Service-Oriented Agent Architecture for Autonomous Maritime Vehicles

Carlos Ceferino Insaurralde

Master of Philosophy

Institute of Sensors, Signals and Systems

School of Engineering and Physical Sciences

Heriot-Watt University

November 2013

Abstract

Advanced ocean systems are increasing their capabilities and the degree of autonomy more and more in order to perform more sophisticated maritime missions. Remotely operated vehicles are no longer cost-effective since they are limited by economic support costs, and the presence and skills of the human operator. Alternatively, autonomous surface and underwater vehicles have the potential to operate with greatly reduced overhead costs and level of operator intervention. This Thesis proposes an Intelligent Control Architecture (ICA) to enable multiple collaborating marine vehicles to autonomously carry out underwater intervention missions. The ICA is generic in nature but aimed at a case study where a marine surface craft and an underwater vehicle are required to work cooperatively. They are capable of cooperating autonomously towards the execution of complex activities since they have different but complementary capabilities. The architectural foundation to achieve the ICA lays on the flexibility of service-oriented computing and agent technology. An ontological database captures the operator skills, platform capabilities and, changes in the environment. The information captured, stored as knowledge, enables reasoning agents to plan missions based on the current situation. The ICA implementation is verified in simulation, and validated in trials by means of a team of autonomous marine robots. This Thesis also presents architectural details and evaluation scenarios of the ICA, results of simulations and trials from different maritime operations, and future research directions.

Keywords

Maritime robotics; autonomous marine missions; unmanned water vehicles; cognitive control architectures; on-board decision-making

To my beloved wife and two daughters; Carla, Florence and Ariadne

Acknowledgements

Thanks to my supervisors Prof Yvan R. Petillot and Prof David M. Lane for their non-stop support. In particular, Yvan giving me the opportunity to be creative and innovative in everyday work tasks and David for trusting in me to deal with challenging activities.

A special thanks to the all students, researchers and technicians from the Ocean Systems Laboratory. With special thanks to Joel Cartwright for his unconditional willingness to always collaborate and provide me with technical support.

Thanks to the European Commission for the financial support to carry out this piece of research work. Also, my most sincere thanks to all the partners from the TRIDENT Project [1] for the support information provided about the marine vehicles platform to improve my research work.

My true appreciation to my beloved family (Carla, Florence, and Ariadne) who understand and bear many hours I usually devote to my research work. Especially, my wife Carla for being always supportive, and providing me with her huge love and endless company. Also, my two little starts called Florence and Ariadne for changing my life for good, and lighting every moment of my life.

ACADEMIC REGISTRY
Research Thesis Submission

Name*:*	Carlos Ceferino Insaurralde		
School/PGI:	Engineering and Physical Sciences / Institute of Sensors, Signals and Systems		
Version: *(i.e. First, Resubmission, Final)*	Final	Degree Sought (Award **and** Subject area)	MPhil

Declaration

In accordance with the appropriate regulations I hereby submit my thesis and I declare that:

1) the thesis embodies the results of my own work and has been composed by myself
2) where appropriate, I have made acknowledgement of the work of others and have made reference to work carried out in collaboration with other persons
3) the thesis is the correct version of the thesis for submission and is the same version as any electronic versions submitted*.
4) my thesis for the award referred to, deposited in the Heriot-Watt University Library, should be made available for loan or photocopying and be available via the Institutional Repository, subject to such conditions as the Librarian may require
5) I understand that as a student of the University I am required to abide by the Regulations of the University and to conform to its discipline.

* *Please note that it is the responsibility of the candidate to ensure that the correct version of the thesis is submitted.*

Signature of Candidate*:*		Date:	

Submission

Submitted By *(name in capitals):*	
Signature of Individual Submitting:	
Date Submitted:	

For Completion in the Student Service Centre (SSC)

Received in the SSC by (name in capitals):			
Method of Submission (Handed in to SSC; posted through internal/external mail):			
E-thesis Submitted (**mandatory for final theses)**			
Signature:		Date:	

Table of Contents

List of Figures

List of Tables

Acronyms

AMR	Autonomous Marine Robots
AS	Action Service
ASC	Autonomous Surface Craft
CPOSS	Component Profile for Ocean System Services
IAUV	Intervention Autonomous Underwater Vehicle
IDL	Interface Definition Language
MS	Mission Service
MUV	Manned Underwater Vehicle
OCU	Operator Control Unit
OS	Operation Service
ROS	Robotic Operating System
ROV	Remotely Operated Vehicle
SA	Situation Awareness
SOA	Service-Oriented Architecture
SysML	System Modelling Language
TS	Task Service

Glossary

A **System** in the context of this report is the AMR. It involves the OCU, ASC, and IAUV.

An **Action** is the atomic (or component-level) activity.

An **Activity** is a service process (service execution) at different encapsulation levels such as mission, operation, task, or action levels. An activity is invoked by messages.

An **Agent** is a software entity that has one or more capabilities.

A **Capability** is an agent property (aptitude, talent or quality) to carry out a specific activity.

Choreography of services deals with the messages exchanges among services that are executed in parallel (collaborative nature).

A **Component** is the atomic block of the system structure. It is built of the following elemental parts: mechanical, hardware or software elements. However, they can only involve a mechanical, hardware or software part.

Functionality is a basic capability that a system component has. It is known as "behaviour" from the robotics architecture viewpoint.

A **Goal** is the final state of a concrete activity that the system wishes to achieve.

A **Message** is built of two main parts; event and data. It can also be defined as an event without data.

A **Mission** is the process of developing strategies to achieve a goal. Technically, it is a set of messages (as commands) to be performed by one or more components.

An **Operation** is a subsystem activity.

Orchestration is the way in which services are executed. It represents the composition of activities or service processes.

A **Plan** involves and carries out specific activities to go from an initial state to a desired state (goal).

A **Platform** is the AMR system, i.e. the ASC plus the IAUV.

A **Process** is built of system activities

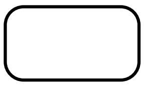
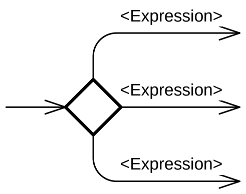
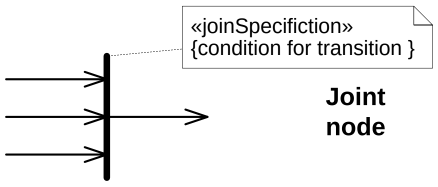
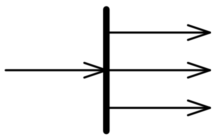
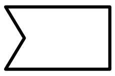
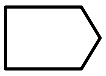
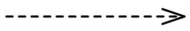
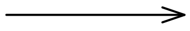
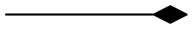
A **Service** encapsulates functionality from system components.

A **Task** is a node (or device) activity.

Serviceability is the ability to provide services.

Symbolism

The representations of services are made by means of SysML diagrams [2]. The main SysML symbols are as follows.

	**Activity** (service execution)	SysML activities are token-driven. A token holds values of inputs, outputs, and control that flow from one action to other. The primitive activity is to represent service execution. A activity takes place when a service is executed. It may be at different levels, mission, operation, task, and action.
 <Expression> <Expression> <Expression>	**Decision node**	A decision node has one input flow and multiple output flows (an input token[1] can only traverse one output flow). The output flow is typically established by placing mutually exclusive guards on all outgoing flows and offering the token to the flow whose guard expression is satisfied. A decision node can have an accompanying decision input behavior, which is used to evaluate each incoming token and whose result can be used in guard expressions.
	**Joint node**	A join node has one output flow and multiple input flows (it has the important characteristic of synchronizing the flow of tokens from many sources). Its default behaviour can be overridden by providing a join specification, which can specify additional control logic.
	**Fork node**	A fork node has one input flow and multiple output flows. It replicates every input token it receives onto each of its output flows. The tokens on each output flow may be handled independently and concurrently.
	**Receive message**	An activity can accept events using an receive message signal. The event has (sometimes hidden) output pins for received data.
	**Send message**	An activity can send signals using a send message signal. It typically has pins corresponding to the signal data to be sent and the target for the signal.
	**Dependency**	
	**Association**	
	**Composition**	

SysML Notation

bdd: block definition diagram

pkg: package

ibd: internal block diagram

uc: use case

sd: sequence diagram

act: activity diagram

stm: state machines

List of Publications

- F. J. Ortiz, C. C. Insaurralde, D. Alonso, F. Sanchez-Ledesma, Y. R. Petillot, “Model-Driven Analysis and Design for Software Development of Autonomous Underwater Vehicles”, Robotica, Cambridge University Press, Oct 2013. Accepted for publication.

- C. C. Insaurralde, Y. R. Petillot, “Intelligent Autonomy for Collaborative Intervention Missions of Unmanned Maritime Vehicles”, OCEANS MTS/IEEE Conference, San Diego, CA, USA, Sep 2013.

- C. C. Insaurralde, Y. R. Petillot, “Capability-Based Engineering for Maritime Robotics - Service-Oriented Agent Technology in Unmanned Marine Vehicles”, in proceeding of the Global Virtual Conference, Apr 2013.

- C. C. Insaurralde, Y. R. Petillot, “From Research Project Objectives to Milestones by means of Requirements Traceability - Realization of an Autonomous Maritime System”, in proceeding of the IEEE International Systems Conference, Orlando, FL, USA, pp. 737-744, Apr 2013.

- C. C. Insaurralde, Y. R. Petillot, “Cyber-Physical Framework for Early Integration of Autonomous Maritime Capabilities”, in proceeding of the IEEE International Systems Conference, Orlando, FL, USA, pp. 559-566, Apr 2013.

- F. J. Ortiz, F. Sanchez, D. Alonso, F. Rosique, C. C. Insaurralde, “C-Forge: a Model-Driven Toolchain for Developing Component-Based Robotics Software”, in proceeding of the 8th full-day Workshop on Software Development and Integration in Robotics (SDIR), IEEE International Conference on Robotics and Automation (ICRA), Karlsruhe, Germany, May 2013.

- F. Maurelli, J. Cartwright, Z. Saigol, C. C. Insaurralde, Y. Petillot, D. Lane, “Marine world representation and acoustic communication: challenges for multi-robot collaboration”, in proceeding of the IEEE/OES AUV 2012, Southampton, UK, Sep 2012.

- P. Sanz, P. Ridao, G. Oliver, G. Casalino, C. Insaurralde, C. Silvestre, C. Melchiorri, A. Turetta, “TRIDENT: Recent Improvements about Autonomous Underwater Intervention Missions”, in proceeding of the International Federation of Automatic Control (IFAC) Workshop on Navigation, Guidance and Control of Underwater Vehicles, Porto, Portugal, vol. 3, part 1, pp. 355-360, Apr 2012.

- C. C. Insaurralde, J. J. Cartwright, Y. R. Petillot, “Cognitive Control Architecture for Autonomous Marine Vehicles”, in proceeding of the IEEE International Systems Conference, pp. 117-124, Vancouver, British Columbia, Canada, Mar 2012.

Chapter 1: Introduction

1.1 Background

The research presented in this Thesis was carried out within a context set by the project TRIDENT [1]. This project proposes an innovative approach for multipurpose underwater intervention tasks. It is based on the integration of different maritime capabilities provided by the cooperation between autonomous surface and underwater vehicles. An effective vehicle control architecture plays a key role to achieve such robotic autonomy (including adaptive planning, re-planning, and fault-tolerance).

Robotics is a multi-engineering discipline that is increasingly present in many application domains. In particular, water-surface and underwater robots have gained considerable interest in the last decades in great part due to the industrial and governmental concern for exploiting oceans, and sea in search of alternative resources from the earth. Thus, Unmanned Marine Vehicles (UMVs) have become a pervasive solution for several maritime businesses by playing a key role as ad-hoc autonomous platforms across different operation fields.

The main consumers of the above ocean engineering technology are institutions and companies (maritime rescue organizations, military and defence industry, aviation security investigation agencies, etc.) working in seawater scenarios. However, most demanding applications are underwater missions (e.g. seabed survey; seafloor data collection for marine biology applications, and target manipulations; push a button, or recovery of any kind of object) since they are a hostile environment for humans which access is not straightforward. Additionally, they usually involve repetitive and time-consuming tasks risky for divers.

The accessibility to deep-water regions is not simple either for human beings or man-made systems mainly due to limitations set by the communication channel (essential interaction for coordination; cooperation and collaboration) as well as high pressures (critical impact on human physiology and instruments). Nevertheless, it is a lot easier for robotic systems than people and on the basis that bandwidth is not a problem for instruments on a wire.

Unfortunately, technological robotics solutions for other challenging domains such as space or ground robots are not directly applicable to maritime robotics since they are not suitable due to the nature of the subaquatic environment (i.e. different sensors, and effectors as well as processing and communication technologies).

1.2 Justification and Motivation

Most of the undersea fieldwork operations such as maritime rescue, inspection and light intervention on offshore structures, and marine biology applications are currently carried out with one of the following underwater vehicles endowed with robotic arms: Manned Underwater Vehicle (MUVs) or Remotely Operated Vehicles (ROVs).

The MUVs have the advantage of placing the operator in the fieldwork so he/she comes in view of the operation scene (e.g. object(s) to be manipulated). The drawbacks of this solution are the limited operation time (typically a few hours), the human presence in a dangerous and hostile environment, and a very high cost because of the need for an expensive supply and supervision of the operation of such vehicles.

ROVs are the de facto technology for deep interventions. They can be remotely operated for long periods. However, they need an expensive support vessel with a heavy crane. Another disadvantageous issue is the physical and mental fatigue of the ROV pilot who has to deal with the ROV and its umbilical cord while interacting with the operator in charge of the robotic arms. What makes ROVs not much attractive is that they are very expensive (basically due to the labour costs) and tie up a ship.

1.3 Problem Statement

The costly and risky situation set by crewed submersibles and remotely-controlled underwater robots comes along with a main issue which is devising a more effective and efficient solution for maritime missions. The intent to reducing costs and risks create opportunities that put system thinking in practice for UMVs. This makes it possible to come up with a solution based on an analysis on how UMVs interact with users and other support systems, and how much they can do working on their own and also with other UMVs. The increment self-governance from the above tightly-coupled actors (i.e. operator, oceanographic vessel, etc.) set challenging activities as to design and operation of UMVs.

The challenge is to develop a system that deploys a team of marine vehicles that can perform complex tasks reliably and with minimal operator intervention. A critical issue to achieve this is to design and build a system with the ability to deal with internal faults, and changes in the environment as well as their impact on sensor outputs used for the planning phase. Therefore, new marine vehicle platforms require a certain degree of autonomy, and a collaborative operation mode in order to minimize the operator intervention.

1.4 Hypothesis and Objectives

The above problem definition enabled by state-of-the-art batteries envisages the technological evolution of the intervention ROVs: the Intervention Autonomous Underwater Vehicle (IAUVs). Thus, IAUVs could theoretically be operated from cheaper vessels without the need for an automatic tether management system and the dynamic position system. Additionally, manipulative operations could last for several days if the operator is removed from the control loop.

The replacement of the support ship by an Autonomous Surface Craft (ASC) could drastically reduce the operative costs, in particular in inland-water or shoreline missions. An ASC and an IAUV form together a heterogeneous team of marine robots with complementary skills to lead ocean/sea missions such as underwater surveys or manipulative interventions.

The aforementioned robotic team can be deployed from an oceanographic vessel to autonomously carry out a mission while other scientific tasks are performed from the ship in different areas. In addition, an ASC equipped with an Ultra-Short Base Line (USBL), an acoustic modem, and a radiomodem can geo-reference the IAUV position as well as to establish a communication link to allow for a remote tracking and supervision of the UMV team to the operator.

The above promising UMV configuration, where the marine robots are launched to do the work autonomously before recovery, is possible at a cost of endowing the vehicles with "intelligence" that in former solutions is provided by the human operator. The approach proposed in this Thesis can be applied to conventional AUVs which are also operated without human in the control loop. They are tightly limited to seabed surveys ("flying" at a safe altitude with respect to the seafloor whilst logging data) whereas IAUVs operate in close proximity of a specific mission scenario (target of interest). This accurate IAUV application makes them even more challenged.

1.5 Research Contributions

This Thesis proposes an Intelligent Control Architecture (ICA) to enable multiple maritime vehicles to carry out autonomous multipurpose underwater intervention missions. The ICA is generic in nature but aimed at a case study where an ASC and an IAUV are required to work cooperatively. They are capable of cooperating towards the execution of complex activities since they have different but complementary capabilities.

The ICA proposed moves away from fixed mission plans and very elementary diagnostics scheme currently utilized to a more robust architecture to deal with the above missions. It also copes with unexpected faults within vehicle, e.g. at the sensor and sensor processing levels, based on either hardware failure or environmental changes. The architectural foundation to achieve the ICA lays on the flexibility of service-oriented computing. Each vehicle module provides basic services which advertise their capabilities to the system. The service also publishes regular updates of its current status.

In addition to the service orientation paradigm, a knowledge-based database captures the domain specific skills of the human expert (how to perform a specific task), as well as the dynamic information concerning the environment and platform capabilities. This makes it possible to include small atomic plans to test and validate the performance of specific services. The knowledge captured enables high-level reasoning agents to monitor, refine, or adapt mission plans based on the current situation. The resulting architectural solution proposed is a service-oriented agent-based approach which is suitable for integrating the vehicle modules as well as the capabilities of each marine vehicle in a collaborative manner. The agents are embedded in the marine vehicles. They are specialized in different disciplines, and provide different capabilities available as platform services (e.g. navigation, mapping, vision, etc.) to the overall system. The ICA is the backbone for the development of agents by providing a de-facto integration approach.

The combination of the two above technologies makes it possible to develop an ICA that is able to dynamically reconfigure and adapt itself in order to deal with changes in the operation environment. Thus, this ability to perform on-the-fly re-planning of activities when needed increases the chance to succeed in a given mission. The ICA performance is tested and demonstrated for a particular case study. However, it is a general solution for maritime autonomy that can be applied to other marine missions, and unmanned maritime vehicles.

1.6 Thesis Organization

This first Chapter presents an introduction to the research topic of the Thesis. It begins by setting the investigation context in the maritime application domain where marine robots play a key role due to industrial and governmental interests. The emphasis is on underwater missions supported from the surface by oceanographic vessels. Currently, ROVs are no longer cost-effective as no cheaper solution exists for such missions since they are limited by economic support costs, and the presence and skills of the human operator. Alternatively, autonomous surface and underwater vehicles (i.e. UMVs) have the potential to operate with greatly reduced overhead costs and level of operator intervention. This Thesis proposes an ICA to enable multiple collaborating UMVs to autonomously carry out underwater intervention missions.

Chapter 2 presents a review of current robotic control architectures. The state of the art classifies the existing technologies based on two main artificial agent approaches: (1) single and multiple agents per vehicles, and (2) intelligent agents. This Chapter also includes a Subsection that compare the most relevant approaches of intelligent control architectures applied to maritime robots.

Chapter 3 describes key architectural foundations by presenting conceptual principles of the ICA proposed. It also shows aspects of the control hierarchy (from goals to behaviours) implemented by the above architecture as well as a detailed explanation of knowledge representation and ontological reasoning methodologies used to implement artificial intelligence for the UMVs.

Chapter 4 shows design details of the ICA proposed. It identifies and analyses the user and system requirements in order to proceed with the architectural design of the ICA which is carried out by means of a top-down decomposition approach. The ICA essence relies on functionality from the UMV modules wrapped in services advertised to, and discovered by the overall Autonomous Marine Robot (AMR) system.

Chapter 5 explains how the ICA proposed is realized. It shows details of the architecture implementation and integration, the operation context, and a case study. The architecture implementation involves developing and building the modules that provide services to the AMR system. The architecture integration entails the combination and assembly of all the above modules and their services. The case study has an operational environment set by a generic underwater intervention carried out by two UMVs: an ASC and an IAUV.

Chapter 6 presents experimental results obtained from simulations and trials from different maritime operations in order to evaluate the ICA proposed when it is implemented in the above marine system platforms (ASC and IAUV). Experiments are carried out by means of computer simulations and sea trials. The former focuses on the simulation of a seabed survey and an underwater target manipulation executed in a 3D simulator. The later focuses on trials carried out in two different places: a lock and a port.

The last Chapter presents the conclusions and future research directions. The former recaps the achievements of this Thesis by summarizing key points, development milestones, and verification and validation of the ICA proposed. The latter emphasizes the next research steps for this Thesis by making a point on enriching the ontological database in order to cope with more complex evaluation scenarios (including other potential faults and unexpected situations).

Appendix A includes details of the main entities of the core ontology developed for the ICA proposed. Appendix B shows a detailed specification of all the low-level functionalities that are wrapped as services, and implemented by the Robot Operating System (ROS). Appendix C entails a full description of all the services and the orchestration and choreography mechanisms for such services of the ICA for the particular case study of this Thesis. Appendix D presents detailed versions of the ROS services interfaces, including coordinate adjustments, including coordinate frames, and coordinates transformations. Appendix E shows the XML file for configuration of the missions studied in this Thesis.

Chapter 2: Existing Robotic Control Architectures

2.1 Single and Multiple Agents per Vehicles

The state of the art of robotic control architectures presented in this Chapter focuses on the ocean engineering and robotics areas. In particular, it only involves mobile agent systems for marine vehicles since this Thesis proposes to apply the agent technology to an ARM system where each Autonomous Marine Vehicle (AMV) is considered a mobile agent.

Addressing the above context, two main classifications can be made. The first is according to the amount of agents implemented per vehicle. The second is according to those approaches that rigorously follow the basic agent architecture and in particular, those which really propose approaches of intelligent agent architecture.

There are many proposals implementing several agents per individual mobile platform. These approaches generally assign an agent to a functional system module (navigator, pilot, vision processor, etc.). Thus, the agents are defined as key components of the infrastructure that supports the system [3]–[9]. An alternative approach some researchers have taken is to implement only one agent per mobile platform [10]–[14], [27]. Many-agent solutions can be computationally distributed within a vehicle so simpler agents can be implemented but the interaction among agents is increased which involves additional communication tasks. One-agent solutions increase the need for computational resources to implement the agent but simply the agency (community of agents), i.e. external communication among agents.

2.2 Intelligent Agents

Researchers at the Center for Intelligent Systems Research (CISR), George Washington University are working on decentralized control for multiple Autonomous Underwater Vehicles (AUVs). They proposed an ontological approach for collective behaviour of intelligent agents that can be applied to AUV fleets [14]. Their proposal is based on collective intentionality in agents. It is an alternative to the traditional Beliefs-Desires-Intentions (BDI) model of individual agents that capitalized on the commitments agents have to one another rather than the commitments the agents have to maximize their own individual utility functions [15]. This collective way to manage agent commitments is good for lowering the number of tasks per agent (less busy agents) but it consequently makes agent more sophisticated to achieve effective interactions.

The Ocean Systems Lab at Heriot-Watt University has started working on situation awareness (SA) in service-oriented agents for AUVs. SA is an agent ability to be conscious to realize about internal and external states (see page 23). The information provided by SA is essential to make decisions. Better decisions can be made based on higher SA. The research mainly focuses on semantic knowledge-based representation for improving SA [9], distributed ontological world model [17], and adapting mission planning [18]. The above approaches pave the way towards reconfigurable control architectures for autonomous maritime vehicles based on service-oriented agents. They provide AUVs with flexible adaptation to mission requirements (e.g. self-repairing capabilities for planning, the ability to autonomously re-plan or repair a plan (totally or partially) without the operator intervention is invaluable and save time during AUV missions).

There are successful stories of underwater operations such as field measurements of scalar temperature, salinity, or pollutant concentration fields in environmental tracking applications of formation control of AUVs [19]. Defence systems are also demanding autonomous air and marine vehicles for networked multi-vehicle systems for an ocean platform control for the mobile offshore base which includes coordinated operations of several AUVs, and unmanned combat air vehicles [20]. It shows some current issues common to the above systems regarding hardware-software co-design, coordinated control strategies, manoeuvres, communication, and real-time constraints.

2.3 Comparison of Agent-based Approaches

The main contribution of this Thesis is a generic architectural solution for autonomous marine vehicles. The ICA proposed is a solution that goes beyond existing approaches by combining and extending the characteristics mentioned above in Subsection 2.2, i.e. autonomy mainly based on adaptive planning of tasks to tackle missions for single and multiple AMV(s) and AMV behaviours generated by means of autonomous on-board decision-making capabilities. Whilst some approaches [15], [16] do propose agents to deal with the coordination of marine vehicles, the ICA additionally proposes a mechanism for dynamic discovery of platform capabilities, and a knowledge database [17] in order to support adaptive planning of missions [18]. None of the approaches presented in this Section has an integrated ability to (1) advertise, (2) discover, (3) (re-)plan based on, (4) monitor health of, and (5) execute system capabilities while dealing with maritime missions in a fault-tolerant manner.

The advertisement and discovery of system capabilities as well as a semantic knowledge database with on-board decision-making to build action plans make the main difference from the current approaches. They are essential elements to endow UMVs with intelligent autonomy. These two essential human-like mechanisms allows operators to fully delegate control to the autonomous maritime system (the ASC, and the IAUV) to autonomously carry out maritime activities that are currently assisted and pre-programmed by human operators in other approaches. This is a clear increase in the degree of maritime autonomy, aiming to reduce the expensive deployment and operation of ROVs. Thus, it also brings within reach complex multi-vehicle collaborative missions that are too costly or logistically infeasible with current approaches (i.e. MUVs, ROVs, and most AUVs) due to the low degree of autonomy they have to deal with underwater missions. This is ultimately limited by the computational and mechanical capabilities they have integrated on board.

Table 1 shows the criterion to compare existing approaches with the ICA proposed in this Thesis. What is being assessed is the ability of each architecture to deal with faults (how much can be coped with), make decisions on board and on the fly, dynamically recognize services available in the system, and compute data.

Table 1. Comparison of most relevant approaches of intelligent control architectures applied to maritime robots.

Architectural Approaches	**Faults Diagnosis & Mitigation**	**Planning & Re-planning**	**Advertise & discovery of capabilities**	**In-vehicle Computing Paradigm**
PN-MAS [14]	Partially supported	Only for obstacle detection	No	Agent
SKR [9], [17], [18]	Some cases supported	Off-board decision-making	No	Object
MARIOUS [7]	Few cases supported	On-board decision-making	No	Multi-agent
MAA [10]	Not supported	Off-board decision-making	No	Multi-agent
DVMA [13]	Not supported	Off-board decision-making	No	Sequential
T-REX [27]	Not supported	Constraint-based reasoning decision-making	No	Agent
JAUS [22]	Not defined	Open platform decision-making	No	Service-based component
REMORAS [28]	Not supported	On-board decision-making	Pre-known functions as agent roles	Multi-agent
Proposed ICA	Supported	On-board decision-making	Supported	Multi-agent based on services

Chapter 3: Intelligent Control Architecture

3.1 Architectural Foundations

This Chapter presents the structural and behavioural aspects of the ICA proposed in this research work. It describes architectural foundations key to develop the ICA and presents conceptual principles as to its structure and behaviour. It involves aspects of the control hierarchy (from goals to behaviours) for the above architecture as well as a detailed explanation of knowledge representation and ontological reasoning methodologies to apply artificial intelligence to UMVs

Figure 1 shows the architectural concepts of the ICA. In the top of the figure, the system deals with hierarchical mission goals that are achieved by the execution of agent plans (sequence of activities listed as command messages). The planning and matching are intellectual agent activities. The planning of tasks for an agent is performed by each agent by matching internal agent capabilities but also taking into account external capabilities from other agents to carry out different activities. Agents are able to discover the capabilities of each other.

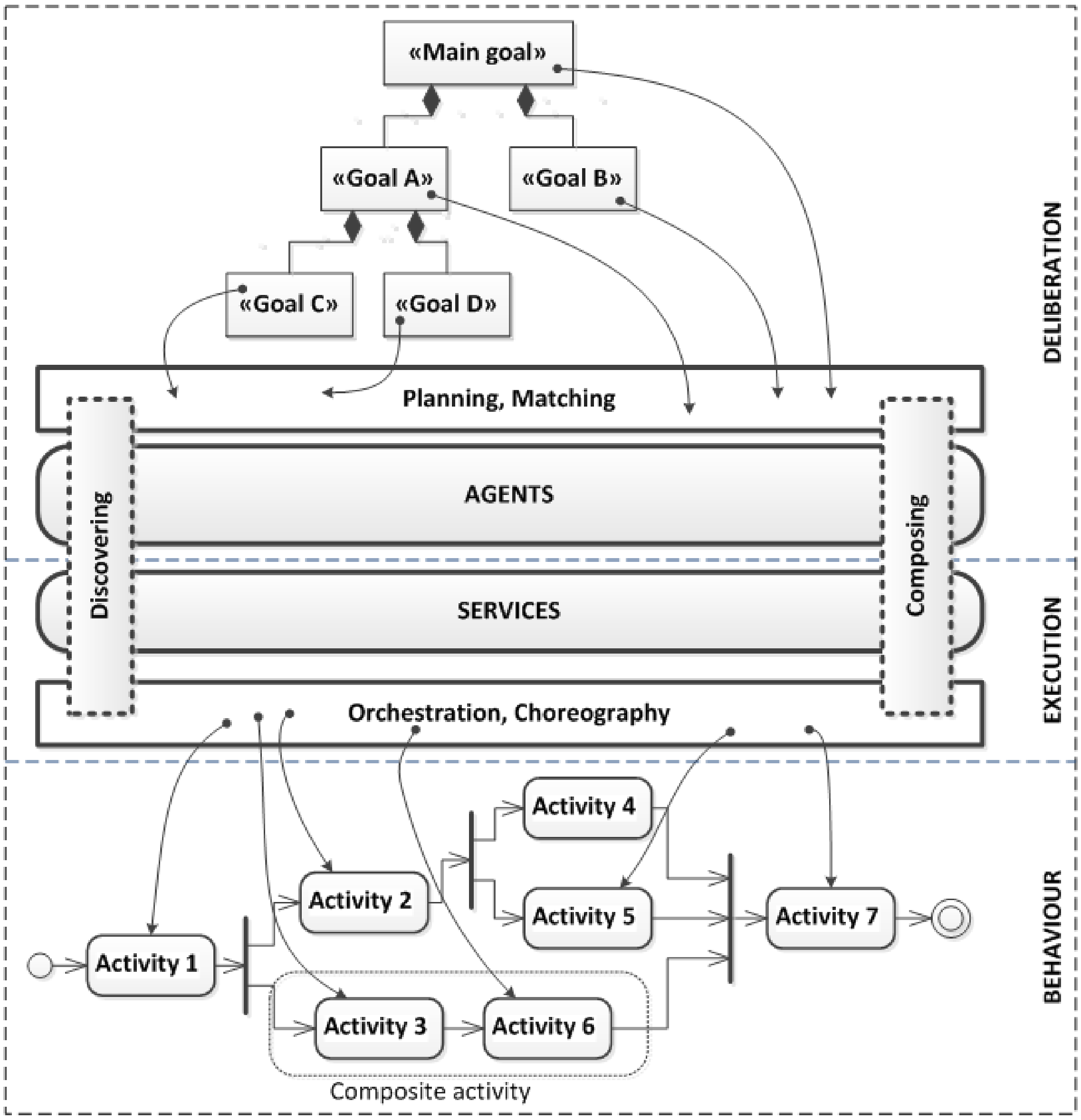


Figure 1. Conceptual view of the service-oriented agent-based architecture.

In the bottom of Figure 1, the activities can be seen as service processes (execution of services, e.g. navigation, manipulation, vision, etc.). They can have a basic or composite structure. The basic processes are indivisible, whereas the composite processes can be decomposed into other activities. This composition of activities or service processes is called orchestration of services. It plays an important role in the system architecture since it can define different encapsulation levels to execute services. On the other hand, choreography of services deals with the messages exchanges among services that are executed in parallel (collaborative nature). Orchestration and choreography are terms from Service-Oriented Architecture (SOA). Based on the conceptual structure presented in Figure 1, a service-oriented agent-based approach is proposed as ICA.

From the robotics viewpoint, missions, goals, planning, matching, and agents correspond to the “deliberation” layer; services, orchestration, and choreography correspond to the “execution” layer; and activities correspond to the “behaviour” layer. For example, in a single-vehicle seabed survey (mission) the main goal is to collect seafloor data from a given exploration area. The agent is an AUV which plans its tasks (diving, path-following, and surface) by means of checking for availability of its capabilities to carry out such tasks. Services for this mission are from navigation, guidance, control, and vision capabilities to carry out activities (also behaviour) such as “dive”, “emerge”, “capture image”, etc.

Table 2 shows the ICA architectural elements, and their different interaction levels. The activities are classified as mission, operation, task, and action. The services are categorized by following the above activities classification. This hierarchical information classification impacts on the knowledge representation and its design. Ontologies are used to represent the knowledge.

Table 2. Interaction levels of the ICA architectural elements.

Integration Level	**Service**	**Physical Entities**	**Logical Entities**	**Maritime Activities**
High	Compositional	Group of vehicles	Holons	Missions
High-mid	Compositional	Vehicles	Agents	Operations
Low-mid	Compositional	Devices	Actors	Tasks
Low	Atomic	Transducers	Workers	Actions

At the lowest level (centre of the Table 2), there are actions from transducers (i.e. sensors and actuators). In the next level up, there are tasks from devices that play a role as actors. Above that, there are operations carried out by vehicles which play a role as agents.

At the highest level (top of the Table 2), there are missions carried out by group of vehicles that play a role as holons (multi-agent interaction). The basic robotics layers are placed between levels.

Figure 2 shows the dependency relations among the key elements of the ICA. The system, i.e. AMRs, fulfils one or more missions (represented by “1..*”), has one or more components (or modules), and use case(s). It also has facilities to sense and act within the environment. Missions are carried out by agents that have one or more goals and plans. A goal is achieved by one or more plans. An agent carries out one or more activities planned according to the platform capabilities, and the goal needs. An activity is carried out by one or more services that encapsulate one or more functionalities of the AMR components. Matching the robotics architecture, functionality means “behaviour”.

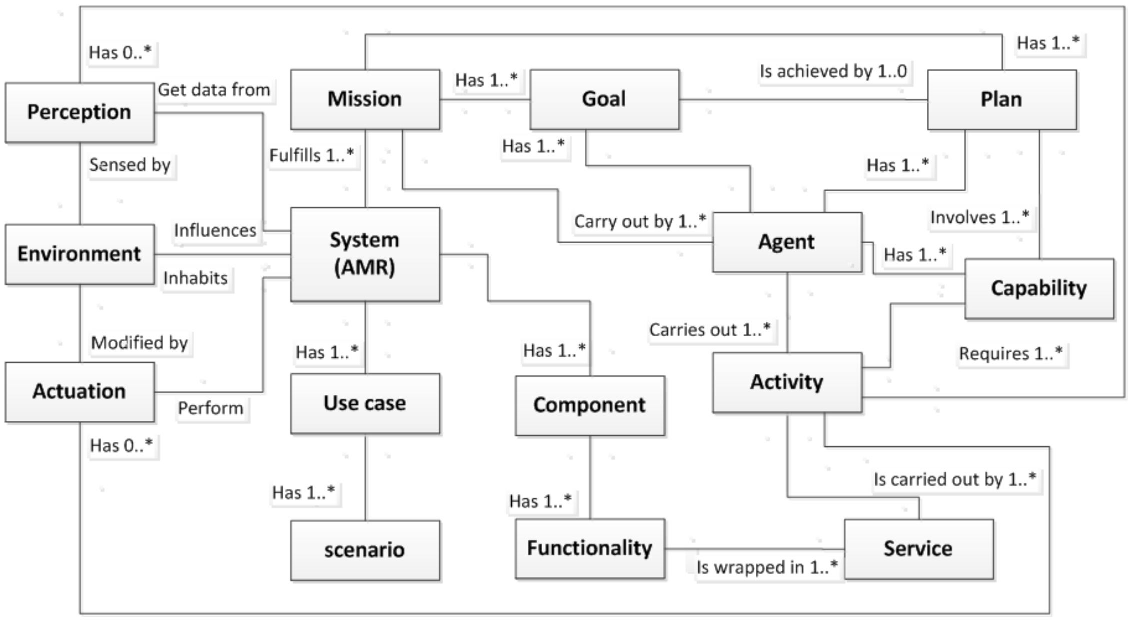


Figure 2. Relationships among the key elements of the ICA.

Following the dependency relations presented in Figure 2, a bottom-to-top development process for the AMR architecture is defined. It begins identifying the functionalities of the platform components (or modules), and ends determining the plans of the agent to achieve the given mission goals. The development steps are as follows.

- Extraction of functionality from platform components (or modules). Grouping and separation of functionalities in order to build clusters of similar functions. Each function can in turn be built of other functions.
- Encapsulation of the above functionalities in basic or composite services gives serviceability to the AMR system. It enables the AMR system to carry out activities (service process or execution of services that encapsulate functionalities) at different interaction levels (mission, operation, task, and action). The activities are based on capabilities derived from the component functionalities.

The capabilities are in turn grouped in order to build an agent. The plan of the agent is built according to the mission goals of the AMR. A database stores the knowledge representation of the entire AMR.

3.2 Hierarchical Control

Figure 3 shows the operation principles of the AMV system. This figure is explained by dividing it into two areas: the top part and bottom part. The former depicts how the system works at the planning level. The latter depicts how the system works at the execution level.

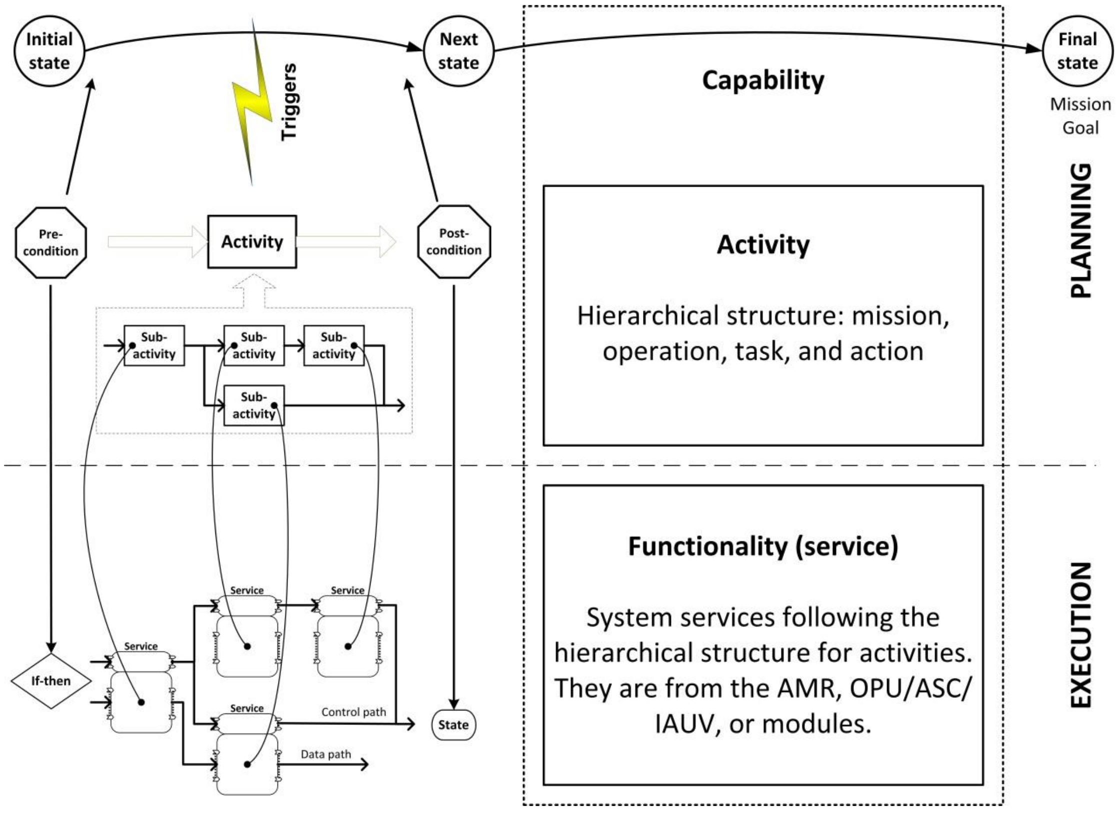


Figure 3. High-low-level agent integration.

Figure 3 presents the existing connections between the planning and execution levels. The concept shown in this figure can also be applied to the internal operation of an agent, i.e. internal planning and execution of agent tasks in a similar way (strategy) as it is happens in a team of agents.

At the left bottom of Figure 3, the network of platform services performs the activities required by the plan (the left top). There is a one-by-one relation between activities and services as shown in the figure. The activities are only triggered when pre-conditions are met. They also generate post-conditions. Pre-conditions are usually evaluated by “if-then” conditional sentences on states of data, and objects. Post-conditions normally result in new states of data, and objects that are used to evaluate the next pre-conditions. Goals are states, so every intermediate state reached can be considered as sub-goals achieved.

On the left of Figure 3 is a description of what a capability is, and the two levels it covers. At the right bottom, the network of platform services is the functionality that the system (AMV), subsystems (OCU, ASC, or IAUV), subsystem nodes, and node components provide. At the right top, the activities are hierarchically categorized as missions, operations, tasks, and actions. Thus, a capability is built of activities and functionalities (services).

Advanced computational systems such as multi-agent systems are suitable to implement biological organizations inspired from social behaviour of their members who can be organized in group, community, etc. according to their role in the system. This enables the system to define an organizational hierarchy, and be part and whole of the system at a time.

Holonic structures offer a powerful abstract modelling for large complex systems. An architectural approach to support the above structure in agency (agent community) with collective behaviour exhibited by groups of agents is by means of holonic systems. The main representational concern in this approach is that interacting agents with particular skills behave as if they were a single entity. Based on the holon concept, elementary entity of a holonic system, groups of agents can be organized in a team of coalesced agents. A holon keeps structural self-similarity by being composed of holons as sub-structures. This hierarchical relationship can be extended recursively, and is called holarchy. Thus, a holon can be seen either as an autonomous individual entity or as a hierarchical organization of sub-holons, according to the viewpoint chosen [35].

Figure 4 shows the hierarchical multi-agent or holonic system defined for TRIDENT [1]. The OCU agent is at a higher control level where it supervises behaviour of the other two agents (ASC and IAUV). Of course, each agent keeps autonomy all the time but in term of organization, the OCU implements organizational techniques to facilitate the interaction among agents, i.e. communication, coordination, cooperation, and collaboration.

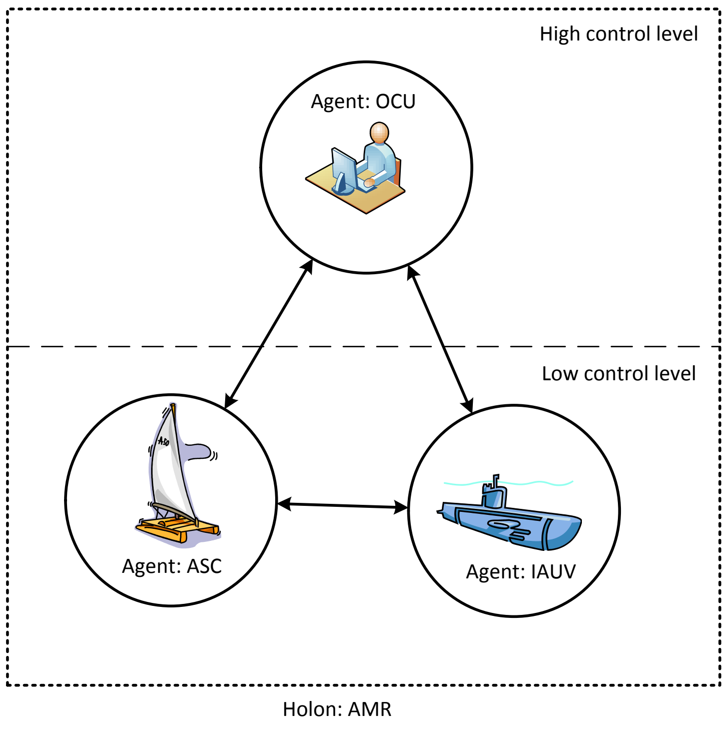


Figure 4. Multi-agent hierarchy

Based on the above holonic structure, the following Subsections describe details of design as to the external behaviour of the AMR agents. They are focused on the mission and operation capabilities provides by the AMR system.

Therefore, planning approach is a global planning for local plan where there are basically two planning: the global plan for the OCU, and the local plans for the ASC and IAUV. They are presented in Section 7.

The foundations of the ICA have multi-disciplinary nature. It comes from the robotics, cognitive science, and computer science. Therefore, the ICA development is based on the following architectural representations: robotic, cognitive, and agentic models. There are currently different reference models for each of the above representational descriptions. In particular, the ICA combines the following approaches.

A **robotic architecture** which is a hybrid approach composed of three-layer architecture (Planning, Sequencing, and Skill) plus a knowledge block; World Model. Figure 5 shows this combined architecture (top left).

A **cognitive architecture** built of two blocks: TBox and ABox which are part of the knowledge representation based on description logics in Figure 5 shows the elements of this cognition process (top right).

An **agentic architecture** based on the Belief-Decide-Intention (BDI) software model. The agent structure is shown in Figure 5 (bottom). It is built of well-defined blocks, i.e. Belief, Desire (goal), and Intention blocks. Additionally, there are Interpreter and Plan blocks.

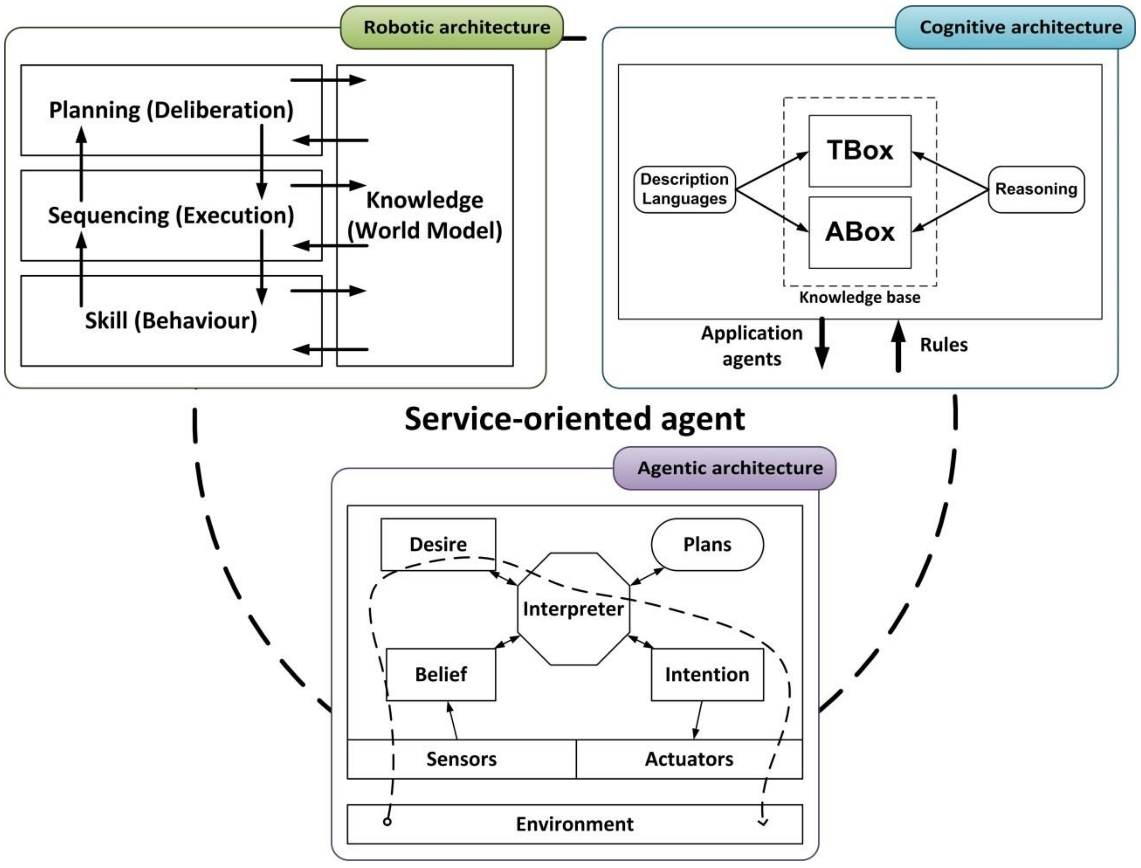


Figure 5. Architectural drivers for the agent structure.

Situation Awareness (SA) is the ability to be aware of and understand what is happening in the surroundings of an agent, both at the present time, and in the future through prediction. This capability allows systems to understand dynamic and complex environments, and operate with them. It can be divided into three separate levels: perception of the elements in the environment, comprehension of the current situation, and projection of future status. SA involves the events, states, condition, and activities of the environment dynamics as to time and space from which some situations arise (in particular those changes that occurred in the environment over some time interval). A situation is defined by a specific state after a sequence of events (with intermediate states, and activities with pre and post conditions). The situation is concerned with the comprehension of the environment features, and with the evolvement of these features over time [36].

SA is essential for decision makers. Within an agent, the decision making cycle is defined by four basic stages: Observation-Orientation-Decision-Action (OODA) loop. The Observation stage is the SA perception level. The Orientation stage takes into account the information acquired from the Observation stage and the knowledge of the agent, to understand the situation (SA comprehension level). The Decision stage is carried out at the SA projection level. The Action stage closes the OODA loop by carrying out actions according to the environmental adaption made in the previous stage.

The mapping of the SA and OODA concepts onto the BDI agent architecture is as follows. The Belief block represents the informational state of the agent, and describes the known state of the world (the world model). It matches the SA perception and comprehension levels or OODA observation and orientation stages. The Desire block represents the motivational state of the agent (goals or situations that the agent would like to accomplish). The Intention block represents the deliberative state of the agent (what the agent has chosen to do). It corresponds to the SA projection level or OODA decision and action stages. The Interpreter block maps to the agent reasoner.

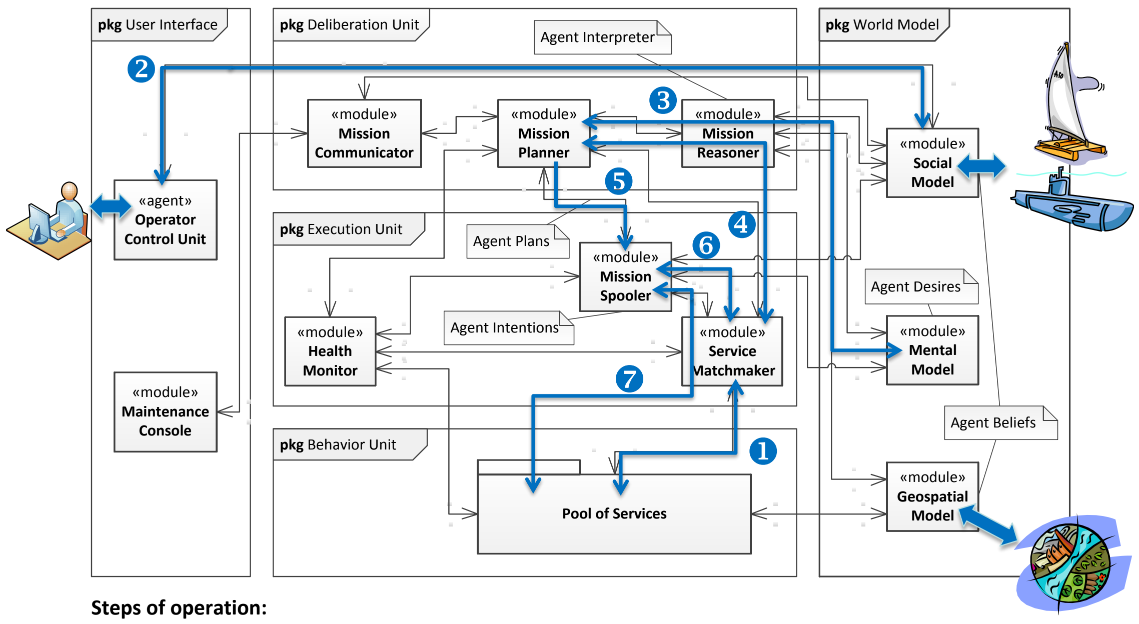
The above approach endows the agent with initiative. Decision making mechanisms are critical for problem-solving processes that are preformed every time an agent receives a mission to be carried out.

The agent anatomy is depicted in Figure 6. It shows the internal structure of the service-oriented agent. There is one agent per marine vehicle. This figure encompasses three architectural models mentioned above:

- A block-layered robotic model as shown in the centre of Figure 6 (linked to the model shown in the top-left of Figure 5) with the following blocks: units of planning (deliberation), sequencing (execution), skill (behaviour), and a world model. In addition, a user interface is taken into account.

- A description-logics model as shown in the top-left of Figure 6 (linked to the model shown in the top-right of Figure 5) which involves the following blocks: deliberation unit (mission reasoner), and world model (mental model; ontology).

- A model with the logical structure of BDI agents as shown in Figure 6 (linked to the model shown in the bottom-centre of Figure 5): beliefs, desires, interpreter, intentions, and plans.



Steps of operation:
❶ Services are advertised and discovery by means of the service matchmaker.
❷ The operator sets the mission, and communicates it to the team of agents (AMVs).
❸ Each agent (AMV) queries itself in order to know how to deal with the given mission.
❹ Capabilities required by the mission are checking for availability.
❺ The planner sends the mission plan to the mission spooler for execution.
❻ The mission spooler checks status of services though the service matchmaker.
❼ The mission spooler executes task as planned by invoking services.

Figure 6. Agent anatomy.

The five main blocks (identified as SysML packages, i.e. “pkg”) in the agent anatomy shown in Figure 6 are:

User interface. The end user is able to deal with the mission, and visualize the mission results through an Operator Control Unit (OCU), e.g. seabed map (image mosaicking), scene and objects characterization, etc.

Deliberation Unit. This has basically three components: the mission communicator, the mission planner and the mission reasoner. The mission communicator, which includes the communication manager (wired and wireless communication channels), communicates with the human operator and with the marine vehicles through the social model. The mission planner, which includes the resource manager, helps to selects the agent capabilities required to take actions according to the decisions made by the agent interpreter. The mission reasoner, which includes the agent interpreter, reads the data perceived from sensors, interprets them according to the knowledge embedded in the mental model, and makes the decision of what to do next. The mission planner output is a plan (list of activities to be carried out by the spooler).

Execution Unit. This is in charge of dealing with the execution of the agent services. The execution is according to the plan generated in the mission planner, and it is executed by the mission spooler. The mission spooler is responsible for parceling out activities listed in the mission plan for execution by platform services. The health monitor deals with the status of the platform services by keeping record of the vital working conditions. It implements the fault diagnosis techniques.

Behaviour Unit. The pool of services of the agent depends on the marine vehicle it is deployed on. They are services at the vehicle level. In the case of the ASC the services provided are: navigation, behaviour management, waypoint list setting, and acoustic/radio communication. In the case of the IAUV the services provided are: navigation, path plan setting, maps generation, seabed data collection, scene/object identification, visual docking, manipulation, grasp specification, and acoustic/radio communication.

World Model. This is a central repository built from the following models. (1) Social Model. It describes the social context which the agent inhabits and interacts with. It is built of the agent directory module which includes the service registry. (2) Mental Model. It describes what the agent is able to know about itself. (3) Geospatial Model. This contains environmental data collected by sensors (perception).

3.3 Knowledge Representation

This Subsection presents architectural aspects of the knowledge representation as well as details of the ontology defined for the ICA.

3.3.1 Cognitive Conceptualization

The knowledge representation in the ICA utilizes ontologies. The main ontology elements are concepts (classes), properties, instances (individuals), and assertions. A concept represents a set of entities or things within a domain. Properties define either relations between an individual and a value, or between two individuals; called data type properties, and object properties, respectively. Knowledge representation based on description logics has a block called TBox which defines the concepts and properties in a domain in addition to specifying terminological axioms for every atomic concept (Figure 7). Axioms are used to constrain the range, and domain of the concepts, e.g. an IAUV is a marine vehicle that has navigation capabilities. A block called ABox contains a finite set of assertions for the classification of individuals, and the properties they have. The combination of the TBox and the ABox forms the knowledge base that can be described with ontologies. Inference over the ontology is provided by a reasoner.

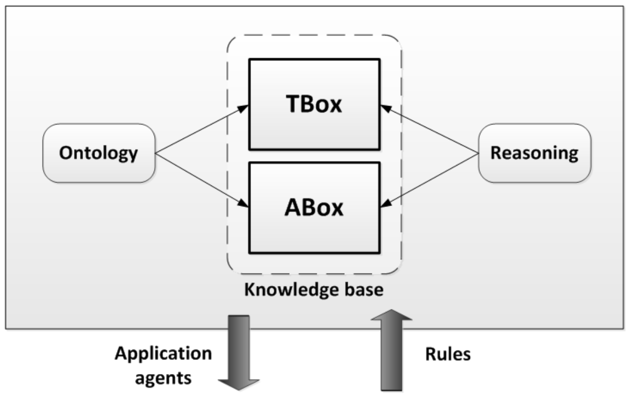


Figure 7. Knowledge representation structure based on description logics.

Situation Awareness (SA) is the ability to be aware of and understand what is happening in the surroundings of an agent, both at the present time, and in the future through prediction [23]. This capability allows systems to understand dynamic and complex environments, and operate with them. It can be divided into three separate levels: perception of the elements in the environment, comprehension of the current situation, and projection of future status [24]. The decision making cycle is defined by four basic stages: Observation-Orientation-Decision-Action (OODA) loop [25]. The Observation stage is the SA perception level. The Orientation stage takes into account the information acquired from the Observation stage and the knowledge of the agent, to understand the situation (SA comprehension level). The Decision stage is carried out at the SA projection level. The Action stage closes the OODA loop by carrying out actions according to the environmental adaption made in the previous stage.

The two above concepts, SA and the OODA loop, are the foundation of AMRs. The SA levels for individual unmanned vehicle systems range from fully human controlled to fully autonomous unmanned capabilities [26].

The ICA is based on agents that apply the above SA and OODA concepts. The agent structure selected for the current approach implements a BDI-based architecture. This architecture is built of well-defined blocks, i.e. Belief, Desire (goal), and Intention blocks. Additionally, there are Interpreter and Plan blocks. The mapping of the SA and OODA concepts onto this architecture is as follows. The Belief block represents the informational state of the agent, and describes the known state of the world (the world model). It matches the SA perception and comprehension levels or OODA observation and orientation stages. The Desire block represents the motivational state of the agent (goals or situations that the agent would like to accomplish). The Intention block represents the deliberative state of the agent (what the agent has chosen to do). It corresponds to the SA projection level or OODA decision and action stages. The Interpreter block maps to the agent reasoner.

There are three ontology levels: foundation/upper, core/domain, and application. The ICA only develops core and application ontologies since the foundational (or upper) ontology represents the very basic principles to ensure reusability across different domains.

3.3.2 Foundation 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 an overlap exists between these different sets of knowledge stores. The approach proposed in this Thesis provides more flexibility in the way the information can be accessed and stored, while being JAUS compliant at the communication level between agents.

Core Ontology

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.

The knowledge concepts identified as essential parts for maritime systems as vehicles, measurable parameters that are 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/effector.
- **Waypoint:** Position in space with coordinate and tolerance.
- **Coordinate:** Local frame, global frame, angular.
- **Velocity:** Linear, angular.
- **Attitude:** Roll, pitch, yaw.

The conceptual structure of the core ontology focuses on the AMR system. The following classification of concepts describes the structure of the core ontology:

- System context
    - Environment (sensing/actuating)
    - Stakeholders (end user interface)
    - Other systems
- System architecture
    - Structural description
        - Composition
            - Data (observation + ...)
            - Software (modules, services, agents)
            - Hardware
            - Mechanics
        - Topology (JAUS; systems, subsystem, nodes, components)
        - Messages (JAUS)
        - System platform elements (group of vehicles, vehicle, device, transducer)
    - Behavioural description
        - Transitions (Events)
        - States
        - Processes (of services or activities; mission, operation, task, action)
- System mission
    - Goals
    - Plans
    - Capabilities (including payload)
    - Targets (physical objects)
- System status

Figure 8 shows the main classes of the Core Ontology. This class is the entry point to the core ontology since the concepts shown are connected to (depend on) the central entity which is called “thing”. This means that any entity (or concept) is a thing. Each of these main entities is developed in details in Appendix A.

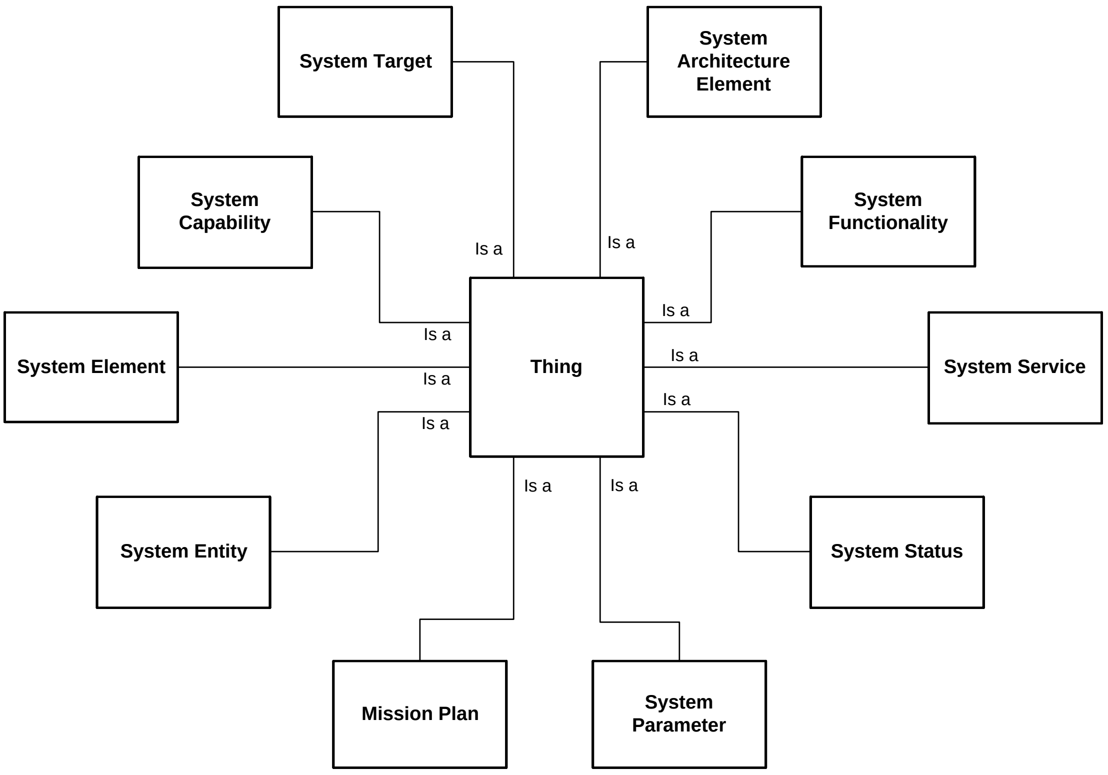


Figure 8. Main classes of the core ontology

Figure 9 shows the relations (properties) among the core ontology individuals (focus on capabilities). The most important cross-entity relation is that between ‘system capability’ and ‘system mission’.

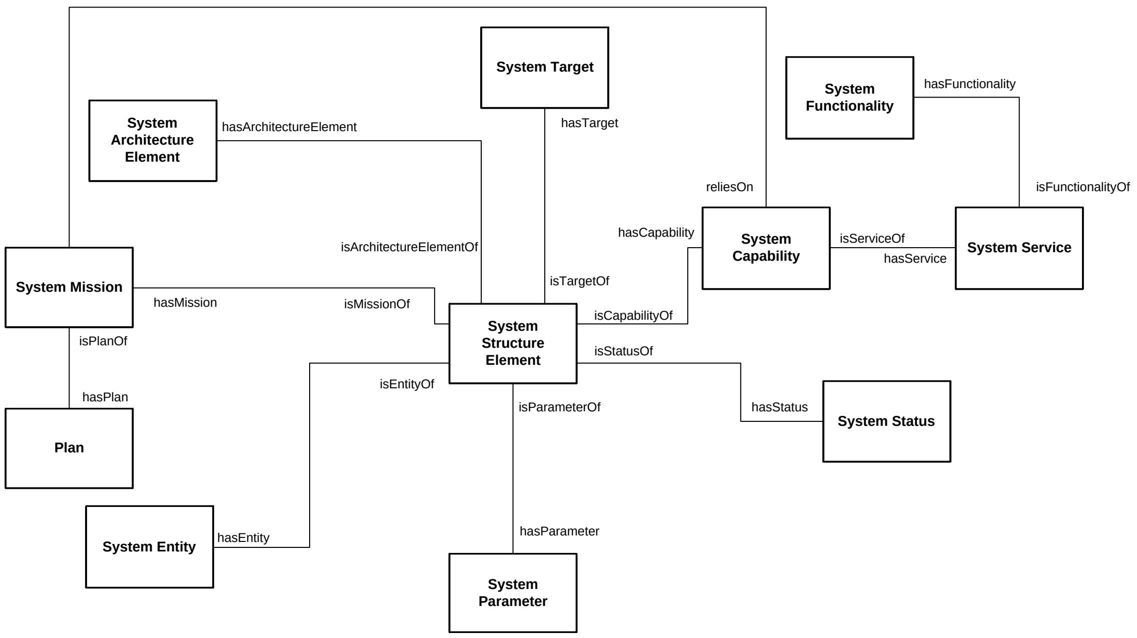


Figure 9. Relations (properties) among the core ontology individuals (focus on capabilities)

Application Ontology

Application concepts are handled at the executive layer and are used to ground the abstract concepts managed by the agents running in the vehicle. Application concepts are specific to the expertise or service provided by each of the intelligent agents. In the case study presented in this Thesis, these agents are the OCU, ASC, and IAUV. These agents make use of the proposed framework and allow the transition from the Deliberative to the Action phase of the OODA loop [25].

The most important concepts identified for service-oriented distributed mission planning are:

- **Resource:** state of an object (physical or abstract) in the environment (vehicle, position, sensor, etc.)
- **Action:** Capability to modify the state of resources (calibrate, classify, explore, etc.)
- **Plan:** A list of time slots containing sequences of instantiated actions
- **Execution:** When an action instantiation is executed successfully

The design of the ontologies encapsulating the knowledge handled by the above agents is described as follows.

The conceptual structure of the application ontology focuses on the AMR system mission. The following classification of concepts describes the structure of the application ontology:

- Missions
  - Goals
  - Plans
  - Core ontology: Capabilities
  - Activities
    - Messages (commands linked to platform capabilities)
    - Core ontology: Services

Figure 10 provides a global and extensible model into which data originating from distinct sources can be mapped and integrated. The knowledge representation in this level is given by a common set of architecture elements and concepts from JAUS.

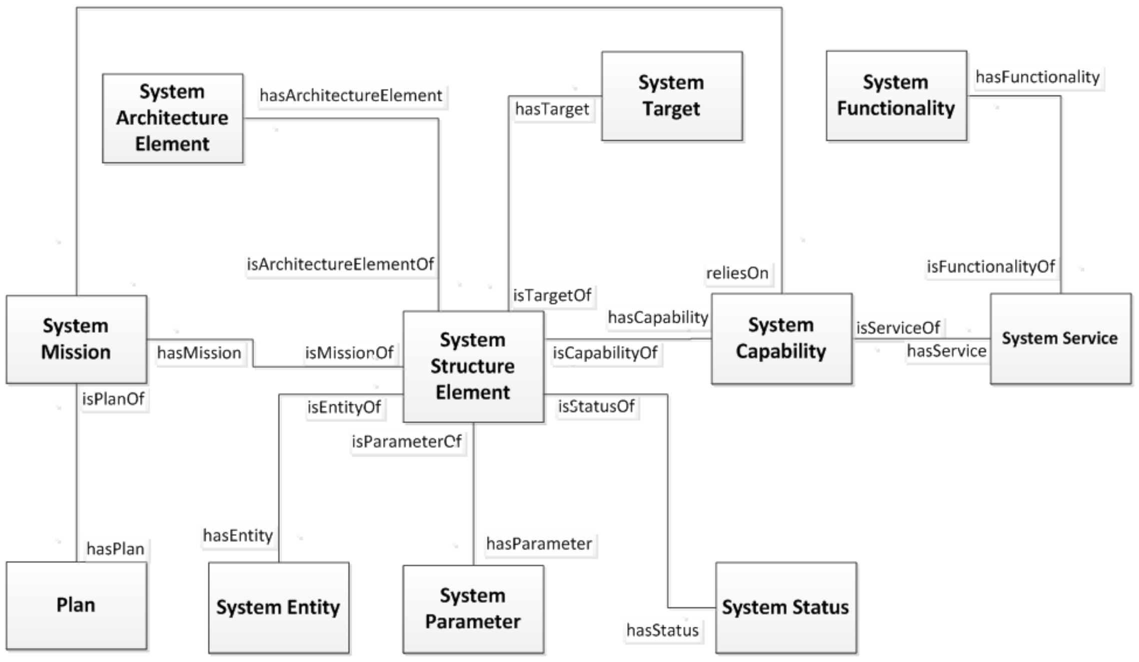


Figure 10. Core ontology.

The application ontology (Figure 11) provides an underlying formal model for tools that integrate source data, and perform a variety of extended functions. The application concepts are handled at the executive layer and are used to ground the abstract concepts managed by the agents running in the vehicle.

Application concepts are specific to the expertise or service provided by each of the intelligent agents. In the case study presented in this Thesis, these agents are two marine vehicles, and the operator console. These agents make use of the proposed framework and allow the transition from the Deliberative to the Action phase of the OODA loop [23].

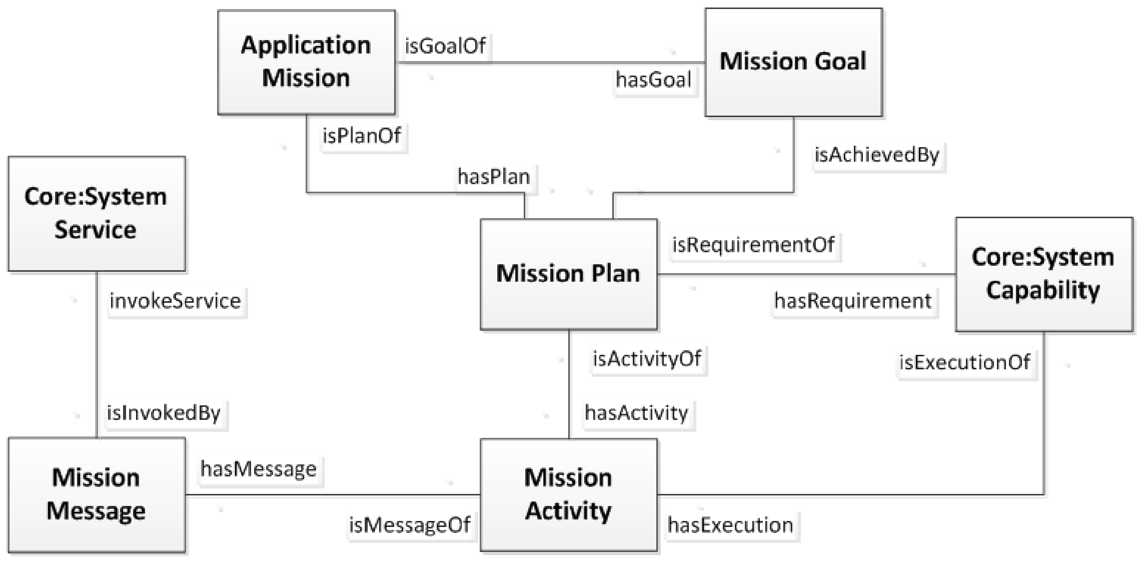


Figure 11. Application ontology.

3.4 Knowledge Reasoning

The human operator sets the mission to be carried out through the OCU. He/she commands this order to communicate to the maritime vehicles (ASC and IAUV), through the mission communicator to the mission planner, the mission assigned. After setting the mission to be carry out, many questions come up. The first question is that to know whether or not a maritime vehicle is really able to carry out such a mission of part of it. The answer to this question comes from the maritime vehicle that responds based on knowledge about itself as a potential platform suitable (capabilities represented by the pool of services) of successfully performing the tasks required. Then, the following questions are: what capabilities are required from the vehicle platform? Can the vehicle do the job (mission) in a time period? Etc. These questions are made by means of the reasoner that interacts with the mental model in order to know the answer. Then, the answer is passed to the mission planner which begins to make the plan based on the information obtained from reasoning and the social and geospatial models.

Initially, two possible approaches for planning based on knowledge representation (same ontological database for the OCU, ASC, and IAUV) are proposed:

Built-in plans. Description of the predefined plans in the ontological database, retrieval of the plan, and then checking capabilities supported by the vehicles to execute the plan.

Built-on-demand plans. Build the plans based on queries performed against the ontological database by using a forward search algorithm. Then, check consistency against the capabilities available in the system platform.

3.4.1 Built-in Plans

The queries to be performed against the ontological database in order to deal with built-in plans are in the following order.

1. Does the marine vehicle have any predefined plan to tackle the mission operation given? If so, retrieve the plan, and go to the next query. If not, the marine vehicle is not able to carry out the mission operation due to lack of plan, and then notify it to the rest of the system. The query select statement to answer this question is as show in Query 1.

$$\boldsymbol{SELECT}\ ?\,Mission\ ?\,Operation\ ?\,Plan$$
$$\boldsymbol{WHERE}\ \{applicat0ion{:}\,hasOperation(?\,Mission\ ?\,Operation)\ \wedge$$
$$application{:}\,hasPlan(?\,Operation)$$

Query 1. Formal search sentence for predefined plan.

2. Does the marine vehicle have the capabilities to implement the plan retrieved? If so, return successfully, and go to the next query. If not, the marine vehicle is not able to carry the mission operation out due to lack of one or more capabilities need, and then notify it to the rest of the system. The query select statement to answer this question is as show in Query 2.

$$\boldsymbol{SELECT}\ ?\,Platform\ ?\,Capability$$
$$\boldsymbol{WHERE}\ \{rdf{:}\,type(\ ?\,Platform, core{:}\,Platform)\ \wedge$$
$$core{:}\,hasCapability(?\,Platform\ ?\,Capability)$$

Query 2. Formal search sentence for capabilities in the platform (marine vehicle).

3. What are the pre-conditions and post-conditions of every plan activity? Retrieve pre-conditions and post-conditions according to the activities specified in the plan. The query select statement to answer this question is as show in Query 3.

$$\textbf{SELECT}\ ?Activity\ ?Plan$$

$$\textbf{WHERE}\ \{application{:}\ hasPrecondition(?Activity\ ?Plan)\ \wedge$$

$$application{:}\ hasPostcondition(?Activity\ ?Plan)$$

Query 3. Formal search sentence for pre and post conditions of activities.

The reasoning algorithm for the built-in planning is shown in Algorithm 1 where m is a mission, s is a state reached in a plan, π is a plan, c is a capability, o is an operation, a_i is the ith activity, A is a set of activities, *prec* is a pre-condition, and *postc* is a post-condition.

```
$s \leftarrow s_0$
$\pi \leftarrow \emptyset$
$c \leftarrow \emptyset$
$\pi \leftarrow query1(m, o, \pi)$
while s $\neq g$ do {
  if $\pi \neq \emptyset$ then
    for each $c.a_i \in \pi$ do {
      $available \leftarrow query2(c.a_i)$
      if $available$ then {
        $c.a.prec \leftarrow query3(c.a_i, \pi).prec$
        $c.a.postc \leftarrow querey3(c.a_i, \pi).postc$
        $s = c.a.postc$
      }
    }
  $return\ failure$
}
```

Algorithm 1. Reasoning logic for the built-on planning.

3.4.2 Built-on-demand Plans

The queries to be performed against the ontological database in order to deal with built-on-demand plans are in the following order.

1. Does the marine vehicle have any plan to tackle the mission operation given? To answer this question a search algorithm performs queries on the ontological database in search of activities that satisfy the intermediate goals placed between the initial goal and the end goal (mission goal). The first activity chosen is one that has the initial goal as a pre-condition, the second activity is one that has the post-condition of the first one as a pre-condition, and so on. Thus, sub-goals are chained by listing activities in a certain order. If it is possible to go from the

initial goal to the end goal by means of selecting activities, then a plan can be defined; go to the next query. If not, the marine vehicle is not able to carry the mission operation out due to lack of a plan, and then notifies the rest of the system. The query select statement to answer this question is as show in Query 4.

$$\boldsymbol{SELECT}\ ?\,Prestate\ ?\,Activity$$

$$\boldsymbol{WHERE}\ \{application\!:\!hasActivity(?\,Activity\ ?\,Prestate)\}$$

Query 4. Formal search sentence to build the plan

2. Does the marine vehicle have the capabilities to implement the plan retrieved? If so, return successfully, and go to the next query. If not, the marine vehicle is not able to carry the mission operation out due to lack of one or more capabilities need, and then notify it to the rest of the system. The query select statement to answer this question is as show in Query 5.

$$\boldsymbol{SELECT}\ ?\,Platform\ ?\,Capability$$

$$\boldsymbol{WHERE}\ \{rdf\!:\!type(\ ?\,Platform, core\!:\!Platform)\ \wedge$$

$$core\!:\!hasCapability(?\,Platform\ ?\,Capability)$$

Query 5. Formal search sentence for capabilities in the platform (marine vehicle).

The reasoning algorithm for the built-on-demand planning is shown in Algorithm 2.

```
$s \leftarrow s_0$
$\pi \leftarrow \emptyset$
$A \leftarrow \emptyset$
$c \leftarrow \emptyset$
while $s \neq g$ do {
   $A \leftarrow query4(s, a)$
   if $A \neq \emptyset$ then $return\ failure$
   $nondeterministically\ choose\ a \in A$
   $s \leftarrow a.postc$
   $\pi \leftarrow \pi.a$
}
for each $a_i\ \in\ \pi$ do {
   $available \leftarrow query5(c.a_i)$
   if $not\ available$ then $return\ failure$
}
```

Algorithm 2. Reasoning logic for the built-on-demand planning.

After answering the above questions, and in either planning approach, the mission reasoner gets back to mission planner in order to generate the plan.

3.5 Goal-Driven Capability-Based Planning

The planning paradigm is time-constrained with activity scheduling according to resource availability. The planning nature comes from classical planning with classical representation [31]. In addition, the planning control strategy is based on Hierarchical Task Network (HTN).

The initial proposal chosen to approach the internal agent planning is very simple. It is inspired by classical approach such as the state-space planning with forward search. The main difference between the classical planning approach considered and the one proposed is that the search mechanism is replaced by a more complex paradigm of search based on reasoning.

Figure 12 shows a comparison between the above planning approaches. A block diagram corresponding to the classical planning is shown on the left and a block diagram corresponding to the planning proposed is shown on the right of the figure.

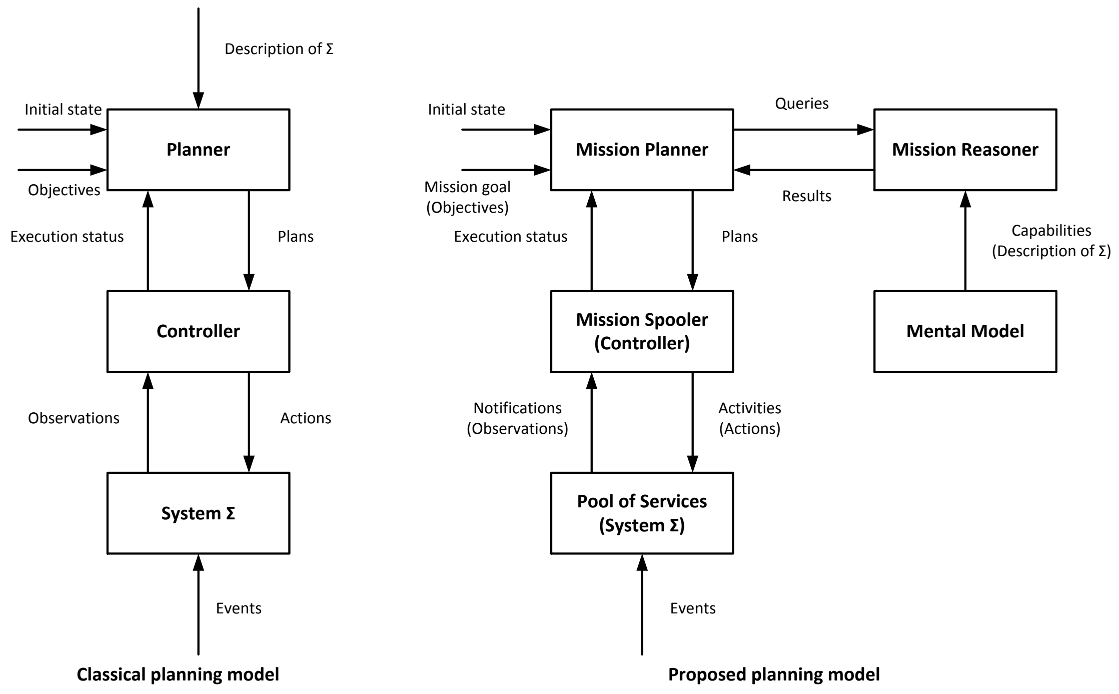


Figure 12. Conceptual planning model comparison

The main difference between the two planning approaches is that for the description of Σ (set of plans). In a traditional planning model (on the left of Figure 12) it is set by the user of the system (AMR system operator). In the proposed planning model such description is embedded in the system (AMR system) as knowledge in the ontological database. The system controller is instructed by the planner to carry out the task-based plan through activities (actions).

The description of the AMR system is given by the ontological database. The reasoner queries this ontology in search of solutions for the planning problem. The initial state of the system is given by the initial states of the marine vehicles, i.e. ASC and IAUV. The objectives, in a more general way are represented by goals, where the main goal matches the mission goal that can be divided into sub-goals.

Once the plan is initially pre-defined with the information obtained from the ontological database, the mission planner checks the plan consistency in terms of capabilities available in the system platform.

Chapter 4: Architecture Design

4.1 User and System Requirements

This Subsection presents how the user and system requirements are defined and specified by analysing different representation models.

4.1.1 Overview of the System Context

This first part of Section 4 summaries the basic ideas and aims for the AMR system in order to extract key requirements from the user and the AMR system for the intelligent control system architecture. In addition, it also addresses the background, framework conditions, and other relevant information from the project environment.

The aim for the AMR system is to perform the missions given by the stakeholder. These missions are described in Subsection 5.3 of this Thesis. To achieve the mission goals, the AMR platform is required to have certain capabilities (generally, multipurpose dexterous manipulation capabilities for intervention operations in unknown, unstructured and underwater environments).

The design and development of the embedded knowledge representation framework and the high-level reasoning agents is required in order to enabling autonomy and on-board decision making of the marine vehicles. To empower the agent technology, the agents are required to be service-oriented entities so that they have the operational flexibility given by SOA, i.e. it provides plug & play facilities that make it possible to integrate agents' capabilities and facilities the diagnosis of available capabilities.

4.1.2 Stakeholder Requirements

The primary stakeholder of the AMR is the end user. The end-user requirements (what the system should be able to do) are the following:

- The platform components should be able to advertise their capabilities.
- The system should discover all the capabilities of the platform.
- The user and marine vehicles should collaborate with each other in order to achieve goals.
- The marine vehicles should be autonomous. At least a high degree of autonomy.
- The system should be automatically reconfigurable in order to deal with changes when planning missions.
- The marine vehicles should be able to communicate with each other and the operator.
- The vehicles of the system should provide internal knowledge representation about the context where it is working.

4.1.3 Systems Requirements

Figure 13 shows the AMR system context. The AMR system inhabits an environment where it interacts with the end user and probably with other systems. As above mentioned, the AMR system has one or more use cases where it fulfils missions. Every system has its architecture. In particular, the intelligent control architecture of the AMR is developed in the current document.

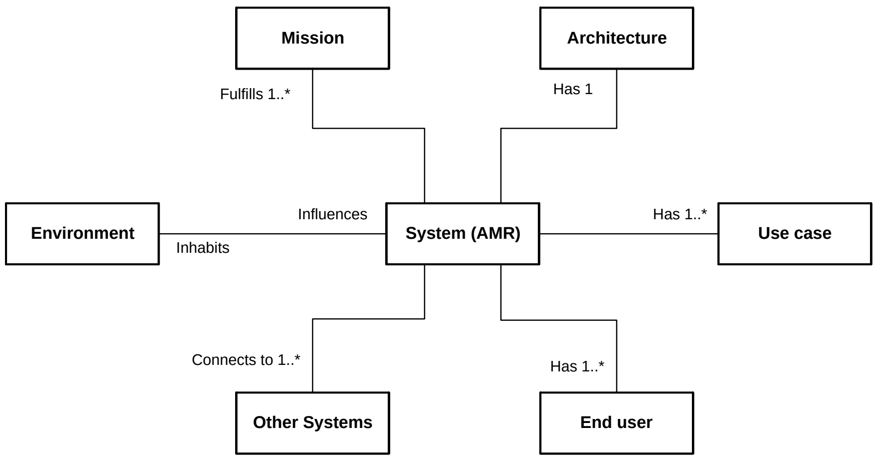


Figure 13. System context

Table 3 presents the External AMR system interfaces with the environment. It basically shows the perceptions, and actuations by means of which the AMR system interacts with the near environment, the end user, and other systems.

Table 3. AMR system interfaces

	Perception	Actuation	Interaction with
OCU	End user input	Scene and object display	End user
ASC	Local pose	Wrench effort command*	
	Velocity state		Environment
IAUV	Local pose	Wrench effort command*	
	Velocity state		
	Scanned image		
	Visual image		
Manipulator	Manipulator joint position	Join effort setting	
	Manipulator end-effector pose	End-effector pose setting	

* A wrench effort command is to guide the AMV to waypoints [22].

As mentioned in Chapter 1, the mission proposed by TRIDENT [1] for the AMR system is a multipurpose generic intervention. It is divided into two phases: survey and manipulation. Following the proposal, as system requirement for the intelligent control architecture, the above missions are implemented as three different sub-missions: seabed survey, target selection, and object manipulation.

Figure 14 shows a network of goal and sub-goals for the above three sub-missions. The main mission has a goal called *Underwater Intervention* and it can be decomposed in three sub-goals called *Vehicles Positioning for Survey, Path/Terrain Following, and IAUV Docking* respectively. The former can be in turn decomposed in two sub-goals called *ASC Positioning*, and *IAUV Positioning*. The target selection goal can be achieved if three sub-goals are reached. They are called *Seabed Image Mosaicing, View and Object Characterization*, and *Grasp Specification*. The latter main mission goal can be decomposed in four sub-goals called *Vehicles Positioning for Manipulation* (decomposed in *ASC Positioning*, and *IAUV Positioning), Object Search*, *Object Intervention* (decomposed in *Floating Manipulation/Station Keeping)*, and *Object Grasping)*, and *IAUV Docking*.

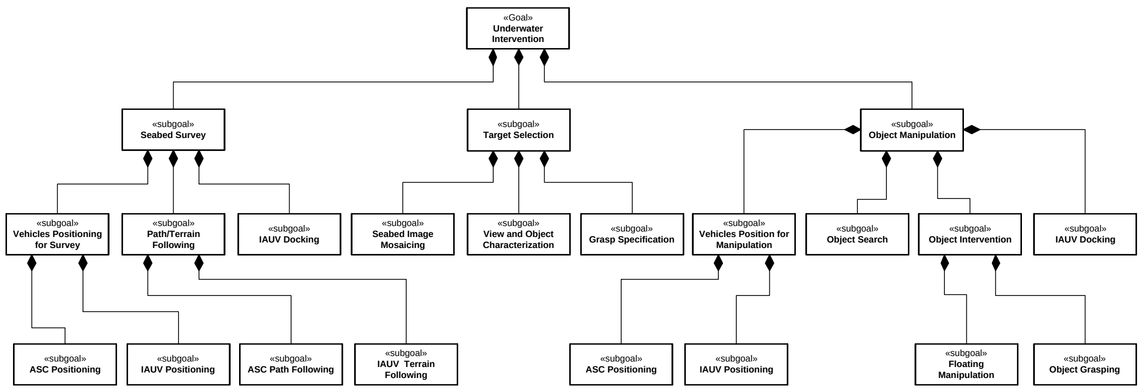

«Goal» Underwater Intervention
«subgoal» Seabed Survey
«subgoal» Target Selection
«subgoal» Object Manipulation
«subgoal» Vehicles Positioning for Survey
«subgoal» Path/Terrain Following
«subgoal» IAUV Docking
«subgoal» Seabed Image Mosaicing
«subgoal» View and Object Characterization
«subgoal» Grasp Specification
«subgoal» Vehicles Position for Manipulation
«subgoal» Object Search
«subgoal» Object Intervention
«subgoal» IAUV Docking
«subgoal» ASC Positioning
«subgoal» IAUV Positioning
«subgoal» ASC Path Following
«subgoal» IAUV Terrain Following
«subgoal» ASC Positioning
«subgoal» IAUV Positioning
«subgoal» Floating Manipulation
«subgoal» Object Grasping

Figure 14. Goals and sub-goals for the selected mission

The high-level functionalities are defined based on the goal and sub-goals shown in Figure 14. They can be seen as capabilities at the mission and operation levels. To achieve the main mission goal (top of Figure 14) the AMR system is required to have functionality at the mission level in order to achieve the *Underwater Intervention* goal. At the operation level, the AMR system is required to have functionality to achieve the *Vehicles Positioning for Survey, Path/Terrain Following, and IAUV Docking*. Thus, there are two high capability levels are defined: mission and operation capabilities.

The following use cases are identified for the AMR system based on the above mission goals (Figure 15). There are three use cases: seeded survey, target selection, and object manipulation. They are the AMR system sub-missions defined above.

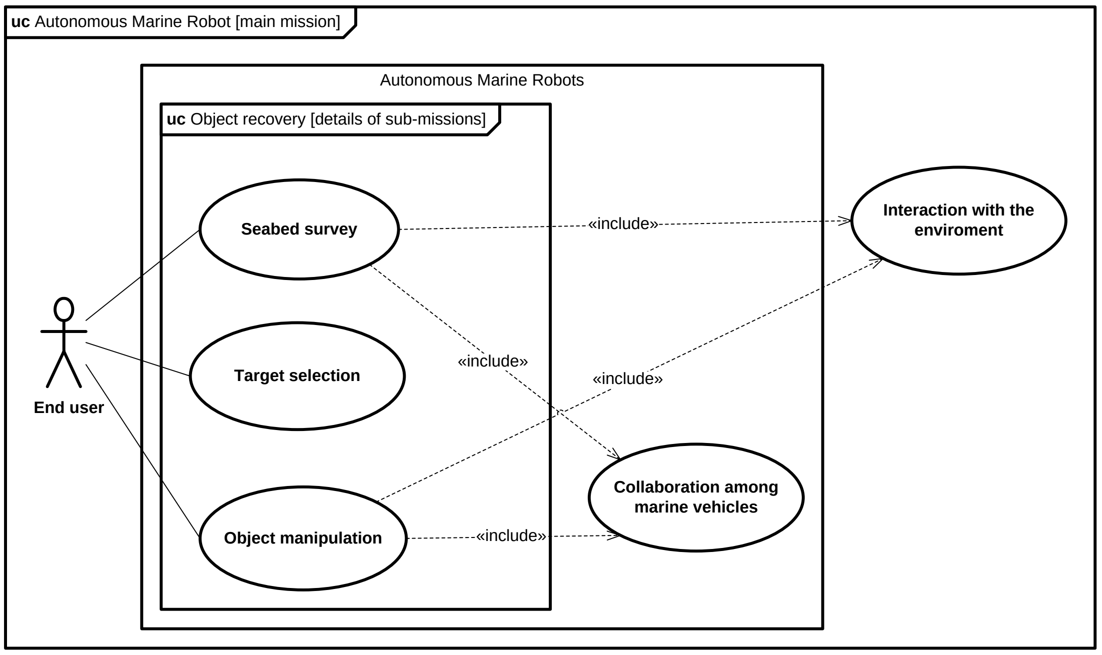

uc Autonomous Marine Robot [main mission]
Autonomous Marine Robots
uc Object recovery [details of sub-missions]
Seabed survey
Target selection
Object manipulation
End user
«include»
«include»
«include»
«include»
Interaction with the enviroment
Collaboration among marine vehicles

Figure 15. System use cases

The AMR system actors (SysML actors) are OCU (end user), ASC, and IAUV. Figure 16 shows the scenario and interaction among AMR system actors for the seabed survey sub-mission.

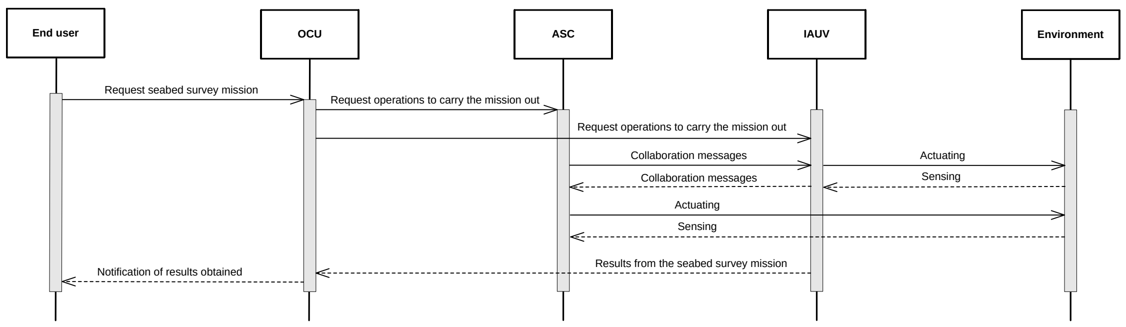

End user
OCU
ASC
IAUV
Environment
Request seabed survey mission
Request operations to carry the mission out
Request operations to carry the mission out
Collaboration messages
Collaboration messages
Actuating
Sensing
Actuating
Sensing
Results from the seabed survey mission
Notification of results obtained

Figure 16. Scenario for the seabed survey sub-mission

Figure 17 shows the scenario and interaction among AMR system actors for the target selection sub-mission.

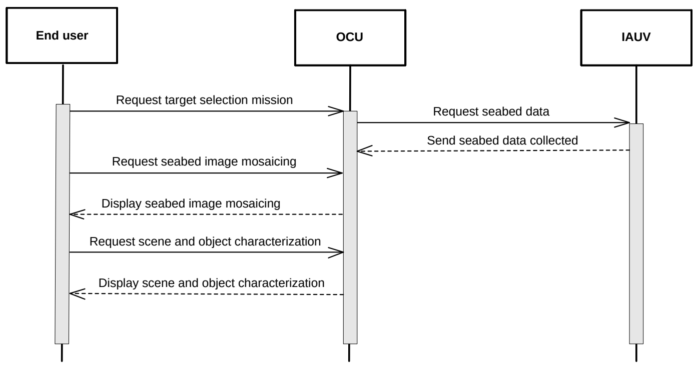

End user
OCU
IAUV
Request target selection mission
Request seabed data
Send seabed data collected
Request seabed image mosaicing
Display seabed image mosaicing
Request scene and object characterization
Display scene and object characterization

Figure 17. Scenario for the target selection sub-mission

Figure 18 shows the scenario and interaction among AMR system actors for the object manipulation sub-mission.

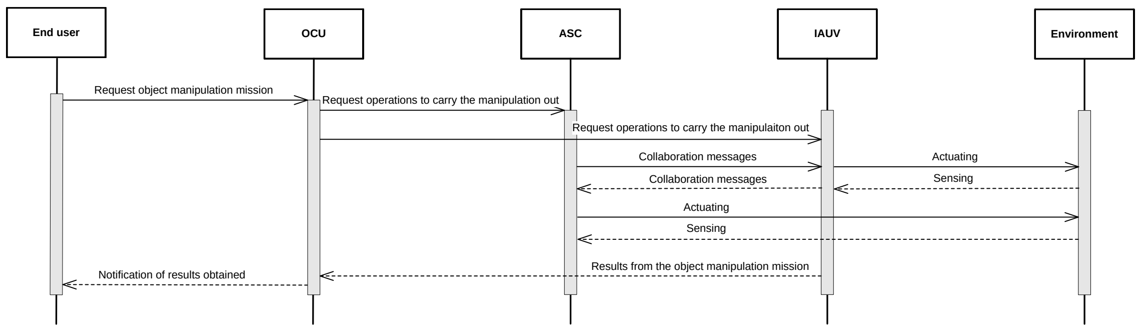

End user
OCU
ASC
IAUV
Environment
Request object manipulation mission
Request operations to carry the manipulation out
Request operations to carry the manipulaiton out
Collaboration messages
Collaboration messages
Actuating
Sensing
Actuating
Sensing
Results from the object manipulation mission
Notification of results obtained

Figure 18. Scenario for the object manipulation sub-mission

Alternative scenarios are defined in the case something goes wrong. Figure 19 shows alternative scenarios for the seabed survey mission when a seabed obstacle is found in the path of the IAUV. There is a need for re-planning of the mission in such a case. The interaction among actor to deal with it is shown inside the diagram box called *Re-planning requirement*.

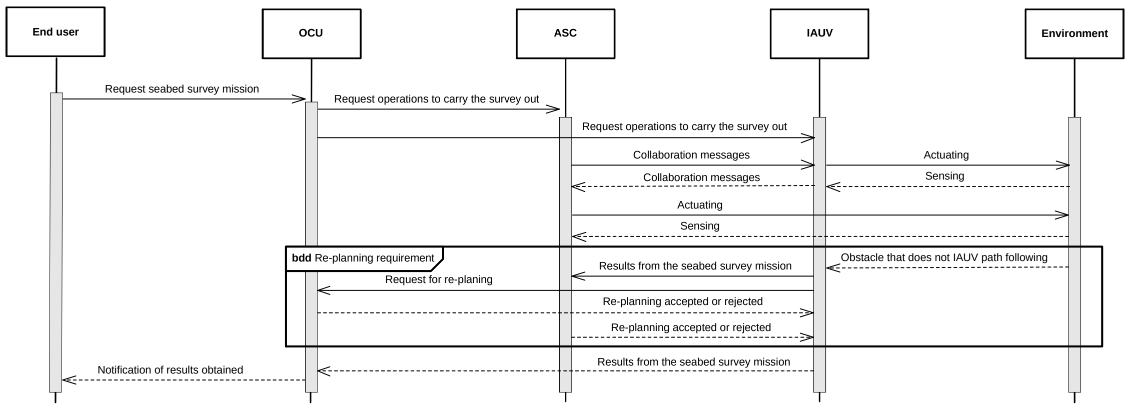

End user
OCU
ASC
IAUV
Environment
Request seabed survey mission
Request operations to carry the survey out
Request operations to carry the survey out
Collaboration messages
Collaboration messages
Actuating
Sensing
Actuating
Sensing
bdd Re-planning requirement
Results from the seabed survey mission
Obstacle that does not IAUV path following
Request for re-planing
Re-planning accepted or rejected
Re-planning accepted or rejected
Results from the seabed survey mission
Notification of results obtained

Figure 19. Alternative scenario for re-planning seabed survey mission (seabed obstacle found in the path)

Figure 20 shows alternative scenarios for the object manipulation mission when a grasp strategy fails. There is a need for re-planning of the mission in order to choose and try another grasp strategy which is, in this case, successful. The interaction among actor to deal with it is shown inside the diagram box called *Re-planning requirement*.

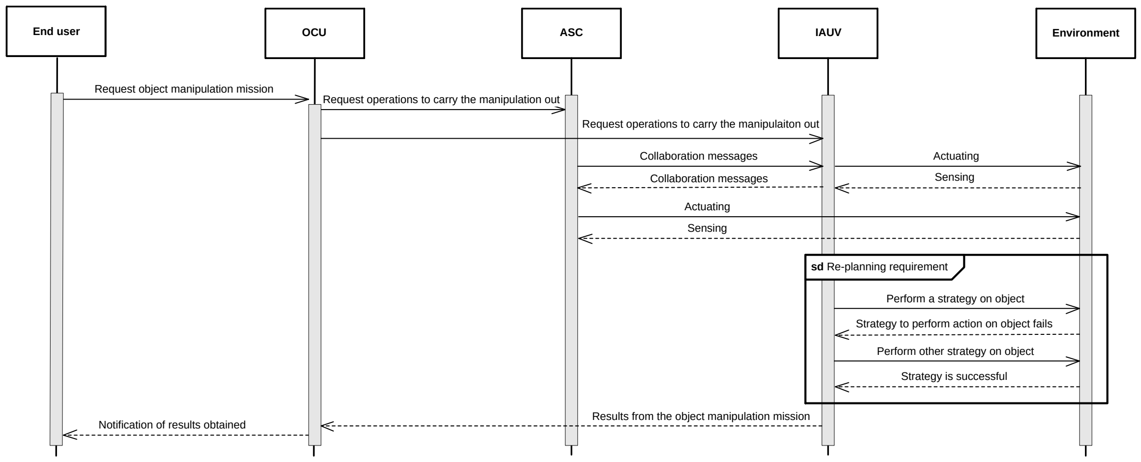

End user
OCU
ASC
IAUV
Environment
Request object manipulation mission
Request operations to carry the manipulation out
Request operations to carry the manipulaiton out
Collaboration messages
Collaboration messages
Actuating
Sensing
Actuating
Sensing
sd Re-planning requirement
Perform a strategy on object
Strategy to perform action on object fails
Perform other strategy on object
Strategy is successful
Results from the object manipulation mission
Notification of results obtained

Figure 20. Alternative scenario for re-planning object manipulation mission (grasp strategy fails)

4.2 Architectural System Design

This Section presents the specification for the intelligent control architecture. The outcomes expected are the full specification of the AMR system components, its low-level functionalities and data coupling. The AMR system specification is done according to the JAUS reference architecture specification [22].

4.2.1 AMR System Components

Figure 21 shows a hierarchical representation of the different AMR system components and how they are organized according to the JAUS standard.

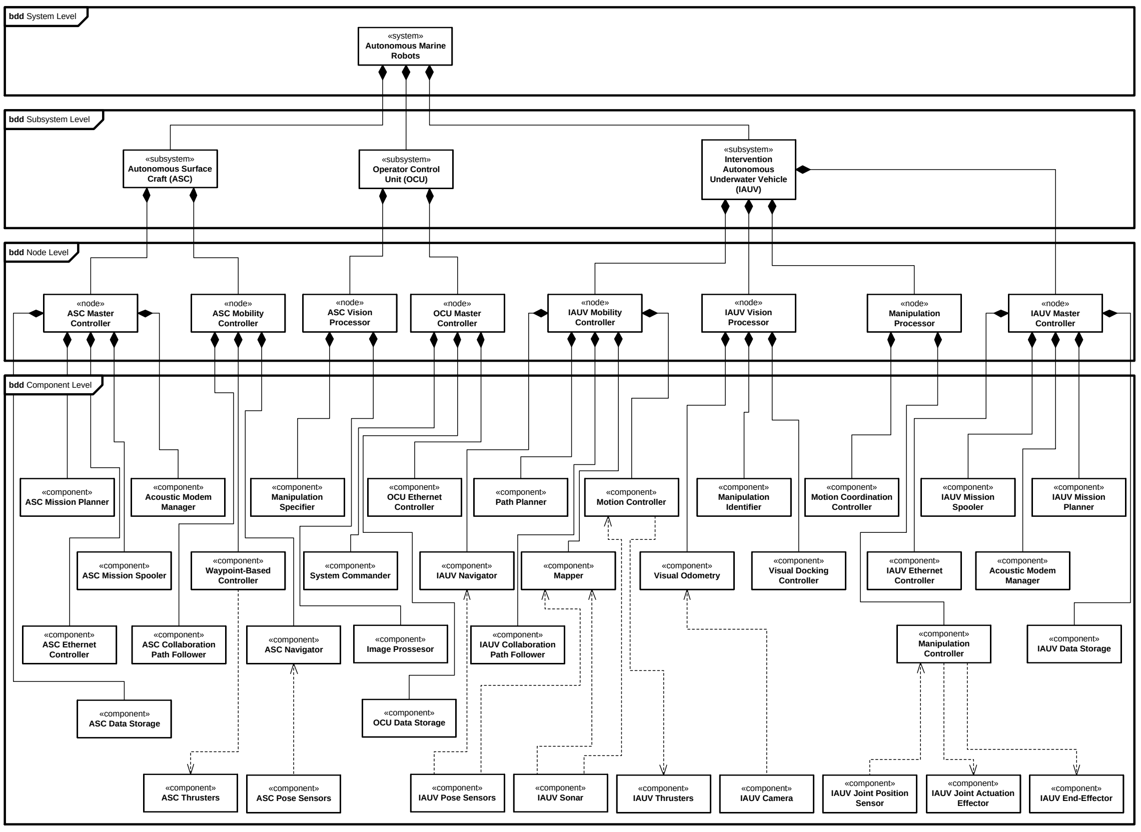

bdd System Level
«system» Autonomous Marine Robots
bdd Subsystem Level
«subsystem» Autonomous Surface Craft (ASC)
«subsystem» Operator Control Unit (OCU)
«subsystem» Intervention Autonomous Underwater Vehicle (IAUV)
bdd Node Level
«node» ASC Master Controller
«node» ASC Mobility Controller
«node» ASC Vision Processor
«node» OCU Master Controller
«node» IAUV Mobility Controller
«node» IAUV Vision Processor
«node» Manipulation Processor
«node» IAUV Master Controller
bdd Component Level
«component» ASC Mission Planner
«component» Acoustic Modem Manager
«component» Manipulation Specifier
«component» OCU Ethernet Controller
«component» Path Planner
«component» Motion Controller
«component» Manipulation Identifier
«component» Motion Coordination Controller
«component» IAUV Mission Spooler
«component» IAUV Mission Planner
«component» ASC Mission Spooler
«component» Waypoint-Based Controller
«component» System Commander
«component» IAUV Navigator
«component» Mapper
«component» Visual Odometry
«component» Visual Docking Controller
«component» IAUV Ethernet Controller
«component» Acoustic Modem Manager
«component» ASC Ethernet Controller
«component» ASC Collaboration Path Follower
«component» ASC Navigator
«component» Image Prossesor
«component» IAUV Collaboration Path Follower
«component» Manipulation Controller
«component» IAUV Data Storage
«component» ASC Data Storage
«component» OCU Data Storage
«component» ASC Thrusters
«component» ASC Pose Sensors
«component» IAUV Pose Sensors
«component» IAUV Sonar
«component» IAUV Thrusters
«component» IAUV Camera
«component» IAUV Joint Position Sensor
«component» IAUV Joint Actuation Effector
«component» IAUV End-Effector

Figure 21. Composite view of the JAUS-compliant system structure

The JAUS reference architecture specification defines a composite topology where a system is built of subsystem(s). A subsystem is built of node(s), and a node is built of component(s). The components are grouped according to their functionalities in order to build nodes, e.g. node *Vision Processor* is built of the following components: *Visual Odometry, Manipulation Identifier*, and *Visual Docking Controller*.

Figure 22 shows how the AMR system components are connected with each other. The small arrows placed in the component boxes show the direction of the data flow.

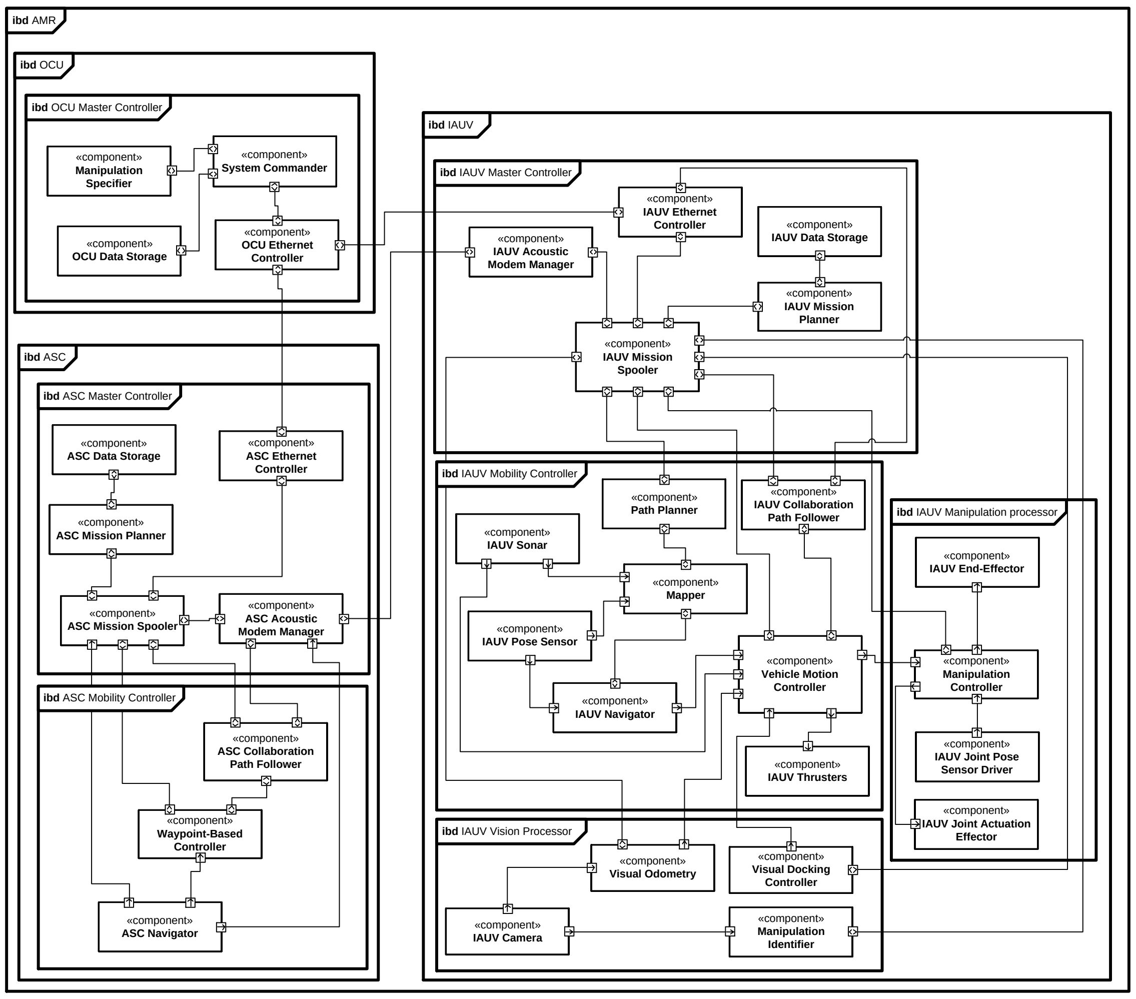

ibd AMR
ibd OCU
ibd OCU Master Controller
«component» Manipulation Specifier
«component» System Commander
«component» OCU Data Storage
«component» OCU Ethernet Controller
ibd ASC
ibd ASC Master Controller
«component» ASC Data Storage
«component» ASC Ethernet Controller
«component» ASC Mission Planner
«component» ASC Mission Spooler
«component» ASC Acoustic Modem Manager
ibd ASC Mobility Controller
«component» ASC Collaboration Path Follower
«component» Waypoint-Based Controller
«component» ASC Navigator
ibd IAUV
ibd IAUV Master Controller
«component» IAUV Ethernet Controller
«component» IAUV Acoustic Modem Manager
«component» IAUV Data Storage
«component» IAUV Mission Planner
«component» IAUV Mission Spooler
ibd IAUV Mobility Controller
«component» Path Planner
«component» IAUV Collaboration Path Follower
«component» IAUV Sonar
«component» Mapper
«component» IAUV Pose Sensor
«component» Vehicle Motion Controller
«component» IAUV Navigator
«component» IAUV Thrusters
ibd IAUV Manipulation processor
«component» IAUV End-Effector
«component» Manipulation Controller
«component» IAUV Joint Pose Sensor Driver
«component» IAUV Joint Actuation Effector
ibd IAUV Vision Processor
«component» Visual Odometry
«component» Visual Docking Controller
«component» IAUV Camera
«component» Manipulation Identifier

Figure 22. Functional view of the JAUS-compliant system structure

4.2.2 Low-level Functionalities and Data Coupling

Functionalities are the atomic elements of a functional structure of the system. The key point to identify basic services in components (Figure 22) is to group or ungroup low-level component functionalities. It is done by following the next criterion:

- to group functionalities:
  - Component functionalities are related
  - Component functionalities require similar information
- to separate functionalities:
  - Component functionalities are not related
  - Component functionalities exist on different hardware platform
  - Different numbers of component functionalities are required at runtime.

A good practice to identify component functionalities that then are wrapped in services is to make a list with all the components available in the platform, and their functions. In addition, the inputs and outputs of the function are provided in order to specify the data coupling among components. The functionalities of the AMR system (or platform) components are listed in Appendix B.

The functionalities (high level functionalities defined in Section 4.1, and low level functionalities above specified) of the platform components allow carrying out activities in the system. The activities can be classified at different levels of interaction. Thus, activities are classified as follows:

- Missions are divided into Operations.
- Operations are divided into Tasks.
- Tasks are divided into Actions.
- Actions are the atomic part of the hierarchy.

4.3 Detailed System Design

According to the JAUS reference architecture specification, services are provided by AMR system components. Services encapsulate functionalities of AMR system components, and can be seen as a block with one or more functions.

4.3.1 Description of Services

The intelligent control architecture supports two types of services:

- Basic services: They are indivisible. Therefore, they are the atomic elements of the SOA in which the intelligent control architecture is based on.
- Composite services: they are composed by in other services (basic or composite services). This composition of service is called orchestration of services in SOA.

Figure 23 shows the different parts of the anatomy of the basic and composite services. The former (a) is built of primitive functions linked to actions (one by one) that have pre-conditions and post-conditions. The latter (b) is built of other services that can be basic or composite services.

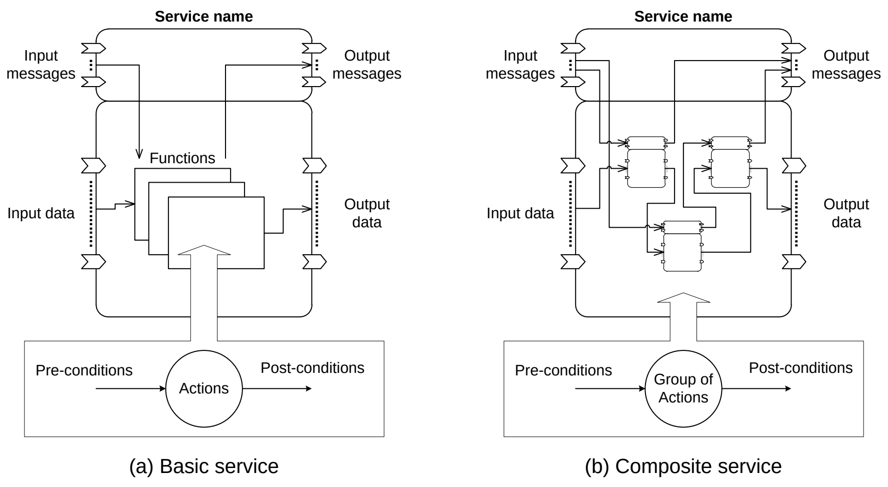


Figure 23. Types of services. (a) Basic service. (b) Composite service.

Since services encapsulate functionalities, they are also classified following the functionality categorization given in Section 4.2. Therefore, the service classification is as follows:

- A mission service is composed of Operation services
- An operation service is composed of Task services
- A task service is composed of Action services
- An action service are the atomic part of the service hierarchy

The first three service classifications are composite services, and the last one is a basic service. The mission and operation services are designed by wrapping high-level functionalities as defined in Section 4.1. The task and action services are designed by wrapping low-level functionalities that provide the platform components shown in Figure 22.

A definition and description of all the services from the ICA is in the Appendix C.

4.3.2 Protocols of Services

The protocols of services are designed based on the use cases and scenarios defined in Section 5.4. The events and data exchange among services are identified from the following interaction diagrams.

Figure 24 shows the interaction among AMR system actors (i.e. as defined in Section) OCU, ASC, and IAUV) for mission and operation services.

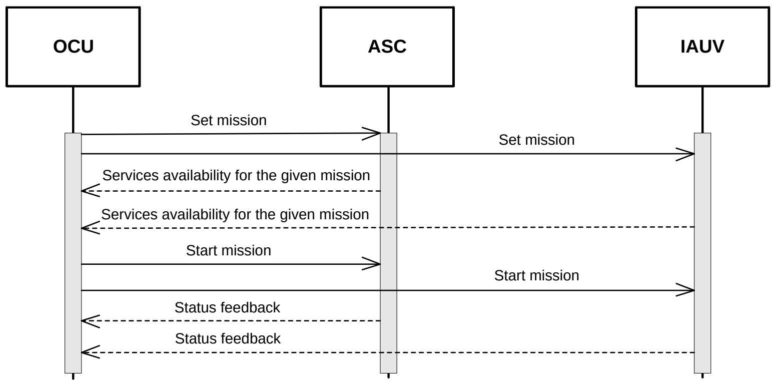


Figure 24. External interaction among OCU, ASC, and IAUV

Figure 25 shows the interaction among the OCU components for task and action services.

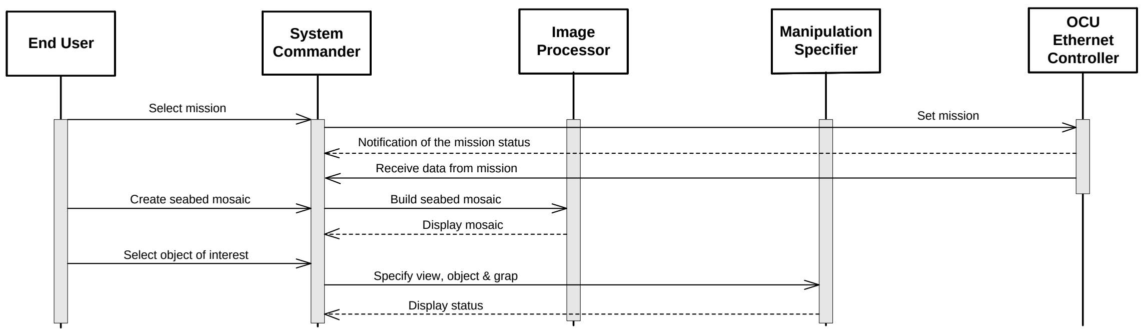


Figure 25. Internal interaction for the OCU

Figure 26 shows the interaction among the ASC components for task and action services.

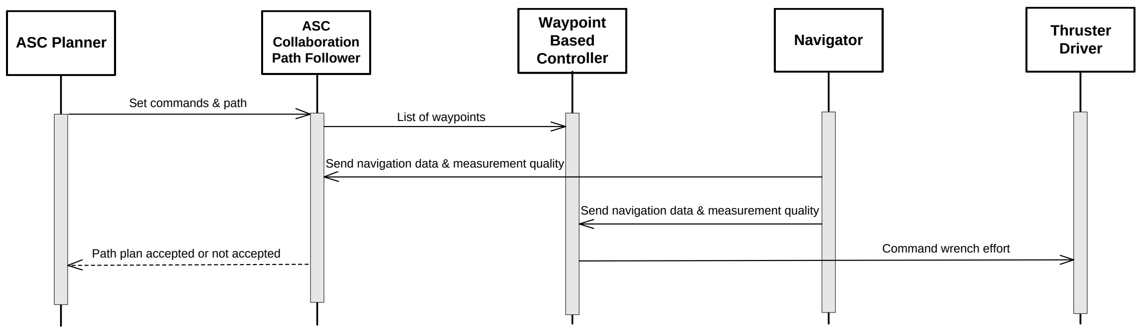

ASC Planner
ASC Collaboration Path Follower
Waypoint Based Controller
Navigator
Thruster Driver
Set commands & path
List of waypoints
Send navigation data & measurement quality
Send navigation data & measurement quality
Command wrench effort
Path plan accepted or not accepted

Figure 26. Internal interaction for the ASC

Figure 27 shows the interaction among the IAUV components for task and action services.

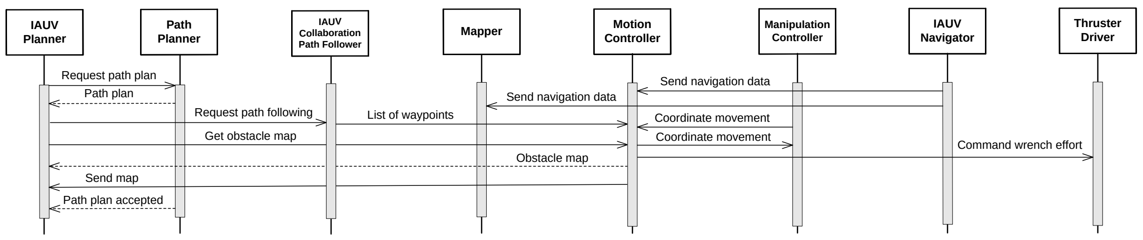


Figure 27. Internal interaction for the IAUV

Table 4 presents the characteristics of the interaction among services. From left to right, providers are AMR system components that provide services. Consumers are the components that demand the services from the component providers.

The communication model is a representation of the data-linking mechanisms to transmit and receive information between communicating parts. This model follows three basic way to get communicated; one to one (Peer-to-Peer), one to many (Publisher/Subscriber), many to one (Client/Server).

The interaction model is a representation of the data-exchanging mechanisms to transmit and receive information between communicating parts. This model involves four way of services interaction; request-response, only request, solicitation-response, only notification. The blocking mechanism takes into account two communication ways; synchronous (blocking), asynchronous (non-blocking).

Table 4. Interaction among services

Connection between services		Communication model	Interaction Model	Blocking mechanism
Provider	**Consumer**			
Image Processor (Seabed Image Mosaicing)	OCU System Commander	Peer-to-peer	Request-Response	Synchronous
Manipulation Specifier (View Characterization)		Peer-to-peer	Request-Response	Synchronous
Manipulation Specifier (Object Characterization)		Peer-to-peer	Request-Response	Synchronous
Manipulation Specifier (Grasp Specification)		Peer-to-peer	Request-Response	Synchronous
ASC Collaboration Path Follower (Behaviour Management)	ASC Planner	Peer-to-peer	Request-Response	Synchronous
Waypoint-Based Controller (Waypoint List Setting)		Peer-to-peer	Request-Response	Synchronous
IAUV Data Storage (IAUV Operation Area Setting)		Peer-to-peer	Request-Response	Synchronous
ASC Navigator (ASC Navigation Data Sending)	Waypoint-Based Controller	Publisher/Subscriber	Notification	Asynchronous
Path Planner (Path Plan Setting)	IAUV Planner	Peer-to-peer	Request-Response	Synchronous
Visual Odometry (Seabed Data Collection)		Peer-to-peer	Request-Response	Synchronous
Visual Docking Controller (Vehicle Docking)		Peer-to-peer	Request-Response	Synchronous
ASC Data Storage (ASC Operation Area Setting)		Peer-to-peer	Request-Response	Synchronous
Manipulation Identifier (Scene Identification)		Peer-to-peer	Request-Response	Synchronous
Manipulation Identifier (Object Identification)		Peer-to-peer	Request-Response	Synchronous
Manipulation Controller (Intervention Configuration Setting)		Peer-to-peer	Request-Response	Synchronous
Manipulation Controller (Object Intervention Manoeuvre)		Peer-to-peer	Request-Response	Synchronous
Mapper (Obstacle Map Generation)	Path Planner	Peer-to-peer	Request-Response	Synchronous
Mapper (Map Sending)		Publisher/Subscriber	Notification	Asynchronous
IAUV Navigator (IAUV Navigation Data Sending)	Waypoint-Based Controller	Publisher/Subscriber	Notification	Asynchronous
	Motion Controller	Publisher/Subscriber	Notification	Asynchronous
	Mapper	Publisher/Subscriber	Notification	Asynchronous
	Manipulation Controller	Publisher/Subscriber	Notification	Asynchronous
	IAUV Planner	Publisher/Subscriber	Notification	Asynchronous
Visual Odometry (Visual Navigation Data)	Manipulation Controller	Publisher/Subscriber	Notification	Asynchronous
Manipulation Controller (Body Force Control)	Motion Controller	Publisher/Subscriber	Notification	Asynchronous

Communication model = {peer to peer, publisher/subscriber, client/server}

Interaction mode = {request-response, request, solicitation-response, notification}

Blocking mechanism = {Synchronous (blocking), Asynchronous (non-blocking)}

4.3.3 Service-oriented architecture interoperability

- The component services have a higher degree of autonomy than conventional ones since Component Profile for Ocean System Services (CPOSS) is supported by the SOA approach proposed. CPOSS defines a minimal set of implementation constraints to enable secure interoperation among services on resource-constrained components.
- Form the control engineering viewpoint, AMR components are categorized as either controlling components or controlled components. However, a given component may play both roles. The interoperation patterns of a component-level SOA (or CPOSS) can be categorized according to the following basic interoperation mechanism for services (set of networking protocols, i.e. Universal Plug and Play). Protocols adapted from [37].
- **Addressing**. This is the foundation for component networking. The way to address services from components is through a Uniform Resource Identifier (URI). The addressing is provided by the IP protocol.
- **Discovery.** Once addressing is established, components need to discover each other. When a controlled component is added to the network, a discovery protocol enables it to advertise its services on the network. When a controlling component enters the network it sends out a search request, and then the components that match the request send a corresponding reply.
- **Description**. Once a controlling component has discovered a controlled component, to learn more about the latter and its capabilities, the controlling component must retrieve the controlled component description. For each service provided by a component, the component description defines the command messages that the service responds to, as well as the associated message formats.
- **Control**. A controlling component can exert control over a controlled component. A controlling component sends a control message to the network endpoint for that service to invoke a component service. The service may or may not return a response message providing any command specific information.
- **Eventing**. Components may communicate through asynchronous eventing. It is usually implemented by a "publisher-subscriber" mechanism through which a service exposes events corresponding to internal state changes. Controlling components can subscribe in order to receive event notifications whenever the corresponding internal state change occurs.

Chapter 5: Architecture Realization

5.1 System Architecture Implementation

This Subsection presents details of the implementation of the ICA by means of the Robotic Operating System (ROS) [30].

5.1.1 Robotic Operating System (ROS) Middleware

Following research into available robotics middleware solutions and discussion, it was decided to use the open source ROS middleware as the basis for software interfaces between the AMR modules. Here are some of the motivations for choosing ROS:

- Rich open source framework for robotics development.
- Allows public (network) interfaces without exposing source code: TRIDENT work can be integrated with anyone else using ROS.
- Broad and growing user base - in November 2010, there were 50 public repositories contributing open-source libraries and tools, and over 50 robots using ROS [29].
- BSD licensed, so free to use, modify, and commercialise upon.
- Range of existing open source algorithms available from groups in academia and industry, including code for robotic manipulation tasks.

At the core of ROS is a well-engineered communications middleware based on a simple C-like Interface Definition Language (IDL). This language may be used to define messages, services, and asynchronous temporal actions. The ROS IDL supports a standard range of primitives, fixed and variable length arrays, and nesting of messages. The ROS build system automatically generates C++, Python, Java, and Matlab marshalling code from IDL definitions.

Figure 28 shows the various layers of the ROS networking stack. Messages are communicated using a UDP or TCP point to point publish-subscribe mechanism, with a ROS 'master' node maintaining information about active publishers and subscribers. It thus serves as a low level communication broker. ROS services are effectively request-response remote procedure calls, transparently built using ROS messages. It is important to note that ROS services are not services in the sense of a Service Oriented Architecture. ROS actions are temporal constructs, defined by a request, optional periodic feedback, and result; actions are also transparently implemented using ROS messages. A ROS action in progress may be cancelled at any time by either the action server or client. Again, please note that ROS actions are not necessarily equivalent to actions defined in a Service Oriented Architecture, or in the context of planning.

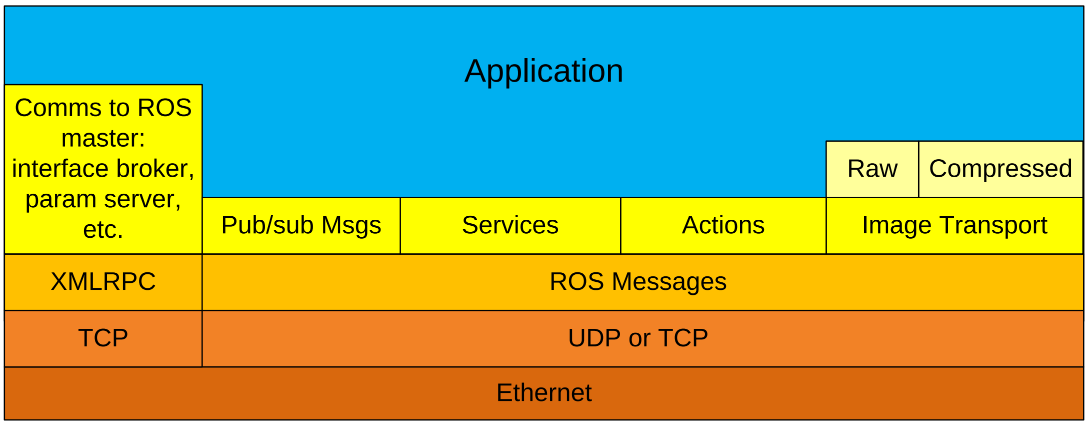


Figure 28: ROS Networking Stack

Graph Resource Names are used by ROS to provide powerful hierarchical naming of resources such as nodes, parameters, message topics, and services (http://www.ros.org/wiki/Names). Example names are /nav/nav_sts, /nav/odometry, /camera1/image. This naming system provides powerful encapsulation of robot functionality, enabling components of a robot to be 'pushed down' into different namespaces, so as not to conflict with each other. For example, separate instances of vision software on a robot could be pushed into the namespaces /camera_left and /camera_right.

On top of the flexible communications middleware, ROS includes powerful tools for package management and building, text output logging, message logging and replay, 3D visualisation, and 2D and 3D simulation. The software is fully open source (BSD licensed), and free for others to use, modify and commercialise upon.

5.1.2 Service Matchmaker

The service interface discovery and pairing process for TRIDENT is illustrated with a sequence diagram in Figure 29. The motivation is to allow the mission planner to dynamically pair service interfaces to achieve the desired mission functionality; this avoids a hard-wired set of modules, facilitating ‘plug-and-play’ integration of modules. Each interface producer/consumer shall be advertised by the software module that hosts it, using a message sent to the Service Oriented Architecture Matchmaker (SOA Matchmaker). Either periodically, or before each re-plan, the mission planner will query the SOA Matchmaker for available services. The query response will reference the SOA Service concepts in the ontology. When the planner has selected the best configuration of services for a particular operation, it will send commands to the SOA Matchmaker to pair these service interfaces. As well as pairing services using dynamically generated ROS graph resource names, the Matchmaker will support cases where the provider or the consumer name is fixed. This allows use cases such as the ‘broadcast’ of vehicle pose by the navigator. Once a service pair is no longer needed, it will be unpaired by a call from the mission planner to the Matchmaker.

A library and reference code is created to allow for easy implementation of service advertisement and configuration within a software module, which will support a simple fixed configuration mode for testing modules without the matchmaker or mission planner.

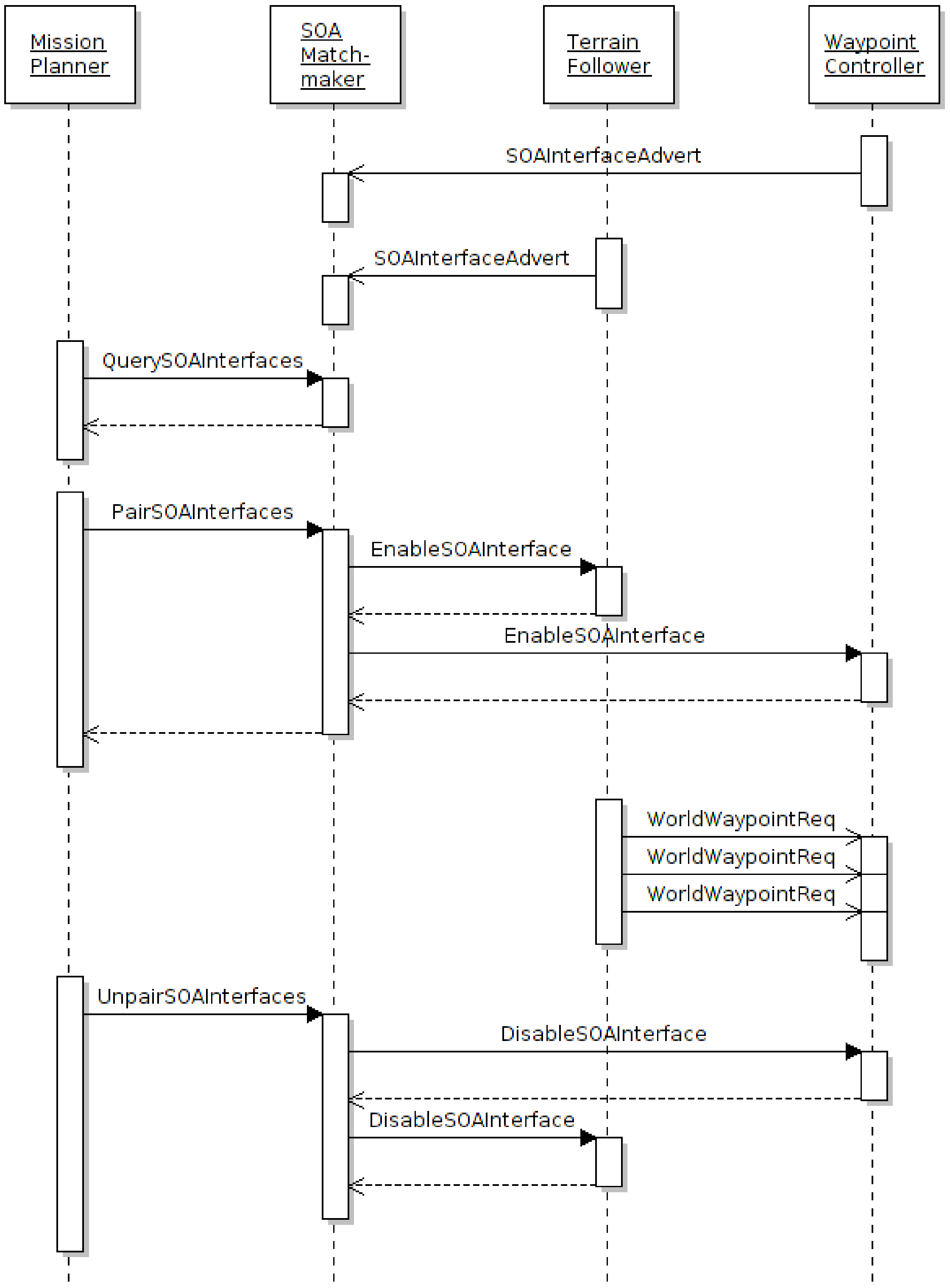


Figure 29. Illustration of service use lifecycle

Table 5 summarises the ROS (M)essages, (S)ervices, and (A)ctions that will be used to implement the service oriented architecture interfaces. '*' after the type letter indicates use of a standard ROS message. Detailed versions of the ROS interfaces are given in the Appendix D. The reference numbers are used to annotate Figure 30 and Figure 31 below.

Table 5: ROS Interfaces

Ref.	Type	Name	Notes
1	M*	geometry_msgs/ TransformStamped	Vehicle pose (forward, left, up frame), tf frame odom→base_link
2	M*	geometry_msgs/ TransformStamped	Manipulator pose, tf frames base_link→{arm…}
3	M*	nav_msgs/Odometry	Vehicle pose and pose velocities, tf frame odom→base_link
4	M	NavSts	Vehicle pose (north, east, down, altitude) equiv. to frame /map→base_link
5	M	WorldWaypointReq	Vehicle pose request (north, east, down, altitude), equiv. to frame /map→base_link
6	M	WaypointSts	Status of current waypoint request
7	M	BodyVelocityReq	Vehicle velocity request in body frame (to act on base_link)
8	M	BodyForceReq	Vehicle thrust force request in body frame (to act on base_link)
9	S	PlanVehicleSearchPath	Produce a path for searching the sea bed
10	S	SetInterventionConfig	Prepares an intervention as specified
11	A	PerformInterventionStrategy	Performs one component of an intervention
12	A	IdentifyView	Attempt to localise the vehicle within the given view based on currently visible environment
13	A	IdentifyObject	Attempt to localise an object in the currently visible environment
14	A	IdentifyDock	Attempt to localise the dock within the currently visible environment
15	M	VisualMotion	Estimate of current vehicle motion, from vision
16	M	sensor_msgs/Image	Visual image
17	M	sensor_msgs/CameraInfo	Camera information associated with image
18	A	FollowTerrainWithPath	Perform terrain following over a path.
19	A	FollowLeaderWithPath	Perform lead vehicle following with a-priori path.
20	A	EnterDock	Instructs the IAUV to enter the dock of the ASC.
21	A	LeaveDock	Instructs the IAUV to leave the dock of the ASC.

Figure 30 and Figure 31 represent the functional relation among ROS services from the different AMR modules. A description of the ROS interfaces is in Table 5.

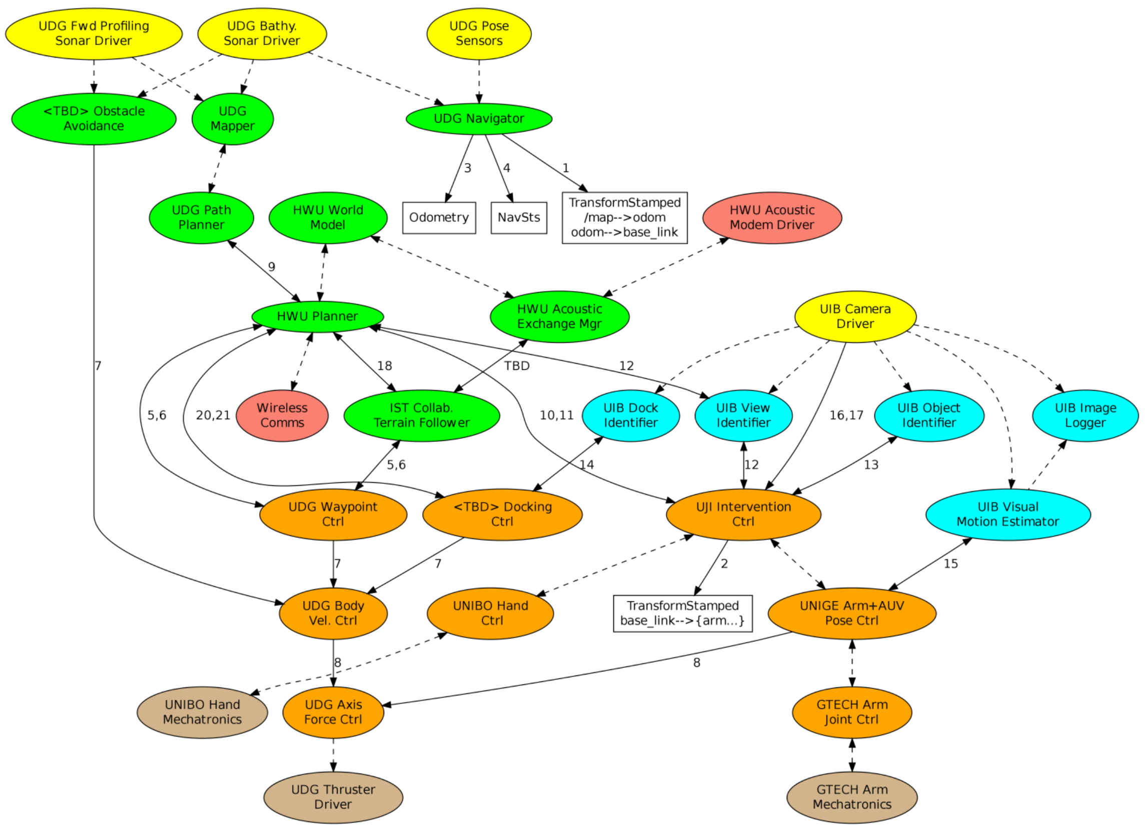


Figure 30: Functional view of IAUV with ROS interfaces

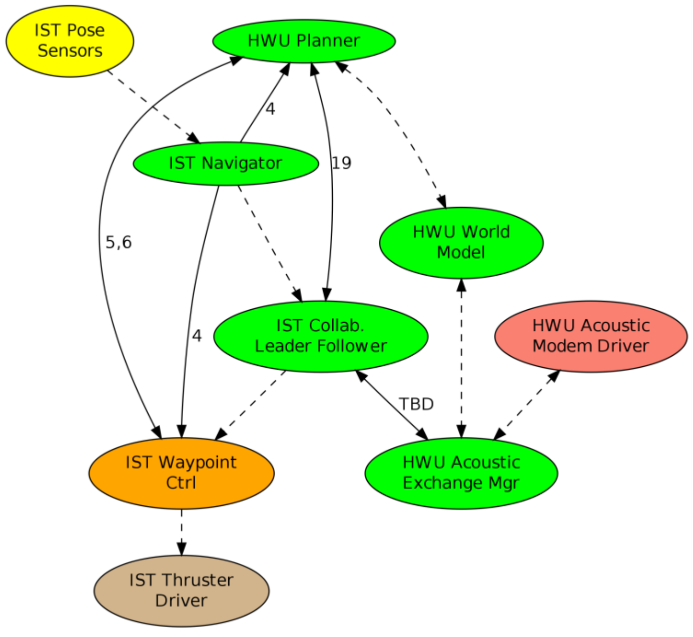


Figure 31: Functional view of ASC with ROS services

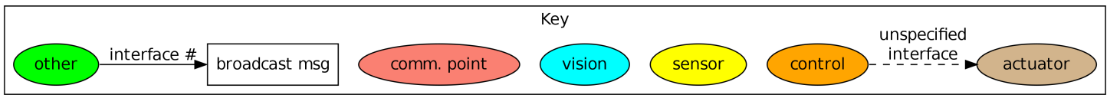


Table 6 below shows the mapping of the service interactions defined in Table 4 to the ROS services given in Table 5 above. Some ROS interfaces, particularly those related to the OCU, are still to be determined.

Table 6: SOA to ROS interface mapping

Connection between services		**ROS Type**	**ROS Interface Name**	**ROS Ref.**
Provider	**Consumer**			
Image Processor (Seabed Image Mosaicing)	OCU System Commander		TBD	
Manipulation Specifier (View Characterization)			TBD	
Manipulation Specifier (Object Characterization)			TBD	
Manipulation Specifier (Grasp Specification)			TBD	
Collaboration Path Follower (Behaviour Management)	ASC Planner	Action	FollowLeaderWithPath	19
Waypoint-Based Controller (Waypoint Setting)		Message	WorldWaypointReq reply WaypointSts	5 6
IAUV Data Storage (IAUV Operation Area Setting)			TBD	
Motion Controller (Manipulation Configuration)		Service	SetInterventionConfig	10
ASC Planner	Waypoint-Based Controller	Message	WaypointSts	6
ASC Navigator (ASC Navigation Data Sending)	Waypoint-Based Controller	Message	NavSts	4
Path Planner (Path Plan Setting)	IAUV Planner	Service	PlanVehicleSearchPath	9
Collaboration Terrain Follower		Service	FollowLeaderWithPath	18
Visual Odometry (Seabed Data Collection)			TBD	
Visual Docking Controller (Vehicle Docking)		Action	EnterDock, LeaveDock	20, 21
ASC Data Storage (ASC Operation Area Setting)			TBD	
Manipulation Identifier (Scene Identification)		Action	IdentifyView	12
Manipulation Controller (Intervention Configuration Setting)		Service	SetInterventionConfig	10
Manipulation Controller (Object Intervention Manoeuvre)		Action	PerformInterventionStrategy	11
Mapper (Obstacle Map Generation)	Path Planner		<Partner internal interface>	
Mapper (Map Sending)			<Partner internal interface>	
IAUV Navigator (IAUV Navigation Data Sending)	Waypoint-Based Controller	Message	TransformStamped* (odom→base_link), Odometry* (odom→base_link), NavSts	1,3,4
	Motion Controller			
	Mapper			
	Manipulation Controller			
	IAUV Planner			
Visual Odometry (Navigation Data Update)	Manipulation Controller	Message	VisualMotion	15
Manipulation Controller (Body Force Control)	Motion Controller	Message	BodyForceReq	8

5.2 System Architecture Integration

This Subsection presents details on how the different AMR system modules are integrated based on the above ROS services.

5.2.1 Physical System Integration

The strategy for the integration plan is based on the categorization of capabilities presented in Section 4.2. Such a plan is structured according to the physical locations of the above capabilities in order to simply the integration process but also satisfy the project milestones.

The operational capabilities are achieved by integrating the functional capabilities (assembling of system module). The integration strategy defines evaluation cases according the scenarios above defined in order to verify and validate the operational and functional capabilities supported by the AMR system.

The integration strategy is tied to the following constraints that the system modules, system nodes or subsystems can have:

- Operational constraints. Contention problem due to concurrent effecting command on the same system component.
- Functional constraints. Functionality of system modules shared by one or more applications.
- Physical constraints. System modules that must be co-located.

The integration rules based on constraints and requirements are given below.

- Physical
    - Components that must be physically together.
    - Components that must be physically apart.
- Functional Control and Data Functions in which the components are involved. The functions can be part of an application or the entire application.
    - Components that are functionally linked.
    - Components that are not functionally linked.

- Operational
    - Components that are utilized by different applications at a time (Contention or concurrency)
    - Components that are not utilized by different applications at a time.

5.2.2 Virtual System Integration

The integration framework involves simulation based on a tool called 'UWSim' [38] that supports the emulation of the AMR system and its operational environment. UWSim [38] involves emulation of the vehicles dynamics and kinematics, and the dynamics of the environmental physics. This simulator allows users to virtually model physical AMR system, and run software application as if they were deployed on the real AMR system.

The integration strategy is based on the three basic system capabilities: Navigation, Guidance, Control (NGC); Manipulation and Multi-propose Intervention (MMI), and Vision and Image Processing (VIP). The integration plan is structured by following the physical locations of the above capabilities in order to simply the integration process.

The purpose is to integrate operational capabilities by assembling functional capabilities. The integration strategy follows the requirements-test cases chain so that the user requirements can be validated by the capabilities supported by the AMR system.

The system capabilities are classified as shown in Table 7.

Table 7. Capability classification

Capability	NGC	VIP	MMI
Seabed survey	X	X	
Target selection	X	X	
Object manipulation	X	X	X
Vehicle homing	X		
Vehicle docking	X		
Path/Terrain/Leader following	X		
(Dynamic) Vehicle positioning	X		
Station-keeping manipulation	X		
Object search	X	X	
Scene/Object identification	X	X	
View/Object characterization		X	
Grasp specification		X	
Seabed data collection	X	X	
Motion estimation	X	X	
Free-floating manipulation	X	X	X
Intervention configuration / manoeuvre			X
Navigation data sending	X		
Motion control	X		
Path plan setting	X		
Obstacle map generation		X	
Seabed image mosaicking			X
Vehicle motion driving	X		
Effector driving			X

5.3 Operation Context

The main mission proposed for the AMR system is a multipurpose generic intervention. It is divided into two phases: seabed survey and target intervention. Figure 32 shows the above two mission phases.

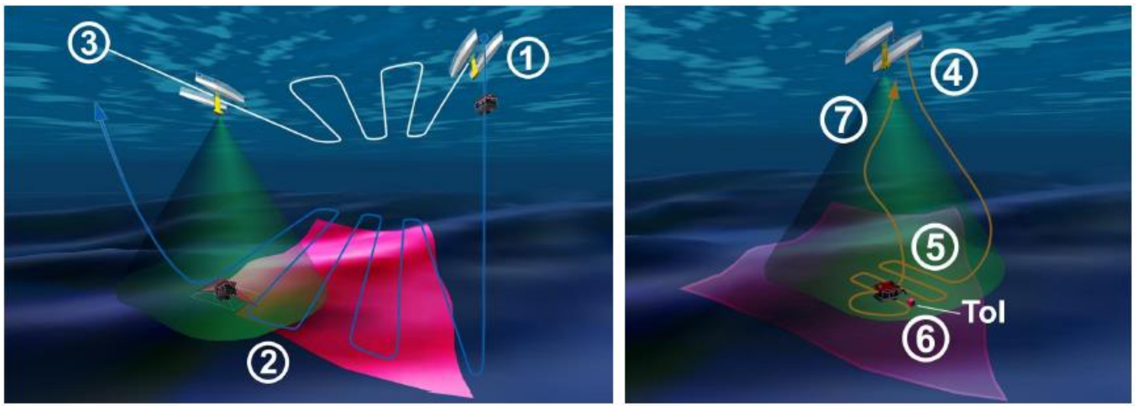


Figure 32. Two online mission stages: seabed survey (left), and target intervention (right).

In the first phase, the IAUV deployed from the ASC (1) executes a pre-plotted survey (2) gathering visual and acoustic data from the seafloor (terrain tracking while ASC path following), whilst the ASC provides geo-referenced navigation data and communication with the end user. The motion of the ASC is coordinated with that of the IAUV in order to achieve precise positioning and reliable acoustic communications. After the seabed data collection phase, the IAUV docks with the ASC (3) and sends the data back to a ground station. With this information, a map is created and a target object is identified by the operator.

In the second phase, the ASC navigates towards a waypoint near the intervention area (4), where the IAUV is launched to search for the object selected (5). When the object (i.e. the Target of the Intervention; ToI) is found, the IAUV switches to free floating navigation mode, including station keeping. The manipulation of the object takes place using a dexterous hand attached to a redundant robot arm mounted on the IAUV (6). After the object manipulation operation, the IAUV meets (homing and docking operations) the ASC on the surface (7), and is free for a new mission.

5.4 Case Study

Figure 33 shows three ICA use cases: seeded survey, target selection, and object manipulation. They are the ARM system sub-missions defined above. The evaluation scenario comprises the Operation Control Unit (OCU), the Autonomous Surface Craft (ASC), and the Intervention Autonomous Underwater Vehicle (IAUV).

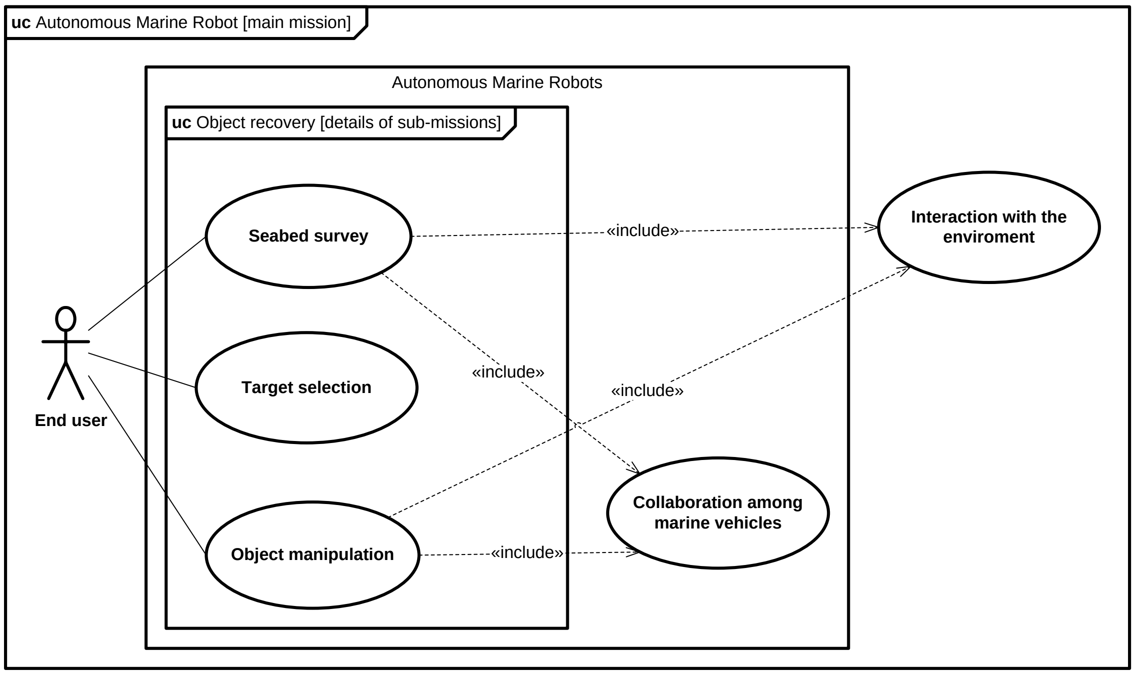

uc Autonomous Marine Robot [main mission]
Autonomous Marine Robots
uc Object recovery [details of sub-missions]
Seabed survey
Target selection
Object manipulation
End user
«include»
«include»
«include»
«include»
Interaction with the enviroment
Collaboration among marine vehicles

Figure 33. AMR system use case.

The AMR system actors are OCU (end user), ASC, and IAUV. This Subsection only shows results from the computer simulation performed for the two online sub-mission processes: seabed survey, and target intervention.

5.4.1 Seabed Survey

Two scenarios are defined for the seabed survey. Scenario A is that where the seabed survey is carried out under a fault-free context.

Scenario A

Figure 34 shows the scenario and interaction among AMR system actors for the seabed survey sub-mission when no faults are present during the sub-mission execution.

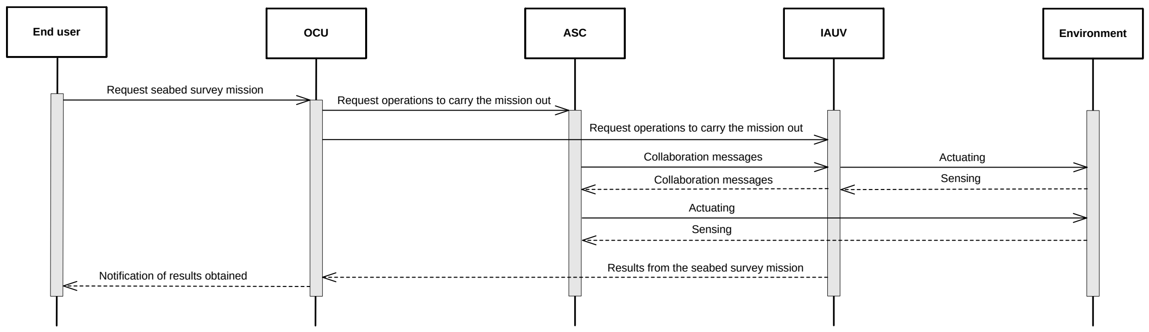

End user
OCU
ASC
IAUV
Environment
Request seabed survey mission
Request operations to carry the mission out
Request operations to carry the mission out
Collaboration messages
Actuating
Collaboration messages
Sensing
Actuating
Sensing
Results from the seabed survey mission
Notification of results obtained

Figure 34. Scenario for the seabed survey sub-mission (fault-free context).

The steps of the seabed survey sub-mission are:

i. The AMR system is switched on. All the system capabilities are published as services available to perform marine activities. Each marine vehicle advertises its own capabilities (mission and operation services) to the overall system based on its component (or module) capabilities (task and action services). For instance, the IAUV is able to perform terrain following (operation service) based on its task and action services, i.e. low-level functionalities such as navigation, path setting, motion control, and visual and acoustic sensors.

ii. The operator (through the OCU) selects the mission to be carried out: seabed survey. The OCU checks the ASC and IAUV capabilities available to the system. These capabilities are the services required to carry out the seabed survey mission. Following the service classification presented in Chapter 3, the seabed survey mission service is composed of these operation services:

a. ASC Positioning and IAUV Positioning (both executed in parallel)

b. ASC Path Following and IAUV Terrain Following (both executed in parallel)

c. IAUV Homing

iii. If all of the services needed to carry out the above mission are available, the system is ready to start the mission. In the case that one or more services fail, the system automatically tries to fill this gap by looking for similar capabilities. In any case the system notifies the operator (through the OCU) about its operational status.

iv. The seabed survey mission is divided into four steps.

a. First step is to position the marine vehicles according to the exploration area given by the operator. Two services are invoked in parallel, i.e. ASC Positioning and IAUV Positioning.

b. Second step is to collect data from the seabed. The IAUV performs a terrain following operation whist the ASC assists this activity by performing a leader following operation.

c. Third step is to dock the IAUV with the ASC.

d. Forth step is to retrieve the seabed data captured by the IAUV.

v. The system is ready to carry out the same mission again or another one (if the capabilities required are available). If any of the marine vehicles are switched off then its capabilities (as operation services) are no longer available to the system. Similarly, if any vehicle component is switched off, removed, or added to the vehicle, the operational capabilities of the vehicle are updated.

Scenario B

Figure 35 shows the scenario and interaction among the AMR system actors for the seabed survey sub-mission with a fault in one of the services. The service that introduces a fault is the leader following. The fault case is that such a service stops working at some point during the seabed survey. The above service is shut down to simulate it failing. If this service fails the IAUV is not able to follow the ASC. Therefore, there is no path for seafloor data collection and the seabed survey sub-mission cannot be carried out.

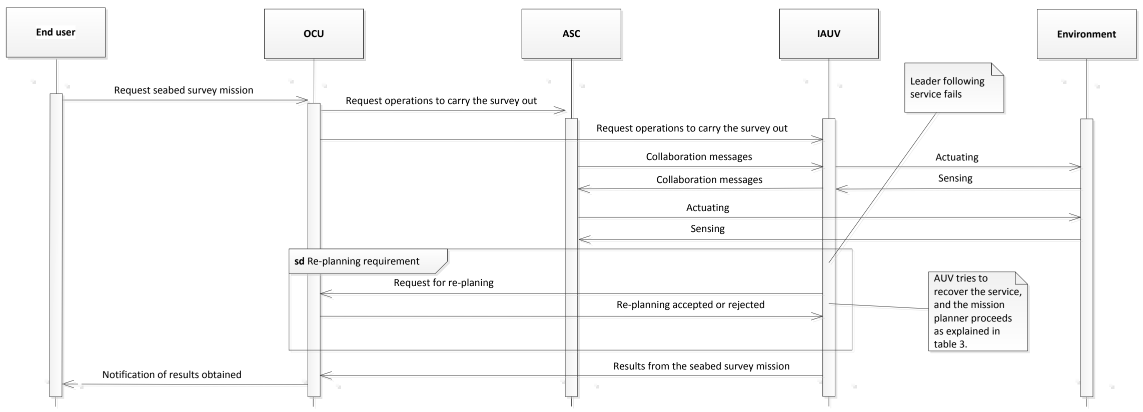

End user
OCU
ASC
IAUV
Environment
Leader following service fails
Request seabed survey mission
Request operations to carry the survey out
Request operations to carry the survey out
Collaboration messages
Actuating
Collaboration messages
Sensing
Actuating
Sensing
sd Re-planning requirement
Request for re-planing
AUV tries to recover the service, and the mission planner proceeds as explained in table 3.
Re-planning accepted or rejected
Results from the seabed survey mission
Notification of results obtained

Figure 35. Scenario for the seabed survey sub-mission (with a fault during service execution).

The seabed survey has three main states:

1. Both vehicles are at the origin position (initial position where the vehicles are deployed to begin the mission).
2. Both vehicles are at the beginning position (initial corner of the exploration area).
3. Both vehicles are at the end position (final corner of the exploration area).

Figure 35 presents the typical what-if scenario where the AMR system is expected to respond as follows in Table 8.

Table 8. Interaction levels of the ICA architectural elements.

Fault Case	**Fault Diagnosis**	**Fault Mitigation**
1	The leader following service does not work or stop working before the seabed survey starts.	If the human operator requests a seabed survey from the OCU, he/she is notified through the OCU display that the seabed survey sub-mission cannot be carried out because a capability (leader following service) is missing.
2	The leader following service stop working during positioning of the vehicles (from origin position to beginning position).	The AMR system keeps waiting for the service to be available. If it does not do it after a time set since the vehicles are in the beginning position, the mission planner makes the decision of aborting the seabed survey and brings the vehicle back to the origin position. If the service is available at some point between the fault and the beginning of the mission, the mission planner makes the decision of using it as usually.
3	The leader following service stop working during seafloor data collection (from beginning position to end position).	The AMR system keeps waiting for the service to be available again. If it does not do it after a time set since the fault occurred, the mission planner makes the decision to bring both vehicles back to the end position, and keeps waiting for a pre-defined time (reasonable time period to wait for). If the service becomes available in such a time period, the mission planner makes the decision of starting the seafloor data collection again from the beginning position as usually. If not, the mission planner makes the decision of aborting the seabed survey and brings the vehicle back to the origin position.
4	The leader following service stop working during IAUV homing or docking, or ASC positioning back to the origin (from end position to origin position).	There is not mitigation plan, just a display message to notify the human operator.

5.4.2 Object Manipulation

Figure 36 shows the scenario and interaction among AMR system actors for the object manipulation sub-mission.

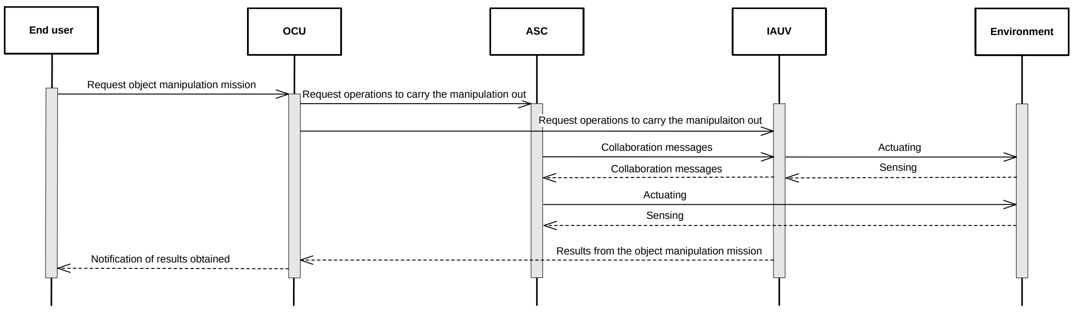

End user
OCU
ASC
IAUV
Environment
Request object manipulation mission
Request operations to carry the manipulation out
Request operations to carry the manipulaiton out
Collaboration messages
Actuating
Collaboration messages
Sensing
Actuating
Sensing
Results from the object manipulation mission
Notification of results obtained

Figure 36. Scenario for the object manipulation sub-mission.

The stages of the target intervention sub-mission are:

i. Ditto step 'i' for the seabed survey sub-mission.

ii. The operator (through the OCU) selects the mission to be carried out: target intervention. The OCU checks the ASC and IAUV capabilities available to the system, now including the manipulation capability form the IAUV. These capabilities are the services required to carry out the target intervention sub-mission. Following the service classification presented in Chapter 3, the target intervention mission service is composed of these operation services:

a. ASC Positioning and IAUV Positioning (both executed in parallel)

b. ASC Dynamic positioning and IAUV Terrain Following (both executed in parallel) until target reacquisition.

c. ASC Positioning and IAUV Homing and Docking.

iii. Ditto step 'iii' for the seabed survey sub-mission.

iv. The target intervention sub-mission is divided into four sub-steps.

a. First step is to position the marine vehicles according to the exploration area given by the operator. Two services are invoked in parallel, i.e. ASC Positioning and IAUV Positioning.

b. Second step is to visually scan the seabed until the object of interest is found. The IAUV performs a terrain following operation whist the ASC assists this activity by performing a dynamic positioning operation.

c. Third step, once the object is found, is to manoeuvre the IAUV to get close enough to the object.

d. Forth step is to perform the manipulation.

e. Once the manipulation is finished, the ASC returns to the origin position by performing a positioning operation, and the IAUV does the same by performing a homing and then docking operation.

v. Ditto step 'v' for the seabed survey sub-mission.

Chapter 6: Architecture Evaluation

6.1 Computer Simulation

The computer simulation entails three experiments based on two simulated seabed surveys, and a target intervention: a seabed survey with no faults introduced (scenario A), a seabed survey with a service fault introduced (scenario B), and a target intervention with no fault introduced (scenario C).

6.1.1 Simulation Setup

The UWSim Simulator [38] runs on Linux. The main setup requirement to run the simulation is to execute the roscore [39] (piece of software running required to run ROS-based software systems in order for ROS services to communicate). All the ROS services, including the roscore are launched from the command line from a Linux terminal. The setup sequence of the above ROS-based applications is as follows:

1. Run ROS core (‘~$**roscore**’)
2. Run Service Matchmaker (‘~$**rosrun** matchmaker matchmaker_server’)
3. Run UWSim (‘~$**rosrun** uwsim uwsim’)
4. Run each of the ROS-based services needed to simulate the different scenarios in no particular order.

ROS-based services (publishers) are able to advertise their capabilities to the AMR system so that other services (subscribers) can discover and make use of such capabilities. Once all the above applications are running, the matchmaker automatically makes ROS-based services able to discover and connect with each other according to their dependency (provided the roscore is running). The matchmaker functionality and details of its matchmaking capability as well as service interfaces and dependencies are described in Sub-subsection 5.1.2.

The seabed survey and target intervention experiments only focus on the guidance, navigation and control of the IAUV for the seabed survey mission. The two following scenarios for the seabed survey are simulated: Scenario A with no faults introduced, and Scenario B with a service fault introduced.

6.1.2 Scenario A: Seabed survey with no faults introduced

After setting up the simulation environment as indicated in Sub-subsection 6.1.1, the execution sequence of the ROS services required by the seabed survey simulation is as follows:

1. Run the path planner (‘~$**rosrun** stub_path_planner path_planner).
2. Run the leader follower (‘~$**rosrun** leaderfollowing LeaderFollowing).
3. Run the data collector (‘~$**rosrun** srv_trident_services data_collection).
4. Run the planner (‘~$**rosrun** planner Planner).

The following information is the input details for the seabed survey for the planner: **Mission:** seabed survey, **Type:** visual, **Area:** 40 m x 40 m, **Depth:** 5 m, **Timeout:** 300 sec. This is a XML file in practice that can be found in the Appendix E.

The execution sequence of the above ROS-based applications is as follows:

1. The simulation is started by selecting the mission to be carried out from the mission planner. Planner user interface including mission details and extra information about the cost of the mission is shown in Figure 37.

2. Once the mission is finished, it can be run again or other mission can be selected to be carried out.

3. The simulation environment is terminated by shutting down each of the ROS-based applications.

```
user@user-computer:~$ rosrun planner Planner.sh
Planner dir: /home/user/src/ros/trident-project/hwu_trident_planner/planner
[INFO]: --------------------------------------------- Mission Report [1] ---------------------------------------------------------
[INFO]: Mission Requirements:
[INFO]: Mission Name: seabedSurvey
[INFO]: Servoing Type: Visual
[INFO]: Exploration Area: [30, 30, 70, 70] coordinates in meters
[INFO]: Exploration Depth: 5 meters
[INFO]: Max Mission Time: 300 minutes
[INFO]: Mission Requirements:
[INFO]: Power required by Nessie: 168.3 watts
[INFO]: Time required by mission is lower than the time needed by the Nessie to complete the mission
[INFO]: Power required by Delfim: 58.8 watts
[INFO]: Time required by mission is lower than the time needed by the Delfim to complete the mission
[INFO]: ---------------------------------------------- Mission Report [2] -------------------------------------------------------
[INFO]: Mission Requirements:
[INFO]: Mission Name: targetIntervention
[INFO]: Servoing Type: Visual
[INFO]: Exploration Area: [30, 30, 50, 50] coordinates in meters
[INFO]: Exploration Depth: 5 meters
[INFO]: Max Mission Time: 100 minutes
[INFO]: Mission Requirements:
[INFO]: Power required by Nessie: 119 watts
[INFO]: Time required by mission is lower than the time needed by the Nessie to complete the mission
[INFO]: Power required by Delfim: 35.1 watts
[INFO]: Time required by mission is lower than the time needed by the Delfim to complete the mission
[INFO]: ---------------------------------------------------------------------------------------------------------------------------
[INFO]: Current Planner mission:
[INFO]: [1] seabedSurvey (Type: Visual; Distance Estimated: 580 metres; Time Estimated: 288 seconds)
[INFO]: [2] targetIntervention (Type: Visual; Distance Estimated: 70 metres; Time Estimated: 230 seconds)
[INFO]: select '0' to quit
```

Figure 37. Mission planner user interface.

The stages defined in Sub-subsection 5.4.1 (Scenario A) are at the group-of-vehicle and vehicle levels. Similar stages can be defined at the device and transducer level inside the vehicles. These stages are as follows:

- First Stage: the ASC and IAUV modules publish their capabilities as services available in their respective platforms. The advertisement and discovery of services are carried out by means of the service matchmaker. The matchmaker is designed to be used to match service providers to direct consumers of those services. It is the responsibility of each service consumer to use the matchmaker to resolve its own direct service dependencies. The IAUV module capabilities implemented for the simulation of the target intervention are: Path planning and Terrain following as operation services, and IAUV navigation, and IAUV motion control as task services.
- Second Stage: The mission planner checks the plan consistency (based on the capabilities required to carry out the mission) against the record kept by the matchmaker. The platform services are either available or not available. When they are available, the mission planner must check their health status in order to know if he can really make use of them. If there is any problem to execute the services or if they are unavailable, the service matchmaker proceeds to find any other capability that can replace the required one. If no capability are available at all, the service matchmaker must decide what to do (if it is still viable or not), and communicates to the rest of the system (AMR).
- Third Stage: once the availability of the required services is confirmed, the IAUV is ready to begin the mission. The mission spooler retrieves all the information needed to execute the services from the matchmaker, i.e. based on the service name, the mission spooler makes a query for status, and invocation method for each service in order to create the execution queue. The mission spooler then invokes the platform services according to the mission plan, and takes into account the health of the services. The IAUV, initially located at the origin position (surface), goes down to a given altitude where it reaches the beginning position to start the data collection from the seafloor. Then, it follows the path given so visual or acoustic data can be collected. When the IAUV reaches the end of the path (end position) for the exploration area, it returns to the surface (origin position) with the stored data. Figure 38 shows the path followed by the IAUV after being commanded to perform the activities for a seabed survey.
- Fourth Stage: once the seabed survey is finished, it can be carried out again. The IAUV will be endowed with more mission, operation, task, and action services in order to include more capabilities.

Figure 38 shows the behaviour results obtained in simulation of the seabed survey with the path followed by the two vehicles (ASC and IAUV). The ASC is the leader (it goes through the path to be followed; filled line) and the IAUV is the follower.

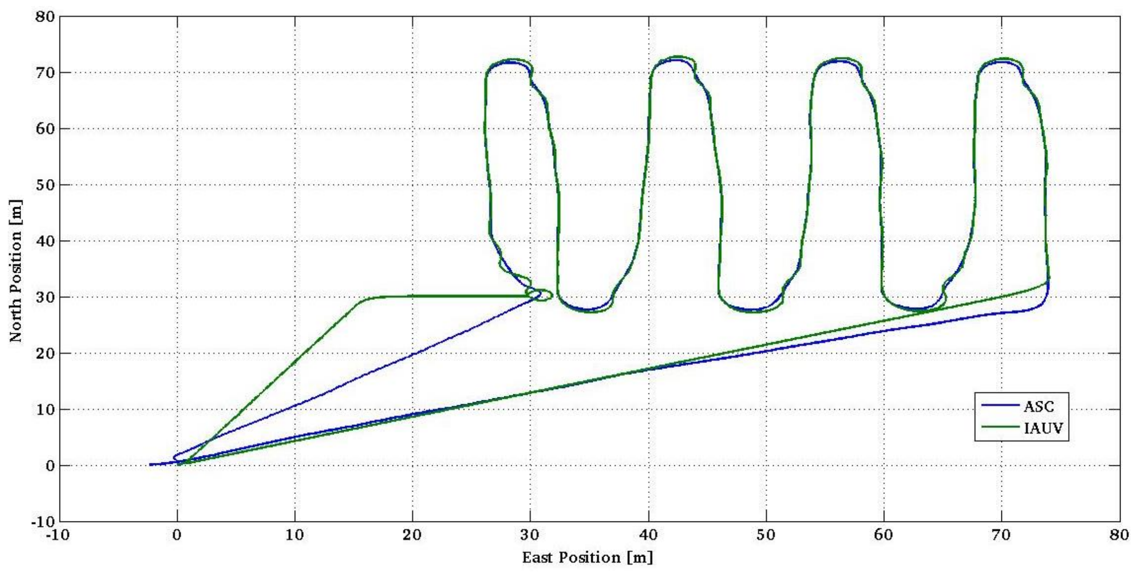


Figure 38. Result of the execution of services for seabed survey.

Figure 39 and Figure 40 show a comparison of the north and east positions of the ASC and IAUV for the above seabed survey. The difference between the ASC and IAUV positions is the positioning error of the IAUV with respect to the ASC. Such position difference between the ASC and IAUV concerns during the path-following/leader-following operation (survey area) where it is small; not relevant. It is due to the control error generated by the control algorithm.

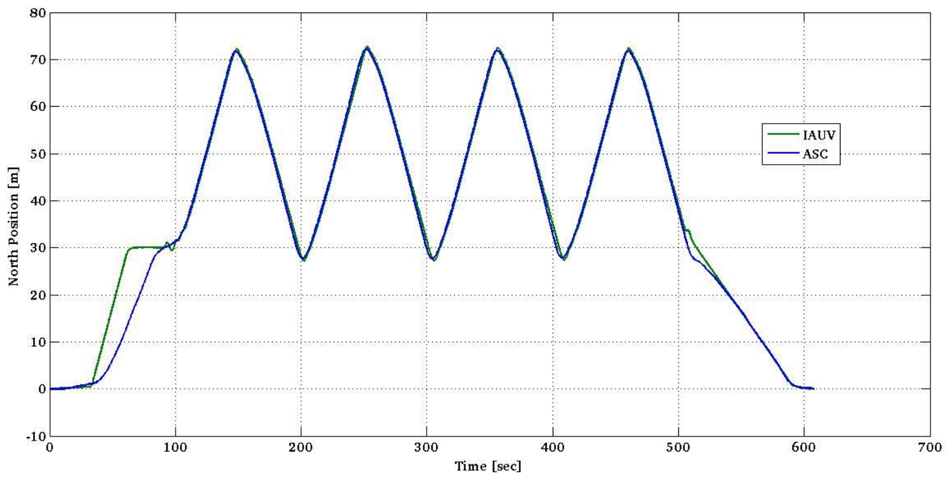


Figure 39. Comparison of north positions of the ASC and IAUV in the above seabed survey.

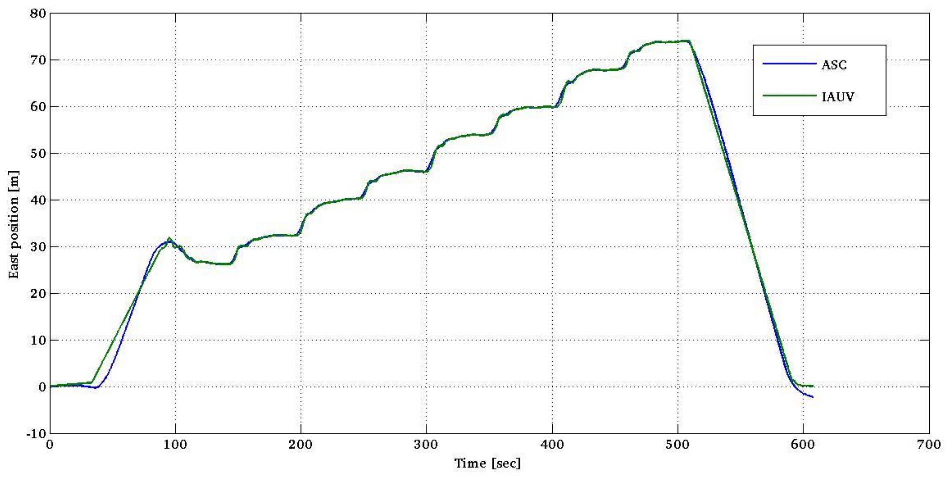


Figure 40. Comparison of east position of the ASC and IAUV in the above seabed survey.

6.1.3 Scenario B: Seabed survey with a service fault introduced

The stages defined in Sub-subsection 5.4.1 (Scenario B) are at the group-of-vehicle and vehicle levels. Similar stages can be defined at the device and transducer level inside the vehicles. These stages are as described above for scenario A but including the description for fault handling as described in Table 8. Scenario B shows the greatest potential of the research contribution since the ASC and IAUV are able to make in-mission decisions (without contacting or getting back to the human operator for advice on what to do). They, by themselves, are capable of dealing with abnormal situations based on the knowledge stored in the ontological database (the operator skills, platform capabilities and, possible changes in the environment).

Figure 41 shows the behaviour results obtained in simulation of the seabed survey with the path followed by the two vehicles (ASC and IAUV) when the IAUV leader following service stop working (68 m in the north position, 40 m in the east position). In this case, the above service does not recover itself nor is there a similar capability to replace THE faulty one. Thus, the vehicles act as described in fault case 3 of Table 8, i.e. firstly both vehicles come back to the beginning position (30 m in the north position, 30 m in the east position), then they come back to the origin position (0 m in the north position, 0 m in the east position).

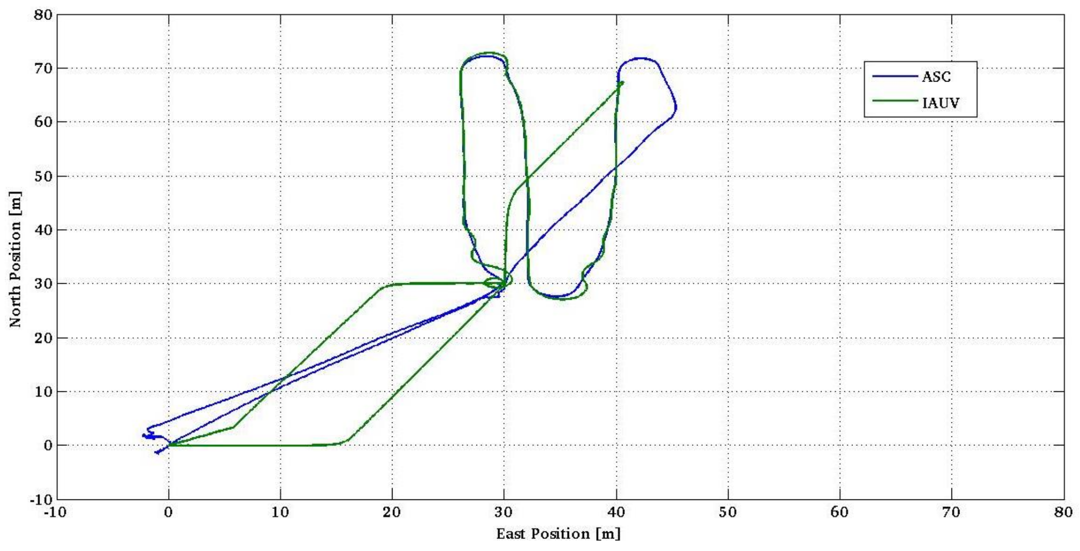


Figure 41. Result of the execution of services for seabed survey.

Figure 42 shows the behaviour results obtained in simulation of the seabed survey with the path followed by the two vehicles (ASC and IAUV) when the IAUV leader following service stop working (70 m in the north position, 40 m in the east position). In this case, the above service does recover itself or there is a similar capability to replace the faulty one. The former happens in this simulation. Thus, the vehicles act as described in fault case 3 of Table 8, i.e. firstly both vehicles come back to the beginning position (30 m in the north position, 30 m in the east position), then they start the seafloor data collection again at the beginning. Finally, they complete the mission given (30 m in the north position, 73 m in the east position), returning at the origin position.

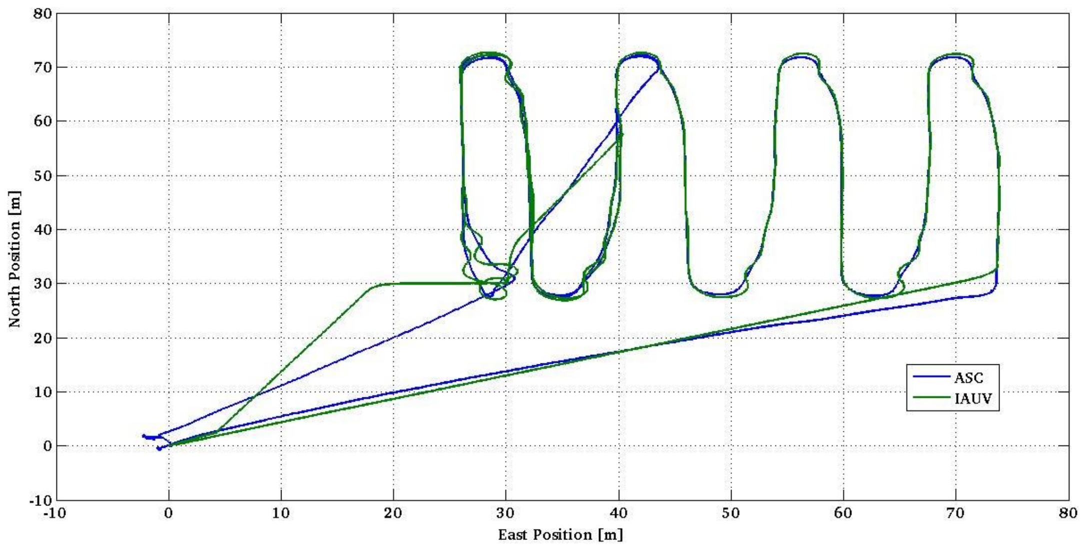


Figure 42. Result of the execution of services for seabed survey.

Figure 50 shows the user interface for the simulator UWSim when a seabed survey is being carried out.

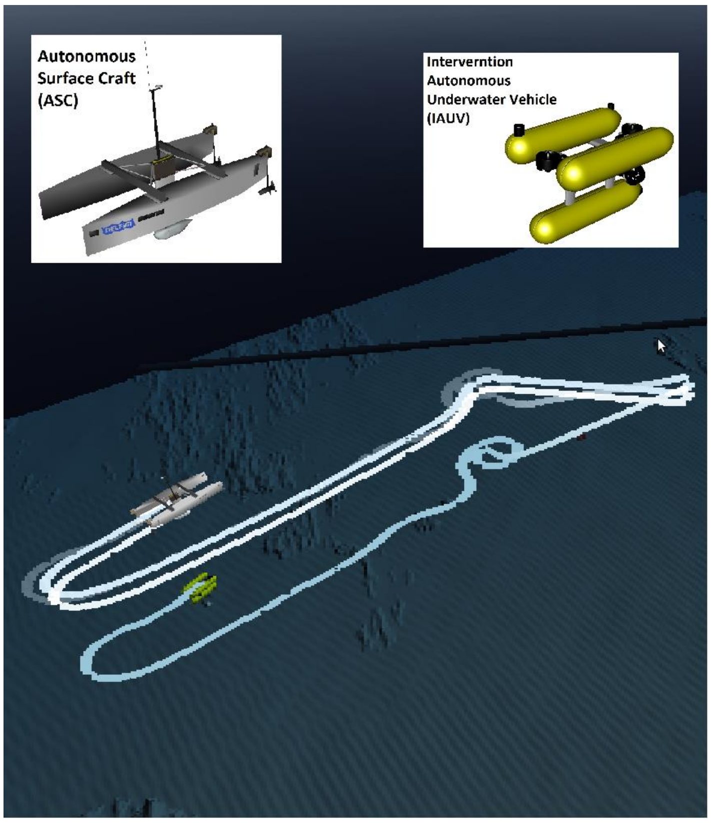


Figure 43. Result of the execution of services for seabed survey.

The execution steps for the target intervention are similar to the steps defined for the seabed survey. The difference is basically on capabilities required for each sub-mission. The target intervention experiment is simplified. It does not include a small seabed survey to find the object of interest (target). Instead, the ASC and IAUV go straight to an area where the target is, and from it, the IAUV starts looking for the target. Once the object is found, the IAUV starts the positioning in order to deal with the object (free-floating manipulation assisted by the ASC from the surface in terms of localization). Once the manipulation task is finished, both maritime vehicles come back to the origin position.

6.1.4 Scenario C: Target intervention with no fault introduced

Figure 44 shows the behaviour results obtained in simulation from the target intervention sub-mission. The path followed by the two vehicles (ASC and IAUV) is shown. The ASC is the leader (it goes through the path to be followed; filled line) and the IAUV is the follower.

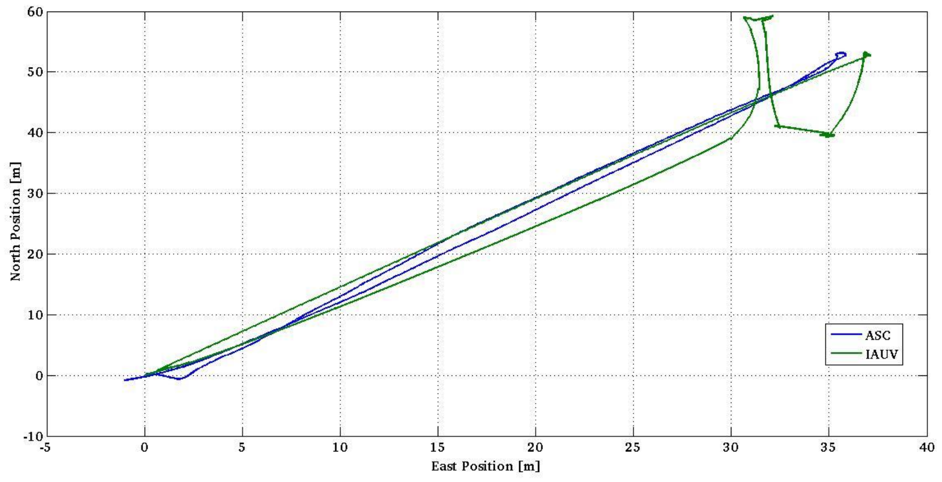


Figure 44. Result of the execution of services for target intervention.

Figure 45 and Figure 46 show a comparison of the north and east positions of the ASC and IAUV for the above target intervention sub-mission. The differences between the ASC and IAUV positions during the IAUV object manipulation in the subarea from (30 m, 40 m) to (40 m, 60 m) is acceptable. The ASC keeps the dynamic position ~ (37 m, 53 m) while the IAUV is supposed to operate under a coverage cone for underwater communication.

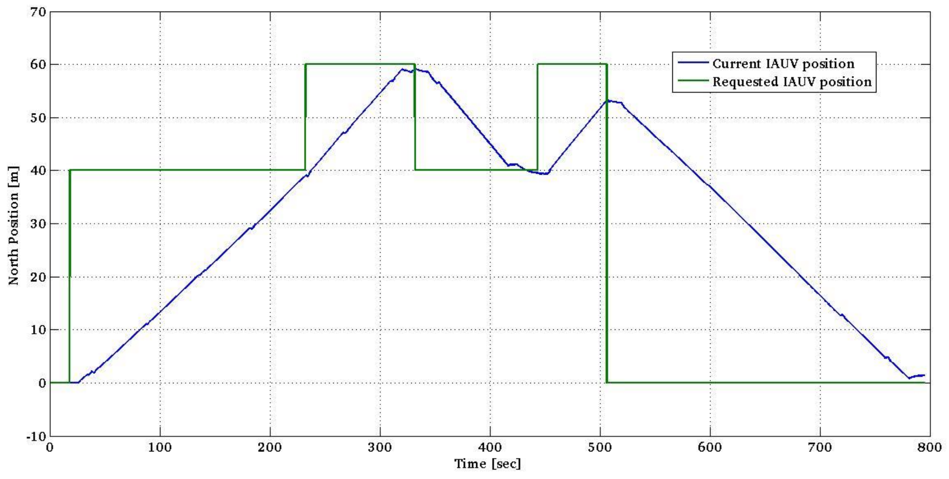


Figure 45. Comparison between the target position and the north IAUV position when manipulating the object (target).

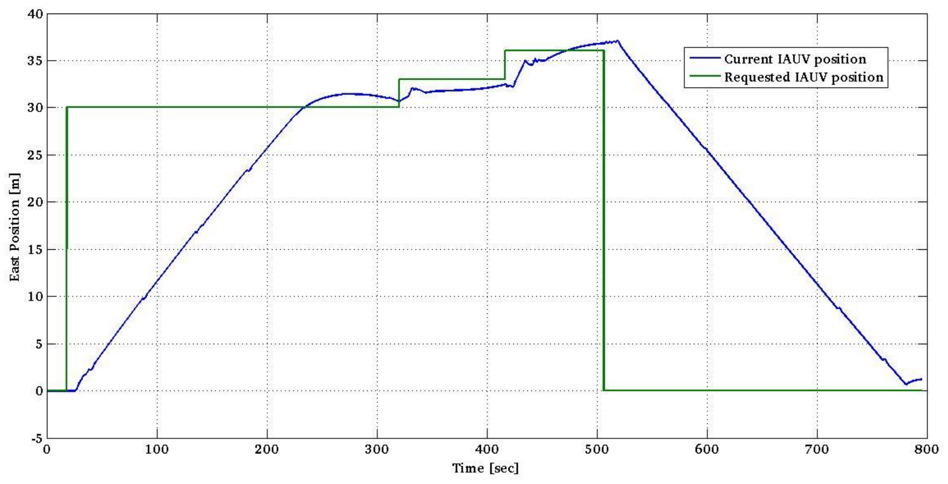


Figure 46. Comparison between the target position and the east IAUV position when manipulating the object (target).

The two vehicles, i.e. the ASC and the IAUV, go to a safe position in case the communication is lost with the mission planner that runs in the OCU.

6.2 Sea Trial

The sea trial entails experiments based on cooperative navigation. The trials carried out in the sea involve key control operations of the ICA for maritime vehicles, i.e. the Delfim ASC (Figure 48), and the Nessie AUV (Figure 49). These operation are homing, docking, leader following, terrain following for the Nessie AUV, and path following for the Delfim ASC. The results obtained from a leader-following operation are only presented in this Thesis.

6.2.1 Sea Trial Setup

The main setup requirement to run ICA implementation on the maritime vehicles, i.e. the Delfim ASC, and the Nessie AUV, is to execute the roscore [39] in each of them and the OCU. All the ROS services, including the roscore are launched from the command line from a remote Linux terminal as in the simulation case presented in the previous subsection. The setup sequence for the ROS-based applications required is as follows:

1. Run ROS core ('~$**roscore**')

2. Run Service Matchmaker (‘~$**rosrun** matchmaker matchmaker_server’)

3. Run each of the ROS-based services needed to provide the Delfim ASC and the Nessie AUV with the capabilities to tackle a leader-following operation.

Advertisement and discovery of ROS-based services as well as the matchmaker operation are as described for the simulation (previous subsection). The leader-following experiment only focusses on the guidance, navigation and control of the Delfim ASC and the Nessie AUV for the mission.

6.2.2 Leader-Following Trial

After setting up the sea trial environment as indicated in Sub-subsection 6.2.1, the execution sequence of the ROS services required by the leader-following trial are:

1. Run the path planner (‘~$**rosrun** stub_path_planner path_planner) in the OCU.

2. Run the leader follower (‘~$**rosrun** leaderfollowing LeaderFollowing) in the Nessie AUV.

3. Run the planner (‘~$**rosrun** planner Planner) the Delfim ASC and the Nessie AUV.

No particular path was provided for the Delfim ASC to be followed but a random path. The object is to demonstrate that the Nessie AUV can follow the Delfim ASC. The execution sequence of the above ROS-based applications is as follows:

1. The trial is started by running all the ROS-based applications as described above in both maritime vehicles.

2. The leader-following operation can be terminated at any point by shutting down the main ROS-based service provided by the leader follower module running in the Nessie AUV.

The results from a combination of two maritime vehicles are cooperative navigation which involves the Delfim ASC path following operation and the Nessie AUV leader following operation. Since the manipulation capability is not required for this trial, the underwater vehicle utilized is the Nessie AUV.

Figure 47 shows the scenario and interaction among AMR system actors for the cooperative navigation. The Delfim ASC performs a path following operation whist the Nessie AUV performs a leader following operation where the leader is the Delfim ASC. Thus the Delfim ASC follows a path given, and the Nessie AUV follows the Delfim ASC.

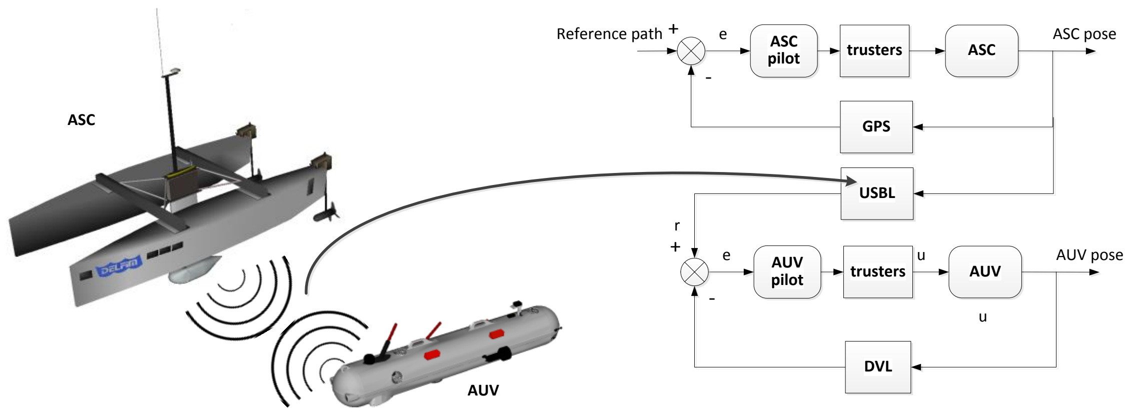


Figure 47. Cooperative navigation operation; Interaction scenario (on the left), and control loop (on the right).

The control diagram presented on the right of Figure 47 shows the logical cross-vehicle control connection between the Delfim ASC and the Nessie AUV. In practice, this link is physically done by means of the acoustic signal transmitted and received by the USBL positioning system. The USBL configuration used for this trial is called “inverted USBL” since the location of the USBL parts are swapped, the opposite of the traditional setup (USBL transponder is usually fixed in some location in the water, and the USBL transducer is attached in a vessel). Therefore, in this trial the part of the USBL (transponder) on board the Delfim ASC acts as a mobile beacon or reference waypoint that is the set point for the control loop of the Nessie AUV which payload is the USBL transducer (sender and receiver).

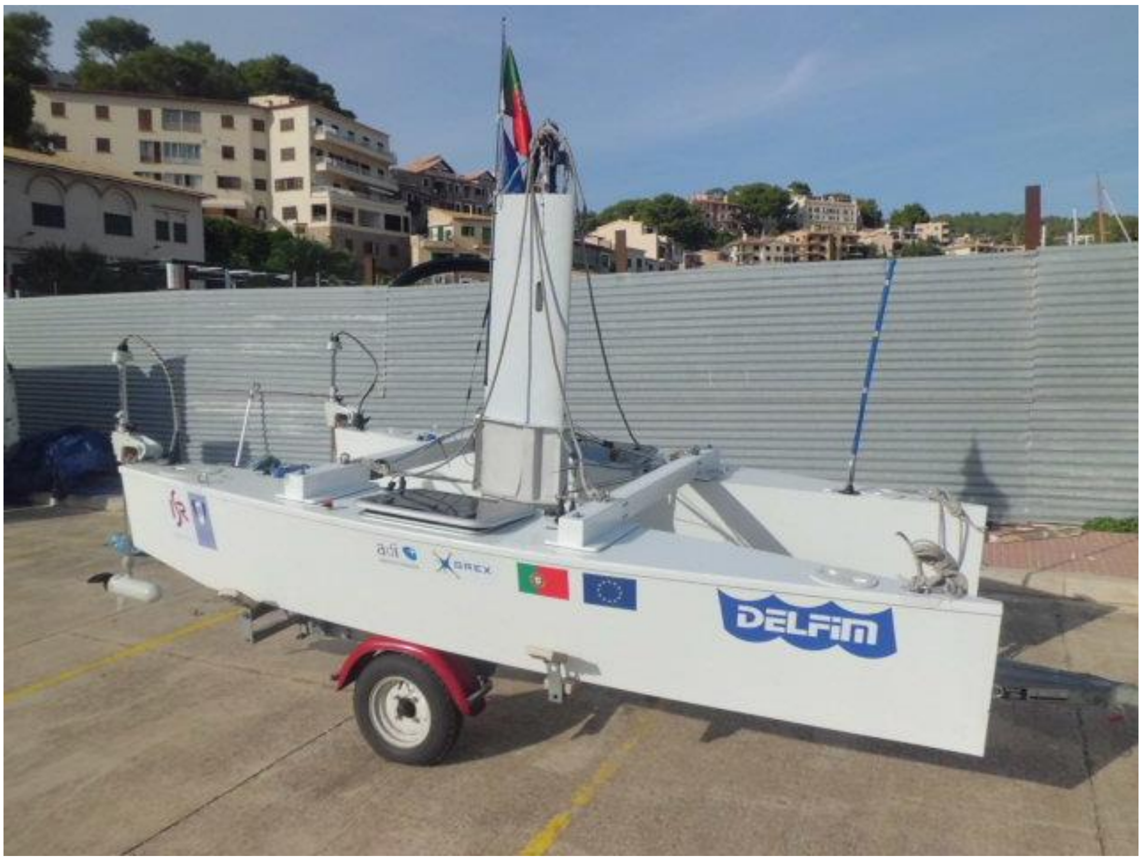


Figure 48. The Delfim ASV used for the trials.

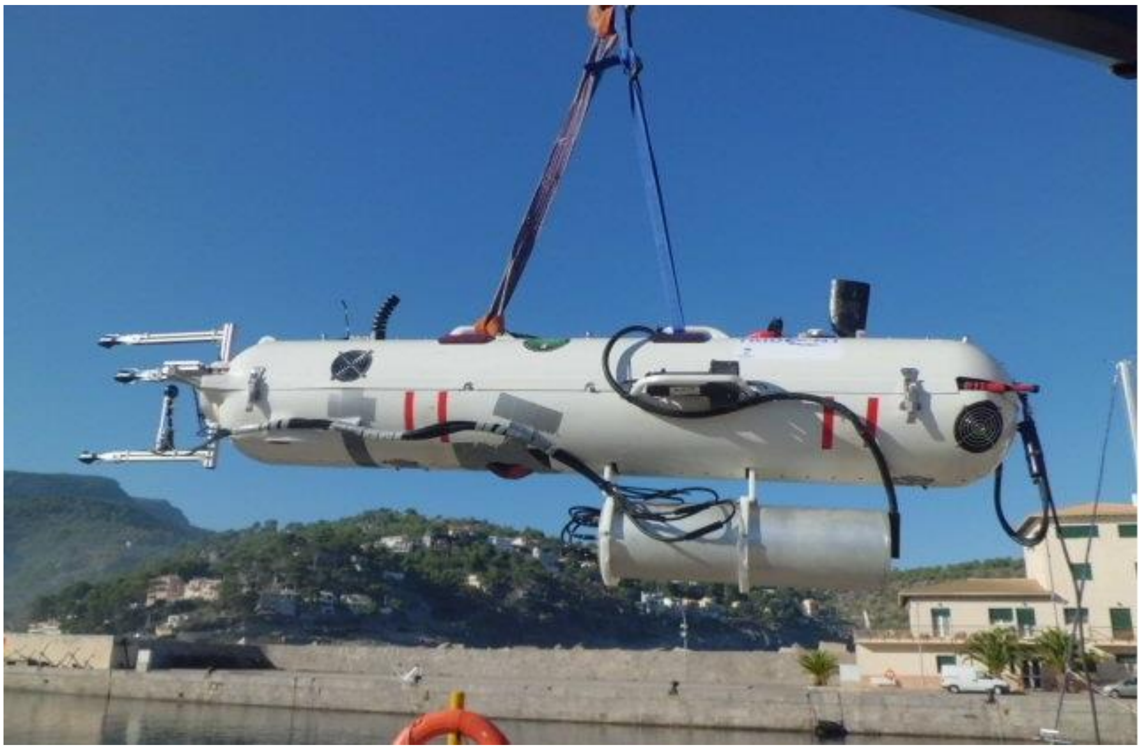


Figure 49. The Nessie AUV plus USBL deployment during trials.

Figure 50 shows the path followed by the Delfim ASC, and how the Nessie AUV follows that path by following the Delfim ASC.

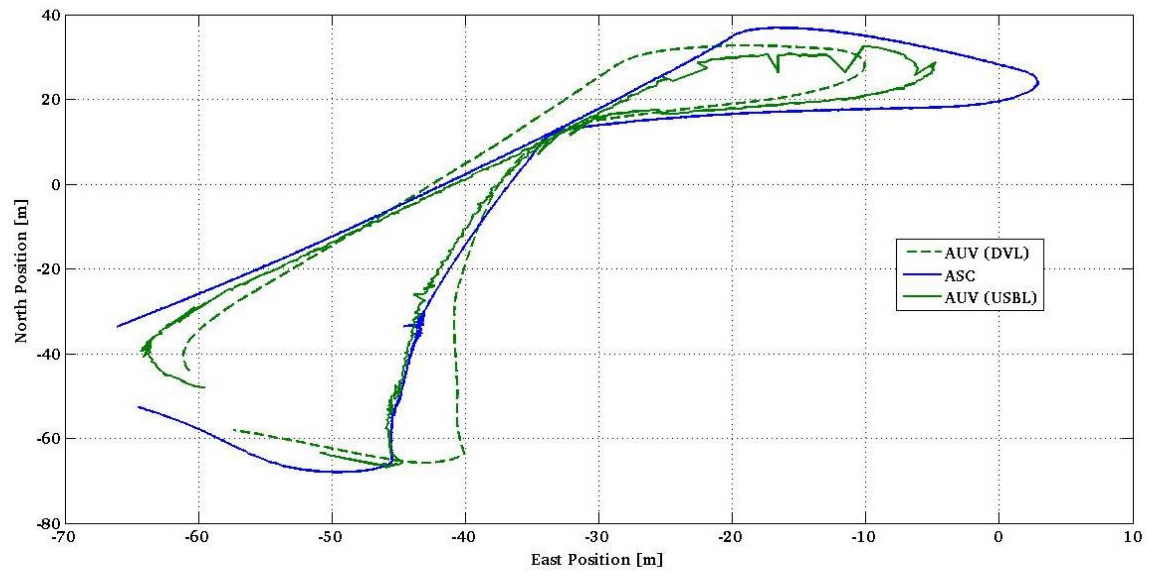


Figure 50. Path followed by the ASC (leader), and the AUV (follower).

The Delfim ASC position is given by the GPS, and the Nessie AUV position is computed based on the data from the DVL, gyrocompass, and the USBL as shown in Figure 47. When the Nessie AUV is on the surface it can take into account the GPS and compass measurements to know its position with respect to the world. When the Nessie AUV is submerged, it can compute its motion based on a DVL and a fiber optic gyroscope. Additionally, the USBL can provide data about the distance and heading with respect to the Delfim ASC. There is an angular rotation between the two Nessie AUV plots (DVL dead reckoning and USBL). This angular rotation is due to the error on the measurement of the compass during the initialization of navigation but does not affect the Nessie AUV positioning which is defined based on the USBL data and the current Delfim ASC position.

Figure 51 and Figure 52 show the posing data from the ASC and the Nessie AUV navigators while performing the above described cooperative operation. The figure also includes the Nessie AUV navigation data from the DVL.

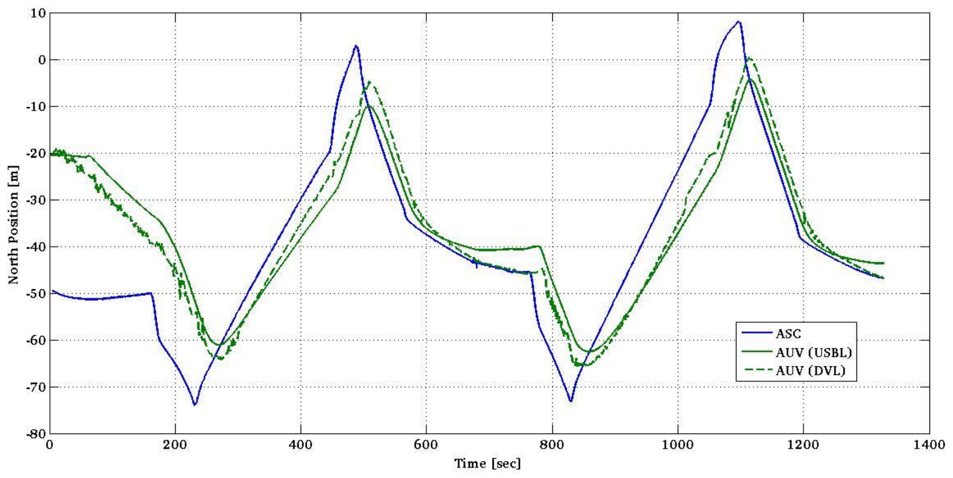


Figure 51. Comparison of north position of the ASC Delfim and the Nessie AUV for cooperative navigation.

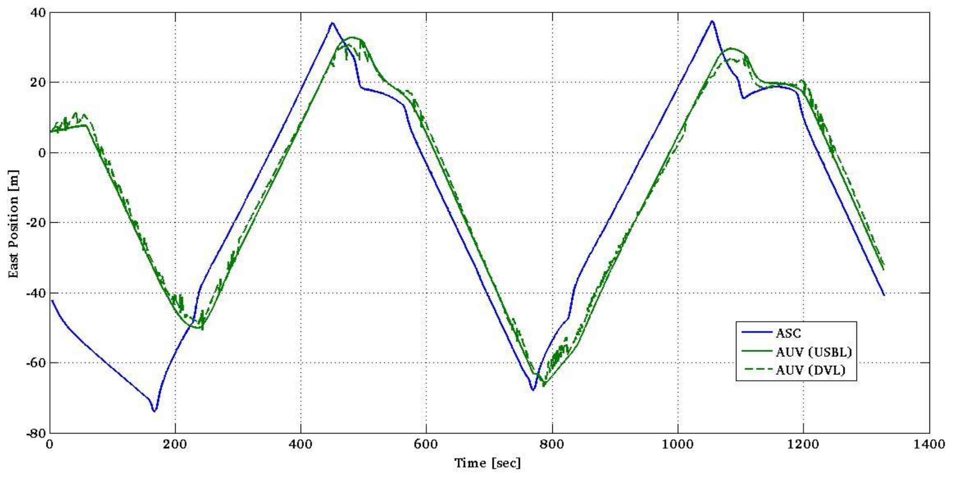


Figure 52. Comparison of east position of the ASC Delfim and the Nessie AUV for cooperative navigation.

The above figures are also to show the benefits provided by using a USBL approach. The position computed based on the speed provided by the DVL (with respect to the Nessie AUV reference frame) is now replaced by the position information provided by the USBL positioning system. This is an improvement in term of localization but most importantly it is the means by which cooperative navigation is carried out.

Chapter 7: Conclusions and Future Work

7.1 Conclusions

This Thesis has presented the fundamental aspects of service-oriented agents that are the core of an Intelligent Control Architecture (ICA) for autonomous marine vehicles. The ICA is generic in nature but aimed at a case study where a marine surface craft and an underwater vehicle are required to work cooperatively. Nevertheless, this fact does not invalidate the architectural approach proposed since the ICA principles of flexibility and adaptability are based on service orientation and agent technology which have been successful in many application domains by providing flexible and adaptable solutions.

The ICA foundation lies on the basic service infrastructure of service-oriented computing, i.e. discovery of system capabilities, dynamic system reconfiguration, and decoupled interaction among applications. The approach also improves the SA of the above vehicles by relying on Observe-Orient-Decide-Act (OODA) loops performed by agents that combine their knowledge and skills with the information acquired during missions. The above architectural elements make it possible to achieve adaptive and reflective mission planning based on a dynamic reconfiguration of plans according to given mission.

The ICA was implemented in the pervasive Robot Operating System (ROS) middleware. Computer simulations and trials of the ICA implementation show the system performance (including the advertisement and discovery mechanisms for services as well as the resource management) were successfully carried out. An experiment of the ICA performance in scenario including faults has also been presented. It shows the greatest potential of the research contribution since maritime vehicles are able to make in-mission decisions without contacting or getting back to the human operator for advice on what to do.

The promising ICA approach is a general solution for maritime autonomy that opens opportunities to be applied to other maritime missions, and Unmanned Marine Vehicles (UMVs). In fact, the knowledge of different missions has to be added to the ontological database and the reasoner has to include updated rules needed to deal with the new operational situations. Additionally, the ICA is platform-independent from the software point of view. Thus, only high-level changes are required in the ICA implementation. The software library provides means to help implementing and integrating any new capability added to the system (no matter whether it is a new AMR system or an existing one).

The autonomous characteristics of the ICA pave the way for a reduction in the expensive deployment and operation of Remotely Operated Vehicles (ROVs), and bring within reach complex multi-vehicle collaborative missions that were previously too costly or logistically infeasible.

7.2 Future Work

There is still investigation to be made into this promising architectural approach. The main aspect to be dealing with in order to optimize the ICA are: knowledge representation and reasoning, and diverse application cases with a strong emphasis on faults (including errors and failures) and marine missions (including a broad set of environmental situations).

The knowledge representation is critical for any decision-making process. Knowledge is a cognitive system property. Data with meaning is information which in turn becomes knowledge when a purpose and the potential to generate action are added. Knowledge is the intellectual machinery used to achieve goals (carrying out actions), and create new information. Artificially-intelligent systems such the ICA-based ones are able to accurately determine what activity will maximize the likelihood of achieving a goal successfully. The ICA has knowledge-based intelligence that relies on ontological reasoners. Semantic dependencies in ontology as well as reasoning rules for ontological inference are essential to have a developmental ICA intellect. Therefore, future steps of this research would be to increase the autonomous maritime capabilities of the AMR systems by the improving ontological database, and the reasoner.

New research directions will also take into account more complex evaluation scenarios by including other potential faults and unexpected situations in order to deal with some faults to be handled through self-repairing capabilities. There are different types of faults to be considered that can arise at the deliberative control layer and from (1) vehicle problems or (2) environmental conditions; or at the reactive control layer, and from (3) vehicle problems or (4) environmental conditions. Considering different faults will allow an updated ICA approach to be able to cope with more realistic operation scenarios, involving software and hardware problems as well as environment changes.

7.3 Exploitation

Opportunities to exploit the outcomes from this Thesis are within the following sectors: research, academics, and industry.

- **Scientific sector.** Contributions to scientific research can be divided into two main branches: state of the art, and research applications. The former represent the current pool of robotic control architectures (including those for any domain). A potential impact can be on competitions of robots such as EURATHLON [32] (a robot competition supported by the European Commission in the FP7), and the DARPA Robotics Challenge [34]. The latter involves tasks from activities related to oceanographic monitoring systems, and marine biology studies so it could take into account the ICA for measuring UMV performance (back-end technology based on the ICA approach).

- **Academic sector.** Practical work in laboratory classes can implement the ICA as a development platform for student projects. Also, appropriate lectures can include the ICA approach as an application example of adaptive solution for robotics control architectures. The ICA applicability can be expanded in order to cope with other application domain so it can be taken into account in academic activities from other engineering field such as software and systems engineering as well as sciences such as computer science. The Student Autonomous Underwater Competition – Europe (SAUC-E) [33] can also benefit from the ICA. It is always looking for novel robot architectures to get high levels of adaptability for missions (in particular, different environments).

- **Industrial sector.** Defence industries and UK Ministry of Defence initiatives can also be interested in the ICA and its potential use for ocean/sea mission such as mine countermeasures. On one hand, there is an increasing concern for autonomy metrics from cooperative solutions for intelligent searches. On the other hand, heterogeneous robot teams (including water, land, and air) are in the thick of defence priorities. Thus, a technology that allows stakeholders to carry out collaborative operations based on an on-demand capability approach are becoming essential for missions such as recce, and rescue of people.

Appendix A: Core Ontology Entities

A.1 System Architecture Element

Figure 53 shows the subclasses of the system architecture element class of the core ontology.

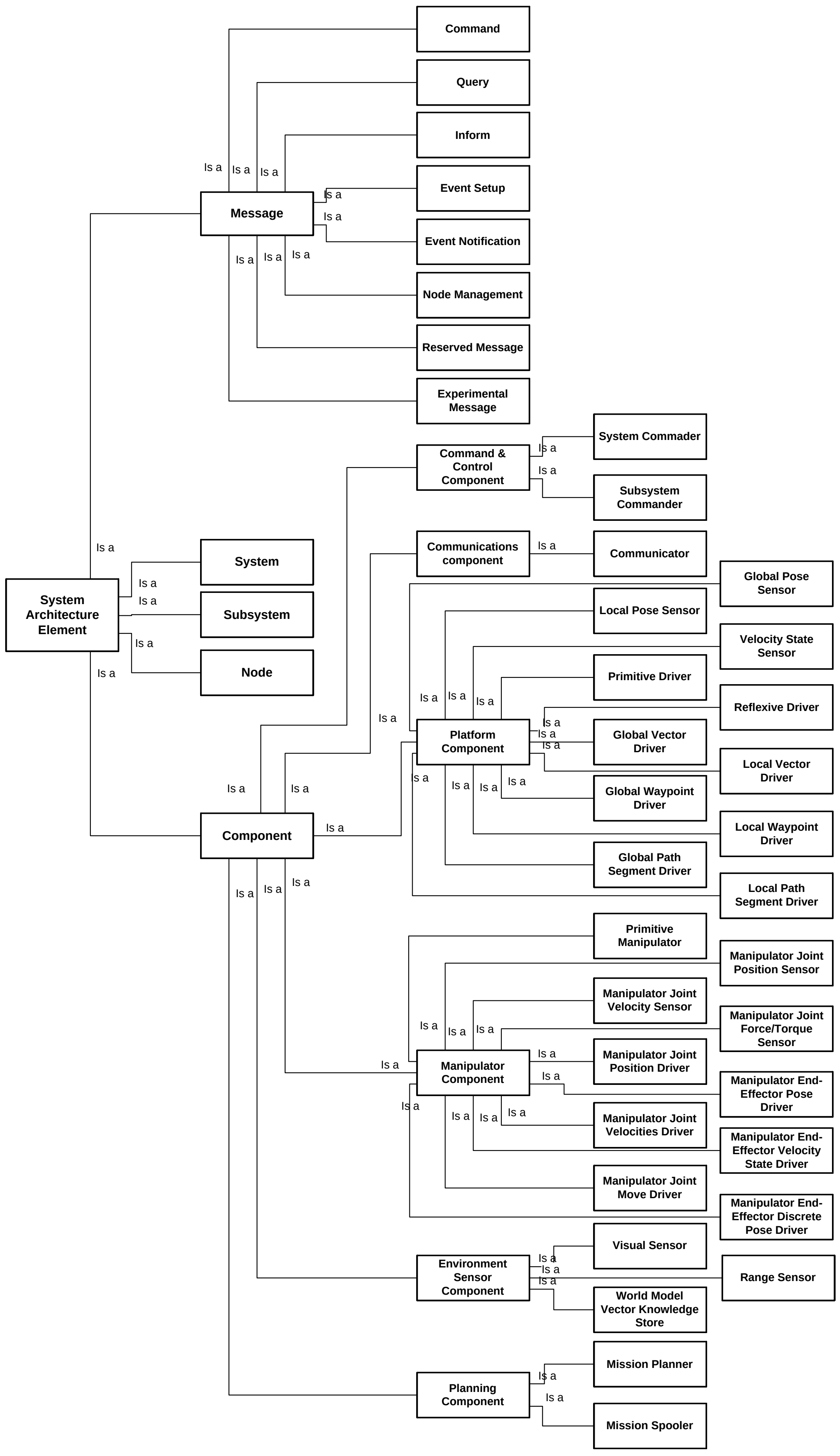


Figure 53. Subclasses of the system architecture element class of the core ontology

A.2 System Functionally

Figure 54 shows the subclasses of the system functionality of the core ontology.

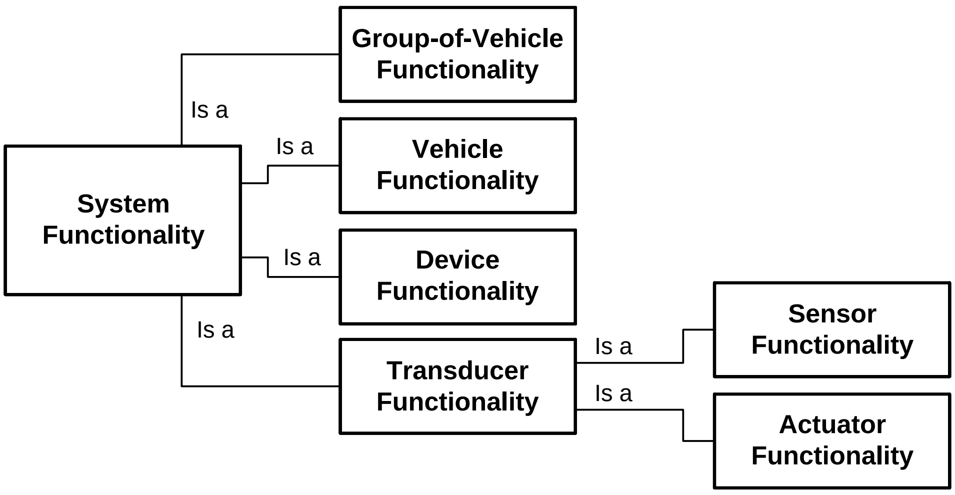


Figure 54. Subclasses of the system functionality of the core ontology

A.3 System Service

Figure 55 shows the subclasses of the system service of the core ontology.

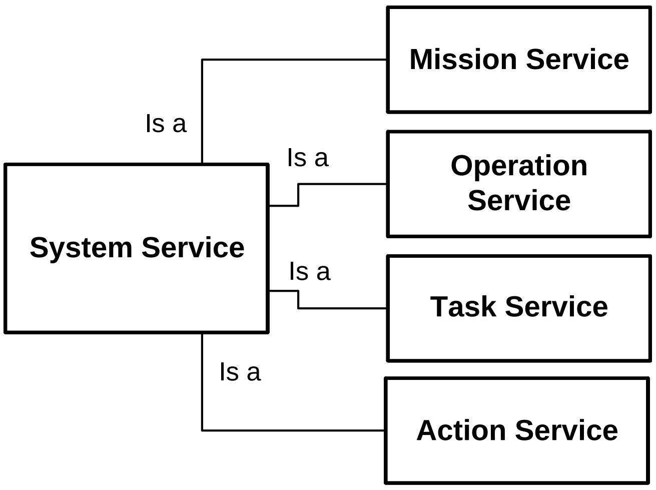


Figure 55. Subclasses of the system service of the core ontology

A.4 System Status

Figure 56 shows the subclasses of the system status of the core ontology.

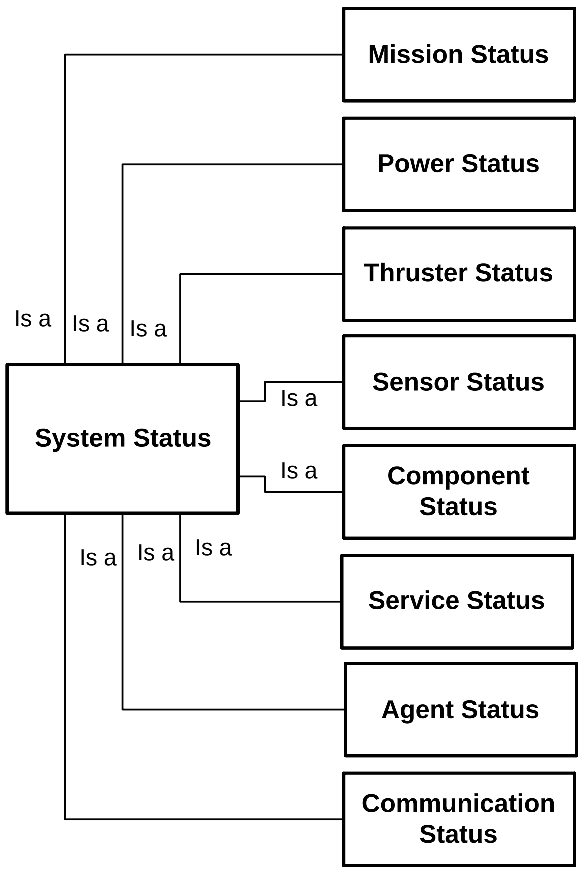


Figure 56. Subclasses of the system status of the core ontology

A.5 Data Parameter

Figure 57 shows the subclasses of the data parameter of the core ontology.

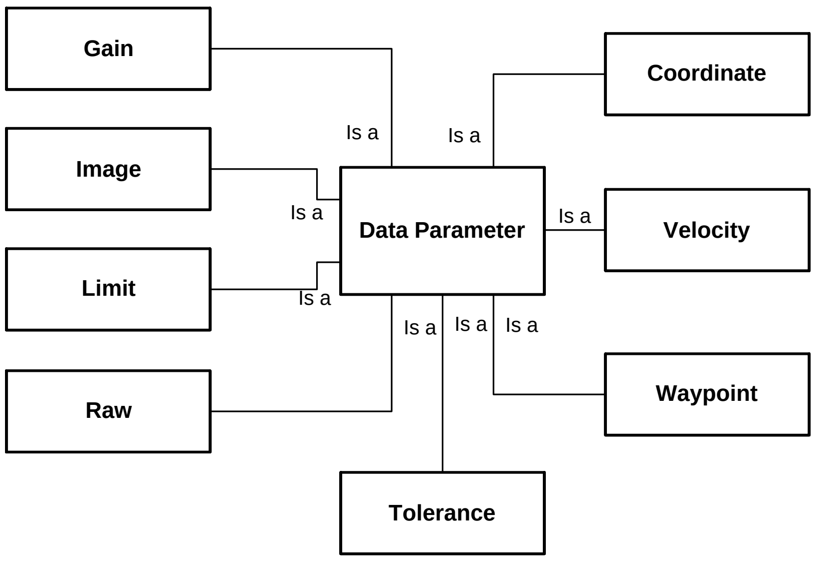


Figure 57. Subclasses of the data parameter of the core ontology

A.6 System Entity

Figure 58 shows the subclasses of the system entity of the core ontology.

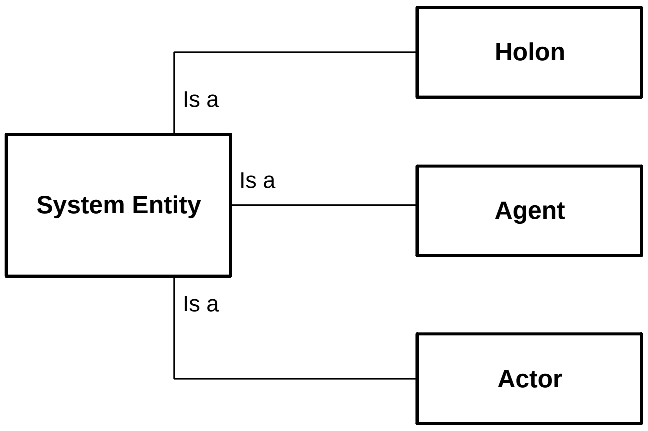


Figure 58. Subclasses of the system entity of the core ontology

A.7 System Element

Figure 59 shows the subclasses of the system element of the core ontology.

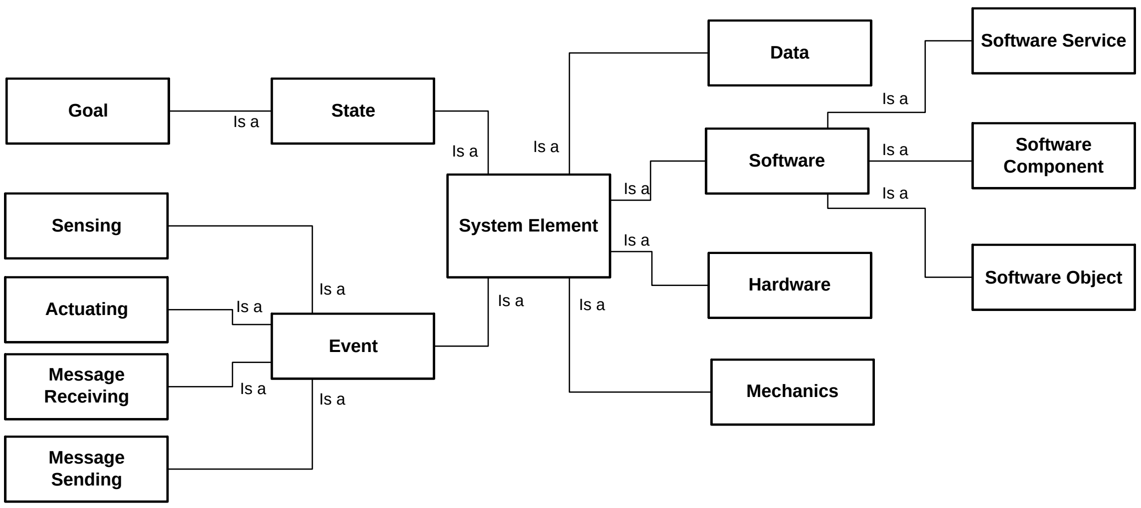


Figure 59. Subclasses of the system element of the core ontology

A.8 System Capability

Figure 60 shows the subclasses of the system capability of the core ontology.

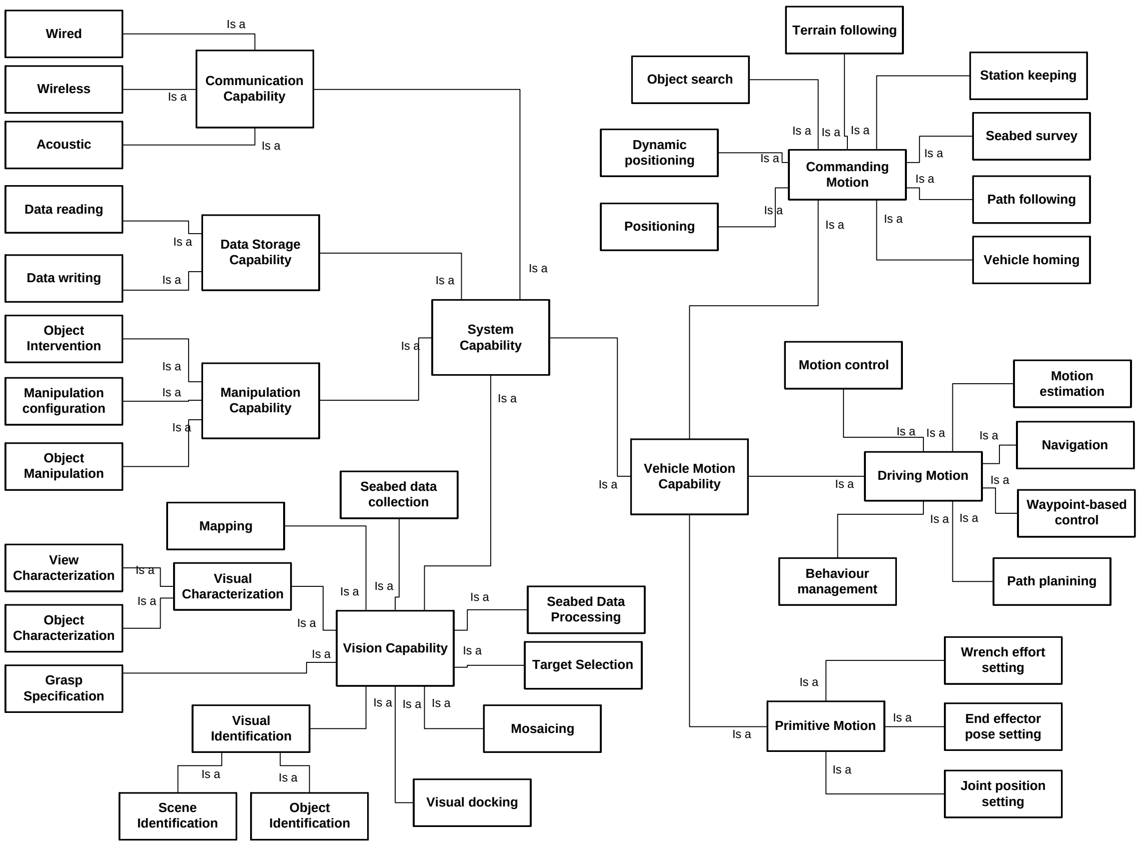


Figure 60. Subclasses of the system capability of the core ontology

A.9 System Target

Figure 61 shows the subclasses of the system target of the core ontology.

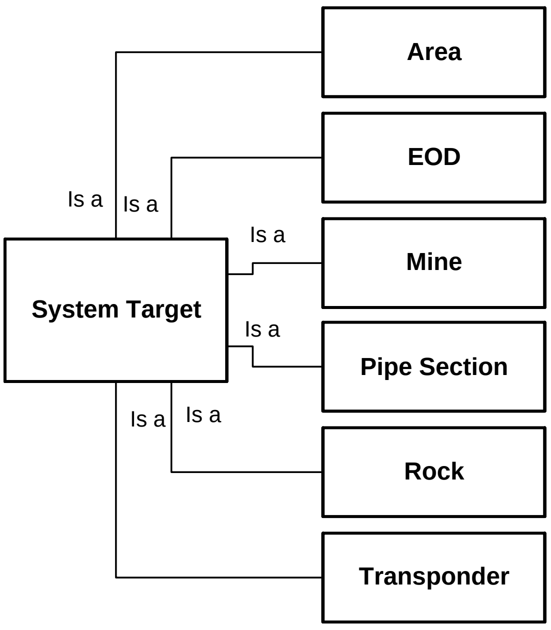


Figure 61. Subclasses of the system target of the core ontology.

Appendix B: System Functionalities

B.1 Manipulation Controller

Table 9. Functionality of the manipulation controller

Component	Manipulation Controller
Function	Set intervention configuration
Description	It sets the configuration for intervention of a given object
Inputs	Configuration parameters
Outputs	None
Function	Perform intervention
Description	It performs an intervention on an object according to the input information
Inputs	Object characterization and task rate
Outputs	Arm plus hand status

B.2 Mission Planner

Table 10. Functionality of the mission planner

Component	Mission Planner
Function	Build plan
Description	It builds the plans required by agents according to the activities they have to carry out in order to achieve a given mission.
Inputs	Mission and goals
Outputs	Mission plan (sequence of activities to be carried out)

B.3 Navigator

Table 11. Functionality of the navigator

Component	Navigator
Function	Send navigation data
Description	It provides the functionality to estimate the vehicle pose.
Inputs	Navigation sensors
Outputs	Vehicle pose and measurement quality

B.4 Ethernet Controller

Table 12. Functionality of the Ethernet controller

Component	Ethernet controller
Function	Receive data
Description	It receives data from the Ethernet-enabled vehicles.
Inputs	None
Outputs	Data received
Function	Send data
Description	It sends data out to other Ethernet-enabled vehicles.
Inputs	Data to be sent
Outputs	Communication status

B.5 Waypoint-Based Controller

Table 13. Functionality of the waypoint-based controller

Component	Waypoint-Based Controller
Function	Set waypoint list
Description	It controls the vehicle motion according to the given waypoints.
Inputs	List of waypoints
Outputs	Commands to thrusters

B.6 Odometry

Table 14. Functionality of the odometry

Component	Visual Odometry
Function	Collect data
Description	It gathers data about the seabed from cameras
Inputs	None
Outputs	Pre-processed data collected
Function	Estimate motion
Description	It estimates the motion that the IAUV must have in order to follow the terrain properly.
Inputs	None
Outputs	Motion data

B.7 Manipulation Identifier

Table 15. Functionality of the manipulation identifier

Component	Manipulation Identifier
Function	Identify View
Description	It identifies the view according to the scene specified while searching for the object of interest.
Inputs	Current frame, view descriptor
Outputs	Status (object identified or not)
Function	Object identification
Description	It identifies the object according to the object of interest selected by the end user.
Inputs	Current frame, object descriptor
Outputs	Status (object identified or not)

B.8 Manipulation Specifier

Table 16. Functionality of the manipulation specifier

Component	Manipulation Specifier
Function	Identify scene
Description	It characterized the scene of interest.
Inputs	Mosaic sub-section (input from user)
Outputs	View descriptor, status (view could be identified or not)
Function	Identify object
Description	After scene identification, it characterizes the object to be manipulated.
Inputs	Mosaic sub-section (input from user)
Outputs	Ojbect descriptor, status (view could be identified or not)

B.9 Visual Docking Controller

Table 17. Functionality of the visual docking controller

Component	Visual Docking Controller
Function	Dock IAUV
Description	It is able to make the IAUV dock the ASC without any end-user intervention, and with the help of the cameras mounted in the IAUV.
Inputs	Vehicle current pose, and pose measurement quality
Outputs	Vehicle docking status (docked or not docked)

B.10 Vehicle Motion Controller

Table 18. Functionality of the vehicle motion controller

Component	Vehicle Motion Controller
Function	Set waypoint list
Description	It controls the IAUV motion according to information provided by the path planner.
Inputs	List of waypoints
Outputs	Commands to thrusters

B.11 System Commander

Table 19. Functionality of the system commander

Component	System Commander
Function	Set mission
Description	It sets the mission required by the end user.
Inputs	Seabed survey, target selection, or object manipulation
Outputs	Mission status, and results from the missions

B.12 Modem Manager

Table 20. Functionality of the modem manager

Component	Modem Manager
Function	Receive data
Description	It receives data from the acoustic modem-enabled vehicles.
Inputs	None
Outputs	Data received
Function	Send data
Description	It sends data out to other Ethernet-enabled vehicles.
Inputs	Data to be sent
Outputs	Communication status

B.13 Mapper

Table 21. Functionality of the mapper

Component	Mapper
Function	Send map
Description	It provides the functionality to map the seabed.
Inputs	Mapping sensors and navigator
Outputs	Map

B.14 Path Planner

Table 22. Functionality of the path planner

Component	Path Planner
Function	Set path plan
Description	It provides the functionality to plan the path to be followed by the vehicles.
Inputs	Start and end points
Outputs	Path plan

B.15 Path Follower

Table 23. Functionality of the collaboration path follower

Component	Collaboration Path Follower
Function	Manage behaviour
Description	It manages the vehicles behaviour in get collaboration between them for path following.
Inputs	Commands and list of waypoints
Outputs	Status

B.16 Data Storage

Table 24. Functionality of the data storage

Component	Data Storage
Function	Read data
Description	It reads data from the data storage.
Inputs	Data requested
Outputs	Data read
Function	Write data
Description	It writes data in data storage.
Inputs	Data to be written
Outputs	Write status

Appendix C: System Services

C.1 Definitions of Services

Table 25 shows all of the services identified in the AMR system according to the hierarchy above proposed.

Table 25. Services identified in the system

Service	Description	Category
Seabed Survey	It provides the functionality to carry out the exploration of the area of interest.	Mission
Target Selection	It provides the functionality to carry out the selection of the object of interest.	
Object Manipulation	It provides the functionality to carry out the object manipulation.	
OCU Ethernet Data Transfer	It provides the functionality to transfer data from and to IUAV.	Operation
Seabed Data Processing	It provides the functionality to process the seabed data collected.	
ASC Positioning	It provides the functionality to position the ASC at a given pose.	
Leader Following	It provides the functionality to make the ASC follow a particular path.	
Dynamic Positioning	It provides the functionality to make the ASC be at a given pose while.	
IAUV Positioning	It provides the functionality to position the IUAV at a given pose.	
Terrain Following	It provides the functionality to make the IAUV follow a particular terrain.	
Pattern Search	It provides the functionality to make the IAUV perform a search for the object of interest.	
IAUV Homing	It provides the functionality to make the IAUV dock to the ASC.	
Object Search	It provides the functionality to search for objects of interest.	

IAUV Ethernet Data Transfer	It provides the functionality to transfer data from and to OCU.	
Intervention Configuration	It provides the functionality to configure the manipulator mounted in the IAUV.	
Object Intervention Manoeuvre	It provides the functionality to manipulate objects of interest when the IAUV is required to perform station keeping while manipulating.	
Floating Manipulation	It provides the functionality to manipulate objects of interest when the IAUV is in free floating mode.	
Seabed Image Mosaicing	It provides the functionality to build the mosaic of the seabed.	Task
Station Keeping	It provides the functionality to search for objects of interest.	
Object Characterization	It provides the functionality to specify the characteristics of the objects of interest.	
View Characterization	It provides the functionality to specify the characteristics of the view of interest.	
Grasp Specification	It provides the functionality to specify the way to grasp the objects of interest.	
Behaviour Management	It provides the functionality to control the motion of the ASC.	
Waypoint List Setting	It provides the functionality to determine the waypoints required to set the desired motion of the ASC.	
ASC Navigation Data Sending	It provides the functionality to send out the ASC pose in order to plan the path to be followed by it.	
IAUV Navigation Data Sending	It provides the functionality to send out the IAUV pose in order to plan the path to be followed by it.	
Motion Control	It provides the functionality to control the motion of the IAUV.	
Path Plan Setting	It provides the functionality to plan the path to be followed by the IAUV.	
Obstacle Map Generation	It provides the functionality to provide an obstacle map when resquested.	
Map Sending	It provides the functionality to send the seabed map out.	
Seabed Data Collection	It provides the functionality to collect data from the seabed.	
Motion Estimation	It provides the functionality to estimate the IAUV motion when collecting seabed data.	
Vehicle Docking	It provides the functionality to dock visually the IAUV to the ASC.	
Object Identification	It provides the functionality to identify the target (object of interest) when the IAUV is heading to the intervention area after the target selection.	
Scene Identification	It provides the functionality to identify the scene (where the object of interest is) when the IAUV is heading to the intervention area after the target selection.	
ASC Operation Area Setting	It provides the functionality to set the area where the ASC operates (stand by) in order to support the IAUV.	
IAUV Operation Area Setting	It provides the functionality to set the area where the IAUV operates (begin looking for the object selected).	

Intervention Configuration Setting	It provides the functionality to configure the intervention strategies for the objects of interest.	
Object Manipulation Control	It provides the functionality to make the specified intervention on the object selected.	
ASC Modem Data Receiving	It provides the functionality to enable the ASC to receive data from the acoustic modem.	
ASC Modem Data Sending	It provides the functionality to enable the ASC to send data through the acoustic modem.	
IAUV Modem Data Receiving	It provides the functionality to enable the IAUV to receive data from the acoustic modem.	
IAUV Modem Data Sending	It provides the functionality to enable the IAUV to send data through the acoustic modem.	
OCU Ethernet Data Receiving	It provides the functionality to enable the OCU to receive data from an Ethernet port.	
OCU Ethernet Data Sending	It provides the functionality to enable the OCU to send data through an Ethernet port.	
ASC Ethernet Data Receiving	It provides the functionality to enable the ASC to receive data from an Ethernet port.	
ASC Ethernet Data Sending	It provides the functionality to enable the ASC to send data through an Ethernet port.	
IAUV Ethernet Data Receiving	It provides the functionality to enable the IAUV to receive data from an Ethernet port.	
IAUV Ethernet Data Sending	It provides the functionality to enable the IAUV to send data through an Ethernet port.	
Vehicle Motion Driving	It provides the functionality to command the necessary wrench effort in order to move the marine vehicle to the waypoints given.	
End Effector Driving	It provides the functionality to command the necessary wrench effort in order to move the end effector to the position given.	Action
Joint Effector Driving	It provides the functionality to command the necessary wrench effort in order to move the joint effector to the position given.	

Table 26 shows the composition of the mission services by means of operation services listed in Table 27.

Table 26. Mission Services

Service	**System**	**Composition based on operation services (**Table 27**)**
Seabed Survey {MS} (Figure 62)	AMR	**(**ASC Positioning || IAUV Positioning**) + (**Leader Following || Terrain Following**) +** IAUV Homing
Target Selection {MS} (Figure 63)	AMR	**(**OCU Ethernet Data Transfer || IAUV Ethernet Data Transfer) + Seabed Data Processing
Object Manipulation {MS} (Figure 64)	AMR	**(**OCU Ethernet Data Transfer || IAUV Ethernet Data Transfer**) +** Intervention Configuration **+** Manipulation Configuration **+ (**ASC Positioning || IAUV Positioning**) + (**Dynamic Positioning || Terrain Following**) +** Object Intervention Manoeuvre **+** IAUV Homing

Table 27 shows the composition of the operation services by means of task services listed in Table 28.

Table 27. Operation Services

Service	**Subsystem**	**Component**	**Composition based on task services (**Table 28**)**
OCU Ethernet Data Transfer {OS} (Figure 71)	OCU	System Commander	OCU→IAUV: OCU Ethernet Data Sending IAUV→OCU: OCU Ethernet Data Receiving
Seabed Data Processing {OS} (Figure 73)			Seabed Image Mosaicing + View Characterization + Object Characterization + Grasp Specification
ASC Positioning {OS} (Figure 65)	ASC	ASC Planner	ASC Navigation Data Sending + Waypoint List Setting
Leader Following {OS} (Figure 67)			ASC Modem Data Receiving + Behaviour Management + Waypoint List Setting + Motion Control + ASC Modem Data Sending
Dynamic Positioning {OS} (Figure 75)			ASC Navigation Data Sending + Waypoint List Setting
IAUV Positioning {OS} (Figure 66)	IAUV	IAUV Planner	IAUV Navigation Data Sending + Path Plan Setting + Motion Control
Terrain Following {OS} (Figure 68)			IAUV Modem Data Receiving + Path Plan Setting + **(**Motion Control + Seabed Data Collection || Motion Estimation**)** + ASC Modem Data Sending
Pattern Search {OS} (Figure 70)			IAUV Modem Data Receiving + Path Plan Setting + Motion Control + Scene Identification + Object Identification + ASC Modem Data Sending
IAUV Homing {OS} (Figure 66)			Path Plan Setting + Vehicle Docking
IAUV Ethernet Data Transfer {OS} (Figure 72)			IAUV →OCU: IAUV Ethernet Data Sending OCU→IAUV: IAUV Ethernet Data Receiving
Intervention Configuration {OS} (Figure 74)			ASC Operation Area Setting + IAUV Operation Area Setting + Intervention Configuration Setting
Object Intervention Manoeuvre {OS} (Figure 76)			Station Keeping || Object Manipulation Control
Floating Manipulation {OS} ()			Motion Control || Object Manipulation Control

Table 28 shows the task services. Some of them have embedded in turn action services listed in Table 29.

Table 28. Task Services

Service	**Node**	**Component**	**Embedding** (Table 29)
Seabed Image Mosaicing {TS}	OCU Vision Processor	Image Processor	None
Object Characterization {TS}		Manipulation Specifier	None
View Characterization {TS}			None
Grasp Specification {TS}			None
OCU Ethernet Data Receiving {TS}	OCU Master Controller	OCU Ethernet Controller	None
OCU Ethernet Data Sending {TS}			None
Behaviour Management {TS}	ASC Mobility Controller	ASC Collaboration Path Follower	None
Waypoint List Setting {TS}		Waypoint-Based Controller	Vehicle Motion Driving
ASC Navigation Data Sending {TS}		ASC Navigator	None
ASC Modem Data Receiving {TS}	ASC Master Controller	ASC Modem Manager	None
ASC Modem Data Sending {TS}			None
ASC Ethernet Data Receiving {TS}		ASC Ethernet Controller	None
ASC Ethernet Data Sending {TS}			None
ASC Operation Area Setting {TS}		ASC Data Storage	None
IAUV Navigation Data Sending {TS}	IAUV Mobility Controller	IAUV Navigator	None
Path Plan Setting {TS}		Path Planner	None
Obstacle Map Generation {TS}		Mapper	None
Map Sending {TS}			
Motion Control {TS}		Motion Controller	Vehicle Motion Driving
Station Keeping {TS}			None
IAUV Modem Data Receiving {TS}	IAUV Master Controller	IAUV Modem Manager	None
IAUV Modem Data Sending {TS}			None
IAUV Ethernet Data Receiving {TS}		IAUV Ethernet Controller	None
IAUV Ethernet Data Sending {TS}			None
IAUV Operation Area Setting {TS}		IAUV Data Storage	None
Seabed Data Collection {TS}	IAUV Vision Processor	Visual Odometry	None
Motion Estimation {TS}			None
Vehicle Docking {TS}		Visual Docking Controller	None
Object Identification {TS}		Manipulation Identifier	None
Scene Identification {TS}			None
Intervention Configuration Setting {TS}	Manipulation Processor	Manipulation Controller	None
Object Manipulation Control {TS}			End Effector Driving || Joint Effector Driving

Table 29 shows the action services.

Table 29. Action Services

Service	Component
Vehicle Motion Driving {AS}	Thruster Driver
End Effector Driving {AS}	End-Effector Driver
Joint Effector Driving {AS}	Joint Effector Driver

Section A.1 in Appendix A shows the description of all the services of the AMR system in details, and Section A.2 in Appendix A shows how orchestration and choreography of the system services are carried out.

C.2 Description of System Services

Table 30 shows the description of the mission services.

Table 30. Interface of the mission services

Mission Services							
Name	Input Information		Behaviour			Output Information	
	Messages	Data	Pre-condition	Action	Post-condition	Data	Messages
Seabed Survey	Seabed Survey Request	Seabed Survery area	Seabed Survery commanded by end user	Perform operations in order to carry the Seabed Survery out	Seabed Survey accomplished	Seabed Survey Status	Seabed Survey Reponse
Target Selection	Target Selection Request	Seabed data collected	Target Selection commanded by end user	Perform operations in order to carry the Target Selection out	Target Selection accomplished, view and object selected	Target Selection Status, View, Object	Target Selection Reponse
Object Manipulation	Object Manipulation Request	View, Object of interest, Manipulation strategy	Object Manipulation commanded by end user	Perform operations in order to carry the Object Manipulation out	Object Manipulation accomplished	Object Manipulation Status	Object Manipulation Reponse

Table 31 shows the description of the operation services.

Table 31. Interface of the operation services

Operation Services							
Name	Input Information		Behaviour			Output Information	
	Messages	Data	Pre-condition	Action	Post-condition	Data	Messages
OCU Ethernet Data Transfer	Ethernet Data Transfer Request	Transfer Direction (Receiving, Sending), Data (if sending)	Ethernet link established	Transfer data from or to the OCU (according to transfer direction)	Data transferred	Data (if receiving), Transfer Status	Ethernet Data Transfer Response
Seabed Data Processing	Processing Request	Seabed Data	None	Process seabed data	Seabed Data processed	Seabed Mosaic, View & Object Characteriaztion, Grasp Specification, Processing Status	Procesing Response
ASC Positioning	ASC Positioning Request	Position Waypoint	ASC deployed	Position ASC according to the position waypoint given	ASC positioned	Positioning Status	ASC Positioning Response
ASC Path Following	ASC Path Following Resquest	List of waypoints	ASC positioned	Follows the path given	ASC ended path folliwng	Path Following Status	ASC Path Following Response
Dynamic Positioning	Dynamic Positioning Request	Position Waypoint	ASC deployed	Position dynamically ASC according to the position waypoint given	ASC dynamically positioned	Positioning Status	Dynamic Positioning Response
IAUV Positioning	IAUV Positioning Request	Position Waypoint	IAUV deployed	Position IAUV according to the position waypoint given	IAUV positioned	Positioning Status	IAUV Positioning Response
Terrain Following	Terrain Following Request	List of waypoints	IAUV positioned	Follows the terrain	IAUV ended terrain following	Terrain Following Status	Terrain Following Response
Pattern Search	Pattern Search Request	List of waypoints, View, Object	IAUV positioned	Search for the pattern given (obejct of interest)	IAUV ended pattern search	Pattern Search Status	Pattern Search Response
IAUV Homing	IAUV Homing Request	Vehicle current pose	IAUV is ready for docking	Dock IAUV with ASC	IAUV docked	Homing Status	IAUV Homing Response
IAUV Ethernet Data Transfer	Ethernet Data Transfer Request	Transfer Direction (Receiving, Sending), Data (if sending)	Ethernet link established	Transfer data from or to the IAUV (according to transfer direction)	Data transferred	Data (if receiving), Transfer Status	Ethernet Data Transfer Response
Intervention Configuration	Intervention Configuration Request	Intervention Configuration Data	Configuration data received by the IAUV	Configure intervention manoeuvre	Intervention strategies configured	Configuration Status	Intervention Configuration Response
Object Intervention Manoeuvre	Object Intervention Manoeuvre Request	Scene, Object	Object found	Intervene the Object of interest	Intervention manoeuvre ended	Manoeuvre Status	Object Intervention Manoeuvre Response
Floating Manimpulation	Floating Manipulation Request	Scene, Object	Object found	Intervene the Object of interest	Floating manipulation ended	Floating Manipulation Status	Floating Manipulation Response

Table 32 shows the description of the task services.

Table 32. Interface of the task services

Task Services							
Name	Input Information		Behaviour			Output Information	
	Messages	Data	Pre-condition	Action	Post-condition	Data	Messages
Seabed Image Mosaicing	Mosaicing Request	Seabed data collected	Seabed scanned	Generate seabed mosaic	Mosaic created	Mosaic, Mosaicing Status	Mosaicing Response
Object Characterization	Object Charaterization Request	Mosaic sub-section (input from user)	Mosaic created, View characterized	Specify object of interest	Object characterized	Object descriptor, Status (Object could be characterized / not)	Object Charaterization Response
View Characterization	View Charaterization Request	Mosaic sub-section (input from user)	Mosaic created	Specify view of interest	View characterized	View descriptor, Status (View could be characterized / not)	View Charaterization Response
Grasp Specification	Grasp Specification Request	Mosaic sub-section (input from user)	Mosaic created, Object characterized	Specify grasp for the object of interest	Grasp specified	Specification Status	Grasp Specification Response
OCU Ethernet Data Receiving	Ethernet Data Receiving Request	None	Data availble in the Ethernet port	Receive data from an Ethernet port	Data received from Ethernet port	Data received, Receiving Status	Ethernet Data Receiving Response
OCU Ethernet Data Sending	Ethernet Data Sending Request	Data to be sent	Ethernet communication is working	Send data through an Ethernet port	Data sent through Ethernet port	Sending Status	Ethernet Data Sending Response
Behaviour Management	Behaviour Management Request	Command (Start, Stop), List of waypoiints	No wrench effort is been commanded	Command ASC motion	Wrench effort commanded	Behaviour Management Status	Behaviour Management Response
Waypoint List Setting	Waypoint List Setting Request	List of waypoints	No waypoint list	Move ASC to according to a list of waypoint	Waypoints specified reached by the ASC	Waypoint List Setting Status	Waypoint List Setting Response
ASC Navigation Data Sending	None	None	Data from the ASC pose sensors is available	Send ASC navigation data	ASC navigation data sent out	Navigation state, Measurement quality	Navigation Data Notification
ASC Modem Data Receiving	Modem Data Receiving Resquest	None	Modem link established	Receive data	Data received from modemt, Status	Data, Transfer Receiving Status	Modem Data Receiving Response
ASC Modem Data Sending	Modem Data Sending Resquest	Data to be sent	Modem link established	Send data	Data sent through modem	Sending Status	Modem Data Sending Response
ASC Ethernet Data Receiving	Ethernet Data Receiving Request	None	Data availble in the Ethernet port	Receive data from an Ethernet port	Data received from Ethernet port	Data received, Receiving Status	Ethernet Data Receiving Response
ASC Ethernet Data Sending	Ethernet Data Sending Request	Data to be sent	Ethernet communication is working	Send data through an Ethernet port	Data sent through Ethernet port	Sending Status	Ethernet Data Sending Response
IAUV Navigation Data Sending	None	None	Data from the IAUV pose sensors is available	Send IAUV navigation data	IAUV navigation data sent out	Navigation state, Measurement quality	Navigation Data Notification
Path Planning Setting	Path Planning Request	List of waypoints (start point, end point), Altitude Mode	Path plan defined	Set path plan	Path plan is set up	Path Planning Status	Path Planning Response
Obstacle Map Generation	Obstacle Map Generation Request	Map region (Start point, End point)	Obstacle map defined	Get obstacle map	Obstacle map provided	Obstacle map, GOM status	Obstacle Map Generation Response
Map Sending	None	None	Data from the IAUV mapping sensors is available	Send map	Map sent	Map, Map Sending status	Map Sending notification

Name	Input Information		Behaviour			Output Information	
	Messages	Data	Pre-condition	Action	Post-condition	Data	Messages
Motion Control	Motion Control Resquest	List of waypoints	None	Control the IAUV motion	IAUV moves to the waypoints given	Motion Control Status	Motion Control Response
Station Keeping	Station Keeping Request	Position	IAUV reaches the object of interest	Keep the IAUV at a given position	IAUV keeps the position given	Station Kepping Status	Station Keeping Response
IAUV Modem Data Receiving	Modem Data Receiving Resquest	None	Modem link established	Receive data	Data received from modemt, Status	Data, Transfer Receiving Status	Modem Data Receiving Response
IAUV Modem Data Sending	Modem Data Sending Resquest	Data to be sent	Modem link established	Send data	Data sent through modem	Sending Status	Modem Data Sending Response
Seabed Data Collection	Seabed Data Collection Request	None	IAUV is in the start position	Collect seabed data	seabed data collected, Images pre-processed	Seabed Data Collection Status	Seabed Data Collection Response
Motion Estimation	Motion Estimation Request	Motion parameters	IAUV is moving	Estimate motion parameters	Motion parameters estimated	Motion Estimation Status	Motion Estimation Response
Navigation Data Update	Navigation Data Update Request	Vehicle current pose, Pose mesuarement quality	None	Update navigation data	Vehicle current pose updated	None	None
Vehicle Docking	Vehicle Docking Request	Vehicle current pose, Pose measurement quality	Intervention has finished, Docking depth attained	Dock IAUV	IAUV docked	Vehicle Docking Status	Vehicle Docking Response
Object Identification	Object Identification Request	Current frame, Object descriptor	Object characterized, View identified	Identify object of interest	Object identified	Object Identification status (Object identified / not identified)	Object Identification Response
Scene Identification	Scene Identification Request	Current frame, View descriptor	View characterized	Identify scene of interest	Scene identified	Scene Identification Status (Scene identified / not identified)	Scene Identification Response
Intervention Configuration Setting	Intervention Configuration Request	Configuration parameters	Manipulator available and working	Set intervention configuration	Excecution of end actions	Intervention Configuration Status	Intervention Configuration Response
Object Intervention Control	Object Intervention Request	Object characterization, Task Specification	Object Found	Control object intervention	Excecution of end actions	Object Intervention Status	Object Intervention Response

Table 33 shows the description of the action services.

Table 33. Interface of the action services

Action Services							
Name	Input Information		Behaviour			Output Information	
	Messages	Data	Pre-condition	Action	Post-condition	Data	Messages
Vehicle Motion Control	Motion Control request	Commanded wrench effort	Thrusters are working well	Move vehicle to the waypoint given	Vehicle at the specified waypoint	Pose, Motion Control Status	Motion Control response
End Effector Control	End Effector Control request	Commanded end effector effort	End effectors are working well	Move hand to the pose given	Hand posed as desired	End Effector Status	End Effector Control response
Joint Effector Control	Joint Effector Control request	Commanded joint effector effort	Joint effectors are working well	Move arm to the pose given	Arm posed as desired	Joint Effector Status	Joint Effector Control response

C.3 Orchestration and choreography

Figure 62 shows the execution order (orchestration) of the operation services that build the mission service Seabed Survey.

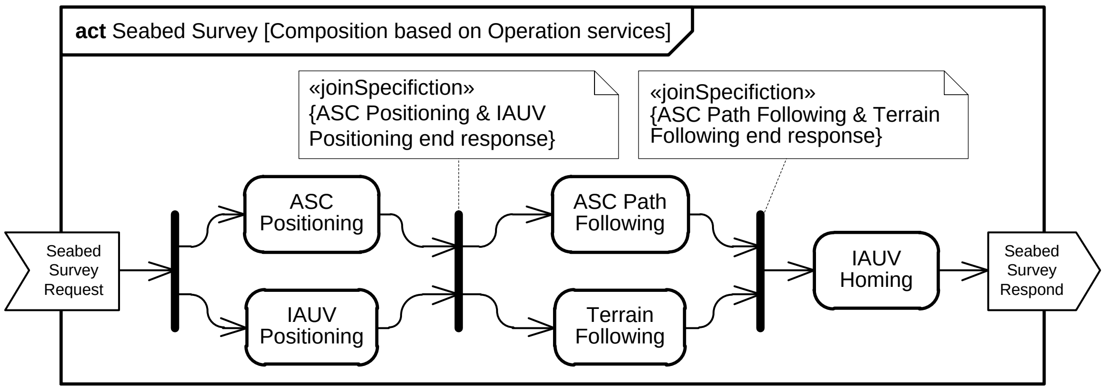

act Seabed Survey [Composition based on Operation services]
«joinSpecifiction» {ASC Positioning & IAUV Positioning end response}
«joinSpecifiction» {ASC Path Following & Terrain Following end response}
Seabed Survey Request
ASC Positioning
IAUV Positioning
ASC Path Following
Terrain Following
IAUV Homing
Seabed Survey Respond

Figure 62. Seabed Survey service composed by operation services

Figure 63 shows the execution order (orchestration) of the operation services that build the mission service Target Selection.

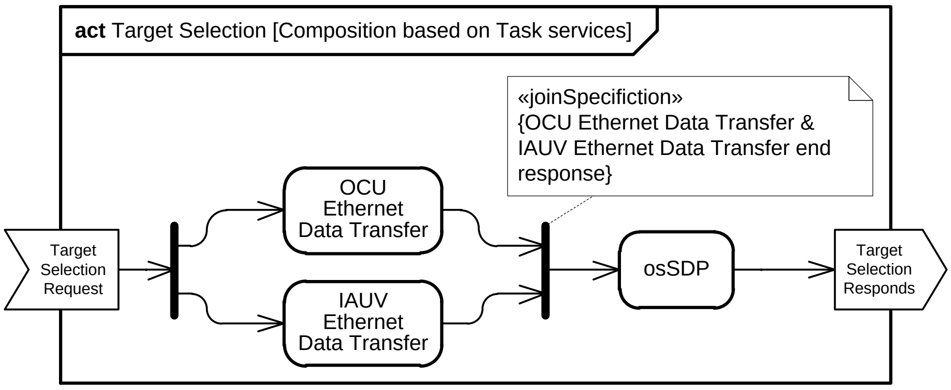

act Target Selection [Composition based on Task services]
«joinSpecifiction»
{OCU Ethernet Data Transfer &
IAUV Ethernet Data Transfer end
response}
Target Selection Request
OCU Ethernet Data Transfer
IAUV Ethernet Data Transfer
osSDP
Target Selection Responds

Figure 63. Composite mission service: Target Selection

Figure 64 shows the execution order (orchestration) of the operation services that build the mission service Object Manipulation.

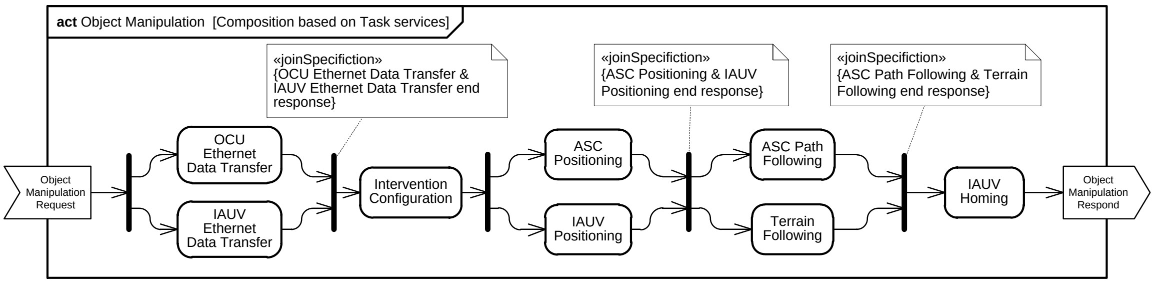

act Object Manipulation [Composition based on Task services]
«joinSpecifiction» {OCU Ethernet Data Transfer & IAUV Ethernet Data Transfer end response}
«joinSpecifiction» {ASC Positioning & IAUV Positioning end response}
«joinSpecifiction» {ASC Path Following & Terrain Following end response}
Object Manipulation Request
OCU Ethernet Data Transfer
IAUV Ethernet Data Transfer
Intervention Configuration
ASC Positioning
IAUV Positioning
ASC Path Following
Terrain Following
IAUV Homing
Object Manipulation Respond

Figure 64. Composite mission service: Object Manipulation

Figure 65 shows the execution order (orchestration) of the task services that build the operation service ASC Positioning.

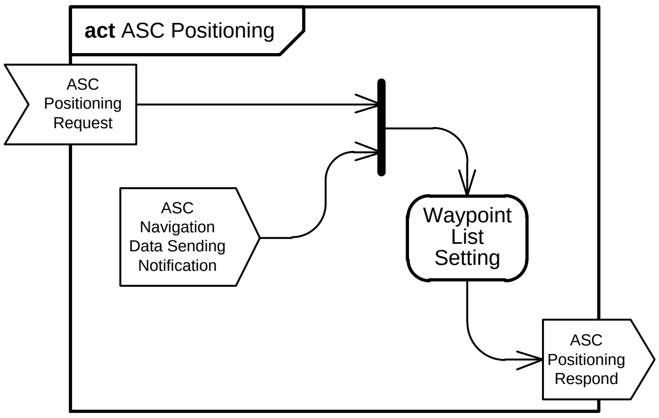


Figure 65. Composite operation service: ASC Positioning

Figure 66 shows the execution order (orchestration) of the task services that build the operation service IAUV Positioning.

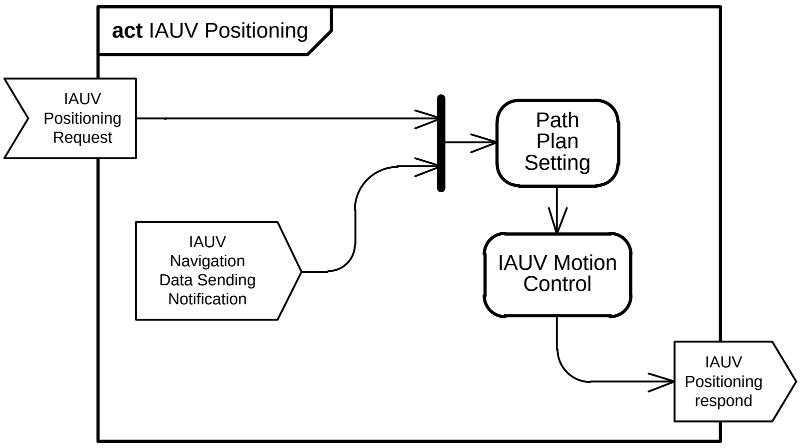


Figure 66. Composite operation service: IAUV Positioning

Figure 67 shows the execution order (orchestration) of the task services that build the operation service Leader Following.

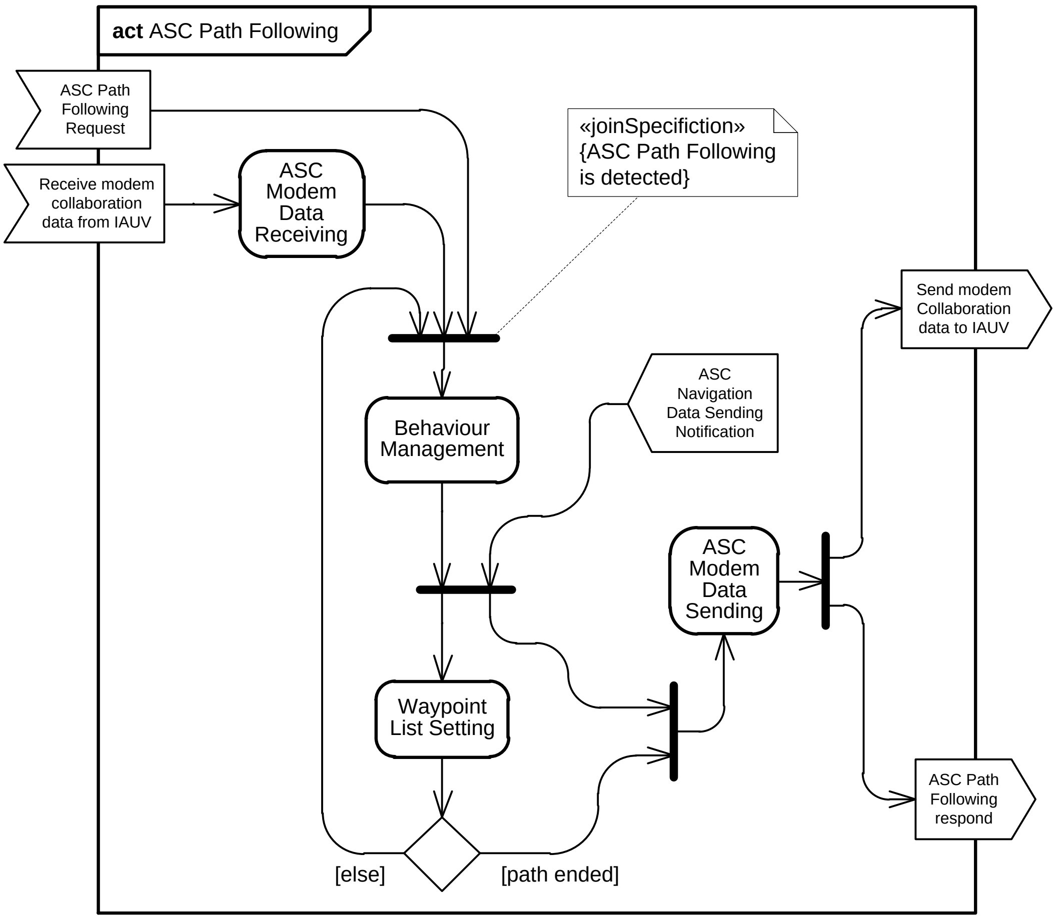

act ASC Path Following
ASC Path Following Request
Receive modem collaboration data from IAUV
ASC Modem Data Receiving
«joinSpecifiction» {ASC Path Following is detected}
Send modem Collaboration data to IAUV
Behaviour Management
ASC Navigation Data Sending Notification
ASC Modem Data Sending
Waypoint List Setting
ASC Path Following respond
[else]
[path ended]

Figure 67. Composite operation service: Leader Following

Figure 68 shows the execution order (orchestration) of the task services that build the operation service IAUV Terrain Following.

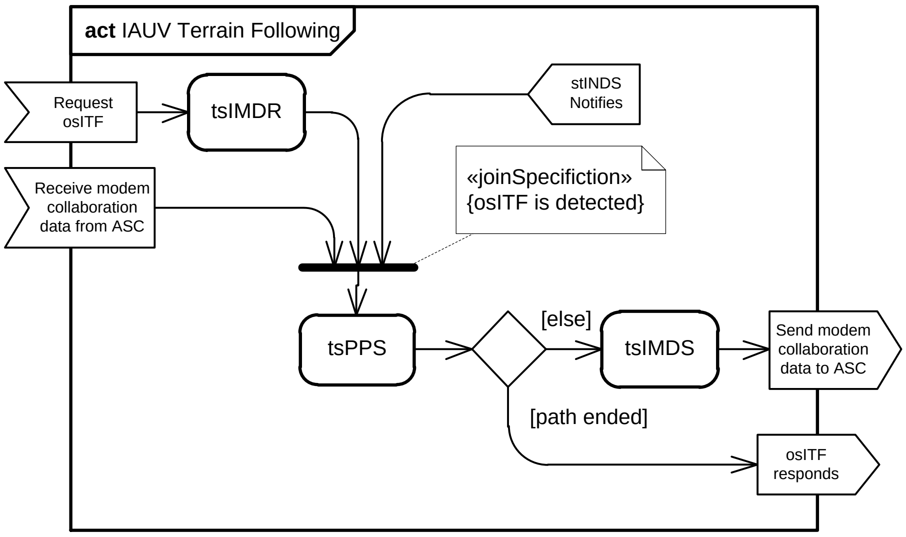

act IAUV Terrain Following
Request osITF
tsIMDR
stINDS Notifies
«joinSpecifiction» {osITF is detected}
Receive modem collaboration data from ASC
tsPPS
[else]
tsIMDS
Send modem collaboration data to ASC
[path ended]
osITF responds

Figure 68. Composite operation service: Terrain Following

Figure 69 shows the execution order (orchestration) of the task services that build the operation service IAUV Homing.

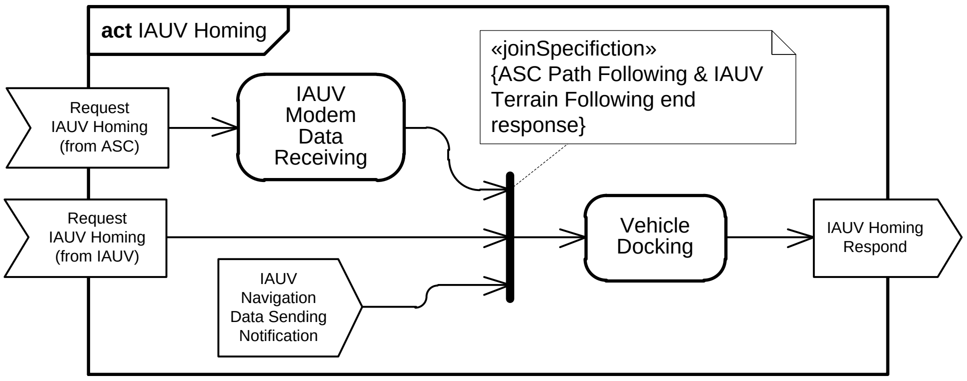

act IAUV Homing
«joinSpecifiction» {ASC Path Following & IAUV Terrain Following end response}
Request IAUV Homing (from ASC)
IAUV Modem Data Receiving
Request IAUV Homing (from IAUV)
Vehicle Docking
IAUV Homing Respond
IAUV Navigation Data Sending Notification

Figure 69. Composite operation service: IAUV Homing

Figure 70 shows the execution order (orchestration) of the task services that build the operation service Pattern Search.

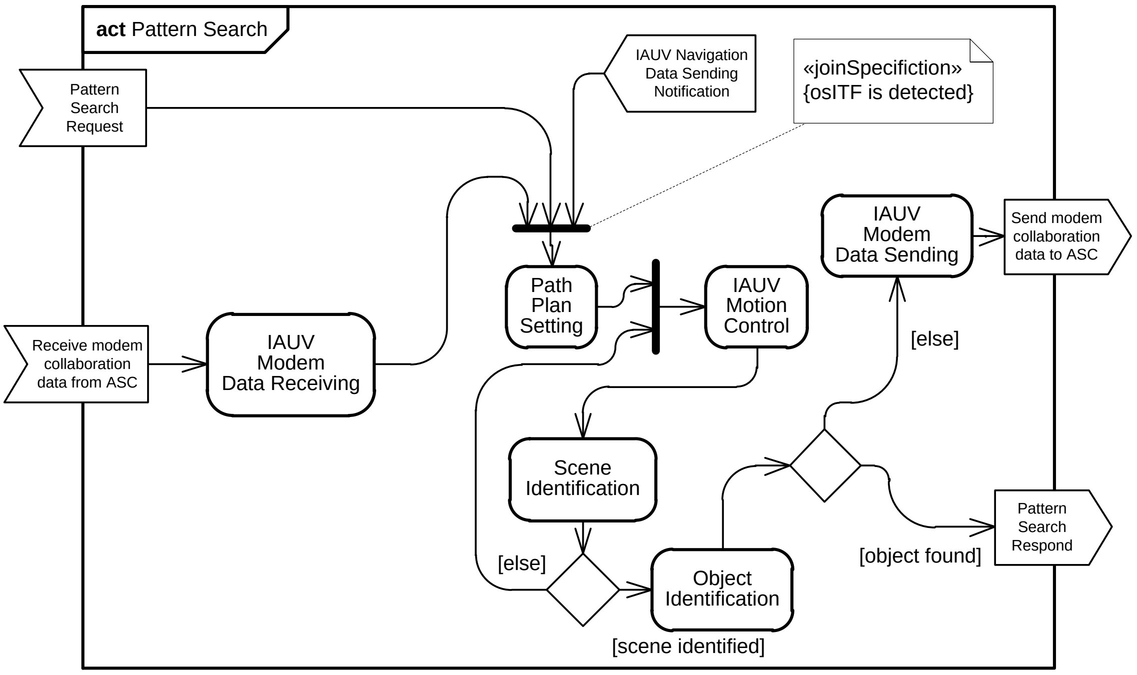


Figure 70. Composite operation service: Pattern Search

Figure 71 shows the execution order (orchestration) of the task services that build the operation service OCU Ethernet Data Transfer.

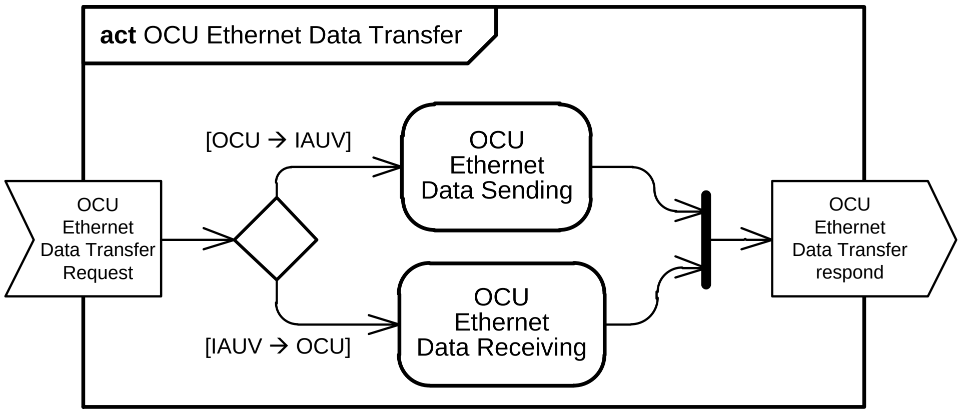

act OCU Ethernet Data Transfer
[OCU → IAUV]
OCU Ethernet Data Sending
OCU Ethernet Data Transfer Request
OCU Ethernet Data Transfer respond
[IAUV → OCU]
OCU Ethernet Data Receiving

Figure 71. Composite operation service: OCU Ethernet Data Transfer

Figure 72 shows the execution order (orchestration) of the task services that build the operation service IAUV Ethernet Data Transfer.

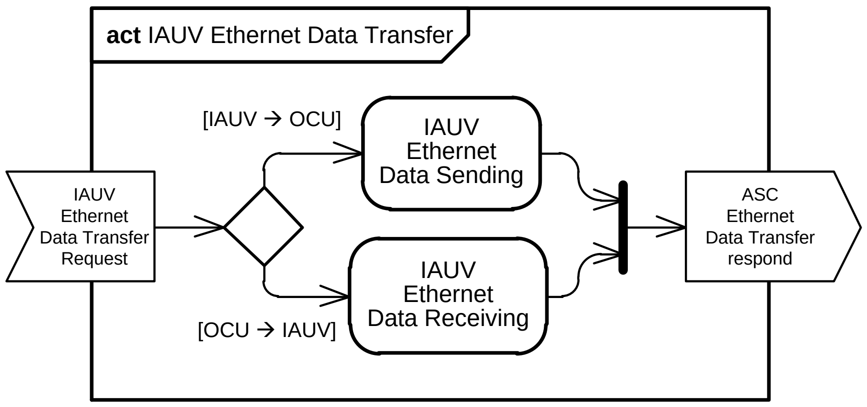

act IAUV Ethernet Data Transfer
IAUV Ethernet Data Transfer Request
[IAUV → OCU]
IAUV Ethernet Data Sending
[OCU → IAUV]
IAUV Ethernet Data Receiving
ASC Ethernet Data Transfer respond

Figure 72. Composite operation service: IAUV Ethernet Data Transfer

Figure 73 shows the execution order (orchestration) of the task services that build the operation service Seabed Data Processing.

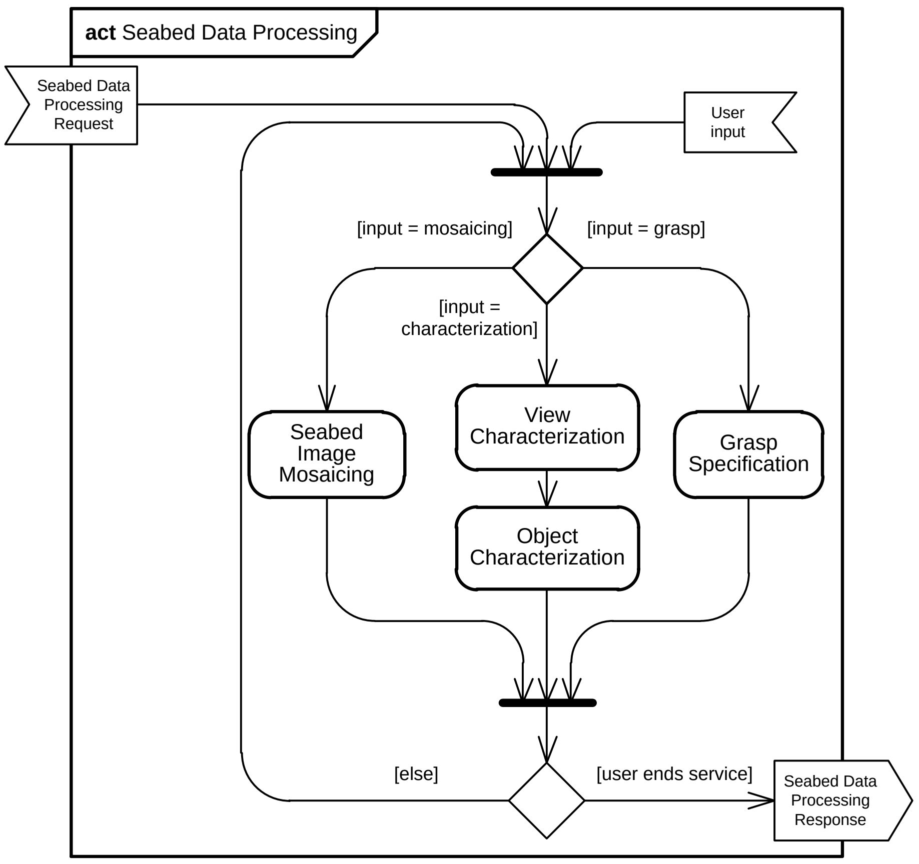


Figure 73. Composite operation service: Seabed Data Processing

Figure 74 shows the execution order (orchestration) of the task services that build the operation service Intervention Configuration.

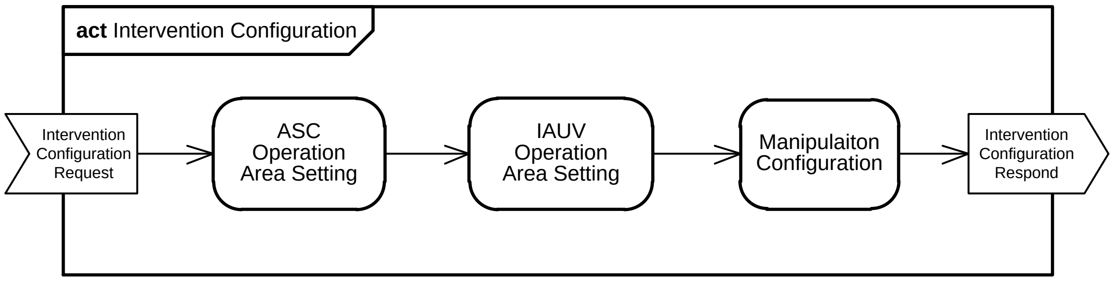

act Intervention Configuration
Intervention Configuration Request
ASC Operation Area Setting
IAUV Operation Area Setting
Manipulaiton Configuration
Intervention Configuration Respond

Figure 74. Composite operation service: Intervention Configuration

Figure 75 shows the execution order (orchestration) of the task services that build the operation service ASC Dynamic Positioning.

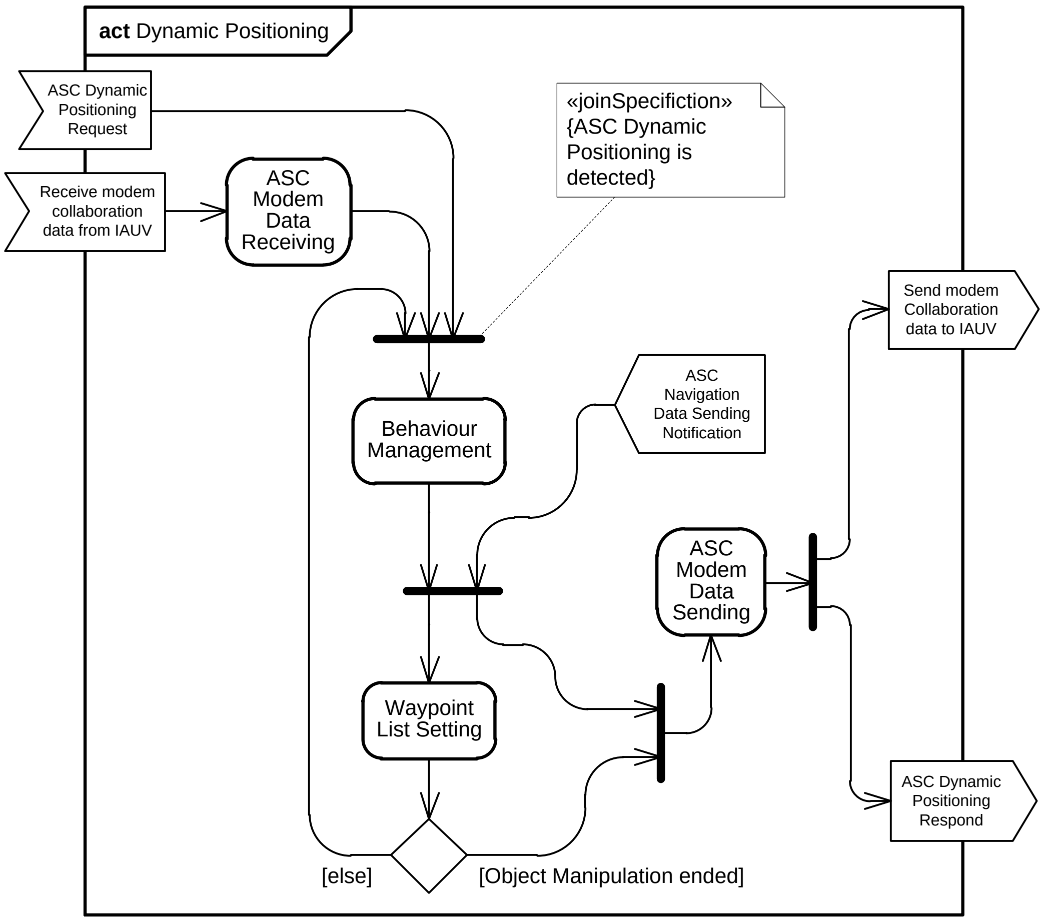

act Dynamic Positioning
ASC Dynamic Positioning Request
Receive modem collaboration data from IAUV
ASC Modem Data Receiving
«joinSpecifiction» {ASC Dynamic Positioning is detected}
Send modem Collaboration data to IAUV
ASC Navigation Data Sending Notification
Behaviour Management
ASC Modem Data Sending
Waypoint List Setting
ASC Dynamic Positioning Respond
[else]
[Object Manipulation ended]

Figure 75. Composite operation service: Dynamic Positioning

Figure 76 shows the execution order (orchestration) of the task services that build the operation service Object Intervention Manoeuvre.

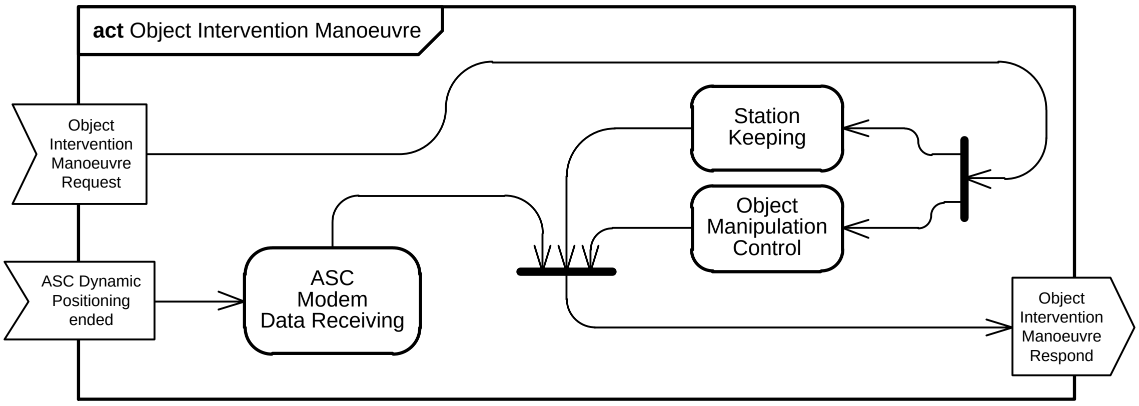

act Object Intervention Manoeuvre
Object Intervention Manoeuvre Request
ASC Dynamic Positioning ended
ASC Modem Data Receiving
Station Keeping
Object Manipulation Control
Object Intervention Manoeuvre Respond

Figure 76. Composite operation service: Object Intervention Manoeuvre

Figure 77 shows the execution order (orchestration) of the task services that build the operation service Floating Manipulation.

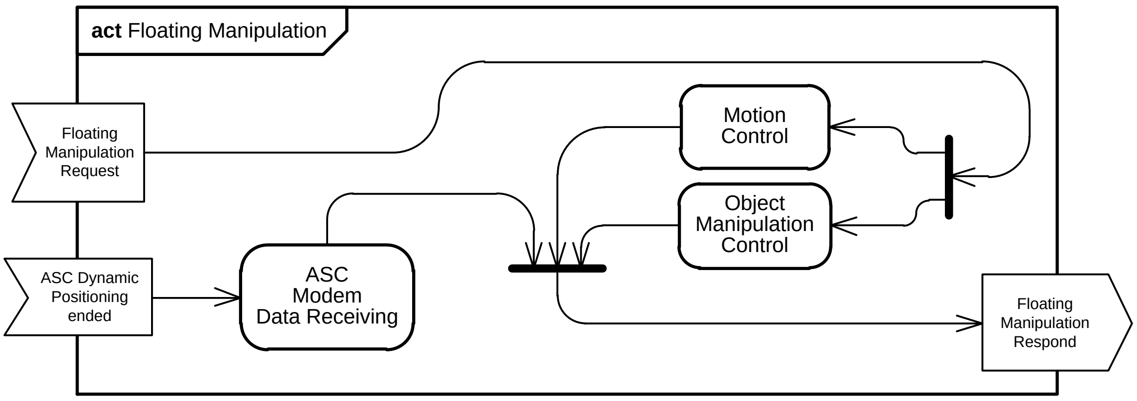

act Floating Manipulation
Floating Manipulation Request
ASC Dynamic Positioning ended
ASC Modem Data Receiving
Motion Control
Object Manipulation Control
Floating Manipulation Respond

Figure 77. Composite operation service: Floating Manipulation

Figure 78 shows the mission service Seabed Survey as composition (orchestration) of task services.

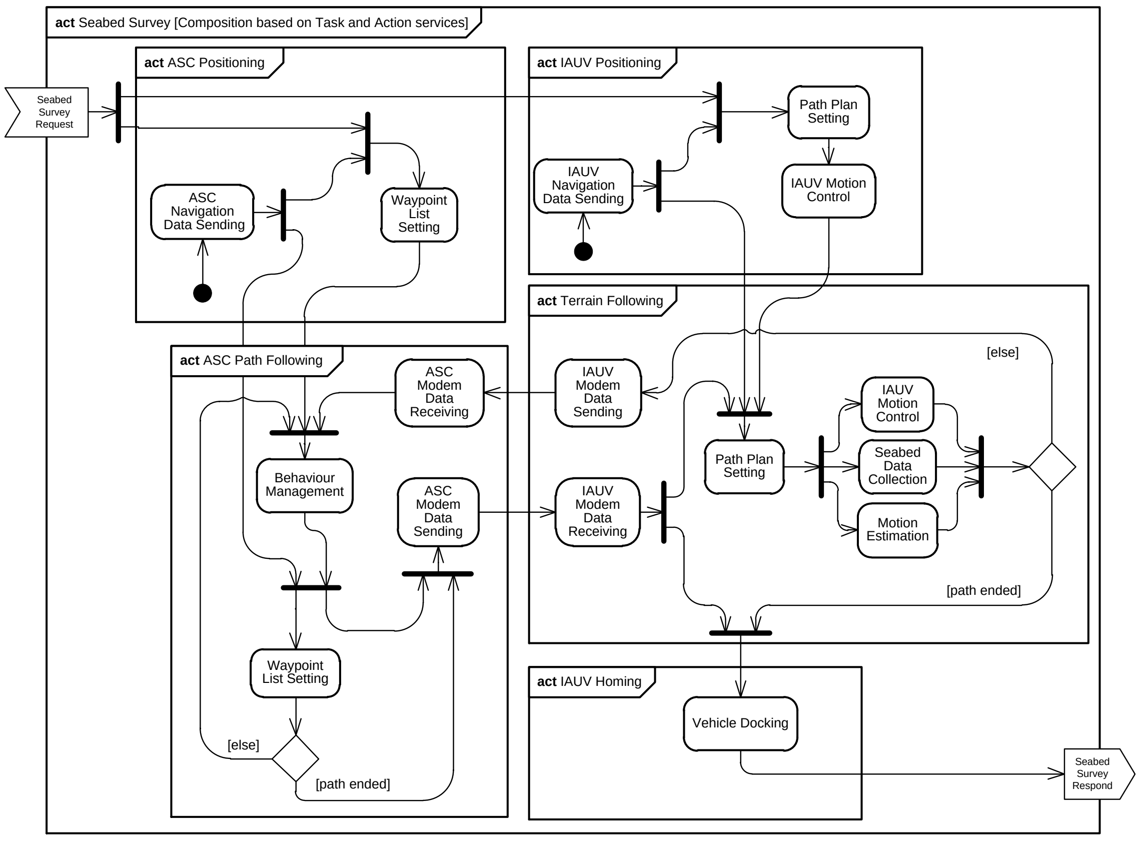


Figure 78. Seabed Survey service composed by task services

Figure 79 shows the mission service Target Selection as composition (orchestration) of task services.

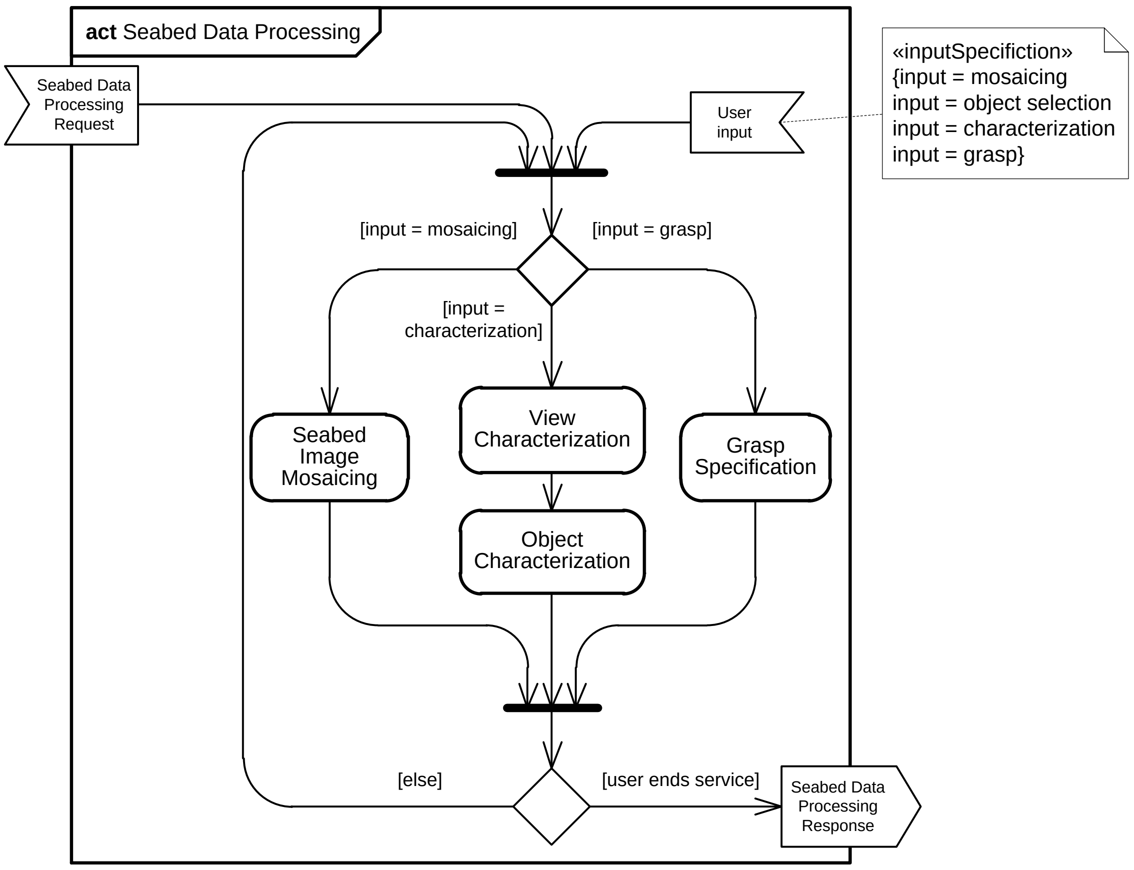


Figure 79. Target Selection service composed by task services

Figure 80 shows the mission service Object Manipulation as composition (orchestration) of task services when performing station keeping for manipulation.

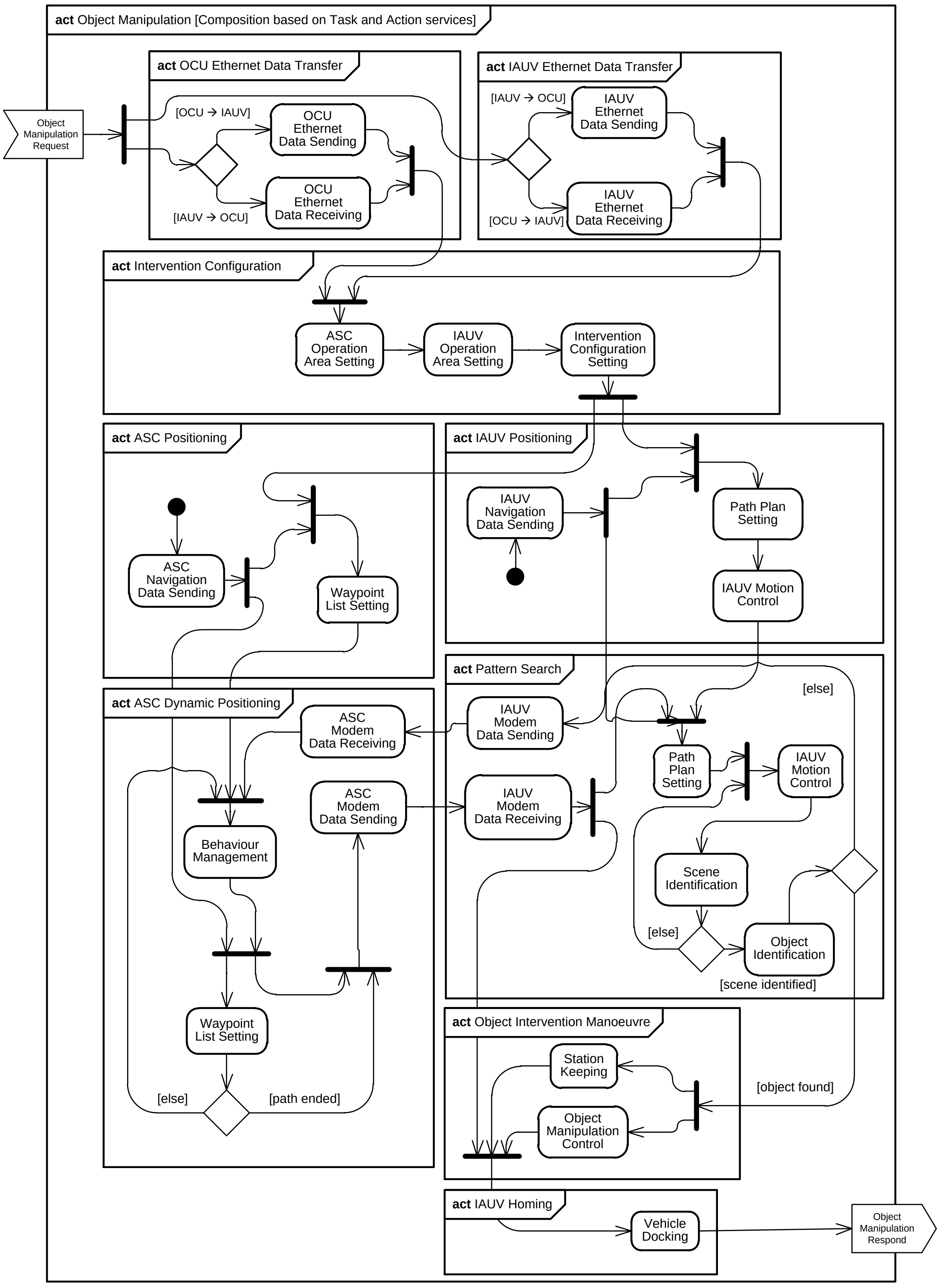

act Object Manipulation [Composition based on Task and Action services]
Object Manipulation Request
act OCU Ethernet Data Transfer
[OCU → IAUV]
OCU Ethernet Data Sending
OCU Ethernet Data Receiving
[IAUV → OCU]
act IAUV Ethernet Data Transfer
[IAUV → OCU]
IAUV Ethernet Data Sending
IAUV Ethernet Data Receiving
[OCU → IAUV]
act Intervention Configuration
ASC Operation Area Setting
IAUV Operation Area Setting
Intervention Configuration Setting
act ASC Positioning
ASC Navigation Data Sending
Waypoint List Setting
act IAUV Positioning
IAUV Navigation Data Sending
Path Plan Setting
IAUV Motion Control
act ASC Dynamic Positioning
ASC Modem Data Receiving
ASC Modem Data Sending
Behaviour Management
Waypoint List Setting
[else]
[path ended]
act Pattern Search
[else]
IAUV Modem Data Sending
IAUV Modem Data Receiving
Path Plan Setting
IAUV Motion Control
Scene Identification
[else]
Object Identification
[scene identified]
act Object Intervention Manoeuvre
Station Keeping
Object Manipulation Control
[object found]
act IAUV Homing
Vehicle Docking
Object Manipulation Respond

Figure 80. Object Manipulation service composed by task services (with station keeping for manipulation)

Figure 81 shows the mission service Object Manipulation as composition (orchestration) of task services when free floating manipulation is required.

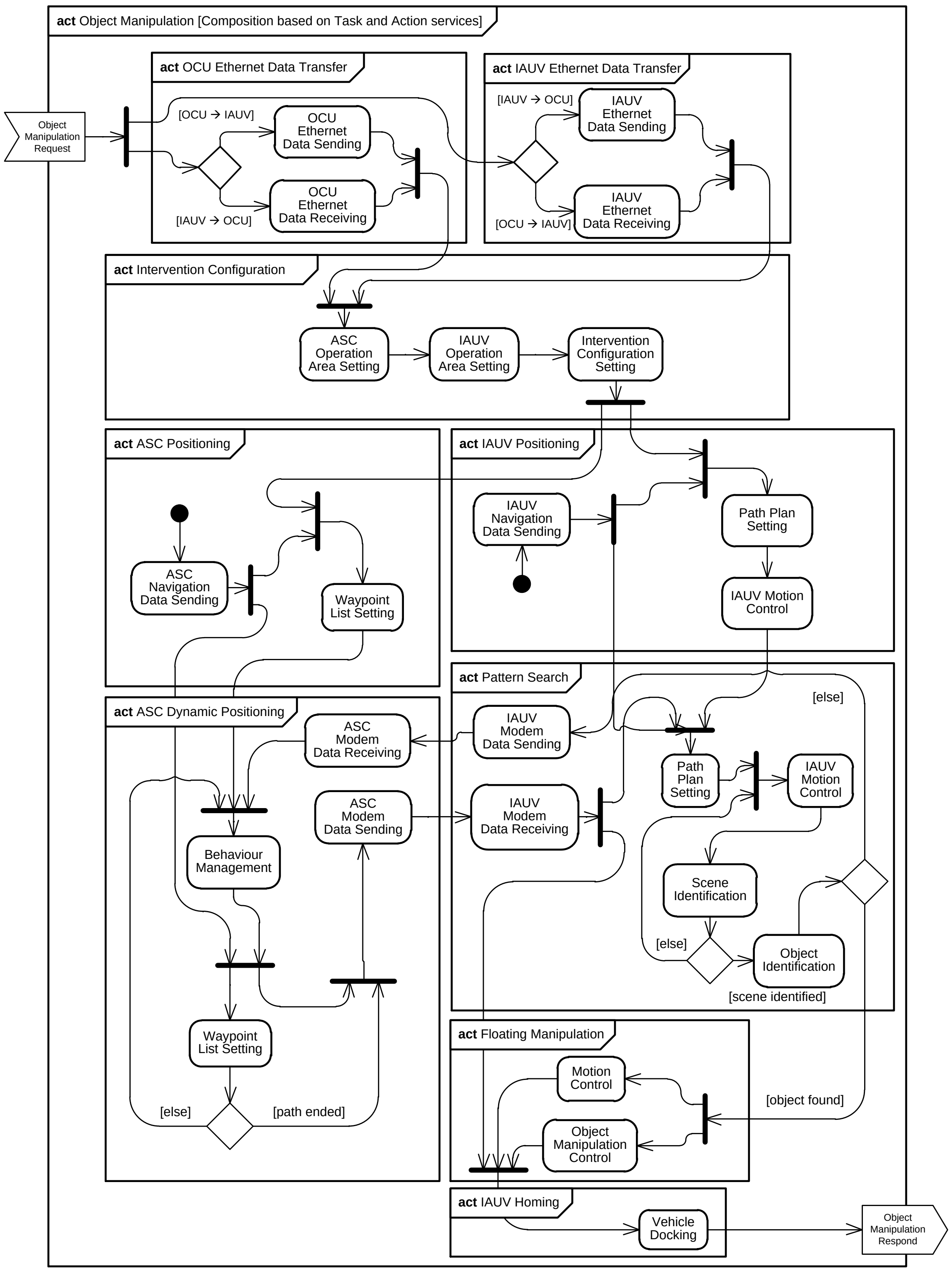

act Object Manipulation [Composition based on Task and Action services]
Object Manipulation Request
act OCU Ethernet Data Transfer
[OCU → IAUV]
OCU Ethernet Data Sending
OCU Ethernet Data Receiving
[IAUV → OCU]
act IAUV Ethernet Data Transfer
[IAUV → OCU]
IAUV Ethernet Data Sending
IAUV Ethernet Data Receiving
[OCU → IAUV]
act Intervention Configuration
ASC Operation Area Setting
IAUV Operation Area Setting
Intervention Configuration Setting
act ASC Positioning
ASC Navigation Data Sending
Waypoint List Setting
act IAUV Positioning
IAUV Navigation Data Sending
Path Plan Setting
IAUV Motion Control
act ASC Dynamic Positioning
ASC Modem Data Receiving
ASC Modem Data Sending
Behaviour Management
Waypoint List Setting
[else]
[path ended]
act Pattern Search
[else]
IAUV Modem Data Sending
IAUV Modem Data Receiving
Path Plan Setting
IAUV Motion Control
Scene Identification
[else]
Object Identification
[scene identified]
act Floating Manipulation
Motion Control
Object Manipulation Control
[object found]
act IAUV Homing
Vehicle Docking
Object Manipulation Respond

Figure 81. Object Manipulation service composed by task services (with free floating manipulation)

Appendix D: Coordinate Adjustments

D.1 Coordinate Frames

As ROS was initially created for ground-based robots, in its standard messages it uses a different right handed coordinate frame to that typically used on AUVs. In the ROS inertial coordinate frame, X is forward, Y is left, and Z is up. With AUVs, typically X is forward, Y is right, and Z is down, as with aerial vehicles. However, the difference is just a simple 180 degree rotation about the X axis, equivalent to a sign inversion on the Y and Z axes (position and orientation values). Altitude from vehicle to seabed is considered separate to the coordinate frames, as it is a local measurement and cannot be transformed in the same way as depth.

To minimise code change involved in moving everything to a new set of coordinate frames, retain the more AUV-standard coordinate systems defined below for vehicle navigation purposes is proposed. However, the use of the ROS coordinate system when it comes to manipulator control and related transformations is proposed, as this allows us to benefit from the ROS transform library (tf). The ROS conventions described here should be consistent with [30].

D.1.1 Global

latitude	decimal degrees, +90 to north of equator, -90 to south of equator
longitude	decimal degrees, +180 to east of Prime Meridian, -180 to west of Prime Meridian
gps_altitude	optional, altitude above sea level in metres, NOT above sea bed

Note that for sufficient local precision with latitude and longitude, a 64 bit floating point representation is needed.

D.1.2 ROS Standard

This is a right handed coordinate system. Relative fixed axis rotations are used, in radians. Within tf transforms in ROS, rotations are actually stored in quaternion form (x, y, z, w), but library calls are provided to map to and from this representation.

X / forward	metres, positive forward
Y / left	metres, positive to left of X axis, perpendicular to X axis
Z / up	metres, positive upwards, perpendicular to X-Y plane

X rotation / roll	+/- π/2 radians, clockwise rotation about the X axis, looking toward +ve X
Y rotation / pitch	+/- π/2 radians, clockwise rotation about the Y axis, looking toward +ve Y
Z rotation / yaw	+/- π radians, clockwise rotation about the Z axis, looking toward +ve Z

D.1.3 Camera

As with other sensors, cameras should just use the ROS standard frame. X is away from the camera (forward), projecting directly out the lens, Y is left of the camera, and Z is up.

D.1.4 Image

This coordinate frame is for images, and is also known as the camera optical frame. Not to be confused with the standard frame used for cameras.

X	positive to the right in the image, zero at the optical centre of the image
Y	positive down in the image, zero at the optical centre of the image
Z	positive into the image

D.1.5 AUV World

For the world/map frame of the AUV and USV, the "Earth Axes" coordinate system also commonly used in planes is used. This is a right handed coordinate system, and simply equivalent to the ROS World frame rotated 180 degrees about the x axis. Relative fixed axis rotations are used, in radians.

X / forward	metres, positive forward, zero at map origin
Y / right	metres, positive in the direction of east, zero at map origin
Z / down	metres, positive towards centre of the earth, zero at sea level

roll	+/- π/2 radians, positive clockwise rotation about the X axis looking towards positive X, relative to the horizon ('right wing down')
pitch	+/- π/2 radians, positive upwards rotation of the vehicle nose about the Y axis, relative to the horizon
yaw	+/- π radians, positive clockwise rotation about the Z axis looking towards the centre of the earth, from north to south heading

D.1.6 AUV Body

For the body or inertial frame of the AUV and USV, the "Inertial Axes" coordinate system also commonly used in planes is used. This is a right handed coordinate system, and simply equivalent to the ROS Body frame rotated 180 degrees about the x axis. Relative fixed axis rotations are used, in radians.

X/ surge	metres, positive forward, through nose of vehicle
Y / sway	metres, positive to right/starboard of X axis, perpendicular to X axis
Z / heave	metres, positive downwards, perpendicular to X-Y plane

X rotation / roll	+/- π/2 radians, clockwise rotation about the X axis, looking toward +ve X
Y rotation / pitch	+/- π/2 radians, clockwise rotation about the Y axis, looking toward +ve Y
Z rotation / yaw	+/- π radians, clockwise rotation about the Z axis, looking toward +ve Z

D.2 Coordinate Transforms

The ROS tf library [29] supports coordinate frame communication and manipulation. It uses standard messages (TransformStamped) for publishing transform data to a distributed system. Listeners listen to transform messages and cache all data up to a limit. Publishers publish transforms between coordinate frames. There is no central source of transform information. The tf library includes code for manipulating and filtering this transform data. With tf transforms, the standard ROS coordinate system (forward, left, up) is always used.

Frames have Graph Resource Names, and ROS convention [30] suggests a set of three tf frames to form the basis of a vehicle's frame system: map, odom, and base_link

- map is anchored to a point in the real world, e.g. a latitude and longitude origin.
- odom (short for odometry) may make discrete jumps relative to the map frame, as the vehicle is localised, e.g. with an acoustic Long Base Line (LBL) or other absolute positioning system.
- base_link is rigidly attached to the robot itself and moves smoothly relative to the odom frame.

The frame hierarchy would be: map→odom→base_link.

The transform typically used for local sensing and acting would be odom→base_link, as it changes in a continuous fashion. For a long term global reference, map→base_link would be used, as it should not drift considerably over time, but may jump in discrete steps. On an AUV, a compass and DVL could be used to update the odom→base_link transform. The transform map→odom represents the error in the odometry calculations relative to absolute map position. For example, a localisation component using an LBL sensor might compute the absolute transform map→base_link (vehicle pose from map origin), and then use the latest odom→base_link transform to calculate and publish the transform map→ odom.

Figure 82 shows an example tree of coordinate frames for an AUV and an ASC, and the messages used to exchange information about these frames. Except for the common map frame, here the use of a prefix of /auv1 or /asc1 is applied to indicate which vehicle the coordinate frame relates to. The prefix could be automatically applied to the modules within each vehicle using a ROS namespace. In that case, within the vehicle software, the /map frame would be the only frame to be specified absolutely, with a leading slash.

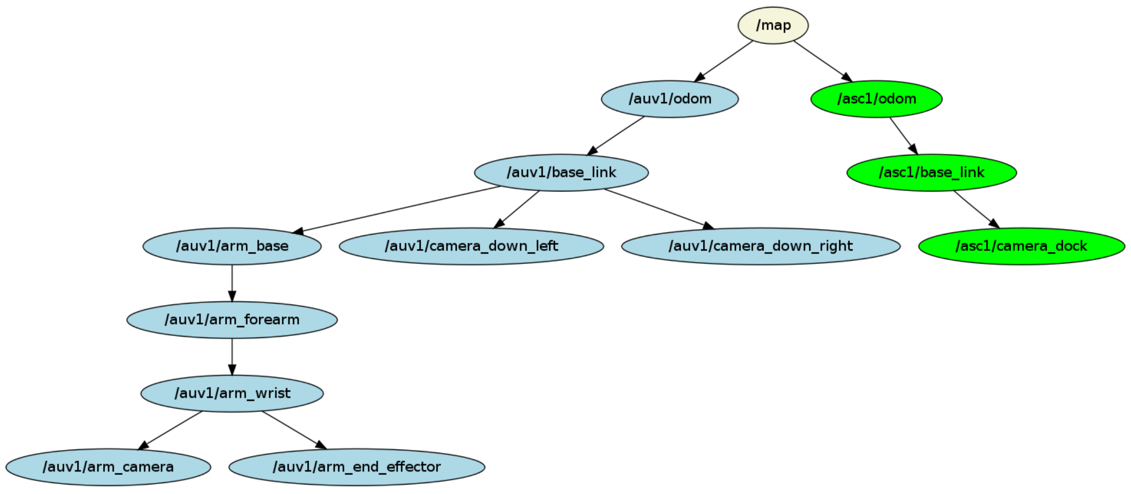


Figure 82. coordinate frames for an AUV and a ASC

As an example of relating sensor data to transforms, images produced by the down right camera would reference frame '/auv1/camera_down_right'. The idea being that sensor data is published in the frame in which it is observed. The transform for a fixed camera, such as /auv1_base_link→/auv1/camera_down_left would just contain the same transform values each time, whereas the transform /auv1/base_link→/auv1/arm_camera would change when the arm is moved. The standard ROS node static_transform_publisher can be used for publishing unchanging transforms at a set rate.

Appendix E: Mission Configuration

E.1 XML file for mission configuration

```xml
<!--****************************************************************************
* Module: Missions.xml                                                         *
*                                                                              *
* Description: Service-based mission plan for execution of agent activities of *
*              marine vehicles.                                                *
*                                                                              *
* Comments: This XML file is the input for the Mission Planner.                *
* ---------------------------------------------------------------------------- *
* Version  Author Date       Reason                                            *
* ---------------------------------------------------------------------------- *
* SM100R00 CCI    30/05/2012 First mission plan that works with MP200R00.      *
* SM101R00 CCI    24/08/2012 Multiple and different missions supported.        *
*****************************************************************************-->

<?xml version="1.0" encoding="UTF-8"?>
<template version="SM101R00"/>

<missions>
    <mission name="seabedSurvey">
        <parameter name="type" value="Visual"/> <!-- Visual or Acoustic (Bathymetry) -->
        <parameter name="area" xbValue="30" ybValue="30" xeValue="70" yeValue="70"/> <!-- Xbegin,Ybegin,Xend,Yend in metres-->
        <parameter name="depth" value="5"/> <!-- depth in metres -->
        <parameter name="timeout" value="300"/> <!-- timeout in seconds -->
    </mission>
    <mission name="targetIntervention">
        <parameter name="mode" value="freeFloating"/> <!-- freeFloating or stationKeeping -->
        <parameter name="specificationID" value="1"/>
        <parameter name="type" value="Visual"/> <!-- Visual or Acoustic (Bathymetry) -->
        <parameter name="area" xbValue="40" ybValue="40" xeValue="45" yeValue="45"/> <!-- Xbegin,Ybegin,Xend,Yend in metres-->
        <parameter name="depth" value="5"/> <!-- depth in metres -->
        <parameter name="timeout" value="100"/> <!-- timeout in seconds -->
    </mission>
</missions>

<!--****************************************************************************
********************************************************************************
********************************************************************************
***** Work Package: [3] Vehicles Intelligent Control Architecture         *****
*****                                                                      *****
***** Project: Trident                                                     *****
*****                                                                      *****
***** WP Leader: Ocean Systems Laboratory                                  *****
*****            Heriot-Watt University                                    *****
********************************************************************************
********************************************************************************
*****************************************************************************-->
```

Bibliography

[1] Trident project, available in http://www.irs.uji.es/trident/index.html.

[2] S. Friedenthal, A. Moore, R. Steiner, A Practical Guide to SysML, 2nd Ed., Morgan Kaufmann, 2011.

[3] L. Molnar, and S. M. Veres, "System Verification of Autonomous Underwater Vehicles by Model Checking", in Proc. OCEANS, Bremen, Germany, 2008.

[4] Z. Yan, and S. Hou, "A Coordinated Method Based on Hybrid Intelligent Control Agent for Multi-AUVs Control", in Proc. Int. Conf. on Information Acquisition, Weihai, Shandong, China, 2006.

[5] J. Borges de Sousa, and F. Lobo Pereira, "Real-time Hybrid Control of Multiple Autonomous Underwater Vehicles", in Proc. 37th IEEE Conf. on Decision and Control, Tampa, Florida, USA, 1998.

[6] T. Y. Teck, M. Chitre, and P. Vadakkepat, "Hierarchical Agent-based Command and Control System for Autonomous Underwater Vehicles", in Proc. Int. Conf. on Autonomous and Intelligent Systems, Singapore, 2010.

[7] J. P. Pignon, and C. Bizingre, "Multiple Agents Architecture for Intelligent Command and Control System of AUVs : Application to the MARIUS Vehicle", in Proc. OCEANS, vol. 3, pp. III/126 – II/131, 1994.

[8] T. R. Cuff, and R. W. Wall, "Support Platform and Communications to Manage Cooperative AUV Operations", in Proc. OCEANS Asia Pacific, 2007.

[9] P. Patrón, E. Miguelañez, J. Cartwright, and Y. Petillot, "Semantic Knowledge-based Representation for improving situation awareness in service oriented agents of autonomous underwater vehicles", Proc. OCEANS Europe, 2009.

[10] C. C. Sotzing, J. Evans, and D. M. Lane, "A Multi-Agent Architecture to Increase Coordination Efficiency in Multi-AUV Operations", in Proc. OCEANS Europe, pp. 1-6, 2007.

[11] P. Szymak, and T. Praczyk, "Control Systems of Underwater Vehicles in Multi-agent System of Underwater Inspection", in Proc. 11th WSEAS Int. Conf. on Automatic Control, Modelling and Simulation, Vouliagmeni, Athens, Greece, 2009.

[12] H. Li, A. Popa, and C. Thibault, “A Software Framework for Multi-Agent Control of Multiple Autonomous Underwater Vehicles for Underwater Mine Counter-Measures”, in Proc. Int. Conf. on Autonomous and Intelligent Systems, Povoa de Varzim, Portugal, 2010.

[13] T. Fujii, and F. Ura, “Autonomous Underwater Robots with Distributed Behavior Control Architecture”, in Proc. IEEE Int. Conf. on Robotics and Automation, Nagoya, Aichi, Japan, 1995.

[14] X. Q. Bian, T. Chen, J. Zhou, and Z. Yan, “Research of Autonomous Control Based on Multi-Agent for AUV”, in Proc. Int. Workshop on Intelligent Systems and Applications, WuHan, China, 2009.

[15] P. Ray, M. O’Rourke, and D. Edwards, “The Ontological Status of Autonomous Underwater Vehicle Fleets”, in Proc. OCEANS MTS/IEEE Biloxi, Biloxi, Mississippi, USA, 2009.

[16] P. Ray, M. O’Rourke, and D. Edwards, “Using Collective Intentionality to Model Fleets of Autonomous Underwater Vehicles”, in Proc. of Oceans MTS/IEEE Biloxi, Biloxi, Mississippi, USA, 2009.

[17] J. Cartwright, P. Patrón, J. Evans, and D. Lane, “Developing a Distributed Ontological World Model for Autonomous Multi Vehicle operations”, 2nd SEAS DTC Technical Conf., Edinburgh, UK, 2007.

[18] P. Patrón, and D. M. Lane, “Adaptive Mission Planning: the Embedded OODA Loop”, 3rd SEAS DTC Technical Conf., Edinburgh, UK, 2008.

[19] D. G. Roberson, “Environmental Tracking and Formation Control for an Autonomous Underwater Vehicle Platoon with Limited Communication”, Ph.D. dissertation, Dept. Elect. Eng., Virginia Polytechnic Institute and State Univ., Blacksburg, VA, USA, 2008.

[20] A. R. Girard, J. Borges de Sousa, and J. K. Hedrick, “An Overview of Emerging Results in Networked Multi-Vehicle Systems”, in Proc. 40th IEEE Conf. on Decision and Control, Orlando, FL, USA, 2001.

[21] L. Padgham, and M. Winikoff, Developing intelligent agent systems: a practical guide. Wiley, 2004.

[22] JAUS Architecture Framework, Reference Architecture Specification, Vol. II, Part 1, 2007.

[23] M. R. Endsley, B. Bolte, and D. G. Jones, Designing for Situation Awareness: An Approach to User-Centered Design, Taylor and Francis, 2003.

[24] R. So, and L. Sonenberg, “Situation Awareness in Intelligent Agents: Foundations for a Theory of Proactive Agent Behavior”, in Proc. IEEE/WIC/ACM Int. Conf. on Intelligent Agent Technology, Beijing, China, 2004, pp. 86-92.

[25] J. R. Boyd, The Essence of Winning and Losing, 1996.

[26] J. A. Adams, “Unmanned vehicle situation awareness: A path forward”, in Proc. Human Systems Integration Symposium, Annapolis, Maryland, USA, 2007.

[27] C. McGann, F. Py, K. Rajan, H. Thomas, R. Henthorn, R. McEwen, “T-REX: A Model-Based Architecture for AUV Control”, in proceedings of the International Conference on Automated Planning & Scheduling, Providence, Rhode Island, USA, 2007.

[28] N. Carlési, F. Michel, B. Jouvencel, J. Ferber, “Generic Architecture for Multi-AUV Cooperation Based on a Multi-Agent Reactive Organizational Approach”, in proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems San Francisco, CA, USA, 2011.

[29] ROS wiki, available in http://www.ros.org.

[30] ROS StartGuide, available in http://www.ros.org/wiki/ROS/StartGuide.

[31] M. Ghallab, D. Nau, and P. Traverso, “Automated Planning – Theory and Practice”, Morgan Kaufmann, 2004.

[32] Eurathlon robotics competition, available at http://www.eurathlon.eu.

[33] Student Autonomous Underwater Vehicle Challenge – Europe, available at http://sauc-europe.org.

[34] DARPA Robot Challenge, available at http://www.darpa.mil/Our_Work/TTO/Programs/DARPA_Robotics_Challenge.aspx.

[35] S. Rodriguez, V. Hilaire, N. Gaud, S. Galland, and A. Koukam, “Holonic Multi-Agent Systems”, Chapter 11, Natural Computing Series, Self-organising Software, Part 2, pp. 251-279, 2011.

[36] C. C. Insaurralde, J. J. Cartwright, Y. R. Petillot, “Cognitive Control Architecture for Autonomous Marine Vehicles”, in proc. IEEE Int. Systems Conf., Vancouver, BC, Canada, 2012, pp. 117-124.

[37] F. Jammes, A. Mensch, H. Smit, Service-Oriented Device Communications using the Devices Profile for Web Services, proceedings of the 21st International Conference on Advanced Information Networking and Applications Workshops, vol. 1, 2007.

[38] The UnderWater Simulator (UWSim), available in http://www.irs.uji.es/uwsim.

[39] roscore, http://wiki.ros.org/roscore.