Semantic knowledge-based representation for improving situation awareness in service oriented agents of autonomous underwater vehicles

Pedro Patrón, Emilio Miguélanchez, Joel Cartwright, Yvan R. Petillot
Ocean Systems Laboratory, Heriot-Watt University
Edinburgh, Scotland, UK, EH14 4AS
Email: {p.patron, e.miguelanchez, jjc5, y.r.petillot}@hw.ac.uk
Telephone: +44 (0) 131 451 3320
Fax: +44 (0) 131 451 4155

Abstract—This paper proposes a semantic world model framework for hierarchical distributed representation of knowledge in autonomous underwater systems. This framework aims to provide a more capable and holistic system, involving semantic interoperability among all involved information sources. This will enhance interoperability, independence of operation, and situation awareness of the embedded service-oriented agents for autonomous platforms. The results obtained specifically impact on mission flexibility, robustness and autonomy. The presented framework makes use of the idea that heterogeneous real-world data of very different types must be processed by (and run through) several different layers, to be finally available in a suited format and at the right place to be accessible by high-level decision making agents. In this sense, the presented approach shows how to abstract away from the raw real-world data step by step by means of semantic technologies. The paper concludes by demonstrating the benefits of the framework in a real scenario. A hardware fault is simulated in a REMUS 100 AUV while performing a mission. This triggers a knowledge exchange between the incipient fault diagnosis agent and the adaptive mission planner embedded agent. By using the proposed framework, both services can interchange information while remaining domain independent during their interaction with the platform. The results of this paper are readily applicable to land and air robotics.

I. INTRODUCTION

With the growing use of autonomous and semi-autonomous platforms and the increased data flows in modern Maritime operations, it is critical that the data is handled efficiently across multiple platforms and domains.

At present, knowledge representation is embryonic and targets simple mono-platform and mono-domain applications, therefore limiting the potential of multiple coordinated actions between agents. Consequently, the main application for autonomous underwater vehicles is information gathering from sensor data. In a standard mission flow, data is collected during mission and then post-processed off-line.

However, as decision making technologies evolve towards providing higher levels of autonomy, embedded service-oriented agents require access to higher levels of data representation. These higher levels of information will be required to provide knowledge representation for contextual awareness, temporal awareness and behavioural awareness.

Two sources can provide this type of information: the domain knowledge extracted from the original expert or the inferred knowledge from the processed sensor data. In both cases, it will be necessary for the information to be stored, accessed and shared efficiently by the deliberative agents while performing a mission. These agents, providing different capabilities, might be distributed among the different platforms working in collaboration.

This article focuses on the study of a semantic world model framework for hierarchical distributed representation of knowledge in autonomous underwater systems. The framework uses a pool of hierarchical ontologies for representation of the knowledge extracted from the expert and the processed sensor data. Based on their capabilities, service-oriented agents can gain access to the different levels of information and might also contribute to the enrichment of the knowledge. If the required information is unavailable, the framework provides the facility for requesting that information be generated by other agents with the necessary capabilities.

The proposed framework enhances interoperability, independence of operation, and situation awareness of the embedded service oriented agents for autonomous platforms. The results obtained specifically impact on mission flexibility, robustness and autonomy.

This paper is structured as follows: Section II describes our evolved view of the sensing-decision-acting loop for autonomy for AUVs. Section III presents an overview of the role of ontologies as knowledge representation, including the main features behind this approach. Section IV describes the framework focusing on the semantic representation of the vehicle’s knowledge. Section V presents the interaction of the diagnosis and planner using the framework. Section VI demonstrates the benefits of the proposed framework in a real scenario, where a hardware fault is simulated in a REMUS 100 AUV while performing a mission. This paper ends in Section VII with the conclusions and future work.
II. SITUATION AWARENESS FOR AUTONOMY

The human capability for dealing with and understanding highly dynamic and complex environments is known as situation awareness (SAH). SAH breaks down into three separate levels: perception of the environment, comprehension of the situation and projection of the future status.

According to Boyd, decision making occurs in a cycle of observe-orient-decide-act [1]. The Observation component corresponds to the perception level of SAH. The Orientation component contains the previously acquired knowledge and understanding of the situation. The Decision component represents the SAH levels of comprehension and projection. This last stage is the central mechanism enabling adaptation before closing the loop with the final Action stage. Note that it is possible to take decisions by looking only at orientation inputs without making any use of observations.

Based on the autonomy levels and environmental characteristics, SAH definitions can be directly applied to the notion of unmanned vehicle situation awareness (SAV) [2]. The levels of situation awareness for individual unmanned vehicle systems (SAV) span from full human control to fully autonomous unmanned capabilities (see Fig. 1).

In current implementations, the human operator constitutes the decision making phase. When high-bandwidth communication links exist, the operator remains in the loop during the mission execution. Examples of the implementation of this architecture are existing Remote Operated Underwater Vehicles (ROVs). However, when the communication is poor, unreliable or not allowed, the operator tries, based only on the initial orientation or expertise, to include all possible behaviours to cope with execution alternatives. This has unpredictable consequences, in which unexpected situations can cause the mission to abort and might even cause the loss of the vehicle. Examples of this architecture are current implementations for AUVs.

In order to achieve autonomous decision making, the service oriented agents in the platform must be supplied with the same level of knowledge as the operator. This can be achieved by using a semantic world model and a battery of ontologies for each of the agent’s domains.

III. OVERVIEW IN THE ROLE OF ONTOLOGIES AS KNOWLEDGE REPRESENTATION

From a philosophical point, the term ontology has been defined as the study or concern about what kinds of things exist – what entities or 'things' there are in the universe [3]. From a practical point, ontologies are viewed as the working model of entities and interactions either generically (e.g. the SUMO ontology) or in some particular domain of knowledge or practice, such as predictive maintenance or subsea operations. The following definition is given in [4]:

An ontology may take a variety of forms, but necessarily it will include a vocabulary of terms, and some specification of their meaning. This includes definitions and an indication of how concepts are inter-related which collectively impose a structure on the domain and constrain the possible interpretations of terms.

Gruber defines an ontology as 'the specification of conceptualisations', used to help programs and humans share knowledge [5]. The combination of these two terms embraces the knowledge about the world in terms of entities (things, the relationships they hold and the constraints between them), with its representation in a concrete form. One step in this specification is the encoding of the conceptualisation in a knowledge representation language. The goal is to create an agreed-upon vocabulary and semantic structure for exchanging information about that domain.

The main components of an ontology are concepts, relations, instances and axioms. A concept represents a set or class of entities or things within a domain. A fault is an example of a concept within the domain of diagnostic. A finite set of definitions is called the TBox, where is also included the set of terminological axioms for every atomic concept. Axioms are used to constrain the range and domain of the concepts, such as the father is a man that has a child.

Instances are the 'things' represented by a concept, such as a FaultySensor is an instance of the concept Fault. Strictly speaking, an ontology should not contain any instances, because it is supposed to be a conceptualisation of the domain. The combination of an ontology with associated instances is what is known as a knowledge base (see Fig. 2). However, deciding whether something is a concept or an instance is difficult, and often depends on the application [6]. This finite set of assertions about individuals is called the ABox.

In the ABox, besides the concept assertions, one can also specify role assertions or, in other words, the relations describing the interactions between individuals. For example, the property isComponentOf might link the individual SensorX to the individual PlatformY.

Following the definition and characterization of ontologies, one of the main objectives for an ontology is that it should be
re-usable [4]. This ambition distinguishes an ontology from a database schema, even though both are conceptualisations. For example, a database schema is intended to satisfy only one application, but an ontology could be re-used in many applications. However, an ontology is only re-usable when it is to be used for the same purpose for which it was developed. Not all ontologies have the same intended purpose and may have parts that are re-usable and other parts that are not. They will also vary in their coverage and level of detail.

Furthermore, one of the benefits of the ontology approach is the extended querying that it provides, even across heterogeneous data systems. The meta-knowledge within an ontology can assist an intelligent search engine with processing your query. Part of this intelligent processing is due to the capability of reasoning that makes possible the publication of machine understandable meta-data, opening opportunities for automated information processing and analysis. For instance a diagnostic system, using an ontology of the system, could automatically suggest the location of a fault in relation to the occurrence of symptoms and alarms in the system. The system may not even have a specific sensor in that location, and the fault may not even be categorized in a fault tree. The reasoning interactions with the ontology are provided by the reasoner, which is an application that enables the domain’s logic to be specified with respect to the context model and applied to the corresponding knowledge, i.e. the instances of the model (see Fig. 2). A detailed description of how the reasoner works is outside the scope of this paper.

**IV. FRAMEWORK**

SAV consists in enabling the vehicle to autonomously understand the 'big picture'. This picture is composed of the experience achieved from previous missions (orientation) and the information obtained from the sensors while on mission (observation). The TBox and Abox that have already been introduced as main components in any knowledge base can be assigned to the orientation and observation components of SAV respectively. For each knowledge representation, its TBox-ABox pair will not only describe the relationships between concepts but also facilitate the decision making process of the service-oriented agents. Reasoning capabilities allow concept consistency providing reassurance that SAV remains stable through the evolution of the mission. Also, inference of concepts and relationships allows new knowledge to be extracted or derived from the observed data.

A set of ontologies has been developed in order to represent the knowledge information required for SAV. A key part in the ontology engineering discipline is the construction and organization of these libraries of ontologies, which should be designed for maximum reusability [7], [5]. A major challenge in building these libraries is to define how these ontologies are constructed and organized, and what relations should be present between them. Existing approaches propose three discrete levels of vertical segmentation including (1) upper/foundation, (2) core/domain and (3) application (see Fig. 3).

The top layer in Fig. 3 refers to the foundational ontologies (FOs) (or upper ontologies) which represent the very basic principles, and meets the practical need of a model that has as much generality as possible, to ensure reusability across different domains. There are several standardized upper ontologies available for use, including Dublin Core, GFO, OpenCyc/ResearchCyc, SUMO, and DOLCE.

The second level of the ontology hierarchy represents the core domain ontology, which is arguably another of the building blocks for information integration. The goal of a core ontology is to provide a global and extensible model into which data originating from distinct sources can be mapped and integrated. This canonical form can then provide a single knowledge base for cross-domain tools and services (e.g., vehicle resource/capabilities discovery, vehicle physical breakdown, and vehicle status). A single model avoids the inevitable combinatorial explosion and application complexities that results from pair-wise mappings between individual metadata formats and/or ontologies.

At the bottom layer, an application ontology provides an underlying formal model for tools that integrate source data and perform a variety of extended functions. As such, higher levels of complexity are tolerable and the design should be motivated more by completeness and logical correctness than human comprehension. Target areas of these application ontologies are found in the diagnostics of the vehicle and the planning of the mission.

Raw data gets parsed from sensors into assertions during the mission using an adapter (see Fig. 4).
A. Foundation and Core Ontology

To lay the foundation for the knowledge representation of unmanned vehicles, consideration was placed on the Joint Architecture for Unmanned Systems (JAUS). This was originally developed for the ground domain only, and has recently been extended to all domains trying to provide a common set of architecture elements and concepts [8]. The JAUS model separates the service-oriented agents, called Functional Agents, in six different functional sets: Command, Telecommunications, Mobility, Payload, Maintenance and Training. It also classifies four different sets of Knowledge Stores: Status, World map, Library and Log. Our experience has shown that there overlap exits between these different sets of knowledge stores. The approach proposed in this paper provides more flexibility in the way the information can be accessed and stored, while being JAUS compliant at the communication level between agents.

Within the proposed framework, JAUS concepts are considered as the foundation for the knowledge representation. The core ontology developed in this work extends these concepts while remaining focused in the domain of unmanned systems. Some of the knowledge concepts identified related with this domain are:

- Platform: Static or mobile (ground, air, underwater vehicles),
- Payload: Hardware with particular properties, sensors or modules,
- Module: Software with specific capabilities,
- Sensor: A device that receives and responds to a signal or stimulus,
- Driver: Module for interaction with a specific sensor/actuator,
- Waypoint: Position in space with coordinate and tolerance,
- Coordinate: Local frame, global frame, angular,
- Velocity: Linear, angular,
- Attitude: Roll, pitch, yaw, . . .

Fig. 5 represents a snapshot of the core ontology showing the key concepts involved in the test case scenario, and the relationships between them.

To support generic context-aware concepts, this framework makes use of the SOUPA (Standard Ontology for Ubiquitous and Pervasive Applications) [9] ontology, which is the core of the Context Broker Architecture (CoBrA) [10], a system for supporting context-aware computing. The contribution of the SOUPA ontology is the spatio-temporal representation, which allows the representation of time instants, intervals, and temporal relations, as well as space enhanced by high-level concepts such as movable spatial thing, geographical entity or geometry concepts.

B. Application ontology

Application concepts are handled at the executive layer and are used to ground the abstract concepts managed by the software agents running in the vehicle. In the case study presented in this paper, these agents are the mission planner and the FDD system. These agents make use of the proposed framework and allow the transition from the Deliberative to the Action phase of the OODA loop. The ontologies encapsulating the knowledge handled by these agents are described in the following subsections.

V. FAULT TOLERANT ADAPTIVE MISSION PLANNING

In recent years, emphasis for increasing AUV’s operability has been focused in increasing AUV’s survivability by reducing the susceptibility and vulnerability of the platform [11]. Recent approaches in rules of collision [12] and wave propagation techniques [13] for obstacle avoidance, collision avoidance and escape scenarios [14] have focused on reducing susceptibility by looking at the adaptation of the vehicle’s trajectory plan.

However, when the vehicle faces unforeseen events, such as unexpected component failures or unplanned interactions with the surrounding environment, the focus of the mission should shift to ‘reconfigure’ itself to use alternative combinations of the remaining resources. The underwater domain has scarce communications bandwidth and tight response constraints to keep the operator in the loop. In such a challenging environment, autonomous embedded recoverability is a key capability for vehicle’s endurance. This can be achieved via adaptation of the vehicle’s mission plan.

Adapting a mission on the fly in response to events is feasible with embedded planners. However, they are limited to the quality and scope of the available information. For them to be effective, the mission programmer must predict all possible situations, which is clearly impractical. Therefore, to adapt mission plans due to unforeseen and incipient faults, it is required that accurate information is available, to recognize that a fault has occurred and deduce the root cause of the failure. For example, if a propeller drive shaft developed too much shaft friction, then the added current load may overburden the drive motor. Identification of such a condition before the shaft bearing fails would allow rectification of the mission before a major breakdown occurs.
AUVs are generally equipped with Failure Diagnosis and Detection (FDD) systems based on damage control that results in the vehicle resurfacing in the event of any fault in the system. But future industrial and military AUVs may require systems that operate even while partially damaged. Hence, it is of importance to develop a system which not only detects failure in the underwater vehicle but also provides meaningful and reliable information to counterpart modules, such as the mission planner, to adapt and recover from the failures.

A. Diagnosis

In the field of diagnostics, the gathering and processing of knowledge in AUVs, as in most robotic systems, is classified into two categories (i) model free and (ii) model based.

Model free methods, such as rule-based, use limit checking of sensors for the detection of faults. Rule-based diagnostic is the most intuitive form of diagnostic, where through a set of mathematical rules, observed parameters are assessed for conformance to anticipated system condition. Knowledge gained is thus explicit as rules are either satisfied or not. Rule-based reasoning is an easy concept to employ, and if kept simple requires little development time, provided that expert tacit knowledge (system behaviour awareness) can be straightforwardly transformed to explicit knowledge (rules). However, these rules use knowledge gained from outside observation of the system rather a representation of any internal mechanisms. In other words, they represent only the relationship between symptoms and failures, and cannot provide a coherent explanation of the failure. Furthermore, they exhibit a lack of flexibility as only faults that have been explicitly described can be diagnosed. The main advantage of a rule-based system is that execution time is generally much faster than other methods using more sophisticated models.

Model-based diagnosis systems rely on the development of a model constructed from detailed in-depth knowledge of the system (preferably from first principles). There is a wide range of models available for diagnosis, including mathematical, functional and abstract [15]. The fault detection and isolation (FDI) community has tackled the diagnostic task by comparing simulated results to real results, and detecting abnormalities accordingly based mainly on analytical redundancy. The main advantage of this approach is that it can be adapted easily to changes in the system environment by changing inputs to the model being simulated. However, the numerical models are based on behaviour of the system, with little knowledge of the structure and functionality of the components. Also, there is no mechanism to detect multiple faults and it requires expensive computation.

Currently, there is an increasing move away from FDD model-based to structure and data-driven methods, because complex dynamic systems are difficult to model, based on analytical redundancy alone. Uppal and Patton argue that the interesting combination of certain aspects of qualitative and quantitative modelling strategies can be made [16]. They further state that qualitative methods alone should be used if faults cause qualitative changes in system performance and when qualitative information is sufficient for diagnosis. Qualitative methods are essential if measurements are approximate or if the system cannot be described by differential equations with known parameters. The FDD community uses numerical models to describe the physical system. It is a complicated task to model large interconnected systems using analytical numerical models, which require expensive computation and are not able to detect all types of faults.

Recent developments in defining ontologies as a knowledge representation approach for a domain provide significant potential in model design, able to encapsulate the essence of the diagnostic semantic into concepts and to describe the key relationships between the components of the system being diagnosed. In this paper, a framework based on the knowledge space of the system is developed to guide the fault detection process and better automated knowledge discovery to improve diagnostics.

The aim of the ontological diagnostic model is to model the behaviour of the AUV internal systems, in order to monitor the health of the vehicle and to report any critical or incipient status.

To model the behaviour of all components and subsystems considering from sensor data to possible model outputs, the diagnostic domain ontology is designed and built based on ontology design patterns [17]. Ontology patterns facilitate the construction of the ontology, promote the main goal of re-use, and guarantee the consistency of the ontology if it is applied in a different domain. In this work, the representation of the diagnostic concepts are based on a system observation design pattern, which is shown in Fig. 6.

![Fig. 6. Representation of the System Observation Pattern.](image)

Note that this knowledge representation of diagnostic concepts is linked to concepts already described in the core ontology, such as the status of the system which is the key component in the exchange of information between the FDD and mission planner agents.

B. Mission Planning

On the mission planning front, knowledge modelling is implemented by using some kind of language representation. This
language is then used to express the input to the planner. Language vocabularies generally include the information concepts and the grammars are used for describing the relationships and constraints between these concepts.

The STRIPS language from [18] is generally acknowledged as the basis for classical planning. Basic concepts in this language are: an initial state, a goal state (or set of goal states) and a set of actions which the planner can perform. Each action consists of a set of preconditions, which must be true for the action to be performed, and a set of postconditions, which will be true after the action has been completed. A classical planner then normally attempts to apply actions whose postconditions satisfy the goal, recursively applying other actions to meet the preconditions until a complete plan is formed (if available), which can be executed from the initial state and ends at the goal.

Hierarchical Task Network (HTN) [19] planning extends STRIPS and provides a more expressive planning framework. A planning problem described in HTN consists of: (i) a set of primitive tasks, which are roughly analogous to the actions in the STRIPS system and can be executed directly, (ii) a set of compound tasks and rules for decomposing them into primitive tasks, (iii) a set of goal tasks, which are roughly analogous to the goals in the STRIPS system, but more general. These are specified in terms of conditions which must be made true. A normal HTN planner then attempts to satisfy the goal task by decomposing it into smaller and smaller subtasks, until the goal can be achieved entirely with primitive tasks. The plan itself is represented as hierarchical network of tasks with the goal task at the top and the primitive tasks at the bottom, also showing any dependencies between the tasks.

The PDDL language was originally created by [20] and stands for Planning Domain Definition Language. It is the planning language used by the planning community during the bi-annual International Planning Competition that takes place during the ICAPS Conference [21]. It can be considered as an extension of the original STRIPS language with extra functionality added. PDDL intends to express the physics of a planning domain, including what predicates there are, what actions are possible, what the structure of compound actions is and what the effects of actions are. At its last version, it contains extensions for dealing with extended goals where good quality plans are as valid as optimal quality plans. It is a very complex language with complex syntactic features such as specification of safety constraints or hierarchical actions. State of the art plan generators are not able to handle the entire set of PDDL language features. In consequence, several versions of the language have been produced, describing a subset of features called requirements that are extended every two years when the competition takes place. The language takes the same role as the input to STRIPS and HTN planning systems, but is vastly more expanded and more flexible.

The adaptive decision making process requires concepts for generating a mission following a declarative goal-based approach instead of a classic procedural waypoint-based description. This can be achieved by looking at the concepts described by PDDL. However, in order to provide a solution to a mission failure, it also requires concepts capable of representing incidents or problems occurring during the mission. These concepts have been extracted from [22]. Some of the most important concepts identified for mission plan adaptability are:

- Resource: state of an object (physical or abstract) in the environment, (vehicle, position, sensor,...),
- Action: Modification of the state of resources. (calibrate, classify, explore,),
- Catalyst resource: Resources that are not consumed for an action but needed for the proper execution of the action. (sensor activation),
- Plan gap: Actions that may no longer be applicable. At least two ways of achieving a subgoal but a commitment has not been taken yet,
- Gap: A non-executable action,
- Execution: When an action is executed successfully,
- Mission Failure: An unsuccessful execution.

The representation of the planning concepts related to the mission plan and mission actions is shown in Fig. 7.

![Fig. 7. Representation of the mission plan description.](image)

Note that this knowledge representation of planning concepts are linked to concepts already described in the core ontology, such as the list of capabilities required to perform each of the mission actions.

VI. TEST CASE SCENARIO

This section aims to demonstrate the benefits of the framework inside the mine counter measure (MCM) scenario using AUVs. In this scenario, AUVs support and provide solutions for mine-hunting and neutralisation. The operation involves high levels of uncertainty and risk of damage to the vehicle. Navigating in such a hazardous environment is likely to compromise the vulnerability of the platform. Additionally, if a vehicle is damaged or some of its components fail, mission adaption will be required to cope with the new restricted capabilities.

The example shows the benefits of using an ontological representation to describe the $SA_W$ and how the FDD and adaptive planner modules are capable of interchanging knowledge in order to dynamically adapt the mission to the changes occurring in the environment.
A. Pre-mission reasoning

One of the first contributions of the framework is that it can to automatically answer some important questions before starting the mission:

- Is this platform configuration suitable to successfully perform this mission?

In order to answer this question, new knowledge can be inferred from the original information provided by the operator. The Core Ontology rule engine is executed providing additional knowledge. A set of predefined rules orient the knowledge base into inferring new relationships between instances. An example of a rule dealing with the transfer of payload capabilities to the platform is represented in Eq. 1.

\[
\text{core : isCapabilityOf(\?Capability, \?Payload)} \land \\
\text{core : isPayloadOf(\?Payload, \?Platform)} \\
\rightarrow \text{core : isCapabilityOf(\?Capability, \?Platform)} \quad (1)
\]

Once all the possible knowledge has been extracted, it is possible to query the knowledge base in order to extract the list of capabilities of the platform (see Eq. 2) and the list of requirements of the mission (see Eq. 3).

\[
\text{SELECT} \quad \?\text{Platform, \?Cap} \\
\text{WHERE} \quad \{ \text{rdf:type (\?Platform, core:Platform)} \land \\
\text{core:hasCapability(\?Platform, \?Cap)} \} \quad (2)
\]

\[
\text{SELECT} \quad \?\text{Mission, \?Req} \\
\text{WHERE} \quad \{ \text{plan:hasAction(\?Mission, \?Action)} \land \\
\text{plan:hasRequirement(\?Action, \?Req)} \} \quad (3)
\]

This way, it is possible to define that the requirements of the mission are:

- WaypointManeuver Capability ∈ core:JAUS_Manuever_Capability
- ComputerAidedClassification Capability ∈ core:JAUS_Autonomous_RSTA-I_Capability
- ComputerAidedDetection Capability ∈ core:JAUS_Autonomous_RSTA-I_Capability
- SidescanSensor Capability ∈ core:JAUS_Environmental_Sensing_Capability

which are a subset of the platform capabilities. Therefore, in this particular case, the platform configuration suits the mission requirements.

B. In mission adaptation

The framework can also provide knowledge transfer between the FDD and the adaptive mission planner while on mission. When the environmental observations being monitored by the FDD indicate that the mission under execution is affected, the mission planner is activated and the mission plan is adapted. When a mission failure occurs during the execution, two possible levels of repair can be identified: mission execution repair and mission plan repair. Execution repair changes the instantiation of the mission plan such that either: an action that was previously instantiated by some execution parameters is no longer instantiated, or an action that was previously instantiated is newly bound by an execution already part of the plan. Plan repair modifies the plan itself, so that it uses a different composition, though it still uses some of the same constraints between actions. It might also entail the elimination of actions which have already been instantiated. The process of how the mission is adapted is out of the scope of this paper. The reasoning process that evolves from the observations of the environment to the inputs to the adaptive mission planner is described by looking at the way that the framework can provide an answer to the following question:

- Are the observations coming from the environment affecting the mission currently under execution?

In order to explain the reasoning process involved, a component fault has been simulated in the vehicle while performing the mission. For this case, the gains of the starboard transducer of the sidescan sonar of the vehicle were modelled to drop to their minimum levels half way through the execution of the seaboat survey action.

The first step is to deal with the information coming from the sensor, which is signalling a low gain in the starboard transducer. This signal triggers a symptom instance, which has an associated event level. This event level, represented in the diagnostic ontology using a value partition pattern, plays a key role in the classification of the instances in the Fault concept between being critical or incipient. This classification is represented axiomatically in the Eq. 4 and 5.

\[
\text{diag:CriticalFault} \subseteq \ldots \\
\text{diag:Fault} \ni \exists \text{diag:causedBySymptom} \ldots \\
\text{diag:Symptom} \ni \exists \text{diag:hasEventLevel} \ldots \\
\text{diag:Level} \ni \exists \text{diag:High}) \quad (4)
\]

\[
\text{diag:IncipientFault} \subseteq \ldots \\
\text{diag:Fault} \ni \exists \text{diag:causedBySymptom} \ldots \\
\text{diag:Symptom} \ni \exists \text{diag:hasEventLevel} \ldots \\
\text{diag:Level} \ni \exists \text{diag:Med}) \quad (5)
\]

Once the fault individuals are re-classified, the status of the related system is instantiated using the most updated fault information, which is represented by an annotation property. Therefore, a critical status of the sidescan starboard transducer is caused by a critical fault. However, due to the fact that the sidescan sonar is composed of two transducers, one
malfunctioned transducer only causes an incipient status of the sidescan sonar.

The characteristics of the Status concepts and reusability presented here supports the transfer of knowledge between the two involved agents. Now, the mission planner agent is responsible to adapt the mission according to this new piece of information.

Eq.6 reports to the mission planner that the two survey actions in the mission are affected by the incipient status of the sidescan sonar. In this case, the sensor required by the action is reporting an incipient fault. The action can be therefore modified by only adapting the way it is being executed, an execution repair. If both transducers were down and the FDD reported a critical status of the sidescan sensor, a plan repair adaptation of the mission plan would have been required instead. In that case, it would have been necessary to look for redundant components or platforms with similar capabilities in order to be able to perform the action.
VII. CONCLUSION

We have presented a semantic-based framework that provides the core architecture for knowledge representation for service oriented agents in autonomous vehicles. The framework combines the initial expert orientation and the observations acquired during mission in order to improve the situation awareness of the vehicle. It has direct impact on the knowledge distribution between embedded agents at all levels of representation.

This work is highly relevant to underwater platforms especially where autonomy and on-board decision making are required. The approach is extendable to other agents and provides benefits in single missions and collaborative missions, improving local (agent level) and global (system level) situation awareness, temporal context and planned mission behaviour. This is currently unavailable in autonomous systems.

The framework has been applied to the problem of fault-tolerant adaptive mission planning. In this scenario, the approach has shown how the FDD and the mission planner agent can collaborate together in order to provide a solution to mission action failures when component faults are detected in the platform.

ACKNOWLEDGMENT

Our thanks to the members of the Ocean Systems Laboratory for their inputs and helpful critique. Thanks also to all at SeeByte Ltd, Edinburgh for providing the necessary AUV mobilisation, practical trials infrastructure and knowledge to run the experiments in the real world.

The work reported in this paper is partly funded by the Project RT/COM/5/059 from the Competition of Ideas and by the Project SEAS-DTC-AA-012 from the Systems Engineering for Autonomous Systems Defence Technology Centre, both established by the UK Ministry of Defence.

REFERENCES


Fig. 10. Representation of the mission planning action execution parameters.