Intro, motivation

Intellectual Merit. How important is the proposed activity to advancing knowledge and understanding within its own field or across different fields? How well qualified is the proposer (individual or team) to conduct the project? (If appropriate, the reviewer will comment on the quality of the prior work.) To what extent does the proposed activity suggest and explore creative, original, or potentially transformative concepts? How well conceived and organized is the proposed activity? Is there sufficient access to resources?

- *Challenge 1.*
- *Challenge 2.*

Broader Impacts. How well does the activity advance discovery and understanding while promoting teaching, training, and learning? How well does the proposed activity broaden the participation of underrepresented groups (e.g., gender, ethnicity, disability, geographic, etc.)? To what extent will it enhance the infrastructure for research and education, such as facilities, instrumentation, networks, and partnerships? Will the results be disseminated broadly to enhance scientific and technological understanding? What may be the benefits of the proposed activity to society?

Keywords: at least 3 keywords.

1 Introduction

Developing for cyber-physical systems is complicated.

Integration of sensors and actuators into a single cohesive system often requires a composition of service-based components that either produce, consume or transform information.

It is not uncommon to find soldiers tele operating robots to safely disarm improvised explosive devices (IED). Soldiers become attached to specific robots, even performing infield repairs and modifications. Since robots are tele operated, the operator can account for physical changes when using the tele operation primitives. However, since the ultimate goal is for robots to autonomously perform tasks with little or no operator input, we cannot count upon the controller adjusting to physical changes in an appropriate, task-specific way. Making physical changes visible to the autonomous program would require a highly trained robotics expert to reprogram the controller.

This example illustrates a more general problem of integrating of service-based embedded systems into a single architecture. Changes to physical configurations require two updates: once in hardware and once in software. This results from the fact that there is no standard system of record for actuating embedded systems.

2 Motivation

2.1 Current Approaches to robotics hardware integration

The problem of managing to unproven paradigms is confounded by not considering relevant advances in related disciplines that could elevate the usability and applicability, directly affecting adoption and contributions. Applying human interaction, software engineering, interactive design in physical computing and configuration management research principles to the design of robotics tools could provide significant advances in the number and productivity of engineers working with robots for research.

Software engineering concerns itself with the science of creating software that is cost-effective, reliable, maintainable and reusable. Although each criterion is a concern for robotics controllers, the issues of reliability and reusability directly affect engineersÕ ability to adopt certain toolsets. Many approaches have been used to increase reusability of software artifacts. Player/Stage drivers can be used directly in ROS [8]. Microsoft Robotics Studio uses a service-based paradigm to abstract data sources. Orocos [9] leverages the component design to package related software into a reusable piece. Carmen[15], LCM[16] and ROS [8] abstract the message content from the message generation by providing a transport between modules with self-describing data. These approaches utilize fairly advanced computer science paradigms of distributed development (utilized by many Linux-based open source projects). Unfamiliarity with these paradigms may discourage many researchers, hobbyists and students from attempting to reuse software, even when it can teach domain specific concepts.

Reliability can be considered a function of reusing well-tested software artifacts. Unfortunately, the most replaced parts of a robot system are the hardware components. Software reuse does not often distinguish between the non-functional aspects of hardware such as error characterization, resolution changes and timing differences beyond the initial layer of sensor data consumption. Making such issues explicit may improve reliability and encourage tacit knowledge acquisition.

Physical computing is a new movement in hardware design that considers how humans create and express themselves physically. Grassroots [1,17] efforts are empowering users to personalize electronics design and creation. Specifically, the ÒSketch in HardwareÓ toolkits [2] allow designers to sketch the perceived inner workings of a design, which allows reasoning about its dynamic properties. The

ÒsketchÓ is a lightly committed set of connections between components that allow for the interaction of predefined graphical and hardware components as prototypes. The user can experiment with many different configurations (i.e. sensors, processors, etc) during the course of design. One of the goals of 2009 Sketch in Hardware conference was Òtools to support the design of better, more complex experiencesÓ [17]. Although the richness of software designs comes at the price of complexity, there is an innate recognition of the learning process as iterative, requiring periodic validation. Certainly the depth of abstraction is not easily represented in a single-dimension programming environment. In addition, these tools would benefit from streamlined configuration process. Configuration management, a research staple in the information systems domain, is relegated to a required support function within existing architectures. Mechanisms for defining the hardware for the architecture rely upon a file [7,18], a service [8,15] or component selection [2]. In every case, software drivers must explicitly support the hardware that will be used.

If we consider the problem of making good research tools based on scientifically supported paradigms in related areas, there is an opportunity to measure the increase in self-efficacy, domain knowledge, and robotics research adoption:

1. Use hiding to simplify configuration management,
2. Move reusability from the abstraction interface to the hardware level,
3. Use abstraction and visualization appropriately to enable the gaining of tacit,
4. Create dimensioned software tools that provide layered exposure, and
5. Support a Òsketching in softwareÓ concept to support the learning and creativity in software design process.

2.2 Preliminary Work

Robot Hardware Descriptions: Existing approaches: Many frameworks use a declarative description of the robots. Player/Stage [7] is both a 2D simulator and a robot control framework. Robot description files (shown in Figure 1:left) are broken into two pieces: 1) a 2D description of the robot and its sensors and 2) a set of interfaces that abstract the data produced by hardware to a standard format. The description, used for simulating the robot, consists of a polygon-based footprint with sensor locations marked. Actuation and sensor characteristics along with parameters for simplified error models are used to complete the model of the robot. A domain-specific set of classes and message types describe what data can be obtained or how the robot can be manipulated including position (pose2d) and distance to other objects (single point is range, multipoint is laser). The classes and message types represent the interface that abstracts the robot hardware to the data that it can produce or consume. Writing software to the interfaces that a robot can utilize (rather than the specific robot) allows software to be written either for a simulated robot or a real robot, which in turns eases the transition from simulation to physical implementation.

ROS [8] targets a 3D simulation framework (Gazebo) and more sophisticated intelligent controller, which require a more rigorous description. UDRF (Uniform Robot Description Format) provides a 3D physical description broken into links and joints to facilitate not only mobile robots but manipulators as well (Figure 1: right). Geometric bounding boxes and meshes allow for collision detection and realistic visualization. Like Player Stage, ROS utilizes a message-based model to decouple data providers from data producers. Ideally robots that provide and consume similar data types can be controlled similarly.

Unlike Player Stage, URDF not only serves as a mechanism for simulating robots but also allows for the visualization of real robots in both real-time and off-line (through saved messages).

A select number of robot control frameworks move beyond visualization information and relevant interface declaration in the hardware description. PREOP (Figure 2), an Alice-based programming interface [19, 20, 21] for robots takes this paradigm further. Not only is 3D visualization information supplied but also the programming interface is completely specified by the selection of the robot object. This is accomplished by linking the real-time control mechanism and exposed API available to the user within the robot object.

All robotic architectures require that the user provide information regarding the target hardware devices and the resulting data at development time in absentia of the hardware. Two problems result from this process: 1) users must understand the hardware and resulting data and 2) encode the hardware details properly in the framework of choice. A mechanism that provides the user with the details of available resources could remove this step and the associated issues with attempting to properly characterize the hardware. More importantly, hardware that can self-describe could lessen both frustration and the need for in depth hardware knowledge on the part of the user.

Robot Hardware Descriptions: Creating a robot hardware discovery mechanism: We propose a new paradigm where knowledge of the hardware mechanism is embedded in the hardware, rather than declared in software. This novel universal design called the (RDS) Robot Device Specification has three purposes: 1) provide enough information for simulation and visualization of hardware and controllers, 2) declaratively specify the mechanism for requesting data and actuation, and 3) inform users of standard message types that can be obtained from the hardware to facilitate connection to existing frameworks. The challenge in successfully defining the RDS is in creating a model that captures the generalizable aspects of robots and provides a mechanism to specialize the aspects that vary.

RDS (and general reuse in the community) relies upon the relatively invariant nature of mobile robots. Although some robots are built for a specific task, general use robots within education and research communities tend to leverage designs that provide closed loop inverse kinematic solutions; differential drive being part of this class of robots. In addition, many robots including the popular Mobile Robots Pioneer class, iRobot Creates, K-Team robots, Erratic ER-1, White Box Robotics Model 914, Ar.Drones, and BirdBrain Finches contain an embedded firmware controller that accepts commands via a serial, Bluetooth, WiFI or USB interface rather than require the users to download a program to onboard memory. Even robots that require a local software program to run have modes where the local software program presents an API to an external computer (i.e. Lego Mindstorms via Lejos and E-Puck). In both of these cases, the users create an autonomous controller program that communicates with the firmware to affect actuation and to obtain sensor information.

The RDS specification can be defined as domain specific language (DSL) that is used to program robots at a higher, domain-specific level. Although the literature reveals very few attempts at using DSLs for hardware device drivers, Thibault et al report the creation of efficient video device drivers using a novel DSL [22]. In the application of robots, the existence of firmware (not programming hardware directly) simplifies the process. Designing a declarative specification for robot hardware requires a solid domain model of the hardware connection and services that are provided. Domain models, when designed properly, can be somewhat invariant to changes and can provide a stable basis for deciding the structure and parameters of the specification. Figure 3 shows a typical implementation of mobile robot device driver. Primary concepts include connection, message formats, primitives and exposed interfaces. Connections are often leveraged as either request-reply (service-based and requested as needed) or periodic updates that are published or expected (subscribed). Threading models include single (one loop that services incoming and outgoing data), dual threaded (one thread for servicing incoming requests and one thread for periodic requests), or multiple threads (requests create threads and periodic

requests are on different frequencies). Some drivers save state (current position relative to the starting point) and the validation routines for incoming data and read and write routines can vary. Also some raw data must be processed before presenting to the programmer as a result of an API call.

A preliminary RDS has been implemented for the Finch robot from Bird Brain and the K-Team Koala. Figure 4 shows an excerpt from the existing Finch model, its RDS and the resulting code using a template to generate a command line program. In this design, a properties-based description file is used (although any syntax can be used including XML). The intermediate product is an abstract syntax tree that represents the Finch details in a domain specific model. This intermediate format can be further processed to verify conformance to the specification. End products are generated from the verified syntax tree, either in a single or multiple passes, using templates that format data based on the model. The preliminary result [23] of this approach includes RDS specifications and grammars that generate a command line program and a ROS driver. The ROS driver looks for specific interface signatures that match to ROS message structures.

This preliminary result supports the idea that general robot devices can be described declaratively in a manner that supports discovery and that links to the backend processes. The RDS must be expanded to be useful in a larger context. These tasks include but are not limited to: 1) addition of a complete kinematic, visual and collision description consistent with existing simulators and frameworks, 2) error handling at both the communication and primitive levels, 3) implementation of additional threading models, 4) refinement of the state concept and how it matches to primitives and interfaces, 5) management of sensor and actuator error models consistent with existing frameworks, and 6) match internal mechanisms to framework standard interfaces and message types through linking the description and the exposed API instead of relying upon matching external interface signatures. These changes require updates to the specification and the underlying parsers, lexers, tree grammars and string templates.

3 Research Plan

Implementation Pathways: Embedding in Hardware: Although there are many benefits to utilizing the RDS outside of the hardwareÕs firmware, the ultimate goal to provide accessible programming is best met through embedding the robot device descriptions within the device. Discovery occurs when the design environment queries the device for its supported services (or APIs). The initial approach for platforms that support onboard reconfigurable firmware is to augment the firmware to support a single command that communicates the RDS. The information provided by the RDS can be used by any RDS-enabled development environment (discussed in the next section). It is expected that manufacturers will choose to RDS enable their devices once there are more RDS-enabled environments are available. The Finch designer has already agreed to update the firmware of a couple of select models and has committed to including the command once it is more stable and is being adopted (see attached letter of collaboration).

Implementation Pathways: RDIS Creation, Validation and Driver Generation Toolset: Even as we work toward adoption by hardware manufacturers, RDS can provide other efficiencies that also serve the purpose of increasing accessibility. As a side effect of describing the robot device hardware communication interface, the creation of device drivers for frameworks can be automated. This is not a trivial benefit. Each runtime framework requires either a custom device driver that links the framework to the device or a bridge between the target framework and a framework that does have device support. At a minimum, the RDS relieves the manufacturer from providing point solutions for languages, development tools and frameworks. Target toolsets include C, C++, Java, Python, ROS and Websocket server (provides HTML5 and Javascript support). The biggest challenge to this task is the cross-platform sup-

port of the communication interfaces. However, consideration for different platforms can be built into the templates using compiler directives.

It is important to note that the RDS toolset is enabled by ANTLR and StringTemplate. These are open source libraries that parse and process data according to grammars. These grammars are often used to define domain specific languages that are subsequently processed either by interpreters or translators. The sample RDS and the translation to a C-based command line controller and a ROS node was achieved through the use of grammars and the ANTLR and StringTemplate libraries. Although these libraries provide many built-in features, the ability to embed code to customize processing is important to using these tools effectively.

Implementation Pathways: Custom and Reconfigurable Robot Device: In the ÒSketching in HardwareÓ community, many of the robotic devices are custom built by connecting actuators and sensors to netbooks or single board computers (SBCs). Although the RDS cannot completely specify all possible interactions with every sensor and actuator, the communication to the computer and to the peripheral connection mechanisms can be queried and described. A microprocessor-based software module, RDSConnect, would execute on the computer or SBC to act as both a development and runtime interface for the connected peripherals. In this way, RDSConnect mimics the functionality of firmware to provide a cohesive, discoverable interface to attached components. RDSConnect interacts with graphical tools that allow building and validation of RDS files connected using the described communication primitive in real time could be used to integrate a custom microprocessor-based mechanism with supported tools and languages. This approach is useful for robots composed of communication-enabled parts (i.e. the Calliope with Dynamixel USB-enabled actuators) and allows custom robots built from commodity parts to be easily interfaced to development packages.

Reconfigurable hardware provides a unique challenge. Rather than a static description, the RDS becomes a mechanism for communicating changes. As a longer-term goal, research methods that learn the current configuration based on embedded sensors could be appropriately applied here. Either building blocks or configurable linkages would embed proprioceptive sensors to establish kinematic parameters needed for programming. This kinematic learning can also be accomplished by augmenting existing systems with small standalone, networked sensors that are placed at joints [24].

3.1 Team Member Roles, Responsibilities and Expertise

The success of the proposed research requires both breadth and depth in robotics hardware and software. The PIs main research area centers on distributed robot systems. However, her twelve years as a software engineer (culminating in a IT Architect position at IBM) has made the shortcomings of existing tools apparent. She designed the circuit board and the software for the eROSI, a novel robot platform [46]. This platform featured an ASCII message-based API available via Bluetooth and images were served wirelessly via UHF channels. These choices directly affected the platformÕs usability for research students such as the undergraduate team at Berea College [47]. In addition, the PI has either programmed or modified drivers or firmware for robots and devices including the Koala, Finch, Create, Ar.Drone, Lego Mindstorms, Calliope (Dynamixel-based 4 DOF arm), Handyboard, GPS modules, lasers, analog and digital IR and ultrasonic sensors, and custom boards using ATMega chip and serial Bluetooth ICs to interface with Player/Stage, Alice/PREOP, ROS, Carmen, Webots and custom firmware applications. In addition, the PI has studied open source implementations of other drivers and frameworks including PSOS (Pioneer robots), Tekkotsu, and LCM. She was the organizer of the 2010 AAAI Robotics Workshop titled ÒEnabling Intelligence through MiddlewareÓ; an effort to increase awareness of tools within the community and tool needs outside the community [48]. The PI is in a unique position given her expertise, experience and enthusiasm to affect accessibility of robotics

controller development within the larger community.

3.2 Dissemination Plan

Broad dissemination plans include both the academic and non-academic communities. Traditional academic dissemination plans include publications and conference presentations in both the educational and robotics software engineering communities. Future plans include workshops at conference venues as well as at partner campuses will provide hands-on opportunities for those that wish to use the tool for teaching or design. A website will be maintained and all software will be open source. However, primary means of dissemination will utilize partnerships with manufacturers or tool developers as we have seen this method is effective at making hardware widely available [42,43]. The PI will also hold hands-on workshops both to inform and to gather feedback from the community.

3.3 Methodology

The research approach consists of iterative design and integrated usability checks. Rather than attempt to create the research tools in a single iteration, it is better to start with a core of features and iteratively add to those features. This approach allows for the usability testing within the first 18 months as an input into the next iteration. The schedule details the work for two graduate students for the first three years: one student to focus on grammars and templates and one student to focus on RDSWare and usability testing. Undergraduate students will assist with coding, testing and support activities. The project team will meet weekly to discuss progress on features and use cases and any design issues. Ongoing activities include dissemination through web sites, conferences and site visits.

4 Curriculum Development Activities

5 Schedule

Tasks and timeline.

6 Intellectual Merit

The intellectual merit of the proposed work centers on research that enables non-academics and non-roboticists to interactively design intelligent robotics controllers. Three areas need to be addressed: abstract the hardware from the software by making the hardware the Òsystem of recordÓ, provide tools that encourage interactive design experiences and identify and validate correct tool design principles while investigating appropriate GUI-based design strategies that manage complexity.

To use any existing robotics software tools, the user must first define the hardware to be programmed. Misconnects can occur when the definition of data returned or capability does not match the existing hardware. This step of configuring software to match the hardware is daunting, especially when coupled with the sophistication level required to use existing tool chains. Hardware that can communicate its capabilities and programming API to a general programming tool would offload the knowledge of the hardware to the system of origin (the correct system of record). In addition, the addition of visual information would support a zero-configuration programming environment that provides fewer frustrations to non-academics or novice programmers. This aspect of the proposed project is

transformative; it could potentially affect the creation of new hardware platforms, reduce the complexity of device drivers (encourage new devices) and lower the learning curve to working with platforms. Although initial iterations will focus on static hardware designs, the application to reconfigurable robots could enable ÒOn-the-flyÓ physical reconfiguration with reasonable ability to program new capabilities in the field.

Interactive design has bought new energy and rigor to fabrication and hardware prototyping. Incorporating interactive design principles into robotics software tools would encourage software prototyping and experimentation, providing learning experiences in computational thinking, computer science and robotics. These interactive experiences leverage the creativity and natural problem solving abilities of future scientists to create a wave of personal robots that are engineered to meet personal needs.

Currently, software tools for robotics are engineered based on untested, conflicting design principles. For example, some tools create specialized robotics development tools while others focus on using standard tool chains. Providing templates and structure within a tool either enhances or detracts from usability. No research study to date has focused on if or when a particular approach to robotics software tools is appropriate. This research provides guidance on the correct tool design principles for the target group. Graphical programming tools seem to have an advantage in development but graphically managing multiple dimensions in software can be hard. This results in graphical tools that are low-level and simplistic while research-based robotics development environments are high-level and require expert knowledge in computer science and software development. This proposal includes the novel approach of automatic component discovery and event-based programming within a visual framework that attempts to control and communicate complexity.

7 Broader Impact

The broader impacts of the proposed work can be categorized as either enabling software design by engaging a broader audience or enabling new research. A new zero-configuration paradigm along with a well-designed, researched graphical programming tool would enable hobbyists to design more sophisticated controllers without requiring expert domain knowledge. In addition, interactive design environments would enable learning about computer science and computational thinking. Citizen roboticists are not the only group that benefits. For example, military personnel are asked to operate and adapt autonomous equipment in the field. Software that acknowledges the learning curve and written to accommodate the expected sophistication and experiences of the user could increase productivity and reduce frustration. Students from novice to expert would also be able to leverage tools to learn and create intelligent products that can be personalized to their own needs. The principles of ÒUniversal DesignÓ [45] in this space may infer that more accessible tools will benefit even advanced users.

The proposed work also enables new research. The RDS will change the components in robotics kits so that they can be reconfigured. This requires additional research in kinematic discovery with low cost components. New instruments for measuring self-efficacy in robotics skills will enable the validation of new tools and techniques in robotics tool development. Automatic component discovery can be leveraged in other domains where the interfaces are fairly static and there are many data providers and many candidate algorithms. This research enables a new model of reuse within the research community that could advance validation of published results through sharing of software artifacts. Additional broader impacts relate directly to the activities of the PI. The PI participated in the 2010 NSF CISE Summit on broadening participation, bringing the message and potential outcomes back to her department and college. The PI serves as an active mentor within her university and research community. She has hosted five REUs from the Software Engineering REU site. She has also taken many students

to conferences including Grace Hopper, Richard Tapia, IJCAI and AAAI. She also advises a senior project group in electrical engineering tasked with building a chess playing mobile robot from commodity parts (Figure 7). As a PI in the ARTSI alliance (Advancing Robotics Technology for Societal Impact), she has fostered productive relationships with other partner schools that have resulted in talks, workshops, sharing of technical expertise and REU student placement (seven students to date). All REUs/undergraduate advisees receive active mentoring including introductions to graduate faculty at other schools where appropriate, review of application materials, recommendation letters, job reference letters and publication and speaking opportunities.

8 Results from Prior NSF Support

1. PI for the NSF Grant: CCLI: Enhancing student motivation in the first year computing science curriculum (DUE- 0736789), 04/01/2008-08/31/2011. This is an ongoing project that addresses retention issues due to motivation through specially designed modules that use PREOP in a CS0 or CS1 laboratory setting. We provided PREOP workshops to teachers at the ARTSI Faculty Meeting, demonstrated PREOP at the Google education summit and added PREOP to Alice instructor training (July 2011). Three papers have been published including two conference papers and two journal papers [13,20,21,23].

2. PI for the Grant: ARTSI-Advancing Robotic Technology for Societal Impact (BPC-A- 0742123), 09/01/07-12/31/2012. This is an ongoing broadening participation grant that increases the number of students interested in computer science and robotics through both research and creative experiences. Robotics programs have been established or enhanced at 14 HBCUs. The PI has provided each school with both onsite and remote instructional support ranging from hands-on workshops on open-source robotics development environments and controller development to hardware assembly instructions. Seven summer research assistants have been hosted at UA, with one student contributing to a submitted paper [49] and resulting in numerous poster presentations.

3. PI for the Grant: SGER: Impact of Intermittent Networks on Team-based Coverage (IIS-0846976), 09/01/2008-08/31/2009. This project investigated both surveillance in constrained environments using limited resources and methods for coordinating robot teams with limited communications capability. The novel approach of using ALUL and a QOS metric allowed for balanced target acquisition and tracking [50-53].

4. Co-PI for the Grant: EMT: Collaborative Research: Primate-inspired Heterogeneous Mobile and Static Sensor Networks (CNS: 0829827) 09/01/2008-08/31/2011. This project investigates bio-inspired approaches for multirobot coordination. We show that given a greedy approach to selecting the next point to search, observation information provides more relevant input to the next action selected. Experimental results show that using observation to infer state and intent to disperse nodes performs better than non-communicative and potential field based cooperation and in some cases direct communications. [49,54-58].

5. PI for the Grant: University of Alabama Participation of EPSCOR Institution in SSR-RC: QOS service metrics for unmanned surveillance (IIP: 1026528). 06/15/2010-06/14/2012. We consider ALUL as a metric that indicates the target acquisition ability of a particular system configuration. When used in conjunction with temporal constraints, we can intelligently automate camera coverage to improve target acquisition and tracking performance of the system. The results from the experiments confirm that the coverage in constrained environments when the existing camera

configuration cannot view large portions of the region of our interest improves when ALUL is considered. To date, one conference paper and one journal article have been accepted [59-60].

PI Syriani is a new investigator and has no prior NSF support.

Data Management Plan

We will publish our results in leading conferences and journals. As is typical in the software engineering community, we will rigorously define our experimental procedures. For example, for each case study we will provide the definition and context, including subject software systems, benchmarks, metrics, and the study setting. In addition, we will document our research questions and hypotheses, our data collection and analysis procedures, results, and threats to validity. We will provide access to our data files via online appendices.

We will release all software artifacts developed in association with this project under the *BSD 3-Clause License* (the "New BSD" license) and will make them available via an appropriate outlet such as Google Project Hosting.

We will distribute annual reports via our web site.