Innovative Management of Remote Future Internet Experimentation over Mobile Nodes

Journal:	*IEEE Internet of Things Journal*
Manuscript ID	IoT-0962-2016
Manuscript Type:	Regular Article
Date Submitted by the Author:	09-Mar-2016
Complete List of Authors:	Panagidi, Kyriaki; National and Kapodistrian University of Athens, Informatics and Telecommunications Kapoutsis, Athanasios; Democritus University of Thrace, Electrical and Computer Engineering; Centre for Research and Technology-Hellas Informatics and Telematics Institute, Information Technologies Institute Kolomvatsos, kostas; National and Kapodistrian University of Athens, Informatics and Telecommunications Chatzichristofis, Savvas; Democritus University of Thrace, Electrical and Computer Engineering; Centre for Research and Technology-Hellas Informatics and Telematics Institute, Information Technologies Institute Tusa, Giovanni; IES Solutions — Intelligence for Environment and Security Heckel, Marcel; Fraunhofer Institute for Transportation and Infrastructure Systems Ramapuram, Jason-Emmanuel; Hautes Ecoles Spécialisées Genève HES-SO Kosmatopoulos, Elias; Democritus University of Thrace, Electrical and Computer Engineering; Centre for Research and Technology-Hellas Informatics and Telematics Institute, Information Technologies Institute Hadjiefthymiades, Stathes; National and Kapodistrian University of Athens, Informatics and Telecommunications
Keywords:	Test-bed and Trials < Sub-Area 3: Services, Applications, and Other Topics for IoT, Service Middleware and Platform < Sub-Area 3: Services, Applications, and Other Topics for IoT, Application Platform < Sub-Area 3: Services, Applications, and Other Topics for IoT, Network Architecture < Sub-Area 2: Communications and Networking for IoT, Service-Oriented Architecture < Sub-Area 3: Services, Applications, and Other Topics for IoT, Future Internet < Sub-Area 2: Communications and Networking for IoT

Innovative Management of Remote Future Internet Experimentation over Mobile Nodes

K. Panagidi, A. Kapoutsis, K. Kolomvatsos, S. Chatzichristofis, G. Tusa, M. Heckel, J. Ramapuram, E. Kosmatopoulos, S. Hadjiefthymiades

***Abstract*—Mobile IoT applications realize an innovative field where numerous moving devices collect, process and exchange huge amounts of data with central systems or their peers. Novel applications could be built on top of this setting that aim to facilitate people's lives and produce new products. Some of the potential applications are real-time marketing support, enhanced situational awareness, intelligent decision analytics on top of sensors measurements and so on. For engineering novel applications, experimentation plays a significant role. Especially, when experimentation is performed remotely, it offers many advantages while reducing the cost and the efforts spent to realize physical experimentation. In this paper, we present the experimentation and the resources (i.e., mobile devices / nodes) management proposed by *Road-, Air- and Water-based Future Internet Experimentation* (RAWFIE). RAWFIE aims to interconnect numerous devices in the form of testbeds where experiments will take place. It offers an editor where experimenters can remotely insert their experiments and a set of powerful components responsible to handle the underlying architecture. Among them, the resource controller is responsible to manage the available devices and transform the experiments commands into to commands executed by the nodes. Hence, through a user friendly environment, the complexity of the devices and the underlying architecture is hidden from experimenters offering an efficient scheme for remotely experimenting with devices and managing the observed data.**

***Index Terms*—Future internet experimentation, Cloud based Robotics platform**

I. Introduction

SINCE the 1970s, autonomous robots have been in daily use at any altitude, for deep-sea and space exploration as well as in almost all aircraft [1]. The last decades, an increasing interest has been recorded on the exploitation of unmanned vehicles in fields such as environmental monitoring [2], commercial air surveillance [3], domestic policing, geophysical surveys, disaster relief, scientific research, civilian casualties, search and rescue operations, archaeology, maritime patrol, seabed mapping, traffic management, etc. Regardless the domain (i.e., aerial, ground or surface) that they belong to, the key elements that distinguish them as the leading edge of their technology are the provided degree of autonomy (i.e., the ability to make decisions without human intervention), the endurance and the payload that they can support.

The mobile IoT paradigm introduces many different technical challenges that call for efficient solutions either horizontally (application-neutral) or vertically (application-specific). Such challenges are faced continuously by researchers and innovators Worldwide. In this domain, experimentation is a key component of engineering novel applications. Remote experimentation can offer many advantages as physical experimentation is expensive, difficult to maintain and restricted to specific areas. The *Road-, Air- and Water-based Future Internet Experimentation* (RAWFIE) platform comes to offer such functionalities and deliver a framework for interconnecting numerous testbeds over which remote experimentation will be realized. RAWFIE platform originates in a European Union-funded (H2020 call: FIRE+ initiative) project which focuses on the mobile *Internet of Things* (IoT) paradigm and provides research and experimentation facilities through the ever growing domain of unmanned networked devices (vehicles). The IoT paradigm can support an intelligent network which connects all things to the Internet for the purpose of exchanging information and communicating through the information sensing devices in accordance with agreed protocols [4]. IoT imagines a not so distant future, in which the objects of ordinary life will be equipped with microcontrollers, transceivers for computerized correspondence, and suitable protocol stacks that will make them ready to communicate with one another and with the users, becoming an integral part of the Internet [5][6].

As trillions of mobile things will be connected to the Internet, it is necessary to have an adequate architecture that permits easy connectivity, control, communications, and useful applications [7]. This paper aims at providing a consolidated architecture for RAWFIE paying special attention on the resource management and the creation of experiments over multiple testbeds interconnected in the RAWFIE framework. The purpose of the proposed structure is to create a federation of different network testbeds that work together to make their resources available under a common framework. Specifically, it aims at delivering a unique, mixed experimentation environment across the space and technology dimensions. RAWFIE integrates numerous testbeds for experimenting in vehicular (road), aerial and maritime environments. Support software for experiment management, data collection and post-analysis will be virtualized to enable experimentation from everywhere in the world. The vision of Experimentation-as-a-Service (EaaS) is promoted through RAWFIE. RAWFIE offers an Experiment Description Language (EDL), i.e., a Domain Specific Language (DSL), and an editor devoted to assist non-experienced users to easily define their experiments. A code generation component is responsible to translate each experiment expressed in the EDL into the information transferred to mobile nodes. Hence, RAWFIE efficiently interconnects experimenters coming from various domains with the node present in numerous testbeds.

To the best of our knowledge, this paper describes one of

the first future Internet experimentation platform for managing multiple UxVs (i.e., mobile vehicles of different categories) in real-world scenarios. It is worth noting that RAWFIE platform is invariant to the robotic architecture or the distinctive parts used and can easily incorporate additional test cases with minimal adaptations, thanks to the fact that the envisaged platform can easily support the addition of new data formats for data exchange with UxV nodes. And that only open, not proprietary solutions are used as far as the communication between different Tiers is concerned, as it will be more clear in the following of the document.

The rest of the paper is organized as follows: Section II presents the related work while Section III provides an overview over the RAWFIE architecture and its layers. Section IV outlines the testbeds and resources management process and Section V discusses the RAWFIE data management approach. Section VI describes the the experiment management process while Section VII presents our approach related to the RAWFIE GUI. Finally, Section VIII draws concluding remarks.

II. Related Work

There have been several research initiatives in the Future Internet Research & Experimentation (FIRE) open research environment[1], and especially targeting an heterogeneous Testbeds Federations, as RAWFIE does. While the RAWFIE architecture has similarities with some of these projects, surely it targets a special and new idea, which is the use of UxV nodes in the Testbeds. The federated testbeds manly have in common, that you can access several testbeds in the federation in a common way, which reduces the efforts to test an algorithm in another testbed. RAWFIE transfers this benefit to UxVs testbed. Examples for these are Fed4FIRE [8], [9], OneLab[2] or WISEBED [10].

Fed4FIRE[3], the Federation for Future Internet Research and Experimentation [8], [9], is an Integrating Project (IP) under the European Unions Seventh Framework Programme (FP7). The Fed4Fire architecture includes all components needed for testbeds and resource discovery and provisioning, experiments monitoring and measurements collection. It considers the use of two specific protocols for Resource Control, Resource Monitoring and measurements collection, namely the FRCP and OML protocols. However, OML does not provide any mechanism for addressing network availability/connection problems, which will regularly occur with moving UxVs in RAWFIE. Furthermore, in RAWFIE we start from the idea to allow the easy integration of any kind of UxVs resources, therefore no specific data schemas for sending commands or receiving measurements are imposed, for the communication with the resources. A message bus based communication within the testbeds and between the testbed and the middle tier is just defined, together with a mechanism for allowing the automatic support of new message schemas as needed by new resources to be included.

A special idea followed IoT Lab [4] (also accessible via OneLab), where testbed became crowd sourced via an Android application that enable the smartphone to be used as testbed node. To handle the heterogeneous testbeds in a common way, the testbeds are virtualized via a common API: There are four individual IoT Lab testbeds. Each one with a different architecture set of provided functionalities, etc. Each testbed is responsible for exposing its resources and services over a commonly understood API. Each testbed is responsible for dealing with its specific characteristics, such as network connectivity or its special resources. This virtualization results in testbeds could be handled in a common way, provide a Fed4FIRE-compliant point of access to the outside world with a common addressing scheme for all resources in IoT Lab and the Resource Directory can be maintained at the IoT Lab cloud (instead in the TB themselves).The ORBIT Measurement Library (OML) [5] is used. The networking in IoTLab is fully based on IPv6. This guaranties interoperability between multiple physical interfaces (Ethernet, Wi-Fi, Bluetooth, IEEE802.15.4, etc) and enables end-to-end connectivity without the need for a NAT. In addition, IPsec is used to enable secure/encrypted communication of potential non-secure networks. The direct use of OML is not reasonable in the context of RAWFIE, as OML was initially designed for regular networks. OML does not provide with any mechanism for addressing network availability/connection problems, which will regularly occur with moving UxVs. So OML may only be used as interface between the Testbed proxy and the Middle Tier. But inside the testbed, other technologies should be used to transmit measurements (e.g. messages buses with persistence). The approach of virtualizing the testbed will also fit well to RAWFIE, as the different testbeds for the UxV will be very heterogeneous. So, integrating a testbed into RAWFIE requires a common interface (e.g. FED4FIRE compliant) exposed by the Testbed Proxy, which should hide internal management, networking and other issues. RAWFIE should also follow the consequent use of IPv6 to address resources.

The WISEBED[6] project was a 3-years EU project (2008 - 2011) funded by the European Commission under the FP7 framework [10]. The aim of the project, was to build a distributed infrastructure of heterogeneous and interconnected testbeds, for testing the more recent research results on algorithms and protocols for Wireless Sensor Networks measurements and communication. WISEBED was therefore specifically focused on the WSN domain, and on the interconnection of completely separated, and already existing experimentation platforms, each of them providing its own Web accessing Portal and software infrastructures. Conversely, RAWFIE will allow multidisciplinary experiments with potentially unlimited types of technologies, and will develop a common platform for the management of a federation of testbeds, by providing a single point of access and a common software infrastructure (RAWFIE Frontend, Middle Tier and Data Tier components),

[1] http://www.ict-fire.eu/home.html
[2] http://https://www.onelab.eu/
[3] http://www.fed4fire.eu/
[4] http://www.iotlab.eu/
[5] https://mytestbed.net/projects/oml
[6] http://www.wisebed.eu/

for management, execution and analysis of experiments carried out using UxV resources which will be available at the different testbed facilities with minimal adaptations, thanks to the fact that the envisaged platform can easily support the addition of new data formats for data exchange with UxV nodes.

III. Architectural Overview

A. Framework High Level Description

From the architectural point of view, RAWFIE leverages the *multi-tier design pattern* in order to facilitate the implementation of a highly and easily extensible remote testing platform. The functionalities for the presentation of the information to the experimenters, the implementation of the core parts of the business logic and the software interfaces for the integration of the different modules along with the data persistence, are separated in different tiers. More specifically, a specific ***Testbed Tier*** is envisaged for the actual control of the test-bed resources and the communication with the upper layers. A Web based ***Front-End Tier*** (presentation) allows experimenters remote access to the platform, authorization and visualization of the information, and the interaction with test-beds resources. The ***Middle Tier*** implements most of the business logic and, particularly, the communication between the Front-End and the Test-bed tiers. The ***Data Tier*** will be in charge of ensuring data persistence. Figure 1 depicts the tiers of the RAWFIE infrastructure.

RAWFIE follows the *Service Oriented Architecture* [11] paradigm: all components provide clearly defined interfaces, so that they can be easily accessed by other component or they may be easily replaced by other/better component with the same interface. The services are described in languages such as *Web Services Description Language* (WSDL) [12]. Interacting with them is made possible by the use of remote service control protocols such as *Simple Object Access Protocol* (SOAP) [12] or the *Representational State Transfer* (REST) [13] resource invocation style, which are based on the popular HTTP.

Additionally, a message-based middleware (via a *Message Bus*) is used where suitable. This provides a coherent communication model with distribution, replication, reliability, availability, redundancy, backup, consistency, and services across distributed heterogeneous systems. The envisioned message bus interconnects all the components and all tiers. It is used for asynchronous notifications and method calls / response handling. As such, it may be used for transmitting measurements that will be routed from producers (e.g., UxVs) to the consumers pertaining to the Middle tier / Data tier (e.g., experiment monitoring, visualisation or data repositories).

B. The RAWFIE Layered Architecture

The RAWFIE architecture consist of four tiers (see Figure 1):

- ***The Front-end tier***: It provides a web based GUI that enables the user to interact with the RAWFIE system. The front-end tier includes the services and tools that RAWFIE provides to the experimenters to define and perform the experimentation scenarios. The main parts of this tier are the *RAWFIE Portal* and the *Experimentation Suite*.
- ***The Middle tier***: It involves a collection of services that provide different management and processing functionalities. The RAWFIE Middle tier is the layer that lies between the UxV testbeds and the experimenters (Front-End tier). It provides the software interfaces needed, and includes useful software components related to security, trust, control and visualization aspects. This tier provides the infrastructure which facilitates the creation and integration of applications in the RAWFIE platform. As shown in Figure 1, the RAWFIE middleware is a virtualized infrastructure, i.e., *Infrastructure as a Service* (IaaS), indicating the maturity and versatility of the developed RAWFIE layered architecture. The main modules of the RAWFIE Middle tier are the *Experiment Authorization*, *Testbed Directory*, *Experiment Control* and *Infrastructure Monitoring*.
- ***The Testbed tier***: It includes the software and hardware components that are needed to run the testbeds and UxVs. Testbeds are comprised of various software components related to the communication with the upper layers (i.e., Middle and Front-End Tier), the management of the infrastructure and the control of the UxV resources. Each testbed available through the RAWFIE federation is comprised of nodes with the same 'x' type (i.e., ground, air or water surface) that exploit the same communication protocols for the ease of their integration in a unified and fully controllable environment (i.e., testbed). Hence, every RAWFIE testbed has two main tiers of its own. The first tier is related to the software components that must be implemented while the second tier describes the resources that are provided.
- ***The Data tier***: It is a collection of repositories that store the different data types generated and collected by RAWFIE components. All the integrated testbeds and resources accessible from the federated facilities are listed. It is a central service that provides the pointers to the different testbeds belonging to the RAWFIE federation. The Data tier manages information relevant to the testbeds and resources (i.e., location, facilities) as well information on the capabilities of a particular resource and its requirements for executing experiments e.g., in terms of interconnectivity or dependencies. The provided data repositories are accessed whenever an experimenter wants to retrieve information related to available testbeds and resources using the respective Front End tool. Data Tier provides a large, secure, cloud-based central repository in which collected data can be anonymized and made available to users.

IV. RAWFIE Testbeds and Resources Management

A. Testbeds Management

RAWFIE offers a registry service called ***Testbed Directory*** where all the integrated testbeds and resources accessible from

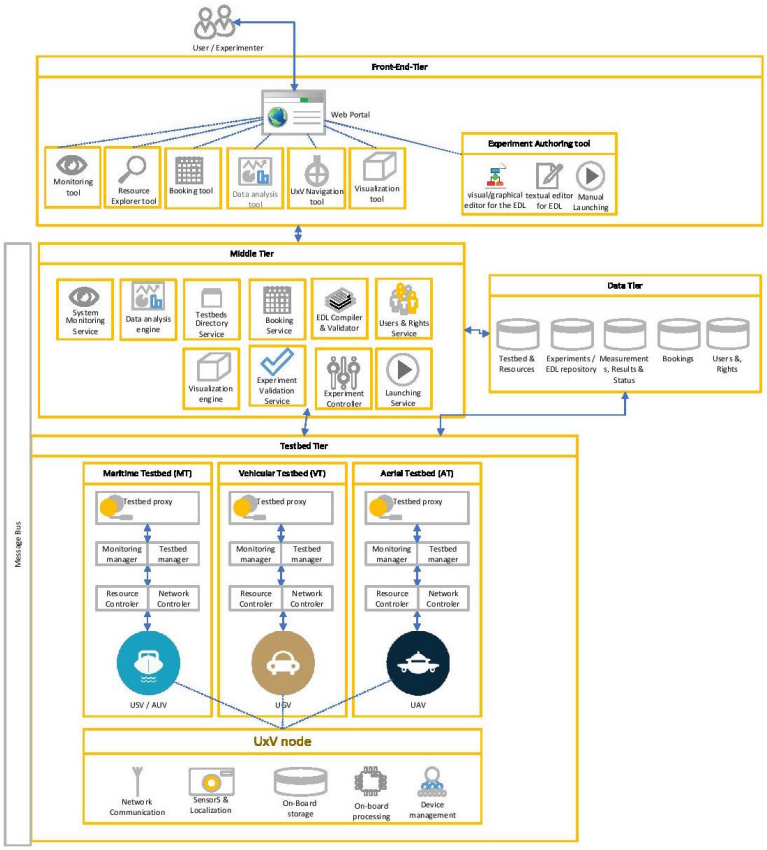

Fig. 1. RAWFIE Architecture.

the federated facilities are listed. The respective tool on the RAWFIE portals side combines all received lists of testbeds and resources that are provided by the Testbed Directory component in a single view. More specifically, the Testbed Directory can be considered as a central service that exposes a list of IP addresses corresponding to the integrated testbeds. These suites gather information related to the testbed that they monitor which is converted by the Front End tool in a human readable format (i.e., textual high level description of testbeds and resources, potential pictures, etc.). The UxV testbeds of RAWFIE are comprised of various software components related to the communication with the upper layers (i.e., Middle and Front-End Tier), the management of the infrastructure and the control of the UxV resources.

Every testbed that will be provided through RAWFIE to potential experimenters is enhanced with peripheral components that ensure the integration of the infrastructure to the rest architecture. At the software level, a ***Testbed Proxy*** is developed in every RAWFIE compliant testbed in order to handle the communication between the facility and the rest tiers of RAWFIE architecture. The Testbed Proxy lies on the server side of each testbed facility and ensures the linkage channels with the Middle and the Front-End tiers. The first important part of the Testbed Proxy is the ***Testbed Manager***. It is responsible for the general functionality of the testbed and its resources. It interfaces a local authorization module for allowing direct booking and executing RAWFIE compliant experiments. Every command received from the ***Ground Control Module*** of the middleware is forwarded to the referenced resource through the exploitation of the ***Resource Controller*** entities. The transformation of the commands to a format that is understandable from each UxV node is performed by this component. Moreover, the Testbed Manager is responsible for data gathering, storing, processing and transmitting them directly to the experimenter or through the middleware elements.

In addition, RAWFIE offers a monitoring service for the available testbeds. The ***Testbed Monitoring Manager*** acts as an observer for the seamless operation of the testbeds resources. It observes and reports the status of the middle tiers modules while periodically checking the current status of the available resources in the facility. At the end of every monitoring 'era' the monitoring manager passes the observation report to the Testbed Manager. Subsequently, this information is forwarded to higher levels of the RAWFIE architecture in order to inform them about the availability of the testbeds resources.

Finally, the ***Network Controller*** manages the network connections and the switching between different technologies in each testbed. For instance, if a problem occurs in the communication of the resource with the ***Resource Controller*** and subsequently with the ***Experiment Controller***, a fall-back interface is engaged. Through this procedure, the other net-

working interface/device is enabled to avoid the uncontrolled operation of the mobile unit and associated damages in the infrastructure. In addition this component is responsible for security issues. The switching alternative can be also triggered by the executed experiment.

B. UxV Resources Management

The other keystone of a testbed facility is the UxV resources. Since the devices provided via the RAWFIE architecture (i.e., UxVs) are meant to cover a wide variety of experimentation/research fields, e.g., mobility, localization, communication/networking analysis etc., they support an expandable infrastructure. UxVs are able to provide a scalable platform with expandable (on demand) networking and smart sensing facilities along with a rather powerful processing board to tackle the need of hosting an *Operating System* (OS).

Each node is equipped with at least two different types of networking interfaces and is able to support the addition of extra interfaces in an 'on demand' approach. On the one hand, by adopting such an architecture, every node should be able to support the sort range communications, i.e., Bluetooth, ZigBee, Infrared, etc., that may be exploited for intra-UxV communication. On the other hand, long range networking interfaces, i.e., IEEE 802.11 WiFi, IEEE 802.16 WiMAX, etc., will be utilized as communication channels between the device and the ***Testbed Proxy***. For instance, the networking interfaces will assist the *over-the-air* (OTA) deployment of node-related applications. Assuming that each UxV platform consists of a main board, new networking interfaces are able to be plugged as modules through commonly used connection standards like USB, etc. The restriction of at least two different networking modules is mandatory in order to support a network problem recovery scheme. If there is a problem with the network module that is currently exploited for the communication of the resource with the Testbed Manager and subsequently with the Experiment Controller, the fall-back mechanism is engaged. Through this procedure, the other networking interface/device is enabled to avoid the uncontrolled operation of the mobile unit and associated damages in the infrastructure.

Every UxV is also accounted as an autonomous sensor system, thus, enhancing the WSN nature of the integrated testbeds and the provision of live data streams. Similarly to the networking layers case, the sensorial system of each node is quite expandable in terms of sensor additions. Hence, the UxV board should provide widely accepted connection standards and communication interfaces, e.g., IEEE 1451, to support the expansion of the sensor pool. The supported sensors can be simple, i.e., microsensors that monitor phenomena such as temperature, humidity, gas, etc., or more advanced like optical sensors.

Apart from the expandable networking layer, the RAWFIE UxV nodes will be equipped also with a processing board. The processing power provided from the embedded processing unit (e.g., 2-core CPU) will be sufficient to support the execution of experimental applications that will be hosted in the installed/supported OS (e.g., Linux). Every resource will also host some build-in modules/applications that will not be controllable from the user/experimenter. Such a module could be a collision checking component which permanently monitors the obstacles in front (i.e., direction of flight) of the device and should force the UxV to stop or follow a different path if a collision is immediately ahead or a power control component that warns / reacts on a low power status.

The ***Resource Controller*** on the UxV node level acts as an agent/daemon that runs on the resources board and invokes actions on the request of the experimenter. Through the module, each resource can be made discoverable to the rest infrastructure while the access control is also ensured. This information is published to the Testbed Manager. Moreover, the Resource Controller is responsible for the dispatch of information related to the current status of the node (i.e., energy reserves, currently active modules, location, velocity etc.). The Monitoring Module communicates periodically with the resources (or the categorys) controller and extracts/requests such information. In addition, the sensed values from the various smart sensors that lie on the board are published to the message bus in order to be made available to the experimenter that initiated/booked the executed experiment. Finally, the Resource Controller is responsible for receiving control commands regarding the altering of the nodes course and fulfils their execution from the referenced resource. If a resource is not able to finish/execute the requested operation, the Testbed Manager is informed through the Resource Controller. The same stands for a successful operation. In general, the Resource Controller can be considered as a *Cloud Robot and Automation system* [14]. The ***Launching Tool*** (see below) interacts with the Experiment Controller (see below) so as to transfer users preferences and instructions regarding the experiment. The Experiment Controller initially, triggers the Experiments and EDL Repository (see below) and receives the users directions, translated in a form of a set of waypoints. These waypoints provide basic information about the preferable locations for each UxV. The set of the waypoints for each robot defines the path that the experimenters have shaped. For the navigation of a robot from its current position to the location described by the next waypoint, the system requires a turn. The main objective of the Resource Controller component is to optimize the navigation process which takes place during a turn. Resource Controller will be able to detect and identify possible safety violations. If the given instructions violate the safety constraints, e.g., the experimenter guides two units at the same position, the Resource Controller identifies and ignores these directions returning to the portal appropriate warning messages.

Additionally, the Resource Controller ensures that the system is performing as intended. If one of the following conditions occurs, automatically, the component activates an emergency scenario.

- The component does not receive any feedback from the units for several time steps.
- The component receives feedback from the units which report severe localization issues.

In such a situation, the component collaborates with the Testbed Proxy so as to navigate the units back to a safe posi-

tion, as soon as possible. In other words, resource controller transforms the navigation problem into an optimization one, which in every time step the goal is to optimize the location of the UxVs so to meet the objectives of the experimenter's needs with respect to a set of constraints. In general, the optimization criterion can be expressed as a function of the robot's positions:

$$J_k = \mathcal{J}(x_k) \tag{1}$$

where $k = 0, 1, 2, \ldots$ denotes the time-index, J_k denotes the value of the optimization criterion at the k-th time-step, x_k denote the position of the robot and $\mathcal{J}$ is a nonlinear function which depends – apart from the robot's positions – on the particular environment where the robots live; for instance, it depends on the location of the various obstacles that are present. At each time-step k, an estimate of J_k is available through robot's sensor measurements,

$$J_k^n = \mathcal{J}(x_k) + \xi_k \tag{2}$$

where J_k^n denotes the estimate of J_k and ξ_k denotes the noise introduced in the estimation of J_k due to the presence of noise in the robot's sensors.

In other words, at each time-instant k, the vector x_k should satisfy a set of constraints which, in general, can be represented as follows:

$$\mathcal{C}(x_k) \leq 0 \tag{3}$$

where $\mathcal{C}$ is a set of nonlinear functions of the robot's positions. As in the case of $\mathcal{J}$, the function $\mathcal{C}$ depends on the particular environment characteristics (e.g. location of obstacles as well as the location o the other robots).

Given the mathematical description presented above, the problem can be mathematically described as the problem of moving x_k to a position that solves the following constrained optimization problem:

$$\begin{array}{ll} \text{maximize} & (1) \\ \text{subject to} & (3)\,. \end{array} \tag{4}$$

The algorithm to be used is based on the so called *Cognitive-based Adaptive Optimization* (CAO) approach [15], [16], [17]. CAO algorithm was originally proposed for the optimization of functions for which an explicit form is unknown but their measurements are available as well as for the adaptive fine-tuning of large-scale nonlinear control systems [18], [19], [20]. In the sequel, the algorithm has been applied in a wide range of robotics related applications. In [21] CAO was used to align a team of flying robots so as to perform surveillance coverage missions over an unknown 3D terrain of complex and non-convex morphology. In the sequel, CAO was fused in [22], with a state-of-the-art visual-SLAM algorithm [23] in a two-step procedure which allowed the alignment of a team of aerial robots to perform terrain surveillance coverage over a terrain of arbitrary morphology by using only onboard vision mechanism. Moreover, CAO was also implemented in the case of teams Autonomous Underwater Vehicles (AUVs), to fully-autonomously navigate them when deployed in exploration of unknown static and dynamic environments towards providing accurate static and/or dynamic maps of the area [24]. Another application in the case of mobile robots is presented in [25], where CAO was utilized to facilitate navigation in an unknown complex environment, while interacting with humans considering their comfort. Recently, CAO was employed in order address a twofold challenge of realistic search and rescue robotic exploration operations; the ability to efficiently handle multiple temporal goals while satisfying the mission constraints [26]. The proposed strategy was able to effectively address multiple non binary temporal goals utilizing a low computational cost cognitive optimization algorithm.

The Resource Controller navigates simultaneously all the units of the squad. It is worth noting that the time needed for each robot to reach its desired location is not the same for all units. Thus, the turn concludes when all the robots reach their next location. Additionally, it is worth mentioning that in case of emergence, the RC collaborates with the Testbed Proxy so as to navigate the units back to a safe position, as soon as possible. A core subcomponent of Resource Controller is the controller that translates the experimenter's instructions into a 'global form' of waypoints or mission objectives and transmits these points or objectives to the controlled units. In other words, this sub-module converts the experimenter's instructions into a reference scheme, compatible with the build-in navigation system of the UxVs. An existing global remote controller tool (www.noptilus-fp7.eu) has been already developed for the guidance of AUVs [27].

V. RAWFIE Data Management

RAWFIE will support multiple, simultaneous, diverse experiments from physical to application layers. Therefore, the ***Experiment Controller*** includes a service enabling the data collection, analysis and processing. This service is responsible for storing the measurement streams in the underlying infrastructure. RAWFIE provides a large, secure, cloud-based central repository in which collected data can be anonymized and made available to users. Furthermore, RAWFIE adopts resource-aware versions of data mining algorithms, which are appropriate for deployment and execution over the IoT. More specifically, RAWFIE devices (i.e., UxV resources) that will make part of the IoT will yield massive volumes of disparate, dynamic, heterogeneous, and geographically distributed data. The raw data from such devices are efficiently managed (cleaned, homogenized, normalized, transformed) to deliver usable information in order to apply knowledge discovery and data mining techniques, which can deliver insights. The developed knowledge discovery algorithms will be able to model for data dependencies, interactions, redundancies, as well as for the spatiotemporal dimensions. For example, any model-building algorithm for the IoT environment should account for the spatial dimensions of the data sources, modelling for potential data redundancies, e.g., neighbouring nodes measuring the same quantity. In addition, the developed tools will deal with the temporal dimension and the dynamic nature of the data being able to detect and adapt to changes of the underlying data generating distribution

The ***Testbed and Resources Repository*** contains relevant information about available testbeds (federated through the RAWFIE platform) and their resources, such as: (i) Testbed name and testbed URL (if a dedicated access portal is also available for a specific testbed), (ii) Description and overview of each testbed facilities, and corresponding resources (e.g., available UxVs), (iii) Overview of the reservations linked to each specific testbed, (iv) Description and overview of specific resources (e.g., type, technologies, tests that can be executed, and so on) for each given testbed and (v) Information on the capabilities of a particular resource and its requirements for executing experiments e.g. in terms of interconnectivity or dependencies.

The ***Experiments and EDL Repository*** provides the necessary functionalities for having the experiments and EDL related data stored into the data tier. The EDL scripts, templates and pre-defined constraints are stored in the appropriate format in order to be efficiently retrieved by the rest component of the RAWFIE framework. It should be noted that the appropriate metadata are adopted for each experiment. Finally, additional repositories are adopted to store bookings, reservations of resources as well as management of authorizations and access rights. All the retrieved measurements of the executed experiments are also stored in order to be the subject of further processing.

VI. RAWFIE Experiments Management

The ***Experiment Description Language*** (EDL) is a *Domain Specific Language* (DSL) for creating simple as well as complex experiment scenarios for the IoT domain. The EDL is designed for the RAWFIE purposes aiming to help domain experts or non-experienced users (e.g., experimenters) to effectively create and handle such type of scenarios. The major goal of the EDL is the provision of a high level of abstraction that shields experimenters from the complexities of the underlying implementation of RAWFIE platform. In the most interesting case, the EDL provides elements for handling resource requirements/configuration, location/topology information, task description, testbed-specific commands etc. Its syntax is simple and combines some common characteristics of well-known XML based languages. The EDL is built with the help of the Xtext framework[7]. Figure 2 depicts a small part of the proposed EDL grammar.

An experiment as realized through the EDL terminology is seen to have the following parts:

- *Metadata section*. It contains generic information related to each experiment like the name, the date, etc. This information is important to define the necessary description for each experiment and, thus, to facilitate the efficient management of the available experiments.
- *Requirements section*. it contains information related to the requirements of each experiment in terms of the testbed data, the location, the duration or the distance that the nodes should cover during the experiment execution. In addition, in this section, the experimenter should define the number of nodes that will be involved in the experiment and, thus, the RAWFIE platform is capable of knowing the needs for the experiments under consideration.
- *Declarations section*. It concerns the necessary declarations like constants and variables declaration adopted to store data during the experiment execution. The discussed declarations are the key element to connect the experiment business logic with the data retrieved by UxVs and perform processing in a higher level than the device itself.
- *Execution section*. It involves commands related to the management of the core business logic of each experiment. The EDL offers statements for the nodes or group of nodes management. Every aspect of nodes / groups behaviour can be realized with specific terminology in the execution section. In addition, specific statements are devoted to: (i) waypoints management; (ii) time line management (e.g., sequential or parallel execution); (iii) coordination management; (iv) control management (e.g., activation / deactivation of sensors); (v) configuration management (e.g., data management in each node); (vi) communication management (e.g., change in network interfaces).

[7]https://eclipse.org/Xtext/

```
Experiment:
        'Experiment'
            metadata = MetadataSection
            (requirements=RequirementsSection)?
            (declarations=DeclarationsSection)?
            execution=ExecutionSection
        '~Experiment'
;

/********** Metadata Section **********/
MetadataSection:
    'Metadata'
         met+=Metadata
    '~Metadata';

Metadata:
    'Name' name = ID
    (experimentVersion=Version)?
    (experimentDescritpion=Description)?
    (experimentDate=Date)?;
```

Fig. 2. A part of the EDL grammar.

It should be noted that 'typical' commands originated in legacy programming languages are also included in the EDL. Hence, assignments, conditionals statements (i.e., if, switch) and iterations (i.e., for, while) are also in place. In Figure 3, we present a small part of an EDL script related to the definition of the behaviour of a node.

On top of the EDL terminology, a *textual editor* is provided in two modes: (i) a standalone version that works as an Eclipse plugin and (ii) a Web based version. In both modes, the EDL editor is responsible to provide the necessary functionalities to the experimenters towards the creation, update, compilation and validation of their experiments. The editor is a collection of tools for defining experiments and authoring EDL scripts through the RAWFIE Web portal or through Eclipse. Rich

```
Node
    ID node1
    Route[
            WP<1, 1 - 1 - 12>
            WP<3, 3 - 1 - 15>
            WP<7, 7 - 1 - 15>
            WP<10, 12 - 1 - 15>
        ]
    DataManagement
            Time 7 Algorithm average(history = 10)
            ~DataManagement
    NodeCommunication
            NIC WiFi
            ~NodeCommunication
    DataManagement
            Time 9 Algorithm average(history = 5)
            ~DataManagement
~Node
```

Fig. 3. A part of an EDL script.

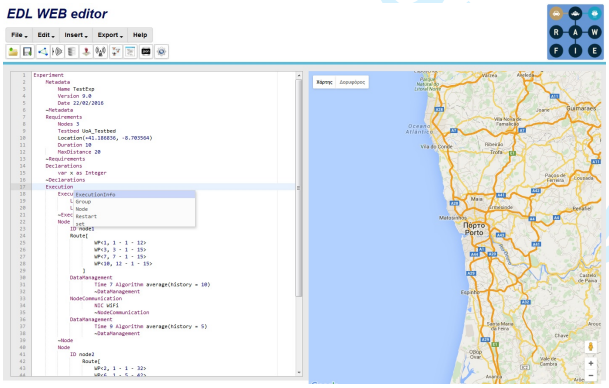

Fig. 4. A part of an EDL script.

editing facilities are supported together with an advanced content assist and checking mechanism at syntax time. The EDL keywords are highlighted with different color while the code folding functionality enables blocks of code to be hidden or expanded at will. Some of the provided functionalities are: (i) syntax coloring; (ii) content assist; (iii) validation and quick fixes; (iv) code completion; (v) error checking. A set of additional tools for syntactic and semantic validation are also available. The editor gives 'access' to the EDL concepts through which an experiment will be defined. Future enhancement is the synchronization of the textual editor with a visual editor where experimenters will have the opportunity to define nodes routes and other related information directly on map of the area under consideration. In Figure 4, we see a snapshot of the provided editor where the content assist functionality gives us hints about the upcoming commands that should be inserted in an experiment.

The EDL validator is responsible for performing syntactic and semantic analysis on the provided EDL scripts. The validation will be performed on top of the proposed EDL model that will be based on the EDL grammar. The validator accesses the provided script and identify any semantic errors that could jeopardize the execution of the experiment. Specific constraints should be fulfilled when the experiment workflow is defined. These constraints are continuously checked by the proposed editor and in case some of them are validated to be false, the errors will be presented to the experimenters through various means (e.g., warnings). Finally, when no errors are present, the component will have the opportunity to generate specific files e.g., part of the final code to be uploaded in the UxVs, input to the validator, input to the Testbed Proxy). The main responsibilities of the component are:

- It provides syntactic and semantic validation of each experiment workflow.
- It applies a set of constraints that should be met in order to have a valid experiment.
- It is capable of applying semantic checking for nodes communication, spatio-temporal management, sensing and data management.
- It performs code generation in the appropriate format in order to be uploaded into the RAWFIE nodes.

Schedules and launches executions of the experiments together with the assigned booked resources The ***Launching Service*** is responsible for scheduling the execution of experiments. It supports two aspects of launching:

1) *Short-term launching*: The service through a specific interface gives the opportunity to experimenters to execute in real time pre-defined and pre-approved experiments stored in the RAWFIE system. It should be noted that this functionality is available if the corresponding testbed is already configured (i.e., UxVs are in place and the necessary code is uploaded to nodes).
2) *Long-term launching*: The service identifies which experiment should be executed according to the available bookings. It should be noted, that the Launching Service executes only authorized and approved experiments based on spatio-temporal constraints.

The ***Experiment Controller*** is responsible to monitor the smooth execution of each experiment. The main task is the monitoring of the experiment execution while acting as 'broker' between the experimenter and the resources in (near) real time. The Experiment Controller provides capabilities to support 'complex' experiments possibly involving multiple testbeds as well as to support the manual override of specific instructions to the resources while the experiment is running. The Experiment Controller identifies if the experiment runs smoothly and will inform the upper layer in order to present the necessary information to the experimenter. In addition, the Experiment Controller controls the data sent back by the nodes. Hence, the Experiment Controller, among others, will have access in the Data tier in order to be capable of retrieving the necessary data. The use of the Experiment Controller in the middle tier gives RAWFIE the opportunity to include more intelligence in the functionalities provided related to the execution of the experiments and the level description to waypoints (e.g., implement patterns of vehicle movement like expanding ring).For instance, the system could have a view on the correct execution of the experiment workflow, to combine multiple UxV / Testbed types in the same experiment or to be able to monitor the execution of more complex scenarios.

VII. RAWFIE Graphical User Interface

A. Portal

The ***RAWFIE Portal*** provides a Web interface to federation resources and services. The goal of the Portal is to provide a user-friendly *Graphical User Interface* (GUI), acting as a central point of access to all the necessary resources and services used by the experimenters. Another objective of the Portal is to illustrate all the essential information for the RAWFIE federation that the experimenters should take advantage of, in a straightforward manner. A well-organized tutorial and any other kind of documentation needed is provided to the experimenters for considering the design, the use and the variety of resources, the testbed facilities, any available real experiment or simulation tools, for data refinement and replication of their experiments, etc. For instance, a wiki could be used as repository of information related but not limited to the RAWFIE design, the experimenters potentiality, the testbed facilities and the resources supported. These functionalities are available to all possible future experimenters that may be interested in RAWFIE federation and want to explore its capabilities. The initial central starting place for the authenticated experimenters of the RAWFIE infrastructure is the browsing of the available testbed facilities and their resources. An experimenter will be able to discover either in a simple drop down list or in a more elaborated visualized way the availability of the active testbed facilities. Furthermore, for the active resources, information such as their current status, (e.g., battery state), and their technical capabilities (e.g., sensing facilities), will be also available to the experimenters.

A ***Visualization Engine*** is provided to process the received resource traces and dispatches the result to the visualization tool of the Front End tier. This engine is a software consisting of several sub-modules and data components (related to the raw data from UxV on board sensors) with rendering for high-fidelity real-time data visualization in 2D or/and 3D. Additionally, the Visualization Engine can be used for creating and managing collaborative sensing missions, working on desktop computers for the operator to have the full view of the sensed information and also on the web for the visualization of the experiments results.

The ***Booking*** functionality allows experimenters to book a spatiotemporal interval (Figure 5) for running their experiments, thus, providing automatic coordination in the use of the testbed resources among experimenters. Experimenters are allowed to access the testbed list first and, for a given testbed, reserve the UxV resources required for the execution of the defined experiment beforehand through the experiments editor tool. By making use of suitable software interfaces provided by the Data Tier, the Booking tool should query the data storage in order to:

1) Visualize, in a calendar view, the available dates and timeslots for each testbeds resources
2) Select the preferred date, timeslot and space fragment in a testbed (based on the availability of the required nodes) to execute the desired experimentation scenario

Once the triplet (date, timeslots/duration, space/sub-space) is chosen and booked for a testbed, the reserved resources

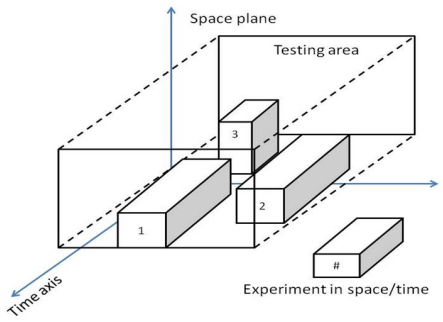

Fig. 5. Experiments in spatiotemporal intervals.

of the corresponding testbed, from that time on, will not be available for any other experimenter. Hence, busy or available UxV nodes per timeslot on a given testbed are easily identified by the experimenters as 'booked' or 'available'. The association with the chosen experimentation scenario to run is visible as well. It is possible to edit, enhance and add new experimentation scenarios based on the reserved resources from the selected testbed as well as to bind a part or even the entire testbed to the related experimentation scenarios in any available timeslots. Additional functionalities such as notification mechanisms to remind to the experimenter the date and the timeslot allocated for running his/her experiment on the RAWFIE infrastructure should also be envisaged to improve the user experience.

Furthermore, the ***Monitoring Tool*** manages the presentation of the information needed for monitoring the status of the nodes and the data collected during the experiments. Moreover, specific parameters calculated as part of the experiments outcomes (e.g., benchmark parameters concerning the quality of the network, and the performance of the adopted communication technologies) are also presented. An example of information for the current status of each of the resources in a given testbed could be the energy reserves, the status of the different networking and sensing modules and the position in the testbed site.

Finally, the ***Data Analysis Tool*** starts data analysis learning tasks and displays their results, visualizes data from the 'Measurements, Results, Status' repository, browses the results from past analysis, provides commands to the Data Analysis Engine and specifies data analytical/learning tasks to be executed on specific streaming datasets.

VIII. Conclusion

RAWFIE will be the first federation that will combine different real-world testbeds of unmanned vehicles into a common framework for performing experiments in terrestrial, aerial and maritime environments. It will be diverse by combining different UxV technologies and resources, spanning the spectrum of wireless technologies available today. The RAWFIE architecture is quite flexible, with explicitly defined

procedures and system interfaces that makes it easy to incorporate additional technologies, including those that do not exist today.

RAWFIE aims to be a cloud-based infrastructure where each interested/signed experimenter gathers resources into its own isolated 'slice' and then configures them to support the designed experiment(s). RAWFIE components either on the middle tier or on the front end tier will be offered to users as Infrastructure as a Service (IaaS) and Platform as a Service (PaaS) respectively. Given the state of the art on the cloud computing technologies the objective of the RAWFIE project is to adapt different existing technologies in order to achieve better scalability and performance by providing excellent speed of computations and high availability as well. Moreover, such a virtualized approach ensures the provision of reliable and flexible backup/recovery solutions. The data reside on the cloud and not on physical devices simplifying the process of storing and recovering data and also helps in the avoidance of networking issues that frequently arise in distributed data gathering schemes.

ACKNOWLEDGMENT

This project is funded by the European Commission (FIRE+ challenge, Horizon 2020) that aims to provide for research, technological development and demonstration under grant agreement no 645220 (RAWFIE)

REFERENCES

[1] K. Goldberg, "Robotics: Countering singularity sensationalism," *Nature*, vol. 526, no. 7573, pp. 320–321, 2015.

[2] M. Dunbabin and L. Marques, "Robots for environmental monitoring: Significant advancements and applications," *Robotics & Automation Magazine, IEEE*, vol. 19, no. 1, pp. 24–39, 2012.

[3] B. Grocholsky, J. Keller, V. Kumar, and G. Pappas, "Cooperative air and ground surveillance," *Robotics & Automation Magazine, IEEE*, vol. 13, no. 3, pp. 16–25, 2006.

[4] S. Chen, H. Xu, D. Liu, B. Hu, and H. Wang, "A vision of iot: Applications, challenges, and opportunities with china perspective," *IEEE Internet of Things Journal*, vol. 1, no. 4, pp. 349–359, 2014. [Online]. Available: http://dx.doi.org/10.1109/JIOT.2014.2337336

[5] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[6] A. Zanella, N. Bui, A. P. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 22–32, 2014. [Online]. Available: http://dx.doi.org/10.1109/JIOT.2014.2306328

[7] J. A. Stankovic, "Research directions for the internet of things," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 3–9, 2014. [Online]. Available: http://dx.doi.org/10.1109/JIOT.2014.2312291

[8] Y. Al-Hazmi, A. Willner, O. O. Ozpehlivan, D. Nehls, S. Covaci, and T. Magedanz, "An automated health monitoring solution for future internet infrastructure marketplaces," in *2014 26th International Teletraffic Congress (ITC), Karlskrona, Sweden, September 9-11, 2014*, 2014, pp. 1–6.

[9] G. Gonzalez, R. Perez, J. Becedas, M. J. Latorre, and F. Pedrera, "Measurement and modelling of planetlab network impairments for fed4fire's geo-cloud experiment," in *2014 26th International Teletraffic Congress (ITC), Karlskrona, Sweden, September 9-11, 2014*, 2014, pp. 1–4.

[10] G. Coulson, B. Porter, I. Chatzigiannakis, C. Koninis, S. Fischer, D. Pfisterer, D. Bimschas, T. Braun, P. Hurni, M. Anwander, G. Wagenknecht, S. P. Fekete, A. Kröller, and T. Baumgartner, "Flexible experimentation in wireless sensor networks," *Commun. ACM*, vol. 55, no. 1, pp. 82–90, 2012.

[11] R. Perrey and M. Lycett, "Service-oriented architecture," in *Applications and the Internet Workshops, 2003. Proceedings. 2003 Symposium on*. IEEE, 2003, pp. 116–119.

[12] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, and S. Weerawarana, "Unraveling the web services web: an introduction to soap, wsdl, and uddi," *IEEE Internet computing*, no. 2, pp. 86–93, 2002.

[13] R. Fielding, "Representational state transfer," *Architectural Styles and the Design of Netowork-based Software Architecture*, pp. 76–85, 2000.

[14] B. Kehoe, S. Patil, P. Abbeel, and K. Goldberg, "A survey of research on cloud robotics and automation," *Automation Science and Engineering, IEEE Transactions on*, vol. 12, no. 2, pp. 398–409, 2015.

[15] D. Scaramuzza, M. Achtelik, L. Doitsidis, F. Friedrich, E. Kosmatopoulos, A. Martinelli, M. Achtelik, M. Chli, S. Chatzichristofis, L. Kneip, D. Gurdan, L. Heng, G. Lee, S. Lynen, M. Pollefeys, A. Renzaglia, R. Siegwart, J. Stumpf, P. Tanskanen, C. Troiani, S. Weiss, and L. Meier, "Vision-controlled micro flying robots: From system design to autonomous navigation and mapping in gps-denied environments," *Robotics Automation Magazine, IEEE*, vol. 21, no. 3, pp. 26–40, Sept 2014.

[16] A. Amanatiadis, S. Chatzichristofis, K. Charalampous, L. Doitsidis, E. Kosmatopoulos, P. Tsalides, A. Gasteratos, and S. Roumeliotis, "A multi-objective exploration strategy for mobile robots under operational constraints," *Access, IEEE*, vol. 1, pp. 691–702, 2013.

[17] A. Kapoutsis, S. Chatzichristofis, L. Doitsidis, J. de Sousa, J. Pinto, J. Braga, and E. Kosmatopoulos, "Real-time adaptive multi-robot exploration with application to underwater map construction," *Autonomous Robots*, pp. 1–29, 2015. [Online]. Available: http://dx.doi.org/10.1007/s10514-015-9510-8

[18] E. Kosmatopoulos, M. Papageorgiou, A. Vakouli, and A. Kouvelas, "Adaptive fine-tuning of nonlinear control systems with application to the urban traffic control strategy tuc," *IEEE Trans. Contr. Syst. Technol.*, vol. 15, no. 6, pp. 991–1002, 2007.

[19] E. Kosmatopoulos, "An adaptive optimization scheme with satisfactory transient performance," *Automatica*, vol. 45, no. 3, pp. 716–723, 2009.

[20] E. Kosmatopoulos and A. Kouvelas, "Large-scale nonlinear control system fine-tuning through learning," *IEEE Trans. Neural Netw.*, vol. 20, no. 6, pp. 1009–1023, 2009.

[21] A. Renzaglia, L. Doitsidis, A. Martinelli, and E. Kosmatopoulos, "Multi-robot 3d coverage of unknown areas," *Int. J. Robot. Res.*, vol. 31, no. 6, pp. 738–752, 2012.

[22] L. Doitsidis, S. Weiss, A. Renzaglia, M. W. Achtelik, E. Kosmatopoulos, R. Siegwart, and D. Scaramuzza, "Optimal surveillance coverage for teams of micro aerial vehicles in gps-denied environments using onboad vision," *Auton. Robot.*, vol. 33, no. 1-2, pp. 173–188, 2012.

[23] S. Weiss, M. Achtelik, L. Kneip, D. Scaramuzza, and R. Siegwart, "Large-scale nonlinear control system fine-tuning through learning," *Journal of Intelligent & Robotic Systems*, vol. 61, pp. 473–493, 2011.

[24] A. Kapoutsis, S. Chatzicristofis, L. Doitsidis, J. Sousa, J. Pinto, J. Braga, and E. Kosmatopoulos, "Real-time adaptive multi-robot exploration with application to underwater map construction," *Autonomous Robots*.

[25] J. R. Martinez, A. Renzaglia, A. Spalanzani, A. Martinelli, and C. Laugier, "Navigating between people: a stochastic optimization approach," in *Proc. IEEE Int. Conf. Robot. Autom.*, St. Paul, MN, USA, 2012.

[26] A. Amanatiadis, S. Chatzichristofis, K. Charalampous, L. Doitsidis, E. Kosmatopoulos, F. Tsalides, A. Gasteratos, and S. Roumeliotis, "A multi-objective exploration strategy for mobile robots under operational constraints," *IEEE Access*, no. 99, pp. 1–1, 2013.

[27] S. Chatzicristofis, A. Kapoutsis, E. Kosmatopoulos, L. Doitsidis, D. Rovas, and J. Sousa, "The noptilus project: Autonomous multi-auv navigation for exploration of unknown environments," in *IFAC Workshop on Navigation, Guidance and Control of Underwater Vehicles (NGCUV2012)*, vol. 3, 2012, pp. 373–380.