Establishing a Lightweight Communicative Multiagent Java Framework

Braxton McCraw, Justin Ruger, Roberto A. Flores
Department of Physics, Computer Science & Engineering
Christopher Newport University
Newport News, USA
braxton.mccraw.09@cnu.edu
justin.ruger.07@cnu.edu
roberto.flores@cnu.edu

Robert C. Kremer
Department of Computer Science
University of Calgary
Calgary, Canada
kremer@cpsc.ucalgary.ca

***Abstract*—This paper presents work in progress towards building $CASA_{LITE}$, which is a Java programming framework to create lightweight, communicational, hybrid multiagent systems. Our goal is to create a framework that runs on small and large devices with a minimal footprint (lightweight), that relies in message communications as the basic mechanism of interaction (communicational), and that allows building a mix of agents ranging from purely software-based to robotic-based (hybrid). To validate our work we plan test cases for single robot control and multiple robot collaboration. $CASA_{LITE}$ will adopt a robot simulator for offline testing of robot programs.**

Keywords: multiagent; robotic; programming; framework.

I. INTRODUCTION

When considering the advancement of technology, it is important not only to consider the importance of solving new problems but also finding ways to improve upon existing solutions. Perhaps no field of study in computer science has contributed more to the automation and efficiency of optimizing problem solving than artificial intelligence. The traditional view in this discipline holds that with a more complex problem we must construct a more complex solution, mostly through more sophisticated abilities for an individual problem-solver component. An alternative approach is to use not one complex (and usually expensive) component but several low-cost units to handle separate parts of the problem. However, it has been observed that having large team sizes and a greater variety of components raises the complexity of the system [1].

On an orthogonal dimension to multiagency we find the means of implementation, where systems are not limited to purely software or purely hardware components: it is increasingly common to find hybrids (such as in robotics), where versatile software programs are imbedded in the control of complex hardware devices. The focus of our research lies in this area, where groups of multiple separate components (named agents) work together on a task (either cooperatively or additively) and together comprise what is referred to as a multi-agent system (MAS).

Agents can have varying degrees of cooperation and communication between them as well as a range of decision-making independence. Agents can either have the same nature and abilities (homogenous) favoring tasks that are scalable via agent addition, or have different specializations (heterogeneous), favoring applications that benefit from a division of labor. The greatest strength of a multiagent approach is the low coupling afforded by the modularity of its components. General multi-agent frameworks can then be tailored to an application's requirements, leading to a world of possible implementations [2].

Multi-agent systems, while providing a useful abstraction are not fully adopted yet for general use. This can be attributed to a lack of awareness of the potential of agents working in tandem, small publicity of successfully implementations, over-expectations of early adopters of agent technologies, aversion to taking risks on a relatively young and unproven technology, and the lack of developmental and design tools for creating agent systems has led to trepidation of investing time and money into widespread multi-agent applications [3].

Although MAS is an appealing abstraction to organize complex systems, we are concerned with the lack of appropriate tools to implement such systems and, in particular, hybrid communicative MAS. A hybrid framework would allow the implementation of potentially mixed populations of software-controlled hardware agents (e.g., sensors, robots) and purely software agents (e.g., centralized coordinators, decision-makers) that communicate through explicit messaging to organize and coordinate their actions. In our experience, several frameworks could be used to implement such systems; among them are Player/Stage, MS Robotics Developer Studio, JADE and CASA. Given the objectives defined by their creators, the features in these frameworks cannot squarely be compared vis-à-vis. However, these features to a varying degree make them amenable to hybrid MAS implementations.

In this paper, we present as our contribution our early efforts implementing $CASA_{LITE}$, a small-footprint framework to build hybrid communicational multiagent systems. Our framework is planned to be lightweight for deployment in a range of devices, from small and embedded devices (such as SUN/Oracle SPOT [4]) to hand-held devices (such as phones and tablets), and computers with larger capacities. We chose Java as the implementation language to maximize the array of devices in which we could deploy our framework and for its suitability as a familiar language for undergraduate students. As a test case, we'll deploy a $CASA_{LITE}$ agent on an Android tablet that both interfaces with an iRobot Create and is able to exchange messages and video feedback to a remote $CASA_{LITE}$ agent running in a laptop. To validate the appropriateness of $CASA_{LITE}$ as a multiplatform framework,

 ISBN: 978-1-61208-352-0

we'll also implement a $CASA_{LITE}$ agent for the NAO humanoid robot [5]. To validate $CASA_{LITE}$ as a collaborative framework we will implement multi-robot collaboration to solve a maze. Lastly, $CASA_{LITE}$ supports inter-agent communication through socket-based streams transmitting text messages with LISP-like syntax compliant with KQML (Knowledge Query Manipulation Language) [6] and FIPA (Foundation for Intelligent Physical Agents) [7].

The remainder of the paper is organized as follows. Section 2 discusses MAS taxonomy and frameworks considered as existing alternatives to implement hybrid communicational MAS. Section 3 briefly describes an overview of the design of $CASA_{LITE}$, and Section 4 presents our planned experiments and conclusions.

II. MAS TAXONOMY & FRAMEWORKS

In one dimension, agents are organized by their degree of sophistication, from simple reactive components to components of massive decision-making complexity. In another dimension, agents are organized by their degree of collaboration, from isolated components to components that can function in teams and organizations [8]. In one other dimension, agents are organized by their behaviors; for example, grazing – where a robot traverses an area [9]. Naturally, this task is enhanced with a multi-agent approach, since several robots can coordinate their actions to cover an area faster. In such cases, the coordination mechanism must survive the worst-case scenario of unit loss, which should not be handled on a unit-to-unit basis but rather as a collective [10] enabling the team to continue working even in cases of unit loss [11]. We assume that message communication is a coordinating mechanism that complements or even subsumes other coordination mechanisms afforded by the environment. For example, a robot waiting for a block to be moved by another agent could perceive that the block has been moved (thus making the block the coordinating device) or could wait until the agent pursuing the task notifies that the block has been moved (thus making the message the coordinating device). In the former case, it is assumed that agents have the awareness and comprehension to know when a block has been fully moved (c.f., acting when the block is in an intermediate and not final moving state) whereas the latter waits until the agent responsible for the moving action has cleared its completion. Our approach is to assume that messages are the intrinsic coordination device and that agents use communication to coordinate their actions.

Our initial approach towards finding a suitable hybrid MAS platform was to survey existing frameworks to identify candidates. Our ideal framework would be hybrid and multiplatform (able to implement robot interfaces and software agents), provide a simulator (in cases where hardware is not readily available) and communicative (it must support autonomous communication to enable explicit collaboration between agents). An additional requirement is that its communications comply to some degree to standards such as KQML and FIPA. The first framework identified was JADE (Java Agent Development Framework). Being written in a familiar language made JADE an attractive option as well as its inclusion of several Java packages that could be used as-is or modified for the specific platform. However, JADE lack of a robotic simulator yielded a less attractive option than other frameworks [12]. JADE has been used to implement MAS for human-robotic interaction [13], for coordinating a citywide taxi ordering service [14], and an intelligent hotel booking system [15] (further examples can be found in [16]).

We also explored Player [17], which is a robotic device server bundled with a robot simulator named Stage. Stage is relatively lightweight, and is able to simulate hundreds of robots on a standard desktop PC. Communications in Player are achieved through socket streams (which is in line with our goals), making it compatible with programs written in languages supporting sockets. In addition to the Stage simulator, the biggest draw to Player is its simplicity, since the server core has been simplified and reworked to the point where all the functionality is in a single thread of execution. Player's lightweight approach has been explored in large distributed systems, such as the DARPA SDR program, which implemented a 100-robot experiment [18]. On the other hand, there have been reports of compatibility issues between Player drivers and certain robot models, with some problems being operating system specific [19].

Robotics Developer Studio (MSRDS) [20][21] is Microsoft's development environment for designing robot applications across a variety of programming languages. MSRDS has support for iRobot Create and LEGO Mind storm platforms. MSRDS features a robust simulator that can be adjusted through user-made scripts to define simulation parameters. The simulator was the most enticing aspect of MSRDS, supporting simple user-defined polygons to represent the robot and obstacles in the environment. In our view it is the easiest to use framework investigated, although scripting was tedious and it does not lend itself well to modification.

The Collaborative Agent Systems Architecture (CASA) [22][23] is an elaborated framework for agent interaction written in Java. CASA has a robust message and conversational structure based on social commitments, and implements a basic robot simulator for iRobot Create. Its computational footprint, however, makes CASA an unlikely choice to implement agents for small devices.

After reviewing these frameworks, we weighted their communicational abilities, robotic simulation potential and programming fitness for undergraduate students and decided to explore redesigning the core functionality in CASA to support hybrid MAS systems.

III. $CASA_{LITE}$

$CASA_{LITE}$ is a small-footprint framework to build hybrid communicational multiagent systems. It distances itself from existing frameworks with its adaptability and simplicity while incorporating features from other implementations.

Figure 1 shows the core design of our framework. AbstractAgent is the super-class of all $CASA_{LITE}$ agents. It has an event hub (to queue and process events) and a message hub (to queue incoming and handle outgoing messages). Events (not shown) can be synchronous (agents wait for its completion) or asynchronous (executing

 ISBN: 978-1-61208-352-0

independently of an agent's thread), and can be recurrent (executing more than one time); if so, they can be timed to occur at intervals. Message hubs implement socket streams for receiving and sending text messages. Messages are text-base strings whose syntax is KQML/FIPA-compliant. An example message can be "(request :content (curve :speed 200 :radius -3000 :bump any) :language iRobot)". In the context of our iRobotCreate implementation, this message requests the robot to drive forward at a certain speed and radius (i.e., drive in a curve) and stop when the bump sensors detect an obstacle. Events can have an event handler reacting to the event's transitions, e.g., enqueue, dequeue, updates. Event handlers are useful to coordinate responses to message requests when the events created by these requests are resolved. AndroidAgent and NAOAgent (the latter not yet implemented) are agents that implement bare-bone scaffolding programs for their corresponding architecture (namely Android OS and NAO's OS, respectively). AndroidiRobotAgent is an agent interfacing with the core implementation of an iRobotCreate controller, which has its own event hub to queue Create specific commands. This abstract class is extended by either the hardware-aware class of a concrete Create instance (iRobotActual) or by the simulation compatible class (iRobotSimulated) that runs within a 2-dimensional simulator ported from CASA. As will be described in the next section we implement an Android agent as the robot controller running in a tablet sitting atop an iRobot Create robot. Communication between the tablet and the Create are supported through a Bluetooth connection. For practical purposes both the tablet and the Create are treated as one autonomous component.

IV. PLANNED EXPERIMENTS & CONCLUSIONS

We plan several test cases to assert the adequacy of our framework, first for robot control and then for collaboration.

Our first experiment will focus mostly on robot control, with minimal communicational interaction and decision-making. In particular, we will implement a $CASA_{LITE}$ agent in an Android tablet directly interfacing with an iRobot Create through a Bluetooth connection. As mentioned earlier, both the tablet and robot are considered a sole agent.

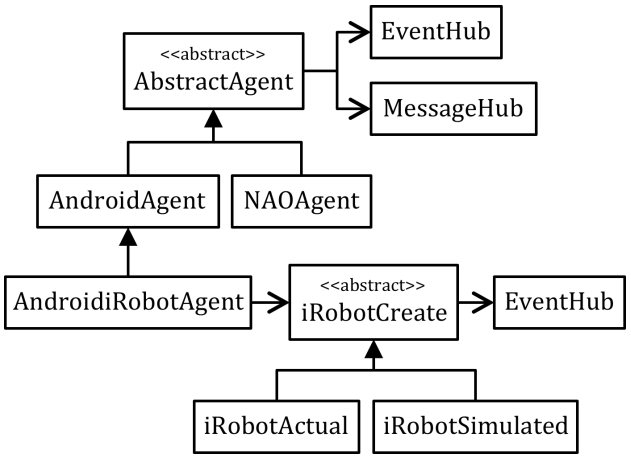

Figure 1. Overview of the main class hierarchy in $CASA_{LITE}$.

This agent will receive messages with commands to control the robot from another agent located in a remote laptop. Likewise, the laptop agent will receive notifications from the Android agent informing of the success or failure of submitted commands plus notifications about the state of the robot. We are designing these notifications under a subscription model, in which the laptop agent subscribes to state changes on the robot monitored by the Android agent, including changes to bumper, wall, floor and cliff sensors. In addition, we will program the Android agent to stream video to the remote laptop agent and to display text messages sent from the laptop agent. At the end, the iRobot/Android agent should be able to drive (guided by a human operator on the laptop agent) through our building and to a different floor by taking an elevator (by using its text interface to request the help of human bystanders to push elevator buttons) and return to the place where we deployed it.

Our second experiment will consist of multiple robots searching for an exit in a rectangular maze. Robots are deployed randomly without a priori knowledge of the environment although they will be aware of other robots through their communications. Initially robots are only aware of their immediate surroundings as afforded by their local sensors, and begin by traversing the maze in single-decision making mode, acquiring knowledge of the maze as they advance and using a simple search algorithm to identify paths to traverse. A different mindset takes over once agents come in contact with each other. At that point, agents communicate their individual maze mappings and combine them (taking as reference their point of contact) into common ground to start division of labor. Any new search paths are negotiated between these agents and any new map space discoveries are shared through their communications, with the potential to add other agents (either isolated or part of other clusters) as they come in contact with each other. Agents will continue exploring the maze until one of the robots finds an exit and communicates its location to all agents in its cluster. To facilitate traversal of the maze, robots will need an ultrasound sensor not currently provided on the Create. This feature will also need to be implemented in the robot simulator.

To conclude, in this paper we present our earlier efforts to build $CASA_{LITE}$, a Java-based framework for implementing lightweight, communicational, hybrid multiagent systems. Agents are programmed with basic communicational abilities to transmit KQML/FIPA-syntax compliant text messages through network streams as a way to enable collaboration. Currently, we have implemented the basic functionality of an abstract agent and the interface to the iRobot Create. Shortly we will begin implementing the Android agent to be deployed in our initial single robot test case scenario. After this test case we will integrate to $CASA_{LITE}$ the robot simulator from the CASA framework and use it to program the collaborative test case in which several robots communicate to find a maze exit.

 ISBN: 978-1-61208-352-0

V. ACKNOWLEDGEMENTS

The work presented in this paper is partially supported by the National Science Foundation under Grant Number (NSF 0841295), GK12 Program, CNU W.I.S.E.

REFERENCES

[1] B. Gerkey and M.J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems." The International Journal of Robotics Research, volume 23, issue 9, pages 939-954, September 2004.

[2] K.P. Sycara, "Multiagent Systems."AI Magazine, American Association for Artificial Intelligence (AAAI), volume 19, number 2, pages 79-92, 1998.

[3] M. Pěchouček and V. Mařík, "Industrial deployment of multi-agent technologies: review and selected case studies." Autonomous Agents and Multi-Agent Systems, volume 17, issue 3, pages 397-431, 2008. doi: 10.1007/s10458-008-9050-0

[4] Sun/Oracle SPOT, Sun SPOT World [Online] Available from: http://www.sunspotworld.com/ [retrieved: May 2014]

[5] NAO, Aldebaran Robotics [Online] Available from: http://www.aldebaran-robotics.com/ [retrieved: May 2014]

[6] T. Finin, Y. Labrou, and J. Mayfield, "KQML as an agent communication language." In Software Agents, J.M. Bradshaw (Ed.), AAAI/MIT Press, pages 291-316, 1997.

[7] FIPA, Foundation of Intelligent Physical Agents [Online] Available from: http://fipa.org/ [retrieved: May 2014]

[8] A. Farinelli, L. Iocchi, and D. Nardi, "Multi-robot systems: A classication focused on coordination." IEEE Transactions on System Man and Cybernetics, Part B, pages 2015-2028, 2004.

[9] T. Balch and R.C. Arkin, "Communication in reactive multiagent robotic systems." Autonomous Robots, volume 1, issue 1, pages 27-52, 1994. doi: 10.1007/BF00735341

[10] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes, "A taxonomy for multi-agent robotics." Autonomous Robots, volume 3, issue 4, pages 375-397, 1996. doi:10.1007/BF00240651

[11] J. Ferber, "Multi-agent system: An introduction to distributed artificial intelligence." Addison Wesley Longman, 1999. ISBN 0-201-36048-9

[12] F. Bellifemine, A. Poggi, and G. Rimassa, "JADE–A FIPA-compliant agent framework." Proceedings of Practical Applications of Agents and Multi-Agents (PAAM), pages 97-108, London, 1999.

[13] V.S. Santos, C.P. Cândido, P. Santana, L.C. Correia, and J.B. Barata, "Developments on a system for human-robot teams." Conference on Autonomous Robot Systems & Competitions (Robótica 2007), volume 1, pages 1-7, Paderne, Portugal, 2007.

[14] A. Moreno, A. Valls, and A. Viejo, "Using JADE-LEAP to implement agents in mobile devices." EXP-In Search for Innovation (special issue on JADE), Telecom Italia Lab, volume 3, issue 3, 2003. [Online] Available from: http://jade.tilab.com/papers/EXP/02Moreno.pdf [retrieved: May 2014]

[15] C. McTavish and S. Sankaranarayanan, "Intelligent agent based hotel search & booking system." 9th WSEAS International Conference on Telecommunications and Informatics (TELE-INFO '10), pages 61-66, Catania, Sicily, Italy, May 29-31, 2010 [Online] Available from: http://wseas.us/e-library/conferences/2010/Catania/TELE-INFO/TELE-INFO-09.pdf [retrieved: May 2014]

[16] JADE, Java Agent Development Framework. [Online] Available from: http://jade.tilab.com/ [retrieved: May 2014]

[17] The Player Project: Software for robot & sensor applications. [Online] Available from: http://playerstage.sourceforge.net/ [retrieved: May 2014]

[18] M. Kranz, R.B. Rusu, A. Maldonado, M. Beetz, and A. Schmidt, "A player/stage system for context-aware intelligent environments." Proceedings of UbiSys'06, System Support for Ubiquitous Computing Workshop, 8th Annual Conference on Ubiquitous Computing (Ubicomp 2006), Orange County California, September 2006, pages 17-21. [Online] http://www.eislab.net/publications/2006/Rusu06UbiSys_preprint.pdf [retrieved: May 2014]

[19] D.S. Michal and L. Etzkorn, "A comparison of player/stage/gazebo and Microsoft robotics developer studio." Proceedings of the 49th Annual Southeast Regional Conference (ACM-SE '11), ACM, pages 60-66, Kennesaw, Georgia, 2011. doi: 10.1145/2016039.2016062

[20] K. Johns and T. Taylor, "Professional Microsoft robotics developer studio." Wiley Publishing, Indianapolis, 2008. ISBN 978-0-470-14107-6

[21] J.S. Cepeda, L. Chaimowicz, and R. Soto, "Exploring Microsoft robotics studio as a mechanism for service-oriented robotics." Proceedings of the 2010 Latin American Robotics Symposium and Intelligent Robotics Meeting (LARS '10), IEEE Computer Society, pages 7-12, 2010. doi 10.1109/LARS.2010.18

[22] R.C. Kremer, "CASA User Manual." [Online] Available from: http://casa.cpsc.ucalgary.ca/doc/CasaUserManual.pdf [retrieved: May 2014]

[23] R.C. Kremer, R.A. Flores, and C. La Fournie, "A performative type hierarchy and other considerations in the design of the CASA agent communication architecture." In F. Dignum (ed.), Advances in Agent Communication, Lecture Notes in Artificial Intelligence, Volume 2922, Springer-Verlag, pages 59-74, 2004.

 ISBN: 978-1-61208-352-0