

# A Graphical Mission Specification and Partitioning Tool for Unmanned Underwater Vehicles

Gary Giger, Mahmut Kandemir  
Pennsylvania State University  
University Park, PA 16802, USA  
{giger, kandemir}@cse.psu.edu

John Dzielski  
Applied Research Laboratory/PSU  
University Park, PA 16802  
jed@enterprise.arl.psu.edu

**Abstract** - The use of Unmanned Underwater Vehicles (UUVs) has been proposed for several different types of applications including hydrographic surveys (e.g., mapping the ocean floor and exploring sunken wreckage), mine detection and identification, law enforcement (e.g., enforcing certain fishing regulations), environmental and pollution monitoring, and even performing surveys to find potential drilling locations on the ocean floor for the oil industry. Recently the idea of using multiple, cooperating UUVs to execute these missions has also been proposed. There are two main factors that dictate a particular mission's success. The first factor regards creating a mission that is free from errors, both syntactically and semantically. The second factor deals with properly splitting a mission into a set of sub-missions and assigning each sub-mission to a group of UUVs. Even though tools have been developed to help reduce these potential problems such as high level mission programming languages, compilers for these languages, and utilities to automatically split a user specified mission, the potential still exists for errors when creating a mission (e.g. semantic errors introduced from programming and maintaining the code for existing missions). The goal of this paper is to present a programming-free, parallel mission generation utility that uses an existing graphical tool called Nobeltec. Our utility allows an operator to graphically specify a mission and the resulting set of parallel missions are displayed on screen. The main contribution of this tool is that it relieves the user from low-level mission programming, all the potential problems that come with it, and manual partitioning of the mission across the available UUVs. Thus, no matter what the application is, this tool is another step towards successfully creating missions for either for a single UUV or a group of UUVs.

## I. INTRODUCTION

The use of Unmanned Underwater Vehicles (UUVs) has been proposed for several different types of applications including hydrographic surveys for mapping the ocean floor and exploring sunken wreckage [3], mine detection and identification [4], law enforcement such as enforcing certain fishing regulations [16][17], environmental monitoring such as pollution monitoring [17], and even performing surveys to find potential spots on the ocean floor for oil drilling operations [5]. Recently the idea of using multiple, cooperating UUVs to execute these missions has also been proposed. Whether the application for a UUV is finding potential drilling locations for an oil platform or searching for mines, successfully creating missions (either for a single UUV or a group of UUVs) poses many challenging problems.

There are two critical issues in this context. The first is attempting to create a mission free from syntactic errors as

well as semantic errors. Second, if multiple UUVs are used for a mission, one has to properly split these missions into a set of sub-missions and assign them in an optimal way to each vehicle. Both of these factors play a key role in a given mission's success. Fortunately, tools have been created recently to aid in the resolution of some of these issues. Such tools include mission programming languages (MPLs) [1][7][12][13] with compiler support that allow an operator to create missions easily (similar to that of general purpose high level languages such as C, C++ and Java) as well as utilities that generate parallel sub-missions from a given high level sequential mission description. Even though these tools were developed to make mission creation and maintenance easier rather than using a low level language or writing a set of parallel missions by hand, there are still many potential problems that have to be addressed. For example, if an operator writes a mission using an MPL, the potential still exists for the operator to make mistakes the same as with any programmer writing a program using a generic high level language. While a compiler typically catches some of these errors (e.g., those related to the syntactic structure of the program), semantic errors are hard to catch and fix at compile time. In addition, when the mission description written in an MPL is to be read by someone who is not the author of that mission text, it may be difficult to understand easily what the mission is supposed to do. As a result, mission text maintenance and update can be very problematic. Third, an MPL is typically specific to a UUV and does not port to other UUVs at all, or require extensive effort on the programmer side to port it. While there already exist several attempts at developing a universal MPL [14][15][18] that can execute on any vehicle, this is not expected to happen very soon in practice. Even if this is realized someday, such an MPL has to be extended periodically to keep up with new vehicle capabilities and emerging mission requirements.

Therefore, there is a clear need for developing more user-friendly programming environments for writing and visualizing missions that require multiple, cooperating UUVs. This paper is an attempt at this direction, and proposes a programming-free parallel mission generation through a graphical tool. Our tool has four major components. The first component is written within a maritime navigational package called NobelTec [8] and helps the user specify the high level mission very easily without indicating explicitly any low level details about the parallelization and workload distribution. For example, using this specification, the user can indicate a

survey area that has to be covered by multiple UUVs and the types of the UUVs to employ to cover that area. The second component is a translator from this graphical description to a mission programming language (MPL). While one can potentially use an existing MPL for this, our current implementation uses a custom one based on the mission programming language of Seahorse UUVs. The third component is an automatic mission parallelizer which takes the description of the mission in this (user-transparent) MPL and parallelizes it across the specified set of UUVs. This parallelization is transparent to the user and can be targeted at different objective functions such as minimizing total mission execution latency or reducing power consumption. The fourth component of our tool maps back the parallel sub-missions generated by the third component above to the NobelTec representation, again in a totally user-transparent manner.

The combined effect of all these components is that the user defines his/her mission requirements and available resources (UUVs and their sensors) using a graphical interface and the specified mission is automatically parallelized by the tool and the result of this parallelization is displayed on screen. The main contribution of this tool is that it relieves the user from low-level mission programming (using an MPL), all the potential problems that come with it, and manual partitioning of the mission across the available vehicles. Therefore, this approach does not only eliminate the need for programming but also allows the user to visualize his/her parallelized mission in a very user-friendly fashion. When/if necessary, the user can modify the mission description and/or the available set of vehicles (or their characteristics) and ask the tool to repeat the three components.

The rest of this paper is organized as follows. Section II discusses our existing tools including our high-level mission programming language, its corresponding compiler, and a mission splitting utility. Section III presents our new graphical mission specification and partitioning tool for creating a set of parallel sub-missions for a group of UUVs. More specifically, the details of the four components of our tool are described in detail and an experimental evaluation of it will be shown. Section IV discusses our future work and our conclusion.

## II. EXISTING TOOLS

This section discusses some existing tools including a high-level MPL, a compiler for this MPL, and a utility for splitting a user specified mission into a set of parallel sub-missions.

### A. Our High-Level Mission Programming Language

We developed a high-level Mission Programming Language (MPL) [1] at the Applied Research Lab at the Pennsylvania State University for use with the Seahorse UUV [2] to allow an operator to easily specify missions without having to worry about any details for a particular UUV (similar to a programmer writing a C++ program without having to worry about any of the underlying hardware details for a particular machine's architecture). Our high-level MPL has different types of orders that can be used to specify a mission. Some of these orders include Waypoint Navigation Orders (WNOs) and Survey Orders (SOs). Examples of these orders are shown

```

{
  Start                : Waypoint_Navigation_Order
  Destination_Latitude : 40.0000 Degrees
  Destination_Longitude : -74.0000 Degrees
  Transit_Mode         : Steer_to_Point
  Transit_Depth        : 10 Meters
  Transit_Speed_In_Water : 5.0 Knots
  Use_SSS              : False

  Start                : Survey_Order
  TopLeft_Latitude     : 40.0000 Degrees
  TopLeft_Longitude   : -74.0000 Degrees
  TopRight_Latitude    : 40.0000 Degrees
  TopRight_Longitude   : -73.9500 Degrees
  BottomLeft_Latitude  : 39.9500 Degrees
  BottomLeft_Longitude : -74.0000 Degrees
  BottomRight_Latitude : -39.9500 Degrees
  BottomRight_Longitude : -73.9500 Degrees
  Survey_Depth         : 30 Meters
  Survey_Speed         : 3.5 Knots
  StartPoint          : TopLeft
}

```

Figure 1 – A sample mission involving a WNO and a SO.

in Figure 1, a sample mission for the Seahorse UUV. Here the WNO is used to specify a destination latitude and longitude from the UUV's current position. The SO is used to specify an area of the ocean floor to be surveyed or scanned. A SO could be used for different applications such as mapping the ocean floor [3], mine detection and identification [4], and exploring potential drilling locations for the oil industry [5]. Next we discuss the supporting compiler for this high-level MPL.

### B. Compiler Support for our Mission Programming Language

Each order type supported by our high-level MPL has a corresponding language specification. The language specification for each order type is a rule on how to use each particular order type. The language specifications for the WNO type and SO type are shown in Figure 2. This language has critical elements as well as optional elements. Critical elements must be specified by the user whereas optional elements may be omitted. Note that if an optional element is omitted then a default value is substituted for this element. Default values are shown in parenthesis. For example, with respect to the WNO, if the *Destination Latitude* and *Destination Longitude* are omitted, the UUV will have no way of knowing where to travel to from its current position. The MPL cannot simply assume a value for these elements. If it

Start	Waypoint_Navigation_Order
Destination_Latitude	Critical Float Deg/Rad
Destination_Longitude	Critical Float Deg/Rad
Transit_Mode	Critical Integer Steer_to_Line/Steer_to_Point
Transit_Depth	Critical Float Meters/Feet/Yards
Transit_Speed_In_Water	Critical Float mps/Knots
Use_SSS	Optional Boolean True/False(True)
Do_VBTrim	Optional Boolean True/False(True)
Start	Survey_Order
TopLeft_Latitude	Critical Float Deg/Rad
TopLeft_Longitude	Critical Float Deg/Rad
TopRight_Latitude	Critical Float Deg/Rad
TopRight_Longitude	Critical Float Deg/Rad
BottomLeft_Latitude	Critical Float Deg/Rad
BottomLeft_Longitude	Critical Float Deg/Rad
BottomRight_Latitude	Critical Float Deg/Rad
BottomRight_Longitude	Critical Float Deg/Rad
Survey_Depth	Critical Float Meters/Feet/Yards
Survey_Speed	Critical Float Mps/Knots

Figure 2 – Sample of the language specifications for both WNOs and SOs.

WNO	->	waypoint_start dest_lat dest_long transit_depth transit_speed transit_mode use_sss do_vbs_trim
waypoint_start	->	Start : Waypoint_Navigation_Order
dest_lat	->	Destination_Latitude : float length_dr
dest_long	->	Destination_Longitude : float length_dr
transit_depth	->	Transit_Depth : float length_mfy
transit_speed	->	Transit_Speed_In_Water float weight_mkk
transit_mode	->	Transit_Mode : steer_type   <No Token>
use_sss	->	Use_SSS : bool   <no token>
do_vbs_trim	->	Do_VBSTrim : bool   <no token>
steer_type	->	Steer_to_Point   Steer_to_Line
weight_mkk	->	mps   Knots
length_mfy	->	Meters   Feet   Yards
length_dr	->	Deg   Rad
bool	->	True   False
float	->	number . number

Figure 3 – The corresponding grammar for a WNO based on its corresponding language specification.

does assume a value, the default values for the *Destination Latitude* and *Destination Longitude* could be a position far outside of the operating area for the UUV. On the other hand, if the user does not specify the optional element *Use\_SSS* (which indicates whether to use the Side Scan Sonar or SSS), the MPL can assume this is to be used since there are only two options for this element, True and False.

To develop the compiler for this high-level MPL, we derived a grammar for the different order types based on the corresponding order type specifications. Once we had the grammar representation for each order type, we incorporated the grammars for each order type into a compiler for this high-level MPL. An example of the grammar for the WNO is shown in Figure 3. The grammars used in our compiler are in the style of Backus-Naur Form or BNF [6]. Note that some productions in this grammar have a *<no token>* token and other productions do not have this token. The *<no token>* token is used to represent the optional order elements as in the production for *Use\_SSS*. Other productions that do not include this *<no token>* token are the critical elements. For a detailed discussion regarding our high-level MPL and its corresponding compiler, we refer the reader to [1].

### C. A UUV Mission Partitioning Utility

Since research has been moving towards using multiple UUVs [7] in recent years, we needed a way to add support for this capability to our existing high-level MPL. We added a feature (called the *parallel region* construct) that allows a user to easily express complex missions involving multiple UUVs. An example of this construct can be seen in Figure 4. This construct allows a user to quickly and easily convert user specified missions for a single UUV into missions involving multiple UUV's by simply enclosing the preferred orders in the parallel region construct as shown in Figure 4. If a user wants to convert this mission back into a mission for a single UUV, the user simply removes the parallel region construct. The compiler has access to the available UUVs when the mission is compiled and, based on the available UUVs, the compiler can generate the corresponding set of parallel sub-missions from the user specified mission.

```

{
  ...
  Parallel_Region_Start
  {
    Start                : Survey_Order
    TopLeft_Latitude    : 40.0000 Degrees
    TopLeft_Longitude   : -74.0000 Degrees
    TopRight_Latitude   : 40.0000 Degrees
    TopRight_Longitude  : -73.9500 Degrees
    BottomLeft_Latitude : 39.9500 Degrees
    BottomLeft_Longitude : -74.0000 Degrees
    BottomRight_Latitude : -39.9500 Degrees
    BottomRight_Longitude : -73.9500 Degrees
    Survey_Depth        : 30 Meters
    Survey_Speed        : 3.5 Knots
    Start Point         : TopLeft
  }
  Parallel_Region_End
  ...
}

```

Figure 4 – Example mission using the *Parallel Region* construct

We started out very simple mission splitting algorithm. Given a user specified mission and a set of  $n$  vehicles (each one with the same capabilities) our mission splitting algorithm would divide the mission into  $n$  sub missions. For example, when the parallel mission from Figure 4 is compiled and we have four UUVs available, our mission splitting algorithm will create a set of four sub-missions depicted graphically in Figure 5. Note that this mission splitting algorithm is very simple and assumes all available vehicles have the same capability (i.e., equipped with the same set of sensors and have the same speed). One practical application for this algorithm would be mine detection using a set of identical vehicles for a specified region of the ocean floor. More details on parallel mission partitioning can be found in [11].

### III. THE GRAPHICAL MISSION SPECIFICATION AND PARTITIONING TOOL

This section discusses our graphical mission specification and partitioning tool and all of the sub components that were integrated to realize this package. We remark on the function of each sub component and explain how it interacts with the rest of the system.

#### A. The Nobeltec Visual Navigation Suite Package

To create our graphical mission specification and partitioning tool, we integrated a set of independent tools to work together. First, we used the package called Nobeltec's

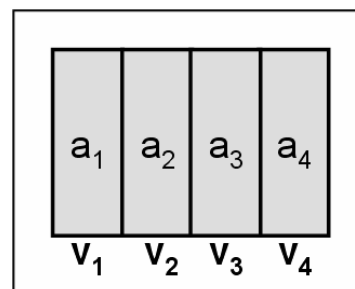


Figure 5 – The set of parallel sub-missions generated from the mission in Figure 4.

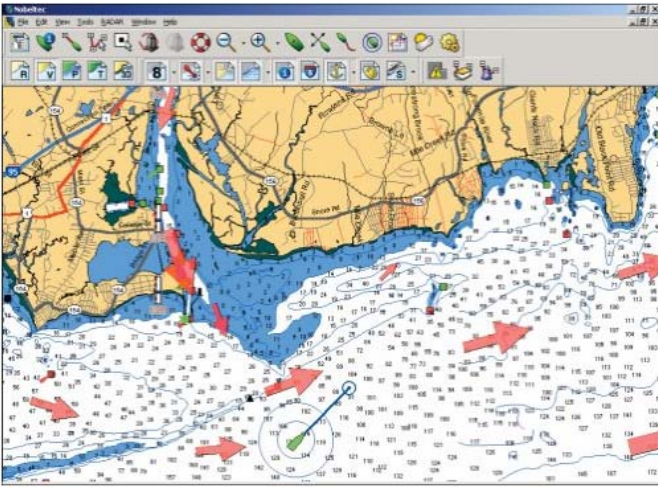


Figure 6 – Screenshot of Nobeltec’s Visual Navigation Suite maritime software package [8].

Visual Navigation Suite (VNS) [8] by Jeppesen Marine, which is a commercially available package to aid in the navigation of sea vessels. A screenshot of this package is shown in Figure 6. This package provides an operator with a graphical user interface (GUI) allowing the operator to view various locations in the ocean and any nearby coastal regions. This tool lets the operator specify waypoints to visit and areas to explore by using a simple point and click interface. An operator can easily specify a series of waypoints and areas of interest to quickly create a mission for a sea vessel.

We decided to leverage the power of this commercial package for use in creating missions for UUV’s. More specifically we wanted a tool that an operator can use to specify one or more survey areas for a group of available UUV’s. The idea was to let the operator specify regions that need to be surveyed without worrying about how the survey areas were to be split up among the group of UUV’s. Figure 7 shows a close-up of a user specified mission. Here there is a single SO represented by the rectangle and two WNOs, one to the survey area and one back to the vessel on the right side of the screen. Once the operator specifies all of the orders for a mission, this mission can then be exported from the VNS to a text file represented using the Open Navigation Format (ONF) [8], which is based on the INI-style ASCII text file format. This file format is discussed in more detail in the next sub-section.

### B. Converting from ONF to MPL

After the mission is exported to a text file represented in the ONF format, it is ready for processing by the Nobeltec to MPL (NT2MPL) translation component. This component reads the user specified mission represented in the ONF format and converts the mission orders to the corresponding order types in the MPL format. The translation is a simple one-to-one translation. For example, a WNO is represented by what is known as a *Mark* in the Nobeltec package [8]. An example of the ONF representation for a WNO is shown in Figure 8. A *Mark* includes the destination latitude and longitude and other data for the display of this *mark* in the

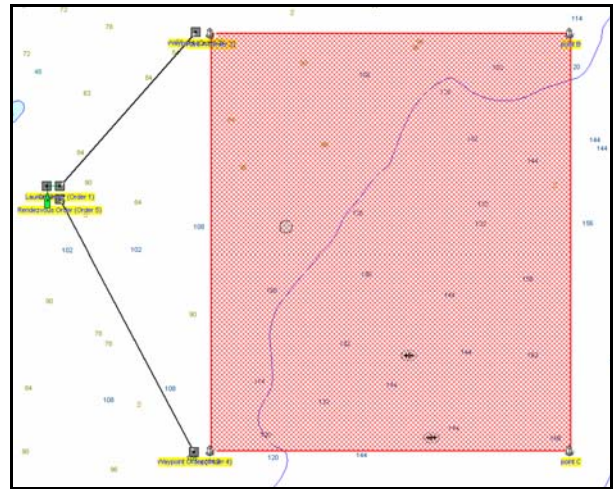


Figure 7 – Screenshot of a user specified SO in the Nobeltec Visual Navigation Suite maritime software package [8].

Nobeltec environment. Note that some data pertaining to the WNO (e.g. *Transit\_Depth*) cannot be directly represented in the ONF format (i.e., there is no concept for this type of data in Nobeltec). Fortunately, the Nobeltec [8] package provides a description field (called *Desc*) where characteristics relating to the WNO in the MPL representation can be stored. When the NT2MPL component translates a particular mission order from the ONF format to the MPL format, specific data relating to the mission order (e.g., a WNO) can be preserved and added to the corresponding MPL order type. In short, this ONF format captures all of the mission details in a text file similar to the way configuration settings are saved in the INI-style ASCII text file format used by many Windows applications. Once the ONF representation of the user specified mission is converted to its MPL equivalent, the mission is ready to be split into its corresponding set of parallel sub-missions. We discuss the mission splitting component in the next sub section.

### C. Generating the Set of Parallel Sub-Missions

After the mission represented in the ONF format has been converted into the corresponding MPL format, the mission is ready to be split into a set of parallel sub-missions. The

```
[++f10f5d8c-3c66-474d-9281-8aac70dd6c4c++]
Name = Waypoint Order (Order 4)
Type = Mark
CreateTime = 2007-03-15 21:57:16Z
Hidden = FALSE
Locked = FALSE
Desc = {{
  (Order 4)
  start_order:           Waypoint_Navigation_Order
  Scheduling_Info_Is_Timed:  False
  Transit_Mode:         Steer_to_Point
  Transit_Depth:        50 Meters
}}
Color = 0x000000
LatLon = 37 46.96640 N 074 49.56436 W
Bmpidx = 0
ArrivalCircleRadius = 0.050 nm
ShowName = TRUE
ShowDescription = FALSE
RangeCircleColor = 0xf0000
ExistsOutsideCollection = FALSE
NameDisplayMoved = TRUE
```

Figure 8 – A sample of the ONF representation for a WNO.

mission splitting component or Mission Parallelizer (MP) reads the MPL representation of the user specified mission and begins the mission splitting process. For the purposes of this mission specification and partitioning tool, we choose a very simple mission splitting algorithm. Note that under future work we discuss more complex mission splitting routines that can be used in place of this simple mission splitting algorithm. Since we are using a very simple algorithm, we simply read the dimensions of the entire survey area and divide it (as explained in Section II.C) into  $n$  equal sub areas, where  $n$  is determined by the number of vehicles available for the mission. Each sub-mission is then written to a separate file in the MPL format. Each file can then be loaded onto the mission controller of its corresponding UUV when the mission is to be executed as discussed in [1]. Once the set of parallel missions is generated from the user specified mission, these missions are ready for converting back into the ONF representation so they can be loaded back in the Nobeltec package [8] to be viewed by the operator.

#### D. Converting from MPL back to ONF

After the user specified mission is split into its corresponding set of sub-missions, this set of sub-missions must be converted back into the ONF representation to be imported back into the Nobeltec VNS package [8]. Again this translation is a one-to-one translation process. The entire set of sub-mission in the MPL format are read by the MPL to Nobeltec (MPL2NT) translation component and translates the contents of each sub-mission file back into its corresponding ONF representation. All sub-missions are then written to one Nobeltec file in the ONF format. Once this file is created it can then be imported back into Nobeltec [8] for review by the operator. Figure 9 shows the corresponding set of generated sub-missions based on the user specified mission in Nobeltec [8] as depicted in Figure 7. Here the user specified mission was split into a set of four sub-missions and placed over the original user specified mission.

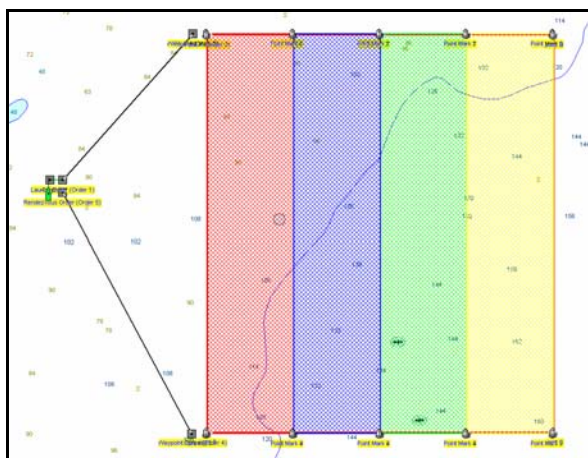


Figure 9 – Screenshot of the set of generated sub-missions after being imported back into the Nobeltec Visual Navigation Suite maritime software package [8].

#### E. Bringing it all Together, an Operator View

We have presented all of the sub-components of our graphical mission specification and partitioning tool. We now want to give a complete summary of the different steps for this tool from an operator’s standpoint. Figure 10 shows a diagram of the entire process of generating a set of parallel sub-mission from a user specified mission using our graphical mission specification and partitioning tool. Here, step 1 involves exporting the user specified mission from Nobeltec [8] to a text file containing the ONF representation of this mission. Step 2 involves the operator invoking the compiler (a command line utility) where this user-specified mission in the ONF format is provided to our compiler. Steps 3, 4, 5, and 6 all happen automatically in succession. In step 3, the ONF representation of the mission is converted into the corresponding MPL representation by the NT2MPL component. In step 4, the MPL representation of this mission is split into its corresponding set of parallel sub-missions. These missions are the ones to be loaded onto the actual vehicles as mentioned before is discussed in [1]. In step 5, the set of parallel sub-missions in the MPL representation are fed as input into the MPL2NT component, and then converted into the corresponding ONF representation, and output as one file in step 6. Step 7 involves the operator importing the set of newly-generated sub-missions back into the Nobeltec VNS [8] for review. From an operator’s standpoint, the only manual steps are step 1 (exporting the mission from Nobeltec) step 2 (invoking the compiler to generate the corresponding set of parallel sub-missions), and step 7 (importing the ONF representation of the set of parallel sub-missions back into Nobeltec [8] for inspection). Steps 3, 4, 5, and 6 are contained in the compiler and are transparent to the operator.

#### IV. SUMMARY AND FUTURE WORK

In summary, we have presented previous work regarding a high-level MPL, the supporting compiler, and the ability of this compiler to generate a set of parallel sub-missions based on a user specified mission. We also presented the integration

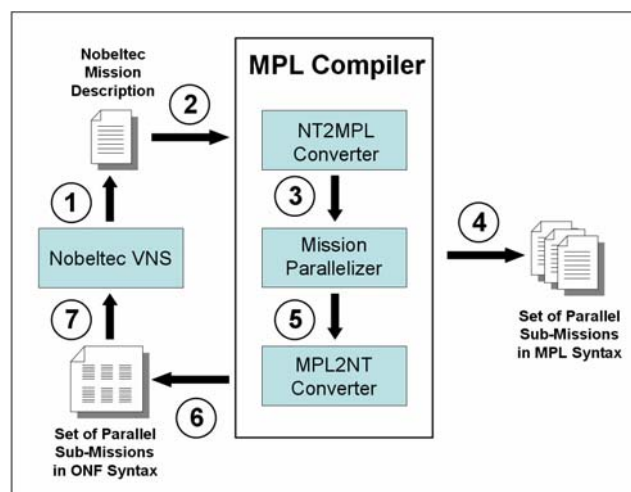


Figure 10 – High-level view of the Graphical Mission Specification and Partitioning Tool.

of our high-level MPL and compiler with a graphical interface that allows the user to create missions (both for a single UUV and multiple UUVs) in a point-and-click fashion while only writing a minimal amount of code (e.g., specifying a few order elements in the NobelTec description fields). The preliminary experiments involving this tool show much promise as a tool that can aid in the UUV mission creation and planning phases and remove from the operator much of the burden of creating missions for multiple UUVs.

Although the current mission partitioning component used in our utility is very simple, it can be replaced with a different component that can split a user-specified mission based on different criteria. We recently developed other mission splitting algorithms [10] that generate a set of parallel submissions for a group of UUV based on different objective functions such as using the minimum number of vehicles and completing a mission in the shortest period of time. Since our graphical mission specification and partitioning tool is modular, we can simply remove the current mission splitting component and replace it with one of these new splitting routines. Our goal is to devise a whole series of mission splitting algorithms using different objective functions and provide to the operator a complete UUV mission planning tool.

#### REFERENCES

- [1] G. Giger, L. Xue, S. Tangirala, and M. Kandemir, "High Level Mission Programming Support for Autonomous Underwater Vehicles," presented at AUVSI's Unmanned Systems North America, Orlando, FL, 2006.
- [2] Systems and Unmanned Vehicle, Applied Research Laboratory at the Pennsylvania State University, "Seahorse - Autonomous Underwater Vehicle (AUV)," 2006, [http://www.arl.psu.edu/capabilities/at\\_suv.html](http://www.arl.psu.edu/capabilities/at_suv.html)
- [3] R. Henthorn, D. W. Caress, H. Thomas, R. McEwen, W. J. Kirkwood, C. K. Paull, and R. Keaten, "High-Resolution Multibeam and Subbottom Surveys of Submarine Canyons, Deep-Sea Fan Channels, and Gas Seeps Using the MBARI Mapping AUV," in *OCEANS 2006*, pp. 1-6, Sept. 2006.
- [4] L. Freitag, M. Grund, C. von Alt, R. Stokey, and T. Austin, "A Shallow Water Acoustic Network for Mine Countermeasures with Autonomous Underwater Vehicles," presented at IEEE Oceans Conference, Washington D.C., Sept. 2005.
- [5] D. Bingham, T. Drake, A. Hill, and R. Lott, "The Application of Autonomous Underwater Vehicle (AUV) Technology in the Oil Industry - Vision and Experience," in *Proceedings of the International Federation of Surveyors' 22<sup>nd</sup> Congress*, Washington DC, 2002.
- [6] M. L. Scott, *Programming Language Pragmatics*. Academic Press, 2000.
- [7] E. Eberbach, C. Duarte, C. Buzzell, and G. Martel, "A Portable Language for Control of Multiple Autonomous Vehicles and Distributed Problem Solving," in *Proceedings of the 2<sup>nd</sup> International Conference on Computational Intelligence, Robotics and Autonomous Systems*, 2003, pp. 15-18.
- [8] Products, NobelTec, "NobelTec Visual Navigation Suite," 2007, [http://www.nobeltec.com/products/prod\\_suite.asp](http://www.nobeltec.com/products/prod_suite.asp)
- [9] Open Navigation Format, NobelTec, 2007, <http://www.nobeltec.com/onf/ONFBody.htm>
- [10] G. Giger, M. Kandemir, S. D. Lovell, and J. Dzielski, "Automated Mission Parallelization for a Group of UUVs," to appear in *Proceedings of the 15<sup>th</sup> International Symposium on Unmanned Untethered Submersible Technology*, Aug 2007.
- [11] G. Giger, M. Kandemir, S. D. Lovell, J. Dzielski, and S. Tangirala, "Automated Mission Parallelization for Unmanned Underwater Vehicles," to appear in *AAAI Fall Symposium on Regarding the Intelligence in Distributed Intelligent Systems*, Nov. 2007.
- [12] C. N. Duarte, C. Buzzell, G. R. Martel, D. Crimmins, R. Komerska, S. Mupparapu, S. Chappell, D. R. Blidberg, and R. Nitzel, "A Common Control Language to Support Multiple Cooperating AUVs," in *Proceedings of the 14<sup>th</sup> International Symposium on Unmanned Untethered Submersible Technology*, Durham, NH., Aug. 2005.
- [13] R. L. Stokey, L. Freitag, and M. Grund, "A Compact Control Language for AUV Acoustic Communication," in *Oceans 2005 - Europe*, vol. 2, pp. 1133-1137, June 2005.
- [14] F. F. Ingrand, R. Chatila, R. Alami, F. Robert, "PRS: A High Level Supervision and Control Language for Autonomous Mobile Robots," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 1, pp. 43-49, April 1996.
- [15] N. A. Anisimov, A. A. Kovalenko, G. V. Tarasov, A. V. Inzartsev, and A. Scherbatyuk, "A Graphical Environment for AUV Mission Programming and Verification," in *Proceedings of the 10<sup>th</sup> International Symposium on Unmanned Untethered Submersible Technology*, Durham, NH., Sept. 1997.
- [16] J. E. Manley, "Multiple AUV missions in the National Oceanic and Atmospheric Administration," in *Autonomous Underwater Vehicles, 2004 IEEE/OES*, pp., 17-18, June 2004.
- [17] S. T. Tripp, "Autonomous Underwater Vehicles (AUVs): A Look at Coast Guard Needs to Close Performance Gaps and Enhance Current Mission Performance," Coast Guard Research and Development Center, Groton, CT., Tech. Rep. ADA450814, 2006.
- [18] A. Rajala, M. O'Rourke, and D. B. Edwards, "AUVish: An Application-Based Language for Cooperating AUVs," in *OCEANS 2006*, pp.1-6, Sept. 2006.