ABSTRACT

To design failure management expert systems, engineers mentally analyze the effects of failures and procedures as they propagate through device configurations. CONFIG is a generic device modeling tool for use in discrete event simulation, to support such analyses. CONFIG permits graphical modeling of device configurations and qualitative specification of local operating modes of device components. Computation requirements are reduced by focusing the level of component description on operating modes and failure modes, and specifying qualitative ranges of variables relative to mode transition boundaries. A time-step approach is avoided, and simulation processing occurs only when modes change or variables cross qualitative boundaries. Device models are built graphically, using components from libraries. Components are connected at ports by graphical relations that define data flow between components. The core of a component model is its state-transition diagram, which specifies modes of operation and transitions among them. Process statements describe state transitions and within-state processing, and have three parts: invocations (preconditions for effects execution), effects, and delays for each effect. A process language supports writing statements with several qualitative and quantitative syntaxes, including table lookups. CONFIG has been used to build device models in two domains, digital circuits and thermal systems.

REFERENCES

1. Harris, R.A. 1987. ON THE DESIGN OF A QUALITATIVE SIMULATION TOOL FOR ANALYZING BEHAVIOR OF ENGINEERED DEVICES. Nasa - Lyndon B. Johnson Space Center, Houston, TX 77058.


and some component models and relations from its associated library are shown in Figure 1.

COMPONENT DEFINITION

Computation and specification requirements are reduced by focussing the level of component description on operating modes and failure modes, and specifying qualitative ranges of component variables relative to mode transition boundaries.

A component model can be viewed both as a composite of modes and as a composite of ports. The core of a component model is its state-transition diagram, which graphically specifies modes of operation (both normal and failed) and the transitions among them. State transitions and within-state variable transformations are specified as processes. The other decomposition of a component model is into its ports. Ports designate component inputs and outputs. Relations connect ports for data propagation between components. Examples of both types of decomposition are shown in Figure 2. The boxed area on the right of the figure is a decomposition of the boxed area in Figure 1.

DISCRETE EVENT SIMULATION APPROACH

Using an approach similar to Pan [3], discrete events are defined at the level of changes in operating modes. A time-step approach is avoided, and simulation processing need occur only when modes change or variables cross qualitative boundaries.

Rather than constraint propagation, discrete event processes determine the consequences of component changes. The event structure controls the propagation of behavior changes among components. Scheduled events pass data between ports, change variable values, and make state transitions. The primary event is the update of a component, which is triggered by a change in an input variable, local variable, or component state. In such an event, appropriate processes are inspected, and the effects of invoked processes are scheduled with corresponding delays. Updates originating from many components can be scheduled at the same time on the discrete event clock.

PROCESSES AND PROCESS LANGUAGE CONSTRUCTOR

There are three kinds of processes, mode independent processes, mode dependent processes, and mode transition processes (which are actually also mode dependent processes). Processes consist of three parts: invocations (preconditions for effects execution), effects (executed if all invocations are satisfied), and delays corresponding to each effect (effect completions scheduled at increments to the current time). Invocations and effects are defined in terms of variables, modes, and processes.

A key capability is a process language constructor and interpreter that permits process statements to be written with an array of qualitative and quantitative syntaxes, including table lookups. A process language interprets statements that define invocations and effects of processes. The language constructor supports experimenting with representations, by permitting the definition of data-structure types and operators.

Component variables can be specified quantitatively or qualitatively, but a small number of qualitative ranges is desirable (e.g., abnormal-low, low, medium, high, abnormal-high). Continuous behavior is partitioned into trends and breakpoints. The durations of trends can be represented "qualitatively" as an approximate order of magnitude, which is translated into an interval on the discrete event clock. Trend effects are represented by scheduling a process for the end of a trend, at which time the duration of the trend is confirmed before executing the effect.

Examples of processes, data structures and operators from the thermal library are shown in Figures 3 and 4.

CONCLUSION

CONFIG has been used to build device models in two domains, digital circuits and thermal systems. The CONFIG project is ongoing, with plans for additional capabilities beyond the current modeling capabilities. These additional capabilities will include support for analysis of failure effects, and support for development of differential diagnostic measures and tests. An additional goal is to support incremental development of quantitative process simulation algorithms.

ACKNOWLEDGEMENT

The authors wish to thank Jodi Seaborn for developing the thermal library and thermal bus model using CONFIG.

REFERENCES


State Diagram for Pumps

State Diagram for Evaporators

Decomposition of Evaporator.213, Pipe.207 Regenerator.202, and Pipe.209

Figure 2 - Thermal Component Decompositions
Process with Delays

Delay Constants

Figure 3 - Thermal Language Data Structures
Figure 4 - Thermal Language Operators