Using a state machine language for behavioural modelling

Jon Bell

Doc ref SD/TR/FSM/01: 28 March 2002

1 Introduction

This report proposes an approach to using the existing Statebuilder state machine language for modelling the behaviour of components in hybrid (electrical / software) systems. It suggests extensions to the language that are necessary if is to be used for modelling subsystems at a behavioural level. An example would be a group of components that communicate via a network (such as CANbus) rather than electrically.

The suggested approach was arrived at in the light of investigations that resulted in the report *Events and Signals* [1] and subsequent discussion with Neal Snooke. It also follows on from the earlier technical report *Approaches to network simulation* [3].

The following section will outline the proposed approach, and further sections will then discuss its advantages and problems and its relationship with the earlier proposals, outlined in the other reports. Other ideas arising will also be discussed, as will limitations and future work.

2 The proposed approach

The suggested approach is to add facilities to the existing AutoSteve state machine language to allow different component's state charts to send and receive 'signals'. While there are similarities with the 'first cut' Statebuilder based approach suggested in [3], this elaboration of that idea solves problems associated with the earlier proposal.

Instead of combining all component state charts into one composite state chart, whose components are identified by connections to a network (identified by 'nets' in the schematic?) these network connections are used to specify 'signal routes' by which individual state machines can send and receive signals. This adds one of the features of SDL to the existing language, and it is hoped might do so almost invisibly to the user, though it is accepted that components' state charts will be complicated. This has the advantage over the composite state chart approach that each component's state chart is more independent. The component will send its signals, and other components on the network can respond to those signals in their own way. As the outgoing signals are specified there is (arguably) less need for the user to match signals across the chart,

though clearly s/he will have to ensure receivers are specified to respond to correct signals. The architecture of the proposed approach might be as illustrated in Figure 1.

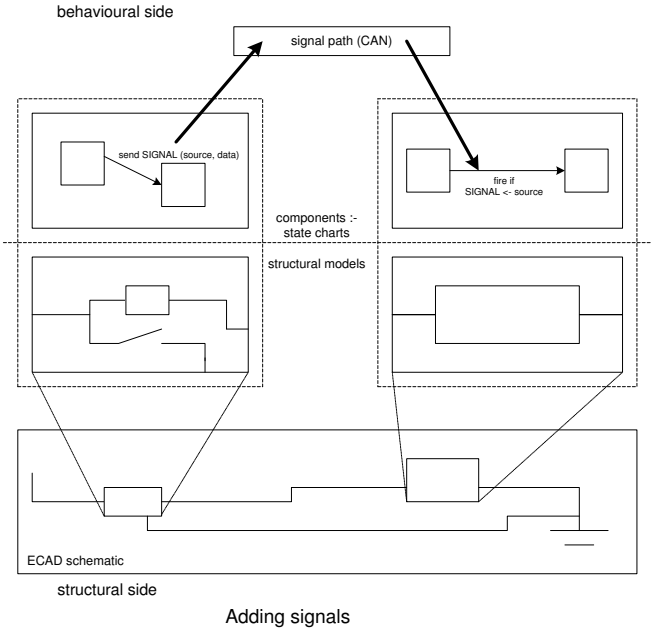

Figure 1: Adding signals to the AutoSteve model set

The signal itself also needs modelling. In the diagram, it has been suggested that a signal will have two attributes, a source and data (the content of the message). This is a simple model of a CAN message, as they are identified by source. We might also need to model messages without the source attribute, for example if CAN terminals are modelled as separate components one of whose roles will be to add the source. We might also wish to model simpler signals when modelling software components. As the current facility only allows one part of a state chart to fire an event in another part, we need to add a mechanism for passing data (for example from sensors) which this extension provides. Incoming signals will trigger transitions in the receiver's state chart. These transitions can, of course, depend on the source and/or the content of the incoming signal. This addresses the difficulty with the earlier Statebuilder proposal that messages are limited to instructions. It should also ease model building as the signals are not dependent on the receiving component's reaction to them. ¡p¿The fact that an incoming event might be a data carrying signal means that the recipient might well need a slightly more complex state chart, as it will need intermediate states to model its processing of the signal's data. Figure 2 suggests the change that might be necessary to a state chart for the heater ECU component in the heater circuit discussed in [3].

An incoming signal from the Sensor will fire an event from either 'HEATER ON' or 'HEATER OFF' to an intermediate state ('PROCESSING'). Which transition is fired from this state will depend solely on the result of the process-

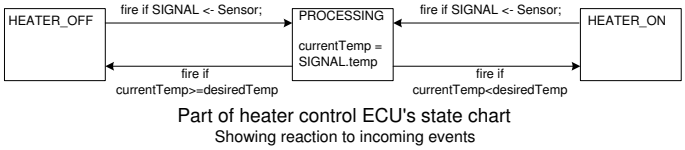

Figure 2: State chart of part of example system

ing. In the diagram, the electrical actions associated with 'HEATER ON' and 'HEATER OFF' have been omitted. They will, of course, be similar to those used now, and will relate to the component's structural model. Entering the 'PROCESSING' state will, of course, cause no change to the electrical state of the component.

These additional intermediate states are not unlike those used in SDL behaviour charts. In this case, where the result is a decision, a Statemate like decision Where a signal's data is tantamount to an instruction (for example a signal passing on the current setting of a switch) these intermediate states could, perhaps, be omitted by adding a condition to the firing of the transition along the lines of "fire if SIGNAL from switch ECU and SIGNAL.content = 1". This approach could perhaps be used in some cases even where data (as opposed to an instruction) is passed, provided the incoming data has no side effects other than possibly causing the transition. In the case illustrated, it is necessary to update the value of 'currentTemp'. This must be done whether the change from 'HEATER ON' to 'HEATER OFF' happens, if we are to place the updating of the 'currentTemp' variable in an action associated with a transition, we will need to add loop transitions for when the signal is received but does not lead to a change of state.

To summarise, it is proposed that we replace the idea of a composite state chart with an interaction diagram, whose graphical representation is the ECAD schematic (using nets to identify network connections) and we add a signal model to the state machine language. A component can send a signal (typically as an action associated with a transition) and it can fire a transition in a receiving component's state chart. This proposal provides a more accurate model of CAN messages, and it is suggested can readily be used to model other interactions at a behavioural level, such as within complex software components. It seems worth pointing out that there seems no reason to suppose that this approach cannot be used at any level of structural decomposition where a schematic is used, so at subsystem or system level.

3 Defining the relationship between the structural and behavioural models

There are four layers in the structural/behvioural modelling diagram. At the bottom is the electrical schematic (a structural interaction diagram). Above that is the component structural model and above that the component behavioural model. Finally the new addition is a behavioural interaction layer. We need to define relationships between these layers. The principle here seems to be that

each layer only interacts with its immediate neighbours. Therefore the new layer should not cause any change in the current relationship between the component structural model and the ECAD schematic. Also there seems no reason to alter the existing relationship between a component's structural and behavioural models. The behavioural model should only detect changes local to the component and should only make changes in the structure of the actual component as is currently the case. Is it the case that this is currently defined, and if so is this formalism sufficient? The new layer does provide a new means for components to interact, and this should be limited to sending messages that might or might not cause a change in the receiver's state. I imagine that typically, behavioural level actions (such as sending a message) will be associated with transitions and structural level actions (such as changing an arc's resistance) will be associated with states.

The idea that the ECAD schematic and the new behavioural layer are both interaction diagrams suggests a alternative view of the proposed architecture. This is discussed briefly in the section discussing the proposed approach's relationship with functional modelling.

As the figure 1 suggests, the interaction layer in the state language remains part of the behavioural level modelling. The interface between behavioural and structural modelling will remain at the component level. The existing Component Builder tool can still be used to model a component's behaviour, but the modeller will be able to add actions that result in the sending of signals. Actions will still be able to alter the structural state of the component (i.e. change an arc's resistance value) and transitions can still be triggered by a change in the component's electrical state. A component's actions will not be able to change the state of another component (either its structure or behaviour) all it can do is request a change. This models the behaviour of a switch ECU, say, telling the network that the switch position is such that the headlamps should dip. Note that the switch cannot force the lamps to dip in such a system, as that behaviour will depend on the receiving ECU responding correctly (and indeed on its receiving the message at all!). Therefore we should not allow any component's behaviour to directly influence any other component's behaviour, so ensuring that the relationships are correctly modelled. This is a weakness of the earlier state builder proposal as this relationship is not properly defined in that the sending component directly drives a change of state in the receiver.

However, the behavioural interaction layer might be considered a second tier within the behavioural modelling part of the system. It seems possible (and interesting) to consider the message passing layer both as a second tier within the behavioural model, and also as a new class of component with no structural model. This allows the possibility of introducing failure modes for the message route. These will, of course, be modelled purely in terms of behviour, which might limit their "resolution", as failure modes will be along the lines of "message delayed", for example.

The use of the schematic defines ports through which components can send messages. This will model broadcast messages, as the component will simply send an unaddressed message. It should be possible to add message addressing in future, but it is suggested that at present all components with ports to the same network will receive all messages though, of course, they need not act upon them. This seems to strengthen the modelling of ports compared to the earlier Statebuilder proposal, and more so compared the idea of using output

properties, where any component can detect the output property of any other. The ports therefore mirror the use of pins to define where a component interact with the electrical system.

During discussions, Neal suggested a possible relationship between a component's structural and behavioural models, that might be illustrated by Figure 3.

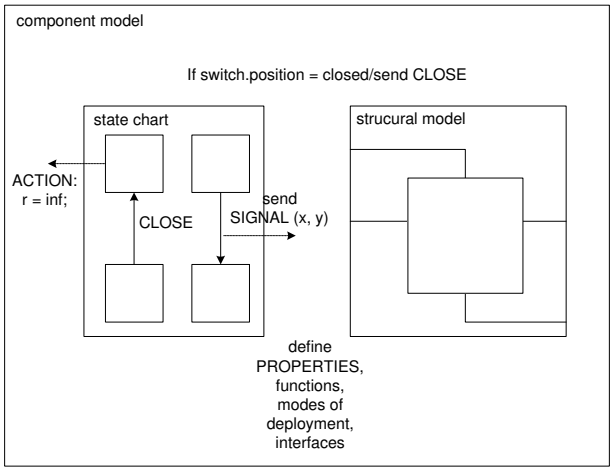

Figure 3: Different models of a component

The idea is to formalise relationships between the component's structural, behavioural and functional models. There seems to me no reason why this formalisation cannot be handled by existing tools and so be invisible to the user. This is perhaps a more theoretical question than need be addressed by the project itself, though it seems an interesting area of investigation for my PhD, especially as devising a more specified relationship between the models might help in demonstrating the system's usefulness in other domains.

4 Other state based languages

This section briefly discusses the proposed approach in relation to other state based languages, specifically SDL and Statemate. The proposed behavioural interaction layer is a close equivalent of the SDL block diagram or the Statemate "Activity Chart" in that it defines routes for signals. This raises the question "should we not use one of these languages?" There seem to be reasons for extending the existing language in preference, however.

- Neither SDL nor Statemate model broadcast messages. The proposed extension to State Builder allows the idea of a message being placed on a network and all other terminals receiving the message. This is a more correct model than can readily be built with SDL.

- SDL, in particular, has quite complex behaviour charts, which this approach avoids, even though we may need to complicate components' state charts slightly, as outlined above.
- The proposed extension seems reasonably simple to add - the interaction diagram itself is derived from the schematic, so we only need to add the modelling of signals to the state machine language. This is likely to be no more complex than adding interfaces to an alternative language.
- No need for user to use an alternative modelling tool, so the new facility is seamlessly integrated with the existing tool. It might well be feasible to make the other state based language invisible, at the expense of complicating the interface with that language.

Although this approach is intended to use the existing language, it may well be feasible to allow the network component (behavioural interaction layer) to be modelled using an alternative tool. Devising an interface between the AutoSteve component behavioural models and a network model using CANoe springs to mind. Other possibilities might include using code (or even hardware?) as the behavioural model of software based components.

5 Relationship with functional modelling

The new features are to be added at the behavioural level, so there might well be little call for change to AutoSteve's existing functional modelling. Function can still be related to components' output properties and system teleology. There is, perhaps scope here for some work on formalising the relationship between structure, behaviour and function. Figure 4 attempts to illustrate the relationships between these three sides of the model.

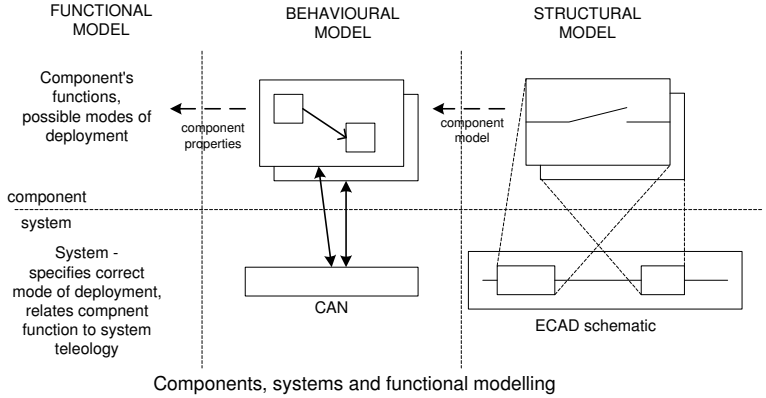

Figure 4: Adding functional models

Note that this diagram has been flattened. It is perhaps better to view the functional side as orthogonal to the structural and behavioural sides. This view also 'folds' the layers in the first diagram into 'component' and 'system' layers

that mirror relationships between structural, behavioural and functional models. The idea is that a component might have its own functional model, or models, associated with possible modes of deployment. The mode of deployment, and so the correct functional model will be specified in terms of the system.

This is an extension to the current approach, in which system functions map between teleology and component output properties. There seems no reason to suppose that the current approach cannot still be used. The questions of relationship between models are perhaps of a fairly theoretical nature, especially as it seems likely that the existing relationships can be used, and as such may be of more interest to my PhD than to the project itself.

6 Limitations and previous work

This section introduces limitations with the suggested approach and outlines its relationship with approaches suggested earlier in the project. This proposal fits somewhere between the two complementary approaches introduced in [3]. It could perhaps be argued that it bridges the gap between them, in that it's more sophisticated modelling of network signals extends the state builder based approach so that some modelling of network failures is now possible, and the idea that alternative tools be used to model the behavioural interaction layer (the network) allows it to fulfill one of the roles of the proposed model framework. Investigations have shown that the original state builder approach will work, if we model signals as instructions. This approach allows us to model data in signals. This avoids the difficulty with the earlier proposal that behaviour is moved from the correct component. For example, the sensor in the heater circuit need not send a signal "turn off heater", implying that it knows what the desired temperature is and also that it knows what to do about it. It can now simply send the value of the current temperature.

The main limitation of this approach is that it does not address the difficulties discussed in [2] - the need to model enough of the environment to give sensors something to measure and the need to find a way of modelling dynamic systems. It is suggested that these problems be addressed by some external model (if they are to be addressed at all) which replaces the model framework as a means of modelling such aspects of the system. This means that the model framework is replaced by this approach for its system modelling roles and by a possible 'external' model for its environment and time modelling roles.

We will clearly need to add some sort of temporal modelling if failures such as delayed messages are to be modelled. This might simply be done in terms of deadlines, and it seems possible that existing qualitative temporal labels could be used, though it is appreciated that a deadline could be missed by a delay modelled in terms of the same temporal label. Clearly if the deadline is specified in terms of microseconds, than a delay of milliseconds misses the deadline, but so might a delay of microseconds if there are enough of them.

7 Introducing other domains

This section is highly speculative, but it occurred to me that it is not impossible that a component's behavioural model can be used to define relationships be-

tween behaviours in different domains, allowing modelling of components that cross domains. The obvious examples are transducers, such as an electric motor converting current into torque. It seems possible to map a component's behavioural model onto more than one domain specific structural model, as illustrated in figure 5.

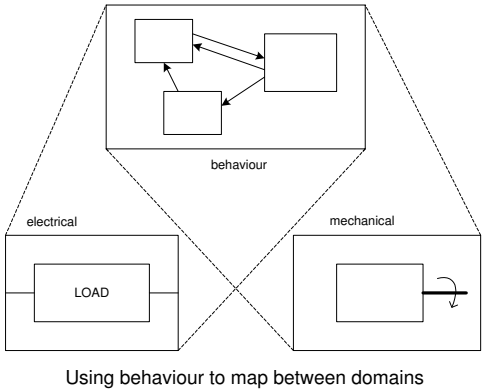

Figure 5: Component behaviour crossing domains

This provides some component level formalisation of relationships between the aspects of a component that relate to each domain, while avoiding the need for a qualitative physics. This implies the user being able to produce a sufficient mapping between domains via the behavioural model, of course.

8 Conclusions and future work

It is suggested that the proposed approach constitutes a sufficient enhancement of the earlier state builder approach that it can replace both it and the hitherto rather ill-defined model framework approach. It is not, however, a complete solution, as we still need some temporal modelling to allow message delays to be included. We might also want to add a facility for some modelling of the system's environment, though this is not currently to be regarded as a priority. The model framework had acquired two distinct roles in earlier reports. The idea that this approach removes its internal (system modelling) role avoids this confusion but we might still need some way of modelling external items. This external modelling tool might also be used for temporal modelling, though changes to the functional model will also be needed.

8.1 Future work

An early task will be to examine the existing state machine language to investigate adding the proposed extensions. It may well be worth trying to model some suitable example system (by hand) by way of further investigation of the usefulness of the proposed solution, and also to allow its working to be better demonstrated. There seems to be no reason why implementation should not start soon, following these investigations. It is perhaps worth considering

whether the original state builder based approach should be implemented as a first cut solution. It is worth noting that this solution does not build directly on that approach. Time spent implementing the composite state chart will not contribute to this solution. It may be useful as a simple way of starting work on the existing language, however, more or less as part of the investigation into changes necessary. We perhaps need more investigation into the limitations of this approach, and we certainly need to give more thought to temporal modelling might be included. The proposal suggested in [2] is of interest for modelling dynamic systems, though this is not a priority, but we certainly need to give more thought as to how to capture the effects of delays to messages.

References

[1] Jon Bell. Events and signals. SoftFMEA Document ref. SD/TR/FSM/03, 2002.

[2] Jon Bell. The heater circuit - matters arising. SoftFMEA internal report, 2002.

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