Robotics Collaborative Technology Alliance: An Open Architecture Approach to Integrated Research

Robert Michael S Dean[a1,] Charles A DiBerardino[a]
[a] General Dynamics Land Systems Robotics Division,1231 Tech Court, Westminster, MD, USA 21157-3029

ABSTRACT

The Robotics Collaborative Technology Alliance (RCTA) seeks to provide adaptive robot capabilities which move beyond traditional metric algorithms to include cognitive capabilities [1]. Research occurs in 5 main Task Areas: Intelligence, Perception, Dexterous Manipulation and Unique Mobility (DMUM), Human Robot Interaction (HRI), and Integrated Research (IR). This last task of Integrated Research is especially critical and challenging. Individual research components can only be fully assessed when integrated onto a robot where they interact with other aspects of the system to create cross-Task capabilities which move beyond the State of the Art. Adding to the complexity, the RCTA is comprised of 12+ independent organizations across the United States. Each has its own constraints due to development environments, ITAR, "lab" vs "real-time" implementations, and legacy software investments from previous and ongoing programs. We have developed three main components to manage the Integration Task. The first is RFrame, a data-centric transport agnostic middleware which unifies the disparate environments, protocols, and data collection mechanisms. Second is the modular Intelligence Architecture built around the Common World Model (CWM). The CWM instantiates a Common Data Model and provides access services. Third is RIVET, an ITAR free Hardware-In-The-Loop simulator based on 3D game technology. RIVET provides each researcher a common test-bed for development prior to integration, and a regression test mechanism. Once components are integrated and verified, they are released back to the consortium to provide the RIVET baseline for further research. This approach allows Integration of new and legacy systems built upon different architectures, by application of Open Architecture principles.

Keywords: Robotic Collaborative Technology Alliance, Robotics, Simulation, Middleware, Open Architecture

1. INTRODUCTION

System integration is a challenging problem for any complex system. Various techniques have been developed to simplify this process. Open Architecture of hardware and software components enables adding, upgrading and swapping of components. Further refinement of the concept adds loosely coupled components communicating through well-defined published interfaces. There are currently multiple established and emerging approaches to this within DoD depending on program needs such as: Joint Architecture for Unmanned Systems (JAUS) [2], Future Airborne Capability Environment (FACE) [3], and Vehicle Integration for C4ISR/EW Interoperability (VICTORY) [4]. Within each domain, interoperability is possible due to enforcement of common middleware and data models. While systems developed within each of these environments are Interoperable within that domain, achieving interoperability between them and with existing legacy investments is again a challenge requiring additional testing and verification.

RCTA faces these same issues, with the additional constraint that we wish the researchers to be as productive as possible. Therefore, enforcing common middleware and software frameworks are an impediment as it would take valuable research dollars away from the research. Replacing the researchers preferred environment with a new system also results in loss of time and program dollars while the new system is learned. As a result, research occurs in multiple development environments, using a variety of tools. For example, one of the consortium members is Jet Propulsion Labs (JPL) who in the past transitioned RCTA developments to and from other DoD programs such as Legged Squad Support System. JPL has over the years developed real-time capable design practices and infrastructure to deal with the rigors of other worlds. On the other side of the spectrum are the University researchers who use less structured, non-real-time environments such as Robot Operating System (ROS) [5] and Matlab. To force JPL to use ROS would result in additional time cost when transitioning research, while forcing graduate

[1] *rdean@gdrs.com; phone 1 410 876-9200; fax 1 410 876-9470; www.gdrs.com

Open Architecture/Open Business Model Net-Centric Systems and Defense Transformation 2014,
edited by Raja Suresh, Proc. of SPIE Vol. 9096, 90960M · © 2014 SPIE
CCC code: 0277-786X/14/$18 · doi: 10.1117/12.2050366

students into real-time environments adds complexity to immediate task. Additionally, research environment changes based on research area and individual researcher preferences. For example, what is most efficient for Intelligence is not necessarily the best toolbox for Perception. It falls upon Integrated Research aspect of RCTA to find solutions which allow all consortium members to collaborate, integrate, and assess the joint technology in a single platform, and the data logs for evaluation by the Army Research Labs assessment team. Given integrated capability, this team collaborates to develop Integrated Research Assessment (IRA) scenarios consistent with transition to fielded solutions. Data from these IRAs are then reviewed to provide feedback to the consortium on how it is meeting expectations, documents research maturity, and is used to inform government leadership as to the status of the program. There are 4 basic elements to facilitate integration: RFrame, a Common World Model, RIVET simulator, and a common ITAR free Integration Platform.

2. RFRAME

The first element of the IR approach is the RFrame middleware layer which provides transport agnostic, data centric communications between modular components. RFrame was designed by surveying common interoperability approaches to identify common functionality and take into consideration lessons learned from GDLS's 23 years of real-time robotics experience. This survey included transports most common within the robotics community and potential transition to DoD programs: RCSLIB[6], DDS[7], JAUS, and ROS. Follow on reviews of LCM[8], MsgPack[8], and Protobuf [10] support the initial survey conclusions. This analysis found the following characteristics:

- Common use of Interface Description Languages (IDL) and code generation to create communications
- All the IDLs essentially describe the same structural concepts: classes, sequences, dictionaries, primitive types (int, float, bool, etc)
- IDLs needs to be machine readable (i.e. XML).
- Each provides the concept of an unique ID for code generated messages. JAUS manually assigns IDs, while ROS uses an MD5 sum of the IDL file.
- Messaging is commonly implemented using a combination of create, read, write, delete API model with extensions to enable asynchronous callbacks [26].
- System modularity is de-facto the approach to Open Architecture and existing robotic systems. We should support modularity at the inter-process and intra-process modalities. It should be simple to move modules between computing elements.
- The ability to collect statistics on how the system is behaving internally (such as performance timing, data rates) is vital to being able to debug the system
- Each approach provides similar tools for diagnostics, such as network monitoring, recording and playback.
- There is a wide variety of definitions of what "real-time" and "real-time" communications means. Two real time systems may/will have different latency allowances when delivering data.

During the investigation, the following realizations occurred:

- Creation of a superset IDL, which allows import of other IDLs is feasible. For example, the system should be able to simultaneously communicate JAUS and ROS transports. JAUS supports future transition, while ROS simplifies integration of University research. This would also save time when moving code between IDLs due to program requirements.
- The ability to speak multiple protocols enables a unified logging, playback and configuration infrastructure.
- Data should be separated from Messaging. JAUS Toolset, for example, implements this concept by providing a message wrapper around a data payload. ROS on the other hand, attaches the message metadata into the data structure itself, which increases memory overhead.
- Message metadata is useful to the client algorithms: time sent, time received, and who sent the message.
- The potential of code generated data structures to be used internally to simplify development. Removes the need to develop separate internal and external interfaces or "bridges".
- Addition of data type tracking would allow introspection into the running system. In order to work, introspection requires a mechanism to numerically uniquely identify each data structure, and its version.
- The framework should support hard and soft real-time designs.

RFrame is implemented using C++ as dynamically loadable modules from simple core interfaces and common utilities such as configuration, transparent logging, thread safety, and data structures such as UUID and Sequence. Changing data transport, supporting multiple transports at runtime, upgrading/exchanging functionality, or relocating modules between processing elements is a configuration file change not a code change. Extending existing codebases is minimally invasive as RFrame may be integrated as an I/O library meeting the common open, close, read, write API model. Being transport agnostic enables modules to be integrated side by side with existing components. There is a rich research community and multiple interoperability efforts, each with de-facto, current and emerging standards, built on separate messaging and transport protocols, leaving a fractured landscape for Integrators to navigate. RFrame enables interoperability between new and legacy systems built upon different middleware architectures through the use of a superset Interface Description Language and template based code generation. This role is defined in the FACE standard as a "Paradigm Translator".

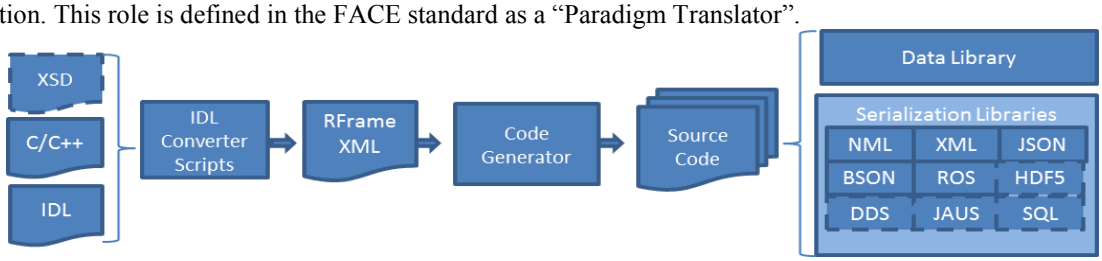

Figure 1: RFrame code generation imports from multiple specification formats into a common XML based IDL; generates serialization for multiple data format & protocols, enabling a data-centric transport agnostic design strategy which reduces development time.

For RCTA, this simply means that RFrame allows each researcher to use their preferred tools as it is simple to import their data, create data sets, and provide unified system logs to the government assessment team. Collaborator data are imported into RFrame XML using simple import conversion scripts (figure 1). These scripts will populate missing fields of the superset IDL if necessary. For example, if the input IDL contains a manually assigned id, it is maintained. If an ID is not provided, one is auto-generated by hashing the name of the imported structure with its namespace. If the IDL requires specialized information for its own protocol, it is also maintained. For example, ROS uses specialized MD5 checksums instead of message id and version numbers. Code generators then produce the serialization libraries for each requested transport protocol, which are dynamically loaded at runtime based on system configuration. As a result, protocols are transparent to the RFrame native modules minimizing extra bridging/interface code. RFrame is therefore data-centric and transport agnostic. The messaging system handles logging of all messaging regardless of transport, as well as data playback in simulation mode by using specialized modules. For example switching logging between disk and database solutions. All messages are transparently logged by default so ensure that all data is available when performing system analysis.

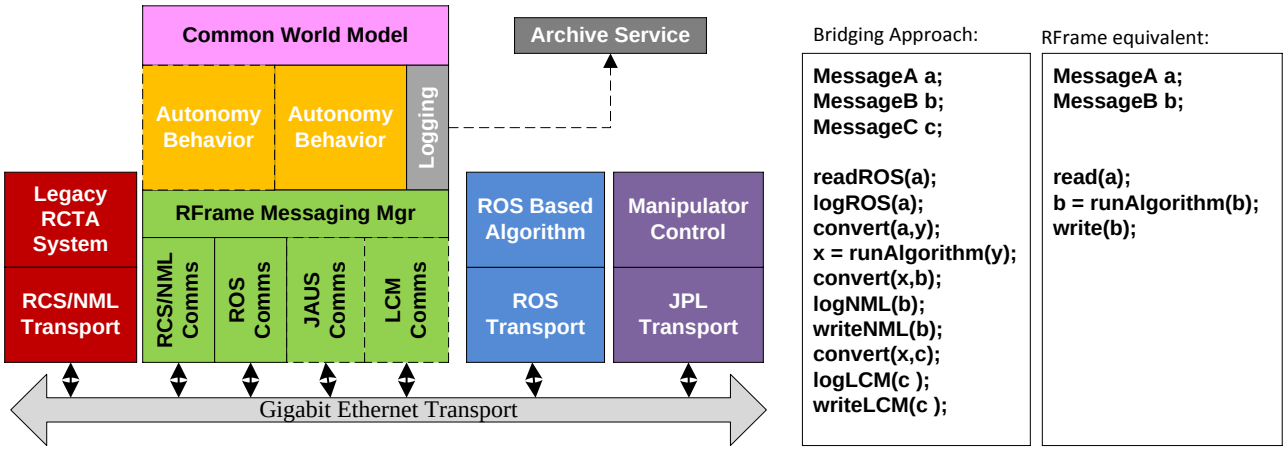

Figure 2: RFrame code generation imports from multiple specification formats into a common XML based IDL; generates serialization for multiple data format & protocols, enabling a data-centric transport agnostic design strategy reducing development time.

Structurally, RFrame adopts a layered approach (figure 2). At the lowest level are the Connection Managers such as ROS comms and JAUS comms which handle messaging for a specific protocol. Above the protocols is the Messaging Manager which provides a single interface to the client modules, and handles message routing to/from the Connection Managers and distribution of messages to client callbacks if necessary. RFrame supports both callback and non-callback ("direct") read/write operations in order to support legacy, current, and future design patterns. Both the Connection Managers and Messaging Manager adopt a common interface API. Client modules are implemented using a MessagingClient, which extends the common interface API to simplify use of these client interfaces by taking advantage of the data introspection capabilities created by the code generator. As mentioned above, each client is implemented as a dynamically loaded module, and runs within a separate thread of execution. To support threading all modules are built around common state machine (construct -> init -> start -> exec main loop "once" -> stop -> shutdown ->destruct). This pattern allows RFrame core to handle complexity for the developer such as thread safety, module loop period, potential I/O deadlocks during startup/shutdown. This design enables an RFrame module to support both hard-real time designs such as the 4D/RCS [11] based GDLS systems, and soft-real time designs such as JAUS or ROS. We are able to interact with non-4D/RCS nodes while maintaining our proven real-time concepts.

3. WORLD MODEL

Element two, which is central to our Architecture, is the RCTA Common World Model (CWM) shown in Figure 3. The CWM defines and instantiates the Data Model for the RCTA Intelligence Architecture with a data-centric approach providing a common, centralized Intelligent Data Store services. The data model was defined by need to represent 3 layers of information: Metric, Semantic, and Self. Specific data types within these layers were determined by querying RCTA collaborators to determine the best mechanisms to represent the desired data types.

The CWM leverages the transport agnostic capabilities of RFrame to be accessible by the preferred environments of RCTA Collaborators. The modular nature of RFrame makes simplifies integration by providing a stable, consistent mechanism for linking disparate system components to update and extend the CWM. This approach was proven successful during the Capstone Assessment of the previous RCTA program [14,15,16,17]. The assessment covered integration of technology developed over an eight-year research program into autonomous navigation. A week prior to the assessment, a vision based water detector [18] was integrated in four hours. Integration included linking output of the detector into the world model, defining the water types as obstacles in a configuration file, and completing successful live field testing. This integration was considered "low risk" primarily due to the use of the proven world model.

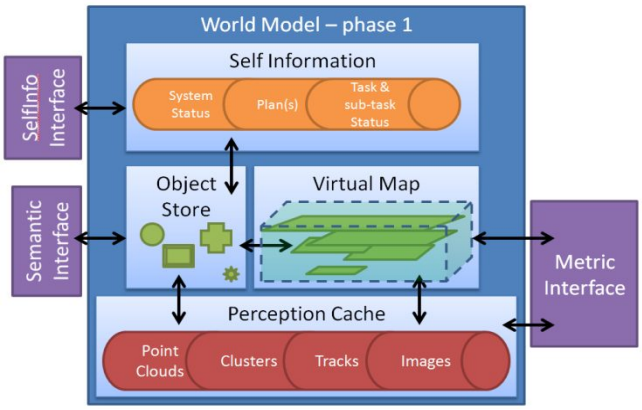

Figure 3: High level component layout of the world model "Phase 1" [12]. "Phase 2" is currently active [13]

The world model is divided into three main concepts: Metric (sensor data and aggregates), Semantic Objects (class descriptions and instances), and Self Information. An access interface is provided for each of these following the

"CRUD" database interface model: Create, Replace, Update, and Delete. Internally, the world model knows how the various data sources inter-relate, and when appropriate propagates changes between the three levels. This is why we refer to the world model as an "Intelligent" Data Store. It is more than just a database, it provides background data aggregation and calculation processes. Self-Information contains data relative to the robot's "Self". By encoding current capability, component status, task execution state, and their histories, we track information which enables the robot to reason and adapt its performance using Meta-Cognition and Machine Learning principles. This allows the system to answer questions such as: Where am I? What can I do? What am I commanded to do? and How well am I doing it?

The CWM is fully documented. The API, manual, example client programs, and technical support are provided to the consortium. More detailed information on the CWM may be found in [12][13].

4. RIVET SIMULATOR

The third basic element is simulation. RCTA has developed a Hardware In The Loop (HITL) simulator called RIVET to meet the needs of RCTA researchers. RIVET is based on a commercially available 3D Gaming engine, and industry standard PhysX high fidelity physics.

Testing with a real robot is required to fully access a robotic capability. While necessary, physical testing is expensive: acquiring robots, maintenance, travel costs, range time and possibility of delays (weather, equipment failures, over headed vehicles, etc). Immature algorithms increase these costs. RIVET provides a highly detailed, capable environment for sensor and algorithm development, integration and assessment using common off the shelf computers. Researchers are able to execute an order of magnitude more runs in the simulator, than would be possible with the real robot. This savings of time directly translates to cost savings, increased productivity, and more refined algorithms ready for system integration. RCTA uses RIVET as a "gateway" to placing research onto the physical platforms, as well as a regression test mechanism for the integrated system.

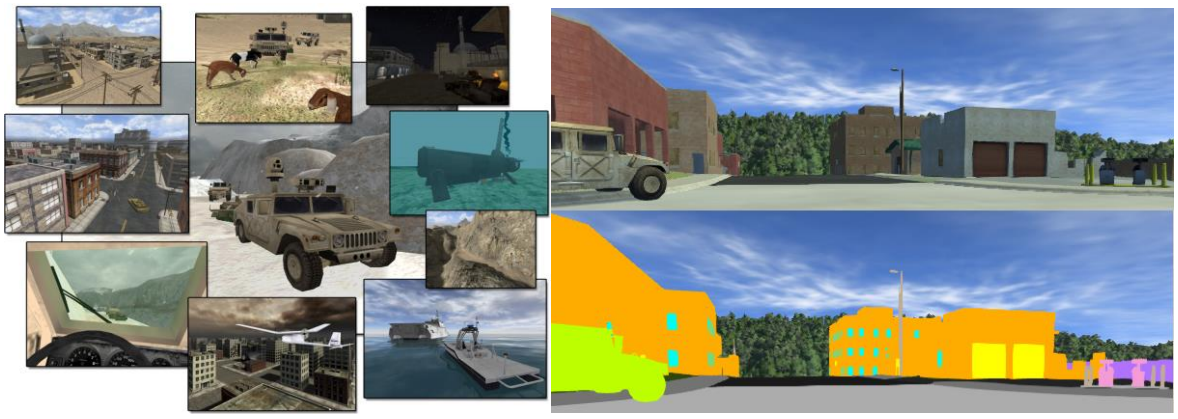

Figure 4: a) Left: Example RIVET environments. Simulated robots include ground, surface, and air vehicles,
b) Right: Example of matched ground truth for Semantic Labeling of the scene showing buildings (orange), windows (cyan), vehicle (lime), doorways (yellow), gas pumps (pink) and wall (purple).

Documented interfaces to RIVET are available through the use of the Basic Operation Level interface (BOLT) extension library. BOLT provides simple low level control and sensor interfaces to RIVET, allowing researchers to tailor the level of the simulation to their needs. RIVET installation provides a detailed model of the Fort Indiantown Gap CACTF test site, enabling research in a similar environment to real world assessment. Multiple vehicles and robot types are included (Figure 4a). Many sensors are also simulated including millimeter wave RADAR, 2D Ladar, 3D Ladar, monocular and stereo camera. Recently, Rivet has been upgraded to provide simultaneous ground truth for perception algorithms, which enables rapid training of the machine learning based perception algorithms while reducing false classifications due to human error.

5. INTEGRATION

RCTA as a program recognizes that individual research components can only be fully assessed when integrated onto a robot where they interact with other aspects of the system to create cross-Task capabilities which move beyond the State of the Art. To address the challenges inherent in integration, we have presented a multi-pronged approach of tools: RFrame enables transport agnostic messaging, CWM provides common Data Model and services, and RIVET provides a common cost effective initial integration platform.

For field assessment of integrated solutions, the fourth element supporting the RCTA program provides a common hardware platform. We are in the process of building 8 of these platforms: 3 fully outfitted to be used for program integration and assessment, and 5 basic setups provided to RCTA and ARL collaborators. The base configuration consists of a Clearpath Husky platform [23], Hokuyo line scanner [24], navigation unit, and bumblebee stereo camera, and a mac-mini running platform mobility control software. The full configuration (figure 5) adds a GDLS Adonis hi-dynamic range imager, and a GDLS microLadar 360 Ladar [26], and 3 mac-minis to run researcher software.

We are adopting a "rolling integration" solution. As components are integrated and verified to successfully work together, they are version controlled and released back to the consortium to enable future research. For example, a researcher in path planning algorithms is able to use the latest vetted perception algorithms with RIVET to evaluate their research. A designated regression test SIL is being setup to assist with this purpose.

To date the program has performed 4 Integrated Research Assessments. These assessments are designed and executed by the ARL assessment team and designed to challenge the integrated system in a statistically sound manner. The goal is to find out how the algorithms respond to these scenarios in order to improve the science itself. Positive or negative outcomes during the assessment are simply results providing the researchers with data which allows them to continue improvement

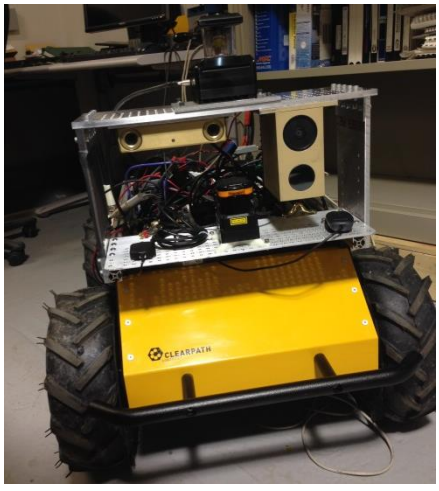

Figure 5: Fully configured research platform.Top shelf: bumblebee stereo (left), GDRS uLadar (center), ADONIS (right). Bottom shelf: gps (left), hokuyo Ladar (center), teleop receiver (right).

6. ACKNOWLEDGEMENTS

Research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-10-2-0016. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

The author would like to thank the membership of the RCTA Alliance: Army Research Labs, General Dynamics Robotic Systems, Carnegie Mellon University, Florida State University, University of Central Florida, University of Pennsylvania, Boston Dynamics, QinetiQ North America, and Cal Tech/Jet Propulsion Lab.

General Dynamics Land Systems approved for public release, LogNo 2014-11, Distribution Unlimited, April 17, 2014.

REFERENCES

[1] Bornstein, J., and Mitchell, R., "Foundations for autonomy for ground robotics," Proc SPIE 8387, (2012)
[2] Lloyd, D. "AS-4 Unmanned Systems Steering Committee" Society of Automotive Engineers, 2014, <http://www.sae.org/works/committeeHome.do?comtID=TEAAS4> (2014)
[3] The Open Group, Technical Standard for Future Airborne Capability Environment (FACE), The Open Group, Oct 2013
[4] TARDEC, "Vehicular Integration for C4ISR/EW Interoperability" <http://victory-standards.org>
[5] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," Proc. Open-Source Software workshop of the International Conference on Robotics and Automation (ICRA), (2009)
[6] Shackleford, D. "Real-Time Control Systems Library – Software and Documentation", National Institute for Standards and Technology, 2 February 2014, <http://www.isd.mel.nist.gov/projects/rcslib/> (2 February 2014)
[7] Object Modeling Group, "Data Distribution Service (DDS) version 1.2," OMG, 1 January 2007, <http://www.omg.org/spec/DDS/1.2/> (1 January 2007)
[8] "Lightweight Communications and Marshalling," <https://code.google.com/p/lcm/> (8 June 2013)
[9] Furuhashi, S., "MessagePack," <http://msgpack.org>, 2013, (2013)
[10] "Protocol Buffers – Google's data interchange format", <https://code.google.com/p/protobuf> (2 April 2012)
[11] Gazi, V., Moore, M., Passion, K., Shackleford, D., Proctor, F., Albus, J., The RCS Handbook: Tools for Real-Time Control Systems Software Development, Wiley-Interscience, , (June 8 2001)
[12] Dean, R. "Common World Model for Unmanned Systems," Proc SPIE 8741 (2013)
[13] Dean, R. "Common World Model for Unmanned Systems: phase 2" Proc SPIE 9096 (2014)
[14] Childers, M., Bodt, B., Hill, S., Camden, R., Gonzalez, J. P., Dean, R., Dodson, W., Kreafle, G., Lacaze, A., and Sapronov, L., "Unmanned Ground Vehicle Two-Level Planning Technology Assessment (ARL-TR-5331)", U.S. Army Research Laboratory, Aberdeen Proving Ground, MD, (2010)
[15] Bodt, B., Childers, M., Camden, R., "A Capstone Experiment to Assess Unmanned Ground Vehicle Tactical Behaviors Developed under the Robotics Collaborative Technology Alliance", presented at AUVSI North America 2010 Conference, Denver, CO, (2010)
[16] M. Childers, B. Bodt, and R. Camden, "Assessing Unmanned Ground Vehicle Tactical Behaviors Performance," International Journal of Intelligent Control Systems, V16, No 2, pp 52-66, 2011.
[17] Bodt, B. and T. Hong, "UGV Safe Operations Capstone Experiment," Proceedings of the 27th Army Science Conference," E-EO-004, (2010)
[18] A. L. Rankin and H. Matthies "Daytime Mud Detection for Unmanned Ground Vehicle Autonomous Navigation" Proc. 26th Army Science Conference (2008)
[19] Clearpath Robotics, "Husky" <http://www.clearpathrobotics.com/husky/>
[20] Hokuyo "Hokuyo UTM-30LX," Hokuyo, 2009, <https://www.hokuyo-aut.jp/02sensor/07scanner/utm_30lx.html>, (2009)
[21] General Dynamics Robotic Systems, "microLADAR," GDRS, 29 July 2012, <http://www.gdrs.com/20120729%20microLADAR%20LRes.pdf>, (29 July 2012)
[22] Wikipedia, "Callback (computer programming)," Wikipedia, 17 February 2014 <https://en.wikipedia.org/wiki/Callback_(computer_programming)> (17 February 2014)