2018 NDIA GROUND VEHICLE SYSTEMS ENGINEERING AND TECHNOLOGY SYMPOSIUM
AUTONOMOUS GROUND SYSTEMS (AGS) TECHNICAL SESSION
AUGUST 7-9, 2018 - NOVI, MICHIGAN

ROS-Military: Progress and Promise

Jerry Towler
Unmanned Systems
Southwest Research Institute
San Antonio, TX

Matthew Bries
Unmanned Systems
Southwest Research Institute
San Antonio, TX

ABSTRACT

The potential and promise of automated vehicles has been driving U.S. military research and development in the field for decades. Yet despite this investment, progress toward fully autonomous, field-ready, warfighter-engaged systems has been disappointingly slow. The ROS-Military (ROS-M) program aims to create a central registry of defense-related robotic components surrounded by a community of practice with common processes, systems, and standards. These software components will serve as a foundation for future autonomy efforts to build from, both fulfilling the potential of broad collaboration and greatly increasing the pace of development of autonomous platforms. To execute on this promise, the ROS-M program has begun executing on four parallel paths of investigation, prototyping, and demonstration.

INTRODUCTION

The past two decades have seen dramatic advancements in automated and autonomous vehicle technology, enabling the undertaking of vehicle-based missions with little or no human support, intervention, or supervision. Much of this research and progress has resulted from U.S. military investment and support, and yet advancement in autonomous ground vehicle systems for military applications lags behind that of commercial systems. A primary driver of this retardation has been duplication or replication of effort in both research and development. Instead of developing new and advanced capabilities on top of a common basic autonomy software and hardware system, several independent complete autonomous ground vehicle systems have all received funding over the last twenty years.

Analyses of military unmanned systems programs have reached the same conclusion: increased development of, access to, and use of open and modular software architectures and ecosystems would help provide the accelerated progress programs desire. [2][5]

The cross-service evolution of the autonomy system known today as the TARDEC Robotic Technology Kernel (RTK) provides a real-world template for how such an ecosystem, once created, can be efficiently leveraged to enable research to proceed rapidly from capability to capability and mission to mission using the same underlying autonomy toolkit or pool of resources.

With the expressed need, from both experience and analysis, for an open ecosystem of robotics software, in addition to the existence proof from RTK that such an ecosystem would enable efficient, highly leveraged research to rapidly innovate in unmanned vehicles, the final necessary component of the solution came from the rise of the Robot Operating System (ROS) as the *de facto* standard for robotic system research and development across academia, industry, and government organizations.

To bring this vision to fruition, the ROS-Military project began in 2015 with a small, core group of stakeholders—TARDEC; the Open Source Robotics Foundation (OSRF), the developers and curators of ROS; Southwest Research Institute (SwRI), one of the primary developers of the RTK autonomy system; DCS Corporation, the other primary developer of RTK; and On-Line Applications Research (OAR) Corporation, experts in software architecture and open standards. After this small team defined the initial concept and scope of the initiative, additional stakeholders were invited to participate to ensure the entire military ground robotics community had the opportunity to provide input into the shape of the project. This larger group curated a set of questions that would need to be answered and problems that would need to be solved to ensure the eventual success of the project. Over the last year, the team has directly addressed these questions by developing prototypes of each major component identified by the stakeholders.

This paper describes ROS-Military's past—solidifying the concept and identifying questions that needed answering; present—developing prototypes or initial answers to the posed questions and problems; and future—the proposed continued progress of the project and how it can enhance unmanned systems research moving forward.

PAST

The original concept for ROS-Military was based on the convergence of three influences: the increasing mission demands of unmanned ground vehicles; the example ecosystem of the RTK program; and the increasing prevalence of ROS as a standard for robotics research and development.

Unmanned Ground Vehicle Missions

While the earliest attempts at unmanned ground vehicle (UGV) autonomy sought only basic unmanned operation of the platform, in some cases little more than remote control and teleoperation, increasing hardware and software capabilities have yielded corresponding increases in requirements for the unmanned system.

A complete accounting of the mission expectations for military ground vehicle operations is far beyond the scope of this paper, but a brief sampling will suffice to demonstrate the breadth of capabilities hoped of unmanned systems in the recent past and the near future.

Autonomous convoy operations. Targeted at making resupply operations more efficient and less demanding of personnel, these missions involve an arbitrary number of UGVs operating either as a single unit or as multiple independent units each composed of multiple vehicles. Advanced concepts require performing these convoy operations at high speeds, in unmapped or unfamiliar territory.

Robotic wingman. Targeted at leveraging robotic assets to increase the battlefield effectiveness of personnel, these missions involve supervised autonomous control of an unmanned vehicle and semi-autonomous (human-in-the-loop) control of its payload. Advanced missions include cooperative behaviors in unstructured terrain between manned and unmanned vehicles.

Dismount support. Targeted at offloading burden and responsibility from dismounted personnel, these missions involve small UGVs performing transportation (e.g., of injured personnel), support (e.g., carrying supplies), or reconnaissance tasks. To fully offload tasks, the autonomous system should enable coordinating movement and tasking between the UGV and soldiers without a traditional operator control unit (OCU).

Path re-recognition. Targeted at repeated high-speed traversal of known routes, these missions involve UGVs localizing and following a path through an environment that has been previously traversed or otherwise defined in the environment.

Scene recognition. Targeted at advanced autonomous operation in complex scenarios, these missions involve missions requiring semantic scene or object recognition. Advanced behaviors include object path prediction for dynamic UGV path planning with obstacle avoidance.

Each of these advanced tactical behaviors requires a similar autonomy system. In each case, the base capabilities of the vehicle are similar: translate perceptive and proprioceptive sensing inputs into an internal model of the environment; combine operator input with autonomous behaviors to generate a planned path through that environment; and communicate with the vehicle platform to execute the plan.

These commonalities—both vertical, between advanced behaviors and base autonomy platform capabilities; and horizontal, among different behaviors—suggest the utility of a set of building blocks for autonomous behaviors. ROS-M provides the framework for an ecosystem of such building blocks to thrive.

RTK Ecosystem

An early indicator that the ROS-M concept would accomplish the goals set out for it was the evolution of RTK from its predecessors. A detailed history of many of the programs leading up to RTK is presented in [3]; however, the brief overview presented here will demonstrate the concepts relevant to the germination of ROS-M.

In support of the United States Marine Corps (USMC) Logistics-Connector Distributed Operations Mission and funded by the Office of Naval Research (ONR), Southwest Research Institute developed a low-cost electro-optical (EO) perception, localization, and path planning solution for autonomous (driverless) vehicle operation in austere or harsh off-road environments, without dependence on GPS. This program, known as the Small Unit Mobility Enhancement Technologies (SUMET), used an open, modular, scalable, extensible architecture based on ROS that was specifically targeted for re-use with additional autonomous behaviors and sensing modalities.

Leveraging the platform, architecture, and technologies developed for SUMET, the TARDEC Ground Vehicle Robotics (GVR) Dismounted Soldier Autonomy Tools (DSAT) program targeted two key concepts during its development: optional manning and platform agnosticism. Because the base capabilities developed during SUMET provided the foundation for the expanded perception and behavior work required for the soldier support operations in DSAT, the development team was able to focus its resources on innovative technologies instead of replicating past efforts, resulting in a successful deployment for Combat Assessment. The same autonomy system was integrated onto four unique vehicle platforms over the course of the program, reusing common components and replacing only those tied to a specific platform.

The initial version of RTK, a collection of perception, control, and autonomy components enabling various mission behaviors, was developed under the DSAT program. Since then (and since the overview presented in [3]), RTK has provided the basic autonomy system and many of the building blocks for an array of TARDEC UGV programs, including Manned-Unmanned Teaming (MUM-T), Multi-UGV Extended Range (MUER), Wingman, Wireless Aerial Sensor Platform (WASP), Tactical Operation of a Remote Vehicle in a Contested Environment (TORVICE), the Tactical Resupply component of Autonomous Ground Resupply (AGR-TR), and many others.

Each of these projects and programs has its own mission requirements, but the re-use of a common set of components even across Army, Navy, and international collaborative projects has enabled each to use its financial and personnel resources for new capabilities instead of building up an autonomy system or vehicle platform from scratch.

Rise of ROS

The Robot Operating System (ROS) was originally developed by a startup company called Willow Garage as a framework to program and control a specific robotic platform, the PR2. However, this framework rapidly became far more popular than the robot platform itself. The Open Source Robotics Foundation was founded to continue to develop ROS, and it has in the last ten years become the *de facto* standard within the academic, government, and industrial robotics research and development community. While its primary use is for rapid prototyping, the last several years have seen a dramatic increase in its use on deployed robotic systems, even in safety-critical applications like factory automation.

The mass of researchers, students, and engineers using ROS also gave rise to thousands of robotics software modules, a significant subset of which can be modified or directly integrated for military applications. This existing base of expert users and software components made ROS even more attractive.

Given its global reach, enormous existing user base, and lengthening inroads into deployed applications, ROS was a clear choice for the basis of this collaborative robotics ecosystem. The success of the ROS-Industrial consortium, organized to meet the needs of the global industrial automation community by providing both robotic software components and a framework for precompetitive research and development, offered evidence that ROS would also serve the military robotics community. [1]

The initial ROS-M team made the early decision to focus on ROS 2, an evolution of ROS that promises, among other advancements, an update from the custom communications protocol developed at Willow Garage to the Data Distribution Service (DDS) standard. DDS is a middleware standard for distributed, robust, high-performance communication that uses the same publish-subscribe model employed by ROS 1. This similarity allows ROS users to realize the performance and robustness improvements of DDS without changing the communications paradigm of their entire robotic system. Using DDS offers ROS-M an immediate avenue toward interoperability with existing systems and a future path toward the kind of robust, safety-critical, secure systems required by military robotics.

Questions and Community

Once the feasibility and broad direction of this large-scale ecosystem project was agreed upon by all the members of the core team, the rest of the community of stakeholders needed to be consulted. Because ROS-M targets a DoD-wide audience, basing all design decisions on the wisdom of a small group, however expert, could not have the same impact.

To immediately provide access to a wide swath of industry stakeholders, TARDEC contracted with the National Advanced Mobility Consortium, NAMC, both to provide a larger team of active participants and to open the project to the entire membership of the consortium for input. Government stakeholders were also contacted, including research laboratories, service academies, and other research and development organizations.

The outcome of this effort was a series of questions or issues that ROS-M would have to answer or respond to for the project to achieve the envisioned success. These questions fell naturally into four categories, which formed the basis of investigation for the recently completed third phase.

Infrastructure. Contemporaneously with the solicitation of input, the ROS-M team established the concept of the “ROS-M Hub” to encompass the infrastructure needed to enable the envisioned ecosystem. This hub included a software repository, registry, validation and continuous integration pipelines, and documentation. Available as hosted or standalone services, these capabilities would establish an available central location for ROS-M activity without precluding the federated model so often successful in open-source efforts like ROS itself.

Of these five services, the two found to require early prototyping were the repository, a location to host source code, compiled binaries, configuration, and documentation; and the registry, an inventory of all available ROS-M software. To enable the aforementioned federated model, the registry would allow links to both the ROS-M repository as well as other hosting solutions, such as open services like GitHub or GitLab and private, internal repositories for more sensitive artifacts. Metadata elements were defined for registry entries to help users find, evaluate, and select software to use in their systems. Almost all of these elements were adopted in the prototype and are described briefly below.

In addition to the registry and repository, validation was identified as a service that did not require prototyping to prove the overall utility of ROS-M, but that offered opportunities for ROS-M to contribute to commonality among the community by providing automated feedback for software against various standards used for military software, such as the IOP (interoperability profiles) or VICTORY (Vehicular Integration for C4ISR/EW Interoperability) standards.

Seed Software. The ROS-M registry and repository would both require exercising to validate their utility and uncover gaps in functionality. No concrete choice was made of which software suite would best accomplish this evaluation, but its characteristics were established.

Any seed software would need to be a reasonably complex system to exercise the searching, filtering, and dependency tracking in the registry. Using existing software, as opposed to developing new capabilities specifically for ROS-M, would help the team understand the effort needed to apply the ROS-M requirements already established, especially the intended requirement to base all ROS-M software on ROS 2.

Cybersecurity. Existing confidentiality, integrity, and availability requirements were investigated, specifically focusing on policies, standards, and regulations related to the National Institute of Standards and Technology (NIST) and DoD Risk Management Framework (RMF). A preliminary analysis of ROS-SE (ROS with Security Enhancements) and the DDS-Security profile (a secure extension to the ROS 2 communication layer) was performed with respect to these requirements.

ROS-SE, sometimes known as SROS, is a set of security enhancements for ROS 1, including TLS support for node communication, certificate-based trust mechanisms, and AppArmor profiles to secure the system on which ROS is run. In short, it applies basic cybersecurity best practices to the existing ROS 1 infrastructure. DDS-Security is a profile of DDS (the communications middleware used by ROS 2) that integrates similar best practices, including identification and authentication, access control, integrity, and confidentiality.

Of the requirements investigated, integrity was identified as the most important security objective for ROS-M itself. This factor includes integrity of source code or compiled binaries in repositories, integrity of running software on a system, integrity of information passed among running nodes, and access controls within both the registry and repository.

Given the planned scope of ROS-M, implementation of specific cybersecurity standards was (and is) left to programs developing software, but the surveyed stakeholders recommended that ROS-M provide as much assistance as possible in the way of tools, best practices, and training to facilitate these developments.

Business Processes. A key factor in the success of ROS-M will be its long-term sustainment plan. ROS-M should be useful horizontally across services and organizations and vertically from basic research to acquisition and sustainment. Therefore, a broad sustainment model is needed.

Existing open-source projects and open standards were investigated both within and without the defense robotics community, including published robotics strategies from the U.S. Army, Navy, Air Force, and Department of Energy. It also considered other successful open ecosystems, such as the Future Airborne Capability Environment (FACE), UxS Control Segment (UCS), and ROS-Industrial (ROS-I). Finally, it made specific recommendations for decision-making in the prototype phase.

Representative recommendations included the establishment of a ROS-M steering committee; development of a quantitative business case; coordination with other open ecosystems; creation of training materials; identification of incentives and disincentives; and prototyping legal language (both license and contract) for use in research, development, and acquisition.

PRESENT

After the parameters of further investigation and development were established, work began in 2017 to provide a prototype of the complete ecosystem that would identify remaining gaps while also demonstrating the feasibility of ROS-M as a collaboration platform. This work focused on the four major areas identified as crucial for realizing the success of the concept defined at the outset of the project: infrastructure, seed software, cybersecurity, and business processes.

Infrastructure

Because the core concept of ROS-M is that of an open, available registry and repository for military robotics software, the infrastructure to support those functions was of the highest importance for the prototype. Three components were judged most crucial to complete: the repository for holding software and other artifacts, which would be used for the seed software and its accompanying documentation; the web-based registry for submitting, approving, validating, and finding ROS-M components; and the requirements for packages to be submitted to ROS-M for inclusion in the registry.

Repository. After a survey of available options, including custom solutions and ROS-M specific hosting on the Amazon Web Services GovCloud environment, the Defense Intelligence Information Enterprise (DI2E) environment was selected to host the software repository. As an established collaboration and integration environment for DoD software, it natively provided many of the identified ROS-M requirements. In particular, DI2E supports two-factor authentication via a Common Access Card, sophisticated access control capabilities, and availability via the public Internet. These properties together ensured that all ROS-M participants, from university researchers to defense personnel, would all be able to collaborate in the same environment without compromising the security or integrity of any artifacts, components, or teams.

Registry. No similar off-the-shelf solution was found for the registry that satisfied all of the identified requirements. Therefore, a custom prototype registry was developed with a Web-based user interface and a traditional database backend. The registry implemented basic access controls and manual package submission and uploading to provide a low-cost minimal working example. Frameworks to expand the registry's capabilities to satisfy all of the goals of ROS-M were also incorporated.

Metadata. Finally, the metadata requirements were established for submission to and inclusion in the ROS-M registry. These metadata serve as the primary mechanism of communication between package authors and maintainers, making their thorough definition critical to the practical utility of the registry. Native ROS packages require package metadata for a similar reason: to communicate to users of the ROS package. The ROS-M metadata, however, included seven additional components to help developers communicate with users through the registry: maturity level, development standards and processes, validation and certification status, package compatibility, tutorials, issue tracker, and projects and programs.

Of these metadata fields, the first one, maturity level, bears additional attention. To advance the ROS-M goal of increasing the pace of progress in research as well as development and deployment, software components at many different stages of development may be publicized through the registry. Therefore, three discrete levels of software maturity were defined to help identify how stable, tested, documented, and well-used a particular package is. This system was heavily influenced by a similar model used by ROS-Industrial. A ROS-M software component can have one of three maturity levels: experimental; developmental; and production. Experimental software is under active development. While it is sufficiently stable and useful to be distributed to outside users, it should not be expected to have stable APIs, thorough documentation, or have undergone rigorous testing. Developmental software should have rectified each of those three deficiencies: it should have stable APIs; those APIs and the package's general usage should be thoroughly documented; and the package should have been tested in both automated tests and real systems. It is expected that the vast majority of ROS-M software will have this maturity level. Finally, production software has been used on systems that have undergone formal testing, validation, or certification by one or more third parties and have been used in realistic conditions.

There are no formal requirements or automated validation tools to determine the maturity level of a ROS-M package; like applicable standards or development processes, it is one of many signals for potential users to evaluate software at the registry stage before acquiring and integrating it into their systems.

Another metadata field, not unique to ROS-M, nonetheless has important implications for both producers and consumers in the ROS-M ecosystem. Each package may have one or more software licenses associated with it, identified by an appropriate tag within the metadata. ROS-M imposes no requirements on the licenses under which software can be included, allowing both copyleft licenses like the GNU General Public License and fully proprietary software and everything in between. This permissiveness will allow the registry to become more universal in scope, including as much software as possible and thereby attaining the greatest utility possible. If a ROS-M consumer desires to incorporate a package with a nonfree license, they may contact the package's author or maintainer (both also required metadata fields) to inquire about licensing arrangements.

Seed Software

To demonstrate the use of the infrastructure and to test the requirements of submission to the registry, two sets of seed software were prepared during this phase as the initial contents of ROS-M.

The first was the Maverick path planner [4], a fast, sophisticated planner for unstructured environments developed for RTK. Maverick was chosen for two reasons: first, as an advanced path planner integrated into a complete autonomy system, its complexity would effectively benchmark the difficulty of porting software to ROS-M; and second, because new behaviors for autonomous systems typically require new or modified path planning systems, a path planner was a natural choice to investigate the collaborative possibilities enabled by ROS-M. The entirety of Maverick, including all of its dependencies within RTK, were adapted and submitted to ROS-M.

This adaptation was performed in two phases. First, Maverick and its dependencies were converted to ROS 2. The porting process revealed gaps between the functionality made available in ROS 2 compared to ROS 1, and additional tools and libraries were developed to bridge those gaps. In particular, parts of RTK rely heavily on ROS nodelets, software components that share memory for efficient zero-copy message passing; ROS parameters, dynamic configuration variables set at runtime instead of compile-time; and the ROS launch system, a structured description of which nodes and nodelets to activate on system startup. To successfully port Maverick, each of these functions was first replicated in whole or in part within the ROS 2 framework. The additional capability development validated the choice of Maverick, because it outlined the upper end of the required effort to port a complex software component.

The second phase of ROS-M conversion required providing, for Maverick and each of its dependent components, the additional metadata identified above.

Along with the Maverick components that were converted to ROS 2 and adapted for ROS-M, the entirety of RTK Core, a stable, distribution-ready release of RTK, was adapted for inclusion in the registry. Instead of porting the entire codebase of nearly two hundred ROS packages, the metadata for each package was augmented to fulfill the registry requirements. (With one caveat: technically, the metadata requires the identification of which version of ROS 2 the package supports; however, because the remainder of RTK Core does not support ROS 2 at all, this field was populated with the ROS 1 version supported, namely ROS 1 Indigo Igloo.) This adaptation allowed RTK Core to stress the infrastructure software without incurring the engineering cost of porting the whole system.

The other set of seed software ported to ROS-M was a component of a ground vehicle formations library, intended to localize a single vehicle as part of a group. The autonomy system from which this component was chosen does not share a common background with RTK, so selecting this component for submission to ROS-M engaged some of the collaborative aspects of the infrastructure, namely searching across submissions and sharing components among projects.

Cybersecurity

All modern robotic systems share a set of cybersecurity concerns, from typical software vulnerabilities up to adversarial machine learning-based exploits to spoof sensor input and disable or even mislead the automated vehicle. The potential consequences of these vulnerabilities can be dire, and individual systems must identify their own weaknesses and guard against attack.

However, because the ROS-M project primarily provides the organization and infrastructure to support the sharing and re-use of robotic components and not the components themselves, its concerns with respect to cybersecurity were not to solve its systems' vulnerabilities, but to offer a set of basic best practices and a suggested set of tools to achieve them.

The primary concern for the registry and repository, as discussed above, was access control for the ROS-M registry and repository to ensure confidentiality, integrity, and availability of those systems. The repository was successfully hosted on DI2E, addressing each of these concerns for that system. Similar processes are planned for the registry.

An initial set of coding standards was also identified to provide a baseline against which ROS-M software components could operate. The chosen standard is the Carnegie Mellon University Software Engineering Institute CERT C++ (CMU-SEI CERT C++) standards, coupled with a suite of static analysis tools based on the U.S. Navy's Statick tool to assist developers in following them.

Finally, to validate the recommendation for the DDS-Security profile, a man-in-the-middle attack was successfully demonstrated on an unsecured ROS 2 system simulating an automated vehicle driving a planned route. The simulated vehicle was made to modify its repeated path in response to unauthorized commands. It is important to note that this result was expected, as ROS 2 deliberately has no built-in security mechanisms. The goal of the experiment was not to suggest a failing of ROS 2, but to identify a solution easily accessible to most developers. Implementing the DDS-Security profile on the same system successfully denied the attack.

Business Processes

The healthy software ecosystem encouraged by ROS-M requires more than robust, secure infrastructure; high-quality software; and a reasonable, common set of cybersecurity best practices. Sustaining the advantages of this openness requires broad and diverse participation to ensure its utility to all members.

Therefore, a Business Guide was developed—a reference for DoD executives, military executive officers, and senior leadership from both government and industry to help understand the value proposition of ROS-M. The Guide provides an overview of the various components of ROS-M, an understanding how these components impact various stakeholders of the Unmanned Autonomous Systems (UAS) community, and suggested best practices for including ROS-M requirements in program specifications.

To help mitigate the potential complexity of including the open software ecosystem of ROS-M in the acquisitions process, the Business Guide also includes a Contract Guide Template—suggested language for a Contracting Officer or Program Manager to include in a solicitation. Suggested language is provided for the Statement of Work, Technical Specification, Contract Data Requirements List, Evaluation Factors for Award, and others. The Guide offers further suggestions for language regarding rights in technical data and computer software to support the overarching ROS-M goals of reuse of software.

The Business Guide provides education about the vision and potential of ROS-M to those who are new to the program while also offering the tools necessary to realize those benefits within a particular program or project.

FUTURE

The prototypes systems detailed above were demonstrated in early 2018 to interested stakeholders. The populated registry was shown to enable component searching, evaluation, and discovery as intended. The seed software was demonstrated on multiple independent vehicle platforms. Because entire autonomy systems were not ported to ROS-M, each vehicle ran a hybrid software solution incorporating the ROS-M–ready components alongside the pre-existing software. To further demonstrate the interoperability potential of ROS-M systems, a portion of the demonstration was operated on both vehicles from a single OCU. In conjunction with the practical demonstrations, the cybersecurity tools and best practices and the ROS-M Business Guide were presented for consideration and feedback.

In the near term, development will continue on the infrastructure systems to ensure a stable, capable system for promoting, searching, finding, evaluating, and accessing vehicle autonomy software. Automation features for uploading, validating, and processing package metadata will be implemented from the designs developed according to the considerations discussed above.

The conversion of RTK to ROS-M will be completed to ensure that the registry contains a complete autonomy system for other participants to draw from and an exemplar of the diversity of software and quality of documentation expected and available in ROS-M. When ROS-M is made available to the wider community, it will have immediate utility for users, and a broad base on which for organizations to build.

Looking to expand beyond the core development team, the ROS-M stakeholders have identified high-value robotics and autonomy related projects to continue to expand the registry. In particular, the research products of the Robotics Collaborative Technology Alliance (RCTA) would both provide significant capability enhancement to the autonomy suites already present in ROS-M, but also test the software, tools, and processes developed around inclusion in the ROS-M registry and storage in the repository. These capabilities will provide immediately usable components for other robotics systems, and ROS-M will help organize, categorize, and share those components among the projects and programs that can benefit from them most.

The future potential of ROS-M relies not so much on software inclusion, however, as the building out of the community of interest using ROS-M as it is intended: as a starting point for as many automated or unmanned ground vehicle robotics projects as possible, and as a library of capabilities to enhance existing or future projects with as little duplication of effort and as much software re-use as can be achieved. The technical feasibility has already been demonstrated by the activities described in this paper. Most future work will focus not on the technical aspects, but on the community-building aspects, including as many members of the military robotics community as possible to ensure that the benefits of increased openness and communication among robotics software developers are shared universally.

REFERENCES

[1] Edwards, Shaun, and Chris Lewis. "ROS-Industrial: Applying the Robot Operating System (ROS) to Industrial Applications." In *IEEE Int. Conference on Robotics and Automation, ECHORD Workshop*. 2012.

[2] Gonzales, Daniel and Sarah Harting, “Designing Unmanned Systems with Greater Autonomy”, RAND Corporation, Santa Monica, 2014.

[3] Kania, Robert et al., “Dismounted Soldier Autonomy Tools (DSAT)—From Conception to Deployment”, *NDIA Ground Vehicle Systems Engineering and Technology Symposium (GVSETS)*. 2014.

[4] Seegmiller, Neal, Jason Gassaway, Elliot Johnson, and Jerry Towler. "The Maverick planner: An efficient hierarchical planner for autonomous vehicles in unstructured environments." In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pp. 2018-2023. IEEE, 2017.

[5] “The Role of Autonomy in DoD Systems”, Defense Science Board, 2012.