Automatic Fault Detection and Execution Monitoring for AUV Missions

Juhan Ernits, Richard Dearden
School of Computer Science
University of Birmingham
Birmingham, UK
Email: {ernitsj,rwd}@cs.bham.ac.uk

Miles Pebody
National Oceanography Centre
University of Southampton
Southampton, UK
Email: miles.pebody@noc.soton.ac.uk

***Abstract*—Advanced AUV's, particularly those capable of long duration missions, need increasing amounts of autonomy in order to carry out sophisticated missions without requiring constant support from a ship. One important aspect of this autonomy is fault detection and execution monitoring. In this paper we describe the application of the Livingstone 2 diagnosis system on Autosub 6000 and examine in particular the issue of ensuring the mission is being executed correctly.**

Most AUV operations require a fast turnaround between retrieving the vehicle and redeploying it, typically constrained only by the time required to charge batteries. In many circumstances missions are planned very quickly so an automatic way to monitor mission execution is required. We describe an approach to automatically generate diagnosis models that correspond to the mission instructions which can then be combined with pre-built models of the physical components of the AUV to improve fault detection and monitor execution. We show that by incorporating this program model we are able to diagnose faults that were previously undiagnosable, and demonstrate that an actual fault that occurred on Autosub 6000 can successfully be detected.

I. Introduction

As AUVs become more common, and in particular as missions become longer and take place in less well known environments, improving the reliability of the vehicle becomes more and more important. The greatest gains will probably be made with more reliable hardware and control systems, and in designing missions to be safe [1] but we believe that there is also a place for on-board diagnosis of these vehicles to detect faults when they occur and take appropriate actions to mitigate their effects. The environment these vehicles operate in, including extreme pressure, the corrosive effects of seawater, and the risk of damage due to waves while on the surface and collisions during launch and later, is too harsh to expect even the best-designed components to operate flawlessly over repeated missions.

Autosub 6000 [2] is an autonomous underwater vehicle (AUV) designed to operate for up to 48 hours at a time at depths of up to 6000m to collect science data from the deep ocean. During more than 400 previous scientific missions, Autosub 6000 and its predecessors have suffered both near losses and one actual loss. In two cases the AUV has been recovered with a remotely operated underwater vehicle at significant expense. In one case the Autosub2 AUV was permantly lost 17Km under the 200m thick Fimbul Ice Shelf in the Antarctic. There are numerous cases of missions that have had to be aborted but where recovery was possible by the operations team and the attending support ship. Based on the experience of operating the Autosub AUVs a project to apply automated diagnosis and recovery methods for Autosub 6000 was initiated with the primary focus being on the detection of faults that may result in collisions with the seabed. Collision with the seabed is undesirable because it has been demonstrated to have been one of the primary causes of vehicle loss. Section II contains a more detailed description of Autosub as well as a categorisation of the faults that have occurred during previous missions.

The potential benefits of fault detection on an AUV are illustrated by the work of Griffiths and Brito [1] on predicting the risk of loss of a vehicle. Their approach uses a Bayesian network (BN) to predict loss likelihood for a mission under sea ice. The BN probabilities are based on historical data of failures of individual components along with expert judgement of how likely particular failures are to cause loss of vehicle under different mission conditions. By inputing the observed values of variables such as percentage ice cover, ice thickness, etc., these are combined with the historical and expert data to compute an estimate of the probability that the planned mission will lead to the loss of the vehicle. This approach has been used during a recent deployment of Autosub 3 to Antarctica and the missions were then revised to reduce the risk of loss to an acceptable level. On-board fault diagnosis is intended to alter this probability of loss in a different way, by reducing the chance that a particular fault leads to loss of vehicle. As wse said above, a third way to reduce the probability of loss is to engineer more reliable components, thus reducing the probability of faults in the first place. All three of these approaches are valuable in extending the kinds of missions AUVs can perform.

Automatic fault diagnosis has three useful roles to perform in systems such as Autosub 6000. Firstly it allows the system to mitigate the effects of failures by identifying them—in some cases before they cause any serious problems—and choosing appropriate responses. Secondly, a diagnosis system

This work was funded by the UK Natural Environment Research Council as part of grant NE/F01256X/1, Automated Diagnosis for Fault Detection, Identification and Recovery in Autosub 6000.

based on telemetry can aid mission controllers in identifying unexpected behaviour and allow them to take appropriate responses. Finally, on-board diagnosis can reduce turnaround times between missions by pinpointing the subsystems that need to be replaced or at least examined to ensure they are working nominally.

The approach we are taking is to use the Livingstone 2 (L2) diagnosis engine on Autosub. L2 is a discrete, model-based diagnosis system that is compositional, allowing models of individual components to be plugged together relatively easy to build larger models. As L2 has been deployed on-board an earth observing satellite [3] it is already designed to be lightweight in terms of computational requirements and has been engineered to the standards expected for flight software. Thus it seems a natural choice to apply to AUVs, which in many ways (computational restrictions, control architectures, etc.) resemble spacecraft. L2 is described in more detail in Section III, and the diagnosis model is described in Section IV.

As well as monitoring the health of the hardware components of the AUV, we are also interested in monitoring the execution of the mission. For Autosub, the mission is defined in a file called the *mission script* which is uploaded to the vehicle at the start of every mission. To monitor the execution of this, and also because the mission script contains information about the behaviour of the vehicle that isn't available in the logs, we build a diagnosis model of the mission script automatically from the script itself. At the same time we are building this, we can also perform automatic sanity checks on the script.

In Section V we discuss the use of diagnosis both on-board the vehicle, but also off-board using telemetry transmitted to the support ship acoustically. This has the potential to be quite useful for the AUV's operators, particularly as mission script monitoring is incorporated in this. Since the off-board diagnosis system only has access to a limited subset of variables, the diagnoses that can be made off-board are quite restricted compared with on-board. However, some critical faults can be identified, albeit with a significant delay compared with on-board detection. We report our initial results, but on- and off-board in Section VI.

A. Related Work

There has been a long history of work on fault detection and fault-tolerant control for AUVs. As with most vehicles, Autosub 6000 and its predecessors have the simplest of these, an emergency abort system [4] that is triggered by a variety of events on the vehicle such as critical subsystems being unresponsive, or the vehicle exceeding its maximum depth limit.

A survey of fault detection and fault tolerance schemes for AUVs and ROVs has been presented in [5]. Many of these are model-based approaches, but tend to be only for restricted subsystems, for example the actuators in [6]. The models in that case are continuous, rather than the discrete ones L2 uses, and faults consist of changes to the parameters of the system. The relationship between these kinds of approaches and the one we take is that L2 provides system-level fault detection, rather than component-level. In fact, the output of component fault monitors such as those described in [6] may very well be useful inputs to L2. We expand on this idea in Section VII where we discuss integrating hybrid system-based models with the discrete ones of L2.

II. Analysis of faults of Autosub 6000

Autosub 6000's architecture consists of a network of components including science instruments, control surfaces and motors all controlled by a single mission control component that executes the mission script [7] and distributes the commands in it to the various components. Sensor readings from the various components are transmitted over the network to mission control and also to the logger, which is a PC that stores the data and supplies it to the diagnosis engine and makes it available for acoustic telemetry to the surface.

The vehicle has four actuators, the motor, the releases for the abort weights, the rudder and the stern-plane.Even without releasing the abort weights, the vehicle is still positively buoyant in normal operation. The vehicle is powered by up to 12 custom lithium polymer batteries that can provide power for ranges of up to 550 km and mission durations in excess of 85h [8]. The power subsystem splits the power distribution up into critical (motor) and science instrument (hotel) loads. Actuators and sensors that contribute to the navigation of the vehicle are connected to the critical power circuit and science equipment to a separate, non-critical circuit.

A. Failures and faults in previous missions

Autosub 6000 has to date completed 36 missions, and its predecessors Autosub 1, Autosub 2 and Autosub 3 together more than 430 missions. Thus we are in the privileged position of having access to very large quantities of log data, both nominal and faulty, from the vehicle. This means that rather than testing our fault detection system on a simulator, we have real logged data available for evaluation purposes. The list of failures we have so far observed appears in Table I. This is based on a complete analysis of Autosub 6000's missions, but only a partial analysis of the others. The large number of stern plane actuator faults as compared with other subsystems may partially be a result of the critical nature of the stern plane, which sees even slight changes in behaviour treated very seriously. Autosub 3 and its predecessors used a very different battery technology than Autosub 6000, so we have made no effort to track problems in this area on the older vehicles. The large number of battery problems are probably attributable firstly to the fact that there are several betteries on-board, but also to the novel battery technology used on Autosub 6000.

It should also be noted that the Autosub engineering team have been constantly improving the designs of the AUVs. For example, after the loss of Autosub 2 the critical and non-critical power systems were separated during the design of Autosub 3, and this has continued on its successors.

TABLE I

OBSERVED FAILURES IN AUTOSUB 6000 AND ITS PREDECESSORS. THE 'OCCURRENCES ELSEWHERE' COLUMN REFERS TO FAULTS ON PREDECESSORS OF AUTOSUB 6000. WE HAVE NOT YET ANALYSED ALL THE FAULT DATA FROM THESE MISSIONS; A DASH IN THIS COLUMN INDICATES THAT WE HAVE NOT FOUND ANY EXAMPLES SO FAR.

Component	Failure	Occurrences on Autosub 6000	Occurrences elsewhere
Stern plane actuator	Not responding to commands	1	-
	Actuator feedback potentiometer failed	1	Yes
	Intermittent drive faults	1	-
	Actuator stuck	no	several
	Actuator offset from zero	no	several
	Leak in actuator vessel	1	-
Rudder actuator	Failed	no	1
Propulsion system	Gradual loss of propellor speed	no	2
	Entanglements (mostly with recovery lines)	no	several
Emergency Abort	Failure to abort	no	One found pre-dive
Sensors	GPS lost initialisation	1	-
	GPS Antenna failed	7	-
	Poor navigational accuracy	2	-
	Collision avoidance software fault	1	No collision avoidance
Acoustic modem	Reduced sensitivity	2	-
Argos and WIFI links	Antenna failure	2	-
Batteries	Not switching on	several	Different battery type
	Not sourcing current	2	
	Failed to charge	1	
	Discharge fuses blown	1	
Other power system	Short due to cable failure	several	-
	Failed power switch	1	-
Science instruments	No sonar data recorded	1	1
	No CTD data recorded	No	At least one
Recovery lines	Lines entangles in propellor	No	Several
	Lines dragging in water, prevent diving	2	Several

1) Human errors in mission scripts: In addition to these hardware failures, there have also been problems due to human error. The AUV is controlled by a control node that executes *mission scripts*. Mission scripts are written by human operators on board the support vessel. It was identified by the Autosub engineering team, that typographical errors and configuration problems were introduced by human operators during the deployment of the vehicles while preparing the missions and configuration. Given the often rough circumstances at sea, and the quick turnaround cycle of deployment, it was agreed that additional means to detect potential issues with mission scripts were useful.

While the basic concepts of the architecture of Autosub 1, 2, 3 and Autosub 6000 are similar, there are important differences in technical details, so below we will concentrate on faults and modelling of Autosub 6000.

2) Collisions and other unknowns: There have been a small number of collisions involving the Autosub AUVs. An example of such behaviour occurred on Autosub 3 during the Pine Island Glacier Ice Shelf expedition in Western Antarctic [9]. Autosub 6000 has since been fitted with a forward-looking collision avoidance sonar, and has successfully responded to potential collisions by climbing or where necessary turning the vehicle away from the obstacle.

It is not known precisely what caused the fatal failure of Autosub 2 under Fimbul Ice Shelf in the Antarctic, but a number of hypotheses are listed in [10]. The most likely candidates for failure determined by a panel of experts in the order of likelyhood are Open circuit hardware failures, network failure, leak, short circuit hardware failure and loss of connectivity.

III. MODEL-BASED DIAGNOSIS WITH LIVINGSTONE 2

Traditionally, automatic diagnosis has been performed using expert systems. Approaches such as rule-based systems use knowledge from human experts along with data about symptoms to perform diagnosis. However, these approaches rely on the quality and completeness of the expert data. Fault diagnosis for systems such as Autosub feature events (faults) that are rare and complex interactions between subsystems. Thus this expert data is hard to collect and the rules that must be incorporated in the system become increasingly complex.

Model-based diagnosis (MBD) [11] is an alternative to expert systems that has been successfully applied in a number of domains. This approach uses a model of the physical behaviour of the system. Diagnosis proceeds by comparing the predicted system behaviour according to the model with the observed system behaviour and using various kinds of inference to explain any discrepancies. A number of systems have been developed based on this approach, including GDE [11], Livingstone [12], Hyde [13] and Lydia [14]. All of these are broadly similar in capability, although Hyde, for example, allows a richer set of models, including hybrid discrete-continuous system models. The system we are using is Livingstone 2, due to the fact that it is freely available and that it has already been used in similar applications.

Livingstone 2 (L2) [12], [15] is a MBD system that uses a constraint solver to perform consistency checks between the

predicted and observed system behaviours. Livingstone has been successfully used on two spacecraft [16], [3] as well as a number of testbed systems. As with most MBD systems, L2 models are split into models of individual components which are then composed to build an overall system model. This has the twin advantages of increasing re-use of models since components are often common between systems, and allowing the effects of multiple faults to be modelled without explicitly enumerating all possible sets of faults—the effects of multiple faults naturally arise from the effects of single faults and the connections between the components in the model. To achieve this *compositionality*, L2 allows only discrete variables within components. Components are connected together to build a system by specifying constraints on the values of these variables. For example, a battery component may contain a variable CURRENTOUT that represents the current being produced by the battery, with possible values ZERO, LOW, NOMINAL and HIGH. Similarly, the motor component may contain a CURRENTIN variable, and the system is constructed by declaring constraints such as BATTERY.CURRENTOUT = MOTOR.CURRENTIN.

As well as containing variables, components also have internal state, usually referred to in diagnosis as *modes* of operation, some of which may be nominal and the rest fault modes. For example, our simple battery component may include ON, OFF, FLAT and other fault modes. Changes in state are governed by transitions between modes, some triggered by commands such as SWITCHON, which causes a transition to the ON mode if the battery is currently in the OFF mode, and others triggered by faults. The constraints on variables may change depending on the mode, so for example in the OFF mode, CURRENTOUT = 0 while in ON mode, CURRENTOUT = NOMINAL. Thus, a component model can be thought of as a finite state machine with constraint sets for each state.

In normal operation, L2 intercepts all commands sent to the system and uses them to update the modes of components in its model. The combination of modes is used to predict the behaviour of the system, and this is compared with the observations from the real system. This is done by adding constraints to the system corresponding to the values of the observed system variables. As long as these are consistent with the current set of constraints governing the system, L2 reports that the system is nominal, and nothing else happens. In the event that the observations lead to an inconsistency with the constraints in the model, L2 then searches for a transition or transitions that the model could have taken to leave it in a state consistent with the observations. This search is done using an algorithm called *conflict-directed best-first search*. Essentially, the variables that led to the inconsistency are used to identify the components that could be involved, and the transitions in the finite state machines for those components are searched to find modes consistent with the observations. If a transition is found that leads to a mode consistent with the observations, then this is a candidate hypothesis that can explain the fault. L2 allows multiple candidates to be maintained to represent the fact that there may be multiple explanations for a particular symptom.

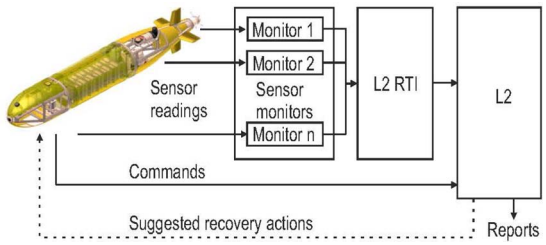

Fig. 1. Architecture of the real-time diagnosis system with Livingstone 2.

All the above assumes that the observations of the system are discrete so they can work with discrete models and constraints. In practice, almost all the observations we get from real systems are continuous, so these must somehow be converted into discrete values, a task performed by what are called *monitors*. This is the great weakness of consistency-based MBD systems such as L2 as building the monitors is rarely a trivial problem as thresholds have to be set, smoothing may need to be applied and short-term fluctuations known as *transients*, particularly near thresholds, need to be dealt with. In some cases, the need to discretise may prevent MBD from working at all. The key point is to make the monitors as simple as possible—if determining the discrete value of some variable requires examining the values of four other variables, then effectively you are performing the diagnosis task in the monitor.

The overall interface between the physical system and L2 is shown in Figure 1. Sensor data from the system is run through the monitors to generate discrete observations, and these, together with commands, are passed into L2 through the real-time interface of L2 [20]. L2 then checks for consistency and updates its internal model (either because of commands or to explain an inconsistency) to one consistent with the system.

A. Application of L2 to Autosub

In Section II we looked at what has gone wrong in previous missions. Before using L2 to diagnose Autosub, we used this data to identify critical faults that put the system at risk. For example, the on-switch failure from Mission 15,while it led to an abort and the premature end of the mission, did not put the vehicle at risk. On the other hand, the stern-plane actuator failure in Mission 12, as well as other similar failures on Autosub 3 directly endangered the vehicle by potentially leading to collision with the seabed. Thus, we identified depth-control as a priority subsystem for diagnosis. In addition, since depth control mostly consists of the vehicle being sent depth demands and trying to meet them, this seems like a relatively good subsystem to model in L2 as the difference between the demanded and actual vehicle depth, and the change in this difference can be used to model vehicle behaviour to a useful fidelity. The second subsystem we chose to model initially was the batteries, largely because a significant number of faults were observed. In addition, we thought that this subsystem

would be challenging to model and would be useful to evaluate the relative merits of a consistency-based approach against a more statistical based approach to diagnosis using particle filters [17], [18].

B. Mitigations of faults in Autosub 6000

Detecting faults doesn't actually increase the reliability and robustness of a system. This can only be achieved by actually taking action based on the diagnoses. L2 achieves this in the same way as it identifies faults: it searches for commanded transitions from the fault mode it finds itself in back into a nominal operating mode. On systems with redundancy built in, such as rocket fuel systems, this typically involves opening and closing valves to allow the redundant pathway to supply fuel to the motor rather than using the faulty one. On Autosub, there is very little redundancy so for many faults there is nothing that can be done to achieve nominal operation, but rather the challenge is to ensure the safety of the vehicle, so the correct mitigation if there is any risk of a seabed collision is to drop the abort weights and return to the surface. Of course in an overhead environment such as operations under an ice sheet, this may not be a safe mitigation either, and more sophisticated strategies that may be beyond the capabilities of L2 may be required.

At present, we do none of this on Autosub 6000 for a variety of reasons. Firstly because as we have said, for many faults there are no suitable mitigations. Secondly, there is the matter of trust. No operator of the vehicle is going to allow an AI program running on-board to take control of the vehicle until they have seen that program running repeatedly and performing correctly. Thus we anticipate a long process of proving the correctness of diagnosis and the reasonableness of the mitigations suggested before L2 would ever be allowed to put a mitigation procedure for a fault into action on the vehicle.

In [16] the main means of mitigation of faults was to reset the failed subsystem. In principle such mitigations can be considered for science instruments on board Autosub6000. For example, in the case the EM2000 swath sonar stops working, it could be passed to the L2 model of the system and, depending on the progress of the mission, a power cycle be performed.

Apart from the science instruments, the lack of redundancy on Autosub means that the only possible mitigation for almost all faults is to abort the mission. In cases where critical actuators or sensors fail, this should be done by dropping the abort weights. In other cases it may be possible to safely ascend to the surface under power. The only two exceptions to this are failures in the inertial navigation system (INS) and the water speed sensor. In the case of the INS, its function can be approximated using compass and ADCP, or to use range only positioning [19]. In the case of the water sensor, its function can be approximated by propellor RPM. In both cases this means that a mission can in principle continue with reduced accuracy of measurement.

Given the relatively narrow selection of possible mitigations, it should be pointed out that false diagnoses are especially undesirable because false positives lose valuable mission and support vessel time and false negatives put the vehicle at risk.

IV. Diagnosis model of Autosub 6000

To illustrate the consistency based diagnosis approach we present a simplified diagnosis model of a subset of Autosub 6000 and discuss key aspects of such modelling. The first area we will examine is the depth control system as we have fault data from it (Autosub 6000's Mission 12), and it is a critical component for vehicle safety.

Autosub 6000 has several depth control modes where different network variables become significant. This point is illustrated in Figure 2. The graph on the left shows the nominal behaviour of the vehicle during the diving phase of the mission. The red line comprised of '$+$' symbols represents depth measurements and the blue line comprised of '$\times$' symbols depth demands taken from the logs of Mission 12. Note that for long periods while the demanded depth is 200m, the vehicle keeps diving. This is because the vehicle dives in a *fixed stern plane angle dive mode* in which depth demands are ignored. Thus, such behaviour is nominal. The black stars represent moments when appropriate lines of mission script get triggered.

In the diagram on the right the situation is different. The vehicle is performing bottom following, maintaining an altitude of 100m, so is in *depth control mode*, where it responds to depth demands. In this mode, a divergence between the demanded depth and the actual depth (for example, the AUV diving while the demanded depth is shallower than the current depth) is a signature of a fault. The moment near 22350 seconds since mission start is when the stern plane actuator potentiometer fails and the stern plane ends up in a stuck down position, causing a steep dive. The vehicle was saved by triggering emergency releases due to exceeding maximum allowed depth. It was a fortunate coincidence that it was possible to specify a safe maximum depth that was less than the ocean depth because often the vehicle is used in rougher terrain in which defining such a safe limit is not possible.

To detect the actuator fault shown in Figure 2 we need to be able to tell at any point whether the AUV is in depth control mode or stern plane control mode (in which depth demands are ignored, and the angle of the stern plane commanded directly). Unfortunately, we do not have direct access to this information on-board as it is not broadcast to the logger. As we will show in Section VI, it is impossible to diagnose the fault without it, so we have to obtain it by other means. The approach we take is to generate a diagnosis model of the mission script automatically, directly from the script itself once it is ready to upload to the vehicle.

Mission scripts used in Autosub 6000 [7] can be thought of as a string of states, each of which contains a number of active demands. In Figure 3 there is an example of a mission script with three states. The argument of a *when* statement is a set of triggering events. When a triggering event occurs, like a *StartCommandReceived* or *GotDepth*, the body of the *when* block is executed where the demands specified in the body are

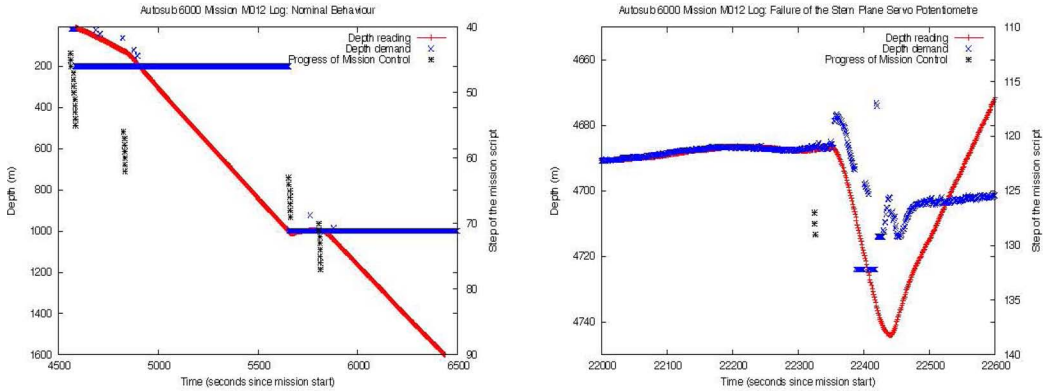

Fig. 2. Nominal and faulty behaviour of Autosub 6000 in M012.

```
when( Start )
  MotorPower( 300W),
  SetElementTimer( 0:0:18:0),
  RudderAngle(5),
  SetDepthThreshold( 1000m),
  SPlaneAngle( -20);

when( GotDepth )
  MotorPower( 250W),
  Altitude( 100m),
  TrackP( StartWayPoint1, EndWayPoint1),
  SetElementTimer( 0:1:2:30);

when( GotPosition, ElementTimeout)
  TrackP( StartWayPoint2, EndWayPoint2),
  SetElementTimer( 0:1:2:30);
```

Fig. 3. A sample fragment of an Autosub 6000 mission script.

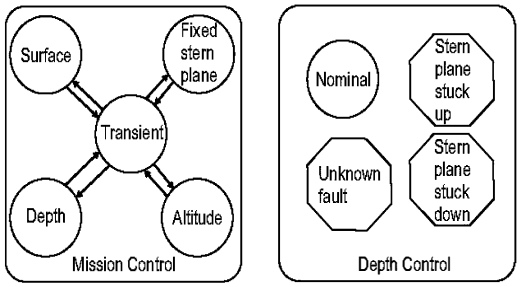

Fig. 4. State machines of the Mission control and Depth control components of the diagnosis model of Autosub 6000. Circles denote nominal states and octagons fault states.

activated. The script blocks at the next *when* block until the triggering condition is satisfied.

In addition to the main sequence of states, each mission script may have up to two separate *termination sequences* that get activated upon abnormal termination of the mission.

Because of the very simple structure of the mission scripts, we can relatively simply generate a model of the program. Essentially, each when block is represented by a mode in the mission script component, and transitions occur when the currently executed line of the mission script, as broadcast to the logger, changes. For example, the fragment of mission script in Figure 3 is represented by a chain of three modes, one for each when block. The first mode (a fixed stern plane angle dive) contains the constraints MOTOR.POWER=HIGH, DEPTH-CONTROL.VERTICAL-MODE=STERN-PLANE-DEMAND, STERN-PLANE.ANGLE=NEGATIVE, while the second (altitude following) contains DEPTH-CONTROL.VERTICAL-MODE=DEPTH-DEMAND and so on. The DEPTH-CONTROL component has a variable inside it called VERTICAL-MODE which the mission script model is setting constraints on, and similarly for the other components. To diagnose the fault, the depth control component has a constraint that says that if the depth control component is nominal and if the VERTICAL-MODE equals DEPTH-DEMAND, then the actual depth should not diverge from the commanded depth.

The state machine representations of the Mission Control and Depth Control components of the diagnosis model are given in Figure 4. The transitions in the Mission Control component are specified by the mission script, thus state of the component is inferred from observation of progress in executing the mission script. Each state corresponds to a set of constraints that should be valid whenever the system is in that state. There are two kinds of states in the model: nominal and fault states. While nominal states represent the normal behaviour of the system, fault states represent constraints that characterize certain fault states. For example, in the case of the STERN PLANE STUCK DOWN fault, the constraint that characterises that state is *if (*MOTORPOWER == POSITIVE*)* { DELTADEPTH == NEGATIVE }, which means that in case the motor is switched on, depth increases. The UNKNOWN FAULT state is to cope with the situation where none of the modelled fault states are consistent with the observed behaviour of the system.

When modelling faults with L2, it is often the case that faults in different subsystems can produce identical behaviours. For example, a sudden loss of buoyancy, perhaps due to failure of a pressure vessel, would results in similar behaviour to the STERN PLANE STUCK DOWN fault discussed

above. Livingstone provides two ways to cope with such overlapping hypotheses:

1) Include fault probabilities in the model.
2) Allow for multiple candidates with backtracking.

The first approach allows the modeller to adjust the likelihoods of different faults so that more probable explanations for a particular behaviour are preferred. A common example of this is that unknown faults usually have much lower probability than any modelled fault to ensure that the modelled faults are preferred to other explanations. Multiple candidates with backtracking stands for the capability of the diagnosis system to consider several explanations at a time and to refine the diagnosis when more observations, i.e. evidence, become available. In the case above, the two faults may be distinguishable if the motor is switched off—the loss of buoyancy fault will result in the vehicle continuing to descend while the vehicle should ascend in the case of the stern plane fault.

A. Depth profile

At the same time that we generate the mission script model, we also run a number of checks on the mission script and produce a depth profile of the mission based on an approximation of the rate of climb and descent of the vehicle, the demands in the mission script, and the timeouts on the length of each when block.

The comparison of the estimated and measured depth profiles is shown in Figure 5. On the left is a screenshot of the tool that performs the sanity checks on mission scripts and outputs the estimated depth profile. The right graph is a plot of the depth profile of the mission as measured by the pressure sensor of the AUV. It can be seen that the graphs look similar, but the estimated profile takes longer than the actual one. The reason being that the depth profiler uses only depth related information which is not sufficient for determining the estimated time for reaching certain waypoints. As reaching each waypoint is also associated with a timeout, the depth profiler includes all such timeouts. Thus the current implementation presents the user the longest possible duration of the mission permitted by the mission script.

V. Offboard diagnosis

Thus far we have considered the scenario where the diagnosis system is deployed on the vehicle (or as in our experiments, runs using data directly from raw log files recorded during previous missions).

One of the other applications of the automated fault diagnosis system for Autosub 6000 AUV is remote monitoring by telemetry as an aid to the operators. Telemetry is communicated from Autosub to the support vessel by acoustic link. While the acoustic link provides communication up to a range of 7km, its throughput is limited to 80 byte packets that are sent every 20-30 seconds. The probability of package loss increases proportionally with distance, and it is not unusual to see lost packet rates in excess of 20 percent. This provides a significant challenge for diagnosis as now our diagnoses are based on very little data, and some of the commands and state information may be lost for long periods.

Since the telemetry packets are so small, the Autosub engineers designed a system where packets containing different subsets of the vehicle's health data can be requested. By alternating among the five different packets we can, over the course of several minutes, eventually collect a reasonably complete picture of the vehicle's most critical systems which we can use for diagnosis. Of course, this too greatly increases the potential delay before a fault can be identified.

The purpose of the off-board fault diagnosis system is to alert the human operator in the event that a fault has been detected. Depending on the severity of the fault, it is then possible to pause or abort the vehicle's mission. One of the major advantages of Livingstone's consistency-based diagnosis approach is that we can use exactly the same models of components for the telemetry-based diagnosis as we used for the on-board diagnosis. This is because if data about particular observations isn't available, L2 will happily assume a value for that variable that is consistent with the model. The only component that this isn't true for is the automatically generated mission script component. There, since we rely on the line number of the mission script to transition from one mode to another, we can end up missing out a transition if two occur between message with the line number arriving at the support ship. To cope with this, a second version of the generated component is created (also automatically) that allows arbitrary transitions to any future mode.

VI. Experiments

To test the efficacy of the consistency based diagnosis system and our models we ran the diagnosis engine on logged data from twenty five actual Autosub 6000 missions, including the mission that includes the nominal and potentiometer fault data shown in Figure 2. For comparison, we ran the diagnosis model without the mission script model included. In all cases the models generated no false positive fault detections. However, in the case of the potentiometer fault scenario, the model that did not include the mission script component failed to detect the fault, while the model with the mission script component detected it at mission time 5:32:40, approximately 54 seconds after evidence of the fault first appears in the data and eight seconds before the abort weights were dropped because the vehicle exceeded the maximum allowed depth. Without the mission script component, the diagnosis system can't tell that the system is still in depth-control mode, so it explains away the faulty behaviour by assuming that the vehicle switched to stern plane control mode and is ignoring depth demands.

We also ran the off-board telemetry on the same scenario using simulated telemetry generated from the logs, and without any packet losses. Telemetry messages were generated every 20 seconds, so each of the five different sets of data would appear once in 100 seconds. In that case, the fault was detected at mission time 5:33:40, one minute later than the on-board diagnosis would detect it, and in this case 52 seconds after

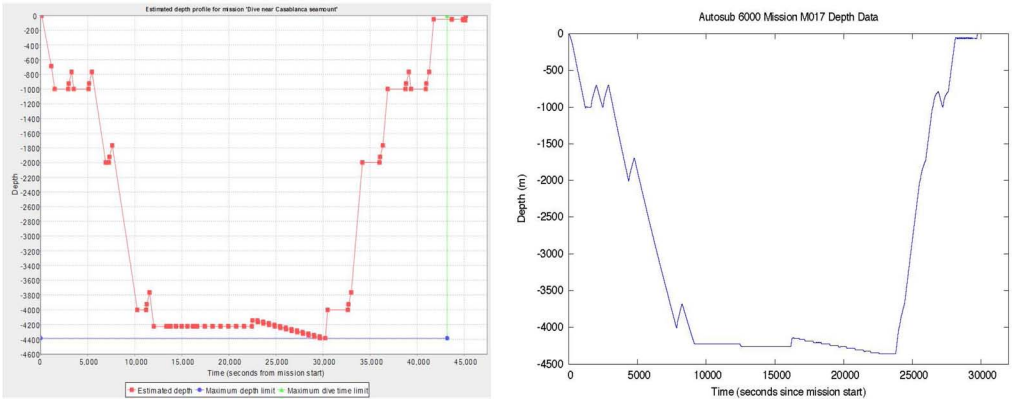

Fig. 5. Estimated (left) and measured (right) depth profile of mission 17 of Autosub 6000.

the abort weights were dropped. This delay was largely due to the fact that the mission line only appears in one of the five telemetry messages, and the fault couldn't be detected until a mission line had been received.

While detecting the fault only eight seconds before the abort weights were dropped (or a minute after) may seem somewhat unspectacular, the abort weights were dropped in the real mission only because the vehicle exceeded its maximum permitted depth. Had the vehicle not already been very close to its maximum depth the abort weights would have been released much later. In particular, if the maximum depth had been deeper than the seabed at that point in the mission, the vehicle would have collided with the seabed and would not have dropped its weights. This is very similar to an event that led to a vehicle loss for an earlier Autosub vehicle, which had to be retrieved using an ROV.

In addition to the depth control system model, we have also run a model of the batteries on real data from 13 missions in which significant battery problems have occurred. The diagnosis system correctly identified a number of problems with the batteries including several where on-battery sensors reported a pressure problem. In these cases L2 correctly detected a fault, but was unable to distinguish between a fault in the sensor and a fault in the battery. It also identified one case where a battery failed to switch on (in fact, the battery hadn't been installed on the vehicle), and one where a short circuit had occurred, although it was unable to determine which battery this occurred in. However, the system also found one false positive due to transient noise when one of the batteries was first switched on.

The battery system is quite a challenging one for Livingstone because many of the faults manifest themselves as subtle changes in the continuous measurement of current coming from the batteries. While the current model is useful, we feel that a hybrid model of a battery can provide a much more reliable diagnosis of many of these faults and will remove the false positives that we are currently seeing. As we say below, we are currently working on a way to integrate diagnoses from simple hybrid models into L2.

VII. Future work

At present, work is under way on a successor to Autosub 6000 called Autosub Long Range. This vehicle will have the capability to perform missions lasting up to six months at a time, so the ability to detect and mitigate faults is far more critical than for the existing Autosub vehicles. Autosub Long Range will be far from any support vessel for most of its missions, so no human monitoring will be possible unless it surfaces to 'phone home'. Thus a major goal for the future is to transfer our fault detection capabilities to the new platform. This brings with it a significant advantage, in that we can think about fault detection and what sensors and logging would be needed at the time the vehicle is being designed, which obviously makes it easier to identify faults. The challenge with Autosub Long Range will be power consumption—nobody will be interested in running a diagnosis system that will shorten the mission duration by a month—to keep the power low enough not to interfere with science data collection.

A second major strand of future work is in mitigations and recovery from faults. As we have said, there is relatively little mitigation that is possible, but in some circumstances a quickly taken recovery action is critical. In the Mission 12 scenario that we presented in Section VI the fault is detected just before the vehicle exceeds its maximum depth limit, but importantly also just before it collides with the bottom. If the bottom had been shallower than the maximum depth limit, then only on-board diagnosis causing an abort would have saved the vehicle; data from telemetry arrived too late. This makes demonstrating that the approach is reliable and safe enough to allow it to mitigate faults a crucial piece of future work. For missions under ice, this capability is even more crucial.

Finally, we haven't looked in this paper at a number of subsystems that can't be easily modelled using the discrete approach of L2. We have already talked about related work on diagnosis of thrusters for AUVs [6], but equally we may want to worry about entanglement with the propeller and how to distinguish that from operating against a strong current. Another example is the batteries. At present we have a very abstract model that usually concludes that there is an unknown

fault because it can't distinguish enough behaviours. A hybrid model that has discrete faults but also predicts continuous sensor measurements holds out much more hope for diagnosing systems like these. We have developed diagnosis algorithms for these kinds of models [18] based on particle filtering algorithms, but at present we haven't integrated them with L2. The particle filtering approaches are fine for small subsystems, but in their present form they simply won't work on large system models, so we hope to use L2 to integrate diagnosis information from a range of simpler diagnosers for subsystems.

VIII. Conclusion

We have shown that model-based diagnosis can effectively detect critical faults in an AUV, and could potentially prevent the loss of the vehicle should such a fault occur in the future. We have also shown that monitoring the execution of the mission has an important role in improving the speed and accuracy of this diagnosis. While some of the reason for this may be unique to Autosub, the combination of hardware and control software diagnosis makes more faults detectable, and gives greater information about the vehicle's state.

On the basis of these results we believe that MBD can be a very useful tool to increase the longevity of AUVs, and to help enable longer missions and greater autonomy in a wide variety of applications. The diagnosis system we are using is general purpose, so can be used on other vehicles, and we are slowly building a library of component models that could be reused for different vehicles than Autosub 6000. However, it is important to emphasize that there is still a very significant modelling task to get a system like L2 to work on a new vehicle: The system has to be understood by the modellers; the discrete models have to be built; the monitors have to be designed in conjunction with the models, and have to be implemented, and parameters of the model need to be tuned. We are developing tools that will hopefully make this easier, but there is still much to be done.

References

[1] G. Griffiths and M. Brito, "Predicting risk in missions under sea ice with autonomous underwater vehicles," in *Proceedings of the IEEE AUV2008 Workshop on Polar AUVs*, Woods Hole, 2008.

[2] S. McPhail, M. Furlong, V. Huvenne, and P. Stevenson, "Autosub6000: Results of its engineering trials and first science missions," 2008, presented at the 10th Unmanned Underwater Vehicle Showcase, The National Oceanography Centre, Southampton, UK.

[3] S. C. Hayden, A. J. Sweet, and S. Shulman, "Lessons learned in the Livingstone 2 on Earth Observing One flight experiment," in *Proc. AIAA 1st Intelligent Systems Tech. Conf., Am. Inst. Aeronautics and Astronautics*, 2004.

[4] S. D. McPhail and M. Pebody, "Navigation and control of an autonomous underwater vehicle using a distributed, networked control architecture," *The International Journal of the Society for Underwater Technology*, vol. 23, pp. 19–30, 1998.

[5] G. Antonelli, "4. fault detection/tolerance strategies for auvs and rovs," in *Underwater Robots 2nd Edition*, ser. Springer Tracts in Advanced Robotics. Springer Berlin / Heidelberg, 2006, vol. 2, pp. 79–91. [Online]. Available: http://dx.doi.org/10.1007/11540199_4

[6] A. Alessandri, T. Hawkinson, A. J. Healey, and G. Veruggio, "Robust model-based fault diagnosis for unmanned underwater vehicles using sliding mode-observers," in *11th International Symposium on Unmanned Untethered Submersible Technology, Autonomous Undersea Systems Institute*, 1999, pp. 22–25.

[7] M. Pebody, "The contribution of scripted command sequences and low level control behaviours to autonomous underwater vehicle control systems and their impact on reliability and mission success," *OCEANS 2007 - Europe*, pp. 1–5, June 2007.

[8] S. McPhail, "Autosub6000: A deep diving long range AUV," *Journal of Bionic Engineering*, vol. 6, no. 1, pp. 55 – 62, 2009. [Online]. Available: http://dx.doi.org/10.1016/S1672-6529(08)60095-5

[9] S. McPhail, M. Furlong, M. Pebody, J. Perrett, P. Stevenson, A. Webb, and D. White, "Exploring beneath the pig ice shelf with the autosub3 auv," in *OCEANS 2009-EUROPE, 2009. OCEANS '09.*, 11-14 2009, pp. 1 –8.

[10] J. Strutt, "Report of the inquiry into the loss of Autosub2 under the Fimbulisen," Southampton, UK, p. 39pp, 2006, national Oceanography Centre Southampton Research and Consultancy Report, 12.

[11] J. de Kleer and B. Williams, "Diagnosing multiple faults," *Artificial Intelligence*, vol. 32, no. 1, pp. 97–130, 1987.

[12] B. C. Williams and P. P. Nayak, "A model-based approach to reactive self-configuring systems," in *Proceedings of AAAI-96*, 1996, pp. 971–978.

[13] S. Narasimhan and L. Brownston, "Hyde a general framework for stochastic and hybrid model-based diagnosis," in *Proceedings of the 18th International Workshop on Principles of Diagnosis*, 2007.

[14] A. Feldman, "Approximation algorithms for model-based diagnosis," Ph.D. dissertation, Technische Universitiet Delft, 2010.

[15] J. Kurien and P. P. Nayak, "Back to the future for consistency-based trajectory tracking," in *In Proceedings of AAAI-00*, 2000, pp. 370–377.

[16] N. Muscettola, P. P. Nayak, B. Pell, and B. Williams, "Remote agent: To boldly go where no AI system has gone before," *Artificial Intelligence*, vol. 103, no. 1–2, pp. 5–47, 1998.

[17] F. Hutter and R. Dearden, "The gaussian particle filter for diagnosis of non-linear syste ms," in *Proceedings of the Fourteenth International Workshop on the Pr inciples of Diagnosis*, Washington, DC, 2003.

[18] N. de Freitas, R. Dearden, F. Hutter, R. n Morales-Menendez, J. Mutch, and D. Poole, "Diagnosis by a waiter and a mars explorer," *Invited paper for Proceedings of the IEEE, special issue on se quential state estimation*, 2003.

[19] S. McPhail and M. Pebody, "Range-only positioning of a deep-diving autonomous underwater vehicle from a surface ship," *Oceanic Engineering, IEEE Journal of*, vol. 34, no. 4, pp. 669 –677, oct. 2009.

[20] S. C. Hayden, A. J. Sweet, S. E. Christa, D. Tran, and S. Shulman, "Advanced diagnostic system on Earth Observing One," pp. 1–14, 2004, AIAA paper 2004-6108. AIAA Space Conference and Exhibit, 28-30 Sep., San Diego, CA, United States.