Supporting Human-Aware Mission Specification and Explainable Planning for Underwater Vehicles

Alan Lindsay[1], Andrés A. Ramírez-Duque[1], Bart Craenen[1], Andrea Munafò[2], Laurence Boé[2], Adam Campbell[2], Ronald P. A. Petrick[1]

[1]Heriot-Watt University, Edinburgh, UK
[2]SeeByte, Edinburgh, UK
alan.lindsay@hw.ac.uk

Abstract

The problem of effectively supporting human operators as they model missions, supervise their planning, and observe execution, combines several open questions in human-aware planning. These include how to provide support during the mission modelling and specification phase, and how to provide effective assistance during the planning phase. In this paper, we present ongoing work to create a tool for human operators, which aims to support effective interactions, supporting operator understanding and awareness throughout autonomous underwater vehicle missions. We present an architecture which incorporates modelling assistance, plan generation, explanation, plan space exploration, and execution monitoring. Our approach starts with a task-bot interface and aims to provide modelling assistance during the mission specification step and comprehensible plan explanations during planning while supporting the operator in exploring the plan space. In addition, we demonstrate our system with an example walk through to showcase the human-machine interaction and explanation representations.

Introduction

The intention in this project is to design and develop a system capable of providing effective support to a human operator in the context of specification, planning, and monitoring of surveillance missions for underwater and surface bots. In these missions, the operator must take into account various types of information, including topology, weather, robot capabilities, mission objectives and parameters, and safety requirements. This makes specifying the mission time-consuming, and potentially error-prone. Furthermore, in the context of planning the mission, there are various competing factors that must be considered, which makes finding a satisfactory plan a process of trial and error. Finally, during deployment, the operator should be able to monitor and query the system in order to understand the specific execution as it unfolds.

Mission specification is time-consuming as there are various types of information that must be captured in the model, and specification typically requires carefully describing each aspect. This includes aspects such as mission objectives, launch points, and available assets, but also includes sandbanks and exclusion zones. However, the various existing information sources provide rich information about the developing mission, which could be used to provide assistance in specification (e.g., specifying coordinates by exploiting existing known positions), and even in anticipating potential structures of the mission. For example, by using depth information it is possible to provide the user with the option of automatically placing appropriate exclusion zones, where the water is too shallow (e.g., sandbanks, around the coast).

Using the proprietary Seetrack software, mission planning utilises a two phase planning approach: resource allocation (selecting behaviours to satisfy objectives, order objectives, and allocate assets to objectives), and rehearsal (similar to path planning). As part of this process, the operator takes an active role in modifying the allocation, in order to explore the plan space, and discover better trade-offs. Within Explainable Planning (XAIP) techniques have been developed that help the user in understanding the generated plans (Magnaguagno et al. 2017), and support their exploration of the plan space (Krarup et al. 2021). However, these are currently considered using the planning model, and they do not consider how an operator will interact with the concepts: how they will express queries, and understand the generated explanations.

We are in the process of developing a system with the purpose of supporting operators through each of the stages of a mission. The system's development so far has been guided by Seebyte, one of the partners on the project, which is a company specialising in command and control end-to-end mission software. With this guidance we have developed a prototype. Our starting point for the interaction is a task-bot, which the user can use to interact with the system during each of the three stages of the mission. From the task-bot the operator can: build a mission; query the system about and lead the exploration through alternative plans; and during execution, the operator can monitor the mission. In this paper, we focus on the mission specification and planning phases. We present the main components of our system and how the functions of the task-bot are realised in the system. Finally, we present a walk through of the current system to demonstrate its current capabilities.

Related Work

Existing research has considered the planning problem for autonomous underwater vehicles, including problem formulation (Rajan, Py, and Barreiro 2013), search strategies (Carreno et al. 2020), and monitoring information opportunities during execution (Cashmore et al. 2017).

Human-aware planning is a growing field in automated planning. Our work is related to existing research that attempts to inform and support the user throughout the planning life-cycle, including modelling, planning, and executing. These include tools for providing explanations during the modelling process (Sreedharan et al. 2020), planning (Sengupta et al. 2017), and more general frameworks, e.g., (Chakraborti et al. 2018), which provides a panel of alternative views on the developing information, including the plan. There are also approaches that combine elicitation and planning/execution within a single framework, such as the factory setting, user tailoring, and execution tuning (FUTE) framework (Canal, Alenyà, and Torras 2016). These frameworks are typically iterative, specialising plan generation through either interleaving elicitation and planning episodes (Sanneman 2019), or learning from observations over time (Canal, Alenyà, and Torras 2016).

Research has considered how best to create effective explanations for users, including considering various levels of granularity (Canal et al. 2021), and approaches for sharing explanations between related domains (Lindsay 2020). A growing number of approaches to XAIP have used visualisations (Magnaguagno et al. 2017; Porteous, Lindsay, and Charles 2023; De Pellegrin and Petrick 2021; Kumar et al. 2022), and these have proven successful in creating explanations (Kumar et al. 2022). There are now several alternative methods of plan visualising, e.g., (De Pellegrin and Petrick 2021) that operate from the planning model, plan and some form of metadata, which are used to parameterise the visualisation system. Using a similar approach (Kumar et al. 2022) generates explanations for model reconciliation and uses characterising traits of planning problems to simplify the metadata specification.

In automated planning, modelling has been identified as a bottleneck, due to the skills required to develop these models. This has inspired a variety of methods for supporting the authoring of domain models, including frameworks similar to Integrated Development Environments for use by software engineers, e.g., the GIPO (Simpson, Kitchin, and McCluskey 2007), itSIMPLE (Vaquero et al. 2007) and KIWI (Wickler, Chrpa, and McCluskey 2014) systems. These modelling tools are useful for rapid development of domains by an experienced domain modeller. Another avenue of research to aid in the modelling process is based on learning models from observations: namely that of domain model acquisition, e.g., (Wu, Yang, and Jiang 2007; Mourão, Petrick, and Steedman 2010; Lindsay et al. 2017). In this work we investigate how existing data sets can be brought together in order to support the user in the modelling process, which is related to work that builds the planning model from ontologies, e.g., (Wickler, Chrpa, and McCluskey 2014; Crosby et al. 2017; Louadah et al. 2021).

Background

Following (Fox and Long 2003), a temporal planning problem can be defined as follows:

Definition 1. A Temporal Planning Problem is a planning problem, $P = \langle F, A, I, G \rangle$, with propositional and numeric fluents, F, actions, A, initial state, I, and goals, G. An action, a, is defined by a duration ($dur(a)$), start ($cond_{\vdash}(a)$), invariant ($cond_{\leftrightarrow}(a)$) and end ($cond_{\dashv}(a)$) conditions, and start ($\mathit{eff}_{\vdash}(a)$) and end ($\mathit{eff}_{\dashv}(a)$) effects, which each describe the add and delete propositions, and numeric effects. A solution (a plan) is a schedule σ, which is a sequence of pairs $\langle a, t \rangle$, where $a \in A$ is an action, $t \in \mathbb{R}^{0+}$ is the action's start time, and G holds in the state after all actions have completed. For each pair in the schedule we have action, a, starting at time, t (requiring its start conditions are satisfied), and ending at time, $t + dur(a)$ (requiring its end conditions are satisfied), with any invariant conditions holding from t to $t + dur(a)$. The start effects of a are applied at t, and the end effects are applied at time $t + dur(a)$.

The aim in temporal planning is typically to find short plans (time of final action to complete).

XAIP-as-a-Service

XAIP-as-a-service (Krarup et al. 2021) is an approach within XAIP that operates from user queries about generated plans. The approach is based on taking the user's query about a feature of a plan and generating a plan that exhibits the opposite feature. For example, taking the query, *'Why is action A in the plan?'*, the approach generates a plan without action A. It then compares the plans empirically and structurally, demonstrating how the plan changes when action A is not used. In (Krarup et al. 2021) they consider queries including:

- Q1: *'Why is A in the plan?'*
- Q2: *'Why is A not in the plan?'*
- Q3: *'Why is A used before B?'*

The approach relies on mapping user queries into constraints that can be added to the planning model, forcing any plans to exhibit the required property. For example, in the case of Q3, the aim is to generate contrastive plans where B is before A. To do this, we can use a proposition p to indicate that action B has been applied and proposition q to indicate that action A has been applied. This is achieved by adding add effects to these actions. To force B to be used before A we add p to the precondition of A and add q to the goal.

The Underwater Autonomous Vehicle Mission Scenario

The scenario under investigation then revolves around one or more underwater autonomous vehicles (UAV), or *assets* for short (Ferri et al. 2017; Hamilton et al. 2020). Assets can have different configurations, and this determines what tasks they can perform, and how they behave when performing them. For example, an asset with a side-scanning sonar can perform a sonar sweep of an area, while an asset with a camera can take underwater images or ascertain the exact location of a target.

In our scenario, a *mission* consists of assets performing a number of subsequent tasks to achieve some objectives. The first and last objective of each asset during a mission are the *launch* and *recovery* objectives, respectively. In addition,

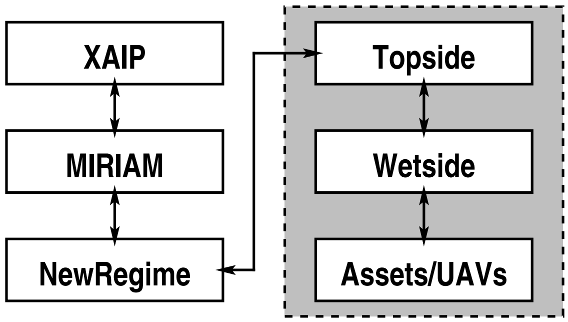

Figure 1: Diagram of the system architecture

there are two main objectives the assets can be set: to *survey* an area (for example with a sonar); and to *acquire* (the exact location of) a target. Since these objectives are normally placed in different locations, the assets must also be capable of travelling from one location to another, i.e., performing a *transit* task. Transit generally takes place on a path along several locations, called *waypoints*. Finally, part of the area in which the mission takes place can be dangerous for the assets to operate in, for example, when a sandbank is present. These areas can be declared off-limit to the asset by defining *exclusion zones*.

Tasks and objectives can be achieved in different ways, also depending on the configuration of the asset tasked to achieve them. This can be done by specifying a *behaviour* (or sequence of behaviours) to the asset for a task or objective. For example, an area can be surveyed using a lawnmower pattern behaviour, or by using spiral pattern behaviour. An asset can also be asked to exhibit certain behaviours at a waypoint, e.g., always getting a fix on its exact location before attempting to acquire a target.

Given the above, an example scenario for a single asset could then be for it: to be launched; to transit to a survey area while avoiding an exclusion zone; to survey an area using the lawnmower pattern behaviour; to transit to a waypoint to get an exact fix of its location; to attempt to acquire the exact location of a target; to transit back to a recovery location; to be recovered there.

The problem of effectively supporting human operators as they plan UAV missions and observe their execution combines several open questions in human-aware planning. These include how to provide support during the mission specification phase, and how to provide effective assistance during the planning phase. In this work we are developing a framework for supporting human operators throughout underwater autonomous vehicle missions.

System Architecture

The system architecture we present in this section builds on existing proprietary software (SeeTrack (SeeByte 2018b) and Neptune (SeeByte 2018a)) and components (Wetside and Topside), and extends these with three components (NewRegime, MIRIAM, and a planning and XAIP module) specifically designed to support a human operator in the UAV mission scenario (see Figure 1). The system used for setting up missions, or defining scenarios, consists of several parts. At the bottom layer are the assets themselves; i.e., the underwater autonomous vehicles (UAVs) with their specific hardware configurations. Running on the assets is what is called the *Wetside* component from the Neptune software, which connects the various components of the assets with a queue of actions, tasks, and behaviours. These are downloaded from above, from the Neptune software, from what is called the *Topside* component. The SeeTrack software provides a user-interface to an *operator*, allowing them to specify launch, recovery, survey, and target objectives, as well as exclusion zones etc. into the *Topside* component. The Topside component subsequently allows operators to allocate assets to the mission, providing a visual rehearsal of what would happen if the mission was run as it was defined thus far. Using this feedback, the operator can then adjust the mission plan until, eventually, they can use Topside to upload the mission plan to the Wetside software on the assets. The Topside software can also be used to monitor a mission in progress.

As part of this project, we have developed additional components to support modelling and planning. Connected to Topside through a gRPC interface is a middleware software called NewRegime. NewRegime provides, through interfaces, the capability to programmatically apply similar functions that the operator would apply through the Topside user-interface, e.g., adding or removing objectives, allocating resources, changing behaviours, etc. NewRegime also makes available extra or external information to the planning process, e.g., locations of hazardous areas (e.g., sandbanks) etc. This allows for (parts of) the mission specification process to be automated, or supported by other systems or information sources.

On top of NewRegime, we developed a task-bot assistant based on the MIRIAM (Multimodal Intelligent inteRactIon for Autonomous systeMs) system (Hastie et al. 2017), which guides operators in a planning-oriented interaction to actively shape the underlying task.

Finally, we have developed a component for supporting the operator in understanding plans and exploring alternative plans. We have developed this component in order to decouple the new features from the proprietary software, which allows us to exploit a concise representation of the planning problem, and its clean separation from the solver. Within this component we use a PDDL planner to solve the problems extracted through NewRegime. The extracted models are also used to support query interpretation, and are used to construct answers to the operator queries, and generate explanations of the plans.

Natural and Visual Language Interface

As the operator and the system have their own models of the scenario, it is important that the system can facilitate their communication. As a consequence, we have focused on developing approaches for both mapping natural language queries onto the machine's problem representation and mission concepts, and providing visual and textual methods of communicating the results back to the operator. These approaches extend the MIRIAM system (Hastie et al. 2017), allowing the operator to benefit from support during model specification, and to both deepen their understanding of the trade-offs between different planning decisions, and also to influence the eventual mission plan, by adding additional (soft) constraints that were missing from the initial problem specification.

User Input and Mapping

MIRIAM core is based on Keith Sterling's chatbot framework, programm-y[1]. This framework leverages the benefits of using Artificial Intelligence Mark-up Language (AIML) to quickly generate representations of task-specific indexable knowledge for chatbot QA and command and control (C2) interaction. The MIRIAM interface has two main components (Hastie et al. 2017). Firstly, an AIML-based Natural Language Processing (NLP) engine parses the user's input for the category, a basic unit of knowledge, formalising it as a semantic representation. In addition, each category designed is complemented by a series of reductions and recursion to add flexibility to the pattern-matching and normalisation-denormalisation process. Secondly, a set of extension libraries written in Python3 create a suitable reply by processing and retrieving the relevant information from the NewRegime and XAIP modules.

During execution, the NLP engine analyses the user input to categorise it, identifies the relevant parameters in the query, and populates the slots with this information. Then, the system matches each identified category with a function from the extensions library and processes the filled slots accordingly.

The extension library currently supports the execution of functions grouped into three topics: mission planning, mission monitoring, and plan contrastive explainability. Using MIRIAM, the operator can create and manipulate any objective, assign tasks to available assets, and monitor the assets' behaviours, properties, and progress after they have been launched. Possible natural language interactions include asking about the asset's current status, the mission and its current objectives, the asset's estimated arrival time at specific locations, and the mission's objectives completed. For example, given a user input *'Could you show me the current mission progress, please?'*, the NLP engine will first normalise and remove punctuation and, subsequently, will perform a word-level search on the pattern tree. Once a pattern is detected, the AIML interpreter matches wild cards, invokes the Python extension and passes them through. Figure 2 shows two simplified examples of category definition and the AIML interpreter process.

While operators create the mission, MIRIAM provides them access to NewRegime shortcut functions, locations database, and previous mission information, such as hazardous areas, the target's location, and survey areas. Offshore wind farms are an example of underwater mission inspection, where relevant locations are known and stored in a database. In that scenario, an operator could upload the locations into NewRegime and access them using the extension libraries available on MIRIAM. Additionally, during the planning phase, MIRIAM provides the user with access to essential functions using NL to explore the planning model, one function for each type of XAIP query, and additional functions to, for example, force the plan to keep or drop any constraints and to show the current plan.

[1]A fully compliant AIML 2.1 chatbot framework written in Python3: https://github.com/keiffster/program-y

```
<category>
  <pattern># MISSION PROGRESS ^</pattern>
  <template>
    <extension
    path="miriam2_bot.extensions.core.mission.MissionExtension">
    MISSION_PROGRESS <star/>
    </extension>
  </template>
</category>
 >> Normalising input from
   [Could you show me the current mission progress , please ?]
  to
  [COULD YOU SHOW ME THE CURRENT MISSION PROGRESS , PLEASE ?]
- Removing punctuation...
- Topic pattern detected = [MONITORING]
- Matching [COULD YOU SHOW ME THE CURRENT MISSION PROGRESS PLEASE]
- Matches...
- 1: Match=word Node=ZEROORMORE [#] Matched=COULD YOU SHOW ME THE CURRENT
- 2: Match=word Node=WORD [MISSION] Matched=MISSION
- 3: Match=word Node=WORD [PROGRESS] Matched=PROGRESS
- 4: Match=word Node=ZEROORMORE [#] [PROGRESS] Matched=PROGRESS
- AIML Parser evaluating template resolved to [MISSION_PROGRESS]
- Importing module [miriam2_bot.extensions.core.mission.MissionExtension]
- [[EXTENSION miriam2_bot.extensions.core.mission.MissionExtension]]
  resolved to
  [Up to this point, 35 percent of the mission has been executed]
  [and the estimated time remaining is 45 minutes]
...
<category>
  <pattern># CREATE # TARGET *</pattern>
  <template>
    <srai> CREATE TARGET CARRIED OUT
      <extension path="miriam2_bot.extensions.core.planning.PlanningExtension">
      CREATE_OBJECTIVE <denormalize><star index="2"/></denormalize> target
      </extension>
    </srai>
  </template>
</category>
 >> Normalising input from
  [Let's create a Target for pile]
  to
  [LET CREATE A TARGET FOR PILE]
- Removing punctuation...
- Topic pattern detected = [PLANNING]
- Matching [LET CREATE A TARGET FOR PILE]
- Matches...
- 1: Match=word Node=ZEROORMORE [#] Matched=LET
- 2: Match=word Node=WORD [CREATE] Matched=CREATE
- 3: Match=word Node=ZEROORMORE [#] Matched=A
- 4: Match=word Node=WORD [TARGET] Matched=TARGET
- 5: Match=word Node=ONEORMORE [*] Matched=FOR PILE
- AIML Parser evaluating template resolved to [CREATE_OBJECTIVE]
- Importing module [miriam2_bot.extensions.core.planning.PlanningExtension]
- [[EXTENSION miriam2_bot.extensions.core.planning.PlanningExtension]]
  resolved to
  [<srai> CREATE TARGET CARRIED OUT true </srai>]
- SRAI resolved to
  [The operation create target was carried out successfully]
```

Figure 2: World-level matching pattern process: a fundamental task of the NLP Engine.

Rendering System Responses

To provide a mechanism for the user to understand and explore the planning task using different multimedia resources, we developed a web-based user interface as the front end of the MIRIAM interface. MIRIAM exposes an input text channel as an RESP endpoint using standard HTTP requests to allow the user NL interaction. Then, after the backend NLP engine processes the query, the answer is sent back to the client, where the response's elements are rendered as embedded images, verbalised text and extended NL explanations.

Regarding the XAIP module, when the user asks MIRIAM to explain some actions presented in the plan, the XAIP extension response is rendered by applying visual and textual representations as follows:

Textual Representation Our approach to rendering plans in text follows (Canal et al. 2021) and relies on an annotated domain file to generate relevant sets of candidate rules for matching action descriptions. Each action is associated with tags detailing verbs and prepositions that might be used in describing the action. For example, for the move action, we include 'move', 'navigate', 'go', and 'drive' as alternative verbs, and prepositions such as 'from [the] *from*', 'starting at [the] *from*', and 'from [the] *?from*'. We also support summarising rules, which allow certain chains of actions to be condensed where an effect summarisation exists (e.g., as in the case of sequences of transit actions).

Visual Representation The plans are plotted over a top-down view, identifying the mission's key aspects, landmarks, areas, objectives, etc. Each asset is depicted using an individual line. This representation is based on the visual language that operators are familiar with, and as such, should support effective communication of the plans.

Supporting Mission Specification in NewRegime

NewRegime sits between the taskbot interface and the mission system. During modelling, NewRegime can exploit its access to additional information, such as the topology of the seabed, weather and current information, etc., so that while the operator builds the model through the taskbot, NewRegime is proactively interrogating the partially specified mission in order to identify potential missing structure, which it can then propose that it adds. For example, in a mission where the assets are supposed to survey areas deep underwater, the location of dangerous shallow waters, e.g., sandbanks, will have a major effect on the mission specification. It could lead to the software suggesting the addition of an exclusion zone over the sandbank.

Exclusion zones come with their own set of rules as well, and NewRegime is capable of checking whether they are satisfied or violated. Assets, for example, are to avoid entering (or going too near to) an exclusion zone while attempting an objective, or while in transit from one objective to the next. Launch and recovery locations cannot be inside exclusion zones either, nor can a survey area be entirely encompassed inside an exclusion. And if an exclusion zone partially overlaps a survey area, only the part outside the zone will be surveyed. Likewise, the location of an acquire objective cannot be inside an exclusion zone.

As such, the addition of an exclusion zone, simply to ensure that assets do not go near to a dangerous part of the operation area, can have a profound effect on what the eventual specification of the plan. And NewRegime can respond re-actively, or pro-actively to this use of extraneous information. It can respond re-actively by alerting the operator to a violation of the rules, e.g., by explaining that part of a survey area is now covered by an exclusion zone, and will therefore not be surveyed during the mission. Or it can respond pro-actively by, e.g., moving the launch and recovery locations of the mission out of the newly added exclusion zone to new locations; locations selected based on operator supplied parameters or some other logic.

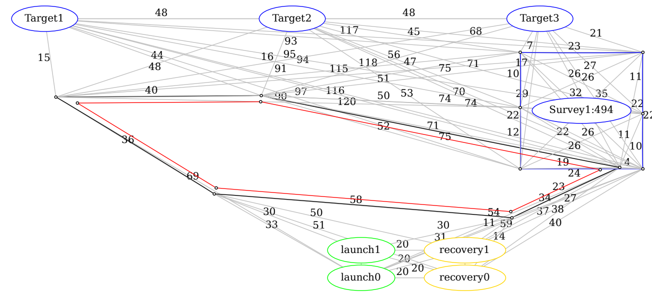

Figure 3: An example structure extracted from the mission model and represented in PDDL.

Supporting Plan Understanding and Plan Space Exploration

To decouple from the proprietary planning system, we built a separate planning module based on a temporal PDDL model. NewRegime extracts the key mission data, and MIRIAM uses it to populate a readable structure for XAIP to define a problem model. From this model, we can use a general-purpose planner to generate plans, use existing explainability techniques to explain the plan and allow the user to explore the plan space.

Problem Extraction and Plan Generation

The main mission elements are gathered from the mission specification, and used to define a planning model (see Figure 3). The first step is to extract the main coordinates for the mission, including exclusion zones, objectives, asset launch and recovery points, and transponder positions. These coordinates are used to define a basic topology: each pair of points is determined as a traversable edge if it does not pass through an exclusion zone. Time to traverse is determined based on the edge lengths. Objects are included to represent underwater vehicles, waypoints, and objectives. A goal is made for each objective (e.g., observation points, and survey areas), and for each vehicle to be at its recovery position. Finally, the overall metric is specified, e.g., to minimise time, or to balance asset use.

We have defined a domain model that represents the necessary actions: navigation, diving, climbing, surveying, and observing. These actions are sufficient to consider the general differences between allocating resources to objectives, and varying the order that objectives are satisfied. The propositions and functions of the model capture the scenario: the topology, the actions' durations, and which objectives have been satisfied, and each robots' positions (represented by waypoints), its height.

Our approach uses the OPTIC planner (Benton, Coles, and Coles 2012), which is sensitive to alternative metric functions.

Plan Understanding and Plan Space Exploration

An important part of the human-machine interaction is in the planning of the mission. The standard method requires the operator to detail the mission, and its properties. They then

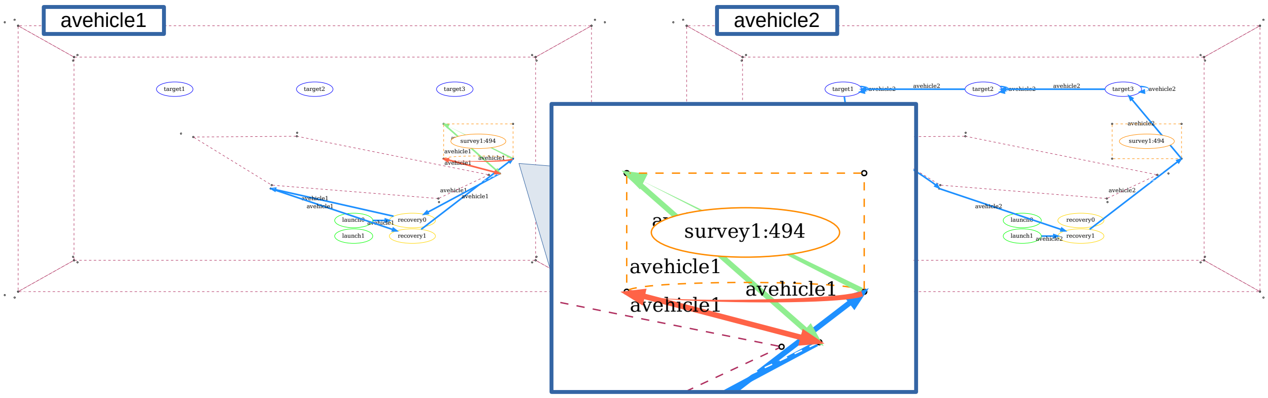

Figure 4: The figure compares the partial plans for two assets. For *avehicle2*, the plans are identical. For *avehicle1*, the plans differ in the specific survey activity (shown in the inset).

have a hands-on approach, allocating assets to objectives, and creating the order in which they should be attempted. We are investigating using XAIP techniques to provide the operator with support as they attempt to understand the plan space. This process has been considered in the context of user queries. Our system can generate explanations of the current plan, or it can use the query to generate an alternative constrastive plan. In the latter case, through a series of operator queries, the system can allow the user to guide a joint exploration through the plan space.

The most appropriate type of explanation will, of course, depend on the type of the user's query. The first type of explanation that we support relies on a causal graph analysis, which justifies an action in a plan by demonstrating how it forms part of a chain of actions that supports part of the goal. This type of query is particularly useful in explaining which goals a particular action is supporting. For example, here is an example textual description – a graphical presentation is also provided to the user.

```
User: Which goals does surveying with avehicle2 support?
MIRIAM: It leads directly to achieving the goal fact:
   * area_surveyed survey1
 Overall, the action is in the causal chain of the
   following goals:
   * area_surveyed survey1
   * loc_at avehicle2 recovery1
 <Causal analysis figure link>
```

We also support contrastive explanations (following the approach in (Krarup et al. 2021)), which allow the operator to explore and understand the space of possible plans by asking queries to the system. For example, the operator can ask a query, such as '*why is avehicle1 being used to survey area 1?*'. In this case, the system uses the query to generate a plan where the opposite is true. For example, in the example above, a plan would be generated where *avehicle1* is not used to survey *area 1*. These new plans can be used to generate an explanation, such as '*If avehicle2 is used to survey area 1 then the plan becomes 10 steps longer*'. Through chaining queries together, the operator can influence the next generated plan, and build an understanding of the plan space.

Understanding the Difference Between Plans

We observe that it is natural in this domain to view the assets as agents, even if it is necessary to plan all of the agents together. We have implemented a plan comparison visualisation, which exploits the multi-agent nature of the plans, by plotting the plan comparison in a series of separate visualisations – one per asset. For example, Figure 4 presents the visualisation for *avehicle1* (left) and *avehicle2* (right). For a particular asset, ϕ_i, we can extract the parts of plan that are relevant, denoted π^{ϕ_i}. For the original plan (π_0) and the new plan (π_1), and asset ϕ_i, we can identify the pair of asset specific partial plans: $\pi_0^{\phi_i}$ and $\pi_1^{\phi_i}$. We then use the plan to generate three line segments: the parts where the plans match (presented in blue); the part of the original plan $\pi_0^{\phi_i}$ that is not in the new plan $\pi_1^{\phi_i}$ (in green), and the part of the new plan $\pi_1^{\phi_i}$, which is not in the old plan (in red). To do this, we used the Levenshtein distance (Levenshtein 1966): the distance between two-word sequences which provides a measure of the edit difference between the sequences while also respecting order. In our case, we use unique words for each ground action and extract the best match between the two action sequences. In the figure, the partial plan on the right is plotted entirely in blue, meaning that the plans are the same. On the left, the specific survey start/stop points have changed (this is shown in the inset).

Generating Explanations for Queries With Unknowns

When creating explanations, it has generally been assumed that a specific target is identified. For example, it is assumed in (Krarup et al. 2021) that the parameters (e.g., actions A and B in queries of type Q1-3), are known. However, a key issue with user queries is that when describing actions, people tend to miss out details, e.g., (Lindsay et al. 2017), such as the starting location of a *move* action. As a consequence it is not always possible to fully specify the query's parameters (i.e., identify specific actions) from the user's input. As part of our solution we have generalised the definition of the mapping between actions and constraints to include partially

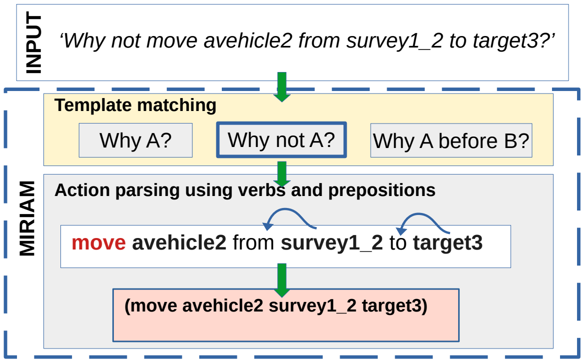

Figure 5: Templating is used to select the appropriate function for a user query (top). Each action description within the query is then individually processed to (partially) specify an action header.

specified actions. In this section, we describe how our system supports natural language user queries.

Action Specification

The MIRIAM system uses the AIML-NLP engine to parse the user queries with an appropriate category (see 'User Input and Mapping'). As part of this process, we expect that the appropriate query type (e.g., '*Why A?*') has been identified and the action descriptions have been extracted. We then further process these action descriptions to identify the appropriate ground actions relevant to the user's query. This process is achieved within the planning component, which allows us to exploit naming information from the specific mission, including assets, landmarks, and objectives.

As with for plan description generation (see 'Textual Representation'), our approach follows (Canal et al. 2021) and relies on an annotated domain file to generate relevant sets of candidate rules for matching action descriptions. We can exploit the same tagging information for verbs and prepositions. As with (Canal et al. 2021) the verbs are then extended using `mlconjug3` (Diao 2023), which can be used to provide alternative verb conjugations, supporting user input in various tenses.

This information is used in a simple matching algorithm to determine the best matching action for the description. For each action the approach attempts to fill as many of the parameters as possible. The sentence is examined in the context of each preposition, and the number of covered parameters are noted. A parameter is covered if the sentence includes a preposition followed by an object name of the correct type. This score is combined with a score indicating whether an appropriate verb was used. The action with the highest score is selected. Figure 5 (bottom) demonstrates the approach. In the top, the '*Why not A?*' template was selected. The appropriate part of the sentence is identified and the prepositions and verbs are matched to the *move* action. The final step is to build the action template by extracting the objects that were identified by prepositions. We also force in any additional objects referenced in cases where there is no ambiguity (i.e., they are the only object referenced with an appropriate type for a parameter). Notice that because we attempt to match with each action in turn, the approach can work even if the user does not use an anticipated verb.

Partial Action Specification Typically, these action descriptions will not fully describe one single instance of an action. For example, if the planner generates the plan:

- *move truck1 from loc1 to loc2*
- ...
- *move truck1 from loc2 to loc3*
- *move truck1 from loc3 to loc2*.

The user might ask '*why did truck1 go to loc2?*', which does not have sufficient information to fully describe an action. In the matching approach above, when a parameter is not indicated by a relevant preposition, and the parameter cannot be unambiguously forced, then the parameter is marked as unknown (denoted '_'). In the example above, the action description does provide enough information to describe a partial `move` action header, e.g., *(move truck1 _ loc2)*, where the second parameter is unknown.

Matching Actions With Unknowns

The interpretation of a partially specified action header relies on defining the space of all possible actions, the subset of possible actions in a specific plan π, and the actions that are possible, but not in the plan.

Definition 2 (Possible Actions)**.** The set of actions that are reachable from the initial state.

Definition 3 (Plan Actions)**.** The set of actions that are in a plan, π.

Definition 4 (Non-Plan Actions)**.** The set of actions that are not in a plan, π (i.e., the difference between the sets of possible and plan actions).

Given a query of '*Why A?*' or '*Why A then B*' type, we expect that the action descriptions for A and B will exist in the *Plan Actions* set. In the case of '*Why not A?*' then we would expect the action A to be in the *Non-Plan Actions* set. This provides the scope for interpreting the action descriptions of partially specified actions. Of course, there may be incorrectly formed queries, and if they can be identified (e.g., a '*Why A?*' query and A is not in *Plan Actions*) then they can be addressed separately.

Depending on the context of the action in the query there are different interpretations available given a partially specified action header as part of a query:

- Use the partially specified action header,
- Disambiguate – More information from the user,
- Disambiguate – Use the plan ('*Why A?*' or '*Why A then B?*' only).

In our current system we use the first and last of these depending on the type of query.

Causal Graph Explanation In order to generate a causal graph explanation —a type of explanation that is only generated for an action in the plan— then we use the plan to disambiguate. In our current approach we simply generate the explanation based on the first action that satisfies the partially specified action.

```
 User: First, create a new mission
 MIRIAM: OK, it's called Scenario 1
 User: OK, please move the view to the NnG offshore wind farm location.
 MIRIAM: OK, the view location is 56.26 degrees north -2.34 degrees east.
 User: Let's create a Target for pile_1, pile_2 and pile_3
    <Looking up pile locations in NnG db>
 MIRIAM: The targets have been created successfully
 User: Add an exclusion zone over the sandbank_1 300 meters south of the pile_3
    <Looking up sandbank_1 location in NnG db>
 MIRIAM: I have added an exclusion zone south of the piles successfully
 User: Please create a launch and recovery point
    <Looking up objectives' default locations in NnG db>
 MIRIAM: I have created launch and recovery points outside the exclusion zone
 User: can you move objective launch1 500 meters south from target pile_1?
 MIRIAM: The launch1 objective has been moved successfully
...
 User: Does the mission look OK?
 MIRIAM: Do you want to put a boundary around the mission?
 User: Yes, please.
 MIRIAM: OK, I have created boundary exclusion zones around the mission.
...
 User: Let's allocate assets vehicle1 and vehicle2 and then do a rehearsal.
 MIRIAM: OK done! Here's my current plan.
     <Shows plan>
 User: Why not move vehicle2 from survey1_2 to target3?
 MIRIAM: Well, the plan becomes 30 seconds longer if we use that action.
     Do you want to see the comparison?
     <Loading comparison visualisation>
```

Figure 6: An example interaction using the system. The operator first specifies the mission, then positions the view location over a wind farm. Next, inspecting tasks are added for three pylons. Then, the operator adds hazardous areas, such as sandbanks, and creates an exclusion zone around them. After modelling is completed, the operator conducts a rehearsal and queries the generated plan using XAIP.

Contrastive Explanations In the case of contrastive explanations, we assume that the use of the partially specified action header will result in the clearest explanation. For example, in the case of '*Why A?*' queries, the use of the partially specified action header is translated into a constraint that prevents any of the possible actions that is consistent with the partial action header.

The mappings between queries and constraints defined in (Krarup et al. 2021) (overview in the 'Background' Section) are extended to allow them to use partially specified action headers. Each partially specified action can be used with the possible actions set in order to make a set of all actions that are consistent with the partially specified action header. In the original approach, propositions are used in the model to either force or prevent the use of actions in the model. In our approach, these propositions are used similarly, except they force the use of any of a set of actions, or prevent the use of any of a set of actions.

The use of partial action headers would allow the user to be precise about the query that they want to express, as they can include the parameters that they want to be part forced in the constraint. In a future elicitation exercise we will interview operators in order to ascertain how to make the mapping between specific user action descriptions and the precise constraint generated intuitive.

Interaction Walk Through

In Figure 6, we present an example walk through, demonstrating how the interface can be used to support the user in specifying a mission by supporting the necessary functions and incorporating appropriate defaults where possible. For example, after setting up a mission, the operator positions the simulator's view at the centre of the target area, in this case, the wind farm of interest. Next, the operator adds an inspecting task for three pylons and assigns a target to each one. The locations of the pylons have already been stored in a database. The operator then uses information from the database again to identify hazardous areas, such as sandbanks and proceeds to create an exclusion zone around them. After other default interactions, such as defining launch and recovery points and bounding the area, the operator decides to conduct a rehearsal, ending the modelling stage. At this point, everything is ready for planning and XAIP.

Once modelling is complete, the planning component pulls the relevant structure from the mission and makes a plan. The user can then use MIRIAM to examine the plan. They can ask for a visualisation, a comparison, or an explanation. The explanation queries will either be one of the three (Q1-Q3) introduced in 'XAIP-as-a-Service' or will ask for a causal explanation. In each case, MIRIAM identifies the appropriate type of query and requests the appropriate function. Finally, in order for the appropriate parameters to be extracted, we have detailed above how we extract the information from user action descriptions.

In the example from Figure 6, the user queries why a particular transit was not used (i.e., moving $avehicle2$ from $survey1\_2$ to $target3$). The system first extracts an appropriate action header as a parameter for the Q2 query; in this case ($move\ avehicle2\ survey1\_2\ target3$). It then adds a constraint to the planning model in order to force the planner to select this action as part of any valid plan. It then generates a new plan and uses the original and new plan as the basis of a comparison. The new plan, in this case, is longer. The user can view a comparison visualisation (e.g., similar to Figure 4).

Conclusion and Future Work

In this paper, we have presented ongoing work that aims to build a tool to support human operators to improve planning understanding and awareness throughout underwater autonomous vehicle missions. We presented our architecture, which incorporates modelling assistance, plan generation, explanation, plan space exploration, and execution monitoring. We have developed a user interface that extends the task-bot interface with visual content. Our system exploits existing information resources to provide modelling assistance, making suggestions to the operator to reduce the aspects of the model that must be specified. Our planning support component uses textual and visual explanation approaches and exploits explainable planning techniques in order to facilitate the operator's explanation and understanding of the plan space. One aspect of the work so far has involved dealing with incomplete user queries, and we demonstrate how this is managed through a generalisation of existing explainable planning approaches. We presented a walkthrough to demonstrate the main features of our system. We are currently in the process of designing an elicitation exercise, where we will conduct partially structured interviews with expert operators of the mission software. Our main objective in this study is to establish how the system and operator can communicate effectively: what types of explanations are useful, how they would prefer information presented to them, and how they will communicate with the system.

Acknowledgments

This work was partially supported by EPSRC Grant No. EP/V05676X/1 through the HUME Prosperity Partnership.

References

Benton, J.; Coles, A.; and Coles, A. 2012. Temporal planning with preferences and time-dependent continuous costs. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 22, 2–10.

Canal, G.; Alenyà, G.; and Torras, C. 2016. Personalization framework for adaptive robotic feeding assistance. In *International Conference on Social Robotics*.

Canal, G.; Krivic, S.; Luff, P.; and Coles, A. 2021. Task Plan verbalizations with causal justifications. In *ICAPS 2021 Workshop on Explainable AI Planning (XAIP)*.

Carreno, Y.; Pairet, È.; Petillot, Y.; and Petrick, R. P. 2020. A decentralised strategy for heterogeneous auv missions via goal distribution and temporal planning. In *Proceedings of the international conference on automated planning and scheduling*, volume 30, 431–439.

Cashmore, M.; Fox, M.; Long, D.; Magazzeni, D.; and Ridder, B. 2017. Opportunistic Planning in Autonomous Underwater Missions. *IEEE Transactions on Automation Science and Engineering*, 15(2): 519 – 530.

Chakraborti, T.; Fadnis, K. P.; Talamadupula, K.; Dholakia, M.; Srivastava, B.; Kephart, J. O.; and Bellamy, R. K. E. 2018. Visualizations for an Explainable Planning Agent. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*.

Crosby, M.; Petrick, R.; Rovida, F.; and Krueger, V. 2017. Integrating mission and task planning in an industrial robotics framework. In *Proceedings of the International Conference on Automated Planning and Scheduling*.

De Pellegrin, E.; and Petrick, R. P. 2021. Automated Planning and Robotics Simulation with PDSim. In *Proceedings of the ICAPS Workshop on Knowledge Engineering for Planning and Scheduling (KEPS)*.

Diao, S. 2023. mlconjug3. *GitHub. Note: https://github.com/Ars-Linguistica/mlconjug3 Cited by*.

Ferri, G.; Munafò, A.; Tesei, A.; Braca, P.; Meyer, F.; Pelekanakis, K.; Petroccia, R.; Alves, J.; Strode, C.; and LePage, K. 2017. Cooperative robotic networks for underwater surveillance: an overview. *IET Radar, Sonar & Navigation*, 11(12): 1740–1761.

Fox, M.; and Long, D. 2003. PDDL2.1: An extension to PDDL for expressing temporal planning domains. *J. Artif. Intell. Res. (JAIR)*, 20: 61–124.

Hamilton, A.; Holdcroft, S.; Fenucci, D.; Mitchell, P.; Morozs, N.; Munafò, A.; and Sitbon, J. 2020. Adaptable Underwater Networks: The Relation between Autonomy and Communications. *Remote Sensing*, 12(20).

Hastie, H.; Garcia, F. J. C.; Robb, D. A.; Patron, P.; and Laskov, A. 2017. MIRIAM: a multimodal chat-based interface for autonomous systems. In *Proceedings of the 19th ACM International Conference on Multimodal Interaction*.

Krarup, B.; Krivic, S.; Magazzeni, D.; Long, D.; Cashmore, M.; and Smith, D. 2021. Contrastive explanations of plans through model restrictions. *J. Artif. Intell. Res. (JAIR)*.

Kumar, A.; Vasileiou, S. L.; Bancilhon, M.; Ottley, A.; and Yeoh, W. 2022. VizXP: A Visualization Framework for Conveying Explanations to Users in Model Reconciliation Problems. In *Proceedings of the International Conference on Automated Planning and Scheduling*.

Levenshtein, V. I. 1966. Binary codes capable of correcting deletions, insertions and reversals. *Cybernetics and Control Theory*, 10: 707–710.

Lindsay, A. 2020. Using Generic Subproblems for Understanding and Answering Queries in XAIP. In *ICAPS 2020 Workshop on Knowledge Engineering for Planning and Scheduling (KEPS)*.

Lindsay, A.; Read, J.; Ferreira, J. F.; Hayton, T.; Porteous, J.; and Gregory, P. J. 2017. Framer: Planning models from natural language action descriptions. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*.

Louadah, H.; Papadakis, E.; Mccluskey, T. L.; Tucker, G.; Hughes, P.; and Bevan, A. 2021. Translating ontological knowledge to PDDL to do Planning in Train Depot Management Operations. In *36th Workshop of the UK Planning and Scheduling Special Interest Group*. AAAI press.

Magnaguagno, M. C.; Pereira, R. F.; Móre, M. D.; and Meneguzzi, F. 2017. WEB PLANNER: A tool to develop classical planning domains and visualize heuristic state-space search. In *Proceedings of the Workshop on User Interfaces and Scheduling and Planning, UISP*, 32–38.

Mourão, K.; Petrick, R. P. A.; and Steedman, M. 2010. Learning action effects in partially observable domains. In *Proc. 19th European Conference on AI (ECAI)*. IOS Press.

Porteous, J.; Lindsay, A.; and Charles, F. 2023. Communicating agent intentions for human-agent decision making under uncertainty. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*.

Rajan, K.; Py, F.; and Barreiro, J. 2013. *Marine Robot Autonomy*, chapter Towards Deliberative Control in Marine Robotics. Springer New York.

Sanneman, L. 2019. Preference Elicitation and Explanation in Iterative Planning. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

SeeByte. 2018a. Neptune Technical Whitepaper. Accessed March, 2024. https://www.seebyte.com/media/1427/neptune-technical-whitepaper.pdf.

SeeByte. 2018b. SeeTrack v4 Technical Whitepaper. Accessed March, 2024. https://www.seebyte.com/media/1136/seetrack-v4-technical-whitepaper.pdf.

Sengupta, S.; Chakraborti, T.; Sreedharan, S.; Vadlamudi, S. G.; and Kambhampati, S. 2017. Radar—a proactive decision support system for human-in-the-loop planning. In *2017 AAAI Fall Symposium Series*.

Simpson, R. M.; Kitchin, D. E.; and McCluskey, T. L. 2007. Planning domain definition using GIPO. *Knowledge Engineering Review*.

Sreedharan, S.; Chakraborti, T.; Muise, C.; Khazaeni, Y.; and Kambhampati, S. 2020. –D3WA+–A Case Study of XAIP in a Model Acquisition Task for Dialogue Planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*.

Vaquero, T. S.; Romero, V.; Tonidandel, F.; and Silva, J. R. 2007. itSIMPLE 2.0: An Integrated Tool for Designing Planning Domains. In *International Conference on Automated Planning and Scheduling*, 336–343.

Wickler, G.; Chrpa, L.; and McCluskey, T. L. 2014. KEWI - A Knowledge Engineering Tool for Modelling AI Planning Tasks. In *International Conference on Knowledge Engineering and Ontology Development*, 36–47.

Wu, K.; Yang, Q.; and Jiang, Y. 2007. ARMS: An automatic knowledge engineering tool for learning action models for AI planning. *The Knowledge Engineering Review*.