


A Distributed Framework for Embedded Collaborative Autonomy

Thomas C. Furfaro
NATO STO Centre for Maritime Research and Experimentation (CMRE)
La Spezia, SP Italy
email: thomas.furfaro@cmre.nato.int
 <https://orcid.org/0000-0002-0561-6053>

Abstract—We have designed and implemented a framework for the development and deployment of multi-agent task allocation solutions, with particular awareness of the operational constraints encountered by Unmanned Maritime Systems (UMS). This framework, which we call the Distributed/Decoupled Collaborative Autonomy Framework (D²CAF), is a set of software modules that allow researchers to develop the underlying task allocation algorithms largely independent of engineering concerns (communications, vehicle interface, etc.). The framework is *distributed* as it provides tools to perform over-the-network (including underwater) communications between instances of the framework hosted on physically disconnected agents. The framework is *decoupled* in that it realizes an abstract separation between the invocation of a task and its execution. D²CAF has been used in several at-sea campaigns, deployed on real autonomous underwater vehicles (AUVs) promoting the execution of MCM missions.

Index Terms—multi-robot systems, multitasking, planning, unmanned autonomous vehicles, unmanned underwater vehicles, mine countermeasures

I. INTRODUCTION

With the increasing capability and performance of unmanned assets, the employment of those systems in networked cooperative operations has become likewise increasingly attractive to researchers and operators. Networked fleets of unmanned assets can provide greater coverage rates, greater communications ranges, and more unit-level redundancy than their monolithic analogues. Such a networked fleet, with elements operating in parallel, maps well to scenarios that have a dynamic mixture of tasks to be executed that may be separated in time and space.

From an implementation perspective, the infrastructure that must be built to achieve a collaborating, cooperating, networked fleet is not unsubstantial. The primary integration difficulties include managing network communications, which bring an entire field of sub-problems [1], and interfacing non-standard system interfaces.

Above all, D²CAF is meant to provide tools for rapid development and deployment of prototype algorithms for networked robotic task management in real scenarios and systems. Specifically, we rely on lower layer solutions to implement task capabilities in terms of a service contract. In

This work is supported by NATO Allied Command Transformation (ACT) Future Solutions Branch (FuSol).

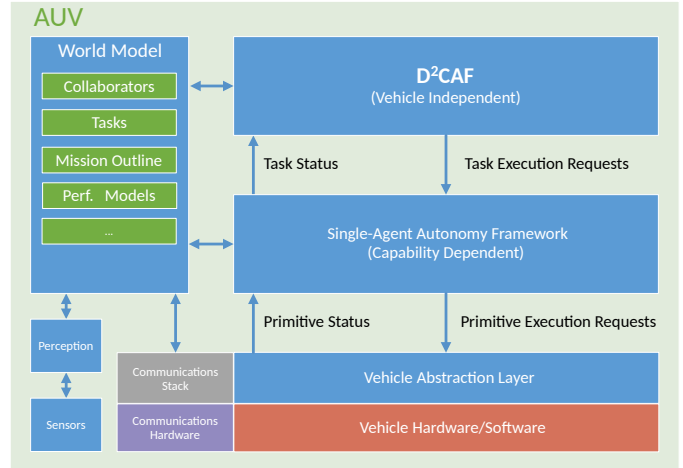


Fig. 1. The D²CAF framework as it is typically integrated on an AUV.

this way we separate the responsibility for task allocation and handling from the individual platform-specific implementation details of task execution.

II. ARCHITECTURE

D²CAF consists of an extensible plugin architecture, where developers implement core interface functionalities. Primarily, these functionalities are that of *task* and *collaborator* management. Tasks are the atomic unit of action that are communicated within the D²CAF framework. The definition of tasks and their descriptive attributes are themselves extensions that are implemented by researchers. Collaborators are the resources within the fleet that may execute tasks. We consider that collaborators are not known *a priori*, and allow for the possibility of ad hoc coalition building.

A high level representation of how D²CAF is integrated into an autonomous vehicle, is shown in Fig. 1. The framework sits at the top level of vehicle primitive invocation. This is above an autonomy framework, which exposes a task interface to D²CAF. This task interface is responsible for exposing the task capabilities as available services to D²CAF.

The autonomy framework is responsible for processing task execution requests by performing the requested task, and reporting back results (and progress) to the upper layer. This autonomy framework, in our case, is generally *capability*

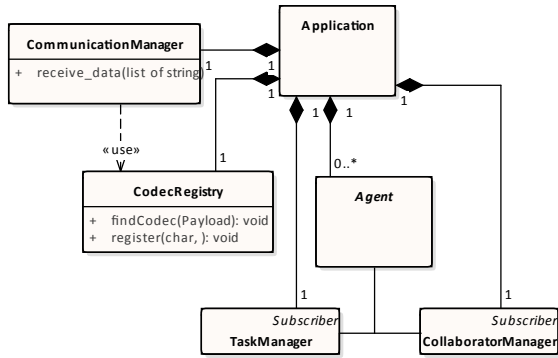


Fig. 2. The top level UML class diagram for a D²CAF runtime. There are always one of each of the TaskManagers, CollaboratorManagers, and CommunicationManagers for each Application, and a variable number of Agent objects.

dependent — that is, the particular single-agent framework reflects what capabilities are embodied in the system, but should have minimal bindings to the specific underlying vehicle system¹. This underlying vehicle is depicted in the lower two layers, a vehicle abstraction layer which abstracts the vehicle hardware/software interface. Both D²CAF and the single-agent autonomy layer interact with a world model, which is a generic, conceptual container where information regarding the state (current, past and possibly future) of the world are stored. Lastly, there is a connection with a communications stack, which manages a network connection using whatever technologies are available.

A. Features

1) *Distributed deployment*: Though deployments may be composed into a monolithic, centralised task handler, D²CAF has been ultimately formulated to provide *distributed* instances that communicate via available network channels. Each D²CAF instance is composed at runtime, based on a configuration file and a matching application factory mechanism, as shown in Fig. 2. This application, at its base, consists of a CommunicationsManager, a TaskManager, a CollaboratorManager, and possibly a number of Agent objects. In a fully-distributed scenario, there is a single Agent object in each D²CAF instance co-located with the vehicles for which they are performing task arbitration.

2) *Domain independence*: Our framework explicitly separates the *task dictionary*, itself coupled to the domain specifics, from the interconnecting tools. This allows users to expand or implement new task concepts and still reuse much of the existing infrastructure. A developer who wishes to deploy D²CAF in a new domain simply needs to develop the particular task dictionary that describes that domain.

3) *Detached deployment*: Some systems, for engineering and operational reasons, may not be open to have software

¹This delineation is made in order to increase the reusability of the components in the single-agent autonomy architecture by decreasing the coupling between the implemented autonomy capabilities and the specific hardware.

(such as D²CAF) integrated directly onboard. This is a common attribute of what we refer to as *operational* systems, i.e., systems that are specifically used to accomplish a given task and are not open, modifiable research platforms. When these operational systems are available to be used, without hosting software onboard, a D²CAF instance that interfaces with a task execution service may be run in a *detached* mode, where the task execution service layer is presented in a physically different location than where the task is executed. This allows task invocation to be performed through existing, single-agent command and control infrastructure (e.g. [2]), and thus preserving the native integrity of operational systems.

From a deployment perspective, this means that the architecture makes use of the multiplicity flexibility of the Agent object described in Fig. 2. Inside of a D²CAF network, there are as many Agent objects as there are collaborating squad elements. These agents, which act in the interest of their assigned element in participating in the task allocation mechanism, can be deployed in different assets than that which they represent. For example, we have run a D²CAF daemon on a gateway buoy with an Agent that represents an operational asset asset.

4) *Platform independence*: The delineation between the multi-agent task handling layer (D²CAF) and the single-agent autonomy layer in Fig. 1 means that D²CAF is decoupled from the platform itself. The single point of interface between the two layers are the task execution interface, where the autonomy system performs tasks in response to requests.

5) *Ad hoc coalition formation*: We have implemented a simple synchronisation mechanism to allow for the ad hoc composition of a squad. During the synchronisation state, units suspend task execution and periodically transmit identifying information about their own task capabilities and their current conception of the squad size. Any squad elements that overhear broadcast synchronisation messages with squad size information that doesn't match their own model transitions to the synchronisation state. The elements transition out of the synchronisation state after all overheard messages report consensus on the total squad size.

6) *Explicit non-features*: Moreover, D²CAF specifically does not aim to address some sub-problems, for which we consider other approaches, architectures or solutions to be more appropriate:

a) *Communications stack*: D²CAF explicitly does not attempt to address all specific elements of underwater networking, and instead relies on other infrastructure to provide end-to-end connectivity. In the OSI model, D²CAF is an application integrating with the communications stack at the application layer, thus adhering to 'separation of concerns' [3]. CMRE has other specific solutions that are meant to address the particularities of underwater communication and networking, such as [4], [5] — these exist in the 'Communications Stack' block in the lower right of Fig. 1.

What D²CAF does provide to developers is a flexible subscriber-codec infrastructure where codecs that encode and decode data arriving from external data streams are registered

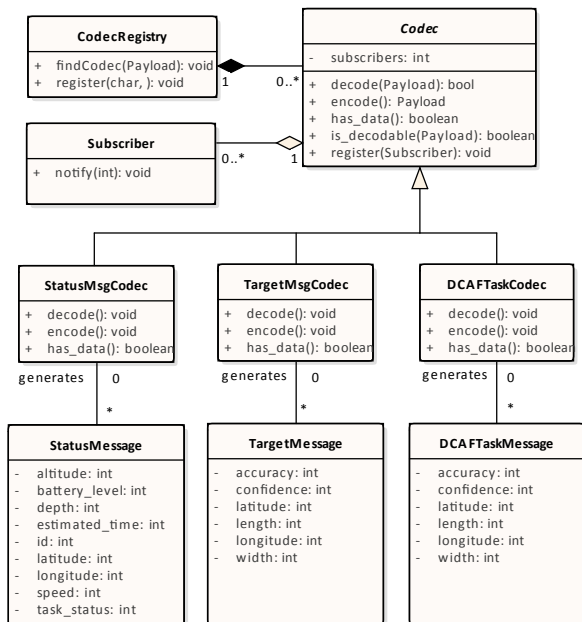


Fig. 3. An example of codec implementation in D²CAF, where the Codec interface class is implemented by specific concrete classes (e.g. StatusMsgCodec) that generate the underlying data structures (e.g. StatusMessage). Subscribers use the CodecRegistry to register for specific codecs by key, and a CommunicationsManager calls the findCodec method for messages that arrive from a configurable set of data streams.

in a registry for use by many different subscribers inside D²CAF. An example of a few codecs, and how they relate with the registry and subscriber objects, is shown in Fig. 3. We have found that such flexibility is frequently required when dealing with a heterogeneous set of assets, where the interfaces, or ‘hooks’ can vary greatly between platforms. This additionally decouples the encoding logic from the underlying data model of messages, further increasing reusability and modularity.

b) *Task execution*: D²CAF also does not purport to provide advanced task execution strategies. That is, the responsibility of actual performing the work associated with a task is the responsibility of a specific autonomy system. The assumption and paradigm of D²CAF is that the provider of such an autonomy system can build more performant and appropriate task execution mechanisms based on more detailed knowledge of the internal workings. These functions are relegated to the underlying ‘single agent autonomy architecture’, depicted as the right-center block in Fig. 1. That is not to say that elements of D²CAF cannot be extended to fulfil this role (in fact, in our implementation we do such a thing), but that it is not one of the core goals.

III. IMPLEMENTATION

The implementation of the current D²CAF paradigm began in 2017, with developments ongoing. The framework itself has asymptotically stabilized, with the greatest activity being focused on algorithm development and standardisation efforts.

A. Plugin Architecture

D²CAF is implemented as a plugin architecture, where the D²CAF application is composed via dynamic loading of modules and associated parameters from a configuration file. The selection of the modules and their parameters at start up, dictate the total behavior of the system. Developers can extend or develop specific modules that address sub-functionalities of D²CAF and simply load those at runtime.

B. Language

D²CAF is implemented in the Python programming language, which we have found to provide more than satisfactory performance in this application. Task allocation is a relatively low-bandwidth endeavor, with high levels of abstraction away from dense sensor data that may require highly efficient computing. Additionally, Python allows non-expert researchers to quickly implement task-handling algorithms with the extensive support of scientific computation libraries commonly used in Python.

C. World Model

We refer to the *world model* as the container which holds all of the relevant, contextual information related to autonomous operations. This approach is driven by our experiences in developing autonomy solutions, where inevitably, without explicit foresight and design, data products generated by different components in the architecture tend to exist in different places, with different methods of access and stored in different formats. The world model is meant to consolidate this information for a more integrated and holistic approach to promote more advanced autonomy solutions. The world model functions as the single point of reference where modules may look up information regarding the environment, the system status or performance, or collaborator information discovered through the network.

We also see advantages when it comes to *data lifetimes*. In practice, the timeline for deployment and recovery of numerous unmanned marine systems is highly dynamic, with elements being added and removed to the operation continuously for logistical and operational reasons. The concept of a squad mission starting and ending for all units at the same time is unrealistic and impractical. So, it is useful to have long term knowledge of the events, schedule, tasking and other mission data, even if a given platform was not activated for the entire duration. Using a centralised world model ensures that client modules that need access to historical data can rely on central lifetime guarantees, as opposed to having to reconstruct a local world model from different sources.

Specifically with respect to D²CAF, the world model tracks two basic types of information: that relating to tasks and collaborators. The elements of the collaborator and task models are based on data received through the network, regarding task creation and prosecution, and squad member status. The implementation of the task and collaborator interfaces and tools are described in Fig. 4 and Fig. 5.

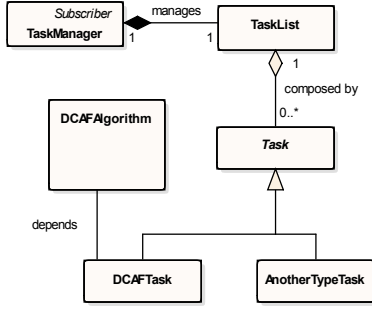


Fig. 4. The client module API for interacting with tasks is realized in the TaskManager, which manages a TaskList, itself composed by zero or more Tasks. The realization of the abstract Tasks is where developers can deploy new task types, upon which a specific task allocation algorithm might depend (e.g. DCAFAAlgorithm).

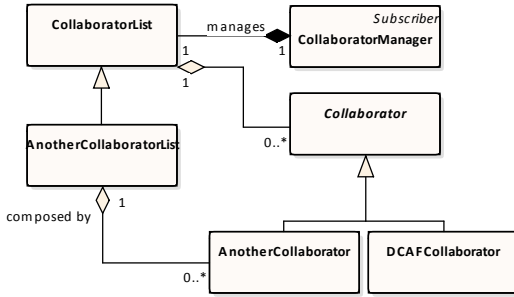


Fig. 5. The interaction and management of collaborators and their status is handled by a CollaboratorManager. Collaborators can be specialized to provide functionality that is more coupled to the specific algorithm employed for task allocation.

D. Agent Integration

In order to integrate a new system into the D²CAF environment, developers must specialize a base class, the VehicleAdapter, as shown in Fig. 6. Typically this involves a conversion between the single- and multi-vehicle data models for status and task execution information. In practice we have found that this realization must sometimes provide task-centric state management to systems that don't have native concepts of task identification. Note that the concrete implementations here may use the codec infrastructure (denoted as Subscriber subclasses). In related work outside the scope of this publication, we are also working on ways to normalize this interface, such that the same tasking interface may be presented to the D²CAF layer, regardless of the task execution implementation in the single-agent layer (see Fig. 1).

Part of this interface between the agent and D²CAF is the description of task *capabilities*. We consider a capability description to be a description of the platforms ability to fulfil a task request. The capability of a platform may vary over time, depending on the system's status or the status of subsystems (i.e. sensors or processing chains). Additionally, part of the interface includes a performance prediction capability. Basically, D²CAF can request from the single-agent perspective what the expected performance (e.g. energy cost, coverage, duration) would be for a specific instance of a task. This

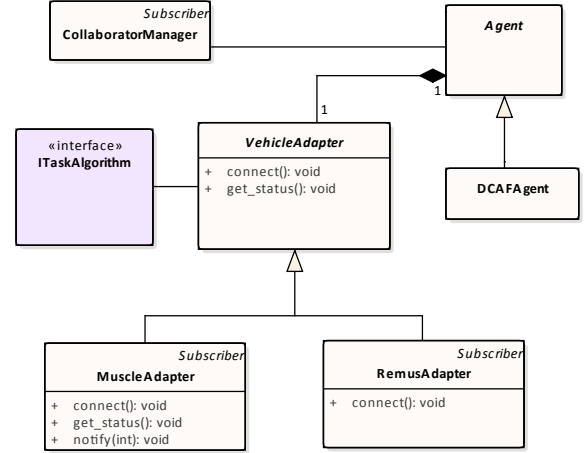


Fig. 6. For a specific type of platform, integrators implement a concrete VehicleAdapter subclass that handles the job of interfacing with the vehicle autonomy system.

information is made available to the task allocation algorithm, such that the specifics of platform can be decoupled from the total score calculation. For the estimation of the performance of foreign squad members, simpler models are used to estimate their performance.

E. Operator Interface

We have also implemented a diagnostic/monitoring API which is exposed over the local network via JSON-RPC, [6] over ZeroMQ, [7]. Again using the launch-time plugin configuration, the architecture in Fig. 1 can be launched without an Agent in a *topside mode* with the monitoring interface enabled. Then we connect a local webserver via the API to the daemon to receive information about collaborators and tasks overheard through the network, publishing this information as a dynamic website to provide operators with situational awareness. This API also allows the web service to inject new tasks generated by the user, which is then propagated by the D²CAF service into the network.

IV. TASK ALLOCATION EXPERIMENTS

The subject framework has been a critical tool in the development of task allocation algorithms for real operational domains.

A. Task Domain

We have deployed D²CAF in several in-field trials, connecting assets with different task capabilities to collaboratively generate, distribute and allocate tasks. Our particular domain is that of naval mine countermeasures (MCM), a defence application that is at the forefront of operational use of AUV systems.

Our D²CAF-connected squad consists typically of a large-area survey vehicle with a SAS (CMRE's MUSCLE AUV), and a number of heterogeneous reacquisition systems (such as Bluefin 21 or REMUS 100 AUVs), some of which may be platforms of opportunity that are subsumed into the squad.

Conceptually, MCM consists of a high level clearance task, which decomposes into a search/exploration task (usually called a survey) followed by subsequent reacquisition, identification, and neutralization tasks to be performed on targets discovered during the survey. Furthermore, the typical sensors required (or most well suited) for the survey task, versus the subsequent requisition/identification/neutralization tasks are quite different. Underwater surveys in MCM are usually performed with long range side-scan or synthetic aperture sonars (SSS, SAS), which may have ranges in the order of 100 m. The following three phases (reacquisition, identification and neutralization) require higher resolution, closer range sensors, such as short range sonars, and optical or acoustic cameras, with less than 10 m range. Moreover the mobility dynamics of a survey-capable vehicle may be quite different to those of a follow-on system, where a survey vehicle might need to be highly stable directionally, and the follow-on systems may need more manoeuvrability.

B. Task Allocation

The first task allocation strategy implemented in D²CAF derives strongly from the well-known L-ALLIANCE ([8]). In short summary, individual agents model the utility for individual scores based on concepts of cost (what is the cost of performing a given task), impatience (how long has a given task been unfulfilled) and acquiescence (how well might other agents perform the task relative to the given platform). This “activation” approach is inspired by biological systems, where the decision to do a specific action is a summation of many contributing factors.

Additionally, in the current approach, each instance of the task allocation algorithm (i.e. in each D²CAF instance) predicts the scoring for all known unit elements. That is, the performance of each task in every remote collaborator is locally estimated, and used as a factor in the selection of tasks to execute.

It is important to say that our task handling approach makes few assumptions about the quality of service of a given networking infrastructure. The implemented solutions for task selection use very low data rates that are typical of underwater networks (tens of bytes per minute). Moreover, loss of connectivity is naturally assumed, with some timeout and backoff mechanisms used to mitigate failures during periods of blackout. In the worst cases, the implemented algorithms fails back to a simple single vehicle task scheduler.

Our approach is still highly developmental, and will be the subject of follow-on publications.

C. Experimental Activities

The major in-water event for D²CAF in its current form took place in May 2018, where it was used to connect different mine hunting AUVs to perform mine clearance operations in an exercise context. CMRE participated in the Italian MINEX 2018 with the MUSCLE and BlackCAT AUVs, which are 21 inch Bluefin vehicles adapted as research platforms supporting the development of advanced autonomy solution for

mine countermeasures activities. The MUSCLE specializes in MCM wide-area survey tasks, with a high resolution synthetic aperture sonar (SAS), algorithms that run online for automatic target recognition (ATR, [9]), and additional capabilities for the *in situ* estimation of sensor coverage and data quality. The BlackCAT system is focused on reacquisition tasks, where contacts of interest are targeted specifically using other sensing modalities, including acoustic and optical cameras, as well as a multi-beam echo sounder (MBES, [10]). Additionally, the Italian Navy, the *Marina Militare*, participated in the D²CAF serials with their REMUS100 AUV, an operational unit that is part of the Italian minehunting command (MARICODRAG).

In these serials, during which D²CAF was deployed for the first time in its current form, both of the CMRE assets carried as part of the onboard software their respective D²CAF agents. The REMUS system was operated in detached mode, where the responsible agent for the vehicle tasking was instantiated as part of the topside infrastructure. In these configuration, the MUSCLE AUV would elect to perform survey tasks, as they matched most closely to its own capability and availability. During the execution of the survey, based on the onboard processing results, the MUSCLE would then dynamically generate new reacquisition tasks and disseminate them to the network. The REMUS and BlackCAT AUVs would then select the follow on tasks to perform based on the scoring method described above.

V. DISCUSSION AND CONCLUSION

The development of D²CAF has been driven largely by the requirement to integrate any task allocation algorithms into a real fleet of heterogeneous systems. The complexity in handling systems in operational scenarios is seen as a problem orthogonally related to the underlying research question about autonomous system tasking. Thus we have created D²CAF to fill the gap between the host platform and the researchers prototype, to encourage more rapid development and ease of experimentation.

A. Relation to broader scope

CMRE is also involved in other efforts relating to task allocation and multi-agent robotics in the maritime domain. For example, for a different defence domain (anti-submarine warfare, ASW), our colleagues have taken a market-based approach in their development of Periodic Auctions Distributed Algorithm (PADA) [11]. We see our effort here as complementary, in that D²CAF can provide the infrastructure inside of which PADA may be implemented and deployed for operations.

A similarly related commercial product is Neptune, [12], which is found frequently in operational contexts. D²CAF and Neptune are somewhat disjoint in their particular functionalities, mostly in that Neptune also provides a task execution framework, and is highly targeted (at least in its current form) towards MCM explicitly.

Lastly, CMRE is involved with a NATO panel activity (under the purview of the Collaboration Support Office,

CSO) titled “SCI-288: Autonomy in Communications Limited Environments”, [13], [14]. One of the focus points of the activity is a broader agreement regarding the prescient information required to enable collaborative task handling inside squads of operational or near-operational unmanned systems. Again, D²CAF has a complementary design, where the SCI-288 outcomes may inform the specific design of the inter-vehicle communications messaging layer that connects disparate D²CAF instances.

B. Way Forward

The implemented task allocation algorithm is subject to ongoing testing (at sea and in simulation) and reporting. There are many related problems which we hope to work on:

1) *Task decomposition*: We currently work with a “flat” model of tasks — that is, tasks are not conceptually decomposed from a higher level task.

2) *Parallel Execution*: Task execution on a single platform, from the perspective of allocation, is constrained to be serial process, which may be suboptimal on more advanced systems.

3) *Waterspace Management*: Waterspace management is an issue not currently addressed from the D²CAF architecture or the implemented algorithms, but requires explicit attention.

4) *Interoperability*: We are hoping to exercise more diverse task handling algorithms within our framework to exercise and validate the extensibility of underlying models and interfaces.

* * *

We have developed and tested in real AUV operations the Distributed/Decoupled Collaborative Autonomy Framework. This D²CAF is designed as a framework upon which prototype approaches for multi-system task handling may be implemented. There is a high level of customisability and modularity which we think provides a tool-rich environment in which to deploy task handling prototype algorithms for research purposes. The framework we have described is in active use on real embedded systems operating in real scenarios, supporting the ongoing refinement of algorithms addressing particularly the domain of autonomous naval mine countermeasures.

ACKNOWLEDGEMENTS

This work is an evolution of that initiated by Warren Connors during his tenure at NATO STO CMRE, and we thank

him for his contribution. We would also like to thank Renato Sirola, who has been a critical developer in the implementation of the current version of D²CAF. Lastly, we would like to thank the REMUS 100 team from the Italian *Marina Militare*’s MCM command (MARICODRAG), and their leadership, who have been eager, willing partners in our work.

REFERENCES

- [1] J. Partan, J. Kurose, and B. N. Levine, “A survey of practical issues in underwater networks,” *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 11, no. 4, pp. 23–33, Oct. 2007. [Online]. Available: <http://doi.acm.org/10.1145/1347364.1347372>
- [2] R. P. Stokey, L. E. Freitag, and M. D. Grund, “A compact control language for auv acoustic communication,” in *Oceans 2005-Europe*, vol. 2. IEEE, 2005, pp. 1133–1137.
- [3] E. W. Dijkstra, “On the role of scientific thought,” in *Selected writings on computing: a personal perspective*. Springer, 1982, pp. 60–66.
- [4] A. Vermeij, T. C. Furfaro, and J. Alves, “NEMO: An architecture for software communications research in the maritime domain,” in *Proceedings of MTS/IEEE OCEANS 2015*, Genova, Italy, May 18–21 2015, pp. 1–4.
- [5] R. Petroccia, G. Zappa, T. Furfaro, J. Alves, and L. D’Amaro, “NEMO: An architecture for software communications research in the maritime domain,” in *Proceedings of MTS/IEEE OCEANS 2018*, Charleston, SC, USA, Oct 21–23 2018, to appear in.
- [6] JSON-RPC Working Group. (2013, Apr.) JSON-RPC 2.0 Specification. [Online]. Available: <https://www.jsonrpc.org/specification>
- [7] P. Hintjens, *ZeroMQ: messaging for many applications*. O’Reilly Media, Inc., 2013.
- [8] L. E. Parker, “L-alliance: Task-oriented multi-robot learning in behavior-based systems,” *Advanced Robotics*, vol. 11, no. 4, pp. 305–322, 1996.
- [9] D. P. Williams, “Fast target detection in synthetic aperture sonar imagery: A new algorithm and large-scale performance analysis,” *IEEE Journal of Oceanic Engineering*, vol. 40, no. 1, pp. 71–92, Jan 2015.
- [10] D. Machado, T. Furfaro, and S. Dugelay, “Micro-bathymetry data acquisition for 3d reconstruction of objects on the sea floor,” in *OCEANS 2017-Aberdeen*. IEEE, 2017, pp. 1–7.
- [11] G. Ferri, A. Munafo, A. Tesei, and K. LePage, “A market-based task allocation framework for autonomous underwater surveillance networks,” in *OCEANS 2017 - Aberdeen*, June 2017, pp. 1–10.
- [12] S. Ltd. (2018) Neptune: Technical whitepaper. [Online]. Available: <http://www.seebyte.com/wp-content/uploads/2018/02/Neptune-Technical-Whitepaper.pdf>
- [13] NATO STO CSO. (2016) Autonomy in Communications-Limited Environments. [Online]. Available: <https://www.sto.nato.int/Lists/test1/activitydetails.aspx?ID=16106>
- [14] —. (2017) Autonomy in Communications-Limited Environments. [Online]. Available: <https://www.sto.nato.int/SitePages/newsitem.aspx?ID=3551>