applied sciences

MDPI

Article

Survey of Mission Planning and Management Architectures for Underwater Cooperative Robotics Operations

Néstor Lucas Martínez *, José-Fernán Martínez-Ortega, Pedro Castillejo and Victoria Beltrán Martínez

Departamento de Ingeniería Telemática y Electrónica (DTE), Escuela Técnica Superior de Ingeniería y Sistemas de Telecomunicación (ETSIST), Universidad Politécnica de Madrid (UPM), C/Nikola Tesla, s/n/, 28031 Madrid, Spain; jf.martinez@upm.es (J.-F.M.-O.); pedro.castillejo@upm.es (P.C.); mv.beltran@upm.es (V.B.M.)

* Correspondence: nestor.lucas@upm.es; Tel.: +34-910-673-499

Received: 1 December 2019; Accepted: 30 January 2020; Published: 6 February 2020

Featured Application: Mission management for cooperative autonomous robotics.

Abstract: Almost every research project that focuses on the cooperation of autonomous robots for underwater operations designs their own architectures. As a result, most of these architectures are tightly coupled with the available robots/vehicles for their respective developments, and therefore the mission plan and management is done using an ad-hoc solution. Typically, this solution is tightly coupled to just one underwater autonomous vehicle (AUV), or a restricted set of them selected for the specific project. However, as the use of AUVs for underwater operations increases, there is the need to identify some commonalities and weaknesses of these architectures, specifically in relation to mission planning and management. In this paper, we review a selected number of architectures and frameworks that in one way or another make use of different approaches to mission planning and management. Most of the selected works were developed for underwater operations. Still, we have included some other architectures and frameworks from other domains that can be of interest for the survey. The explored works have been assessed using selected features related to mission planning and management, considering that underwater operations are performed in an uncertain and unreliable environment, and where unexpected events are not strange. Furthermore, we have identified and highlighted some potential challenges for the design and implementation of mission managers. This provides a reference point for the development of a mission manager component to be integrated in architectures for cooperative robotics in underwater operations, and it can serve for the same purposes in other domains of application.

Keywords: mission management; mission plan; mission plan adaptation; cooperative robotics; system architectures; agent virtualization; mission plan dispatching and execution

1. Introduction

In recent years, the use of autonomous underwater vehicles (AUVs) for cooperative robotics in underwater operations has increased. This interest includes their use in military interventions [1–3], scientific interventions like the survey and exploration of the Arctic Ocean [4,5], and other marine interventions like, for instance, underwater infrastructure maintenance [6–8] and oil spill response [9–11]. The essential information structure for these operations has been identified as the mission. Kothari et al. defined a mission as a set of goals to be achieved [12], and a mission plan as the schema for achieving the goals of a mission, expressed in terms of tasks like maneuvers, communication, sensors, and

payload primitives. Fernández –Perdomo et al. described a mission as the set of tasks a vehicle must execute [13]. In their proposal, they described the mission life cycle as the process depicted in Figure 1. A mission is first created by a human operator, either by hand or using the aid of an automated planning application. The created mission is validated, and then transferred to the vehicle, where it is executed. The proposed life cycle includes an optional step for replaying the mission in a simulator for offline analysis.

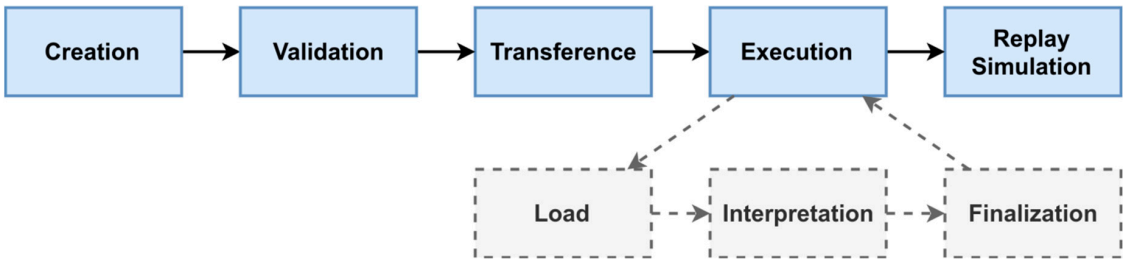

Figure 1. Mission life cycle stages.

Mission management can then be defined as the set of processes and techniques to control the mission life cycle. The study of mission management for AUV interventions has attracted the interest of the research community, but mostly focused on the creation, and maybe the validation of the mission plan.

The definition of the mission plan is equivalent to the definition of a plan as a sequence of actions to achieve a goal used in AI literature [14]. And consequently it is common that mission plans for underwater operations are generated with the aid of automated planning algorithms adapted from the AI literature. For instance, a comprehensive survey of mission planning solutions used for autonomous marine vehicle fleets was done by Thompson and Guihen in [15]. In their work, they restricted the planning paradigms applicable to marine operations to the following four state-action planning methods:

- Deterministic planners: the most common and basic ones, that typically assume that he execution environment does not change at all.
- Multi-Agent planners: where the agents are individual vehicles participating in a mission.
- Temporal planners: where specified goals must be reached in a finite time.
- Non-Deterministic planners: referred to as planners working with partially observable Markov decision processes, and that provide a policy to be used to map any state to a suitable action.

Other works published related to mission management include the survey on autonomous mission planning and management systems done by Atyabi et al. for AUVs and Unmanned Aircraft Vehicles (UAVs) [16]. Their survey focuses mainly on the autonomy and the situation awareness features of the selected architectures. Moreover, most of the architectures included in their work are focused on the mission planning part, including route planning and re-planning. MahmoudZadeh et al. conducted another survey on the state-of-the-art for autonomous mission planning and task managing for unmanned vehicles [17]. Again, most of the cited architectures were focused on topics related to the mission planning part, including rout planning, resource allocation and re-planning.

Nevertheless, we find the need to address other aspects related to mission management beyond the creation of the plan.

First of all, still related to the mission planning, the AI literature related to automated planning includes several specifications for how to describe a plan. However, usually the literature about mission planning and management architectures for cooperative robotics operations do not cover this.

Further, we have to take into account that the underwater environment is usually considered as unreliable and uncertain. And when there is uncertainty, an agent cannot ensure that it will be able to achieve the expected goals [18]. Consequently, mission management architectures should take into

account how to manage unexpected and unforeseen events. Typically, this has been done by delegating the responsibility to adapt to the new situation to the AUV, or to abort the mission [19]. That is because the classical transference of the mission plan was performed by loading the full mission into the vehicle before being launched [20]. However, the advances in underwater communications [21] may enable other possibilities. In particular, it is of interest to know how self-adaptation is considered from the global mission management point of view. Especially in relation to the use of decentralized control patterns [22] that include both the autonomous vehicles and a global management component.

The aforementioned improvement in underwater communications also enables other mechanisms for dispatching the mission plan from the global mission management to the AUVs participating in a mission, thus constituting another topic of interest not covered in previous surveys.

Finally, the availability of a greater variety of vehicles with different capabilities is increasing. Consequently, the possibility of using different vehicles with different mission management capabilities in the same mission is also increasing. This situation presents an opportunity for the use of virtualization mechanisms in the mission management architectures that can enable the integration of legacy systems with novel ones.

In summary, the main contributions for this survey can be described as:

- An identification of a set of four features of interest that a mission manager for cooperative robotics should take into consideration, specially, but not restricted to underwater operations, namely:
    - Mission plan specification.
    - Use of self-adaptive models.
    - Use of virtualization.
    - Mission plan dispatching and execution.
- A categorization of the four features identified for allowing the assessment of the selected architectures.
- A survey of a selected number of architecture proposals for cooperative robotics, highlighting their most relevant characteristics.
- A discussion of the open issues found.
- A discussion of possible future works that can be carried out so that the quality of future mission management architectures will be improved.

The rest of this paper is organized as follows. Section 2 provides a comprehensive description of the selected features to be compared among the selected mission management architectures. Section 3 provides a description of the selected architectures, paying special attention to the features introduced in Section 2. Section 4 provides a summary comparison of the reviewed architectures, and a discussion of their strengths and weaknesses. Finally, Section 5 covers the conclusions from the survey and introduces some possible future works worth for exploration.

2. Selected Features for a Mission Manager for Underwater Cooperative Robotics Operations

In this section, we discuss in detail the parameters that we have offered in the introduction. In particular, there are four different features that have been defined as of interest for the design of a mission planning and management architecture. To ensure the assessment is as objective and accurate as possible, we have defined a set of categories for each of the selected features.

It is important to recall that these categories allow us to provide a classification for the studied architectures. But they do not establish a ranking system.

2.1. Mission Plan Specification

As stated in the introduction, the mission is the essential information structure for underwater operations. Its main constitutive part, the mission plan, matches the formal definition of a plan as a sequence of actions to achieve a goal [14]. In consequence it is common to use an AI automated planning approach to generate the mission plan.

Whether the mission plan follows a deterministic or a non-deterministic paradigm, with or without temporal constraints, and considering one or multiple agents, from an architectural point of view it is important to know how the mission plan is specified. In this survey, we have taken into account the selection of planning paradigms from Thompson and Guihen [15], with the additional consideration of the use of a hierarchical task description, for instance, a hierarchical task network (HTN) [23].

The mission plan specification feature is categorized regarding the planning paradigm identified in the surveyed architecture, with special attention to the way the mission plan is described. For the determination of the categories, we have considered the already existing formal description languages commonly used for automated planning, like: the Stanford Research Institute Problem Solver (STRIPS) [24]; the Planning Domain Definition Language (PDDL) [25]; the Relational Dynamic influence Diagram Language (RDDL) [26]; or any ad-hoc specification created for the architecture proposal. For PDDL, we also take into consideration any of its updates, like PDDL 2.1 [27], PDDL 2.2 [28] and PDDL 3.0 [29–31], and extensions, like the Probabilistic PDDL (PPDDL) [32] and the Multi-Agent PDDL (MA-PDDL) [33].

The categories we have identified for this feature are summarized in Tables 1 and 2. We have decided to split the assessment into two different categorical groups, as the categories from the second group can be present or absent indistinctly in any of the categories of the first group. The first group focuses on the use of deterministic or non-deterministic planning paradigms. A graphical scheme for the proposed assignment of categories for the first group is shown in Figure 2, and the description of suggested assessment is summarized in Table 1.

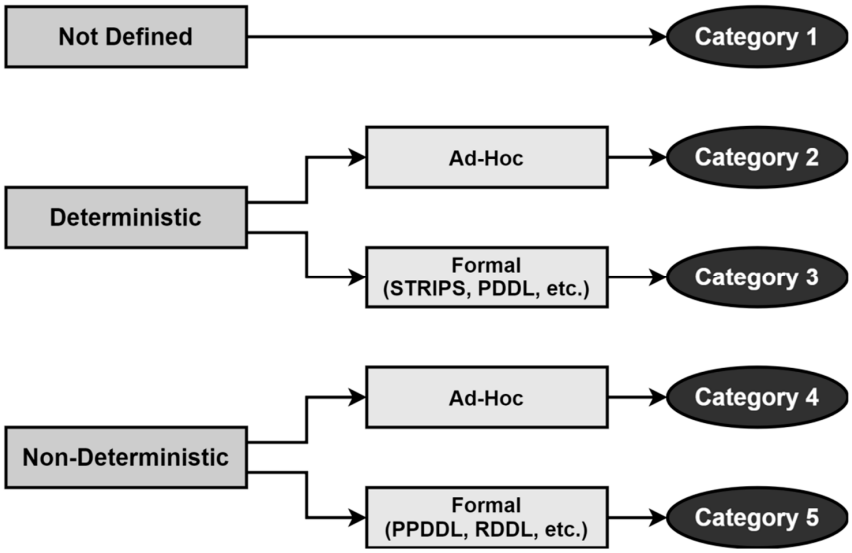

Figure 2. Mission plan specification assessment diagram (1st group).

Table 1. Mission plan specification assessment (1st group).

Category	Description
1	The architecture proposal does not describe the way the mission plan is specified.
2	The architecture proposal follows a deterministic approach and uses an ad-hoc mission plan specification, or proposes a new one.
3	The architecture proposal follows a deterministic approach and uses either STRIPS, any of the PDDL versions or any other formal description language already defined for automated planning.
4	The architecture proposal follows a non-deterministic approach and uses an ad-hoc mission plan specification or proposes a new one.
5	The architecture proposal follows a non-deterministic approach and uses PPDDL, RDDL or any other formal description language already defined for automated planning.

Table 2. Mission plan specification additional categories assessment (2nd group).

Category	Description
M	The architecture proposal includes a multi-agent approach.
T	The architecture proposal includes a temporal constraint approach.
H	The architecture proposal includes a hierarchical task network.

The second assessment group focuses on the use of other planning paradigms that overlap the assessment made in the first group. Here, we included the aforementioned multi-agent, temporal constraint and hierarchical tasks paradigms that can be used both for deterministic and non-deterministic planning. In this case, the category is identified by a letter that is added to the first group category according to the assessment made. For instance, an architecture that uses deterministic planning with a formal specification and a multi-agent approach with temporal constraints will be categorized as "3MT". The description of the assessment categories for this second group is summarized in Table 2.

2.2. Use of Self-Adaptive System Models

In an uncertain and unreliable environment like the underwater one it is common to design mission management architectures considering self-adaption to unforeseen events in order to improve the reliability of the vehicles entrusted with the execution of a mission plan. To better understand this concept, it is interesting to know how a self-adaptive system has been usually defined in the literature. In [34] Martin et al. defined an adaptive system as the set of elements interacting with each other that has at least one process controlling the system adaptation to increase its efficiency to achieve its goals. Weyns et al. described in [22] a self-adaptive system as that kind of system that has the ability to adapt itself to the changes in its execution environment and internal dynamics with the purpose of continuing to achieve its goals. Salehie et al. [35] defined self-adaptive systems as those systems that aim to adjust various artifacts or attributes in response to changes in the self or in the context of the system.

There are several self-adaptive models that have been formalized in the literature regarding the design of self-adaptive systems: the Observe, Orient, Decide, Act loop (OODA) [36]; the Event, Condition, Action (ECA) [37]; the Sense, Plan, Act (SPA) [38]; and the autonomic manager model [39,40], better known as MAPE-K due to the names of its four functional parts (Monitor, Analyze, Plan, Execute), and the use of shared knowledge among them. The OODA loop has been used as a reference in one of the architectures included in this survey, while the ECA and the SPA are typical self-adaptive models used in robotics. However, the most prominent model in recent years, mostly in control systems, has been the MAPE-K. The MAPE-K model defines the concept of an autonomic manager as a component that implements an intelligent control loop consisting in four main functional parts [40], as shown in Figure 3:

- A Monitor function that gathers the information of the managed resource through the sensors.
- An Analyze function that correlates the gathered information from the Monitor to known situations.
- A Plan function that provides the relevant actions to be executed according to the results from the Analyze function.
- An Execute function that dispatches and control the execution of the actions resulting from the Plan function through the effectors of the autonomic manager.

As shown in Figure 3 the MAPE-K includes also a Knowledge function that is shared with the Monitor, Analyze, Plan and Execute functions in the loop.

In regards of the design of a self-adaptive system Salehie et al. [35] proposed the following questions to obtain their essential requirements:

- Where, addressing which layer/component is required to do the change.

- When, addressing at what time it is needed to apply a change.
- What, addressing what attributes or components of the system can and should be changed by means of adaptation, and what is required to be changed.
- Why, addressing the motivations of building a self-adaptive system.
- Who, addressing the level of automation and human involvement in the self-adaptive system.
- How, addressing how the adaptable systems can be changed and which adaptation actions can be appropriate for each situation.

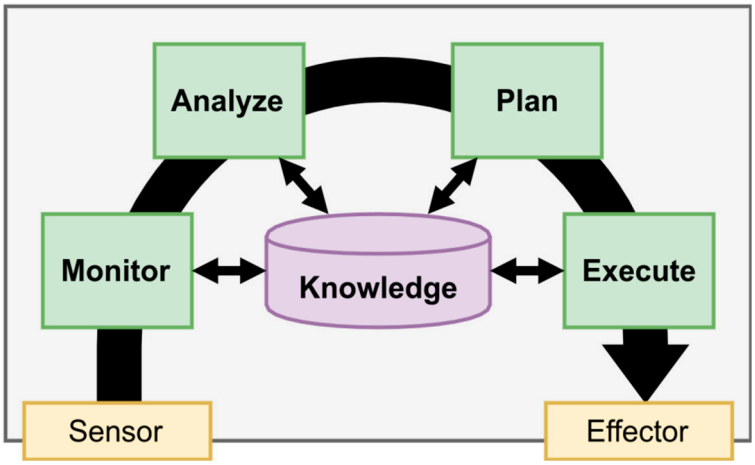

Figure 3. Functional description of a MAPE-K autonomic manager implementing the MAPE-K loop.

Concerning underwater cooperative robotics operations, the responses to when, what, why, and how can be answered as follows: whenever an unforeseen event happens preventing the achievement of a goal in a mission plan (when); whatever is possible, needed and has best chances to solve the situation (what); to still have chances of fulfilling the mission goals (why); and by any means available and suitable (how). The response to the "who" question, as we are talking about automated systems, is usually answered as the automated system. However, the response to the "where" question requires us to know a little more about how underwater interventions are typically managed. In traditional underwater interventions the mission plan was fully loaded into the AUVs to be launched, and once loaded, the AUVs were deployed on the sea surface where they remain waiting for the start command to start the mission [19,20]. If we just follow this approach, the response to the "where" question is clearly that the component responsible to do the required changes must be the AUV.

Although, in recent years there has been an improvement in the communications between the AUVs and the control stations where a global mission control can be running [21,41], enabling other possible responses to the "where" question. For instance, the self-adaptation could be considered:

- Only at the vehicle level.
- Only at the global mission control level.
- Both at the vehicle and the global mission control level, independently.
- Both at the vehicle and the global mission control level, following a decentralized control pattern.

The use of self-adaptation loops at different levels matches the MAPE-K proposal, where a hierarchy of autonomic managers is possible. Figure 4 illustrates an example of a two-level hierarchy. In the presented example there is one autonomic manager at the high level, and three autonomic managers at the low-level (keep in mind that there could be any number of levels and autonomic managers per level, as this is just an example). In this example each autonomic manager is able to perform self-adaptation using the MAPE-K loop individually. However, they could also be using a decentralized control pattern enabling the cooperation between managers at different levels [22]. A practical use case for this example could be a global mission manager using self-adaptation at

the high level, and a set of AUVs doing their own mission management with self-adaptation at the vehicle level.

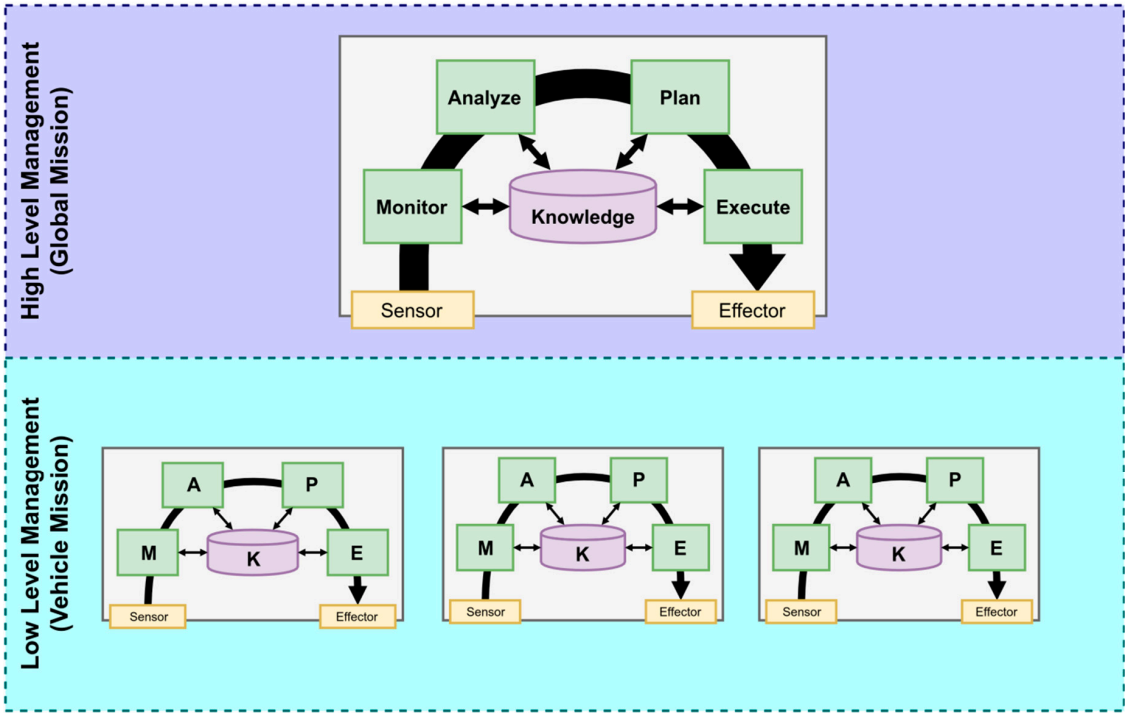

Figure 4. An example of a hierarchy of MAPE-K autonomic managers with a two-level hierarchy.

Iglesia et al. [42] proposed a set of formal templates for the design of self-adaptive systems, and Weyns et al. [22] proposed a graphical notation for patterns of decentralized control for self-adaptive systems, both using MAPE-K as the core base. The patterns introduced by Weyns et al. imply the interaction between the functional components of the MAPE-K loop from different autonomic managers in a given hierarchy. The knowledge function in these patterns is considered apart, as each autonomic manager can use its own internal knowledge, and the hierarchical distribution can also provide additional shared knowledge repositories within the different layers, or even across them.

The work of Weyns et al. also included the identification of five basic patterns:

- Coordinated control: where the decision of how to adapt is made cooperatively among all the autonomic managers.
- Information sharing: where each autonomic manager does the adaptation locally, but requires information from other autonomic managers to know how the locally performed adaptation will affect them.
- Slave/master: where the monitor and execution functions are performed at one level (slave), and the analyze and plan functions are performed at another level (master).
- Regional planning: where the hierarchical distribution is divided in autonomic regions. Each region consists of a number of autonomic managers with a shared plan function that provides the self-adaption planning functionalities for the region.

Hierarchical control, where there is a hierarchical distribution of the autonomic managers at different levels of abstraction, each one focused on different concerns.

For this survey we have categorized the use of self-adaptation in the explored architectures as illustrated in Figure 5. The first category is used for architectures that do not provide a description of the use of any self-adaptation model, and where we have not been able to identify one from the proposal. The second category is for architectures that only consider the use of self-adaptation models

at the vehicle mission management level, while the third category is for architectures that only consider the use of self-adaptation at the global mission management level. The fourth and fifth categories are for architectures that use self-adaptation models at both levels of management, independently or using a decentralized control pattern respectively. Table 3 summarizes the assessment for the self-adaptation feature as we have defined it.

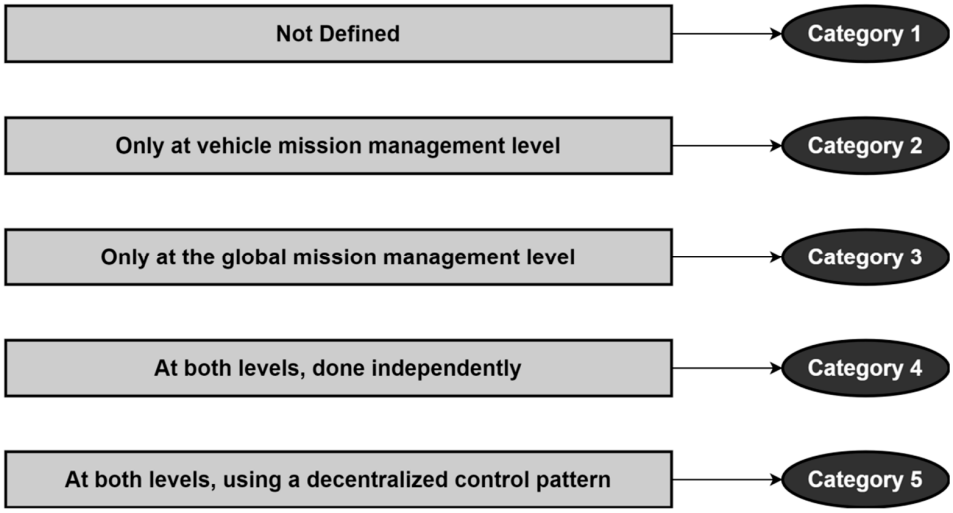

Figure 5. Diagram for the assignment of categories for the self-adaptation feature.

Table 3. Self-adaptation assessment.

Category	Description
1	Self-adaptation is not mentioned and has not been identified in the proposed architecture.
2	Self-adaptation is only considered at the vehicles level.
3	Self-adaptation is only considered at the global mission management level.
4	Self-adaptation is considered both at the vehicles and the mission management levels, but done independently.
5	Self-adaptation is considered both at the vehicles and the mission management levels, in a cooperative way using a decentralized control pattern.

2.3. The Use of Heterogenous Vehicles: Virtualization

Cooperative underwater operations using multiple AUVs require their mission management capabilities to be taken into account by mission management architectures. A simplistic approach could be that participation in cooperative missions be limited to AUVs with homogeneous capabilities tightly represented within the mission management architecture. However, the increasing interest in the use of AUVs for underwater operations implies the also increasing availability of multiple AUVs with different capabilities. Mission management architectures that want to consider the possibility of using AUVs with heterogeneous capabilities may also rely on using a tightly representation of the AUVs mission management capabilities by restricting the heterogeneity. Another approach is to use a well-defined soft representation of a common mission management capability defined for the architecture, enabling the possibility of using legacy and new AUVs in cooperative missions.

A virtual representation of a physical entity can be described as an abstraction mechanism that masks the specifics of such a physical entity, providing a common interface to access to its resources. The use of virtualization for abstracting the capabilities of heterogeneous devices has already been used in other domains. For instance, in [43], Lucas Martínez et al. described a model for the virtualization of event sources inspired by the IoT-A domain model [44]. This model, as partially illustrated in Figure 6, provides the access to the capabilities of a physical entity by the use of the services and resources exposed through its virtual representation as a virtual entity. With respect to virtualization, this domain model represents the most usual way for virtualizing physical entities, like AUVs.

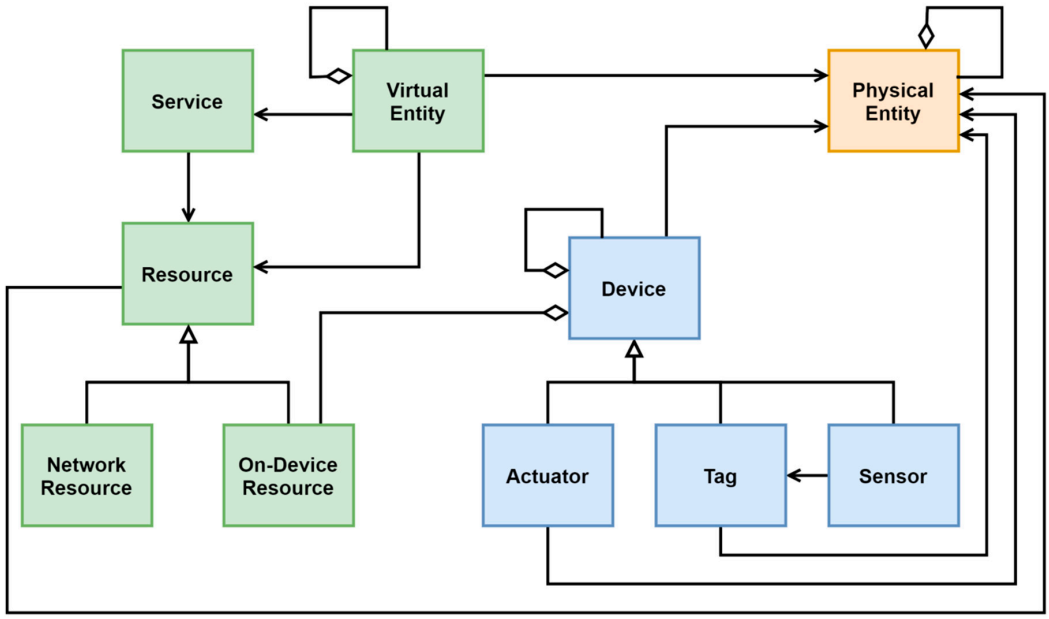

Figure 6. IoT-A domain model restricted to the virtual representation of a physical entity [44].

From the global mission management point of view, the use of AUV virtualization may enable the chance to integrate both legacy and novel AUVs in a cooperative mission, dealing with their heterogeneity regarding their capabilities by using their virtual representations instead of a direct and tightly coupled one.

The schema for the assignment of the categories that we have defined for the virtualization feature is shown in Figure 7. The first category is for architectures that do not mention the use of virtualization or abstraction methods in regards of mission management. The second category is for architectures that do not use virtualization or abstraction at all, relying on a tightly coupled integration of the vehicles within the mission. Architectures in this category can use both homogeneous and heterogeneous vehicles, requiring for the specific integration of the capabilities of each vehicle. The third category is for architectures that use virtualization just for homogeneous vehicles using a tightly coupled virtual representation. The fourth category is for architectures that use virtualization for heterogeneous vehicles with a tightly coupled virtual representation. And finally, the fifth category is for architectures that use virtualization for heterogeneous vehicles with a loosely coupled virtual representation. It is important to notice that architectures within category 2 are similar to architectures in categories 3 and 4, with the difference that architectures in category 2 will not use any virtual representation of the AUVs, while architectures in categories 3 and 4 will use a virtual representation, although being tightly coupled to the capabilities of the AUVs. Table 4 summarizes the assessment of the virtualization feature categories as we have defined it.

Table 4. Virtualization assessment.

Category	Description
1	The architecture does not mention vehicle virtualization or vehicle abstraction.
2	The architecture does not use vehicle virtualization, and either every vehicle is considered to have the same capabilities, or they have capabilities that are tightly coupled into the architecture.
3	The architecture uses vehicle virtualization, but every vehicle is expected to have the same capabilities.
4	The architecture uses vehicle virtualization for a set of heterogeneous vehicles with different capabilities, but using a tightly coupled virtual representation.
5	The architecture uses vehicle virtualization for a set of heterogeneous vehicles with different capabilities enabling the possibility of including new vehicles in a mission.

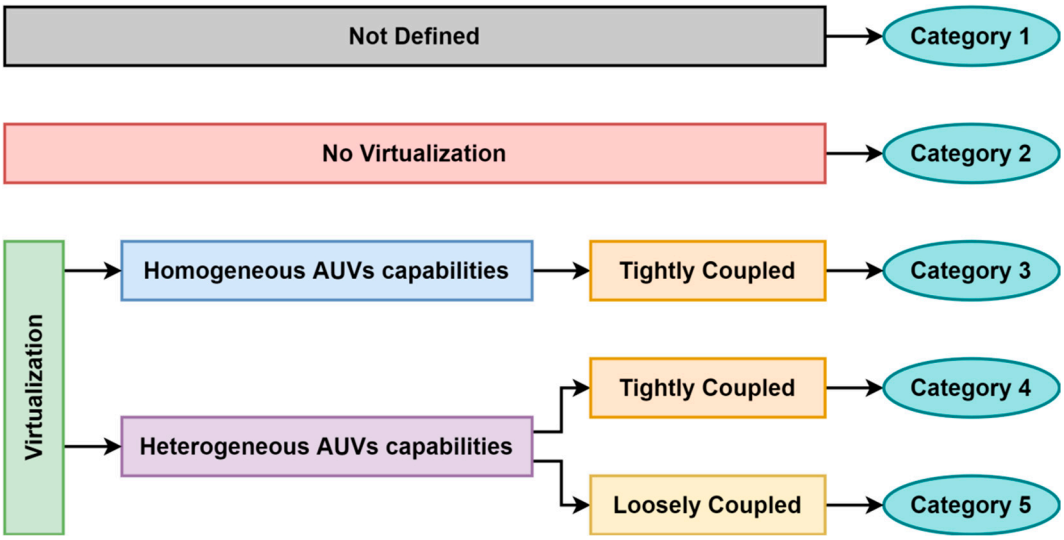

Figure 7. Diagram for the assignment of categories for the virtualization feature.

2.4. Mission Plan Dispatching and Execution

The last feature that we have considered for this survey is related to the way the mission plan is dispatched and executed to the AUVs from the global mission management point of view. The classical procedure implied to pre-load a pre-defined mission plan in the AUVs before launching them [20]. However, the improvement in underwater communications in recent years has the potential to consider other ways of dispatching the mission plan to the AUVs participating in a mission.

For this feature, we have identified five possible categories as illustrated in Figure 8. The first category, as with the rest of the features considered in this survey, is for architectures that do not give a description of how the mission plan is delivered. The second category is for architectures that use the preloading of the full mission plan into the AUVs. The third and fourth categories are for architectures that include the dispatching of the mission plan from the global mission management, either done task-by-task (for the third category) or by sending the full plan (for the fourth category). The fifth and last category is for architectures that using an online dispatching from the global mission management (categories 3 and 4), also take into consideration the possibility of live updating the mission plan. Table 5 summarizes the mission plan dispatching and execution assessment.

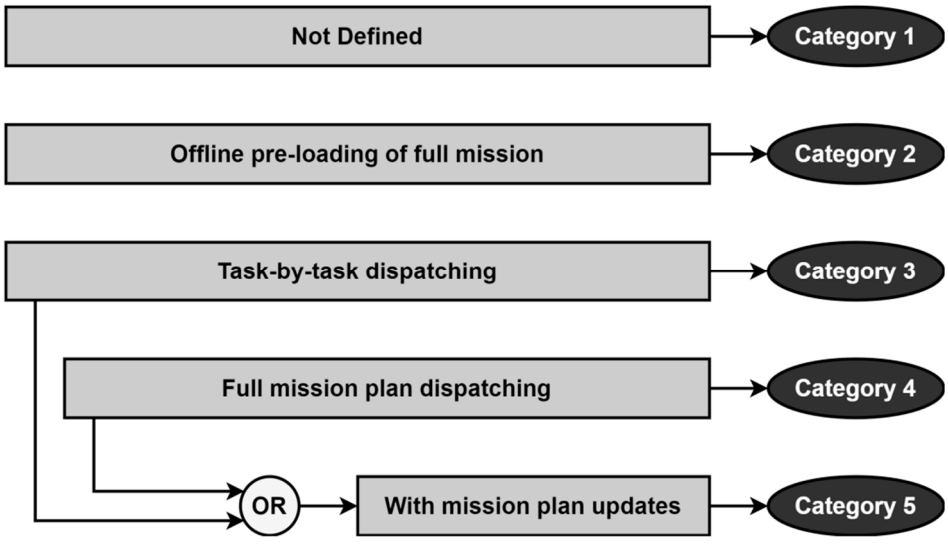

Figure 8. Diagram for the assignment of categories for the mission plan dispatching feature.

Table 5. Mission plan dispatching and execution assessment.

Category	Description
1	The proposal does not mention how the mission plan or the mission plan tasks are dispatched to the vehicles and/or executed.
2	The mission is fully loaded into the vehicles before launch.
3	The mission plan is dispatched to the vehicles task by task, thus the global mission manager being responsible for active supervising of the task execution before dispatching the next task to a vehicle.
4	The mission plan is dispatched completely to the vehicles, thus the supervision of the execution of each task in the mission plan is done at the vehicle level, and without regards of if the mission manager is also actively or passively monitoring the execution progress.
5	Same as 3 or 4, but also including the possibility of updating the mission plan in real time from the global mission manager.

3. Description of the Selected Proposals

A list of well-selected mission planning and management architectures is provided in this section with brief introductions and comprehensive analyses based on the set of technical features that were introduced in the previous section. This list presents a holistic view on the status of mission planning and management architectures. Some of the systems included in the study were not initially conceived for use with underwater environments. However, all these systems, by focusing just on mission management, can be easily adapted to different environments. For instance, the teleo-reactive executive architecture (T-REX) has been successfully implemented for underwater robotics, but also for service robotics [45]. Even a generic executive reactor for any robotic platform [46] has been developed for the T-REX architecture. The ROSPlan framework, developed for the robotic operating system (ROS) and designed for any robotics, was created as part of the PANDORA project for its use with AUVs.

3.1. Teleo-Reactive Executive (T-REX)

The T-REX [47,48] was developed by the Autonomy Group at Monterey Bay Aquarium Research Institute (MBARI), and has been validated in several real case scenarios. For instance, it has been successfully used in some of the experiments of the Controlled, Agile, and Novel Ocean Network (CANON) initiative at MBARI [49,50], as well as other marine and service robotics experiments [45] and simulations [51]. It also had an executive implementation as a node part of ROS up until the C Turtle release [52].

The T-REX architecture was conceived as a goal-oriented system, where the typical mission plan, instead of being a sequence of detailed tasks, it is specified by a sequence of high-level goals and constraints. The reasoning for this change is to deal with the increasing mission uncertainty in underwater environments.

The essentials of T-REX are inspired by the sense-plan-act [38] adaptive model, using a number of concurrent control loops that provide the response to exogenous events and enable the distribution of the deliberation processes among the components of a T-REX agent. An example of a T-REX agent is described in [47], and illustrated in Figure 9. In this case the T-REX agent has four components, formerly named teleo-reactors, or just simply reactors: a mission manager, a science operator, a navigator, and an executive. Each of the reactors in the figure provide the goals for other reactors (full arrows), and monitors their current state through the provided observations (dotted arrows).

Each reactor in T-REX is capable of planning the specific tasks to achieve the goals provided, and to dispatch them to other reactors as their own goals. McGann et al. provided in [53] the example of an integrated system design with three reactors with their internal main components that is shown in Figure 10. In this example, we have three reactors in a hierarchical distribution: a vehicle control subsystem (VCS) at bottom, a navigator at middle, and a mission manager at top. As the figure illustrates, each reactor receives goals from the reactor in its upper layer reactor, and sends their own generated goals to its lower layer reactor. At the same time, each reactor receives the observations

from its lower layer reactor, and generates observations to its upper layer reactor. Each agent has a local database for storing the received observations, and as shown in the figure, to store also plans, models and algorithms. The reactors in the example include a planner, a synchronizer and a dispatcher. The planner uses automated planning techniques for adapting incomplete plans from the database to the current problem. The synchronizer is used to ensure plan consistency and completeness. Finally, the dispatcher is a simple algorithm used to publish goals to other reactors.

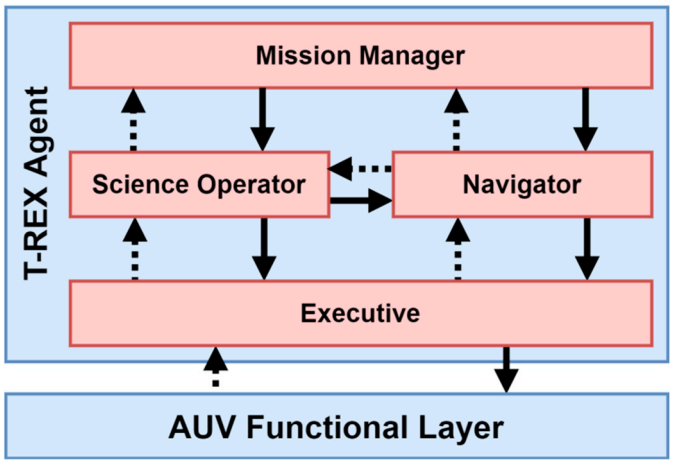

Figure 9. An example T-REX agent composed of four reactors.

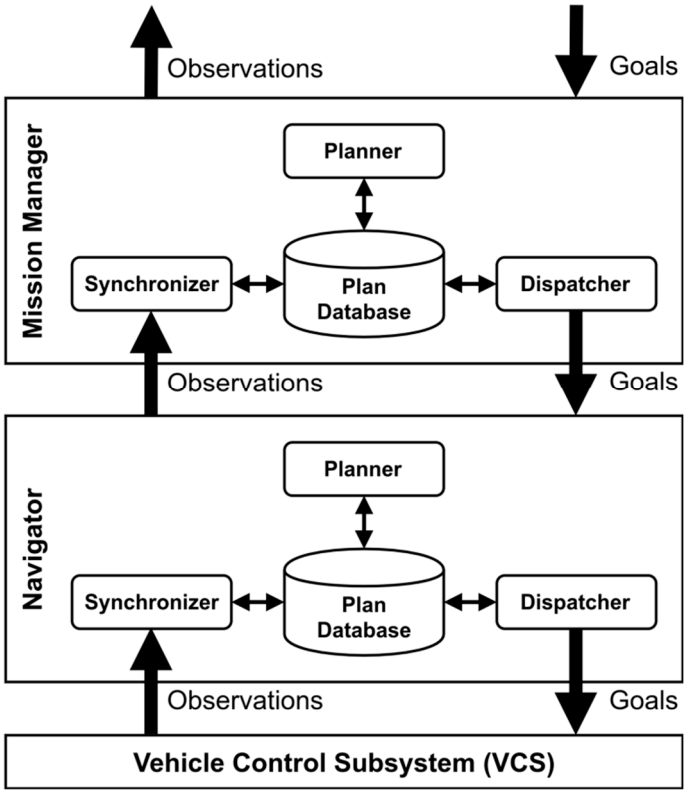

Figure 10. An example of the design of a T-REX integrated system.

The system proposed as an example in Figure 10 can be used as follows: the mission manager reactor receives the goals to fulfill a mission, for instance, a mission to survey an area. With that information and the data available in its local database, the mission manager generates a plan with the navigation steps that is dispatched as goals to the navigator. In turn, the navigator takes these goals, and using the data available in its local database generates a plan for the VCS consisting on the specific

commands the vehicle has to execute. As before, the plan generated by the navigator is dispatched to the VCS as the goals to the VCS.

The dispatching algorithm is ruled by time windows. Each goal has an assigned starting time to be dispatched to the corresponding reactor. The time is calculated taking into account the latency of the reactor, its planning horizon and the execution frontier for the current task in execution, if any.

As for the features considered in this survey, in regards to the plan specification feature the T-REX proposal does not describe how the plan is specified at any level. Therefore, we can only assume that it is done using an ad-hoc format. This corresponds to a category 2 for this feature. Besides, the use of multiple reactors allows for a multi-agent structure with a hierarchical mission plan, and the description of the dispatching algorithm expresses that each task has a temporal constraint. In consequence this feature has also the "MHT" modifiers, resulting in a final assessment of a category 2MHT.

In regards to the adaptation feature, the literature about the T-REX architecture describes explicitly that it is based on the "sense-plan-act" self-adaptive model [53]. Specifically, the architecture uses a partitioned control where each reactor has its own internal self-adaptive loop working in a coordinated pattern with other reactors. In consequence we assign to T-REX a category 5 in relation to self-adaptation.

With respect to the virtualization feature, the T-REX literature mentions that a T-REX agent uses a single model for control at various levels of abstraction [48]. However, no specifics are given about if there is any use of virtualization for the management capabilities of the AUVs participating in a mission, and as a result, we concede a category 1 in regards of the virtualization feature.

Finally, the dispatching of tasks in T-REX is described in [48]. As we have described previously, the dispatching of tasks in T-REX is done using time windows, with a new task being dispatched when the time window starts. Thus, we have assigned a category 3 to the dispatching feature.

A summary of the categorization for the measured features is included in Table 6.

Table 6. T-REX assessment.

Feature	Category	Explanation
Plan Specification	2MHT	The reviewed literature for T-REX does not provide any insights of how the plan is specified, thus it seems reasonable to assume that the plan specification is done using an ad-hoc format. The use of different reactors as agents capable of receiving goals and generating plans seems compatible to a multi-agent approach. It also means that the mission plan, seen from the top, can be considered as a hierarchical plan, since a plan from a reactor can be seen as the lower-level plan for the goals provided by another reactor.
Adaptation	5	As T-REX is based in sense-plan-act, it uses self-adaptation by default. The partitioning of the control loops in the executive is done using a coordinated pattern.
Virtualization	1	There is no mention to virtualization or abstraction.
Dispatching	3	The dispatcher sends the goals to the corresponding reactors one by one. It is important to notice that in this case the description of the algorithm implies that there is no real supervision made from the upper reactor, and that the dispatching of a new goal is just ruled by its planned starting time.

3.2. RAUVI Architecture and the Mission Control System (MCS)

The Reconfigurable AUV for Intervention missions (RAUVI) [54] was a research project aimed to the development of a new reconfigurable AUV capable of performing underwater operations using a robotic arm [55]. A distributed architecture was proposed [56] and validated through simulations and experiments in controlled scenarios [57]. The proposed architecture was designed to combine two pre-existing architectures with a mission control system (MCS), as shown in Figure 11. These two architectures are the manipulation and the vehicle architectures respectively.

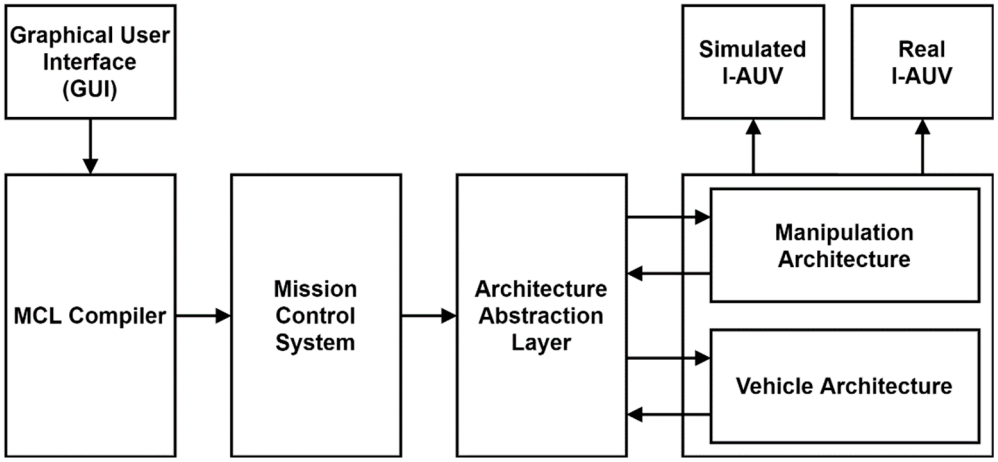

Figure 11. An overview of the RAUVI software architecture.

The manipulation architecture is aimed to control the robotic arm in the AUV, and consists mainly of two modules: perception and action. On the other hand, the vehicle architecture is in charge of ensuring the functionality of the AUV, and includes three main modules: robot interface, perception and control. The third piece in the proposed distributed architecture is de aforementioned MCS, in charge of defining the task execution flow to fulfill a mission. A detailed functional diagram of the proposed architecture is shown in Figure 12. Four main functional blocks are shown in the diagram. At top sits the MCS, with the Architecture Abstraction Layer. At the bottom, we can find the manipulator architecture on the left, and the vehicle architecture on the right. Finally, in the middle, there is a centralized blackboard used as a communication interface between the manipulator and the vehicle architectures.

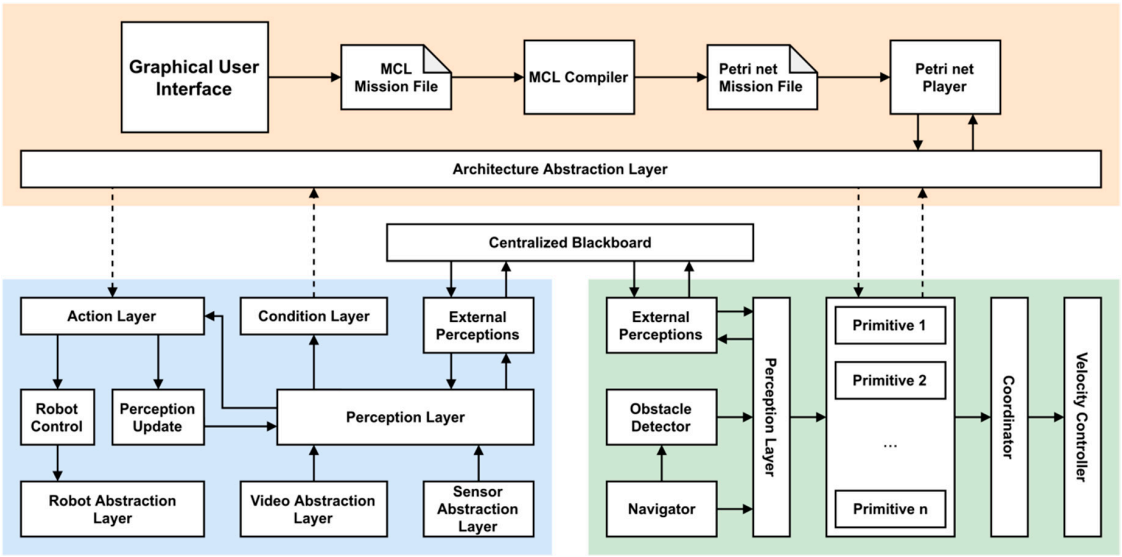

Figure 12. RAUVI proposed architecture.

The mission plan in the RAUVI architecture must be provided by the mission programmer using a graphical user interface to generate a mission file expressed in XML using a mission control language (MCL) [58,59]. The generated file is then parsed by a MCL compiler that generates a Petri net mission file. The MCS receives this Petri net file, and executes it through the abstraction layer, where every task has the exact same interface, and is dispatched to either the manipulator or the vehicle architecture.

The manipulator architecture receives the tasks through the action layer. This module can receive two types of actions: control and perceptual. The control actions are fed to the robot abstraction layer,

where they are executed by the manipulator robot itself. On the other hand, the perceptual actions are feed to the perception layer, updating the robot internal perception. The perception layer receives also information from other parts of the manipulator robot, as internal perceptions, and from the vehicle architecture through the centralized blackboard, as external perceptions. Finally, if some perceptions meet some conditions at the condition layer, an event is sent to the MCS through the abstraction layer.

The vehicle architecture receives the tasks from the MCS in a control module that matches them to the primitives that are sent to the coordinator that manages their execution using the vehicle velocity controller. Besides the tasks received from the MCS, the vehicle architecture has a perception module that receives internal perceptions from the vehicle navigator and obstacle detector, and external perceptions from the manipulator architecture through the centralized blackboard. These perceptions can be used by the primitives in the control module.

Regarding the features being considered in this survey, in RAUVI, the mission file uses an ad hoc specific format, the MCL. This format follows a deterministic approach, without multi-agent considerations, hierarchical structure or temporal constraints. Consequently, we have assigned it a category 2 for the plan specification feature.

With respect to the adaptation feature, there is not mention at all to the use of any self-adaptive model in the architecture, and we have not been able to identify one. Therefore, we assign it a category 1 for the adaption feature.

In regards to the virtualization feature, the RAUVI architecture specifically mentions the use of an architecture abstraction layer [56] between the MCS and the vehicle/manipulator architectures. The purpose of this abstraction layer is to adapt actions and events to and from the corresponding instances of each architecture. This enables the virtualization of the capabilities of the different architectures being used, providing a common access to both the manipulator onboard a vehicle, and the vehicle itself. In consequence we have assigned it a category 5 for the virtualization feature.

Finally, for the dispatching feature, in the RAUVI architecture the mission, once compiled in the MCL, it is fully loaded into the vehicles before being launched. Therefore, the corresponding category for the dispatching feature is category 2.

A summary of the categorization for the measured features is included in Table 7.

Table 7. RAUVI architecture assessment.

Figure	Category	Explanation
Plan Specification	2	The RAUVI proposal includes the definition of a Mission Control Language to define the mission tasks of interest in a friendly high-level language. The mission plan follows a deterministic approach, with no multi-agent capabilities, no hierarchical decomposition, and no use of temporal constraints.
Adaptation	1	Self-adaptation is not mentioned in the description of the proposal.
Virtualization	5	According to the description provided in [56] the architecture abstraction layer is capable of adapting the actions and events to the corresponding instances of the target architectures, allowing the MCS to be architecture-independent.
Dispatching	2	The mission is fully loaded into the vehicle before launch [56].

3.3. Intelligent Control Architecture (ICA)

The TRIDENT project [60] explored and proposed a new methodology to provide multipurpose dexterous manipulation capabilities for intervention operations in unknown, unstructured and underwater environments. As part of this project the intelligent control architecture (ICA) was proposed as a novel cognitive control architecture for AUVs [61,62], enabling the use of multiple vehicles in underwater intervention missions. The ICA was validated in the experiments carried out in the TRIDENT project, including simulations and real experiments in the sea [63].

An overview of the ICA is shown in Figure 13. The components are distributed as in the usual three-layer architecture used in robotics. At the deliberation layer, the system uses a hierarchical

mission plan specified by the main goal and its decomposition in subsequent sub-goals. Each goal is then planned, producing an agent plan that is a sequence of activities, or command messages. Once an agent plan is produced, its activities are matched to the available services that can be discovered from other agents, and that can be either basic or composed services following a service oriented architecture (SOA) approach [64].

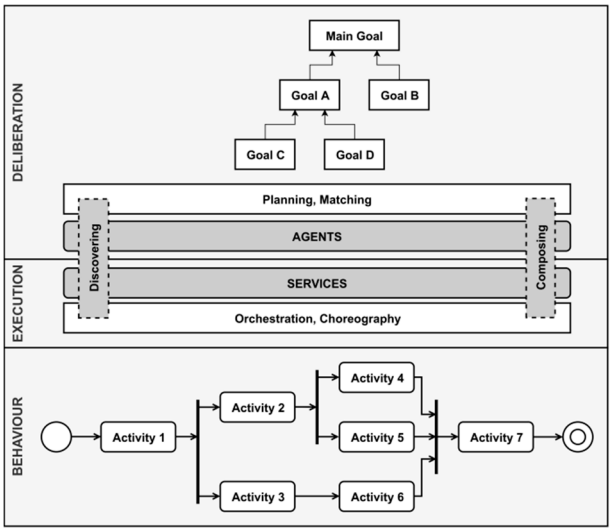

Figure 13. Architectural concepts of the ICA.

The ontology of the ICA, as shown in Figure 14, shows the relations among its core elements. The system, identified as an autonomous maritime robot (AMR) fulfills a mission that has one or more plans and one or more goals, and that is carried out by an agent. Each goal is achieved by a plan that involves one or more capabilities. The agents have plans that achieve goals and carry out them by means of activities that are carried out by services.

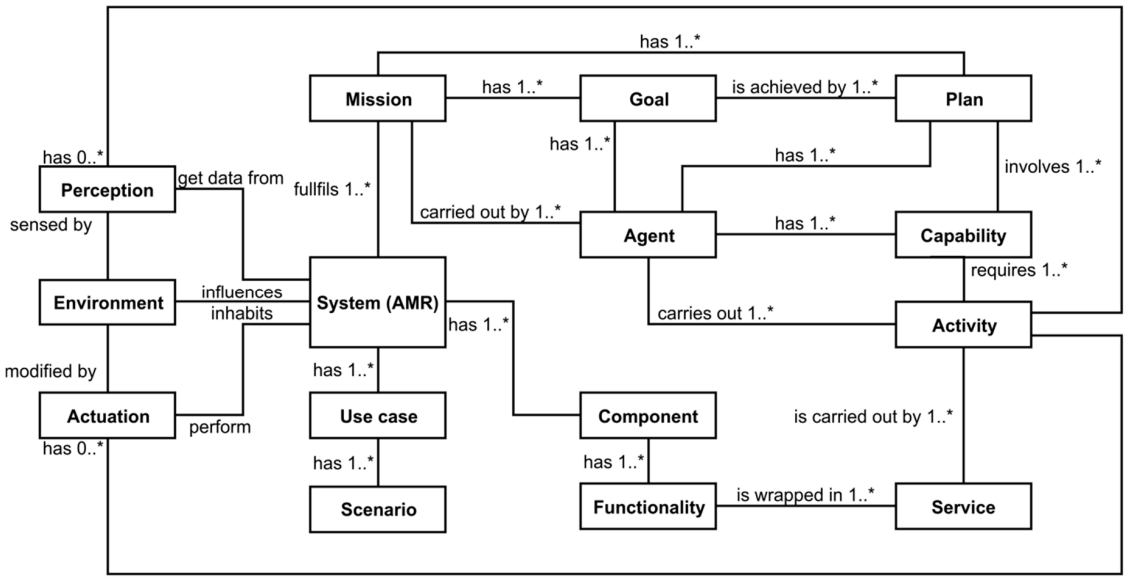

Figure 14. ICA ontology.

The internal structure of an ICA agent is shown in Figure 15. From the robotics model point of view, it is composed of five main blocks: the user interface, the deliberation unit, the execution unit, the behavior unit and the world model.

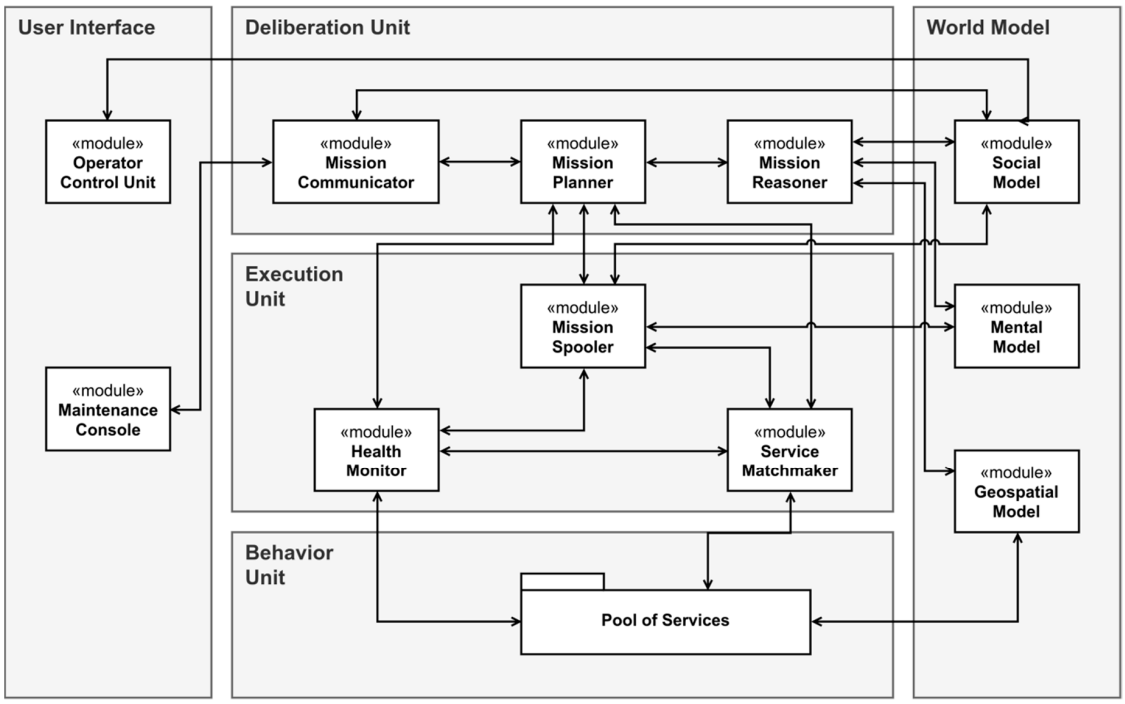

Figure 15. Internal structure of an ICA agent.

The user interface is the block responsible to provide the end user with an interface to manage the mission and check its progress.

The deliberation unit represents the deliberation layer of a three-layer robotics architecture, and has three components related to the generation of a mission plan: the mission communicator, used to communicate with the human operator through the social model; the mission planner, used to generate the mission plan; and the mission reasoner, used to decide what to do next according to the data perceived from the sensors.

The execution unit represents the execution layer of three-layer robotics architecture, and is responsible for the execution of the mission plan through the mission spooler.

The behavior unit represents the behavior layer of three-layer robotics architecture, and is responsible of executing the commands dispatched by the mission spooler according to the services available in the agent.

Finally, the world model is a repository composed of three models: the social model that describes the context for the agent, the mental model, that describes what the agent knows about itself, and the geospatial model, that contains the environmental data gathered from the sensors.

ICA agents also use a situation awareness [65] approach that uses a decision making cycle defined as an OODA loop [36]. In this OODA loop, the observation and orientation stages are matched by the belief blocks at the world model, provided by the social and geospatial models, and the decision and action stages are matched by the intention block, that is, the mission spooler module.

Regarding the assessment of the features considered in this survey, with respect to the plan specification feature, the ICA proposal explicitly states that it "moves away from fixed mission plans". It does that by including a mission reasoner able to adapt a mission plan based on the current situation of a mission. However, a mission plan is still generated by the mission planner from a hierarchical set of goals, although no description is given in the ICA literature about how this mission plan is specified. Additionally, the ICA architecture provides the mechanisms for using multiple agents from the operator point of view, assigning specific agent plans to each agent involved in a mission. As a result, we consider ICA belongs to a category 2MH for the plan specification feature.

Concerning the adaptation feature, the ICA proposal explicitly mentions the use of a OODA loop for the decision making cycle embedded in each agent. Therefore, we assign ICA a category 2 for the adaption feature.

With respect to the virtualization, or any other abstraction mechanism, the ICA literature does not mention any approach, and none can be identified from the proposal, resulting in a category 1 assignment for this feature.

Finally, although the ICA specifically mentions the use of a mission spooler component that dispatches the mission plan to the service matchmaker component, there is no description on how this dispatching is done. In consequence, for the dispatching feature, we assign an ICA with a category 1.

A summary of the categorization for the measured features is included in Table 8.

Table 8. ICA assessment.

Feature	Category	Explanation
Plan Specification	2MH	From the architectural overview it is clear that the mission specification is provided as a hierarchical set of goals used to generate agent plans that are in essence a sequence of commands without following or proposing a formal description, and thus using an ad-hoc one. As the agent plans can be assigned to different agents, we consider that ICA provides a multi-agent view.
Adaptation	2	ICA includes an OODA loop approach for the situation awareness performed by an AUV agent.
Virtualization	1	There is no mention to virtualization or abstraction of the agents in the ICA proposal.
Dispatching	1	Although the internal structure of an ICA agent includes a component named "Mission Spooler" that feeds the "Service Matchmaker" with the agent plans, there is no mention to how this agent plans are really dispatched to the AUV agents.

3.4. SWARMs Architecture

The primary goal of the SWARMs project [66] is to expand the use of underwater and surface vehicles (AUVs, ROVs, USVs) to facilitate the conception, planning and execution of maritime and offshore operations and missions. As part of this project a semantic middleware architecture has been defined, implemented [67], and validated with simulations and real scenarios [68–70]. Figure 16 shows a detailed functional description of the components that constitute the aforementioned architecture.

The SWARMs middleware components are distributed in four main groups:

- Data management: The components in this group are related to the management of the information handled in the middleware. They are:
    - Two repositories, one for historical data and the other for the SWARMs ontology
    - A data access manager for managing the access to the repositories
    - A publish/subscription manager for providing access to the Data Distribution Service (DDS) based communications used in the project.
- High level services: The components in this group are essentially those components accessible from the application layer, and specifically within the SWARMs project, from the application known as the mission management tool (MMT). They are:
    - The missions and tasks register and reporter (MTRR), that receives the mission plan from the mission management tool and is in charge of the mission management from within the middleware.
    - The vehicle and service register, that provides information about the available vehicles and services.

- The semantic query, which processes the semantic queries made by the MMT.
- The rules and policies creator, that allows the insertion of rules and policies by users.

- Low level services: The components in this group represent the interface between the middleware and the vehicles, and they are:
  - The tasks reporter, which tracks and reports the status of the tasks in a mission.
  - The event reporter, which collects the events that occur during a mission.
  - The environment reporter, that tracks and periodically reports collected environmental data relevant for the mission.
- Cross layer services: The components in this group provide services for all the other components in the middleware, and they are:
  - The semantic reasoner, responsible for processing higher level inferences and context awareness from the context information available in the SWARMs ontology.
  - The data pre-processor, that validates the data gathered from the vehicles.
  - The security component, responsible for providing a number of security schemes, like data integrity, authentication, authorization and access identity management.
  - The quality of service component, responsible for enabling the quality of service policies specified by the MMT.

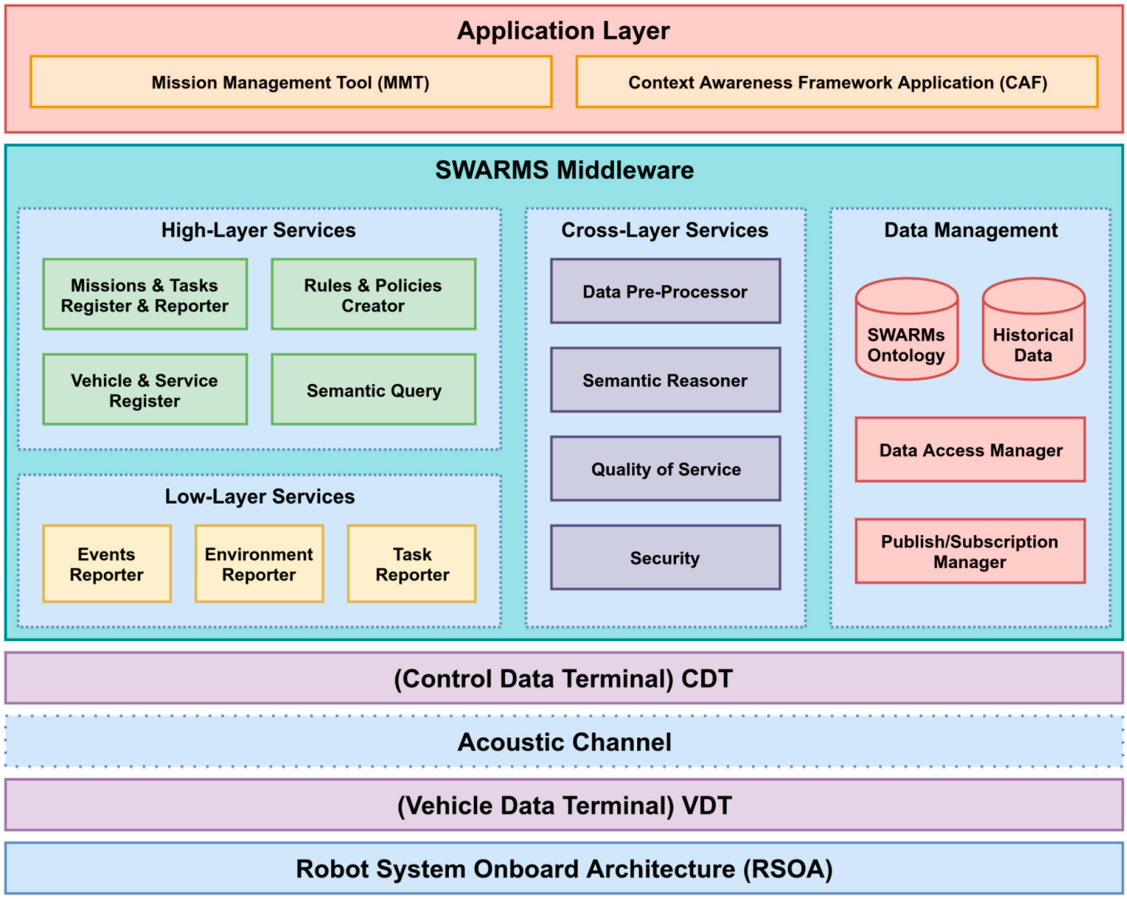

Figure 16. SWARMs middleware architecture.

Regarding the mission planning and management, the mission plan in SWARMs was generated by a deterministic automated planning algorithm executed from the MMT. The mission plan, specified

in an ad-hoc format, consists of a sequence of actions, each one assigned to its corresponding vehicle, allowing for a multi-agent specification. Also, from the point of view of mission planning, the vehicles in SWARMs can be classified in two categories: those able to plan by themselves how to achieve a goal, and those that require to be provided with the specific primitive for the action. Therefore, the mission plan also includes a hierarchical decomposition of the plan, so the MTRR can decide to dispatch to the vehicles either a high-level mission plan or a low-level one depending on their capabilities.

The MTRR is in essence responsible for the mission management within the SWARMs middleware architecture, and it has been validated during the SWARMs final trials [71]. Figure 17 illustrates the relations between the MTRR and other SWARMs components.

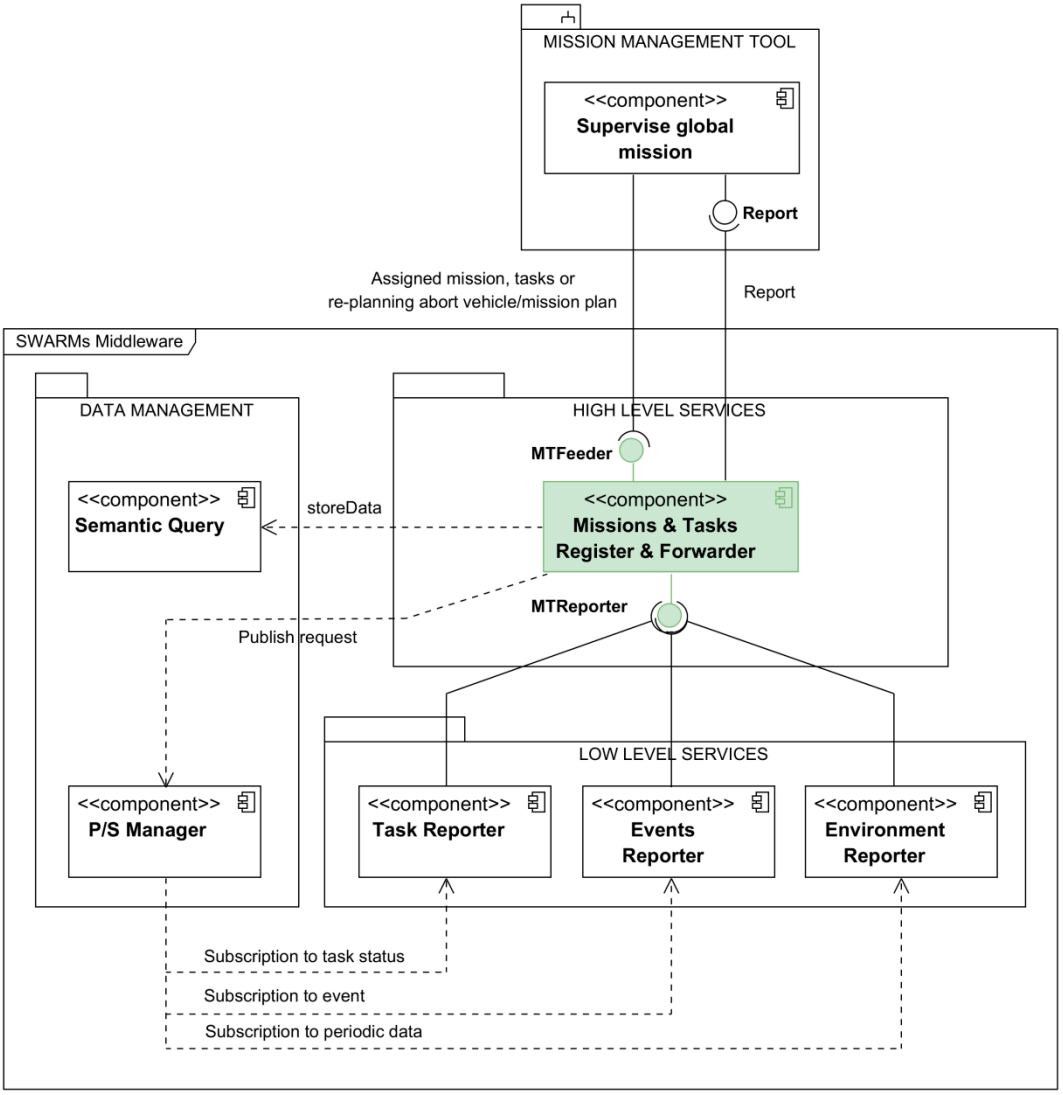

Figure 17. Relations between the mission manager (MTRR) and other SWARMs components.

As explained before, the MTRR receives the mission plan generated at the MMT. Using a tightly coupled virtualization, the MTRR parses the mission plan, and divides it into vehicle plans, using either the high-level actions or the low-level ones depending on the capabilities of the assigned vehicles.

The MTRR is capable of dispatching the full plan to the vehicles, or doing it task by task. Indeed, within the SWARMs project, the dispatching was done task by task due to the restrictions of the on-board system loaded into the vehicles. To avoid problems with the latency between the ending of an action, and the dispatching of the next one in the vehicle plan, a pre-buffering technique was used, where the next action to the current one was also dispatched to the vehicles.

The MTRR was also responsible for handling the status, events, and environmental reports received from the vehicles. Whenever an event prevented the execution of part of a plan, the MTRR notified the MMT so a human operator could decide whether to abort the mission or re-plan it with the new information sent by the vehicles.

Regarding the assessment of the features considered in this survey, the mission plan specification in the SWARMs architecture is done using an ad-hoc format that includes capabilities for multi-agent plans and uses a hierarchical task representation. It also includes temporal estimations for the tasks durations, but they are not provided as temporal constraints. Consequently, we have assigned it a category 2MH for the plan specification feature.

With respect to the use of adaptation, each vehicle integrated in SWARMs is capable of performing certain degree of adaptation during the execution of a mission plan. At the same time the mission manager can abort a vehicle plan at any time, and re-plan just its part of the mission plan, following a decentralized hierarchical control pattern for vehicle plan adaptation. Therefore, we have assigned it a category 4 for the adaptation feature.

The MTRR uses internally a virtualization of the vehicle capabilities with regards to their onboard planning capabilities. This use of the virtualization paradigm is done using a tightly coupled design, where the specific capabilities of the vehicles are modeled into their internal virtual representation. For that reason, we have assigned it a category 4 for the virtualization feature.

Finally, the MTRR can dispatch the mission plan either by sending it a task-by-task to the vehicles, or sending them the full mission. Hence, we have assigned it a category 4 for the dispatching feature.

A summary of the categorization for the measured features is included in Table 9.

Table 9. SWARMs assessment.

Feature	Category	Explanation
Plan Specification	2MH	The plan specification within the SWARMs project is done using an ad-hoc format that includes multi-agent and hierarchical approaches.
Adaptation	4	Each vehicle in SWARMs is capable of performing some adaptation while executing a mission in order to achieve the assigned tasks. The mission manager can abort a vehicle plan at any given time while keeping the rest of the mission plan in execution and re-plan just for that vehicle.
Virtualization	4	The MTRR in SWARMs uses a basic virtualization approach tightly coupled to the heterogeneous capabilities of the available vehicles.
Dispatching	4	The MTRR is able to dispatch the actions in the plan to the vehicles following a task by task approach or sending them the full mission.

3.5. Continuous Planning and Execution Framework (CPEF)

The continuous planning and execution framework (CPEF) [72,73] is a framework designed for the plan generation, execution, monitoring and repairing aimed to solve complex tasks in unpredictable and dynamic environments. It was developed as part of the Defense Advanced Research Projects Agency (DARPA) Joint Forces Air Component Commander (JFACC) program, and validated through several demonstrations [74–77]. Although it was conceived for its use with UAVs, the proposed design for the framework could be of use for any other application domain.

Figure 18 shows the functional overview of the six components that make up CPEF. The Plan Manager is the core component, responsible for the control of the system. It is in charge of controlling the generation of plans, supervise their execution, provide knowledge for both plans and plan executions, respond to unexpected events and control the adaptation of the plan in the event of plan failures that require plan updates. The supervision of the execution of the plans from the plan manager uses a flow model in which the plan manager waits for the outcome of individual action.

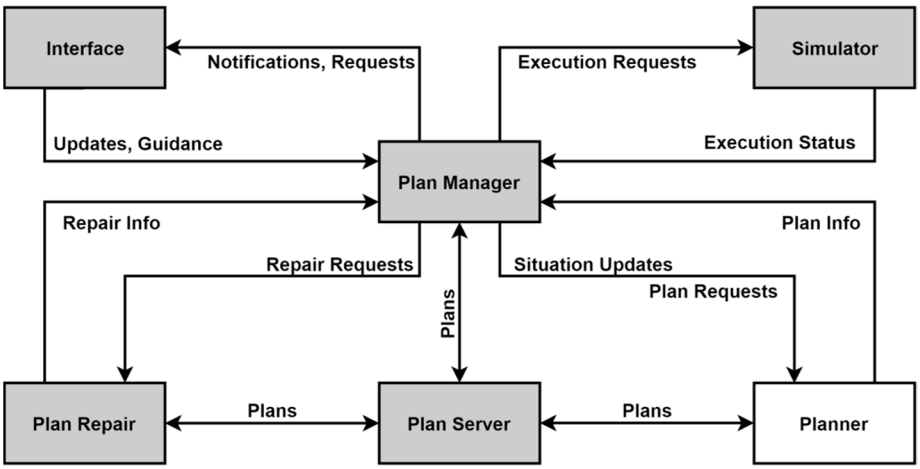

Figure 18. Functional overview of CPEF.

Besides the tracking of the execution through the flow control model used in the plan manager, CPEF resorts to the use of monitors, defined as *event-response* rules. These monitors can trigger the execution of a predefined response whenever they detect a specific event. There are three main categories of monitors defined in CPEF:

- Failure monitors: They provide the responses to the failures that can happen while a plan is being executed.
- Knowledge monitors: They provide the responses for decision-making regarding the changes in the world-state.
- Assumption monitors: They provide the responses for changes in the situation of the plan execution that violate the assumptions on which the plan relies.

Regarding adaption, CPEF, using the monitors already described, defines generalized failure models for assessing the detected fails, and deciding if they are relevant enough to require for the repairing of the plan in execution.

As for the features considered in this survey, the literature about CPEF does not provide a description about the mission plan specification, and therefore the category for the plan specification is category 1.

In regards to the adaptation feature, the CPEF proposal explicitly mentions that the plan manager is able to adapt the plan to the events that trigger the plan repair mechanism. This adaptation is made at the agents implementing the CPEF architecture, without specific mention to the use of a possible decentralized control pattern. Therefore, the category we have assigned to CPEF for the adaptation feature is category 3.

As with the plan specification feature, CPEF does not describe any virtualization or abstraction mechanism, subsequently being categorized as category 1 for the virtualization feature.

Finally, the reviewed material about CPEF does not include how the mission plan is dispatched to the agents. However, it is explicitly explained that the plan manager uses a flow model waiting for the status reports of each task in the mission plan before dispatching the next one. Thus, we have assigned CPEF with a category 3 for the dispatching feature.

A summary of the categorization for the measured features is included in Table 10.

Table 10. CPEF assessment.

Feature	Category	Explanation
Plan Specification	1	The description of CPEF does not include a reference of how the plan has to be specified. Further, the proposal of CPEF predates the specification of PDDL and its updates and extensions, as well as RDDL.
Adaptation	3	The plan manager in CPEF is able to adapt the plan to the events if they trigger the need for repairing the plan. However there is no mention about the use of adaptation in the agents executing the plan.
Virtualization	1	There is no mention in the proposal about the use of virtualization or abstraction of the agents executing the plan.
Dispatching	3	Although there is no specific mention on how the plan is dispatched to the agents, CPEF proposal mentions that the plan manager uses a flow model in which it waits for the status reports of each individual action in the plan.

3.6. Technologies for Reliable Autonomous Control (TRAC)

The technologies for reliable autonomous control (TRAC) of UAVs was developed as part of a DARPA effort under the Software Enabled Control (SEC) initiative [78]. It was created to enable greater mission capability, higher reliability and safety, and greater adaptability to vehicle and mission environment variability, and was in essence the same as the reliable autonomous control technologies (ReACT) for AUVs created by the same authors as a NASA effort for the revolutionary concepts (RevCon) initiative [79]. TRAC/ReACT were developed and demonstrated through simulations [80] and real scenarios demonstrations [79].

As with CPEF, TRAC was not intended for being used in underwater missions. However, some of the concepts introduced by TRAC could be of interest when designing a mission planning and management architecture for underwater operations. Figure 19 illustrates the functional architecture of TRAC.

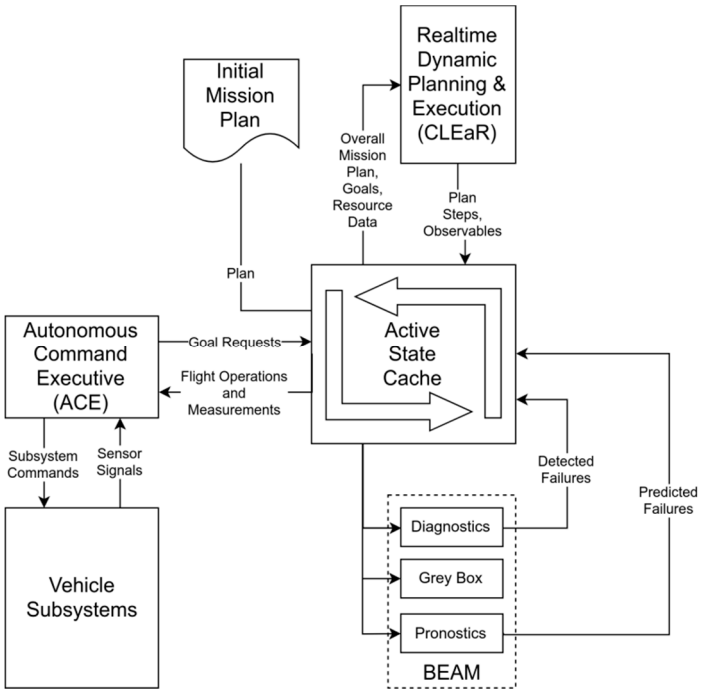

Figure 19. The TRAC functional architecture.

The components of the TRAC architecture include:

- A beacon-based exception analysis for maintenance (BEAM) that monitors the data bus and detects anomalies, and isolates and characterizes failures.
- A spacecraft health inference engine (SHINE), used in conjunction with BEAM for diagnosis and prognosis.
- A closed loop execution and recovery (CLEaR) that provides high-level mission plan management and generates the sequence of commands to the UAV to execute the mission plan.
- An autonomous command executive (ACE) that supervises the execution of the mission plan created by the CLEaR component.

These four components are linked together through the active state cache, which is a data exchange repository for keeping and synchronizing all global information regarding the mission.

The CLEaR component later evolved as an independent framework [81], is just in charge of generating the mission plan and coordinate a goal-driven and event-driven behavior. This is partially done by sharing plan information and continuous status updates both with the planner and the executive. The decision about if a plan conflict must be handled by the planner or by the executive is provided by some heuristics that depend on the mission and the domain.

The other relevant component from the mission management point of view is the ACE [80]. Figure 20 shows the functions performed by ACE, and how they are related to each other.

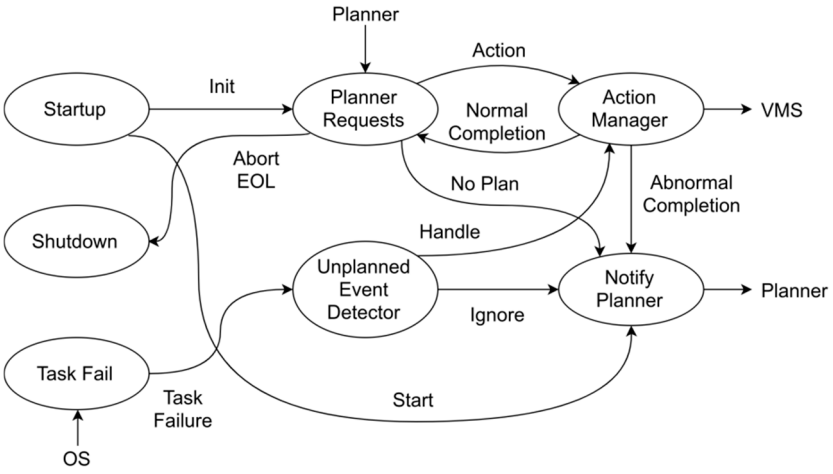

Figure 20. The ACE collaboration diagram.

As illustrated, ACE includes an action manager that receives the actions from the planner requests and the unplanned events to be handled. The action manager is responsible for the execution of the actions in the vehicles through the VMS, and for the decision making about the unexpected events. If an event prevents the normal completion of an action, the action manager will notify the planner, so a new plan can be generated if required to fulfill the mission.

Figure 21 shows the components of the action manager within ACE. Each component is responsible for sampling the current state at the assigned step and calculating if a reaction is needed taking into account the operational limits.

The TRAC description does not mention how the mission plan is specified, and therefore we have assigned it a category 1 for the plan specification feature.

Regarding the adaptation feature, the TRAC proposal does also not mention the use of any self-adaptive model. However, the proposed architecture is continuously evaluating the current state of the system, triggering the re-planning of the mission plan if required, both at the mission control (global mission management), and at the vehicles, in a collaborative way. Hence, we have assigned TRAC a category 5 for the self-adaptation feature.

As with the plan specification, the TRAC proposal does not mention any kind of virtualization or abstraction. Therefore, we have assigned a category 1 for the virtualization feature.

Finally, although it is not explicitly explained in the proposal, the description of the ACE component of the TRAC architecture seems to use a task-by-task dispatching. In consequence, we have assigned it a category 3 for the dispatching feature.

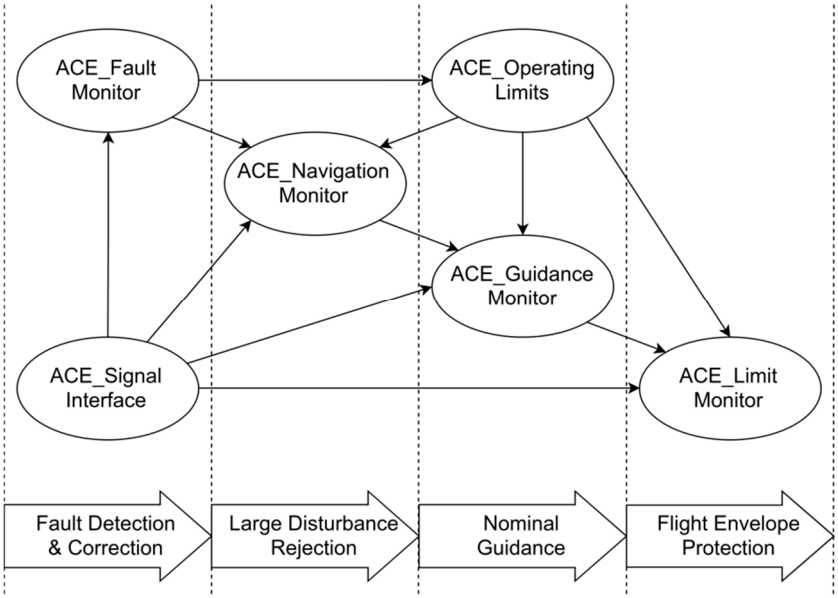

Figure 21. ACE Action Manager components.

A summary of the categorization for the measured features is included in Table 11.

Table 11. TRAC assessment.

Feature	Category	Explanation
Plan Specification	1	The proposal does not imply the use of a predefined mission plan specification.
Adaptation	5	Although there is no clear mention to the use of self-adaptation, the TRAC architecture evaluates the current state of the system continuously, allowing for the re-planning of the mission plan if required, both at the mission control and at the vehicles.
Virtualization	1	There is no mention to the use of virtualization or abstraction of the vehicles capabilities.
Dispatching	3	From the ACE description it seems that the dispatching of the actions to the vehicles is performed task by task.

3.7. ROSPlan

ROSPlan [82] is a framework that, according to their authors, provide a collection of tools for AI planning in a ROS system. Although ROSPlan is designed for any robotics, it was originally developed for the PANDORA [83] project for its use with AUVs, and was validated in the experiments that were carried out in said project [84].

ROSPlan uses a modular design where a number of ROS nodes provide the functionalities for planning, problem generation and plan execution. Initially presented in [85], in June 2018 a complete new version was presented.

The components of ROSPlan are defined as ROS nodes, and the general overview of the system is shown in Figure 22. There are five ROS nodes defined [86]:

- The knowledge base, used to store plan models using PDDL.
- The problem interface, used to generate the description of a problem using PDDL, and publish it on a ROS topic or write it to a file.

- The planner interface, used to invoke a planner and either publish the resulting plan to a ROS topic, or write it to a file.
- The parsing interface, used to translate a PDDL plan into ROS messages.
- The plan dispatch, which is responsible for the plan execution.

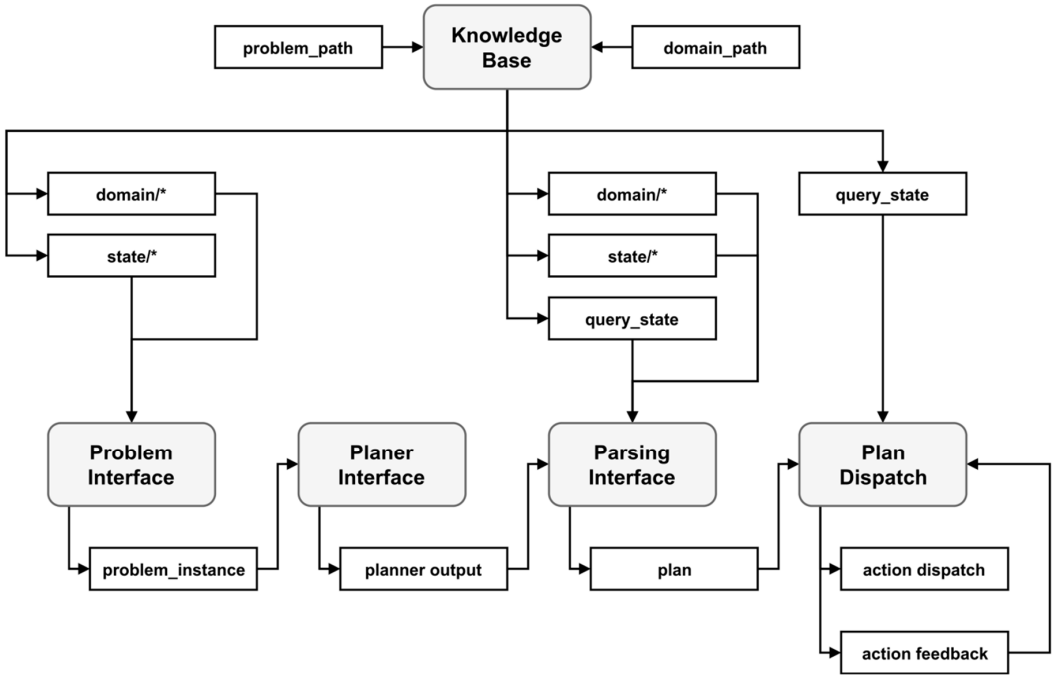

Figure 22. ROSPlan framework overview.

In ROSPlan the process starts by the specification of a problem as a PDDL problem instance using the Problem Interface node and the knowledge base. When the problem description is available, it is published to a ROS topic. A planner interface subscribed to that topic will receive the problem description, and using an AI planner will try to find a solution. If the planner is successful, the solution is published to another ROS topic. In this case it will be addressed to a subscribed parsing interface that will process the published solution to generete a plan representation that can be a petri-net plan, an ESTEREL plan, or a sequential plan. This plan representation is again published to another ROS topic addressed to a subscribed plan disptach. The plan dispatch receives the plan as a whole, and dispatches each action individually. A diagram for the plan dispatch node is shown in Figure 23.

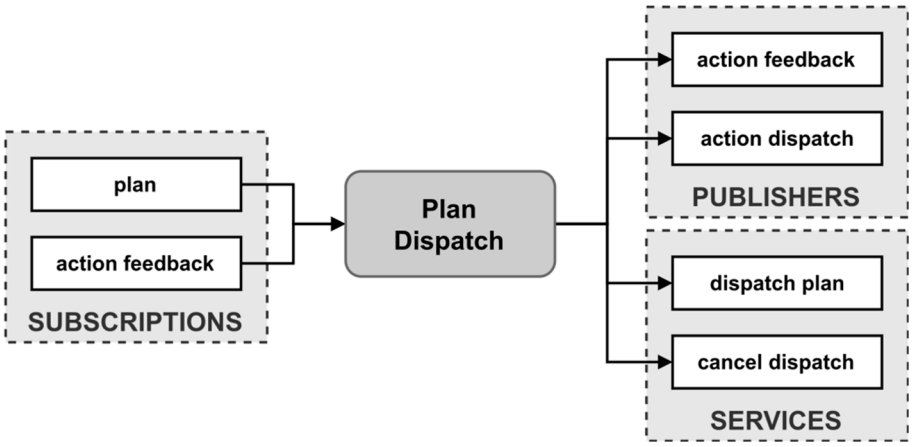

Figure 23. ROSPlan Plan Dispatch node.

Regarding the assessment of the features considered in this survey, the ROSPlan proposal is quite clear in the use of PDDL for the specification of the mission plans. There have been some efforts on developing additional ROS nodes capable for handling probabilistic plans described with RDDL [87], but they are not still part of the ROSPlan framework. Therefore, we have assigned it with a category 3 for the plan specification feature.

However, ROSPlan does not describe either an adaptation or a virtualization mechanism. Therefore, we have assigned a category 1 for both the adaptation and the virtualization features.

Finally, the dispatch of the tasks in the mission plan is done in a sequential way, and subsequently we have assigned a category 3 for the dispatching feature.

A summary of the categorization for the measured features is included in Table 12.

Table 12. ROSPlan assessment.

Feature	Category	Explanation
Plan Specification	3	All the plans managed by ROSPlan are described using PDDL.
Adaptation	1	There is no mention to any self-adaption capabilities provided by ROSPlan, and none can be identified from the proposal description.
Virtualization	1	There is no mention to any kid of virtualization or abstraction of the vehicles
Dispatching	3	The actions are dispatched one by one in a sequential way. The Plan Dispatch node just returns an error if a failure occurs while execution a plan, or a success message if all the actions in the plan are successfully executed.

4. Main Issues and Challenges

After presenting and introducing the selected architectures and frameworks individually in the previous section, it is of interest to make a comparison of them to have a clearer understanding of the status of mission planning and management solutions used for underwater robotics, and even in other domains. A summary of the presented architectures and frameworks is discussed in Section 4.1, and the challenges identified are discussed in Section 4.2.

4.1. Comparisons of the Proposals

In order to consider the overall assessment that has been done with all the proposals, Table 13 shows the categories for each one according to the criteria that were described in Section 2.

Table 13. Comparisons of categories assigned to the reviewed architectures and frameworks.

Architectures	Plan Specification	Adaptation	Virtualization	Dispatching
T-REX	2MHT	5	1	3
RAUVI	2	1	5	2
ICA	2MH	2	1	1
SWARMs	2MH	4	4	4
CPEF	1	3	1	3
TRAC	1	5	1	3
ROSPlan	3	1	1	3

When looking at Table 13 it is important to recall that a lower category does not necessarily mean a lower achievement at the specific feature. It is normal that different scenarios have different requirements, thus needing different capabilities on the selected features. However, it is quite remarkable that none of the architectures and frameworks evaluated have considered the use of probabilistic planning. Most of the solutions to deal with the unreliable nature of the underwater environment are related to react to unexpected events, either at the vehicle level or at the global mission control level. Continuing with planning, it is interesting that only three of the selected architectures, T-REX, ICA and SWARMs consider the use of multi-agent planning and hierarchical planning, and only T-REX included temporal constraints in the mission plan.

The same can be said about the virtualization of the vehicles from the point of view of mission management. Only two of the presented architectures, RAUVI and SWARMs, defined some mechanism to virtualize or abstract the vehicles capabilities. And this can be a potential issue to adapt the architecture to work with new vehicles with different capabilities.

Regarding the adaptation, there is greater diversity. TRAC describes an adaptation mechanism where the information is shared among all the agents in the architecture and the adaptation is performed using a decentralized control pattern. T-REX also mentions the use of a coordination control using self-adaptive loops at the reactors level. SWARMs also share information among the agents, but the decision to adapt is taken independently by each agent. In CPEF, the adaptation of the mission is only mentioned at the plan management level, while in ICA it is only mentioned at the vehicle level. Finally, RAUVI and ROSPlan do not mention adaptation at all, and the use of adaptation models cannot be inferred from their description.

Finally, regarding the dispatching of the mission plan to the vehicles, there is a great variance for the architectures designed for specific use of underwater vehicles (T-REX, RAUVI, ICA and SWARMs). On the other hand, CPEF, TRAC and ROSPlan all use a task-by-task dispatch, possibly because they are not considering the constrained nature of underwater communications. Further, only SWARMs described the possibility of using either a task-by-task or a full mission dispatch from the global mission manager.

4.2. Open Issues

Based on the analyses on the mission planning and managing architectures, a set of challenges have been identified and discussed as open issues in the following subsections.

4.2.1. High Dependency on Deterministic Plans for Non-Deterministic Environments

Probabilistic planning is hard. Instead of generating a sequence of actions, tasks, or even goals, a probabilistic plan has to provide a policy, that is, a function that given a state returns the next action with higher probability and reward to reach another state. However, an agent using a deterministic plan in an uncertain environment cannot guarantee to satisfy its goals [18]. As the underwater environment is not only uncertain, but also unreliable and prone to unpredictable events, the typical approach has been to use a deterministic plan to set the sequence of actions to achieve the desired goals, and also use a subsumption [88] architecture where the agents react to the unexpected events and try to adapt themselves to continue with the plan. The problem is when the plan cannot be fulfilled, and a new mission plan has to be created, either for repairing the current plan, or to just re-plan a whole new mission.

The use of other approaches for planning under uncertainty can provide an extra mechanism to improve the response to unexpected events that can be characterized with a stochastic model.

4.2.2. Lack of a Common Specification for Mission Plans

Continuing with planning, most architectures use an ad-hoc mission plan specification, preventing its resue in other projects. It is true that there are formal description languages from the automatic planning domain, like PDDL and RDDL that could be used. But it is also true that those languages are hard to understand for non-experts in automated planning. In addition, an interesting thought about the specification of the mission plan, including the mission domain and mission problem, is that nowadays systems also use context awareness descriptions that should be synchronized to the domain description of the problem. That being said, it could be worth to consider not only the specification of a common description language for mission plans, but also a mediator component integrated with the mission manager and the context manager that could adapt the descriptions in different formats to the needs of the components that will use them (like mission planners and reasoners), keeping the information synchronized for every participating agent.

4.2.3. Lack of Decentralized Self-Adaptive Control

Self-adaptation should be considered both at the agents executing the mission plan and at the global mission manager, both independently and coordinately. In general, self-adaption is only considered either at the vehicles, for adapting themselves while executing a mission to still be able to achieve the goal, or at the mission management, for re-planning when the outcomes of a task in a mission deviates enough to require a new plan. However, it could be interesting to explore the managing-managed approach from MAPE-K allowing the mission manager not only to react to events notified by the vehicles, but also to update the mission plan if there is new information from other sources that suggest that the update will improve the mission outcome.

4.2.4. Tightly coupling of the mission management to the specific vehicles

In general, the reviewed architectures generate mission plans that are tightly coupled to the vehicles that are being used in their respective projects, or are too generic to delegate the responsibility of matching the vehicles capabilities to the specific implementation for each scenario. However, to improve the reusability of the proposed architectures, a virtualization approach should be considered. By using a virtual representation of the vehicles within the scope of the architecture, it is possible to integrate new and legacy vehicles by just implementing an adaptor that on the on the one side matches to the specifics of the vehicle, and on other side matches the ones of the virtual model defined for the architecture.

5. Conclusions and Future Works

This paper presented a survey on the mission planning and management architectures for underwater cooperative robotics. Before exploring the mission planning and management details the architectures explored in the survey, a set of four features have been identified: mission plan specification, self-adaptation, virtualization, and mission plan dispatching and execution. Accordingly, we have defined a set of categories for each feature to facilitate the assessment of the explored features.

This survey differs from others as it focuses mainly in the mission planning and management, but also in the set of features considered for the comparison among the different architectures.

An assessment of the selected architectures based on the chosen features has been carried out. From the results, we cannot ignore that despite these architectures have been used successfully, there are still problems related to the mission management that have to be faced in future improvements and new designs. It can be foreseen that future works improving the current mission planning and management for cooperative robotics, in underwater environments or in any other unreliable and uncertain domain, or any new mission manager designed from the scratch should emphasize in the following aspects:

- The use of a formal mission plan specification would allow, or at least facilitate, the reuse of the mission planning and management in other domains of application, requiring only minor changes.
- Self-adaptation should be considered at every level of the architecture, including the AUVs, the mission manager, and the whole architecture. A decentralized control for self-adaptive systems should be taken into account.
- Virtualization, or at least the abstraction of the agents' capabilities should be taken into account to allow for the reusability of the mission manage with legacy and new systems, requiring only to implement the corresponding adapter for each new agent as a virtual representation.
- Dispatching and execution of the mission plan should be defined carefully, and taking into account the limitations of the environment in which the agents will execute the mission plan, and also some other security and safety concerns that may prevent an agent to be able to receive subsequent tasks in a mission plan.
- Additionally, it could be of interest to explore the use of probabilistic planning for uncertain environments, as well as other techniques that do not rely only on the use of a subsumption approach.

Author Contributions: Conceptualization, N.L.M. and P.C.; Funding acquisition, J.-F.M.-O.; Investigation, N.L.M.; Methodology, N.L.M.; Project administration, J.-F.M.-O.; Resources, N.L.M.; Supervision, J.-F.M.-O., P.C. and V.B.M.; Writing—original draft, N.L.M.; Writing—review & editing, N.L.M., P.C. and V.B.M. All authors have read and agreed to the published version of the manuscript.

Funding: The research leading to the presented results has been partially undertaken within the SWAMRs European project (Smart and Networking Underwater Robots in Cooperation Meshes), under Grant Agreement n. 662107-SWARMs-ECSEL-2014-1, partially supported by the ECSEL JU and the Spanish Ministry of Economy and Competitiveness (Ref: PCIN-2014-022-C02-02).

Acknowledgments: In this section you can acknowledge any support given which is not covered by the author contribution or funding sections. This may include administrative and technical support, or donations in kind (e.g., materials used for experiments).

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Fletcher, B. UUV master plan: A vision for navy UUV development. In Proceedings of the OCEANS 2000 MTS/IEEE Conference and Exhibition, Conference Proceedings (Cat. No.00CH37158), Providence, RI, USA, 1–14 September 2000; Volume 1, pp. 65–71.
2. Song, K.S. Design of Large Diameter Mine Countermeasure Hybrid Power Unmanned Underwater Vehicle. In *Autonomous Vehicle*; Zak, A., Ed.; IntechOpen: Rijeka, Croatia, 2016; pp. 127–146, ISBN 978-953-51-2585-3.
3. Heo, J.; Kim, J.; Kwon, Y. Technology Development of Unmanned Underwater Vehicles (UUVs). *J. Comput. Commun.* **2017**, *5*, 28–35. [CrossRef]
4. Kaminski, C.; Crees, T.; Ferguson, J.; Forrest, A.; Williams, J.; Hopkin, D.; Heard, G. 12 days under ice—An historic AUV deployment in the Canadian High Arctic. In Proceedings of the 2010 IEEE/OES Autonomous Underwater Vehicles, Monterey, CA, USA, 1–3 September 2010; pp. 1–11.
5. Camus, L.; Peddie, D.; Langeland, T.; Cook, J.; Kristiansen, T.; Tjostheim, S.; Graves, I.; Fietzek, P.; Sperrevik, A.-K.; Christensen, K.H.; et al. Autonomous surface and underwater vehicles reveal new discoveries in the Arctic Ocean. In Proceedings of the OCEANS 2019—Marseille, Marseille, France, 17–20 June 2019; pp. 1–8.
6. Acosta, G.G.; Ibanez, O.A.C.; Curti, H.J.; Rozenfeld, A.F. Low-cost Autonomous Underwater Vehicle for pipeline and cable inspections. In Proceedings of the 2007 Symposium on Underwater Technology and Workshop on Scientific Use of Submarine Cables and Related Technologies, Tokyo, Japan, 17–20 April 2007; pp. 331–336.
7. Mitchell, B.; Mahmoudian, N.; Meadows, G. Autonomous underwater pipeline monitoring navigation system. In Proceedings of the SPIE, Automatic Target Recognition XXIV, Baltimore, MD, USA, 5–6 May 2014; Volume 9090.
8. Jawhar, I.; Mohamed, N.; Al-Jaroodi, J.; Zhang, S. An Architecture for Using Autonomous Underwater Vehicles in Wireless Sensor Networks for Underwater Pipeline Monitoring. *IEEE Trans. Ind. Inf.* **2019**, *15*, 1329–1340. [CrossRef]
9. Diercks, A.; Asper, V.L.; Highsmith, R.; Woolsey, M.; Lohrenz, S.; McLetchie, K.; Gossett, A.; Lowe, M.; Joung, D.; McKay, L.; et al. NIUST—Deepwater horizon oil spill response cruise. In Proceedings of the OCEANS 2010 MTS/IEEE SEATTLE, Seattle, WA, USA, 20–23 September 2010; pp. 1–7.
10. Shukla, A.; Karki, H. Application of robotics in offshore oil and gas industry—A review Part II. *Robot. Auton. Syst.* **2016**, *75*, 508–524. [CrossRef]
11. Li, B.; Moridian, B.; Mahmoudian, N. Autonomous Oil Spill Detection: Mission Planning for ASVs and AUVs with Static Recharging. In Proceedings of the OCEANS 2018 MTS/IEEE Charleston, Charleston, SC, USA, 22–25 October 2018; pp. 1–5.
12. Kothari, M.; Pinto, J.; Prabhu, V.S.; Ribeiro, P.; de Sousa, J.B.; Sujit, P.B. Robust Mission Planning for Underwater Applications: Issues and Challenges. *IFAC Proc. Vol.* **2012**, *45*, 223–229. [CrossRef]
13. Fernández Perdomo, E.; Cabrera Gámez, J.; Domínguez Brito, A.C.; Hernández Sosa, D. Mission specification in underwater robotics. *J. Phys. Agents* **2010**, *4*, 25–33. [CrossRef]
14. Russell, S.; Norvig, P. *Artificial Intelligence: A Modern Approach*, 3rd ed.; Pearson: Harlow, UK, 2009; ISBN 978-0-13-604259-4.

15. Thompson, F.; Guihen, D. Review of mission planning for autonomous marine vehicle fleets. *J. Field Robot.* **2019**, *36*, 333–354. [CrossRef]
16. Atyabi, A.; MahmoudZadeh, S.; Nefti-Meziani, S. Current advancements on autonomous mission planning and management systems: An AUV and UAV perspective. *Annu. Rev. Control* **2018**, *46*, 196–215. [CrossRef]
17. MahmoudZadeh, S.; Powers, D.M.W.; Bairam Zadeh, R. State-of-the-Art in UVs' Autonomous Mission Planning and Task Managing Approach. In *Autonomy and Unmanned Vehicles*; Springer: Singapore, 2019; pp. 17–30.
18. Poole, D.L.; Mackworth, A.K. *Artificial Intelligence: Foundations of Computational Agents*, 2nd ed.; Cambridge University Press: Cambridge, UK, 2017; ISBN 9781107195394.
19. Bellingham, J.G. Platforms: Autonomous Underwater Vehicles. In *Encyclopedia of Ocean Sciences*; Elsevier: London, UK, 2009; pp. 473–484, ISBN 978-0-12-813082-7.
20. DNV GL. Rules For Classification Underwater technology. In *Part 5 Types of UWT Systems Chapter 8 Autonomous Underwater Vehicles*; DNV-GL: Oslo, Norway, 2015.
21. Pacini, F.; Paoli, G.; Kebkal, O.; Kebkal, V.; Kebkal, K.; Bastot, J.; Monteiro, C.; Sucasas, V.; Schipperijn, B. Integrated comunication network for underwater applications: The SWARMs approach. In Proceedings of the 2018 Fourth Underwater Communications and Networking Conference (UComms), Lerici, Italy, 28–30 August 2018; pp. 1–5.
22. Weyns, D.; Schmerl, B.; Grassi, V.; Malek, S.; Mirandola, R.; Prehofer, C.; Wuttke, J.; Andersson, J.; Giese, H.; Göschka, K.M. On Patterns for Decentralized Control in Self-Adaptive Systems. In *Software Engineering for Self-Adaptive Systems II*; Springer: Berlin, Germany, 2013; pp. 76–107, ISBN 978-3-642-35813-5.
23. Ghallab, M.; Nau, D.; Traverso, P. Hierarchical Task Network Planning. In *Automated Planning*; Elsevier: San Francisco, CA, USA, 2004; pp. 229–261, ISBN 978-1-558-60856-6.
24. Fikes, R.E.; Nilsson, N.J. Strips: A new approach to the application of theorem proving to problem solving. *Artif. Intell.* **1971**, *2*, 189–208. [CrossRef]
25. Barrett, A.; Christianson, D.; Friedman, M.; Kwok, C.; Golden, K.; Penberthy, S.; Smith, D.E.; Sun, Y.; Weld, D. PDDL: The Planning Domain Definition Language. In Proceedings of the 1st International Planning Competition (IPC), 4th International Conference on Artificial Intelligence Planning Systems (AIPS), Pittsburgh, PA, USA, 7–10 June 1998.
26. Sanner, S. Relational Dynamic Influence Diagram Language (RDDL): Language Description. In Proceedings of the 7th International Planning Competition (IPC-11), 21st International Conference on Automated Planning and Scheduling (ICAPS), Freiburg, Germany, 11–16 June 2011.
27. Fox, M.; Long, D. PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. *J. Artif. Intell. Res.* **2003**, *20*, 61–124. [CrossRef]
28. Edelkamp, S.; Hoffmann, J. PDDL2.2: The Language for the Classical Part of the 4th International Planning Competition. In Proceedings of the 4th International Planning Competition (IPC), 14th International Conference on Automated Planning & Scheduling (ICAPS-14), Whistler, BC, Canada, 3–7 June 2004.
29. Gerevini, A.; Long, D. Preferences and Soft Constraints in PDDL3. In Proceedings of the ICAPS-2006 Workshop on Preferences and Soft Constraints in Planning, English Lake District, UK, 6–10 June 2006; pp. 46–54.
30. Gerevini, A.E.; Long, D. *Plan Constraints and Preferences in PDDL 3*; Technical Report; Department of Electronics for Automation, University of Brescia: Brescia, Italy, 2005; Volume 75, pp. 1–12.
31. Gerevini, A.; Long, D. Bnf description of PDDL3.0. In Proceedings of the 6th International Planning Competition (IPC-08), Sydney, Australia, 14–18 September 2008.
32. Younes, H.; Littman, M. PPDDL1. 0: An Extension to PDDL for Expressing Planning Domains with Probabilistic Effects. In Proceedings of the 4th International Planning Competition (IPC-04), Whistler, BC, Canada, 3–7 June 2004.
33. Kovacs, D.L. A Multi-Agent Extension of PDDL3. In Proceedings of the 3rd Workshop on the International Planning Competition (IPC), Sao Paulo, Brazil, 25–29 June 2012; pp. 19–27.
34. Martín, H.J.A.; de Lope, J.; Maravall, D. Adaptation, anticipation and rationality in natural and artificial systems: Computational paradigms mimicking nature. *Nat. Comput.* **2009**, *8*, 757–775. [CrossRef]
35. Salehie, M.; Tahvildari, L. Self-adaptive software. *ACM Trans. Auton. Adapt. Syst.* **2009**, *4*, 1–42. [CrossRef]
36. Patrón, P.; Lane, D.M. Adaptive mission planning: The embedded OODA loop. In Proceedings of the 3rd SEAS DTC Technical Conference, Edinburgh, Scotland, 24–25 June 2008.

37. Huebscher, M.C.; McCann, J.A. A survey of autonomic computing—Degrees, models, and applications. *ACM Comput. Surv.* **2008**, *40*, 7. [CrossRef]
38. Brito, M.P.; Bose, N.; Lewis, R.; Alexander, P.; Griffiths, G.; Ferguson, J. The Role of adaptive mission planning and control in persistent autonomous underwater vehicles presence. In Proceedings of the 2012 IEEE/OES Autonomous Underwater Vehicles (AUV), Southampton, UK, 24–27 September 2012; pp. 1–9.
39. Kephart, J.O.; Chess, D.M. The vision of autonomic computing. *Computer* **2003**, *36*, 41–50. [CrossRef]
40. IBM. An architectural blueprint for autonomic computing. *IBM* **2005**, 1–36. Available online: http://www03.ibm.com/autonomic/pdfs/ACBlueprintWhitePaperV7.pdf (accessed on 20 January 2020).
41. Pacini, F.; Paoli, G.; Cayón, I.; Rivera, T.; Sarmiento, B.; Kebkal, K.; Kebkal, O.; Kebkal, V.; Geelhoed, J.; Schipperijn, B.; et al. The SWARMs Approach to Integration of Underwater and Overwater Communication Sub-Networks and Integration of Heterogeneous Underwater Communication Systems. In Proceedings of the ASME 2018 37th International Conference on Ocean, Offshore and Arctic Engineering, Madrid, Spain, 17–22 June 2018; American Society of Mechanical Engineers: New York, NY, USA, 2018; Volume 7.
42. Iglesia, D.G.D.L.; Weyns, D. MAPE-K Formal Templates to Rigorously Design Behaviors for Self-Adaptive Systems. *ACM Trans. Auton. Adapt. Syst.* **2015**, *10*, 15. [CrossRef]
43. Lucas Martínez, N.; Martínez Ortega, J.-F.; Hernández Díaz, V.; Martínez, J.-F.; Hernández Díaz, V. Virtualization of Event Sources in Wireless Sensor Networks for the Internet of Things. *Sensors* **2014**, *14*, 22737–22753. [CrossRef] [PubMed]
44. Bauer, M.; Boussard, M.; Bui, N.; Francois, C.; Jardak, C.; De Loof, J.; Magerkurth, C.; Meissner, S.; Nettsträter, A.; Olivereau, A.; et al. Internet of Things—IoT-A Deliverable D1.5—Final architectural reference model for the IoT v3.0. Technical Report FP7-257521; IoT-A project. 2013. Available online: https://www.researchgate.net/publication/272814818_Internet_of_Things_-_Architecture_IoT-A_Deliverable_D15_-_Final_architecture_reference_model_for_the_IoT_v30 (accessed on 20 January 2020).
45. Py, F.; Rajan, K.; Mcgann, C. A systematic agent framework for situated autonomous systems. In Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems, Toronto, ON, Canada, 10–14 May 2010; Volume 2, pp. 583–590.
46. Ropero, F.; Muñoz, P.; R-Moreno, M.D. A Versatile Executive Based on T-REX for Any Robotic Domain. In Proceedings of the 38th SGAI International Conference on Artificial Intelligence, AI 2018, Cambridge, UK, 11–13 December 2018; pp. 79–91.
47. MBARI—Autonomy—TREX. Available online: https://web.archive.org/web/20140903170721/https://www.mbari.org/autonomy/TREX/index.htm (accessed on 27 November 2019).
48. McGann, C.; Py, F.; Rajan, K.; Thomas, H.; Henthorn, R.; McEwen, R. A deliberative architecture for AUV control. In Proceedings of the IEEE International Conference on Robotics and Automation, Pasadena, CA, USA, 19–23 May 2008; pp. 1049–1054.
49. Controlled, Agile, and Novel Ocean Network (OCEAN) Interdisciplinary Field Experiments. Available online: https://www.mbari.org/science/upper-ocean-systems/canon/ (accessed on 21 January 2020).
50. Graham, R.; Py, F.; Das, J.; Lucas, D.; Maughan, T.; Rajan, K. Exploring Space-Time Tradeoffs in Autonomous Sampling for Marine Robotics. In *Experimental Robotics*; Springer: Heidelberg, Germany, 2013; pp. 819–839.
51. Rajan, K.; Py, F. T-REX: Partitioned inference for AUV mission control. In *Further Advances in Unmanned Marine Vehicles*; Roberts, G.N., Sutton, R., Eds.; Institution of Engineering and Technology: London, UK, 2012; pp. 171–199, ISBN 978-1-84919-480-8.
52. Executive T-Rex for ROS. Available online: http://wiki.ros.org/executive_trex (accessed on 27 November 2019).
53. McGann, C.; Py, F.; Rajan, K.; Ryan, J.; Henthorn, R. Adaptive control for autonomous underwater vehicles. *Proc. Natl. Conf. Artif. Intell.* **2008**, *3*, 1319–1324.
54. RAUVI: Reconfigurable AUV for Intervention. Available online: http://www.irs.uji.es/rauvi/news.html (accessed on 27 November 2019).
55. De Novi, G.; Melchiorri, C.; Garcia, J.C.; Sanz, P.J.; Ridao, P.; Oliver, G. A new approach for a Reconfigurable Autonomous Underwater Vehicle for Intervention. In Proceedings of the 2009 3rd Annual IEEE Systems Conference, Vancouver, BC, Canada, 23–26 March 2009; pp. 23–26.
56. Palomeras, N.; Garcia, J.C.; Prats, M.; Fernandez, J.J.; Sanz, P.J.; Ridao, P. A distributed architecture for enabling autonomous underwater Intervention Missions. In Proceedings of the IEEE International Systems Conference, San Diego, CA, USA, 5–8 April 2010; pp. 159–164.

57. Prats, M.; Ribas, D.; Palomeras, N.; García, J.C.; Nannen, V.; Wirth, S.; Fernández, J.J.; Beltrán, J.P.; Campos, R.; Ridao, P.; et al. Reconfigurable AUV for intervention missions: A case study on underwater object recovery. *Intell. Serv. Robot.* **2012**, *5*, 19–31. [CrossRef]
58. García, J.C.; Javier Fernández, J.; Sanz, R.M.P.J.; Prats, M. Towards specification, planning and sensor-based control of autonomous underwater intervention. *IFAC Proc. Vol.* **2011**, *44*, 10361–10366. [CrossRef]
59. Palomeras, N.; Ridao, P.; Carreras, M.; Silvestre, C. Using petri nets to specify and execute missions for autonomous underwater vehicles. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, St. Louis, MO, USA, 11–15 October 2009; pp. 4439–4444.
60. Trident FP7 European Project. Available online: http://www.irs.uji.es/trident/aboutproject.html (accessed on 28 November 2019).
61. Insaurralde, C.C.; Cartwright, J.J.; Petillot, Y.R. Cognitive Control Architecture for autonomous marine vehicles. In Proceedings of the IEEE International Systems Conference SysCon 2012, Vancouver, BC, Canada, 19–22 March 2012; pp. 1–8.
62. Insaurralde, C.C.; Petillot, Y.R. Intelligent autonomy for collaborative intervention missions of unmanned maritime vehicles. In Proceedings of the 2013 OCEANS—San Diego, San Diego, CA, USA, 23–27 September 2013; pp. 1–6.
63. Sanz, P.J.; Ridao, P.; Oliver, G.; Casalino, G.; Petillo, Y.; Silvestre, C.; Melchiorri, C.; Turetta, A. TRIDENT An European project targeted to increase the autonomy levels for underwater intervention missions. In Proceedings of the 2013 OCEANS, San Diego, CA, USA, 23–27 September 2013; pp. 1–10.
64. MacKenzie, C.M.; Laskey, K.; McCabe, F.; Brown, P.F.; Metz, R. *Reference Model for Service Oriented Architecture 1.0*; Organization for the Advancement of Structured Information Standards (OASIS) 2006; p. 31. Available online: http://docs.oasis-open.org/soa-rm/v1.0 (accessed on 4 February 2020).
65. Endsley, M.R. *Designing for Situation Awareness*, 2nd ed.; CRC Press: Boca Raton, FL, USA, 2016; ISBN 9780429146732.
66. SWARMS. Available online: http://swarms.eu/ (accessed on 30 November 2019).
67. Rodríguez-Molina, J.; Bilbao, S.; Martínez, B.; Frasheri, M.; Cürüklü, B. An Optimized, Data Distribution Service-Based Solution for Reliable Data Exchange Among Autonomous Underwater Vehicles. *Sensors* **2017**, *17*, 1802. [CrossRef]
68. Bastos, J.; Vujicic, Z.; Martínez Ortega, J.-F.; Rodriguez, J.; Johansen, G.; Vagia, M.; Sæter, E.; Martín, T.; Bilbao, S.; Mogos, A.; et al. *SWARMs D9.4 Executive Summary of Project Results v2*; SWARMs, 2016. Available online: http://swarms.eu/PDFs/Delivs/SWARMs_D9.4_Executive_summary_of_project_results_v2.pdf (accessed on 4 February 2020).
69. Bastos, J.; Vujicic, Z.; Martínez Ortega, J.-F.; Rodriguez, J.; Johansen, G.; Vagia, M.; Sæter, E.; Martín, T.; Bilbao, S.; Mogos, A.; et al. *SWARMs D9.5 Executive Summary of Project Results v3*; SWARMs, 2017. Available online: http://swarms.eu/PDFs/Delivs/SWARMs_D9.5_Executive%20summary%20of%20project%20results%20v3.pdf (accessed on 4 February 2020).
70. Bastos, J.; Vujicic, Z.; Martínez Ortega, J.-F.; Rodriguez, J.; Johansen, G.; Vagia, M.; Sæter, E.; Martín, T.; Bilbao, S.; Mogos, A.; et al. *SWARMs D9.6 Executive Summary of Project Results v4*; SWARMs, 2018. Available online: http://swarms.eu/PDFs/Delivs/SWARMs_D9.6_Executive%20summary%20of%20project%20results%20v4_v1.01.pdf (accessed on 4 February 2020).
71. Lucas Martínez, N.; Martínez-Ortega, J.F.; Rodríguez-Molina, J.; Zhai, Z. Proposal of an Automated Mission Manager for Cooperative Autonomous Underwater Vehicles. *Appl. Sci.* **2020**, *10*, 855. [CrossRef]
72. Myers, K.L. Towards a Framework for Continuous Planning and Execution. In Proceedings of the AAAI Fall Symposium on Distributed Continual Planning, Orlando, FL, USA, 22–24 October 1998; p. 6.
73. Continuous Planning and Execution. Available online: http://www.ai.sri.com/~{}cpef/ (accessed on 28 January 2020).
74. Myers, K.L. *JFACC Continuous Planning and Execution*; SRI International: Menlo Park, CA, USA, 2000. Available online: https://apps.dtic.mil/docs/citations/ADA383279 (accessed on 4 February 2020).
75. CPEF. Baseline Demonstration. Available online: http://www.ai.sri.com/~{}cpef/jfacc/acp-demo-dec97.html (accessed on 28 January 2020).
76. CPEF. Phase 2 Demonstration. Available online: http://www.ai.sri.com/~{}cpef/jfacc/ifd2.html (accessed on 28 January 2020).

77. CPEF. Final Demonstration. Available online: http://www.ai.sri.com/~{}cpef/jfacc/final-status.html (accessed on 28 January 2020).
78. Schaefer, P.; Colgren, R.D.; Abbott, R.J.; Park, H.; Fijany, A.; Fisher, F.; James, M.L.; Chien, S.; Mackey, R.; Zak, M.; et al. Technologies for reliable autonomous control (TRAC) of UAVs. In Proceedings of the 19th Digital Avionics Systems Conference. Proceedings (Cat. No.00CH37126), Philadelphia, PA, USA, 7–13 October 2000; Volume 1, pp. 1E3/1–1E3/7.
79. Schaefer, P.; Colgren, R.D.; Abbott, R.J.; Park, H.; Fijany, A.; Fisher, F.; James, M.L.; Chien, S.; Mackey, R.; Zak, M.; et al. Reliable autonomous control technologies (ReACT) for uninhabited air vehicles. In Proceedings of the 2001 IEEE Aerospace Conference Proceedings (Cat. No.01TH8542), Big Sky, MT, USA, 10–17 March 2001; Volume 2, pp. 2/677–2/684.
80. Johnson, T.L.; Sutherland, H.A.; Bush, S.F.; Yan, W.; Eaker, C. The TRAC mission manager autonomous control executive. In Proceedings of the 2001 IEEE Aerospace Conference Proceedings (Cat. No.01TH8542), Big Sky, MT, USA, 10–17 March 2001; Volume 2, pp. 2/639–2/646.
81. Fisher, F.W.; Estlin, T.A.; Gaines, D.; Schaffer, S.R.; Chouinard, C.; Knight, R.L. CLEaR: Closed Loop Execution and Recovery—A Framework for Unified Planning and Execution. *Interplanet. Netw. Dir. Technol. Sci. News* **2002**, *16*, 15–20.
82. What Is ROSPlan? Available online: https://kcl-planning.github.io/ROSPlan/ (accessed on 27 November 2019).
83. Persistent Autonomy through Learning, Adaptation, Observation and Re-planning|PANDORA Project|FP7|CORDIS|European Commission. Available online: https://cordis.europa.eu/project/id/288273 (accessed on 27 January 2020).
84. Maurelli, F.; Carreras, M.; Salvi, J.; Lane, D.; Kyriakopoulos, K.; Karras, G.; Fox, M.; Long, D.; Kormushev, P.; Caldwell, D. The PANDORA project: A success story in AUV autonomy. In Proceedings of the OCEANS 2016—Shanghai, Shanghai, China, 10–14 April 2016; pp. 1–8.
85. Cashmore, M.; Fox, M.; Long, D.; Magazzeni, D.; Ridder, B.; Carrera, A.; Palomeras, N.; Hurtos, N.; Carreras, M. ROSPlan: Planning in the Robot Operating System. In Proceedings of the International Conference on AI Planning and Scheduling (ICAPS), Jerusalem, Israel, 7–11 June 2015; pp. 333–341.
86. ROSPlan Overview. Available online: https://kcl-planning.github.io/ROSPlan/documentation/ (accessed on 29 November 2019).
87. Canal, G.; Cashmore, M.; Krivić, S.; Alenyà, G.; Magazzeni, D.; Torras, C. Probabilistic Planning for Robotics with ROSPlan. In *Towards Autonomous Robotic Systems*; Springer: London, UK, 2019; pp. 236–250.
88. Kortenkamp, D.; Simmons, R. Robotic Systems Architectures and Programming. In *Springer Handbook of Robotics*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 187–206.