The importance of trust between operator and AUV: Crossing the human/computer language barrier.

Nick Johnson*, Pedro Patrón† and David Lane‡
Ocean Systems Laboratory, Heriot-Watt University, Edinburgh, Scotland, EH14 4AS
Contact: *narj1@hw.ac.uk, †p.patron@hw.ac.uk, ‡d.m.lane@hw.ac.uk

Abstract—**AUVs have many advantages over traditional remote presence systems, but their autonomy introduces problems of its own. A deficit in trust may arise between the operator and an AUV which is able to modify its own plan and potentially behave in a not observably deterministic fashion. There have been some notable previous attempts to bring the plan used in an autonomous system closer to the user, but AUVs represent a relatively new technology which operates in a particular and limited domain for which this infrastructure does not yet exist. Here we present a framework to bring state and purpose of the mission as close to the user as possible. Natural language and high level graphics, which take influence from computer games in order to engender familiarity, are used as as key components of both the input and output of our suggested plan creation system. This allows the user to satisfy themselves of the validity of a plan and the competence of the system before it is run on a real AUV.**

I. Introduction

Autonomous Underwater Vehicles (AUVs) have several strong advantages over the more traditional remote presence systems. They have no constricting tether, are controlled by computers (which excel at tedious, repetitive tasks) and require very little surface support and are able run stealthily for long periods of time without needing to return to the surface. Their unmanned nature gives them the potential for a compact design, and the range of the mission is limited only by the amount of power carried by the AUV. Furthermore, whereas the loss of human life should be avoided at all costs, an AUV can have an exact fiscal cost attached to it.

Increased autonomy comes with pitfalls of its own, however. Autonomous systems require a plan in order to specify what tasks must be completed and what dependencies exist between them, and the operator must have complete trust in this plan. Furthermore, they must also trust that the autonomous system has the ability to complete it. This is made more difficult by the fact that an autonomous system must be able to adapt to circumstances, and therefore may appear to be not observably deterministic if the operator is not expecting this behaviour. This issue of trust is noted in both [1] and [2]. The first of these examines, among other things, the effect of feedback on commander trust and notes that the initial deficit is reduced through repeated use of a system, especially if adequate feedback can be provided, and the second further notes as one of its main conclusions that there is a definite need for an interface to provide just that:

> *A key aspect of the development of trust in autonomous systems will be the ability to share plans, a function for which interfaces will need to be developed.* [2]

There exist a large number of systems which allow a user to generate a plan based on their requirements, however many of these systems require the use of extremely complex notations in order to enable their complete expressivity and power. Additionally, the primary source of validation is often a direct test, either with the real system or a simulated equivalent, with any corrections to the plan being implemented either through changes to the original specifications, or a direct edit of the plan itself.

One potential solution to this problem is to provide operators with sufficient training to use these systems, both to create plans and to adapt them to deal with problems. However, since a strong foundation of logic and maths knowledge is often required to fully grasp the complete functionality of many planning systems, any training scheme runs the risk of being either extremely time consuming and expensive, or too superficial and highly inadequate.

This paper considers a potential “worst case scenario” for the user: a multi-AUV system given free reign to adapt its plan in the face of changing circumstances and presents a novel framework for the alternative approach: bringing the interface closer to the user, rather than taking the user closer to the interface. By leveraging and combining existing semantic technologies to bridge the gap between the complex notations used to generate plans and human expression itself, bringing the interface closer to the user, a system for which only a very small amount of training is required can be created. This ensures that the most appropriate communication vocabulary between the user and the system is selected, and furthermore that a coherent plan is generated. The plan can then be accurately simulated or performed by the real system. Planning technologies based on PDDL (Planning Domain Definition Language), used as the basis for planning in the Artificial Intelligence (AI) community, are used for actual plan generation, with additional systems used to attach semantics to the PDDL clauses in order to most accurately match the users requirements. The system achieves a high level interface to the planning system and operator situation awareness at all stages of the AUV mission plan creation and evolution process. A combination of visualisation technologies which use the familiarity which may be found in computer games as a key visual touchstone, together with both Natural Language Processing and Natural Language Generation are employed to achieve this.

The remainder of this paper is divided into four sections. The first gives some detail of *Previous Work* which provides background to the proposed system or directly influences it. This is followed by an overview of the *Framework* itself, and then a description of the current progress of the *Implementation*. Finally, the *Conclusion* gives a brief overview of the project and its importance.

II. Previous Work

In its *Unmanned Undersea Vehicle Master Plan* [3], the US Navy Office of Naval Research gives its road map for employing sub-sea robotics technologies. This document provides a comprehensive deconstruction of the domain itself, specifically related to combat and defence based tasks. The document is not AUV specific, as some of the technologies are intended to be employed on remotely operated systems with little to no autonomy, and thus the term UUV is employed throughout. Through a breakdown of the areas in which the US Navy intends to employ UUVs a number of strengths of unmanned systems, as well as the difficulties of the domain are identified. These strengths include:

- AUVs have the ability to operate independently for extended periods of time.
- Their unmanned nature reduces the danger to human life.
- They can operate fully submerged, with potentially low acoustic and magnetic signatures
- They can operate regardless of harsh weather and conditions.

Identified challenges of vehicles in the domain include:

- They will have a low communication bandwidth and range.
- They must face the possibility of enemy countermeasures.
- They have an increased need for reliability, as they cannot be directly supervised.

During a mission both the vehicles and environment are subject to continuous change. Events sensed by a vehicle constantly change its perception of the world. It is highly likely that there will be unexpected developments that were not considered at the beginning of the mission which might have undesired consequences. In order to achieve the previously described capabilities, high levels of autonomy are required. In the majority of modern systems, vehicles are driven by waypoint-based mission plans specified a-priori by an operator, with decision making done at the level of trajectory planning. Trajectory planning algorithms have become quite mature and efficient algorithms for continuous adaptation of vehicle trajectory are now state of the art [4]. As new applications and more complicated tasks are required for autonomous vehicles, this decision making must be extended to make changes to the mission itself. A planning algorithm may be used pre-mission to generate a plan based on a goal and domain specified by the operator in some domain description representation. However, this technology is very computational expensive, it is not distributed, and it only generates plans, it does not adapt them. As such this technology is not suitable for a direct transference to embedded platforms which typically have lower system resources. One potential solution to this problem is the recently emerged concept of on-line plan repair algorithms [5]. Unlike total re-planning systems, these techniques make use of the initial plan to produce a valid mission which has been adapted to cope with changes in conditions. This gives rise to a system with complete autonomy to change its plan; highly adaptable, but the previously mentioned worse case scenario for building user trust.

PDDL [6] is the state of the art Planning Domain Description Language used each year for the International Planning Competition[1]. Despite its power and flexibility, PDDL is not an eminently suitable method of direct input by users who have no familiarity with similar languages and concepts. As an example:

```
forall (?who - auv exists (?where
- location and (start(?who ?where)
at(?who ?where))))
```

would translate to "All AUVs are at their starting positions", however even this simple clause might be nearly incomprehensible to someone with no planning, logic or programming background. One problem many people without this background would have stems from the fact that PDDL is based on the LISP programming language and therefore uses prefix operators instead of the more usual infix, so "3 + 5" would be written in PDDL (or LISP) as "+ 3 5". This means that the structure of the language lacks familiarity; a very important concept in human computer interaction. Though not suitable for direct input, PDDL's power, flexibility and expressivity make it ideal as an intermediate step to creating a plan to satisfy the user's requirements, especially since so much work has gone into creating algorithms which plan using it.

The DARPA sponsored Pilot's Associate Program [7], was intended to create an intelligent interface for military pilots. It analysed the current state of the pilot's mission, along with real-time data and intelligently provided the pilot with whichever data were pertinent to the current mission objectives. More recently, the Rotary Pilot's Associate (RPA) Program extended this work, giving more focus to the Cockpit Information Manager [8], which dynamically decides which information should be given to the pilot on the various displays. This system performed well in simulated tests, with pilots notably willing to ignore some mistakes on the part of the system in light of the frequency at which it provided the right information and the system's perceived predictability.

Following on from his work on the RPA, Christopher Miller produced the very interesting [9]. This very short paper does not present a human interface system or any results, but rather posits the idea that any advanced user interface is effectively having a conversation with the user and making decisions on their behalf, and in this case should follow rules, or etiquette, similar to those which would be followed by a human who was

[1]See http://zeus.ing.unibs.it/ipc-5/

fulfilling the same function. He then goes on to suggest such a set of rules to accomplish this. These are (to paraphrase):

1) Make many correct conversational moves for every error made.
2) Make it very easy to override and correct your errors.
3) Know when you are wrong, the easiest way to do this is to let the human tell you, and then get out of the way.
4) Don't make the same mistake twice.
5) Don't show off. Just because you can do something, doesn't always mean you should.
6) Be able to talk explicitly about what you're doing and why.
7) Make use of multiple modalities and information exchange channels redundantly; understand the implications of your communications on all the levels on which it operates.
8) Don't assume every user is the same, be sensitive to and adapt to individual and contextual differences.
9) Be aware of what the user knows, especially if s/he knows it because you recently conveyed it.
10) Be cute only to the extent that it furthers your conversational goals.

These rules evolve the similar, but more general rules for user-centred design and human computer interaction which can respectively be found in [10] and [11], two very well respected texts. As such they provide an excellent foundation for a user-centred system which employs an intelligent user interface.

One software area which has created large advances in interface technology is computer games. Although quite easily dismissed due to having what could be considered a frivolous nature, there are two reasons why they should be considered both valid and a source of highly relevant information. The first is familiarity. The Entertainment Software Association indicates that \$7 billion worth of computer and video games, along with \$228.5 million worth of game playing units were sold in 2005 in the US alone[2]. Furthermore, research indicates that the average game player (again in the US) is 33 years old and has been playing for 12 years[3]. It is quite clear from these data that the majority of the contact some people have with computers is through computer games, and this fact is not limited to children. The second is the technology itself. The user interface is an extremely important element of any computer game. They aim to provide interactive entertainment and the interface must support this. If it does not then the game will quickly become frustrating, will not played and, more importantly to the companies which publish computer games, it will not sell.

The style of computer game whose interface is most appropriate for this project is almost certainly the "Real Time Strategy" (or RTS) genre. RTS refers to games in which the player takes control of an army, issuing instructions to single, or groups of, military units, in order to claim victory in the recreation of a battle. These games are typically viewed from

[2]See http://theesa.com/facts/sales_genre_data.php
[3]See http://theesa.com/facts/top_10_facts.php

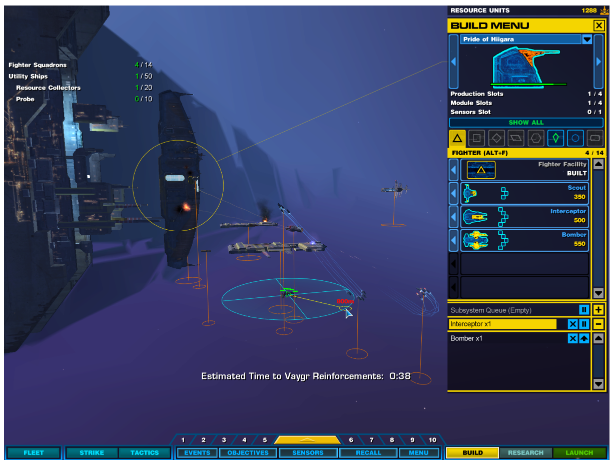

Fig. 1. The Homeworld 2 User Interface, courtesy of Sierra Interactive, an example of a high level user interface, viewing objects with a high degree of freedom. The 3D world uses the maximum amount of the window and all 3D objects are connected to a "ground plain" with a vertical line, in order to aid perspective. Additional 2D geometry is overlaid to present additional information, and linked to objects in the world when relevant. Further context sensitive information is displayed in the bottom centre and top left due to relevance.

a third person perspective, and characterised by the fact that the player has little to no control of the units beyond the high level instructions they can give (go to, attack, etc.). If we are to draw this analogy between RTS games and AUV control, then it seems plausible that we should draw a similar analogy for ROV (Remotely Operated Vehicle) control. In this case, the most obvious likenesses can be drawn to the First Person Shooter (or FPS) genre, in which the player directly controls a character and looks through their eyes as they perform a series of missions. If an operator is used to the current remotely controlled ROV (or other robotic) technology is familiar with these genres of computer games, this analogy could be used as first step to facilitate the change from ROV to AUV control.

Homeworld 2 [12] is one RTS game particularly known for its excellent interface, however this game is also notable due to the degrees of freedom involved in its gameplay. Where as most RTS games give the player control of ground units such as soldiers and tanks, in the Homeworld games the player controls spacecraft with the full six degrees of freedom, which makes it much closer to AUV control than other, more land based, RTS games. The vast amount of other data required for game-play (positions of friendly and enemy units, resources, ships which are being built etc.) is supplied to the player either intelligently by the game or on request, depending on the importance of the information. This means that the interface is never crowded and the player is always give the maximum amount of space to view the task they are currently working on. Fig.1 shows a screen shot from *Homeworld 2*. As can be seen, the entire screen displays the 3D world, with additional information being overlaid on top. A "move" command is being issued, and so the game displays tracking lines which link each object to the present reference plane to assist the user's 3D perception. To the right of the screen a dialogue

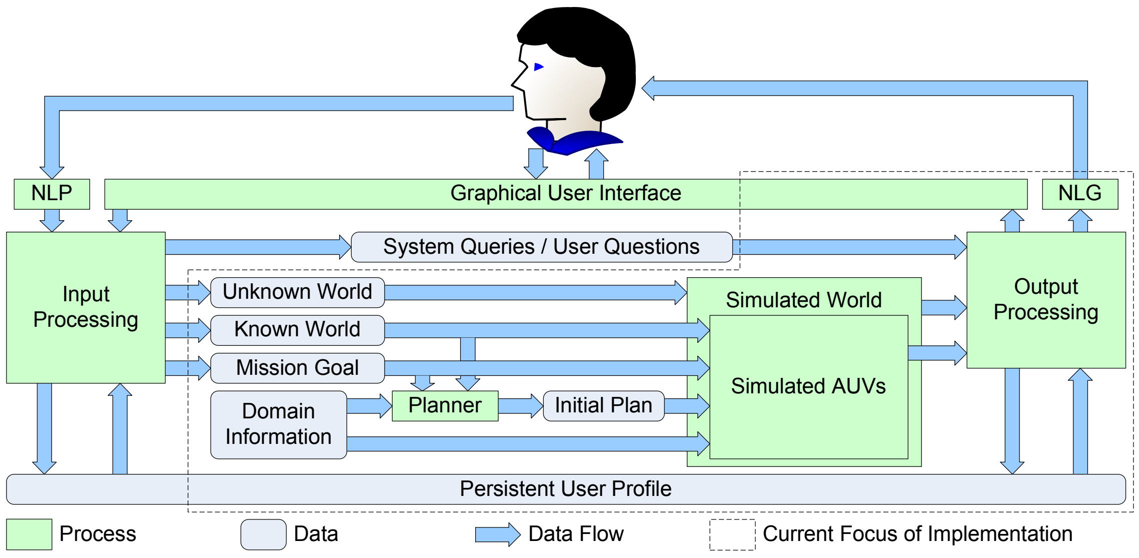

Fig. 2. Framework overview.

displays information related to one of the on screen objects and so is visibly linked to it. Towards the top left and bottom of the screen, additional information regarding current units and the status of reinforcements is displayed, as the game judges it to be pertinent to the users current status. This interface is used as a key visual touchstone in this project.

Also relevant to this study, Natural Language Processing (NLP) is the study of programming computers to input and understand natural (human) languages. Conversely, Natural Language Generation is the study of programming computers to output natural language. These two fields are discussed here only to the extent that they represent key components of the presented framework, but the various techniques for their implementation are beyond the scope of this paper, and therefore will not be discussed. An excellent overview of the field can, however, be found in [13]. The ILEX system, as presented in [14], is one example of a system which employs a NLG component. ILEX is a dynamic hypertext system, which generates and displays information in a format that is similar to that found on many web pages. Where the system differs from a standard hypertext system is that it keeps track of which specifics the user knows, and then simplifies these data when they reoccur, as suggested in Miller's ninth rule.

III. Framework

Fig.2 gives an overview of the framework as a whole, showing the most important processes and data representations together with the data flow between them. Fig.3 represents the chronology of this data flow as it forms a coherent system. These diagrams are described in more detail in the following two subsections.

A. Framework Components

The *NLP* (Natural Language Processing) component processes human language input and passes it to the *Input Processing* component in a more computationally understandable form.

The *Graphical User Interface* serves the dual purpose of providing visual output to the user, as supplied by the *Output Processing* component; and accepting user input in the form of mouse clicks of on screen controls and elements, and passing them to the *Input Processing* component.

The *Input Processing* component takes the user input, passes it to the *Output Processing* component if it is a question about the current system state or uses it to formulate the *Unknown World*, *Known World* and *Mission Goal* representations required for simulation. In the event that there is an ambiguity or contradiction in the current data, this component passes a query to the user (via *Output Processing*) to resolve this.

As previously mentioned, *System Queries* and *User Questions* are passed from the *Input Processing* component to the *Output Processing* component in order to answer the users questions and prompt the user to clarify any ambiguities or contradictions the system has found, as suggested by Miller's third rule.

The *Unknown World* represents the parts of the simulated world the user does not wish the planning system to know about at the start of the mission. For example, in the case of a Mine Counter-Measures (MCM) mission, the this would refer to the position of the simulated mines, as these are the things the user is testing the simulated system's reaction to and these data would not be available to the planner at the start of the real mission.

The *Known World* refers to the initial state of the world used as the starting point for the planning system, including all the AUVs involved in a mission, their capabilities, any objects in the world which are known, and any specific properties they may have.

The *Mission Goal* is the goal used by the planning system, both as a world state to work towards and as a benchmark for

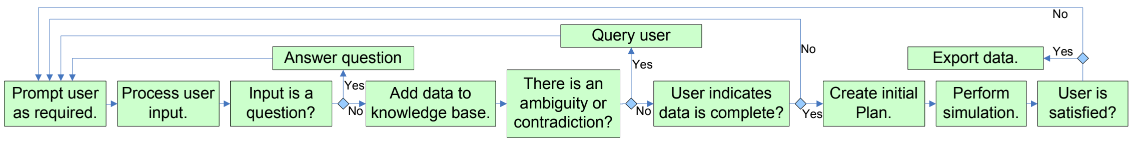

Fig. 3. Sequence of data migration within the framework.

the completion of the mission.

The *Domain Information* directly corresponds to the domain employed in the specific representation of a planning problem in PDDL. It contains the types which can exist in the world, the properties which can be assigned to these types and the actions which can be applied to the world by the planner. Also coupled with this information is any data required to convert the PDDL constructs and symbolic data to natural language for output to the user.

The *Planner* component of this framework is a full planner which is used to create the *Initial Plan* for the AUVs to follow at the start of the mission, and may differ to the more stripped down planning engine which might for re-planning on board the individual AUVs.

The *Initial Plan* is the plan which the AUVs are following at the beginning of the mission, but which may be changed by on-board re-planning systems. Again using the example of an MCM mission, the initial plan would most likely be that an AUV with a search capability should search the specified area. This plan would then be augmented should the search AUV discover any potential mines.

The *Unknown World* is used to create a *Simulated World* and the *Known World*, *Mission Goal* and *Domain Information* used to create a set of *Simulated AUVs* within it. These are then employed to perform a simulation of the mission.

The *Output Processing* component of the framework attempts to answer any user questions about the current plan and system state, and also prompts the user to resolve any ambiguities or contradiction detected by the *Input Processing* component. Its main job, however, is to keep the user situated as to the current progress of the simulation in as much detail as is required, and moreover to provide these data in a structured manner, so the user is not bombarded by unnecessary and potentially confusing information, but also does not lack relevant data when it is required.

The *Natural Language Generation* takes the output of the *Output Processing* component and converts in to (either spoken or written) natural language.

The *Persistent User Profile* component purpose is twofold. Firstly it is used to maintain a store of all the information the system has given to the user, so the *Output Processing* component can avoid repeating redundant data, as suggested in Miller's ninth rule. Secondly, it stores data relating to the user's preferences and how any previous contradictions or ambiguities where resolved, in order to aid the *Input Processing* component's inference of the user's meaning, moving towards the ideas suggested by Miller's fourth rule. The *Persistent User Profile* as a whole supports Miller's eighth rule, allowing the

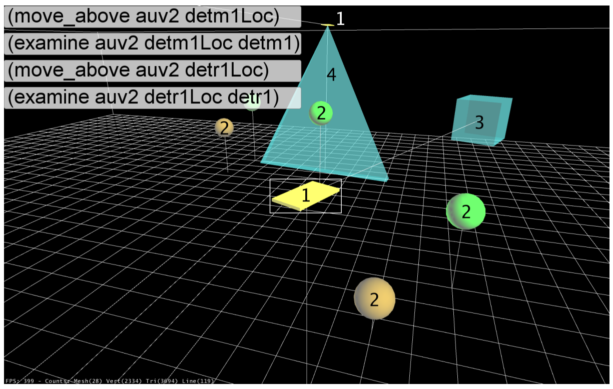
(move_above auv2 detm1Loc)
(examine auv2 detm1Loc detm1)
(move_above auv2 detr1Loc)
(examine auv2 detr1Loc detr1)

Fig. 4. The graphical interface to the current system implementation. Yellow cuboids (1) represent AUVs and various coloured spheres (2) other objects in the world. All are linked by white vertical lines to a "ground plain" to aid perspective, and AUVs are further linked to a light blue box (4) representing their current way-point. Simulated sensor geometry is represented by a light blue sector (4). The task list of the currently selected AUV (highlighted by a thick white box) is overlaid at the top left of the display.

system to adjust itself to different users.

B. Data Flow Sequence

Fig.3 presents the sequence of the data migration within the system. At all stages in this pipeline it should be assumed, even when not explicitly stated, that the *Output Processing* module is passing all relevant information to the user regarding the current system state, and the *Persistent User Profile* is recording both the information being presented to the user and the user's responses to the system queries, as suggested by Miller's sixth rule.

As can be seen, the process is cyclical, enabling the operator to both adjust to the plan for correctness and test the system's reaction to different circumstances. It is noted in [1] that the initial gulf in trust between the user and an autonomous gains can be reduced through repeated use of the system, and although these are not tests performed with the real system, it is thought that these repeated simulated trials will act in lieu of the real system for the purpose of gaining this trust.

IV. Implementation

The current focus of the implementation of the framework is indicated by the dotted line in Fig.2. The system is being implemented in the Java programming language, chosen due familiarity, development speed and the wealth of relevant tools currently available. These tools include the *Java Monkey Engine* [15], a high performance game engine which is used to create the 2D and 3D graphics used in the GUI and *FreeTTS* [16] a free voice synthesiser which is used to create spoken output. Currently, the *Unknown World*, *Known World*, *Mission Goal* and *Domain Information* are read in from XML and used to generate the objects in the Simulated World and the PDDL terms needed for the Planning component and Simulated AUVs. The Simulated World and Simulated AUVs are currently near completion (using a simulator based on the RAUVER AUV citerauver) and a prototype planning system, based on the Metric-FF planning algorithm [18] is in place. The next phase of implementation will seek to enable the NLG component and add the required data to the *Domain Information* (which is represented in XML for easy extensibility), and then move on to the *Output Processing* component in order to correctly structure the generated information, as well as making incremental upgrades to the *Planning* and *Simulation* components as required and building the relevant parts of the *Persistent User Profile*, used the model employed in the ILEX system [14] as a base.

A screenshot of the current GUI implementation is shown in Fig.4. The influence of the previously described *Homeworld 2* user interface should be immediately apparent. Two AUVs can be seen in the screen shot, both represented by simple yellow cuboids (1) which mimic the dimensions of the buoyancy used in the RAUVER vehicle, each of which is linked to a spacial grid on the "floor" of the virtual world by a vertical white line in order to enhance 3D perception, as are other objects in the world, which are displayed as either green or brown spheres (2) in this example. Each AUV is also linked to its current way-point, displayed a a light blue box (3) whose dimensions reflect the tolerances of the way-point, by another line and any simulated sensor geometry is shown as a light blue sector (4). The currently selected AUV is indicated by a thick white box and has its current list of tasks overlaid on the view at the top left of the display. Currently these are displayed purely in PDDL terms, as the NLP component has not been implemented. Presently, the system allows a mission to be simulated at a speed specified by the user, with the viewpoint also fully user controllable.

The implementation is presently focussed in this area as a trusted and proven output system should be in place before the input system can be reliably developed. Without this initial segregation it would be difficult to test the input system, as there would be no reliable output against which the user could measure the accuracy of the system's processing of their input.

V. Conclusion

In this paper we have talked about the potential usefulness of AUV technology, but also attempted to highlight the problem of trust which emerges when an autonomous system is given license to adapt its own plan. This problem is compounded by the fact that there is currently no concrete way of sharing a plan between a human and an autonomous system, as a language barrier is currently in place. To this end, we have presented a framework for user-centred plan creation intended to scale this barrier and follow the advanced user interface rules suggested in [9]. By using high level, partially computer game inspired, graphics we create a sense of familiarity for the user, and then further extend this by employing Natural Language Processing and Natural Language Generation to bring the plan description into the user's own language. Additionally, a cyclical process is employed to allow the user to adapt the plan to their satisfaction and test multiple scenarios, increasing the user's experience with the system and building trust through this. Furthermore we have given details about the current progress of its implementation, and how this will proceed.

Acknowledgment

This work was sponsored by the Engineering and Physical Sciences Research coucil. The Authors would also like to thank all the staff of both the OSL and SeeByte for their help, advice and support.

References

[1] V. Chapman, S. Appleyard, and L. Thomas, "Commander trust in autonomous systems: the role of implicit instructions and system feedback," in *Proceedings of the 1st Annual SEAS DTC Technical Conference*. QinetiQ Centre for Human Sciences, July 2006.

[2] J. L. Hinton, M. Williams, and B. Kirby, "Human centred perspecivies on mission planning for autonomous systems," in *Proceedings of the 1st Annual SEAS DTC Technical Conference*. BAE Systems Advanced Technology Centre, July 2006.

[3] "The navy unmanned undersea vehicle (UUV) master plan," Office of Naval Research, Tech. Rep., July 2004.

[4] C. Pêtrès, Y. Pailhas, P. Patrón, Y. Petillot, J. Evans, and D. Lane, "Path planning for autonomous underwater vehicles," *IEEE Transactions on Robotics (Accepted for publication)*, April 2007.

[5] R. van der Krogt, "Plan repair in single-agent and multi-agent systems," Ph.D. dissertation, The Netherlands TRAIL Research School, 2005.

[6] M. Fox and D. Long, "PDDL 2.1: An extension to PDDL for expressing temperal planning domains," *Journal of Artificial Intelligence Research*, vol. 20, pp. 61–124, December 2003.

[7] S. Banks and C. Lizza, "Pilot's associate: a cooperative, knowledge-based system application," *IEEE Expert*, vol. 6, no. 3, pp. 18 – 29, June 1991.

[8] C. A. Miller and M. D. Hannen, "User acceptance of an intelligent user interface: a rotorcraft pilot's associate example," in *Proceedings of the 4th international conference on Intelligent user interfaces*. ACM Press, New York, NY, USA, 1998, pp. 109 – 116. [Online]. Available: http://portal.acm.org/citation.cfm?id=291100

[9] C. A. Miller, "Rules of etiquette, or how a mannerly AUI should comport itself to gain social acceptance and be perceived as gracious and well-behaved in polite society." in *Working Notes of the AAAI-Spring Symposium on Adaptive User Interfaces*, March 2000, pp. 80–81.

[10] D. A. Norman, *The Design of Everyday Things*, 3rd ed. The MIT Press, October 1998.

[11] A. Dix, J. Finlay, G. D. Abowd, and R. Beale, *Human Computer Interaction*, 2nd ed. Prentice Hall, September 2003.

[12] *Homeworld 2*, Computer Game, Sierra Entertainment Incorperated, 2003.

[13] D. Jurafsky and J. H. Martin, *Speech and Language Processing*. Prentice Hall, 2000.

[14] M. O'Donnell, C. Mellish, J. Oberlander, and A. Knott, "ILEX: an architecture for a dynamic hypertext generation system," *Natural Language Engineering*, vol. 7, no. 3, pp. 225–250, September 2001.

[15] "Java Monkey Engine." [Online]. Available: http://www.jmonkeyengine.com/

[16] "FreeTTS." [Online]. Available: http://freetts.sourceforge.net/docs/index.php

[17] "RAUVER: Hover-capable autonomous underwater vehicle," Online, retreived from: http://www.ece.eps.hw.ed.ac.uk/research/oceans/projects/rauver/oceansweb/rauver/rauver_main_mk2.htm.

[18] J. Hoffmann, "The Metric-FF planning system: Translating 'ignoring delete lists' to numeric state variables," *Journal of Artificial Intelligence Research*, vol. 20, pp. 291–341, December 2003. [Online]. Available: citeseer.ist.psu.edu/article/hoffmann03metricff.html