We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

9

Embedded Knowledge and Autonomous Planning: The Path Towards Permanent Presence of Underwater Networks

Pedro Patrón, Emilio Miguelañez and Yvan R. Petillot
Ocean Systems Laboratory, Heriot-Watt University
United Kingdom

1. Introduction

Oceanographic observatories, year-round energy industry subsea field inspections and continuous homeland security coast patrolling now all require the routine and permanent presence of underwater sensing tools.

These applications require underwater networks of fixed sensors that collaborate with fleets of unmanned underwater vehicles (UUVs). Technological challenges related to the underwater domain, such as power source limitations, communication and perception noise, navigation uncertainties and lack of user delegation, are limiting their current development and establishment. In order to overcome these problems, more evolved embedded tools are needed that can raise the platform's autonomy levels while maintaining the trust of the operator.

Embedded decision making agents that contain reasoning and planning algorithms can optimize the long term management of heterogeneous assets and provide fast dynamic response to events by autonomously coupling global mission requirements and resource capabilities in real time. The problem, however, is that, at present, applications are mono-domain: Mission targets are simply mono-platform, and missions are generally static procedural list of commands described *a-priori* by the operator. All this, leaves the platforms in isolation and limits the potential of multiple coordinated actions between adaptive collaborative agents.

In a standard mission flow, operators describe the mission to each specific platform, data is collected during mission and then post-processed off-line. Consequently, the main use for underwater platforms is to gather information from sensor data on missions that are static and incapable to cope with the long term environmental challenges or resource changes.

In order for embedded service agents to make decisions and interoperate, it is necessary that they have the capability of dealing with and understanding the highly dynamic and complex environments where these networks are going to operate. These decision making tools are constrained to the quality and scope of the available information.

Shared knowledge representation between embedded service-oriented agents is therefore necessary to provide them with the required common situation awareness. Two sources can provide this type of information: the domain knowledge extracted from the expert (orientation) and the inferred knowledge from the processed sensor data (observation). 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 and working in collaboration, might even be distributed among the different platforms or sharing some limited resources.

1.1 Contribution

In this chapter, we first provide a review to the different approaches solving the decision making process for UUV missions. Then, we propose a semantic framework that provides a solution for hierarchical distributed representation of knowledge for multidisciplinary agent interaction. This framework uses a pool of hierarchical ontologies for representation of the knowledge extracted from the expert and the processed sensor data. It provides a common machine understanding between embedded agents that is generic and extendable. It also includes a reasoning interface for inferring new knowledge from the observed data and guarantee knowledge stability by checking for inconsistencies. This framework improves local (machine level) and global (system level) situation awareness at all levels of service capabilities, from adaptive mission planning and autonomous target recognition to deliberative collision avoidance and escape. It acts as as an enabler for on-board decision making.

Based on their capabilities, service-oriented agents can then gain access to the different levels of information and contribute to the enrichment of the knowledge. If the required information is unavailable, the framework provides the facility to request other agents with the necessary capabilities to generate the required information, i.e. an target classification algorithm could query the correspondent agent to provide the required object detection analysis before proceeding with its classification task.

Secondly, we present an algorithm for autonomous mission adaptation. Using the knowledge made available by the semantic framework, our approach releases the operator from decision making tasks. We show how adaptation plays an important role in providing long term autonomy as it allows the platforms to react to events from the environment while at the same time requires less communication with the operator. The aim is to be effective and efficient as a plan costs time to prepare. Once the initial time has been invested preparing the initial plan, when changes occur, it might be more efficient to try to reuse previous efforts by repairing it. Also, commitments might have been made to the current plan: trajectory reported to other intelligent agents, assignment of mission plan sections to executors or assignment of resources, etc. Adapting an existing plan ensures that as few commitments as possible are invalidated. Using plan proximity metrics, we prove how similar plans are more likely to be accepted and trusted by the operator than one that is potentially completely different.

Finally, we show during a series of in-water trials how these two elements combined, a decision making algorithm and shared knowledge representation, provide the required interoperability between embedded service-oriented agents to achieve high-level mission goals, detach the operator from the routinary mission decision making and, ultimately, enable the permanent presence of dynamic sensing networks underwater.

2. Unmanned decision making loop

In this section, we describe the decision making process currently used by UUV systems and we introduce the unmanned decision loop, where observations, orientations, decisions and actions (OODA) occur in a loop enabling adaptive mission planning.

In order to describe the unmanned decision loop, we need to start by modelling the mission environment. A mission environment is defined by the tuple $\Pi = (\Sigma, \Omega)$, where:

- Σ is the mission domain model containing information about domain, i.e. the platform and the environment of execution, and
- Ω is the mission problem model containing information about the problem, i.e. mission status, requirements, and objectives.

The set of all possible mission environments for a given domain is defined as the *domain space* (e.g., the domain space of the underwater domain). It is denoted by Θ. A mission environment Π is an element of one and only one Θ.
From this model, a mission plan π that tries to accomplish the mission objectives can be produced. However, this mission environment evolves over time t as new observations of the domain model Σ_t and the problem model Ω_t continuously modify it:

$$\Pi_t \leftarrow \Pi_{t-1} \cup \Sigma_t \cup \Omega_t \tag{1}$$

The decision making process to calculate a mission plan π_t for a given mission environment Π_t occurs in a cycle of observe-orient-decide-act. This process was termed by Boyd (1992) as the OODA-loop, and it was modelled on human behaviour. Inside this loop, the *Orientation* phase contains the previously acquired knowledge and initial understanding of the situation of the mission environment (Π_{t-1}). The *Observation* phase corresponds to new perceptions of the mission domain model (Σ_t) and the mission problem model (Ω_t) that modify the mission environment. The *Decision* component represents the level of comprehension and projection, the central mechanism enabling adaptation before closing the loop with the *Action* stage.
Note that it is possible to make decisions by looking only at orientation inputs without making any use of observations. In this case, Eq. 1 becomes $\Pi_t \leftarrow \Pi_{t-1}$. In the same way, it is also possible to make decisions by looking only at the observation inputs without making use of available prior knowledge. In this case, Eq. 1 becomes $\Pi_t \leftarrow \Sigma_t \cup \Omega_t$.
In current UUVs implementations, the human operator constitutes the decision phase. See Figure 1 for a schematic representation of the control loop. When high bandwidth communication links exist, the operator remains in the OODA-loop during the mission execution taking the decisions. For each update of the mission environment Π_t received, the operator decides on the correspondent mission plan π_t to be performed. From the list of actions in this mission plan, the mission executive issues the correspondent commands to the platform. Examples of the implementation of this architecture are existing Remotely Operated Vehicles (ROVs).
However, when communication is unreliable or unavailable, the operator must attempt to include all possible if-then-else cases to cope with execution alternatives before the mission starts. This is the case of current UUVs implementations that follow an orientation-only model. Figure 2 shows this model, where the OODA-loop is broken because observations are not reported to the human operator.
We will now discuss a few recent UUV implementations which show where the state-of-the-art is currently positioned. Most implementations rely on pre-scripted mission

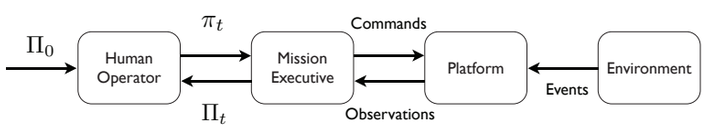


Fig. 1. Observation, Orientation, Decision and Action (OODA) loop for unmanned vehicle systems with decision making provided by the human operator.

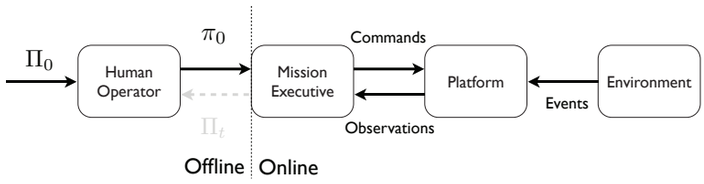


Fig. 2. Broken OODA-loop. Decision stage on the human operator based only on initial pre-mission orientation.

plan managers that are procedural and static and might not even consider conditional executions (Hagen, 2001). At this level, the mission executive follows a sequence of basic command primitives and issues them to the functional control layer of the platform. Description about how these approaches maintain control of underwater vehicles can be found in Fossen (1994), Ridao et al. (1999) and Yuh (2000). In this situation, decisions taken by the operator are made using only orientation inputs related to some previous experience and *a-priori* knowledge. This has unpredictable consequences, in which unexpected situations can cause the mission to abort and might even cause the loss of the vehicle (Griffiths, 2005; von Alt, 2010).

More modern approaches are able to mitigate this lack of adaptability by introducing sets of behaviours that are activated based on observations (Arkin, 1998). Behaviours divide the control system into a parallel set of competence-levels. They can be seen as manually scripted plans generated *a-priori* to encapsulate the decision loop for an individual task. Under this approach, the key factor is to find the right method for coordinating these competing behaviours.

The subsumption model, attributed to Brooks (1986), arbitrates behaviour priorities through the use of inhibition (one signal inhibits another) and suppression (one signal replaces other) networks. Most recent UUV control systems are a variant of the subsumption architecture.

This model was first applied to the control of UUVs by Turner (1995) during the development of the ORCA system. This system used a set of schemas in a case-based framework. However, its scalability remains unclear as trials for its validation were not conducted.

Later, Oliveira et al. (1998) developed and deployed the CORAL system based on Petri nets. The system was in charge of activating the vehicle primitives needed to carry out the mission. These primitives were chained by preconditions and effects.

The scaling problem was addressed by Bennet & Leonard (2000) using a layered control architecture. Layered control is a variant of the subsumption model that restricts of interaction between layers in order to keep it simple (Bellingham et al., 1990). The system was deployed for the application of adaptive feature mapping.

Another approach for coordinating behaviours is vector summation that averages the action between multiple behaviours. Following this principle, the DAMN system developed by Rosenblatt et al. (2002) used a voting-based coordination mechanism for arbitration implementing utility fusion with fuzzy logic.

The MOOS architecture developed by Newman (2002) was also able to guide UUVs by using a mission control system called Helm. Helm's mission plan was described by a set of prioritised primitive tasks. The most suitable action was selected using a set of prioritised mission goals. It used a state-machine for execution, a simplified version of a Petri net.

The O^2CA^2 system (Carreras et al., 2007) also used a Petri net representation of the mission plan (Palomeras et al., 2009). The system maintains the low level control (dynamics) from the

guidance control (kinematics) uncoupled (Caccia & Veruggio, 2000). Although it contained a declarative mission representation, missions were programmed manually.
A detailed survey of other behaviour-based approaches applied to mission control systems for UUVs can be found in Carreras et al. (2006).
More recently, Benjamin et al. (2009) has applied multiple objective decision theory to provide a suitable framework for formulating behaviour-based controllers that generate Pareto-optimal and satisfying behaviours. This approach was motivated by the infeasibility of optimal behaviour selection for real-world applications. This approach has been implemented and deployed as part of the IvP Helm extension to MOOS. This method seems to be a more suitable for behaviour selection, although more computationally expensive. Also, the approach is limited to the control of only the direction and velocity parameters of the host platform.
After reviewing this related work, two problems affecting the effectiveness of the decision loop become evident. Firstly, orientation and observation should be linked together because it is desirable to place the new observations in context. Secondly, decision and action should be iterating continuously. These two problems have not been addressed together by previous approaches. These are the two of the goals that we address in this chapter. In order to achieve them autonomously, two additional components are required: a status monitor and a mission plan adapter. The status monitor reports any changes detected in the mission environment during the execution of a mission. When the mission executive is unable to handle the changes detected by the status monitor, the mission planner is called to generate a new modified mission plan that agrees with the updated mission environment. Figure 3 shows the OODA-loop for autonomous decision making. Comparing it to the previous Figure 2, the addition of status monitor and mission planner removes the need for human decisions in the loop. Note that the original mission plan π_0 could also be autonomously generated as long as the high-level goals are provided by the human operator in Π_0.

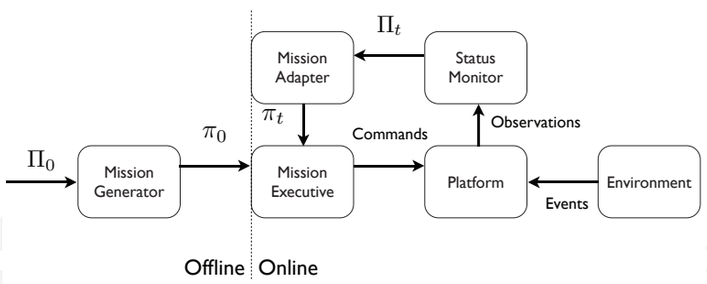


Fig. 3. Required OODA-loop for autonomous decision making in UUVs. Decision stage for adaptation takes place on-board based on initial orientation provided by the operator and observations provided by the status monitor.

Adaptive mission planning enables a true unmanned OODA-loop. This autonomous decision making loop copes with condition changes in the mission environment during the mission execution. As a consequence, it releases the operator from decision making tasks in stressful environments containing high levels of uncertainty and dynamism.
The potential benefits of adaptive mission planning capabilities for autonomous decision making in UUVs were promoted by Turner (2005), Bellingham et al. (2006) and Patrón & Petillot (2008). Possibly the most advanced autonomous decision making framework for UUVs has been developed at the Monterey Bay Aquarium Research Institute. This architecture, known as *T-REX*, has been deployed successfully inside the Dorado AUV (Rajan

et al., 2009). This is now providing adaptive planning capabilities to oceanographers for maximising the science return of their UUV missions (McGann et al., 2007; Rajan et al., 2007). Using deliberative reactors for the concurrent integration of execution and planning (McGann, Py, Rajan & Henthorn, 2008), live sensor data can be analysed during mission to adapt the control of the platform in order to measure dynamic and episodic phenomenon, such as chemical plumes (McGann, Py, Rajan, Henthorn & McEwen, 2008; McGann et al., 2009). Alternative approaches to adaptive plume tracing can also be found in the works of Farrell et al. (2005) and Jakuba (2007). Their research goals of all these approaches have been motivated by scientific applications and do not consider the needs of the human operators or the maritime industry.

However, autonomy cannot be achieved without humans, as it is necessary for this autonomy to be ultimately accepted by an operator. Our research is geared towards improving human access to UUVs in order to solve the maritime industry's primary requirement of improving platform operability (Patrón et al., 2007). We propose a goal-based approach to solving adaptive mission planning. The advantage of this approach is that it provides high levels of mission abstraction. This makes the human interface simple, powerful and platform independent, which greatly eases the operator's task of designing and deploying missions (Patrón, 2009). Ultimately, operators will not need any specialist training for an UUV specific platform, and instead missions will be described purely in terms of their goals. Apart from ease of use, we have also demonstrated using a novel metric (Patrón & Birch, 2009) that adaptive mission planners can produce solutions which are close to what a human planner would produce (Patrón et al., 2009a). This means that our solutions can be trusted by an operator.

Another advantage of our research over other state-of-the-art UUV implementations, is that we are industry focussed. Our service-oriented approach provides goal-based mission planning with discoverable capabilities, which meets industry's need for platform independence (Patrón et al., 2009b). Finally, our plan repair approach optimises the resources required for adaptability and maximises consistency with the original plan, which improves human acceptance of autonomy. Resource optimisation and consistency are very important properties for real world implementations, as we demonstrate in our sea trials (Patrón, Miguelanez, Petillot & Lane, 2008).

Section 3 describes how do we link together orientation and observation. Section 4 presents an approach to the continuous iteration of decision and action.

3. Semantic knowledge-based situation awareness

Unmanned vehicle situation awareness SA_V consists in enabling the vehicle to autonomously understand the '*big picture*' (Adams, 2007). This picture is composed of the experience gained from previous missions (orientation) and the information obtained from the sensors while on mission (observation). Ontologies allow the representation of knowledge of these two components.

Ontologies are models of entities and interactions, either generically or in some particular practice of knowledge (Gruber, 1995). The main components of an ontology are concepts and axioms. A concept represents a set or class of entities within a domain (e.g., a *fault* is a concept within the domain of diagnostics). Axioms are used to constrain the range and domain of the concepts (e.g., a *driver* is a *software* that has a *hardware*). The finite set of concept and axiom definitions is called the *Terminology Box TBox* of the ontology.

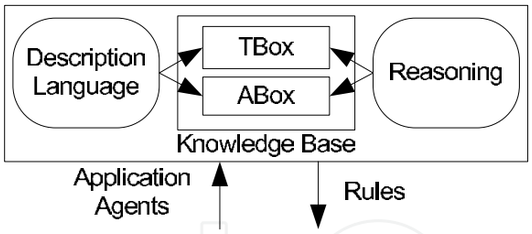


Fig. 4. Knowledge Base representation system including the $TBox$, $ABox$, the description language and the reasoning components. Its interface is made of orientation rules and agent queries.

Instances are the individual entities represented by a concept of the ontology (e.g. a *remus* is an instance of the concept *UUV*). Relations are used to describe the interactions between individuals (e.g. the relation *isComponentOf* might link the individual *SensorX* to the individual *PlatformY*). This finite set of instances and relations about individuals is called the *Assertion Box* $ABox$. The combination of $TBox$ and $ABox$ is what is known as a *Knowledge Base*. $TBox$ aligns naturally to the orientation component of SA_V while $ABox$ aligns to the observation component.

In the past, authors such as Matheus et al. (2003) and Kokar et al. (2009) have used ontologies for situation awareness in order to assist humans during information fusion and situation analysis processes. Our work extends these previous works by using ontologies for providing unmanned situation awareness in order to assist autonomous decision making algorithms in underwater vehicles. One of the main advantages of using a knowledge base over a classical data base schema to represent SA_V is the extended querying that it provides, even across heterogeneous data systems. The meta-knowledge within an ontology can assist an intelligent agent (e.g., status monitor, mission planner, etc.) with processing a query. Part of this intelligent processing is due to the capability of reasoning. This enables the publication of machine understandable meta-data, opening opportunities for automated information processing and analysis.

For instance, a status monitor agent using meta-data about sensor location could automatically infer the location of an event based on observations from nearby sensors (Miguelanez et al., 2008). Inferences over the ontology are made by reasoners. A reasoner 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. 4). A detailed description of how a reasoner works is outside of the scope of this article. For the implementation of our approach, we use the open source reasoner called Pellet (Sirin et al., 2007).

3.1 Semantic knowledge-based framework

A library of knowledge bases comprise the overall knowledge framework used in our approach for building SA_V (Miguelanez et al., 2010; Patrón, Miguelanez, Cartwright & Petillot, 2008). Reasoning capabilities allow concept consistency providing reassurance that SA_V 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. In order to provide with a design that supported maximum reusability (Gruber, 1995; van Heijst et al., 1996), we adopt a three-level segmentation structure that includes the (1) Foundation, (2) Core and (3) Application ontology levels (see Fig. 5).

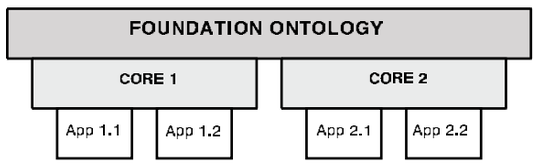


Fig. 5. Levels of generality of the library of knowledge bases for SA_V. They include the Foundation Ontology, the Core Ontology, and the Application Ontology levels.

Foundational Ontologies (FOs) represents the very basic principles and includes Upper and Utility Ontologies. Upper ontologies describe generic concepts (e.g., the Suggested Upper Merged Ontology or SUMO (Niles & Pease, 2001)) while Utility ontologies describe support concepts or properties (e.g. OGC_GML for describing geospatial information (Portele, 2007)). FOs meet the requirement that a model should have as much generality as possible, to ensure reusability across different domains.
The Core Ontology provides a global and extensible model into which data originating from distinct sources can be mapped and integrated. This layer provides a single knowledge base for cross-domain agents 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 ontologies.
In the bottom layer, an Application Ontology provides an underlying formal model for agents that integrate source data and perform a variety of extended functions. As such, higher levels of complexity are tolerable and the design is motivated more by completeness and logical correctness than human comprehension. Target areas of these Application Ontologies are found in the status monitoring of the vehicle and its environment and the planning of the mission.
Figure 6 represents the relationship between the Foundation Ontologies (Upper and Utility), the Core Ontology and the Application Ontology for each service-oriented agent. Raw data gets parsed from sensors into assertions during the mission using a series of adapter modules for each of the sensing capabilities. It also shows that the knowledge handling by the agent during its decision making process is helped by the reasoner and the rule engine process.

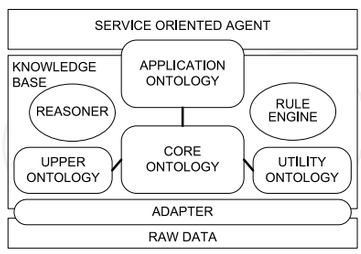


Fig. 6. SA_V representation in the Knowledge Base using Core and Application ontologies supported by Upper and Utility ontologies. Generation of instances from raw data is performed by the Adapter. Handling of knowledge is done by the Reasoner, Rule Engine and the Service-Oriented Agent.

3.2 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) (SAE, 2008a). This

standard was originally developed for the Unmanned Ground Vehicles (UGVs) environment only but has recently been extended to all other environments, such as air and water, trying to provide a common set of architecture elements and concepts. The JAUS model classifies four different sets of Knowledge Stores: Status, World map, Library and Log. Our experience has shown that overlap exists 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 still providing JAUS 'Message Interoperability' (SAE, 2008b) between agents.
Within the proposed framework, JAUS concepts are considered as the Foundation Ontology 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,
- Agent: 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,

Additionally, the Standard Ontology for Ubiquitous and Pervasive Applications (SOUPA) (Chen et al., 2004) is used as an Utility Ontology. By providing generic context-aware concepts, it enables the spatio-temporal representation of concepts in the Core Ontology.

3.3 Application Ontology

Each service-oriented agent has its own Application Ontology. It represents the agent's awareness of the situation by including concepts that are specific to the expertise of the agent. In the case study presented in this chapter, these agents are the status monitor and the mission planner. Together, they provide the *status monitor* and *mission adapter* components described in Fig. 3 required for closing the OODA-loop and provide on-board decision making adaptation.

3.3.1 Status Monitoring Application Ontology

The Status Monitoring Application Ontology is used to express the SA_V of the status monitor agent. To model the behaviour of all components and subsystems considering from sensor data to possible model outputs, the Status Monitoring Application Ontology is designed and built based on ontology design patterns (Blomqvist & Sandkuhl, 2005). Ontology patterns facilitate the construction of the ontology and promote re-use and consistency if it is applied to different environments. In this work, the representation of the monitoring concepts are based on a system observation design pattern. Some of the most important concepts identified for status monitoring are:

- Data: all internal and external variables (gain levels, water current speed),
- Observation: patterns of data (sequences, outliers, residuals,...),
- Symptom: individuals related to interesting patterns of observations (e.g., low gain levels, high average speed),

- Event: represents a series of correlated symptoms (low power consumption, position drift), Two subclasses of Events are defined: *CriticalEvent* for high priority events and IncipientEvent for the remaining ones.
- Status: links the latest and most updated event information to the systems being monitored (e.g. sidescan transducer),

Please note how some of these concepts are related to concepts of the Core Ontology (e.g. an *observation* comes from a *sensor*). These Core Ontology elements are the enablers for the knowledge exchange between service-oriented agents. This will be shown in the demonstration scenario of Section 5.3.

3.3.2 Planning Application Ontology

The Plan Application Ontology is used to express the SA_V of the mission planner agent. It uses concepts originally defined by the *Planning Domain Definition Language* (PDDL). The PDDL language was originally created by Ghallab et al. (1998) to standardise plan representation. Concepts are extracted from the language vocabulary and the language grammar is used for describing the relationships and constraints between these concepts.
For the adaptation mission planning process, the Planning Application Ontology also requires concepts capable of representing the diagnosis of incidents or problems occurring in some parts of the mission plan (van der Krogt, 2005). 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, etc.),
- Action: Modification of the state of resources (calibrate, classify, explore, etc.),
- Gap: A non-executable action,
- Execution: When an action is executed successfully,
- Failure: An unsuccessful execution of an action,

Please note how some of these concepts are also related to concepts of the Core Ontology (e.g. a list of *capability* concepts is required to perform a mission *action*).

4. Adaptive mission planning

The adaptive mission planning process involves the detection of events, the effects that these events have on the mission plan and the response phase. The detection of events is performed by the status monitoring agent. The mission plan diagnosis and repair is undertaken by the adaptive mission planner agent.

4.1 Status Monitor Agent

The Status Monitor Agent considers all symptoms and observations from environmental and internal data in order to identify and classify events according to their priority and their nature (critical or incipient). Based on internal events and context information, this agent is able to infer new knowledge about the current condition of the vehicle with regard to the availability for operation of its components (i.e. status). In a similar way, environmental data is also considered for detecting and classifying external events in order to keep the situation awareness of the vehicle updated.

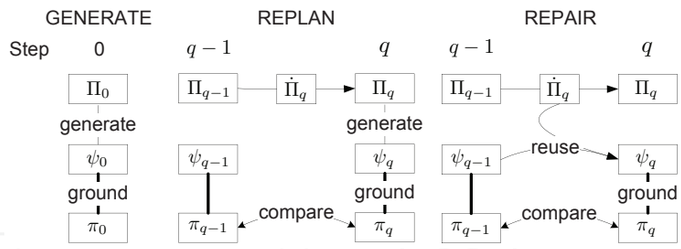


Fig. 7. Schematic representation of the autonomous mission generation, replan and repair processes using partial plan representation of the mission plans.

4.2 Mission plan adaptation agent

This section describes the mission environment model used by the mission plan repair techniques presented in this chapter. We continue using the mission environment model previously described. We generalise this model to any step q in the mission execution timeline. An instance of an UUV mission environment at a given step q can be simply defined as $\Pi_q = \{\Sigma_q, \Omega_q\}$. The mission domain model Σ_q contains the set of propositions defining the available resources in the system P_V and the set of actions or capabilities A_V. The mission problem model Ω_q contains the current platform state x_q and the mission requirements Q_O.
Based on this model, we analyse how to calculate mission plans on the plan space (Sacerdoti, 1975). A plan space is an implicit directed graph whose vertices are partially specified plans and whose edges correspond to refinement operations. In a real environment where optimality can be sacrificed by operability, partial plans are seen as a suitable representation because they are a flexible constrained-based structure capable of being adapted.
A partial plan ψ is a tuple containing a set of partially instantiated actions and a set of constraints over these partially grounded actions. Constraints can be of the form of ordering constraints, interval preservation constraints, point truth constraints and binding constraints. Ordering constraints indicate the ordering in which the actions should be executed. Interval preservation constraints link preconditions and effects over actions already ordered. Point truth constraints assure the existence of precondition facts at certain points of the plan. Binding constraints on the variables of actions are used to ground the actions to variables of the domain. Figure 8 shows a partial plan representation of an UUV mission. Partial plans are flexible to modification. They provide an open approach for handling extensions such as temporal and resource constraints. Due to nature of the constraints, it is easy to explain a partial plan to a user. Additionally, it is easily extensible to distributed multi-agent mission planning.
Figure 7 explains the processes of mission plan repair and mission plan replan for mission plan adaptation for UUVs using a partial plan representation of the mission plans. At the initial step, a partial ordered plan ψ_0 is generated satisfying the original mission environment Π_0. The ψ_0 is then grounded into the minimal mission plan π_0 including all constraints in ψ_0. At step q, the semantic knowledge-based framework is updated by the diagnosis information $\dot{\Pi}_q$ providing a modified awareness of the mission environment Π_q. From here, two mission adaptation processes are possible: *Mission replan* generates a new partial plan ψ_q, as done at the first stage, based only on the knowledge of Π_q. On the other hand, *mission plan repair* re-validates the original plan by ensuring minimal perturbation of it. Given the partial plan at the previous step ψ_{q-1} and the diagnosis information $\dot{\Pi}_q$, the mission repair problem produces a solution partial plan ψ_q that satisfies the updated mission problem Π_q, by modifying ψ_{q-1}. The final step for both approaches is to ground ψ_q to its minimal mission plan

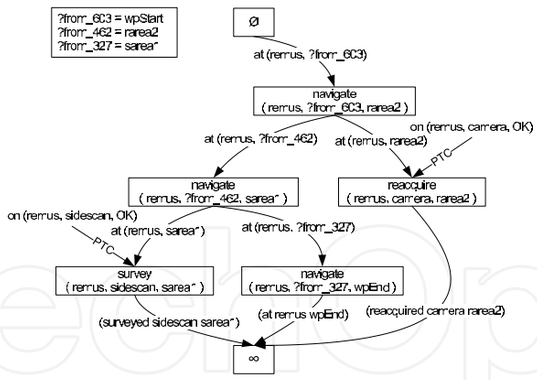


Fig. 8. Example of a partial ordered plan representation of an autonomously generated UUV mission. The ordering constraints are represented using the graph depth, interval preservation constraints are represented with black arrows, point truth constraints are represented with PTC-labelled arrows, and binding constraints are shown in the top left box.

π_q. It can be seen that mission repair better exploits the orientation capabilities for decision making: instead of taking the new mission environment as a given, it uses the diagnosis information about the changes occurred to guide the adaptation process.

We have now identified the benefits of mission plan repair over mission replan. Mission plan repair modifies the partial plan ψ_q, so that it uses a different composition, though it still maintains some of the actions and the constraints between actions from the previous partial plan. However, mission plan adaptation can also be achieved by mission execution repair by looking directly at the mission plan instantiation π_q. Execution repair modifies the instantiation of the mission plan π_q such that a ground action $g_q^{a_h}$ that was previously instantiated by some execution e_q is newly bound by another action execution instance e'_q.

Executive repair is less expensive and it is expected to be handled directly by the mission executive agent. Plan repair, however, is computationally more expensive and requires action of the mission planner agent.

The objective is to maximise the number of execution repairs over plan repairs and, at the plan repair level, maximise the number of decisions reused from the previous mission instantiation. The information provided by the semantic-base knowledge base during the plan diagnosis phase is critical.

Executive repair fixes plan failures identified in the mission plan during the diagnosis stage. Our approach uses ontology reasoning in combination with an action execution template to adapt the mission plan at the executive level.

Once a mission plan π_q is calculated by the mission planner, its list of ground actions is transferred to the executive layer. In this layer, each ground action $g_q^{a_h}$ of π_q gets instantiated into an action execution instance e_q^t using the action template for the action a_h available in the Core Ontology of the knowledge base. At the end of this phase, each e_q^t contains the script of commands required to perform its correspondent ground action. Flexibility in the execution of an action instance is critical in real environments. This is provided by a timer, an execution counter, a time-out register and a register of the maximum number of executions in the action execution instance. Additionally, three different outputs control the success, failure or time-out of its execution. These elements handle the uncertainty during the execution phase and enable the executive repair process. This minimise the number of calls to the adaptive mission planner agent and therefore the response time for adaptation.

Plan repair uses a strategy to repair with new partial plans the plan gaps identified during the plan diagnosis stage. Our approach uses an iteration of unrefinement and refinement strategies on a partial-ordered planning framework to adapt the mission plan.
Planning in the plan space is slower than in the state space because the nodes are more complex. Refinement operations are intended to achieve an open goal from the list of mission requirements or to remove a possible inconsistency in the current partial plan. These techniques are based on the least commitment principle, and they avoid adding to the partial plan any constraint that is not strictly needed. A refinement operation consists of one or more of the following steps: adding an action, an ordering constraint, a variable binding constraint or a causal link.
A partial plan is a solution to the planning problem if has no flaw and if the sets of constraints are consistent. Flaws are either subgoals or threats. Subgoals are open preconditions of actions that have not been linked to the effects of previous actions. Threats are actions that could introduce inconsistencies with other actions or constraints. We implemented a recursive non-deterministic approach based on the Partial ordered Planning (PoP) framework (Penberthy & Weld, 1992). This framework is sound, complete, and systematic. Unlike other Plan space planners that handle both types of flaws (goals and threats) similarly, each PoP recursive step first refines a subgoal and then the associated threats (Ghallab et al., 2004).
In our implementation, we introduce a previous step capable of performing an unrefinement of the partial plan when necessary. During the unrefinement strategy we remove refinements from the partial plan that are reported by the plan diagnosis phase to be affecting the consistency of the mission plan with the mission environment, i.e. to remove constraints and finally the actions if necessary.
In simple terms, when changes on the $ABox$ Planning Application Ontology are sensed ($\dot{\Pi}_q$) that affect the consistency of the current partial plan ψ_{q-1}, the plan repair process is initiated. The plan repair stage starts an unrefinement process that relaxes the constraints in the partial plan ψ_{q-1} that are causing the mission plan to fail.
The remaining temporal mission partial plan ψ'_{q-1} is now relaxed to be able to cope with the new mission environment. However, this relaxation could open some subgoals and introduce threats in the partial plan that need to be addressed. The plan repair stage then executes a refinement process searching for a new mission plan ψ_q that is consistent with the new mission environment Π_q and removing these possible flaws. By doing this, it can be seen that the new mission plan ψ_q is not generated again from Π_q (re-planned) but recycled from ψ_{q-1} (repaired). This allows re-use of the parts of the plan ψ_{q-1} that were still consistent with Π_q.

5. Results

5.1 Architecture

The combination of the status monitor agent, the adaptive mission planner, the mission executive and the semantic knowledge-based framework is termed as the Semantic-based Adaptive Mission Planning system (SAMP). The SAMP system implements the four stages of the OODA-loop. Figure 9 represents the customised version of Figure 3 for SAMP.
The status monitor agent reports to the knowledge base the different changes occurring in the environment and the modifications of the internal status of the platform. The knowledge base stores the ontology-based knowledge containing the expert orientation provided *a priori* and the observations reported by the status monitor. A mission planner agent generates and

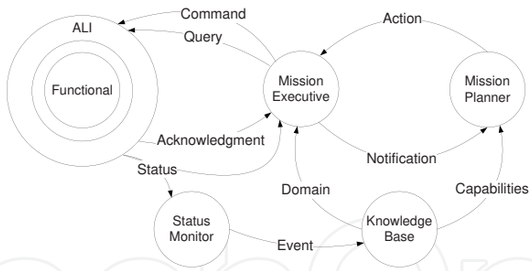


Fig. 9. Architecture of the SAMP system. The embedded agents are the planner, executive, monitor, and knowledge base. These agents interconnect via set of messages. The system integrates to the functional layer of a generic host platform by an abstract layer interface (ALI).

adapts mission plans based on the situation awareness stored in the knowledge base. The mission executive agent executes mission commands in the functional layer based on the sequence of ground actions received from the mission planner. An Abstract Layer Interface (ALI) based on JAUS-like messages (SAE, 2008a) over UDP/IP packages implemented using the OceanSHELL protocol (Oce, 2005) provides independence from the platform's functional layer making the system generic and platform independent.

5.2 Simulation results

A set of synthetic simulated scenarios have been implemented to test the performance of the SAMP system. The tests are based on the mine counter measure (MCM) operation, where UUVs support and provide solutions for mine-hunting and neutralisation.

A set of 15 selected MCM scenarios were simulated covering the variability of missions described by the concepts of operations for unmanned underwater vehicles presented in the UUV (2004) and the JRP (2005). For each scenario, the detection of a failure in one of the components of the system was simulated. The mission plan was adapted to the new constraints using replanning methods and the mission plan repair approach based on partial plans introduced in Section 4.

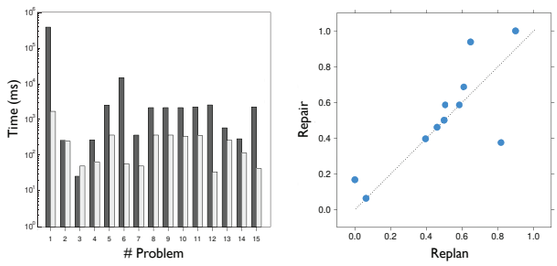


Fig. 10. Left: A semi-log plot displaying the computational time in miliseconds for replan (dark grey bars) and repair approaches (light grey bars). Right: Comparison of Plan Proximity ($PP_{0.5}$) of the replan and repair approaches to the original plan.

The performance of the two approaches was compared by looking at the computation time and the Plan Proximity (Patrón & Birch, 2009) of the adaptive mission plan provided to the original reference mission plan. Figure 10 left shows the computation time in milliseconds required for adapting the mission to the new constraints for replan (dark grey bars) and

repair approaches (light grey bars). Note that a logarithmic scale is used for the time values. Figure 10 right displays the Plan Proximity to the original plan of the replan strategy result versus the repair strategy result. It can be seen that plans adapted using the mission repair strategy tend to be closer to the original plan than using the mission replan strategy. In these results, 14 out of 15 scenarios were computed faster by using mission plan repair. This computation was on average 9.1x times faster. Also, 14 out of 15 scenarios showed that mission plan repair had greater or equal Plan Proximity values as compared to mission replan. In general, our mission repair approach improves performance and time response while at the same time finds a solution that is closer to the original mission plan available before adaptation.

5.3 Experimental results

This section shows the performance of the SAMP system inside the real MCM application using a REMUS 100 UUV platform (see Fig. 11.left) in a set of integrated in-water field trial demonstration days at Loch Earn, Scotland (56^o23.1N,4^o12.0W). The REMUS UUV had a resident guest PC/104 1.4GHz payload computer where the SAMP system was installed. SAMP was capable of communicating with the vehicle's control and status modules and taking control of it by using an interface module that translated the ALI protocol into the manufacturer's *Remote Control* protocol (REM, 2008).

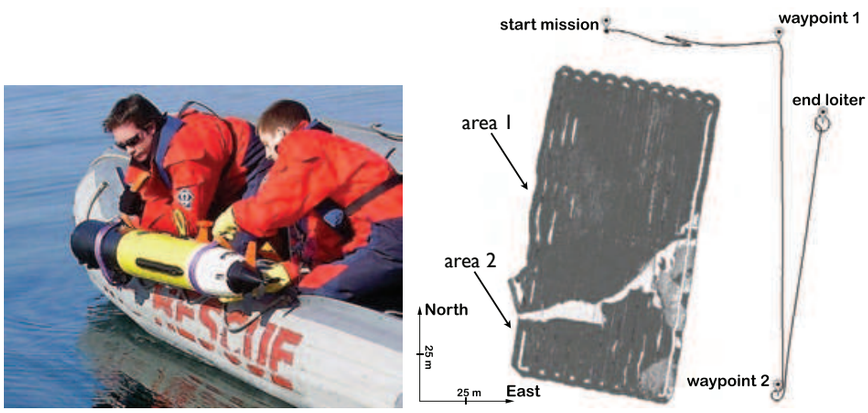


Fig. 11. Left: REMUS UUV deployment before starting one of its missions. Right: Procedural mission uploaded to the vehicle control module and *a priori* seabed classification information stored in the knowledge base. The two dark grey areas correspond to the classified seabed regions.

Figure 11.right shows the procedural waypoint-based mission as it was described to the vehicle's control module. This was known as the baseline mission. It was only used to start the vehicle's control module with a mission in the area of operation before taking control of it using the SAMP system. The baseline mission plan consisted on a start waypoint and two waypoints describing a North to South mission leg at an approximate constant Longitude (4^o16.2W). This leg was approximately 250 meters long and it was followed by a loiter pattern at the recovery location. The track obtained after executing this baseline mission using the vehicle control module is shown in Fig. 11 with a dark line. A small adjustment of the vehicle's location can be observed on the top trajectory after the aided navigation module corrects its solution to the fixes received from the Long Baseline (LBL) transponders previously deployed in the area of operations.

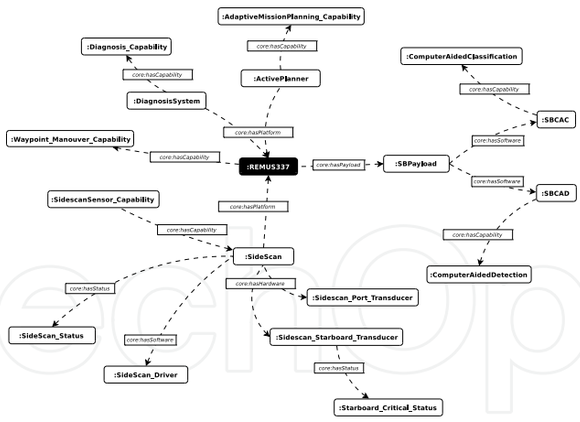


Fig. 12. Core Ontology instances for the demonstration scenario. The diagram represents the main platform, its components and their capabilities.

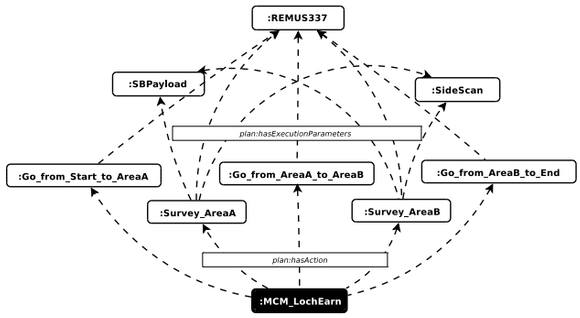


Fig. 13. Plan Application Ontology concepts representing the mission planning actions and their execution parameters and relationships.

On the payload side, the SAMP system was oriented (in the OODA-loop sense) using *a priori* information about the environment and the platform and a declarative description of the goals of the mission. The *a priori* knowledge and the platform configuration capabilities was represented using Core Ontology concepts (see Fig. 12). Knowledge about the environment was provided based on automatic computer-aided seabed classification information generated from previous existent data (Reed et al., 2006). The two classified seabed areas are shown in Fig. 11. The declarative description of the mission requirements was represented using concepts from the Planning Application Ontology. They could be resumed as 'survey all known areas maximizing efficiency'.

5.3.1 Pre-mission reasoning

Please note that the previously described separation between Core knowledge and Planning knowledge gracefully aligns with the separation between platform engineers and mission scientists on current UUV operations. If the platform capabilities were described in Core Ontology terms by the engineers that manufactured the platform, it can be seen how, by using the SAMP approach, a scientific operator that only cares about the data should be able to describe the mission to the platform without knowing anything about the custom properties of the platform. It is, therefore, important to assist the operator in knowing if the platform capabilities can match the mission requirements before starting the mission:

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

In order to answer this question, new knowledge could be inferred from the initial Core Ontology orientation. The Core Ontology rule engine was executed providing with additional knowledge. A set of predefined rules helped orienting 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. 2.

$$\begin{array}{c} core : isCapabilityOf(?Capability, ?Payload) \wedge \\ core : isPayloadOf(?Payload, ?Platform) \\ \rightarrow core : isCapabilityOf(?Capability, ?Platform) \end{array} \quad (2)$$

Once all the possible knowledge was extracted, it was possible to query the knowledge base in order to extract the list of capabilities of the platform (see Eq. 3) and the list of requirements of the mission (see Eq. 4).

$$\begin{array}{l} SELECT\ ?Platform\ ?Cap\ WHERE\ \{\ rdf{:}type(\ ?Platform,\ core{:}Platform) \wedge \\ \qquad\qquad\qquad\qquad\qquad\qquad core{:}hasCapability(?Platform,?Cap)\ \} \end{array} \quad (3)$$

$$\begin{array}{l} SELECT\ ?Mission\ ?Req\ WHERE\ \{\ plan{:}hasAction(\ ?Mission,\ ?Action) \wedge \\ \qquad\qquad\qquad\qquad\qquad\qquad plan{:}hasRequirement(\ ?Action,?Req\ )\ \} \end{array} \quad (4)$$

This way, it was possible to autonomously extract that the requirements of the mission of the experiment were[1]:

- *core:WaypointManeuver_Capability* $\in$ *jaus:Maneuver_Capability*
- *core:ComputerAidedClassification_Capability* $\in$ *jaus:Autonomous_RSTA-I_Capability*
- *core:ComputerAidedDetection_Capability* $\in$ *jaus:Autonomous_RSTA-I_Capability*
- *core:SidescanSensor_Capability* $\in$ *jaus:Environmental_Sensing_Capability*

which were a subset of the platform capabilities. Therefore, for this particular case, the platform configuration suited the mission requirements.

5.3.2 In mission adaptation

For these experiments, SAMP was given a static location in which to take control of the host vehicle. At this point, the mission planner agent generated a mission plan based on the knowledge available and the mission requirements. The instantiation of this mission plan is described in Fig. 13 using Planning Application Ontology concepts. The mission was then passed to the executive agent that took control of the vehicle for its execution.
While the mission was executed the status monitor agent maintained the knowledge base updated (in the OODA-loop sense) by reporting changes in the status of hardware components, such as batteries and sensors, and external parameters, such as water currents.
When observations indicated that some of these changes were affecting the mission under execution, the mission planner was activated in order to adapt the mission to the changes. This indication was detected by the planner agent by querying the knowledge base with the following question:

[1] RSTA-I i.e., Reconnaissance/Surveillance/Target Acquisition & Identification capability concepts inherited from JAUS (SAE, 2006)

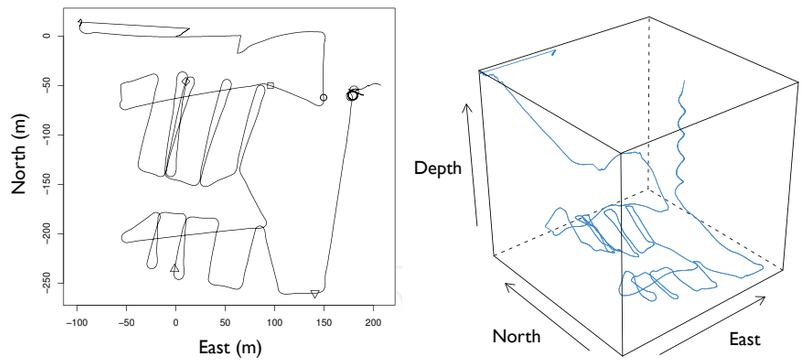


Fig. 14. Left: Vehicle's track during mission in a North-East coordinate frame projection with the origin at the starting point of the mission. Right: Three-dimensional display of the vehicle's track during the mission (Note that depth coordinates are not to scale).

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

In order to explain the reasoning process involved during the event detection, diagnosis and response phases of the mission adaptation process, a component fault as an internal event was temporarily simulated in the host vehicle. The fault simulated the gains of the starboard transducer of the sidescan sonar dropping to their minimum levels half way through the lawn mower survey of the first area.
For the detection phase, the low gain signals from the transducer triggered a symptom instance, which had an associated event level. This event level, represented in the Status Monitoring Application Ontology using a value partition pattern, plays a key role in the classification of the instances in the *Event* concept between being critical or incipient. This classification is represented axiomatically in the Eqs. 5 and 6.

$$\begin{aligned} &\text{status:CriticalEvent} \sqsubseteq \text{status:Event} \sqcap \ni \text{status:causedBySymptom} \ldots \\ &(\text{status:Symptom} \sqcap \ni \text{status:hasEventLevel} \ldots \\ &(\text{status:Level} \sqcap \ni \text{status:High})) \end{aligned} \tag{5}$$

$$\begin{aligned} &\text{status:IncipientEvent} \sqsubseteq \ldots \\ &\text{status:Event} \sqcap \ni \text{status:causedBySymptom} \ldots \\ &(\text{status:Symptom} \sqcap \ni \text{status:hasEventLevel} \ldots \\ &(\text{status:Level} \sqcap \ni \text{status:Med})) \end{aligned} \tag{6}$$

After the *Event* individuals were re-classified, the *Status* property of the related component in the Core Ontology was updated.
During the diagnosis event phase, a critical status of a component is only considered to be caused by a critical event. Therefore, due to the fact that the sidescan sonar component is composed of two transducers, port and starboard, one malfunctioned transducer was only diagnosed as an incipient *Status* of the overall sidescan sonar component.
During the response phase, the *Status* property of the Core Ontology components were used by the mission planner to perform the plan diagnosis of the mission under execution. The query to the knowledge base shown in Eq. 7 reported that the two survey actions in the mission plan were affected by the incipient status of the sidescan sonar.

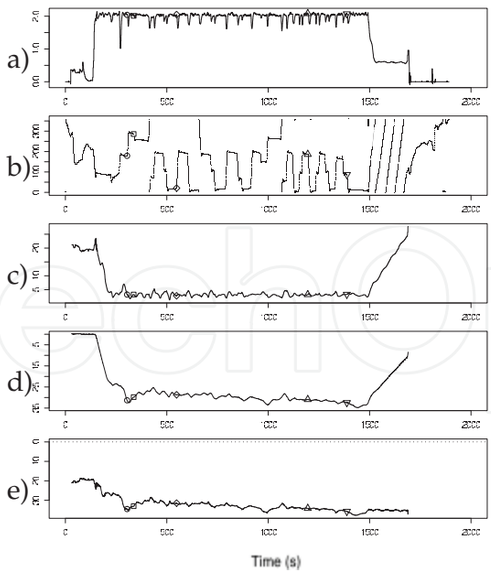


Fig. 15. Vehicle telemetry (top to bottom): a) vehicle velocity (m/s), b) compass heading (degrees), c) altitude (m), d) depth (m) and e) reconstructed profile of the seabed bathymetry (m) during the mission, all plotted against mission time (s).

$$\begin{aligned} &\textit{SELECT} \quad \textit{?Mission ?Action ?Param ?Status} \\ &\textit{WHERE} \ \{ \ \textit{plan:hasAction(?Mission, ?Action)} \wedge \\ &\qquad\qquad \textit{plan:hasExecParam(?Action,?Param)} \wedge \\ &\qquad\qquad \textit{plan:hasStatus(?Param, ?Status)} \ \} \end{aligned} \tag{7}$$

An incipient *Status* of the action parameters indicates that the action can still be performed by adapting the way it is being executed, an execution repair. If both transducers were down, a critical status of the sidescan sensor is diagnosed and a plan repair adaptation of the mission plan would have been required instead. In that case, the adaptive mission planner would have looked for redundant components or similar capabilities to perform the action or to drop the action from the plan.

The same procedure was used after the transducer recovery was reported to adapt the survey action to the normal pattern during the second lawn mower survey. In a similar process, SAMP adapted the lawnmower pattern survey of the areas to the detected water current *Status* at the moment of initialising the survey of the areas.

The timeline of the mission executed using the SAMP approach is described in the following figures: Figure 14 represents the final trajectory of the vehicle in 2D and 3D using a North-East coordinate frame projection with the origin at the starting point of the mission. Figure 15 displays the vehicle's telemetry recorded during the mission. It includes vehicle's velocity, compass heading, altitude, depth measurements and processed bathymetry over time. Figure 16 shows subset of the variables being monitored by the status monitor agent that were relevant to this experiment. These variables include direction of water current, remaining battery power, the availability of the transducers in the sidescan sensor and the mission execution status. Figure 17 represents the system activity of the payload computer recorded during the mission. The system activity logs show percentage of processor usage, memory usage, network activity and disk usage.

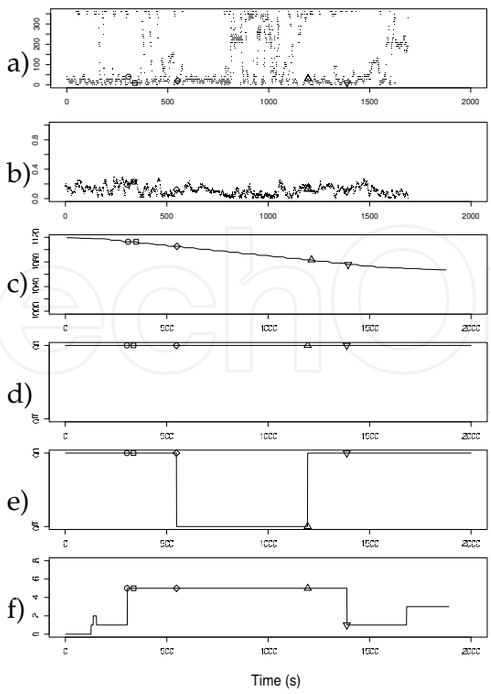


Fig. 16. Status monitoring (top to bottom): a) direction of water current (degrees), b) speed of water current (m/s), c) battery power (Wh), d) sidescan sensor port and e) starboard transducers availability (on/off) and f) mission status binary flag, all plotted against mission time (s).

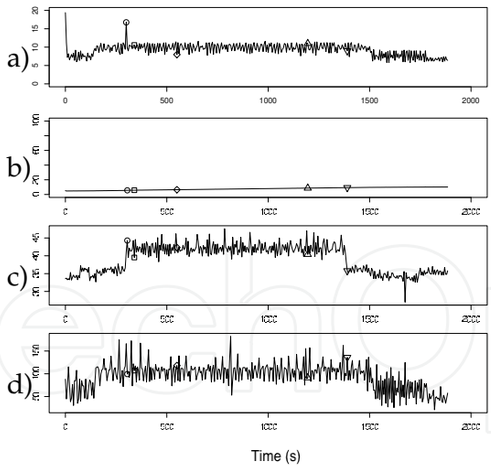


Fig. 17. System activity (top to bottom): a) % processor usage, b) % memory usage, c) network activity (packets/s) and d) disk usage (I/O sectors/s), all plotted against mission time (s).

Each of the symbols ◯, □, ♢, △ and ▽ on the aforementioned figures represents a point during the mission where an event occurred. Symbol ◯ represents the point where SAMP takes control of the vehicle. Note a change on the host platform mission status binary flag that becomes 0x05, i.e. the mission is active (0x01) and the payload is in control (0x04) (Figure 16.e).

Also, a peak on the CPU usage can be noted as this is the point where the mission partial plan gets generated (Figure 17.a).
Symbol □ represents the point where the vehicle arrives to perform the survey of the area. At this point, the action survey gets instantiated based on the properties of the internal elements and external factors. Although the Loch waters where the trials were performed were very still (see Figure 16.b), note how the vehicle heading during the lawnmower pattern performed to survey the areas follows the water current direction sensed at the arrival (approx. 12^o, Symbol □ - Figure 16.a) in order to minimize drag and maximise battery efficiency. The heading of the vehicle during the survey can be observed in Figure 14 and Figure 15.b. The link between the vehicle heading in relation to the water current direction and its effect on the battery consumption was expert orientation knowledge captured by a relationship property between the two concepts in the Core Ontology.
Symbol ♢ represents the point when the status monitor agent detects and reports a critical status in the starboard transducer of the sidescan sonar (Figure 16.d). It can be seen how the lawnmower pattern was adapted to cope with the change and to use the port transducer to cover the odd and even spacing of the survey. This pattern avoids gaps in the sidescan data under the degraded component configuration and maximises sensor coverage for the survey while the transducer is down.
Symbol △ indicates the point where the starboard transducer recovery is diagnosed. It can be observed how the commands executing the action are modified in order to optimise the survey pattern and minimise distance travelled. Although also being monitored, the power status does not report any critical status during the mission that requires modification of the actions (Figure 16.c).
Symbol ▽ shows the location where all the mission goals are considered achieved and the control is given back to the mission control of the host vehicle (see Symbol ▽ - Figure 16.e shows the mission is still active but the payload is not longer in control (0x01)). From this point the host vehicle's control module takes the control back and drives the vehicle to the loiter at the recovery location.

6. Conclusion and future work

The underwater domain is a challenging environment in which to maintain the operability of an UUV. Operability can be improved with the embedded adaptation of the mission plan. We implement a system capable of adapting mission plans autonomously in the face of events while during a mission. We do this by using a combination of ontological hierarchical representation of knowledge and adaptive mission plan repair techniques. The advantage of this approach is that it maximises robustness, system performance and response time. The system performance has been demonstrated in simulation. Additionally, the mission adaptation capability is shown during an in-water field trial demonstration.
In our fully integrated experiments we achieved the following:

- *Knowledge based framework:* We have presented a semantic-based framework that provides the core architecture for knowledge representation for service oriented agents in UUVs. The framework combines the initial expert orientation and the observations acquired during mission in order to improve the situation awareness in the vehicle. This is currently unavailable in UUVs.
- *Goal-oriented plan vs. waypoint-based plan:* The system uses a goal-oriented approach in which the mission is described in terms of 'what to do' instead of a 'how to do' it. The

mission is parametrised and executed based on the available knowledge and vehicle capabilities. This is the first time that an approach to goal-based planning is applied to the adaptation of an underwater mission in order to maintain platform's operability.

- *Adaptation to environmental parameters and internal issues:* The approach shows adaptability to environmental elements, such as water current flows in order to improve mission performance. The approach is also capable of dealing with the critical status of certain components in the platform and can react accordingly.
- *Platform agnostic:* The approach is platform independent making it readily applicable to other domains, such as ground or air vehicles.

SAMP is open to event detections coming from other embedded service-oriented agents. We are planning to apply the approach to more complex scenarios involving other embedded agents, such as agents for automatic target recognition. We are also planning to extend it to a team of vehicles performing a collaborative mission. In this scenario, agents are distributed across the different platforms. We are currently working towards a shared situation awareness for a team of vehicles to which every team member possess the awarenessrequired for its responsibilities. The main advantage of our semantic-based approach is the low bandwidth required to share id-coded ontological concepts and, therefore, to cope with the underwater acoustic communication limitations.

7. References

Adams, J. A. (2007). Unmanned vehicle situation awareness: A path forward, *Proceedings of the 2007 Human Systems Integration Symposium*, Annapolis, Maryland, USA.

Arkin, R. C. (1998). *Behavior-based Robotics*, Massachusetts Institute of Technology Press, ISBN: 978-0262011655.

Bellingham, J., Consi, T., Beaton, R. & Hall, W. (1990). Keeping layered control simple, *Proceedings of the (1990) Symposium on Autonomous Underwater Vehicle Technology (AUV'90)*, pp. 3–8.

Bellingham, J., Kirkwood, B. & Rajan, K. (2006). Tutorial on issues in underwater robotic applications, *16th International Conference on Automated Planning and Scheduling (ICAPS'06)*.

Benjamin, M. R., Leonard, J. J., Schmidt, H. & Newman, P. M. (2009). An overview of MOOS-IvP and a brief users guide to the IvP Helm autonomy software, *Technical Report MIT-CSAIL-TR-2009-028*, Computer Science and Artificial Intelligence Lab, MIT.

Bennet, A. A. & Leonard, J. J. (2000). A behavior-based approach to adaptive feature detection and following with autonomous underwater vehicles, *IEEE Journal of Oceanic Engineering* 25(2): 213–226.

Blomqvist, E. & Sandkuhl, K. (2005). Patterns in ontology engineering âĂŞ classification of ontology patterns, *7th International Conference on Enterprise Information Systems*, Miami, USA.

Boyd, J. (1992). A discourse on winning and losing - organic design for command and control, *Technical report*, http://www.d-n-i.net/dni/john-r-boyd/.

Brooks, R. (1986). A robust layered control system for a mobile robot, *Journal of Robotics and Automation* RA-2(1): 14–23.

Caccia, M. & Veruggio, G. (2000). Guidance and control of a reconfigurable unmanned underwater vehicle, *Control Engineering Practice* 8(1): 21 – 37.

Carreras, M., Palomeras, N., Ridao, P. & Ribas, D. (2007). Design of a mission control system for an AUV, *International Journal of Control* 80(7): 993–1007.

Carreras, M., Ridao, P., Garcia, R. & Batlle, J. (2006). *Behaviour Control of UUVs*, G. Roberts and R. Sutton, Eds Advances in Unmanned Maritime Vehicles (IEE Control Engineering), ISBN: 978-0863414503, chapter 4, pp. 67–86.

Chen, H., Perich, F., Finin, T. & Joshi, A. (2004). Soupa: Standard ontology for ubiquitous and pervasive applications, *International Conference on Mobile and Ubiquitous Systems: Networking and Services, Boston, MA*.

Farrell, J. A., Pang, S. & Li, W. (2005). Chemical plume tracing via an autonomous underwater vehicle, *IEEE Journal of Oceanic Engineering* 30, 2: 428–442.

Fossen, T. I. (1994). *Guidance and Control of Ocean Vehicles*, John Wiley and Sons, ISBN: 0-471-94113-1.

Ghallab, M., Howe, A., Knoblock, C., McDermott, D., Ram, A., Veloso, M., Weld, D. & Wilkins, D. (1998). PDDL: The planning domain definition language, *Technical report*, Yale Center for Computational Vision and Control.

Ghallab, M., Nau, D. & Traverso, P. (2004). *Automated Planning: Theory and Practice*, Morgan Kaufmann, ISBN: 1-55860-856-7.

Griffiths, G. (2005). AUTOSUB under ice, *Ingenia, ISSN: 1472-9768* 22: 30–32.

Gruber, T. R. (1995). Towards principles for the design of ontologies used for knowledge sharing, *International Journal Human-Computer Studies* 43: 907–928.

Hagen, P. E. (2001). AUV/UUV mission planning and real time control with the HUGIN operator systems, *Proceedings of the IEEE International Conference Oceans (Oceans'01)*, Vol. 1, Honolulu, HI, USA, pp. 468–473.

Jakuba, M. (2007). *Stochastic Mapping for Chemical Plume Source Localization with Application to Autonomous Hydrothermal Vent Discovery*, Doctor of philosophy in mechanical engineering, MIT/WHOI Joint Program in Applied Ocean Physics and Engineering.

JRP (2005). Joint robotics program master plan FY2005, *Technical report*, US Department of Defense.

Kokar, M. M., Matheus, C. J. & Baclawski, K. (2009). Ontology-based situation awareness, *Information Fusion* 10(1): 83–98.

Matheus, C., Kokar, M. & Baclawski, K. (2003). A core ontology for situation awareness, *Proceedings of the Sixth International Conference of Information Fusion* 1: 545 – 552.

McGann, C., Py, F., Rajan, K. & Henthorn, R. (2008). Adaptive control for autonomous underwater vehicles, *Association for the Advancement of Artificial Intelligence (AAAI'08)*, Chicago, IL, USA.

McGann, C., Py, F., Rajan, K., Henthorn, R. & McEwen, R. (2008). A deliberative architecture for AUV control, *International Conference on Robotic and Autonomation (ICRA'08)*, Pasadena, CA, USA.

McGann, C., Py, F., Rajan, K. & Olaya, A. G. (2009). Integrated planning and execution for robotic exploration, *International Workshop on Hybrid Control of Autonomous Systems*, Pasadena, CA, USA.

McGann, C., Py, F., Rajan, K., Thomas, H., Henthorn, R. & McEwen, R. (2007). T-REX: A model-based architecture for AUV control, *Workshop in Planning and Plan Execution*

for Real-World Systems: Principles and Practices for Planning in Execution, International Conference of Autonomous Planning and Scheduling (ICAPS'07).

Miguelanez, E., Lewis, R., K.Brown, Roberts, C. & Lane, D. (2008). Fault diagnosis of a train door system based on the semantic knowledge representation, *The 4th IET International Conference on Railway Condition Monitoring RCM 2008*, Derby Conference Centre, Derby, UK, pp. 1 –6.

Miguelanez, E., Patrón, P., Brown, K., Petillot, Y. R. & Lane, D. M. (2010). Semantic knowledge-based framework to improve the situation awareness of autonomous underwater vehicles, *IEEE Transactions on Knowledge and Data Engineering (In Press)* PP(99).

Newman, P. (2002). MOOS – a mission oriented operating suite, *Technical report*, Department of Ocean Engineering, Massachusetts Institute of Technology.

Niles, I. & Pease, A. (2001). Towards a standard upper ontology, *Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS-2001)* .

Oce (2005). OceanSHELL: An embedded library for distributed applications and communications, *Technical report*, Ocean Systems Laboratory, Heriot-Watt University.

Oliveira, P., Pascoal, A., Silva, V. & Silvestre, C. (1998). Mission control of the MARIUS AUV: System design, implementation, and sea trials, *International Journal of Systems Science* 29(10): 1065–1085.

Palomeras, N., Ridao, P., Carreras, M. & Silvestre, C. (2009). Using petri nets to specify and execute missions for autonomous underwater vehicles, *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'09)*.

Patrón, P. (2009). Embedded knowledge and autonomous planning, *Sea Technology Magazine, ISSN. 0093-3651* 50(4): 101.

Patrón, P. & Birch, A. (2009). Plan proximity: an enhanced metric for plan stability, *Workshop on Verification and Validation of Planning and Scheduling Systems, 19th International Conference on Automated Planning and Scheduling (ICAPS'09)*, Thessaloniki, Greece, pp. 74–75.

Patrón, P., Evans, J. & Lane, D. M. (2007). Mission plan recovery for increasing vehicle autonomy, *Proceedings of the Conference of Systems Engineering for Autonomous Systems from the Defence Technology Centre (SEAS-DTC'07)*, Edinburgh, UK.

Patrón, P., Lane, D. M. & Petillot, Y. R. (2009a). Continuous mission plan adaptation for autonomous vehicles: balancing effort and reward, *4th Workshop on Planning and Plan Execution for Real-World Systems, 19th International Conference on Automated Planning and Scheduling (ICAPS'09)*, Thessaloniki, Greece, pp. 50–57.

Patrón, P., Lane, D. M. & Petillot, Y. R. (2009b). Interoperability of agent capabilities for autonomous knowledge acquisition and decision making in unmanned platforms, *Proceedings of the IEEE International Conference Oceans Europe (Oceans Europe'09)*, Bremen, Germany.

Patrón, P., Miguelanez, E., Cartwright, J. & Petillot, Y. R. (2008). Semantic knowledge-based representation for improving situation awareness in service oriented agents of autonomous underwater vehicles, *Proceedings of the IEEE International Conference Oceans (Oceans'08)*, Quebec, Canada.

Patrón, P., Miguelanez, E., Petillot, Y. R. & Lane, D. M. (2008). Fault tolerant adaptive mission planning with semantic knowledge representation for autonomous underwater

vehicles, *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'08)*, Nice, France, pp. 2593–2598.

Patrón, P. & Petillot, Y. R. (2008). The underwater environment: A challenge for planning, *Proceedings of the Conference of the UK Planning Special Interest Group (PlanSIG'08)*, Edinburgh, UK.

Penberthy, J. & Weld, D. S. (1992). UCPOP: A sound, complete, partial order planner for ADL, *Proceedings of the National Conference on Knowledge Representation and Reasoning (KR)*, pp. 103–114.

Portele, C. (2007). OpenGISÂő geography markup language (GML) encoding standard v.3.2.1, *Technical report*, Open Geospatial Consortium Inc.

Rajan, K., McGann, C., Py, F. & Thomas, H. (2007). Robust mission planning using deliberative autonomy for autonomous underwater vehicles, *Workshop in Robotics in challenging and hazardous environments, International Conference on Robotics and Automation (ICRA'07)*.

Rajan, K., Py, F., McGann, C., Ryan, J., O'Reilly, T., Maughan, T. & Roman, B. (2009). Onboard adaptive control of AUVs using automated planning and execution, *International Symposium on Unmanned Untethered Submersible Technology (UUST'09)*, Durham, NH, USA.

Reed, S., Ruiz, I., Capus, C. & Petillot, Y. (2006). The fusion of large scale classified side-scan sonar image mosaics, *IEEE Transactions on Image Processing* 15(7): 2049–2060.

REM (2008). REMUS remote control protocol v1.12, *Technical report*, Hydroid Inc.

Ridao, P., Batlle, J., Amat, J. & Roberts, G. (1999). Recent trends in control architectures for autonomous underwater vehicles, *International Journal of Systems Science* 30(9): 1033–1056.

Rosenblatt, J. K., Williams, S. B. & Durrant-Whyte, H. (2002). Behavior-based control for autonomous underwater exploration, *International Journal of Information Sciences*, 145(1–2): 69–87.

Sacerdoti, E. (1975). The nonlinear nature of plans, *International Joint Conference on Artificial Intelligence (IJCAI'75)*, pp. 206–214.

SAE (2006). Society of automotive engineers AS-4 AIR5664 JAUS history and domain model, *Technical report*, SAE International Group.

SAE (2008a). Society of automotive engineers AS-4 AIR5665 JAUS architecture framework for unmanned systems, *Technical report*, SAE International Group.

SAE (2008b). Society of automotive engineers AS-4 ARP6012 JAUS compliance and interoperability policy, *Technical report*, SAE International Group.

Sirin, E., Parsia, B., Grau, B. C., Kalyanpur, A. & Katz, Y. (2007). Pellet: A practical owl-dl reasoner, *Web Semantics: Science, Services and Agents on the World Wide Web* 5(2): 51–53.

Turner, R. (2005). Intelligent mission planning and control of autonomous underwater vehicles, *Workshop on Planning under uncertainty for autonomous systems, 15th International Conference on Automated Planning and Scheduling (ICAPS'05)*.

Turner, R. M. (1995). Context-sensitive, adaptive reasoning for intelligent auv control: Orca project update, *In Proceedings of the Ninth International Symposium on Unmanned, Untethered Submersible Technology (UUST'95)*, Durham, NH, USA, pp. 426–435.

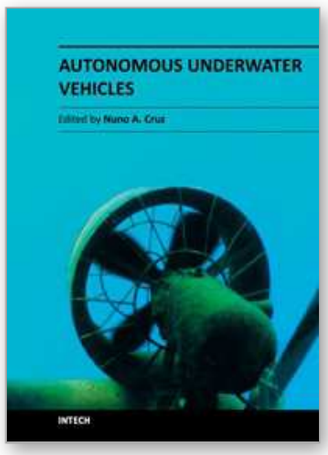
UUV (2004). The navy unmanned undersea vehicle (UUV) master plan, *Technical report*, US Department of the Navy.

van der Krogt, R. (2005). *Plan repair in single-agent and multi-agent systems*, PhD thesis, Netherlands TRAIL Research School.

van Heijst, G., Schreiber, A. & Wielinga, B. (1996). Using explicit ontologies in kbs development, *International Journal of Human-Computer Studies* 46(2-3).

von Alt, C. (2010). Remus technology 2020 and beyond, *NATO Naval Armaments Group – Maritime Capability Group 3 on Mines, Mine Countermeasures and Harbour Protection – Industry Day*, Porton Down, UK.

Yuh, J. (2000). Design and control of autonomous underwater robots: A survey, *Journal of Autonomous Robots* 8(1): 7–24.



Autonomous Underwater Vehicles

Edited by Mr. Nuno Cruz

ISBN 978-953-307-432-0

Hard cover, 258 pages

Publisher InTech

Published online 17, October, 2011

Published in print edition October, 2011

Autonomous Underwater Vehicles (AUVs) are remarkable machines that revolutionized the process of gathering ocean data. Their major breakthroughs resulted from successful developments of complementary technologies to overcome the challenges associated with autonomous operation in harsh environments. Most of these advances aimed at reaching new application scenarios and decreasing the cost of ocean data collection, by reducing ship time and automating the process of data gathering with accurate geo location. With the present capabilities, some novel paradigms are already being employed to further exploit the on board intelligence, by making decisions on line based on real time interpretation of sensor data. This book collects a set of self contained chapters covering different aspects of AUV technology and applications in more detail than is commonly found in journal and conference papers. They are divided into three main sections, addressing innovative vehicle design, navigation and control techniques, and mission preparation and analysis. The progress conveyed in these chapters is inspiring, providing glimpses into what might be the future for vehicle technology and applications.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Pedro Patrón, Emilio Miguelañez and Yvan R. Petillot (2011). Embedded Knowledge and Autonomous Planning: The Path Towards Permanent Presence of Underwater Networks, Autonomous Underwater Vehicles, Mr. Nuno Cruz (Ed.), ISBN: 978-953-307-432-0, InTech, Available from: http://www.intechopen.com/books/autonomous-underwater-vehicles/embedded-knowledge-and-autonomous-planning-the-path-towards-permanent-presence-of-underwater-network

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen