
Stephen McPhail

The Institute of Oceanographic Sciences Deacon Laboratory, Brook Road, Wormley, Surrey GU8 5UB, UK.

Abstract - Autosub is a project funded by the Natural Environment Research Council in the UK, with the aim of developing unmanned autonomous underwater vehicles to carry out long range scientific missions in the deep ocean.

The navigation system is critical to the survival of the vehicle, and so it is important that appropriate software engineering methods are used. Structured analysis and design methods are useful in helping to capture the requirements of the system, and in producing a well structured design that encourages re-use of code and data. This is necessary but not sufficient. We must also address implementation issues such as multi-tasking, sharing of hardware resources, and message passing and testing.

The author describes the methods that he has used in developing the navigation system and gives a personal account of the problems that he has encountered.

I. INTRODUCTION

The Autosub project has the aim of developing Autonomous Underwater Vehicles (AUVs) to carry out scientific missions throughout the deep ocean [1]. There are two long term vehicle concepts.

The DOLPHIN (Deep Ocean Long Path Hydrographic Instrument) would gather biological, chemical and physical data throughout the water column (down to 6000 m depth) across ocean basins (ranges up to 7000 km) and under ice. It will surface at intervals of 30 to 100 km for a Global Positioning System (GPS) fix. One of the main technical difficulties for the DOLPHIN concept is achieving the long range needed. Essential technological developments are: obtaining sufficient buoyancy from the pressure vessel, maintaining near neutral buoyancy at all depths, use of a high energy density source for propulsion power, and the development of a high efficiency propulsion unit.

For DOGGIE (Deep Ocean Geophysical Instrumented Explorer) the area of interest is the sea bed itself. DOGGIE would map the sea bed with instruments that include high resolution sidescan sonar and sub sea-bed profilers. At present, detailed information on the sea-bed is obtained by towing instruments such as the IOSDL TOBI [2] from a support ship, but the drag and weight of the tow cable limit the survey speed to less than 1 m/s, and precise navigation control of the towed body is difficult. DOGGIE will be able to carry out sidescan sonar surveys at more than twice this speed. The main technical difficulty for DOGGIE will be in achieving sufficient navigational accuracy without the use of regular GPS fixes [3]. The solution of this problem will require considerable advances in a combination of technologies including: inertial navigation, Doppler or correlation sonars for ground referenced dead reckoning, and acoustic navigation (triangulation) for absolute position fixing.

The Autosub project is at the stage of developing the sub-systems technologies necessary for the long term goals to be realised. The Demonstrator Test Vehicle (DTV) [4] will be the test bed for these sub-systems, and will provide experience of operating a vehicle of similar weight and size to DOLPHIN and DOGGIE. Table 1 shows the main parameters for the DTV.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speed</td>
<td>2 m/s</td>
</tr>
<tr>
<td>Range</td>
<td>1000 km</td>
</tr>
<tr>
<td>Mass</td>
<td>3700 kg</td>
</tr>
<tr>
<td>Length, diameter</td>
<td>7 m, 0.9 m</td>
</tr>
<tr>
<td>Depth limit</td>
<td>6000 m</td>
</tr>
<tr>
<td>Steepest angle of ascent or descent</td>
<td>45 degrees</td>
</tr>
<tr>
<td>Navigation</td>
<td>GPS when surfaced Dead Reckoning when submerged</td>
</tr>
</tbody>
</table>

Good progress has been made with the outline technical design of the DTV, and the measurement of the hydrodynamic coefficients (Fig 1).

But it is the problem of achieving adequate software reliability that is the concern of this paper. The mission management system (MMS) is the computer system which will manage all aspects of the control of the DTV vehicle including the functions of navigation, flight control, communications and data handling. The mission requirements are well defined and relatively straightforward and so we decided to adopt a largely algorithmic approach to the MMS system software design [5]. Hence the challenge is to achieve a highly reliable, well structured, documented and maintainable system rather than a system with a high capability for autonomous decision making. To assist with this we are using structured development methods.
The navigation sub-system is the first major sub-system of the MMS to be developed using these methods.

II. THE NAVIGATION SYSTEM - HARDWARE AND SENSORS

The function of the Navigation system is to give real time estimates of the vehicle’s position (latitude, longitude, depth and altitude), its velocity, and its attitude (roll, pitch and yaw). The vehicle will use the Global Positioning System when surfaced, and will dead-reckon when submerged, using speed through the water speed sensors and an attitude sensor.

Global Positioning System

GPS is the ideal solution for obtaining position fixes for a vehicle on the surface of the sea [6]. The position fix error of 100 metre rms falls well within the specification for most of the Autosub missions. For the DTV, the problem is that the GPS antenna system is likely to be periodically washed over by waves when the vehicle has surfaced. Computing the position of the vehicle under these conditions is not thought to be a major problem for modern receivers, provided fresh satellite orbital data (ephemeris) are available [7]. It is the acquisition of the ephemeris under washover conditions which is difficult. Work is at present underway with antenna design and receiver software to solve this problem.

Depth

The decision of whether to duplicate or triplicate sensors is determined by a trade off between cost (especially in terms of power consumption) and the benefit of improved reliability. The importance of depth measurement to vehicle control and the relatively low cost of the sensors makes it practical to triplicate sensors and use simple majority vote software to detect faults. I have configured the system to accept inputs from strain gauge sensors or from DIGIQUARTZ pressure sensors. Although the high accuracy (about 0.4 m repeatability for a 10,000 psi range sensor) of the DIGIQUARTZ sensors is not strictly necessary for navigation purposes in deep water it does enable us to carry out trials in relatively shallow water without the risk of accidental surfacing or collision with the sea-bed.

Attitude Estimation

For reasons of cost, reliability, power consumption and weight, the vehicle attitude is measured with a tri-axis magnetic flux gate sensor and a tri-axis accelerometer system. There are, however, two major problems with this approach.

The first problem is that the earth’s magnetic field is subject to unwanted temporal variations (due to magnetic storms, diurnal variation and secular change [8]) and spatial variations (unmapped effects due to magnetised minerals). The vehicle itself will also distort the apparent field direction due to electrical currents and the presence of ferromagnetic materials. We expect that these effects will reduce the heading accuracy to about 1 degree. This is acceptable for the long range hydrographic missions, given GPS position fixes every 50km.

The other problem is that lateral acceleration of the vehicle perturbs the measurement of roll and pitch using the accelerometers. This in turn causes an error in the measurement of the vehicle heading. The concern is that this can cause deterioration in closed loop yaw control of the vehicle. I have modelled this effect, using a frequency domain model of the DTV, with measured values of the vehicle hydrodynamic coefficients. Fig. 2 is a gain-phase plot for the DTV in yaw, for the cases with and without roll error due to lateral acceleration. Although the effect is nominally de-stabilising, it is tolerable (phase margin is reduced from 58 to 54 degrees).

![Fig 2. Open loop Gain - Phase Diagram for the DTV in Yaw.](image)

It is customary to define the attitude of a marine vehicle in terms of the Euler angles of yaw, pitch and roll, but these are not convenient for computational purposes. At angles of pitch approaching 90 degrees, both the roll and heading become ambiguous. I have chosen to represent the attitude of the vehicle with a 3 by 3 matrix, known as the attitude matrix or the direction cosine matrix. Each element
in the matrix is the cosine of the angle between one of the axes of the fixed (earth's geographical) co-ordinate system and one of the axes of the rotated (vehicle's) co-ordinate system [9]. There are several advantages of using this representation of attitude over the Euler angles:

- It represents the attitude of the vehicle without any singularities.
- It is calculated easily from any two vector measurements (in this case gravity and magnetic field) by simple matrix operations, without the need to use any trigonometric functions [10].
- Any vector can be transformed from a fixed to a rotated frame of reference by multiplication with the direction cosine matrix, and the reverse transform from rotated to fixed frames of reference is equally simple.

**Speed Estimation**

Developments in Doppler or correlation logs [3] coupled with improvements in the performance and cost of heading sensors may eventually make ground referenced dead-reckoning a practicality for Autosub missions, but it is planned that the DTV will measure its speed relative to the sea water, using either three axis electromagnetic speed logs or measurements of propeller revolutions per minute and power input to the propulsion motor. The latter has the clear advantage that no extra sensors are needed. If the propulsion system is functioning correctly then the necessary information needed for speed estimation will be available. Fig. 3 is derived from a simple mathematical model for a propeller of similar characteristics to the one that will be used on the DTV. For constant propulsion power, it gives the estimated speed as a function of the measured propeller rpm.

**Structured Analysis**

Structured Analysis is a method for producing an unambiguous and comprehensive specification for the system. Without such a specification, the concept of "reliability" is meaningless, as there is no way of testing whether the system is satisfying the requirements of the user. Data Flow Diagrams [11] (sometimes called process charts or "bubble charts") are process oriented, they provide the main graphical way of specifying the system. In these diagrams the processes are depicted as "bubbles" and the data flows between processes are shown as directed arcs. The method relies on the decomposition of the diagram into more and more detail, until finally the processes can be specified using a simple written description (process specification or pspec). Fig 4 is a simplified data flow diagram for the top level of the navigation system.

**Structured Design.**

Once the specification has been completed and validated against the requirements of the user, then the design of the software can proceed. We are using the structured design methods of Yourdon and Constantine [13]. Program Structure Charts show procedures as rectangles, and parameters as small circles with arrows indicating the direction of data flow. The logic of the procedures is defined by a Program Description Language (PDL).

**Multitasking and Partitioning of the System.**

The above methods describe a development route for a single program, but do not explicitly address a critical stage in the design of a multi-tasking system - the partitioning of the system into separately executable tasks. Before describing the partitioning of the system it is worth considering the arguments for and against use of multi-tasking in a real-time embedded system.

The major advantage is that it offers more efficient utilisation of the processor. When a task is suspended, waiting for some event, another task can use the processor resources. It can also simplify system design. Processes

---

3Software Through Pictures - IDE Inc.
that are inherently asynchronous, or repeat at different rates

However, there are considerable problems and dangers
in using multi-tasking (or concurrency in general):

- It is more difficult to prove that a concurrent system
  meets its specification at all times, especially if there
  are many tasks.
- There are the dangers of dead-lock (all tasks suspended)
  or live-lock (tasks still running but can no longer
  carry out the function of system), which can be
difficult to spot in a design.
- There can be appreciable delays due to the operating
  system switching between tasks.

But the greatest problem with multi-tasking is the
potential for increased complexity. The number of possible
interactions between tasks increases rapidly with the
number of tasks. In addition to this, common real time
kernels (such as VRTX32® [14] which we are using) offer
several different ways in which tasks can exchange data and
control, all at a fairly low level. This could be dangerous
in the wrong hands.

In an attempt to control these problems, I have tried to
reduce the number of tasks to a minimum, and avoid
unnecessary coupling between tasks.

Fig. 6 is the simplified task diagram for the
Navigation system. It shows the external sources and
sinks of data (square boxes), the tasks (oblongs), the mes-
gage passing within software (solid lines), and the physical
data interfaces (dotted lines).

![Fig 6 Simplified Task Diagram for the Navigation System](image)

**Message Passing**

So what is needed is a simple and safe way of passing
data structures and control information between tasks.
VRTX32 offers several ways of exchanging messages and
control information between tasks, for instance Dijkstra
(counting) semaphores, FIFO and LIFO queues, and
mailboxes (32 bit values using global memory). But I did
not want to use these system calls at the design level, be-
cause they are at too low a level and only support the
passing of 32 bit values (which can be data or addresses).
Instead, I defined a message passing protocol and then
implemented it as function calls (using VRTX32 system
calls). There are only four C function calls needed.

```c
async_send(channel, data_object, timeout, error)
  Sender waits for receiver before continuing.
async_send(channel, data_object, error)
  Sender does not wait for receiver.
sync_receive(channel, data_object, timeout, error)
  Receiver waits for data from sender.
async_receive(channel, data_object, error)
  Receiver polls for data.
```

For the task diagram (e.g. Fig. 6) it is assumed that the
name of the channel and the data type are the same, as only
one type of data may be passed through any channel. Hence:

- `data(A, A)`: neither sender nor receiver waits
- `data(S, A)`: sender waits, receiver does not
- `data(A, S)`: receiver waits, sender does not
- `data(S, S)`: receiver waits, receiver waits

In this way the direct use of low level system calls has
been completely avoided, thus considerably simplifying the
design and implementation.

**Resources Sharing.**

The structured design method facilitates the sharing of
procedures. The CASE tool also provides a data structure
tool, using Jackson [15] structure charts which
encourages the sharing of the data types.

Sharing of devices, such as analogue to digital converters (ADCs) can result in a less coupled, more
modular design. For instance, the task diagram (Fig. 6)
shows the output of the attitude sensors going to both the
position estimate and the attitude estimate tasks. This is
made possible by system software that serialises access to
the ADCs, making them safely sharable. Consequently,
the attitude estimate and the position estimate tasks are not
coupled (position estimate gets its attitude data from a call
to the re-entrant function call `get_attitude`). This approach
considerably simplifies testing and debugging.

**IV. CONCLUSION AND DISCUSSION.**

There are at least 6 stages to this process. Is it reason-
able to ask whether there might be too many stages? What
are the merits of each stage?

Structured Analysis (SA) is perhaps most effective
where there is a clearly defined customer /contractor re-
relationship. The structured specification is a basis of a con-
tract, and as such is essential. SA can also serve as a
useful communication tool for use within an organisation
as a data flow diagram is easy to read. But exactly how the
structured analysis and design method should be used by a
small software team employed on a research and
development project is less clear cut. The resources
available must be considered. If the analysis, design and
coding are carried out by one person, then isn't it inevitable
that he or she will make design decisions at the analysis
stage? Is it efficient in these circumstances to totally
disregard implementation issues when producing the specification and design? The maturity of the project should be taken into account. If the requirements are still loosely defined, and will only become stable after the prototype stage of the project, then we could waste time by going into too much detail in the specification. The CASE tool can exacerbate the problem because it makes it so easy to create many levels of decomposition of the data flow diagrams and data structure diagrams. The amount of documentation that it can produce can be quite daunting.

A real time operating system offers much improved processor utilisation, and this could ultimately result in greater total system reliability, as less hardware is needed. But the dangers should not be under-estimated. The misuse of multi-tasking could be very costly.

I have no reservations about using the Structured Design method and CASE tool as it is particularly effective. It helps in producing a safe structured design, encouraging reuse of code, and is helpful with debugging, generating test plans and maintenance. It really does save time.

Use of C as the implementation language did not present many problems that were special to C. The Structured Design method greatly reduces the scope for bad or dangerous style. Most problems (and there were relatively few) were due to logical errors in the design rather than particular limitations of C. One complication, however, which is common to many procedural languages such as C, is that the language has no built in support for multi-tasking, necessitating the "adding on" of a multi-tasking functionality via operating system calls. This is not ideal.

Whereas design was best carried out top-down, I found that it was generally easier to implement and test the code using a mixed bottom-up and top-down approach.

Structured Analysis and Design methods with CASE support can benefit a project such as Autosub provided they are used sensibly according to the project circumstances. But I have definitely found that the old adage "Rubbish in ‡ Rubbish out" is very true!

ACKNOWLEDGMENTS

I am pleased to acknowledge the useful contributions to this work made by my colleagues in the Autosub team at IOSDL.

REFERENCES