorized as either instinctual, rational or emotional. Instinctual motivations provide basic operation of the robot using simple rules. Rational motivations are more related to cognitive processes, such as navigation and planning. Emotional motivations monitor conflictual or transitional situations between tasks. They will also monitor changes in commitments the robot establishes with other agents, humans or robots, in its environment. Motivations can be derived from percepts, results and states of current goals, and by monitoring tasks or BPMs (using the Exploitations link). Behavior exploitation indicates when a BPM is effectively used to control the robot's actuators, and serves as an abstraction of the robot's interactions within the environment. An active behavior may or may not be used to control the robot, depending on the sensory conditions it monitors and the arbitration mechanism used to coordinate the robot's behaviors. An active behavior is exploited only when it provides commands that actually control the robot.
3) Dynamic Task Workspace (DTW) organizes tasks in a tree-like structure according to their interdependences, from high-level/abstract tasks (e.g., deliver message), to primitive/BPM-related tasks (e.g., avoid obstacles). Through the DTW, motivations exchange information asynchronously on how to activate, configure and monitor BPMs. Motivations can add and modify tasks by submitting modification requests (m), queries (q) or subscribe to events (e) regarding the task's status.

Motivations are kept generic and independent from each other and from the BPMs, through tasks posted in the DTW. For instance, one instinctual motivational source may monitor the robot's energy level to issue a recharging task in the DTW, which activates a behavior that would make the robot detect and dock in a charging station. Meanwhile, if the robot knows

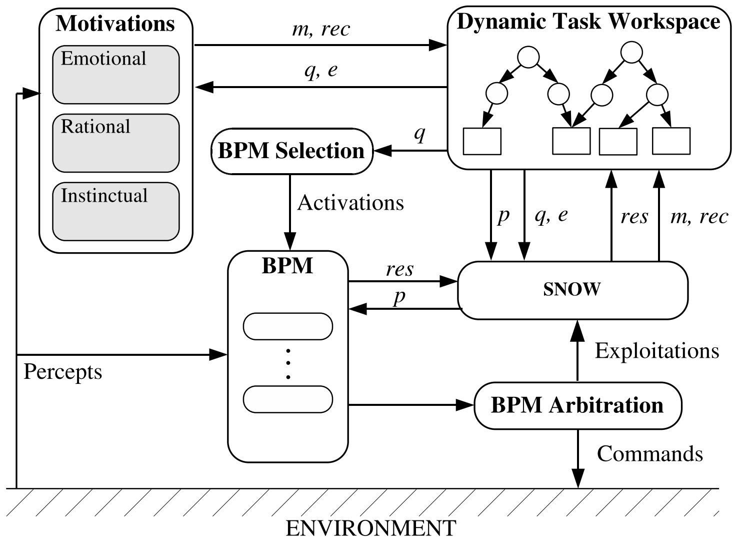


Fig. 2. MBA computational architecture.

where it is and can determine a path to a nearby charging station, a path planning rational motivation can add a subtask of navigating to this position. With multiple tasks being issued by the motivational sources, the BPM Selection module determines which BPMs are to be activated according to recommendations (*rec*) made by motivational sources. A recommendation can either be negative, neutral or positive, or take on real values within this range regarding the desirability of the robot to accomplish specific tasks. The activation values (Activations) reflect the resulting robot's intentions derived from interactions between the motivational sources through the DTW.

MBA is a generalization of EMIB [Michaud2002]. The System Know-How (SNOW) module, not present in EMIB, acts as an adapter between BPMs and DTW, making it is possible to decouple task representation contained in the DTW from BPMs. Consequently, both of them can use their own domain representations and concepts without having to share information directly. For instance, the SNOW contains the necessary knowledge to define and communicate tasks parameters (*p*) and behaviors results (*res*) between BPMs and DTW. This facilitates the addition of new motivational sources and tasks to the decision processes, with minimal changes to the software implementation of the computational architecture. Motivations and tasks are kept independent of the hardware implementation, to facilitate reuse from one robotic platform to another.

III. Software Integration on Spartacus

Our 2005 implementation attempted to have Spartacus participate in all the AAAI Mobile Robot events, to ensure the validity of the integration work in multiple contexts. The following capabilities were integrated:

- **Autonomous Navigation**. When placed at the entrance of the convention center, the robot autonomously find its way to the registration desk by wandering and avoiding obstacles, searching for information regarding the location of the registration desk, following people talking or going towards communicated directions. Once registered, the robot can use a map of the convention center. The two navigation tools used are CARMEN and pmap. CARMEN, the Carnegie Mellon navigation toolkit [Montemerlo, Roy, & Thrun2003], is a software package for laser-based autonomous navigation using previously generated maps. The pmap package[1] provides libraries and utilities for laser-based mapping in 2D environments to produce high-quality occupancy grid maps. These maps were then converted into CARMEN's format.
- **Vision Processing**. Extracting useful information in real time from images taken by the onboard camera enhances interaction with people and the environment. For instance, the robot could benefit from reading various written messages in real life settings, messages that can provide localization information (e.g., room numbers, places) or identity information (e.g., reading name badges). Object recognition and tracking algorithms also make it possible for the robot to interact with people in the environment. We use two algorithms to implement such capabilities : one that can extract symbols and text from a single color image in real world conditions [Letourneau, Michaud, & Valin2004]; and another one for object recognition [Lienhart & Maydt2002] and tracking to identify and follow regions of interest in the image such as human faces and silhouettes (using the Open CV library). Once identified, these regions can be tracked using color information, as achieved in [Perez *et al.*2002].

[1] http://robotics.usc.edu/˜ahoward/pmap

- **Sound Processing**. Hearing is an important sense for interaction in open settings. Simply using one or two omnidirectional microphones on a robot may not be enough: it seems very difficult to filter out all of the noise generated in public places. Using a microphone array reveals to be a robust solution for the localization, tracking and separation of sound sources. Our approach is capable of simultaneously localizing and tracking up to four sound sources that are in motion over a 3 meters range, in the presence of noise and reverberation [Valin *et al.*2004], [Valin, Michaud, & Rouat2006]. We also developed a method to separate in real-time the sound sources [Valin, Rouat, & Michaud2004] to simultaneously process vocal messages from interlocutors using software packages such as Nuance[2]. We tested Nuance with data generated by our system using speech utterances (i.e., four consecutive digits spoken in English) simultaneously made by three separate speakers in an open environment (not crowded). Recognition accuracy was 84% on average for the three speakers, including 87% accuracy for the front speaker. In the same conditions, human listeners achieved 81% recognition rate when trying to focus on the front speaker alone. Only one of the five listeners was able to achieve the same accuracy as the robot, but again for only one (the front speaker) out of the three speakers.
- **Touchscreen Display**. Various information can be communicated through this device, such as: receiving information from people using a menu interface; displaying graphical information such as a PowerPoint presentation or a map of the environment; requesting the execution of a volunteer task (i.e., deliver a message, guard); and expressing emotional states using a virtual face. We chose to use QT3 for the graphical interface development for ease of use and portability.

Software integration of all these elements is not a simple plug-and-play process. Because available algorithms and robotics software are often not designed to work together due to different objectives and constraints, we designed MARIE (Mobile and Autonomous Robot Integrated Environment)[3] [Cote *et al.*2006], [Letourneau *et al.*2006]. MARIE is a system integration framework used to link multiple software packages. MARIE is oriented towards a rapid prototyping approach to develop and integrate new and existing software for robotic

[2]http://www.nuance.com

[3]http://marie.sourceforge.net

systems. MARIE’s design efforts have been focused on distributed robotics component-based middleware framework development, enhancing reusability of applications and providing tools and programming environments to build integrated and coherent robotics systems. The robotics community still has to explore a great variety of ideas, application areas (each one having its own set of constraints, e.g., space, military, human-robot interaction) and to cope with continuously evolving computing technologies. Consequently, being able to reuse previous implementations and adapt to upcoming robotics standards easily, without major code refactoring, provide longer life cycle for actual and future implementations.

MARIE’s philosophy is based on the creation of reusable software blocks, called components, which implement functionalities by encapsulating existing applications, programming environments or dedicated algorithms. The idea is to configure and interconnect these components through common communication mechanisms (either on the same processing node, or distributed across multiple nodes running similar or different operating systems), to implement the desired functionalities, using software applications and tools available through MARIE.

In MARIE, frameworks and software tools managing and abstracting useful functionalities (e.g., interconnections and communications, threads and processes management functions, distributed system management, data filters and conversions, static and dynamic configurations, event handling) support component creation. Four types of components are used: Application Adapter (AA), Communication Adapter (CA), Application Manager (AM) and Communication Manager (CM). AA’s role is to interface applications into MARIE’s application design framework and make them interact with each other. CA ensures communication compatibility between AAs. For example, it makes it possible to connect an AA providing data at a fixed rate with an AA requiring it asynchronously. Currently available CAs in MARIE are Splitters, Switches, Mailboxes and Shared Maps. A Splitter sends data from one source to multiple destinations without the sender needing to be aware of the receivers. A Switch acts like a multiplexer sending data to the selected output. A Mailbox creates a buffered interface between asynchronous components. A Shared Map is used to share data between multiple components. AMs and CMs are system level components that instantiate and manage components either locally or across distributed processing nodes. They can, for instance, restart a component not responding for a while, or can move components from one processing node to another to avoid CPU overloads.

Although the use of MARIE's frameworks and software tools is highly encouraged to save time and efforts, MARIE is not limited to them. Developers can use the best solution to integrate software applications and interconnect components by having the possibility to extend or adapt existing components and available frameworks. MARIE's underlying philosophy is to complement existing applications, programming environments or software tools, and therefore it is to be used only when required and appropriate.

IV. Results

Figure 3 illustrates Spartacus' implementation of MBA using the capabilities made available through MARIE. BPM Arbitration scheme used is priority-based. Recommendations and BPM Activations are binary values. Four actuators are controlled by BPMs, each of them having one or more associated behaviors:

1) The Motor actuator has eight associated behaviors: Emergency Stop, stopping the robot when abnormal conditions are detected; Rest, stopping the robot from moving; Avoid, making the robot move safely in the environment; Obey, following vocal requests; Dock, stopping the robot while waiting to be connected to a charging station; Goto, directing the robot to a specific location; Follow Audio Localization, making the robot follow an audio source; Follow Wall, making the robot follow a wall (or corridor) when detected, otherwise generating a constant forward velocity.
2) The Camera actuator (PTZ color camera) is controlled by four behaviors: Track Text, centering and zooming on detected text areas in the image; Track Face, centering a perceived face in the image; Track Badge, centering and zooming on a detected name badge in the image; Track Audio, pointing the camera in the direction of an audio source.
3) The Audio actuator (i.e., the sound card output) is associated with the Dialogue behavior.
4) The Monitor actuator is linked to the UserInterface (UI) behavior, using the touchscreen interface to interact with users.

Only instinctual and rational motivations are implemented in this version, with rational motivations having greater priority over instinctual ones in case of conflicts (e.g., conflictual recommendations for the same task). For instinctual motivations, Select Task selects one high-level task when none has yet been prioritized. For instance, between tasks that require the robot to go to a specific location, this motivation selects the task where the location is physically

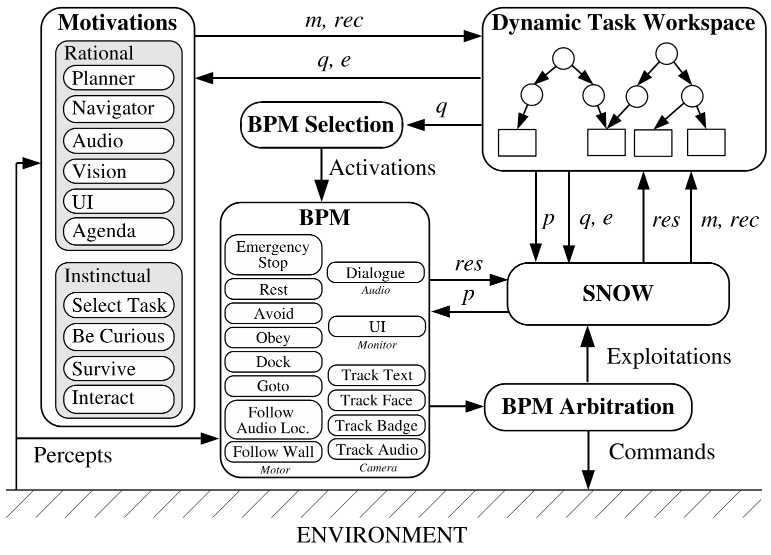


Fig. 3. MBA computational architecture implemented for Spartacus 2005 entry.

closest to the robot. Be Curious motivates the robot to discover and collect information about its environment. Survive urges the robot to maintain its physical integrity by recommending to avoid obstacles. Interact is a process allowing the robot to socialize with people. The other modules are rational motivations. Planner is a motivational source determining which primitive tasks and which sequence of these tasks are necessary to accomplish high-level tasks under temporal constraints and limited capabilities (as defined by BPMs). Our first implementation is a simple reactive planning module that interleaves planning and execution [Beaudry *et al.*2005], like [Haigh & Veloso1998] and [Lemai & Ingrand2004]. Navigator determines the path to a specific location, as required for tasks in the DTW. Audio and Vision motivate the robot to do tasks according to their respective senses (e.g., track badge, localize sound sources). UI is a process allowing the execution of user's requests for tasks. Agenda generates predetermined tasks to accomplish according to the AAAI Mobile Robot Competition context.

The implementation of MBA's computational architecture using MARIE requires 42 components (˜ 50 000 lines of code) composed of 26 AAs, 14 CAs and two external applications (the Audio Server and NUANCE). Except for the two external applications, component interconnections are all sockets-based using Push, Pull and Events dataflow communication mechanisms [Zhao2003] with XML encoding for data representation. The Audio Server and Nuance use their own communication protocols. MARIE's current version is in C++ (˜ 10 000 lines of code) and uses the ACE (Adaptive Communication Environment) library [Schmidt1994] for the low-level operating system functions. MARIE's Application Manager is partially implemented, requiring AAs and CAs to be initialized manually using scripted commands. Also, the Communication Manager is not yet implemented, meaning that component configuration must be set manually. Representation of Spartacus' software architecture is illustrated in Figure 4. Distributing applications across multiple processing nodes was not difficult with MARIE, having chosen network sockets as the communication mechanism. The Vision component uses one on-board laptop computer, Audio components uses the second one, and all other components are executed on Spartacus' on-board Mini-ITX computer. Blocks with similar backgrounds in figure 4 run on the same computer.

In the real robot setup, SpartacusAA combines wheels odometry and gyroscopic (through GyroAA interfacing a gyroscope installed on Spartacus) data, and pushes the result at a fixed rate (200 Hz) to its interconnected component. Laser data is collected by PlayerAA, interfacing the Player library specialized for sensor and actuator abstraction [Vaughan, Gerkey, & Howard2003], supporting the SICK LMS200 laser range finder installed on Spartacus. PlayerAA pushes data at a fixed rate (200 Hz) to connected components. In the simulation setup, odometry and laser data are both collected with PlayerAA, generated either by Stage (2D) or Gazebo (3D) simulators [Vaughan, Gerkey, & Howard2003]. CARMEN Localizer AA and CARMEN Path Planner AA provide path planning and localization capabilities. CARMEN is composed of small processes communicating through a central server. CARMEN's integration was realized by creating an AA that starts several of these processes depending on the required functionality and on data conversion from CARMEN's to MARIE's format.

RobotFlow (RF) and FlowDesigner (FD) programs [Cote *et al.*2004] are two modular data-flow programming environments that facilitate visualization and understanding of the robot's control loops, sensor and actuator processing. They are also appropriate for rapid prototyping since the graphical user interface facilitates the interconnexion of reusable software blocks without having to compile the program every time minor changes are made. They are used to implement Behavior & Arbitration FD AA, handling BPMs and their arbitration. It uses a synchronous pull mechanism to get data coming from different elements such as localization,

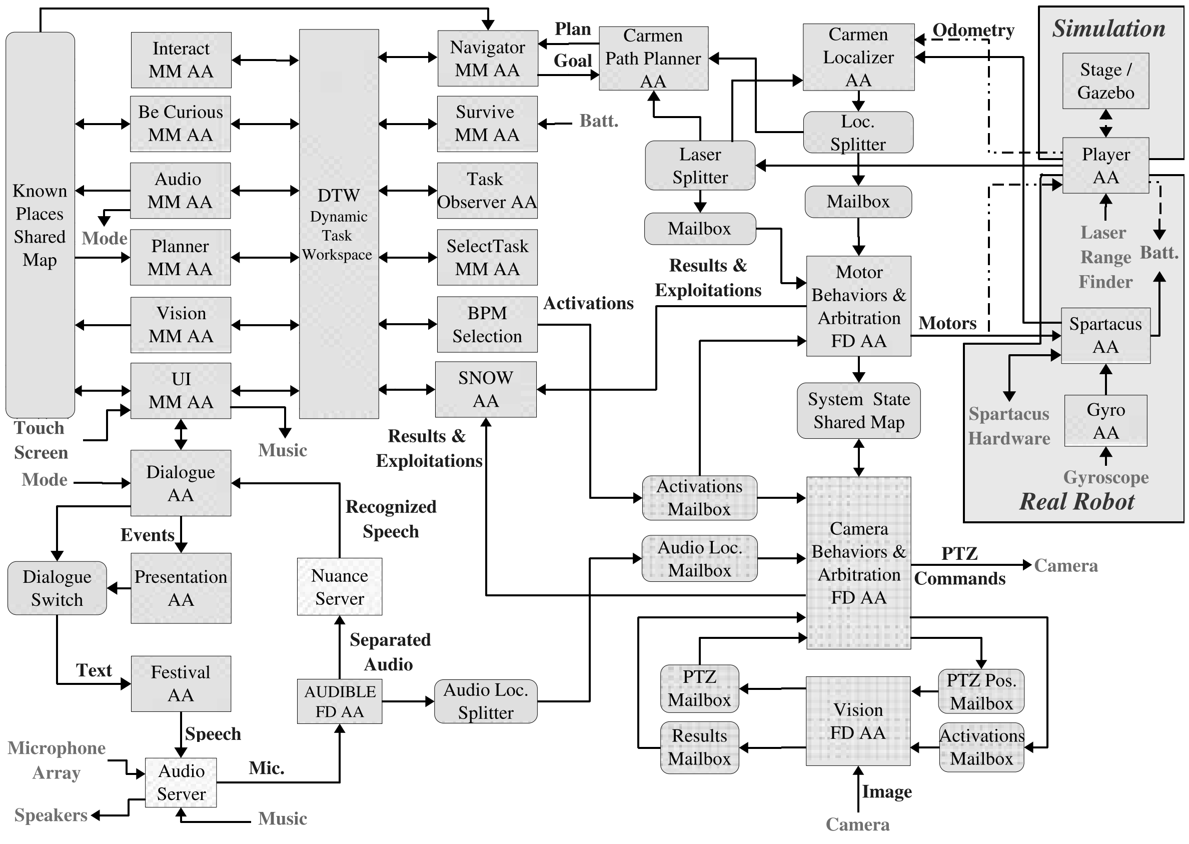


Fig. 4. Representation of Spartacus' software architecture.

path plan, laser, audio localization, dialogue command and system states, requiring the use of Mailbox CA components. Buffering input data, mailboxes allows AAs running at different rates to be interconnected. Behavior & Arbitration FD AA generates motor commands at a fixed rate (5 Hz).

The Audio Server is interfacing the RME Hammerfal DSP Multiface sound card, and Nuance Server is interfacing Nuance. DialogueAA is a stand-alone AA that manages simultaneous conversations with people. This is made possible with the use of AUDIBLE FD AA, interfacing our sound source localization, tracking and separation algorithms implemented with RF/FD and using Spartacus' microphone array. It generates a number of separated audio channels that are sent to Nuance Server and Behavior & Arbitration FD AA. Integrating Nuance in an AA was challenging since it is a proprietary application with a fixed programming interface, and because its execution flow is tightly controlled by Nuance's core application, which is not accessible from the available interface. To solve this problem, we created an independent application that uses a communication protocol already supported by MARIE. Recognized speech data is sent to Dialogue AA, responsible of the human-robot vocal interface. Speech generated by the robot is handled by Festival [Taylor1999]. The Dialogue AA conversation context mode is selected by the Audio MM AA, monitoring the tasks present in the DTW requiring speech interaction.

The global execution of the system is asynchronous, having most of the applications and AAs pushing their results at variable rates, based on the computation length of their algorithms when triggered by new input data. Synchronous execution is realized by having fixed rate sensors readings and actuators commands writings.

Using this implementation, Spartacus did enter all events (Challenge, Scavenger Hunt, Open Interaction) at the 2005 AAAI Mobile Robot Competition. Each event was programmed as a special configuration mode, which could be requested by the experimenter (using speech or from the tactile interface shown in Figure 5).

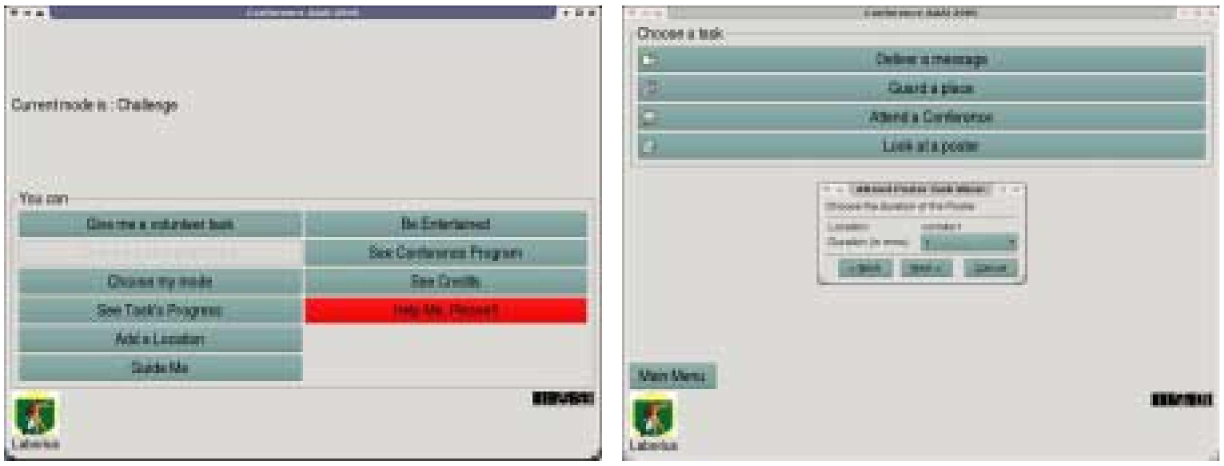


Fig. 5. Touchscreen Interfaces: configuration modes (left), volunteer tasks (right).

Figure 6 are pictures taken during the event. Complete integration was demonstrated, combining map building with localization and path planning, audio tracking of speakers around the robot (positioning the camera in the direction of where the sounds came), speech recognition in “cocktail party effect” conditions, scripted vocal response to specific requests (around 30 different requests), reading the room names (printed on 8.5” $\times$ 11” sheets of paper using Arial and not Times, the font used for the AAAI badges), and dynamic scheduling of tasks set according to temporal constraints. Events such as recognizing a specific sentence or reading a particular message could be memorized and labeled on the map for further reference.

Fig. 6. Pictures of Spartacus at the 2005 AAAI Challenge.

However, to demonstrate Spartacus’ speech recognition capabilities in open settings, we had to sacrifice visual tracking of people. More CPU processing power is required to fully integrate all of our available vision software. Also, speaker volume could not be set to its maximum since a humming sound coming out of the speakers was interfering with two of the nearby microphones.

Complete integration does not mean however that all of the underlying problems are solved, the experimental conditions being extremely challenging:

- Spartacus’ motor drives were set to a secure speed when running on hard surface conditions, but not on carpet (making the robot go slower than expected). We did not want to make any changes to this hardware threshold, in case it would have damaged the motor drives. Eventually, to move on soft carpets and in crowded environments, Spartacus could be designed to use an AZIMUT base for its locomotion. AZIMUT, also demonstrated at the 2005 AAAI Mobile Robot Exhibition, is a symmetrical platform with four independent articulations that can be wheels, legs or tracks, or a combination of these [Michaud *et al.*2005]. By changing the direction of its articulations, AZIMUT is also capable of moving sideways without changing its orientation, making it omnidirectional.

- Redirecting towards one of the five possible elevators in the hotel, as the doors open up, Spartacus did not have enough time to get onboard before the doors would close. Not knowing which door was going to open, and not being able to keep one open long enough to let the robot move in front of it and enter, we finally had to manually operate the robot inside one elevator.
- Trials were conducted in extremelly difficult conditions: registering to a conference does not usually happen right in the midst of a reception, when tables are being installed, in conditions in which it is even difficult for humans to sustain a conversation, fighting the crowd. We should have made Spartacus to be more directive in what it wants to do (e.g., asking people to move away if it has to do something), and not always keep on responding to people's requests.
- Being able to simultaneous extract sound sources and process them with Nuance is CPU intensive. Explaining Spartacus capabilities next to it during demos made Spartacus process long audio streams that sometimes were discarded, and generated inappropriate delays during vocal interactions. Spartacus stops listening only when it speaks so that it does not try to understand itself. This limits the amount of unnecessary processing, but does not allow the robot to understand a request made by someone as it speaks.
- Displaying what the robot said on the touchscreen interface revealed quite useful, as it sometimes was difficult to hear what the robot was saying because of the high-level of noise in the environment. This could be improved by displaying bigger graphical objects and messages.
- Spartacus sometimes stopped moving, not listening to vocal requests, because it was trying to read an unanticipated but potential message in its view. Visual influences were prioritized over audio, and this should be set according to the robot's task. One additional improvement would be to clearly display what the robot is currently doing on its touchscreen interface or its LED electronic display.
- Handling complex situations that may arise when completing the Challenge from start to end requires running the robot for 1 to 1.5 hour (which is approximately Spartacus' entire energetic capability). Trying to demonstrate the capabilities of a planner is therefore difficult

in the limited periods of time allowed for evaluation during the event. For instance, during one trial of 19 minutes, the Planner was invoked only 12 times and took on average 1.259 ms (between 0.15 ms to 12.93 ms) for generating simple plans. These plans were generated because of the addition of new tasks in the DTW (3 times), the addition of new locations on the map (3 times), the occurrence of a message delivery volunteer task (3 times out of 5 because of failure to remain in the delivery location), and replanning caused by earlier achievement of tasks (1 time). No failures were observed because of a planning error, and no tasks were discarded because of anticipated failures. Longuer trials are required to observe and study the performance of the Planner motivation.

- It is also very difficult to come up with quantified results regarding the performance of the robot in the open settings and uncontrolled conditions of the Challenge. To our knowledge, no other entry (listed in Table I) has yet presented such analysis. A way to memorize what happened during a trial, both internally and externally to the robot, is required. This would enable off-line analysis of the performance of the robot, and provide useful information for comparison over trials.

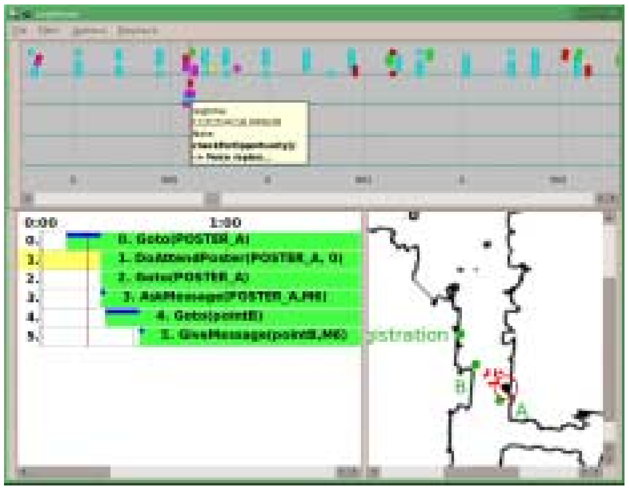
Overall, Spartacus made substantial progress in integrating and demonstrating important capabilities for having an autonomous mobile robot participate to a conference. However, optimization is required to make the integration work more smoothly and to be able to evaluate the global performance of the robot in accomplishing the Challenge.

V. DISCUSSION

Spartacus' first implementation provided interesting insights on software integration challenges in designing autonomous mobile robots. Meeting the integration needs using MARIE's rapid-prototyping approach revealed to be very valuable. MARIE provides the capability of reusing existing programming packages and interconnect them through a system integration framework to benefit from their respective approaches, instead of having to choose only one of them or to reimplement them. This ensures efficient progress in discovering the underlying issues with autonomous reasoning of mobile robots. It also provides a flexible team development framework. At the peak of Spartacus' software development process, eight software developers were working concurrently on the system. Most of them only used Application Adapters to create their components, conducting unit and blackbox testing with pre-configured system setups given by one integrator. The ability to change between simulation and robotic setups with only few system modifications gave us the possibility to do quick simulations and integration tests. Nearly 75% of the system functionalities were validated in simulation and used as is in the real world setup. In both simulated and real setups, configurations of components receiving laser and odometry data were exactly the same, abstracting data sources and benefiting from components modularity and the rapid prototyping approach. Communication protocols and operating system tools for component and application development were added when required. Components were incrementally added to the system as they became available. It took around eight days, spread over a four weeks period, to complete a fully integrated system.

Being able through MARIE to quickly interconnect components to create a complete implementation, without focusing on optimization right away, proved beneficial. Such an exploration strategy gave us the ability to quickly reject software designs or component implementations without investing too much time and effort. For Spartacus, we originally thought that tighter synchronization between components would be necessary to obtain a stable system and support real-time decision-making. For instance, having connected all of Spartacus' components together, we observed that performances were appropriate with the processing power available as long as we did not overload the computers with too many components. Noticing that, we decided to wait before investing time and energy working on component timing constraints, to focus on Spartacus' integration challenges. Our implementation also confirms the importance of having GUI and system management tools in MARIE to build, configure and manage components automatically. Manually configuring and managing the system, with many components executed on multiple processors, is an error-prone and tedious task.

Another tool revealed to be essential during Spartacus' software design process. We were having difficulties tracking decisions made by the system simply by observing its behaviors in the environment or logs from different software components, something that was always possible with simpler implementations. We decided to develop a graphical application to follow on-line or study off-line the decisions made by the robot. This application, named the LogViewer, is partially shown in Figure 7. The upper section of the LogViewer contains a timeline view of DTW (first line) and planner events (second lines), and behaviors' activations and exploitations (not shown). The bottom section shows detailed information according to the position of the horizontal marker on the timeline: a list of DTW's tasks and properties (not shown), active



```
0:00                1:00
0.      0. Goto(POSTER_A)
1.      1. DoAttendPoster(POSTER_A, 0)
2.      2. Goto(POSTER_A)
3.      3. AskMessage(POSTER_A,M6)
4.      4. Goto(pointB)
5.      5. GiveMessage(pointB,M6)
```

Fig. 7. LogViewer: reduced view of the current plan and the map with labeled places.

motivations (not shown), the current plan (shown), the map of the environment (shown) and the trajectory of the robot (not shown), the behaviors and their activations and exploitations (not shown).

VI. Conclusion

Designing a mobile robot that must operate in public settings probably addresses the most complete set of issues related to autonomous and interactive mobile robots, with system integration playing a fundamental role at all levels. With this paper, we explain how these issues are addressed and integrated using Spartacus and MBA's computational architecture. Software like MARIE and the LogViewer illustrate the importance of software engineering tools to address the underlying integration challenges. Such tools play an important part of the scientific process of studying and designing highly-integrated and complex systems.

One of our objectives in the near future is to continue developing these tools and to improve Spartacus' integrated capabilities (e.g., recognizing electric outlets, gestures, and interacting through a natural language interface). We also want to explore new avenues provided by them like developing a comparison framework for different navigation tools and planning algorithms or porting the same implementation on robotic platforms from different manufacturers and with heterogeneous capabilities. As entries will move from proof-of-concept demonstrations to robust systems, the AAAI Challenge is becoming a scientific venue directly taking on the artificial intelligence's fundamental objective of working in the wild, messy world[4].

Acknowledgments

F. Michaud holds the Canada Research Chair (CRC) in Mobile Robotics and Autonomous Intelligent Systems. Support for this work is provided by the Natural Sciences and Engineering Research Council of Canada, the Canada Research Chair program and the Canadian Foundation for Innovation.

REFERENCES

[Beaudry *et al.*2005] Beaudry, E.; Brosseau, Y.; Cote, C.; Raievsky, C.; Letourneau, D.; Kabanza, F.; and Michaud, F. 2005. Reactive planning in a motivated behavioral architecture. In *Proceedings American Association for Artificial Intelligence Conference*, 1242–1247.

[Byers *et al.*2003] Byers, Z.; Dixon, M.; Smart, W. D.; and Grimm, C. M. 2003. Say cheese! Experiences with a robot photographer. In *Proceedings Fifteenth Innovative Applications of Artificial Intelligence Conference*, 65–70.

[Cote *et al.*2004] Cote, C.; Letourneau, D.; Michaud, F.; Valin, J.-M.; Brosseau, Y.; Raievsky, C.; Lemay, M.; and Tran, V. 2004. Code reusability tools for programming mobile robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1820–1825.

[Cote *et al.*2006] Cote, C.; Brosseau, Y.; Letourneau, D.; Raievsky, C.; and Michaud, F. 2006. Using MARIE in software development and integration for autonomous mobile robotics. To be published in *International Journal of Advanced Robotic Systems, Special Issue on Software Development and Integration in Robotics*.

[Gockley *et al.*2004] Gockley, R.; Simmons, R.; Wang, J.; Busquets, D.; DiSalvo, C.; Caffrey, K.; Rosenthal, S.; Mink, J.; Thomas, S.; Adams, W.; Lauducci, T.; Bugajska, M.; Perzanowski, D.; and Schultz, A. 2004. Grace and George: Social robots at AAAI. Technical Report WS-04-11, AAAI Mobile Robot Competition Workshop. pp. 15-20.

[Haigh & Veloso1998] Haigh, K., and Veloso, M. 1998. Planning, execution and learning in a robotic agent. In *Proceedings Fourth International Conference on Artificial Intelligence Planning Systems*, 120–127.

[Kuipers & Stroupe2003] Kuipers, B., and Stroupe, A. 2003. The AAAI-2002 robot challenge. *AI Magazine* 24(1):65–76.

[Lemai & Ingrand2004] Lemai, S., and Ingrand, F. 2004. Interleaving temporeal planning and execution in robotics domains. In *Proceeedings National Conference on Artificial Intelligence*, 617–622.

[Letourneau *et al.*2006] Letourneau, D.; Cote, C.; Raievsky, C.; Brosseau, Y.; and Michaud, F. 2006. Using MARIE for mobile robot software development and integration. To be published in Brugali, D., ed., *Software Engineering for Experimental Robotics*, Springer Tracts on Advanced Robotics. Springer-Verlag. chapter 3.

[Letourneau, Michaud, & Valin2004] Letourneau, D.; Michaud, F.; and Valin, J.-M. 2004. Autonomous robot that can read. *EURASIP Journal on Applied Signal Processing, Special Issue on Advances in Intelligent Vision Systems: Methods and Applications* 17:1–14.

[4]As stated by Ronald Brachman in the 2005 AAAI Conference Presidential Address.

[Lienhart & Maydt2002] Lienhart, R., and Maydt, J. 2002. An extended set of Haar-like features for rapid object detection. In *Proceedings of the International Conference on Image Processing*, 900–903.

[Maxwell *et al.*2004] Maxwell, B.; Smart, W.; Jacoff, A.; Casper, J.; Weiss, B.; Scholtz, J.; Yanco, H.; Micire, M.; Stroupe, A.; Stormont, D.; and Lauwers, T. 2004. 2003 AAAI robot competition and exhibition. *AI Magazine* 25(2):68–80.

[Meeden *et al.*2000] Meeden, L.; Schultz, A.; Balch, T.; Bhargava, R.; Haigh, K. Z.; Bohlen, M.; Stein, C.; and Miller, D. 2000. The AAAI 1999 mobile robot competitions and exhibition. *AI Magazine* 21(3):69–78.

[Michaud *et al.*2001] Michaud, F.; Audet, J.; Letourneau, D.; Lussier, L.; Theberge-Turmel, C.; and Caron, S. 2001. Experiences with an autonomous robot attending the AAAI conference. *IEEE Intelligent Systems* 16(5):23–29.

[Michaud *et al.*2005] Michaud, F.; Letourneau, D.; Arsenault, M.; Bergeron, Y.; Cadrin, R.; Gagnon, F.; Legault, M.-A.; Millette, M.; Pare, J.-F.; Tremblay, M.-C.; Lepage, P.; Morin, Y.; and Caron, S. 2005. Multi-modal locomotion robotic platform using leg-track-wheel articulations. *Autonomous Robots, Special Issue on Unconventional Robotic Mobility* 18(2).

[Michaud2002] Michaud, F. 2002. EMIB – Computational architecture based on emotion and motivation for intentional selection and configuration of behaviour-producing modules. *Cognitive Science Quaterly, Special Issue on Desires, Goals, Intentions, and Values: Computational Architectures* 3-4:340–361.

[Montemerlo, Roy, & Thrun2003] Montemerlo, M.; Roy, N.; and Thrun, S. 2003. Perspectives on standardization in mobile robot programming: The Carnegie Mellon navigation (CARMEN) toolkit. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2436–2441.

[Perez *et al.*2002] Perez, P.; Hue, C.; Vermaak, J.; and Gangnet, M. 2002. Color-based probabilistic tracking. In *Proceedings of the European Conference on Computer Vision*, 661–675.

[Schmidt1994] Schmidt, D. C. 1994. Ace: An object-oriented framework for developing distributed applications. In *Proceedings USENIX C++ Technical Conference*.

[Schultz2001] Schultz, A. C. 2001. The 2000 AAAI mobile robot competitions and exhibition. *AI Magazine* 22(1):67–72.

[Simmons *et al.*2003] Simmons, R.; Goldberg, D.; Goode, A.; Montemerlo, M.; Roy, N.; Sellner, B.; Urmson, C.; Schultz, A.; Abramson, M.; Adams, W.; Atrash, A.; Bugajska, M.; Coblenz, M.; MacMahon, M.; Perzanowski, D.; Horswill, I.; Zubek, R.; Kortenkamp, D.; Wolfe, B.; Milam, T.; and Maxwell, B. 2003. Grace : An autonomous robot for the AAAI robot challenge. *AI Magazine* 24(2):51–72.

[Smart *et al.*2003] Smart, W. D.; Dixon, M.; Melchior, N.; Tucek, J.; and Srinivas, A. 2003. Lewis the graduate student: An entry in the AAAI robot challenge. Technical report, AAAI Workshop on Mobile Robot Competition. p. 46-51.

[Smart *et al.*2005] Smart, W. D.; Tejada, S.; Maxwell, B.; Stroupe, A.; Casper, J.; Jacoff, A.; Yanco, H.; and Bugajska, M. 2005. The 2004 mobile robot competition and exhibition. *AI Magazine* 26(2):25–35.

[Taylor1999] Taylor, P. 1999. The Festival speech architecture. URL: http://www.cstr.ed.ac.uk/projects/festival/.

[Valin *et al.*2004] Valin, J.-M.; Michaud, F.; Hadjou, B.; and Rouat, J. 2004. Localization of simultaneous moving sound sources for mobile robot using a frequency-domaine steered beamformer approach. In *Proceedings IEEE International Conference on Robotics and Automation*, 1033–1038.

[Valin, Michaud, & Rouat2006] Valin, J.-M.; Michaud, F.; and Rouat, J. 2006. Robust 3D localization and tracking of sound sources using beamforming and particle filtering. In *Proceedings International Conference on Audio, Speech and Signal Processing*, 221–224.

[Valin, Rouat, & Michaud2004] Valin, J.-M.; Rouat, J.; and Michaud, F. 2004. Enhanced robot audition based on microphone array source separation with post-filter. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*.

[Vaughan, Gerkey, & Howard2003] Vaughan, R. T.; Gerkey, B. P.; and Howard, A. 2003. On device abstractions for portable, reusable robot code. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2421–2427.

[Yanco & Balch2003] Yanco, H., and Balch, T. 2003. The AAAI-2002 mobile robot competition and exhibition. *AI Magazine* 24(1):45–50.

[Zhao2003] Zhao, Y. 2003. A model of computation with push and pull processing. Master's thesis, University of California at Berkeley, Department of Electrical Engineering and Computer Science.