Enabling Intelligence through Middleware: Report of the AAAI 2010 Workshop

Monica Anderson and Andrea L. Thomaz

■ *The AAAI 2010 Workshop on Enabling Intelligence through Middleware (held during the Twenty-Fourth AAAI Conference on Artificial Intelligence) focused on the issues and opportunities inherent in the robotics middleware packages that we use. The workshop consisted of three invited speakers and six middleware research presenters. This report presents the highlights of that discussion and the packages presented.*

Intelligent robots could provide the key to improving quality of life. For example, baby boomers are aging. Home-based robots that assist with housekeeping tasks enable more of the population to remain in their homes while reducing health-care costs. Using robots to survey, surveil, and secure regions of interest requires intelligent algorithms that can act and react autonomously in dynamic regions. Unfortunately this vision is quite distant from the current reality where deployed military unmanned vehicles can require teams of three to four operators. Researchers are actively pursuing interdisciplinary research that enables robots to function autnomously within arbitrary environments alongside people.

The goal of the AAAI 2010 Workshop on Enabling Intelligence through Middleware was to examine both the successes and opportunities to provide tools that enable a larger pool of researchers to experiment with embodied, intelligent algorithms. The half-day workshop, attended by over 80 people, was held as part of the Twenty-Fourth AAAI Conference on Artificial Intelligence in Atlanta Georgia on July 12, 2010. The workshop consisted of two parts: (1) invited talks and (2) middleware presentations. Robert Kohout (DARPA), Sven Koenig (NSF) and Dustin Heaton (Alabama) delivered the invited talks, which focused on the perceptions of the current issues and possible research approaches. Middleware presenters, selected to encompass a wide range of views on middleware development, represented six frameworks: ROS (Stanford/Willow Garage), YARP (Italian Institute of Technology), CAST (University of Birmingham), LCM (University of Michigan), Tekkotsu (Carnegie Mellon University), and BotSense (Road Narrows/SIU-Edwardsville).

Invited Talks

Robert Kohout's and Sven Koenig's talks centered on how to include middleware and its impact on integration in the scientific discussion. Kohout stated his belief that we will only be able to create truly intelligent machines by integrating a wide variety of intelligent competencies in a single system, grounded in

perception and control. He believes that progress in robotics research has been hindered by the fact that most system-level results cannot be repeated outside of the laboratories in which they are achieved. Kohout presented a view of science as an effort in which a community builds up a shared understanding of empirical phenomena over time. From this perspective, the repeatability of research efforts is fundamental. If results cannot be repeated, they cannot be built upon or extended, and this limits the community's ability to understand and leverage new ideas.

Science, in the narrowest definition, also suffers. The fundamentals of good science are reproducibility. In biological sciences, reproducibility is key to verification and leveraging the work of others. Unfortunately, when trying to reproduce results in robotics papers, it is often unclear if failure to duplicate results is due to incorrect verification, missing information or spuriousness of original results. When published results cannot be reproduced, is it good science? Koenig expanded this idea in his talk by redefining dissemination in grant proposals to include sharing of software in addition to results as a mechanism for enabling verification and leveraging prior work.

Koenig acknowledged that barriers exist to getting middleware research funded as it is considered engineering. The key to securing funding is to focus on the science inherent in making the pieces work together. He also presented some National Science Foundation programs that could be key to securing funding. In addition to the Robust Intelligence program, which is a main funder for robotics research, he mentioned the Computing Research Infrastructure program (CRI). This program can support middleware proposals that assist larger groups with conducting research.

Kohout stressed the need to share platforms and software as a step toward a coherent science of integration. The reality is that as each research effort chooses a middleware or framework on which to conduct research, researchers on other frameworks are no longer able to readily validate or leverage that work. A balance between flexibility of choosing hardware and software should be weighed against the importance of sharing with the community. A narrowing of choices to enable sharing requires that we look at three to four basic platforms and frameworks that support 95 percent of research needs. It may be that over time, the strongest paradigms will emerge as de facto standards. Meeting the needs of the largest number of researchers with the fewest middleware packages could improve the ability of researchers to collaborate and readily share software.

Dustin Heaton and Jeff Carver (The University of Alabama) highlight the larger community that is not being served. Robotics also engages a large "citizen science" community. The magnitude of the community can be seen through the number of local and Internet-based special interest groups. The impact of contributions from citizen-based scientific societies is well documented. For 111 years, citizen scientists have assisted academics with assessing the health of the bird population through the Audubon Society's Christmas bird count. Just as the success of this endeavor requires engagement from researchers through providing materials such as bird guides and checklists, middleware could provide citizen roboticists with tools that encourage intelligent robot design and implementation. Surveys of likely users have highlighted integrated sensor systems (that is, GPS and visual object identification); those that are low-cost, open source, and easy to use are of a particular interest. Given that over 50 percent of citizen roboticists surveyed used either C or C++ as a development language, the sophistication of these users can be better leveraged with more accessible tools and interfaces.

Middleware Packages

Although there may be agreement that a shared framework could be the answer to enabling intelligence, there is no consensus regarding the correct paradigms in that framework. Each middleware researcher approaches the goals of reusability and verification differently through choices in interprocess communication, development tools, and exposed component interfaces.

The most common approach to reusability is modularity. Separation of concerns allows for the encapsulation of implementation. Hardware details are universally abstracted to natural, now de facto interfaces. Other interfaces delineate functions tasked with higher-level intelligence such as localization and path planning. In this manner, high-level intelligent algorithms can be reused across similar sets of hardware. This modularity extends to the run-time environment. Multiple threads of execution are used to separate and parallelize controlling functions. However, the similarity between architectures diverges as the mechanism for managing interprocess communication becomes choices in transport, interfaces, and tools. Each presenter during the workshop discusses the design considerations and implications in each framework.

Robotics Operating System

Morgan Quigley (Stanford University) presented the robotics operating system (ROS),[1] an open source framework designed to accommodate a wide range of robotics systems. The core development team includes researchers at Stanford University and Willow Garage. There are many high-level

algorithms such as door opening, object recognition, and navigation in cluttered, dynamic environments designed to operate on the PR2 robot.

As an evolution of previous robotics frameworks, ROS handles both the technical and social challenges using modern software development techniques. The technical issues concern portability, efficiency, and scale. ROS uses a publish / subscribe messaging paradigm that allows for the producers and consumers of messages to be asynchronous and loosely coupled. Message formats are automatically generated. Messages are named and can be remapped at run time. This approach allows for programs to start/stop/crash in any order, providing some flexibility and reliability.

ROS models its development tools after existing open source tools providing a set of specialized command-line tools that handle high-level management tasks such as package building and dependency checking. Currently there are 1187 packages in known public repository. Groups of versioned packages that work together are called stacks. Distributions are collections of stacks that are stable to build against. Automated unit and integration testing is performed on all committed code to reduce the introduction of errors.

ROS mirrors the distributed development process of open source software with the goal of leveraging distributed expertise through collaborations. Although a main code repository exists at Willow Garage, other collaborating institutions maintain and control separate repositories of packages designed to extend or improve existing modules. This model has encouraged 30 other institutions to collaborate through hosted contributions. Other tools are used to provide cohesiveness to the distributed software effort: a web crawler that indexes known repositories, wiki-based self-documenting packages, an email list, and Trac for bug reports.

Yet Another Robot Platform

Giorgio Metta (in conjunction with Paul Fitzpatrick and Lorenzo Natale) from the Italian Institute of Technology
presented yet another robot platform (YARP).[2] YARP takes a slightly different approach, prioritizing mechanisms that promote code sharing through the creation and utilization of successful components on any platform. Given that writing software is time-consuming and difficult, software collaboration can provide a speed boost as groups leverage pretested, modular components. Unfortunately, there is no reward for producing reusable code. Rather than attempt to provide a single solution for all development groups, Metta suggested that a more viable approach would be to leverage the natural communities around specific robot platforms.

YARP is open source (LGPL) middleware for humanoid robotics. Initially a collaboration between MIT and University of Genoa, it is now used by the RoboCub consortium. Primarily developed in C++, it leverages existing development tools (CMake, and so on) and language bindings (Java, Perl, Python, C#) through Swig. It uses the adaptive communication environment to wrapper operating system–specific application programming interfaces (APIs) to provide cross-platform portability. Yarp is a thin framework, requiring only 54 megabytes of RAM and 28 megabytes of flash memory when running on an embedded system. It provides no special build system, utilizing existing tool chains on each operating system. However, maintaining a cross-platform framework limits the availability of support tools (that is, IDL/code generation for RPC calls).

YARP achieves modularity through separating the algorithms from the data transport. Data exchanged between components is not constrained to framework- or operating system–specific protocols. The YARP network can utilize any number of transport protocols including standards such as TCP, UDP, HTTP to distribute an application across a physical system running a variety of operating systems. This additional layer of abstraction allows YARP to communicate and provide a bridge to other frameworks such as TCPROS to ROS.

Lightweight Communications and Marshalling

Edwin Olson from University of Michigan presented lightweight communications and marshalling (LCM).[3] LCM supports C, C++, Java, Python, Matlab, and C# across a variety of POSIX, BSD, Windows, and embedded platforms. LCM provides data visualization tools that allow developers to inspect messages as they travel through the system. In addition, LCM supports data logging and injection for testing purposes.

LCM also focuses on scalability and reliability through the use of publish/subscribe as the message paradigm. However, LCM differs from ROS and YARP by using multicast UDP and emphasizing run-time type safety. Using multicast UDP as the primary transport supports high-bandwidth, low-latency message passing. In addition, there is no centralized process that manages subscribers and message forwarding, making performance independent of the number of subscribers. With UDP, message delivery is not guaranteed. Olson recognizes that with time sensitive systems it is better to drop messages and process fresh data than to fall behind real time. Messages are defined using an implementation-agnostic specification that is translated into the language-specific implementation at compile time. The LCM transport detects

incompatibilities at run time using the higher-level message type specification.

Tekkotsu

David Touretzky from Carnegie Mellon University presented Tekkotsu,[4] an open source robot application framework built on C++, with GUI-based teleoperation and monitoring tools in Java. Tekkotsu focuses on providing a unified, high-level framework for simplifying robot programming. Developers can specify desired effects without worrying about how to achieve them. For example, they can use a component called the MapBuilder to do visual search, recognize various types of objects, and maintain robot-centered and world-centered maps of the environment. Other high-level modules include the Lookout (manages the sensor package), the Pilot (for navigation), the Grasper (for control of an arm), and a newly proposed Executive Officer (task scheduling and resource management).

Tekkotsu uses an event-based architecture and a parallel, hierarchical state machine formalism. Programmers can quickly compose state machines in a shorthand notation that is automatically compiled into C++ code. Approximately 30–40 different event types are defined, ranging from low level events (button presses, timer expirations) to very high level events (for example, arrival at a navigation destination). These mechanisms hide details from novice programmers, allowing exposure to a broad range of robotics concepts in a layered manner. Tekkotsu is presently in use at dozens of universities and supports a variety of wheeled and legged robots.

CoSy Architecture Schema Toolkit

Nick Hawes presented the CoSy architecture schema toolkig (CAST).[5] CAST is an information-centered framework that uses shared memories for information storage, retrieval, and change notification. Data is exchanged between components through objects in working memory. As a memory-based architecture, processes can refine data in parallel, and the results from one process are available to all. This is a paradigm shift from other participating frameworks that use message passing. The advantage is the simplicity of managing the data and a focus on the architecture and algorithms and not on the data transport and availability. Although this approach is not suited to all domains (for example, those requiring real-time control), Hawes sees a place in the set of middleware packages for a shared memory tool that allows developers to focus on the information-driven integration of many heterogenous components.

CAST uses many standard libraries and frameworks to leverage developer expertise on other platforms. It is built on the Ice middleware package from ZeroC, which provides interoperability between languages and platforms. Other libraries include Log4j and Log4 CXX. Components can be programmed in both C++ and Java.

BotSense

Road Narrows, in conjunction with Jerry Weinberg at SIU Edwardsville, presented BotSense, a relatively new framework. BotSense provides an IP-based proxy server written in C to the POSIX standard for cross-platform compatibility. The client language support includes C, C++, and Python. The hybrid messaging paradigm includes both request/reply and streaming. The server supports plug-in modules for devices, and both messages and configuration files use the XML format.

The target of BotSense is both research and education. A robot's data and commands can be sent and received through a near real-time IP interface that leverages a common distributed paradigm to support learning of the architecture. Therefore, host applications can be quickly developed — a useful resource for teaching computer science, AI, or robotics. With the python interface, GUIs for specific proxied platforms can be quickly implemented.

Conclusion

Middleware frameworks primarily focus on the features that support parallelism and interoperability such as standardization of messaging paradigms. However, improving the ability of researchers to create robot controllers hinges partially on the ability to leverage prior research and software artifacts. Although all middleware frameworks seek to ease controller development by providing a set of modules for low-level device control and higher-level primitives that encompass basic competencies such as path planning and localization, the availability and implementations vary widely.

There are open research questions around middleware. In addition to using mature software paradigms, research on how best to incorporate these technologies such that developers at many sophistication levels can develop robot controllers could widen the appeal of such work. Understanding which of these approaches actually results in increased collaboration and contribution will provide motivation for the narrowing of the available toolset and focus on proven, best practices in robot software development. Future research should focus on consolidating around toolsets, ideally to a core set that supports both novice programmers and advanced developers on a variety of platforms.

Acknowledgements

The authors would like to thank Sven Koenig and Robert Kohout for their participation in the workshop; we also thank Microsoft Research and NSF (IIS-1037866) for their support.

Notes

1. See ros.org.
2. See eris.liralab.it/yarp.
3. See lcm.googlecode.com.
4. See www.tekkotsu.org.
5. See www.cs.bham.ac.uk/go/.

Monica Anderson is an assistant professor in the Department of Computer Science at the University of Alabama. Her research areas include accessible robotics and multi-robot systems.

Andrea L. Thomaz is an assistant professor in the School of Interactive Computing at the Georgia Institute of Technology. She conducts research in the domain of human-robot interaction and interactive machine learning.