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

Robotic systems are an important class of CPS. The ability of robots to interact intelligently with the world rests upon embedded computation and communication, real-time control, and perception of the world around them[1]. Integration of sensors and actuators into a single cohesive robotics system often requires a composition of service-based components that either produce, consume or transform information in a loosely coupled, parallel manner.

Although robot controller developers can create a program to execute directly on a specific robot platform, more often frameworks are used as an intermediary between autonomous controllers and the hardware. Controllers access and manipulate robots via APIs that reflect a generalized model of the robot. The APIs access robot specific interfaces through device drivers. This approach not only speeds development through reuse of code (code is not hardware specific) but also facilitates the repurposing of controllers to other hardware platforms. Device drivers in this context act as a mechanism for mapping individual robot features to domain concepts. Device drivers are usually implemented as processes that manage the hardware connection for the framework while translating requests for data and action from the framework into formatted message requests and responses. Although some increases in latency are experienced with multiple levels of indirection, it is often considered trivial compared to the time investment of attempting to develop and debug remote applications or retrofitting algorithms for varying hardware platform configurations.

There are some opportunities inherent in this approach. Some would say that device driver development is complex. It is probably true that writing reasonable performing device drivers (like all embedded development) requires an understanding of hardware, timing issues, threading, process synchronization, performance tuning and the operating system of the driver host system. However, good examples exist within frameworks and development of reasonable drivers is not prohibitive in terms of skill set. A more pressing issue concerns the multiplicity involved in creating device drivers. As device drivers provide a programmed link between hardware and frameworks, custom drivers have to be created for each framework hardware pair. Although there are some robot frameworks enjoy more popularity than others, many frameworks are routinely used within the robotics research and education community. An additional challenge is that while there are groups that focus on framework development, in house driver support for existing platforms is usually limited to a few popular platforms and certainly does not include any custom platforms or configurations. Robot manufacturers typically provide device drivers for a subset of "supported" frameworks.

Addressing the issue of needing one-to-one mappings between frameworks and devices can introduce some efficiencies and additional flexibility that robotics developers do not currently have. However there is a bigger issue at stake. Although there are many innovations from software engineering that have been utilized successfully within robotics architectures, there is a notable exception when it comes to identifying the system of record for device syntax and semantics. In the context of robotics, it is this failure that results in major duplication in integrating hardware devices into autonomous controllers. Moreover the more important opportunity in making the hardware the system of record for its syntax and semantics is the ability to discover services and provide a component wrapper around devices that allows for composition of intelligent software.

RDIS (Robot Device Interface Specification) [2] is a DSL that captures the syntax and semantics of hardware devices with embedded controller managed external interfaces. This declarative description captures the manufacturer invariant interface made available by firmware in terms of transports and messages and how they map to domain specific concepts. RDIS is currently a preliminary prototype that supports a limited set of processing models and domain concepts. The work proposed as part of this effort will accomplish the following major innovations to RDIS as a contribution to the engineering

of cyber-physical systems: 1) extend the processing models to include a larger set of **execution semantics**, 2) extend the actuation model to include a larger set of **popular kinematic chains** from both manufacturing and exploration robotics, and 3) using RDIS as a **hardware discovery mechanism**, show composition of devices into a controller that is error-aware.

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). This aspect of the proposed project is **transformative** and directly impacts **engineering of cyber-physical systems** ; 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. This project accomplishes a **breakthrough** using the intersection of software engineering and embedded systems. RDIS describes available services much like web-based services while also providing transport and timing specific details. While this breakthrough proposal concerns itself primarily with robotics devices (those that provide sensor information or actuation), it is expected that the process and principles developed here will be generalizable to sensors and actuators in other domains (i.e. car sensors, etc). 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.

2 Background

2.1 Link between domain modeling and creation of DSLs

What are the benefits of DSLs and how does correct modeling relate?

2.2 Hardware descriptions in existing frameworks

Many approaches have been used to increase reusability of software artifacts. Player/Stage drivers can be used directly in ROS [3]. Microsoft Robotics Studio uses a service-based paradigm to abstract data sources. Orocos [4] leverages the component design to package related software into a reusable piece. Carmen [5], LCM [6] and ROS [3] 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.

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 are broken into two pieces: 1) a 2D description of the robot and its sensors(Figure 1: right) 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.

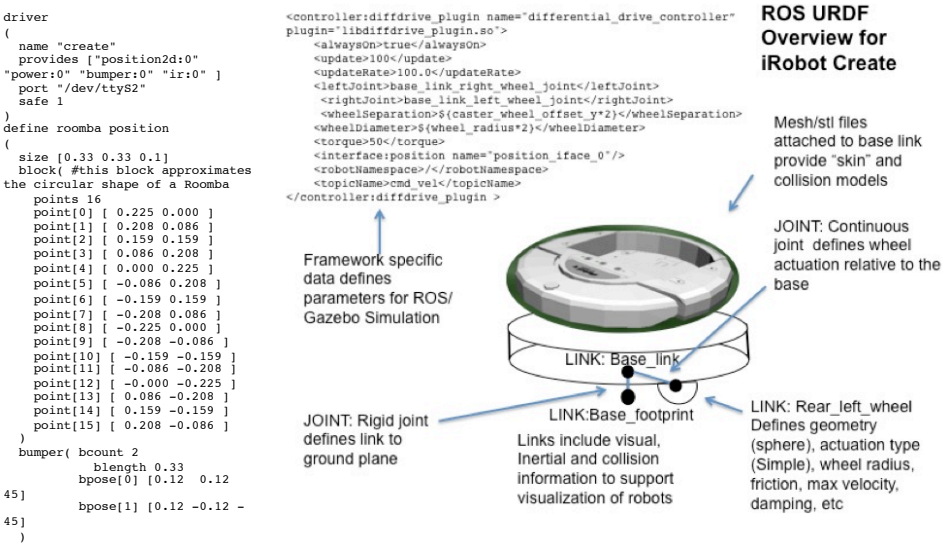

Figure 1: Sample device descriptions within frameworks.

ROS [3] 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, an Alice-based programming interface [8, 9, 10] 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.

2.3 RDIS (Robot Device Interface Specification)

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 (RDIS) Robot Device Interface 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 RDIS is in creating a model that captures the generalizable aspects of robots and provides a mechanism to specialize the aspects that vary.

RDIS (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 RDIS 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 [11]. 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.

A preliminary RDIS has been implemented for the Finch robot from Bird Brain and the K-Team Koala[2]. In this design, a JSON-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 artifacts generated in this approach include RDIS 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.

3 Research Plan

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 RDIS must be expanded to be useful in a larger context. The work proposed as part of this effort will accomplish the following major innovations to RDIS as a contribution to the engineering of cyber-physical systems:

1. Extend the processing models to include a larger set of execution semantics,

2. Extend the actuation model to include a larger set of popular kinematic chains from both manufacturing and exploration robotics, and

3. Using RDIS as a hardware discovery mechanism, show composition of devices into a controller that is error-aware.

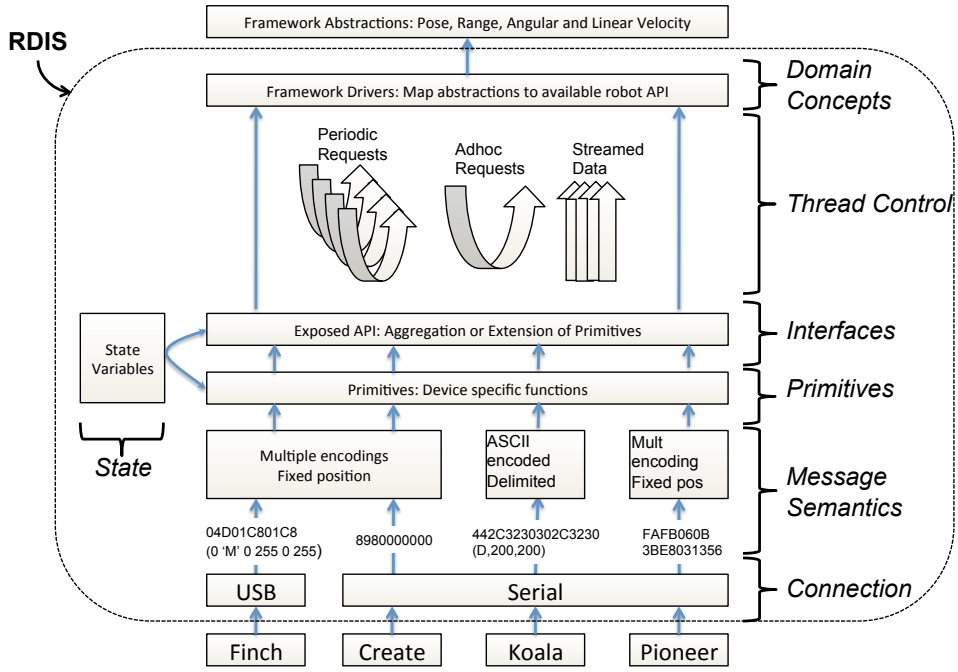

Figure 2: Preliminary domain model for mapping devices to frameworks.

3.1 Expansion of execution semantics

The ability to generalize device to framework mapping requires documenting the concepts in existing mappings. The domain model captures the invariant features of the device to framework mapping drives the contents of RDIS. In Figure 2, the domain is broken into seven related concepts: 1) connections, 2) state, 3) message semantics, 4) primitives, 5) interfaces, 6) thread control, and 7) robotics domain concepts. CONNECTION refers to the transport used to communicate from the device driver to the framework. Information needed to establish and maintain the connection is defined within this concept. The STATE definition serves two purposes: 1) define constants relevant to other sections and 2) allow for state to be saved and retrieved as part of adhoc and periodic requests. The MESSAGE SEMANTICS refers to the structure of the data which is actually sent to the robot over the connection. This specification is decided by the manufacturer and is generally uniform between primitives. PRIMITIVES focuses on describing the firmware interface in order to document the invariant features of the device. The INTERFACE declaration, acting as a logical view of the PRIMITIVES, specifies functions that are available to a developer to control a robot. DOMAIN CONCEPTS map interfaces to domain specific concepts in a framework agnostic manner. These concepts are discussed in detail in [2].

In this research, we expand the basic implementation of each concept to accommodate a larger set of embedded firmware controllers. These tasks include:

1. Addition of a complete kinematic, visual and collision description consistent with existing simulators and frameworks (see Figure 3). Although initial focus will be upon the initial kinematic design (differential drive), the proposed model is designed to accommodate other kinematic designs.

2. Standard mechanism for error handling and notification at both the communication and primitive levels. Error handling examples include re-establishment of connection or actions to take based on output parameters. This is particularly relevant to platforms that require a heartbeat request

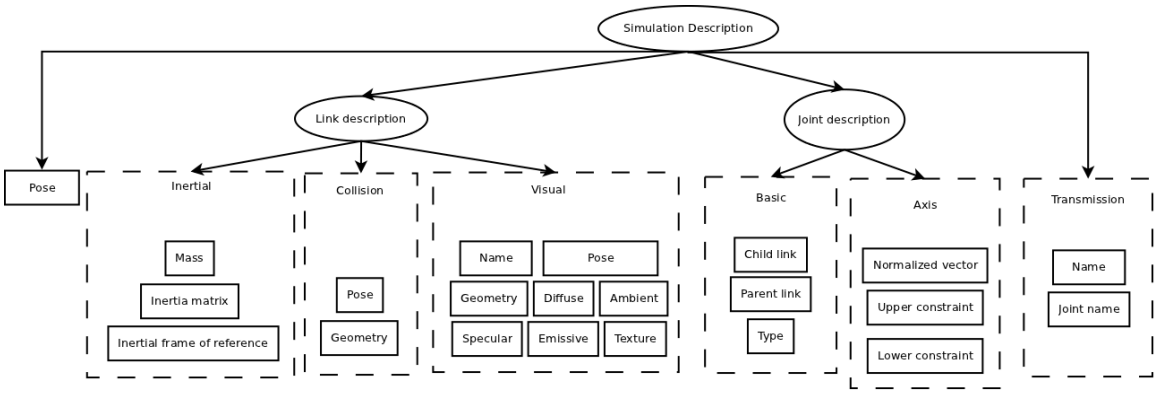

Figure 3: General model for describing kinematics, appearance, collision and dynamics parameters sufficient for simulation on any simulation platform. Currently targeted platforms include Gazebo, Webots, and ROS.

periodically.

3. Expansion of the initial single-threaded implementation to include dual and multi-threading models. A dual model accommodates different input and output threads while a multithreaded model support different frequencies for periodically submitted requests.

4. Refinement of the state concept and how it matches to primitives and interfaces would provide a scripting language for transformation of data. The state variable concept is instrumental in implementing periodic requests asynchronously from the client request/reply system.

5. Management of sensor and actuator error models consistent with existing frameworks will allow for manufacturer provided error models to be propagated to the framework or controller. An example would be a transformation of encoder error to pose error. Although all sources of error (systematic and non-systematic cannot be accounted for in this approach, it is currently better than existing approaches that abstract out specific device errors.

3.2 Expansion of domain concepts to additional kinematic chains

Domain concepts map interfaces to domain specific concepts in a framework agnostic manner. Each framework has a notion of a general set of abstract data types generated from device drivers such as pose estimates, pose relative to an identified landmark, control via linear and angular velocity along a dimension or axis, and point and fields of open distance. The DOMAIN CONCEPTS definition identifies interfaces that map to the abstract concepts allowing for templates to map the abstract concepts to the existing interfaces available on a device.

The initial scope of this research focuses on a set of domain concepts that represent popular platforms and devices and therefore are present in most frameworks as an abstraction. The current set of domain concepts include DIFFERENTIAL DRIVE and RANGE. Differential drive robots (robots with two opposing wheels on a single axis) are controllable using linear and rotational velocity. Some frameworks explicitly recognize differential drive as a kinematic design (Player) while others group all platforms under a single fully expressive interface where the user must know which dimensions are controllable. Acknowledging differential drive explicitly allows for an easier parameterization through state variables that denote odometry resolution, wheel size, offset of axis and wheel base. For example, within the ROS system, a TWIST message type is used to provide control commands to a differential

driver robot through linear and angular velocity. Mapping the TWIST message to the Koala setMotor interface involves calculating left and right velocity from linear and angular velocity and calling the setMotor interface. We map the differential drive domain concept to the SETSPEED interface (defined in the interface section. TRANSVEL and ROTVEL are parameters of the controlDIfferentialDrive abstraction that map to the interface parameters. The unit conversion (based on wheel base and encoder resolution) occurs between the interface and primitive to avoid having to encode the conversion multiple times. Once an interface is designated as fulfilling a domain concept, the mapping better the interface and the framework specific mechanism can be generated. The ROS template (discussed in more detail in [2]) maps each domain concept to the appropriate programming paradigm, which in this case is a subscription to the TWIST message and a callback handler that contains the code to map the TWIST message parameters to the interface parameters.

RDIS will be updated to include additional sensors and kinematic chains. Specifically, the RANGE will be extended to additional dimensions(RANGE ARRAY, and POINT CLOUD) to represent ranges of open space in two and three dimensions. These domain concepts relate to a set of readily available sensors including infrared, laser and ultrasonic range finders. In addition, we can represent newer low-cost, 2D ranging sensors like the Microsoft Kinect. Additional kinematic designs are motivated by partnerships with NASA and RTP. NASA is migrating a robotic platform from VxWorks to Ubuntu with a future update to include RTLinux. This custom kinematic chain involves manipulators with haptic sensors on a MLVDS bus (a new transport). RTP ((Huntsville Research Technology Park)) is an initiative to by the governor of Alabama to create a National Robotics Technology Development and Training Center at Calhoun Community College. Many robot platforms have been purchased and donated for training and are available for evaluation as candidate platforms for RDIS inclusion. Both opportunities expands the platforms that can modeled and contribute to the domain model that can be generalized across other platforms.

3.3 RDIS as a component wrapper an discovery mechanism

talk about how we can use rids to assist with development of controller by using existing tools for domain modeling

4 Project methodology, management, and dissemination plan

4.1 Team Member Roles, Responsibilities and Expertise

need something here about how a domain modeler is important to this research and what tools and expertise is helpful

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 platform 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 "Enabling Intelligence through Middleware" an effort to increase awareness of tools within the community and tool needs outside the community [12]. The PI is in a unique position given her expertise, experience and enthusiasm to affect accessibility of robotics controller development within the larger community.

4.2 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. The PI will also hold hands-on workshops both to inform and to gather feedback from the community.

4.3 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.

5 Curriculum Development Activities

6 Schedule

Tasks and timeline.

7 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 "system of record", 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 "on-the-fly" physical reconfiguration with reasonable ability to program new capabilities in the field.

Reconfigurable hardware provides a unique challenge. Rather than a static description, the RDIS 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].

Implementation Pathways: Embedding in Hardware: Although there are many benefits to utilizing the RDIS outside of the hardwareâ 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 RDIS. The information provided by the RDIS can be used by any RDIS-enabled development environment (discussed in the next section). It is expected that manufacturers will choose to RDS enable their devices once there are more RDIS-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.

Implementation Pathways: RDIS Creation, Validation and Driver Generation Toolset: Even as we work toward adoption by hardware manufacturers, RDIS 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 RDIS 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 support 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 RDIS 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 RDIS 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 "Sketching in Hardware" 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 RDIS 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.

8 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.

9 Results from Prior NSF Support

1. Monica Anderson - 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. Monica Anderson - 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. Monica Anderson - 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. Monica Anderson - 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. Monica Anderson - 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].

6. Eugene Syriani - new investigator and has no prior NSF support.

References

[1] Trinkle, J. and Krogh, B. (2008). CPS Virtual Organization.

[2] Anderson, M., Kilgo, P., and Bowman, J. (2012) RDIS: Generalizing domain concepts to specify device to framework mappings. *International Conference on Robotics and Automation*, Saint Paul MN.

[3] Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., and Ng, A. (2009) ROS: an open-source Robot Operating System. *International Conference on Robotics and Automation* , **?**

[4] Bruyninckx, H. (2001) Open robot control software: the OROCOS project. *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)* , **?**, 2523—-2528.

[5] Montemerlo, M., Roy, N., and Thrun, S. (2003) Perspectives on standardization in mobile robot programming: the Carnegie Mellon Navigation (CARMEN) Toolkit. *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*.

[6] Huang, A., Olson, E., and Moore, D. (2009) Computer Science and Artificial Intelligence Laboratory Technical Report Lightweight Communications and Marshalling for Low-Latency Interprocess Communications. Technical report. MIT.

[7] Vaughan, R. and Gerkey, B. (2007) Really Reusable Robot Code and the Player / Stage Project. *Engineering for Experimental Robotics,* , **?**

[8] Cooper, S., Dann, W., and Pausch, R. (2000) {Alice: a 3-D tool for introductory programming concepts}. *Proceedings of the fifth annual CCSC northeastern conference on The journal of computing in small colleges* , **?**, 107—-116.

[9] Wellman, B. L., Downing, S., Moore, G., and Anderson, M. (2009) Observation-based Cooperation in Mobile Sensor Networks : A Bio-Inspired Approach for Fault Tolerant Coverage. *ISCA First International Conference on Sensor Networks and Applications (SNA-2009)* , **?**, 26—-30.

[10] Anderson, M., Jenkins, O., and Osentoski, S. (2011) Recasting Robotics Challenges as Experiments [Competitions]. *Robotics & Automation Magazine, IEEE*, **18**, 10–11.

[11] Thibault, S. A., Marlet, R., and Consel, C. (1999) Domain-specific languages: from design to implementation application to video device drivers generation. *Software Engineering, IEEE Transactions on*, **25**, 363–377.

[12] Croxell, J. R., Weinberg, J. B., Krauss, R. W., and Smith, S. R. (2008) Robotic Limb Calibration : Accelerometer Based Discovery of Kinematic Constants. *Measurement* , **?**, 1—-4.

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.