Computer Science 773

Robotics and Real-time Control

TESTING

Once a programme is designed, then, unless you're supremely confident or foolhardy, you want to test it. This will still be true even in the unlikely event that someone contrives a reliable method for deriving programmes directly from specifications, because the specifications are at least as likely to be wrong as the programmes are.

As with many other parts of real-time computing, testing is harder than it is with conventional systems. Timing is always a problem, but in testing another common difficulty is to find a realistic execution environment where significant testing is still possible. It is very common for real-time software to be developed on a system which bears little resemblance to that on which it will eventually be run, because it is not usually considered to be economical to close down a factory for a couple of weeks for software development. Even if it were, you probably wouldn't want to, because much of the code might well be destined for execution on isolated microprocessors with no facilities for programme development.

Generally, then, the testing problem is to reproduce a realistic environment for code execution while also having useful facilities for monitoring and diagnosis which, by definition, make the environment unrealistic.

Of course, there is much that can sensibly be done on system components, just as there is with conventional software; the functions of various modules can be tested in isolation, and so can their timings, but – while helpful in general – this is only sufficient in a very simple system.

AUTOMATIC TEST SYSTEMS.

A simple approach which satisfies the requirements of realistic environments is to use automatic testing methods such as are used to test other electronic machinery. A computer system is in fact an electronic machine, so it might be possible to use similar testing techniques.

These work by attaching the device under test to a tester which applies certain input signals, and monitors the output. If the output corresponds to the expected value over the whole range of applied inputs, than the device is deemed to work satisfactorily. While they are really intended for routine testing of large numbers of manufactured articles, there is no reason why something of the sort couldn't be used as a software development aid if it seemed to be worthwhile; this sort of checking is certainly likely to be useful.

In practice, this doesn't work. One difficulty with this method is that any device containing a computer is far too complicated to be tested adequately by simple input-output tests, for its internal state is so complex that an enormous number of tests would be required. The response to individual signals is not usually as important as the response to long input sequences, but the testers' capacity for such sequences is severely limited. A second, and possibly more significant, difficulty is that it only does the easy part of the job; it tells you that the device doesn't work, but gives little or no clue as to why it doesn't work. Finally, it would only work with very small devices having rather few input and

output connections. While that doesn't make it useless, it does rule out the hard cases like large scale automation.

From the point of view of real-time computing, the method remains of interest – because the testing machines themselves are examples. Modern testing equipment is controlled by computer, and is likely to be programmable so that different input sequences can be used, with corresponding output values defined. The programmes must also define the electrical natures of the signals concerned, and a wide range of transducers must be supplied to cope with various different sorts of input and output signal.

SIMULATION.

If you can't use the real system, then perhaps you can use something like it. Simulation techniques of various sorts are widely used in testing real-time software. While it is commonly difficult or impossible to reproduce the real system perfectly, there are compensations; in constructing the simulator, you can build in means of monitoring the behaviour to give information which might not be available with the real system.

There are other benefits to the use of simulators. They are comparatively cheap to run, so you can do many experiments; they can simulate extreme or dangerous conditions, so that you can run furnaces at temperatures on the verge of breakdown or crash aircraft in safety; with careful attention to detail, you can take advantage of the simulation to speed up or slow down or otherwise change the real behaviour in order to cover more cases quicker or to explore behaviour in greater detail when diagnosing faults; you can test a distributed control system within a single computer, making it much easier to check overall performance and communications between the components.

The reliability of simulation testing is determined by the degree of realism of the simulating system. Different simulation techniques are associated with different degrees of realism, and, as a generalisation, the more realistic the method, the more expensive it is. Nevertheless, even a mediocre simulator can give much useful information, and simulators of all sorts are widely used in developing real-time software. Very brief descriptions of three sorts of simulator follow, in order of increasing realism and expense.

Computer simulation using one computer is the simplest method. The simulated system and the control programme run in the same computer. This is cheap(ish), and there are no connection difficulties, so it can usually be done rather easily. On the other hand, it is practically impossible to achieve realistic timing for any but the simplest systems, as the processor must be used for both control and simulation.

Computer simulation using several computers can go at least some way to solving the timing problems; one (or some) machines are used solely to simulate the system to be controlled, while the control system can be run on a realistic computer configuration. This is comparatively expensive, though it can be very effective. A significant defect with a simulated plant is that it is hard to make the simulation imperfect, so irregularities in behaviour which were not foreseen will never occur.

Pilot systems are physical models of the real system. These behave comparatively realistically, including unexpected behaviour, but are expensive to build.

OPEN-LOOP TESTING.

This method moves a step closer to the real machinery. Assuming that it exists and can be operated realistically without the new control system, one can take the plant outputs (sensor readings and other necessary information) and connect them to the new system as they will be connected when in operation. The new control outputs are not connected back to the plant; instead, they are monitored, and the values compared with those expected. In other words, the data acquisition functions of the new system are used, but not the control functions.

This is a valuable technique when you can use it, which is usually when you're producing a new control system for existing equipment. This is not as unusual as it might sound, as there are many ways in which routine plant alterations will fit into this pattern. For example, any improvement to an existing system, perhaps developed to sort out an observed fault or to take advantage of new sensory data, can be tested in this way.

Alan Creak,
March, 1997.