2015 NDIA GROUND VEHICLE SYSTEMS ENGINEERING AND TECHNOLOGY SYMPOSIUM
AUTONOMOUS GROUND SYSTEMS (AGS) TECHNICAL SESSION
AUGUST 4-6, 2015 - NOVI, MICHIGAN

ANVEL-ROS: THE INTEGRATION OF THE ROBOT OPERATING SYSTEM WITH A HIGH-FIDELITY SIMULATOR

Christopher R. Hudson
Alexander Lalejini
Brandon Odom
Cindy L. Bethel, PhD
Computer Science and Engineering Department
Mississippi State University
Mississippi State, MS

Daniel W. Carruth, PhD
Center for Advanced Vehicular Systems
Mississippi State University
Mississippi State, MS

Phillip J. Durst
Christopher Goodin
US Army Engineer Research and Development Center
Vicksburg, MS

ABSTRACT

The objective of this research was to survey and evaluate simulators for use with unmanned ground vehicles and extend the functionality of the ANVEL simulator to include the robot operating system (ROS). The goals of the research were first to determine how the ANVEL simulator would compare to four other currently available simulators on four criteria: physical fidelity, functional fidelity, ease of development, and cost. The second goal was the development of an ANVEL-ROS bridge to expand the robot control functionality in ANVEL. The ANVEL-ROS bridge developed was verified using two robots, Turtlebot2 and Jaguar V4. RViz, a ROS visualization tool, was used to confirm sensor output correctness. Robot control in ANVEL was confirmed using tele-operation through ROS commands.

INTRODUCTION

There is an increase in the prevalence of unmanned ground vehicles (UGVs) due in part to the US Army's current and future plans to integrate UGVs into tactical operations [13]. Due to the high cost of these systems, there is an increased need for accurate simulations of UGVs to provide affordable methods for testing and evaluation. Significant research efforts have been devoted toward the design, implementation, and validation of realistic unmanned ground vehicle simulators [1-7].

While the designers of these simulators have worked to adapt to the needs of the community, each simulator has strengths and weaknesses leaving application gaps that have yet to be addressed. For example, accurate modeling of vehicle mobility in outdoor terrains has not yet been addressed by many of the currently available simulators [3-4, 6]. Simulators are often designed with specialized protocols to cater toward problem-specific requirements, such as SARGE's focus on search and rescue and multi-user support [4-5], MIX Testbed's focus on distributed warfighter support [3], and ANVEL's focus on unmanned ground vehicle outdoor terrain mobility modeling [7]. These problem-specific approaches, do not facilitate the direct transfer from simulation to robotic systems used in field and lab environments. It would be beneficial to standardize UGV simulations through the use of existing robot control protocols and develop standard interfaces between the UGV simulators and production-level robot control system software. This would maximize the effectiveness of simulations and provide an improved mechanism for testing and evaluation of UGV systems. This research focuses on the development of a bridge between the Robot Operating System (ROS), a standardized software framework for robotic systems, and ANVEL, a high-fidelity simulator for UGVs. This bridge allows for ANVEL to be tightly integrated with ROS, combining an advanced simulator with a framework widely used in both academia and industry [9].

RELATED WORK

Through the development and use of accurate robot simulators, researchers can address critical issues, such as sensor utilization and navigation, without the need to interact with physical hardware. The use of simulators accelerates research developments and reduces the costs associated with robot and sensor purchases. In the following section, we describe five simulators *(Gazebo, USARSim, SARGE, MIX Testbed,* and *ANVEL)* available to researchers; however, each is limited in its capabilities and often fail to address the full range of needs required by robot developers (refer to Table 2). In order to fully evaluate these simulators, it was decided to use the criteria presented by Craighead *et al.* in [6].

The four criteria suggested for the evaluation of the fidelity of simulators were physical fidelity, functional fidelity, ease of development, and cost [6]. Physical fidelity is described as the degree of user immersion in the virtual environment. Ease of development is described in terms of four key factors, which include the ability to create new situations inside a simulator; the ability to implement new objects in the simulation; the supported development languages; and the available documentation. Cost is described as the investment – both financial and time – required for use of the simulator [6]. The four criteria provided by Craighead *et al.* form a basis for the evaluation of current and future UGV simulators [6]. The criteria put forth by Craighead *et al.* in [6] for each fidelity rating is summarized in Table 1. A summary of the five simulators discussed in this paper and their evaluation based on the criteria presented in Table 1 is shown in Table 2.

A simulator commonly used by the ROS community is *Gazebo,* an open-source 3D simulator, which is tightly integrated with ROS, and is designed to incorporate a third dimension to the 2D simulator *Stage.* Gazebo has been used as a simulation package in support of the DARPA Virtual Robotics Challenge [1]. Gazebo uses the OGRE 3D graphics engine and supports the following physics engines: Open Dynamics Engine (ODE), Bullet, Simbody, and the Dynamic Animation and Robotics Toolkit (DART). Gazebo boasts a large development community, and its tight integration with the Robot Operating System (ROS) allows ROS robot control software to easily transition from Gazebo to a real robot. Gazebo also allows on-the-fly robot component drag-and-drop functionality and provides access to an online database of community created models [11]. However, Gazebo's design is targeted for smaller populations of robots limiting its ability to simulate a large number of robots [8].

	Physical Fidelity	Functional Fidelity	Ease of Development	Cost
High	High quality visuals / audio	Forces on individual objects	Several languages / Object importing / documentation	Expensive / Hard or time consuming installation
Medium	3D visuals/ Limited audio	Forces on vehicle as a whole	Some languages / Some importing / Some documentation	Low cost / Easy installation
Low	2D / No audio	No simulation of forces, only velocity	No documentation / No importing	Free / Easy installation

Table 1: Evaluation criteria for robot simulators from [6].

The open-source simulator, *Unified System for Automation and Robot Simulation (USARSim),* specializes in urban search and rescue simulations but has been applied to the simulation of other environments and tasks [2]. It leverages the Unreal Developers Kit from Epic Games [2], making it affordable and widely available [6]. USARSim has been used in applications such as RoboCup Virtual Robot Rescue Competition, the IEEE Virtual Manufacturing and Automation Challenge, and the DARPA Urban Challenge [2]. USARSim is platform independent but runs on an outdated version of Unreal Engine. Therefore, other engines may provide improved visual fidelity and more accurate physics models.

Simulator	Physical Fidelity	Functional Fidelity	Ease of Development	Cost
Gazebo	Medium	Low	High	Medium
USARSim	High	Medium	Medium	Low
SARGE	Medium-High	Medium-High	High	Low
MIX	Medium	High	Medium	Low
ANVEL	High	High	Medium	Low

Table 2: Criteria scores for each of the evaluated simulators.

The *Search and Rescue Game Environment (SARGE)* provides a platform that enables easy access to custom, immersive environments in which multiple operators can interact with each other in a multiplayer environment [5]. *SARGE* makes use of intelligent tutor techniques to assess users' skills at given tasks and uses these tutor techniques to alter the simulation's interactions with a particular user to better facilitate improvements [4-5]. SARGE has good API documentation and examples and has an active developer community [12]. SARGE also provides a drag-and-drop environment editor [12]. However, SARGE has limited sensor support [8].

The *Mixed Initiative Experimental (MIX) Testbed* is an open-source simulation testbed developed at the University of Central Florida that addresses the need for increased warfighter support with respect to robotic and unmanned military equipment. The MIX Testbed allows users to send mission plans, tele-operate unmanned ground vehicles with a joystick, and receive video feed from a camera payload. The simulator also supports data and event logging as well as neurophysiological devices for measuring a user's state to better support experimentation [3]. Simulation scenarios are able to be run on one or multiple computers within the MIX Testbed as needed by the scenario [10].

ANVEL is a simulator designed and implemented by Quantum Signal (QS) for use by the US Army Engineer Research and Development Center (ERDC). It boasts a high fidelity simulation that models physics in detail and provides a high quality graphical visualization of the simulated environment [7]. ANVEL provides drag-and-drop environment editing during a simulation scenario, and it integrates an advanced mobility model for wheeled vehicles on a variety of terrains. While ANVEL provides innovative capabilities for simulation, to maximize the benefit to the robotics community, it should support integration with existing robot control software, such as ROS. This paper presents an approach to the integration of ROS with ANVEL.

ROS is an open-source framework used for robot control [9]. ROS is not a traditional operating system; it operates within a host environment (typically Linux). The ROS framework provides support for a topic-based publish-subscribe model and a service-based request and reply model. Processing in ROS is accomplished within modules called

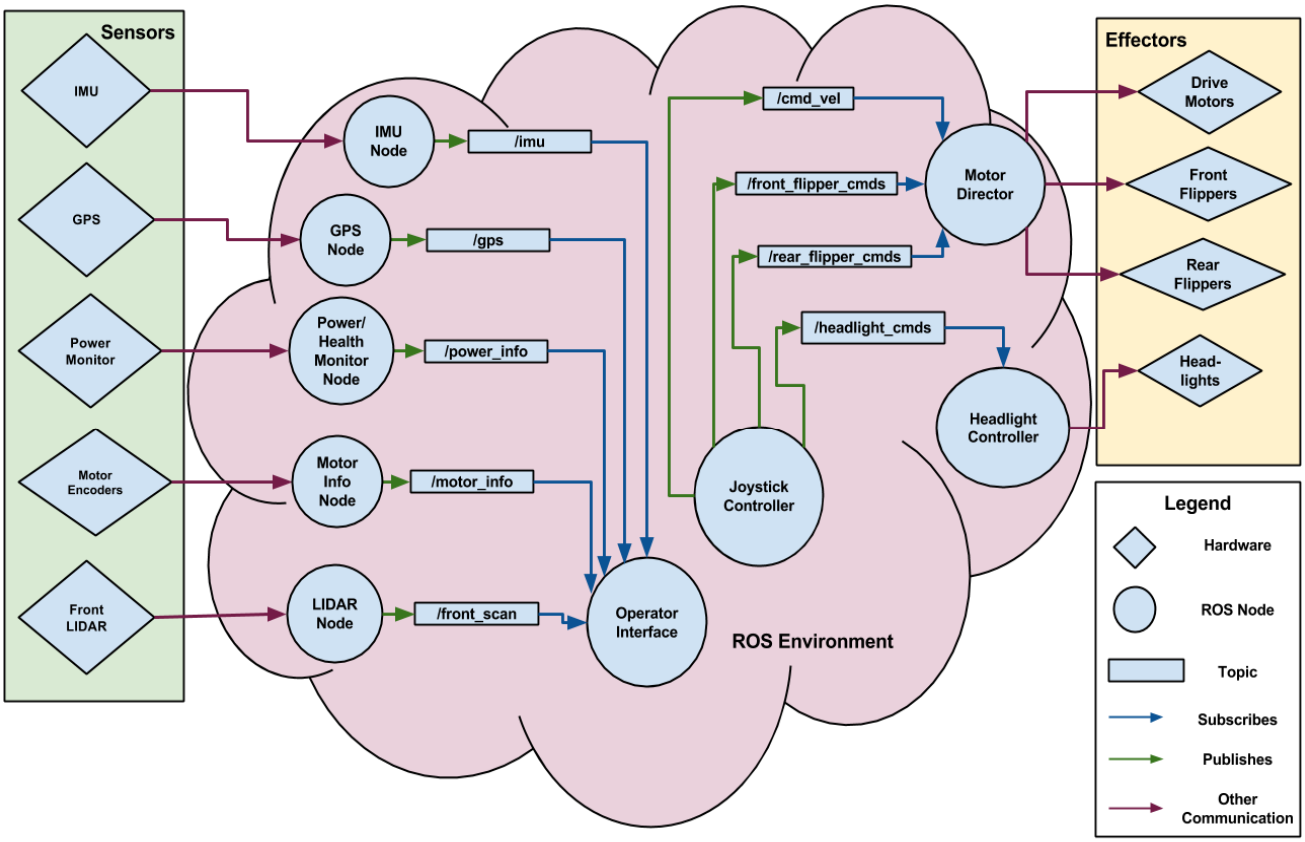

Figure 1: An example ROS Software System Architecture

nodes. ROS leverages the topic-based publish-subscribe model to provide named data pathways among distributed nodes. The service-based request and reply model is implemented as services; nodes within the ROS environment may provide functions, or services, that can be called through ROS protocols by any other node running in the ROS environment. A new node interacts with the ROS framework by subscribing to and processing existing topics and/or publishing new data over topics.

This form of interaction enables nodes to remain unaware of the activities of the other nodes within the ROS ecosystem. A node that is subscribed to and receives data over a particular topic need not be aware of the node or nodes publishing data over that topic; likewise, it is unnecessary for a node publishing data over a topic to be aware of other nodes subscribed to that particular topic. Because of the data-agnostic nature of ROS nodes, ROS allows software to be implemented in a highly modular and distributed manner to facilitate the integration of ROS with simulators.

The simulator provides data interfaces that act as replacements for standard ROS nodes. The remaining nodes in ROS are unaware of whether the data is provided by a simulation or by actual hardware. This maximizes transferability by allowing robot software to be developed in simulation and smoothly transitioned to physical hardware. The only modification required is the replacement of the simulation nodes with the physical hardware nodes.

Figure 1 shows an example of a simple ROS software architecture for an unmanned ground vehicle. In this diagram, a joystick controller is used to control a robot, and sensor data topics are displayed to an operator through an operator interface node. The transferability maximized by ROS allows the same joystick and operator interface nodes to function without modification on the pictured robot system as well as the simulated robot system. The operator interface is not aware of how data is published to the topics it subscribes to just as the joystick controller is unaware of how the data it publishes is used. A simulation models data interfaces to each of the sensors and effectors pictured in Figure 1. These data interfaces replace the ROS hardware drivers (*e.g.* IMU Node, GPS Node, *etc.*) to transition from a real world robot to a simulated robot.

METHODS

This research introduces a framework to replace ROS hardware interfaces with a method to connect to the ANVEL simulator. This ANVEL-ROS (A-ROS) integration implements a bridge to transfer data to and from the simulator (see Figure 2). The bridge consists of two core segments: an ANVEL plugin to publish simulated hardware data and a ROS node to receive, process, and publish the information using standard ROS topics.

To allow for the flow of information to and from the ANVEL simulator, sensor data must be accessible. In ANVEL sensors are objects which produce information, depending on what they are designed to perform. ANVEL has several generic sensors available within the simulation, including cameras, LIDAR, GPS, and IMU. The A-ROS bridge accesses data from each of these sensors and packages it into an XML format to be sent from the ANVEL simulated robot to ROS. On the ROS side, packets of data are received. The node parses the XML tags to determine the appropriate handler for the data type. The handler converts the XML data into a ROS message that gets published to a topic.

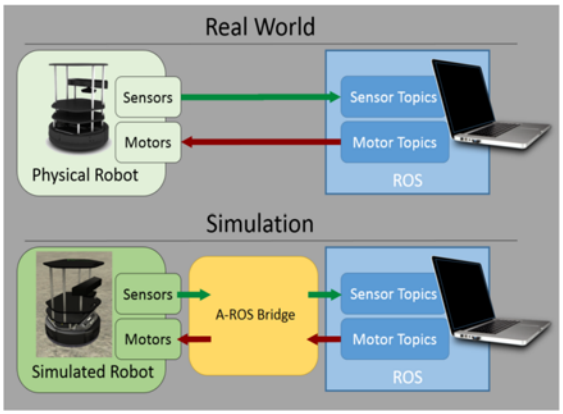

Figure 2: Diagram of A-ROS bridge.

Sensors

Sensors in ANVEL are designed to be generic in nature. Sensor parameters are defined in XML or modified in simulation. Sensor parameters and, in some cases, sensor data are accessible through a property management system. This allows the user to study the effects of sensors with different fidelities on their robotic systems, and analyze how different sensors impact performance. Since the sensor fidelity is defined within the ANVEL simulation, sensor fidelity within the ANVEL simulation can be easily modified without a need to notify ROS of the change. ANVEL provides a subset of possible sensors for developers to use.

IMU and GPS

The inertial measurement unit (IMU) and global positioning system (GPS) are commonly used sensors on robotic systems. The IMU collects data on velocity, orientation, and forces. The GPS collects information on position in an outdoor environment. ANVEL provides a generic implementation of both of these sensors. The A-ROS bridge implements a protocol for extracting this information from ANVEL's internal property management system. On the ROS side, the IMU data is published as a *sensor_msgs/Imu* ROS message,

and the GPS data is published as a *sensor_msgs/NavSatFix* ROS message.

Camera

The A-ROS bridge implements a single lens color camera, which is a subset of the camera models available within the ANVEL simulation. Modifications to the camera model are required to access the simulated video data, encode it into a video stream, and send it over the network to ROS. Once it is received in ROS, the video stream is decoded and published to a standard ROS topic for cameras. These messages can then be read into a visualizer, RViz, within ROS (see Figure 3).

Figure 3: ANVEL simulator (left); a ROS camera feed in RViz (right).

LIDAR

A generic LIDAR model which is a subset of LIDAR sensors within the ANVEL simulation is implemented in the A-ROS bridge. Modifications to the LIDAR model are required to access the simulated laser scan data, package the data, and send it over the network to ROS. The ANVEL LIDAR sensor simulates LIDAR by generating multiple rays, firing these rays out into the environment for a predefined distance, and detecting possible objects that may result in collisions. The number of rays and the distance each ray travels is determined by the sensor definitions for the LIDAR sensor being used. A modification of the LIDAR sensor is necessary to access these collision coordinates which are sent to the native LIDAR visualizer. These modifications take the point cloud data and builds the corresponding laser scan distance data. Once it is received in ROS, the data is published to the standard ROS topic for laser scans, *sensors_msgs/LaserScan*. These messages can then be read into a visualizer within ROS as shown in Figure 4.

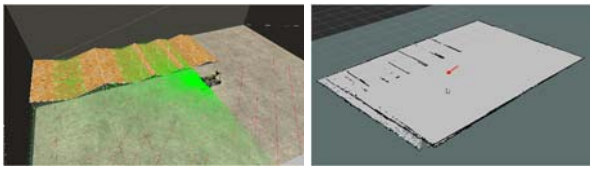

Figure 4: ANVELsimulator (left); A ROS LIDAR feed and map displayed in RViz (right).

ANVEL Plugin

ANVEL has a plugin framework that expands functionality without modifying the core software. The A-ROS bridge uses this plugin framework to manage communication between ANVEL and ROS. The plugin establishes a network connection with a ROS node that handles sending and receiving data between ROS and ANVEL. This connection allows for basic sensor information pulled from the property management system to be transmitted to the ROS simulation, where it is translated and published into appropriate ROS topics. This basic information includes odometry for a vehicle, IMU sensor data, and GPS sensor data. However, it can be expanded to pull any other sensor information the user might require from the property management system. More complex sensors, such as camera or LIDAR sensors, require a more complex plugin to extract the video and point cloud data, which is not mutable or publishable to the property management system.

ROS Nodes

The ROS side of the ANVEL-ROS integration is a collection of ROS nodes that manages communication between ROS and the ANVEL simulator. XML packages of sensor data received from ANVEL are parsed, processed using an appropriate handler, and published over standard ROS topics by an ANVEL listener node. The ROS listener node subscribes to topics relevant to the control of robots in ANVEL, translates the messages received from those topics into a format ANVEL can accept, and transmits that data to the ANVEL plugin. All of the communications between ANVEL and the A-ROS bridge occurs through network sockets.

Drive and Servo Motors

The A-ROS bridge supports two forms of robot actuation control from ROS: drive motor control and servo motor control. In ROS, it is standard for robot base movement to be controlled by *geometry_msgs/Twist* messages, which describe movement in the form of angular and linear velocities in three degrees of freedom. ROS currently has no standard message format for the control of servo motors; for convenience and readability, robot servos in ANVEL are currently controlled using a custom ANVEL servo message type that defines the servo behavior.

RESULTS AND CONCLUSIONS

We verified the A-ROS bridge in two ways. First, we implemented a subset of sensors in ANVEL on two robotic systems: the TurtleBot 2 and the Jaguar V4. Second, the sensor data output was visualized using the ROS visualization tool, RViz, to confirm that the correct data output was obtained from each sensor implemented. Successful visualization of core sensor data (LIDAR, camera, and odometry) has been accomplished. Support for other sensors include GPS and IMU. Depth data from the ANVEL LIDAR sensor was successfully used with Gmapping, a ROS simultaneous localization and mapping tool, to build a map of

an ANVEL virtual environment with ROS. Teleoperation of the robots was performed through a ROS interface for both the Turtlebot2 and Jaguar V4 robots. This operation was verified in simulation, through correct motor command execution.

The current version of the A-ROS bridge supports publishing camera streams, IMU, GPS, LIDAR, and odometry sensor data from ANVEL to ROS. Currently, ROS publishes vehicle drive commands and servo motor data to ANVEL using the A-ROS bridge. We are currently working to implement additional sensors in ANVEL and provide standard interfaces to ROS.

This bridge significantly expands the capabilities of ANVEL by enabling it to use existing robotics packages in the ROS framework. The added compatibility with ROS provides benefit to both the ROS community, in the form of a new high-fidelity simulator, and to ANVEL, which is now able to use complex packages in the ROS framework for testing robotic simulation, designing new control algorithms, and testing new types of sensors in a standard framework.

ACKNOWLEDGEMENTS

Material presented in this paper is a product of the CREATE-GV Element of the Computational Research and Engineering Acquisition Tools and Environments (CREATE) Program sponsored by the U.S. Department of Defense HPC Modernization Program Office. The authors would also like to acknowledge the efforts of John McGinley, Ryan Smith, Dexter Duckworth, and Daniel Sween that provided support critical to the success of this project.

REFERENCES

[1] C.E. Aguero, N. Koenig, I. Chen, H. Boyer, S. Peters, J. Hsu, B. Gerkey, S. Paepcke, J.L. Rivero, J. Manzo, E. Krotkov, G. Pratt, "Inside the Virtual Robotics Challenge: Simulating Real-Time Robotic Disaster Response," *IEEE Transactions on Automation Science and Engineering*, April 2015, vol.12, no.2, pp.494-506.

[2] S. Balakirsky, Z. Kootbally, "USARSim/ROS: A Combined Framework for Robotic Control and Simulation," in *ASME/ISCIE 2012 International Symposium on Flexible Automation*, 2012, pp. 101-108.

[3] D. Barber, L. Davis, D. Nicholson, N. Finkelstein, J. Y.C. Chen, "The Mixed Initiative Experimental (MIX) Testbed for Human Robot Interactions With Varied Levels of Automation," in *26th Army Science Conference (2008)*, 2008.

[4] J. Craighead, "Distributed, Game-based, Intelligent Tutoring Systems — The Next Step in Computer Based Training?," *Collaborative Technologies and Systems, 2008. CTS 2008. International Symposium on*, Irvine, CA, 2008, pp. 247-256.

[5] J. Craighead, R Gutierrez, J. Burke, R. Murphy, "Validating The Search and Rescue Game Environment As A Robot Simulator By Performing A Simulated Anomaly Detection Task," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nice, France, 2008, pp. 2289-2295.

[6] J. Craighead, R. Murphy, J. Burke, and B. Goldiez, "A Survey of Commercial and Open Source Unmanned Vehicle Simulators," in *2007 IEEE International Conference on Robotics and Automation*, 2007, pp. 852-857.

[7] P. J. Durst, C. Goodin, C. Cummins, B. Gates, B. McKinley, T. George, *et al.*, "A Real-Time, Interactive Simulation Environment for Unmanned Ground Vehicles: The Autonomous Navigation Virtual Environment Laboratory (ANVEL)," in *2012 Fifth International Conference on Information and Computing Science (ICIC)*, Liverpool, UK, 2012, pp. 7-10.

[8] A. Harris and J.M. Conrad, "Survey of Popular Robotics Simulators, Frameworks, and Toolkits," in Southeastcon, 2011 Proceedings of IEEE. 2011, pp. 243-249.

[9] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, *et al.*, "ROS: An Open-Source Robot Operating System," in *ICRA Workshop on Open Source Software*, Kobe, Japan, 2009, p. 5.

[10] A. Talone, T. Fincannon, D. Schuster, F. Jentsch, and I. Hudson, "Comparing Physical and Virtual Simulation Use in UGV Research Lessons Learned from HRI Research with Two Test Beds," in *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* Vol. 57, No. 1, 2013, pp. 2017-2021.

[11] D. S. Michal and L. Etzkorn, "A Comparison of Player/Stage/Gazebo and Microsoft Robotics Developer Studio," in *Proceedings of the 49th Annual Southeast Regional Conference*, ACM , 2011. pp. 60-66.

[12] J. Craighead, J. Burke, and R. Murphy, "Using the Unity Game Engine to Develop SARGE: A Case Study," *Computer, 4552*, 2007, pp. 366-372.

[13] J. A. Winnefeld, F. Kendall, "Unmanned Systems Integrated Roadmap FY 2013-2038," *Office of the Secretary of Defense. US*, 2011.