Modelling behaviour

Jon Bell

Doc. ref. SD/TR/MM/01: November 21, 2002

1 Introduction

This report is intended to raise questions on some areas of modelling network and software systems that have not yet received much thought, at least formally. Some informal consideration has been given to various areas not yet covered in any previous reports, as noted in the progress section of my PhD first year report.

The following areas are discussed.

- Interaction between structural, behavioural and functional models.
- Modelling failure mode behaviour.
- Extending functional modelling to show late achievement of a function.
- Modelling intermittent faults.

The idea is to suggest possible approaches, the hope being that discussion will weed out the less appropriate ones.

These thoughts partly arose while writing the PhD report and also from studying the Belt Minder system Statemate diagrams. Problems arising from using different tools for modelling different parts of the systems will be discussed in the section on interaction between models.

This report is a rather hurriedly produced draft, as I felt it worth noting down some questions and ideas that have arisen and also I felt this to be a productive way of using the last few days before going away.

2 Interaction between models

There seem to be good reasons for combining behavioural modelling with AutoSteve's structural modelling for SoftFMEA, in preference to using functional representation. These include:

- It is close to AutoSteve's current approach, so we avoid re-introduction of (redundant) component / subsystem level functional modelling, easing model building.
- It seems to be an appropriate approach to modelling failure mode behaviours. This is discussed further in section 3.

- There exists an appropriate formalism for describing behaviours, state charts being used both by AutoSteve and Statemate.
- Ford already use behavioural descriptions (in Statemate) for the design of systems with complex behaviour.

This was the expected direction of the project - an early task was to examine alternative languages suitable for behavioural modelling. Of these, the two best candidates seem to be SDL or Statemate. A comparison of these and other alternative languages has been produced [1] and I do not propose to cover that ground again here. If we are to use another language, then the fact that Ford already use Statemate suggests that that is the language to choose. However, there is one difference between Statemate and SDL that might be worth considering.

In Statemate the system structure is specified in the module chart and the interactions between state charts in the activity chart. There is a need to ensure that a signal in a Statemate activity chart has a physical route, so there is a need to check the correspondence between these charts. Statemate apparently does this, but it is not clear how - it can, of course, be done statically rather than as part of a simulation, rather like the way AutoSteve checks for consistency between a component model's structure and behaviour on saving the component. In other words, Statemate has a structural view of the system which might be redundant for our purposes, as the ECAD schematic might be used as the basis of AutoSteve's structural simulation. There might not be a one to one mapping between state charts and components, of course, and this will clearly affect the relationship between the Statemate activity chart and the ECAD schematic.

SDL has one class of diagram that shows the interactions between process or between blocks of processes the "block diagram". As the unit of execution in SDL is termed a process, it seems unsafe to assume that these processes are related to distinct components, there appears to be no SDL equivalent of the Statemate module chart. This lack of a structural view of the system in SDL might be argued to eliminate redundancy, if we are to use the ECAD schematic as the structural view.

The limited transmit and received facility in State Builder already uses the ECAD schematic as the structural view, describing routes for behavioural level interaction between components.

One aspect of mixing structural and behavioural modelling for different parts of a hybrid (electrical / software) system is where the boundary between the structurally and behaviourally modelled subsystems is to be found. It will, of course, typically be that component models will form the boundary, or cross it, depending on the pint of view. It is currently the case that an AutoSteve component model has structural and behavioural models and it seems the case that this relationship will form the basis of the interface between structurally modelled and behaviourally modelled subsystems. For example a switch ECU will take electrical inputs (from the switch itself) and generate digital messages as outputs. In other words its input side will be part of the structural (AutoSteve / CIRQ) simulation while its output side will be modelled behaviourally. This can be captured (to a limited extent) using the transmit / received facility in State Builder, but even here it is left to the user to make the behaviour relate to all the electrical inputs. An obvious example is placing all the behaviour in a high level "active" state entered if the component's power supply is live.

All components to be included in the behaviourally modelled subsystem will typically have this relationship with the electrical system, if no other.

The use of a separate tool for behavioural modelling exaggerates this, as then the structural and behavioural views of the component will each be in different tools. We might have a behavioural model in AutoSteve to bridge the gap between the structural model and the other tool's model, but seems decidedly redundant as well as being extra work for the model builder.

2.1 The belt minder model

We have the Statemate state and activity charts for the belt minder system. This system lights a warning light and sounds a chimer if the car is started with the driver's seat belt unbuckled or the front seat passenger's seat belt unbuckled if that seat is occupied. There are overrides, so that the belt can be unbuckled for manoeuvring and the system can be overridden by buckling and unbuckling the seat belt (I think!). I do not propose to discuss its behaviour in detail here, merely to discuss the Statemate charts in terms of the points raised above. It should be noted that the reading of the Statemate charts is speculative, so errors are possible.

The interesting point for us is the relationship between the software and electrical systems, in other words where these charts might fit in an AutoSteve system.

As we have no module chart, there is no way of telling what component an activity is to be associated with. It may well be that all activities in this set are on one PCB. The first chart in the set shows the system input / output requirements, and this shows several external components, which I think we can safely assume would be shown on an ECAD schematic, so be distinct parts of the electrical simulation. Many of the I/O signals are labelled "HW" as part of the name, which I imagine indicates that they are hardware, and I suggest implies they are electrical signals that could be modelled in AutoSteve. This is supported by the fact that the sources are switches — those that detect the buckled state of the seat belts, and that which detects occupancy of the passenger seat. These can, I think, all be regarded as on/off switches, suggested by the fact that the signals are treated as two valued (Boolean) variables. We were told that the system did not use a network, so the nature of the output signals, to external items labelled as "OP" — operator? These mostly seem to be modelled as signals with two possible values (buckled or unbuckled) so again it is quite feasible that they are actually models of electrical states — on or off.

If these speculations are correct, then the system can be regarded as taking the place of an AutoSteve component model. The main difference being simply the extra complexity. The boundary between the two levels of modelling would be the signals. This means the behavioural model needs to detect incoming electrical signals from the electrical model, and generate output ones. It would also, of course, depend on the correct functioning of the power supply, which has been omitted from the Statemate model, but would be part of an AutoSteve one.

It is not clear from this model how failures are to be handled. Even if the belt minder PCB is treated as reliably solid state, we might want to find what happens if, say, the wire connecting the driver's seat belt switch goes open circuit. Assuming this signal is a model of an electrical signal, it is reasonable

to assume that one of its two values is represented by the absence of current, and this value will be returned if the wire goes open circuit. It seems likely that buckling the seat belt will close the switch, so an open circuit will mean buckling is not detected. Clearly for mixed modelling these relationships will need to be explicit.

The main conclusion from this seems to be that the model builder will have to make correspondences between the models explicit. This is to ensure that the structural and behavioural models correspond (there is a physical route for all signals, for example) and to ensure that the translation between electrical states and signal data types is specified. This is, of course, done in AutoSteve as the conditions in the behavioural models are typically derived from the component's electrical state. If we are to use the ECAD schematic as the sole structural view (to avoid redundancy), some explicit mapping between activities and components might also be needed.

3 Modelling failure mode behaviour

Actually there seems to be little doubt of the way forward here. If we are to use behavioural modelling (probably using state charts) for component's behaviour, extending the existing AutoSteve approach then a state chart can be used as readily to model failure mode behaviour as correct behaviour, provided such behaviour is defined. If it is not defined, is there a case for expressing the lack in terms of non-achievement of a function? State charts should also be a suitable formalism for including fault mitigation behaviour, this will add complexity to the component's state chart, but will be an accurate modelling of the intended behaviour.

The obvious alternative to using behavioural modelling is to use some form of functional representation. This has disadvantages. The most obvious of these is the danger of reducing a subsystem's failure to non-achievement of a function, losing any unexpected consequences of the failure. It is not impossible, for example, that a failure in an electrical part of the system also affects the power supply to other components, even though this is not part of that subsystem's intended function. Therefore if all consequences to a failure in the subsystem are to be modelled, there is a need for a more complex functional model of that subsystem. This is not necessary if we use behavioural modelling. The trend during development of AutoSteve has been to reduce use of functional representation, as it was found to be redundant. It seems that attempting to use functional representation here reverses that trend, so is likely to reintroduce this redundancy and make a good deal of extra work for the user when building the model.

Another question that arises is "what need is there to model a component's response to external failures?" Might a wire connecting an ECU to an operator damage the ECU if it shorts to power? If so, how should this be modelled?

4 Late achievement of functions

The present functional labelling in AutoSteve associates a system function with the output of certain significant components, so does not explicitly link function

with an input. Such a link seems essential for modelling late achievement of a function, as we need to model the delay between the input and the achievement of the function. The AutoSteve SCA tool does attach function to input, so that unexpected functions can be recognized in SCA. This might provide a starting point for linking function to input in FMEA. One disadvantage is, of course, the loss of re-usability of the functional model. Having a functional model linked to inputs implies that model having a greater knowledge of the system's structure than is the case - it needs the name and input properties of the relevant switch, for example. One obvious way of avoiding this difficulty is to attach the expected function to the input property, so a headlamp dip switch will have two positions, "dipped" and "main" and each will have an expected function attached. The difficulty here is if the new state of the switch depends on an earlier state, if the switch is a toggle switch, for example. In this case the expected function needs to be attached to the component's behavioural description. The idea of attaching expected function to the component has the same problem of hindering re-use - a dip switch could reuse a generic toggle switch model if not expected functions are attached, but not if they are. One answer would be to attach expected functions to the component instance rather than the generic symbol.

Clearly, if we are to model late achievement of a function, we need to add a deadline by which the function should be achieved. There seems no reason why the present AutoSteve qualitative notion of time should not be used here. The main difficulty is the danger that it is too coarse-grained.

The possibility of message delay could arguably simply modelled as a failure mode for a network component. How this is modelled (all messages delayed or some) is discussed in section 5. One possibility is to give the network some way of modelling loading, and give the nodes some indication of priority, so that some statistical probability of a message delay can be arrived at, and this added to the FMEA report. This works for simple systems with no fault mitigation strategy, but we actually need to specifically model a late message as an event if such a strategy is to be exercised.

5 Intermittent faults

When a system uses a network, it is possible that, during period of high load, some messages but not all will be delayed. Previous work has modelled faults as continuous, but there is a possible need to model the late arrival of some, but not all, messages. This is particularly important if a late message results in a fault mitigation strategy being used, and this strategy is itself to be tested. Various approaches are possible here, including attempting a model of the entire network, which suffers from the difficulty that not enough will be known about its behaviour early in the design life cycle. There also seem to be problems with correctly modelling the interactions of a genuinely concurrent system consisting of several ECUs. Another possible approach is to simply model all possibilities, both have the fault occurring (so testing any fault mitigation strategy) and not occurring, so testing normal behaviour. It might well turn out to be sufficient to model all messages as being delayed so that it is a failure mode like any other. Some probabilistic method might be used, allowing the user to specify the priority of the messages in the subsystem under test and the network loading, resulting in a probable delay being used, so maybe one message in ten is delayed.

This would allow an occurrence figure to be generated for the FMEA report. This might be useful where there is no fault mitigation and a delay in a message being received will simply result in a delay in achievement of a desired function. This implies some method of capturing the priority of the message and possible network loading, to allow the probability of such a delay to be arrived at. There may be no need to model this specifically, we can simply treat late messages as a failure model of the network.

6 Conclusion

It is perhaps too early to suggest solutions to the questions raised herein, several possibilities have been outlined. These outlines are intended to spark discussion. It is hoped that appraisal of the proposed initial State Builder based implementation will inform further consideration of these matters. It would also help to learn more about fault mitigation strategies.

The "model framework" suggested in [2] might be considered a basis for implementing some of these ideas. That notion was not closely defined in the report, as that report was prepared hurriedly in time for the cancelled meeting with Ford in February. At the time, I had in mind the idea of modelling each subsystem fairly independently and using some sort of functional representation to constrain the interactions between the subsystems. My current feeling is that that is not appropriate, for the reasons outlined in section 3. We will (I imagine) still need to model all components' behaviours, therefore, it seems economical of effort to use that level of modelling for the interaction between subsystems. One possible approach is to devise a protocol to constrain such interactions, this protocol being grounded in some concrete relationship between the two models. The source of energy is one basis for such a relationship. Many of the necessary correspondences between the models can be checked for statically, indeed it might be that the the user will need to specify which component structural model corresponds with which behavioural model.

References

[1] Jon Bell. Languages for simulation of network and software components. SoftFMEA document ref. SD/TR/01, 2002.

[2] Jon Bell. Proposed approaches to network simulation. SoftFMEA document ref. SD/TR/03, 2002.