Enhancing Understanding of Model Behavior Through Collaborative Interactions: Explaining Why

C. Michael Overstreet
Irwin B. Levinstein

Computer Science Dept.
Old Dominion University
Norfolk, Virginia USA

Funded by Lockheed Martin, Norfolk, VA USA

Outline

- Objectives
- Context of Study
- Graphical representations used
  - Sample output
- Implementation details
- Concerns
  - Scalability
- Some issues in automated explanations
- Summary

Objectives

- Assist modelers in understanding the causes for model actions
  - Can help identify some types specification & coding errors
  - Can enhance understanding of system being studied
  - Animate graphical version of specification
- Minor
  - How hard to retrofit specification animations into large existing simulation not developed with this objective
Two Parts of Talk

- Describe work incorporating graphically based explanations
  - Feasibility assessment
  - Concept project only
- Conjectures on extensions of this work
  - What’s hard, what’s easy

Context - 1

- Based on large military simulation used for training
  - One of several large simulation training tools developed by US Department of Defense
  - Runs hundreds on hundreds of widely distributed workstations
  - Take simulation to participants rather than vice-versa
- Now called OneSAF
  - http://www.onesaf.org

Context - 2

- Project built using DRE (Distributed Research Environment)
  - Based on ModSaf (MODular Semi-Automated Forces), an earlier version of OneSAF
  - Was intended to be provided free to researchers
    - Lots of interesting problems for PhD students
- Roughly 800K LOC in 2700 source files
  - Essentially C and but some in specialized language (finite state machine)
Context - 3

How does one determine if an executable is correct?  
- Does the model behave as intended?  
- Does the code correctly implement the model?  

Obviously hard; believe some help provided by:  
- Adding explanations of behaviors to existing large simulation by showing causes  
- Using abstraction to eliminate unimportant and distracting details

Explain Causes Graphically

- Animate model representation (to reduce complexity)  
  - Omit distracting details  
- Many possible graphical representation possible:  
  - e.g., activity cycle diagrams, event graphs, hierarchical control flow graphs, action cluster diagrams, finite state machines (FSMs), flow charts, UML

ModSAF uses FSMs

- Sometimes seems a good match, sometimes not  
- In this code, their quality and benefit depended on individual programmer's skill  
  - Some people seemed not to understand FSMs  
- Sometimes behavior too simple; FSMs add nothing  
- Sometimes behavior too complex; FSM seems not to help
Hierarchical Finite State Machines (HSM)

- Harel added hierarchy to Finite State Machines
- Introduces "superstates" which consist of several substates & support concurrency
- Can significantly reduce complexity of the graph by reducing the number of transition edges
- More easily understood? You judge

State Chart Version

- Adding 3 super states greatly reduces edge count
- Indicates default substate entered when superstate is entered
Advantages/Disadvantages of Graphical Representations?

Venn Diagrams

Venn Diagrams for n > 3 sets

Finite State Machines in ModSAF

- Parts of model are specified using Finite State Machine (FSM) language
- The “vehicle move” FSM about medium complexity of the 100’s used
- Previous FSM slide generated directly for the vehicle move code
- C code is also generated directly from FSM code
Conjecture

- "Simple" FSMs are useful
- "Complex" FSMs are difficult for people to assess
- Graphics is helpful when I don't need help, but fails when I do
  - Better than alternatives?
- Likewise "simple" HSMs are helpful
- With complexity, questionable

Project used “BetterState” for Hierarchical FSMs

- Commercial software
- Used to construct HSMs graphically
  - We based them on the existing FSMs
- Can generate implementation (e.g. in C++) directly from HSMs
- Can animate HSMs while the generated code runs
- Can capture events for later replay

Implementation Details

- Vehicle move FSM file is about 2000 lines
- FSM file is mixture of C and FSM specific code
  - Mostly C
  - Also defines Events, States, and Transitions
    - Events here are really Booleans:
      - e.g. if we're in a moving state and "detect" becomes true, we transition to disabled state
- Translated by AWK into C
  - Translator about 1500 lines of AWK
  - We added about 20 lines to add animation ability
- Translated is about 2400 lines of C
Benefits of FSMs or other specification tools

Previous slide implicitly implied that size is major metric in evaluating the benefits of FSMs vs. C as a model specification tool
But their main contribution may be that they provide a way to describe system behavior:

- What are the states?
- What are the events that cause state transitions?
- What are the transitions?

Architecture of Implementation

- Network of many machines runs simulation
- Code in some machines modified to report state changes of some objects
- A new "explanation" machine listens for these state changes and displays these explanations as the simulation runs

Explanation Machine Display

- Presents simple animation of vehicle behavior (next slide)
  - Observer sees what a vehicle is doing
- Also displays causes of what vehicle is doing in 2 forms:
  - An HSM display which animates state changes as they occur
  - Another window displays the boolean condition that cause the state transition
- Animated state changes and text intended to helps observer understand the why of behaviors
Comment

- One often cited benefit of conducting a simulation is the improved **intuition** gained by the modeler as he/she builds and revises a simulation
  - Often cited as more important than the data produced by simulation
  - Explanations of model actions can be useful of enhancing this benefit

Results – Animation Easy

- Provided we could change requirements based on what was easy
- Provided we started with something very close to HSMs
- Several potentially desirable features not implemented
  - AWK ok for simple transformations, but really need to parse code for some changes
    - We found the if statement that controlled a state change and added a function call to send info to the explanation machine
    - Doesn't work for complex logic, e.g., nested ifs
  - We displayed the Boolean expression of the if as text for an explanation.
    - For this case, it worked reasonably well but depends on programmers picking informative variable names
Concerns - Scaling

- What if lots of entities and a long running simulation (these are usually close to real-time because of training objectives)
- Add ability for observer to change focus of attention
- Add "traps" so that observer can specify interesting situations that are displayed if they occur
  - Replay, discussed below, also useful for after the fact analysis

Causal Sequences

- Often interesting (surprising?) model actions are the result of a sequence of events
- When interesting actions occur, identify the causes of the action. These may have occurred at a time prior to the current action
  - Likewise these prior actions also have one or more causes.
- Support interactive exploration of the chain of causes that led to the interesting situation

Causal Sequences and Program Slicing

- Similar to Weiser’s program slicing concept proposed to enhance programmers’ understanding of code
  - Tool displays the subset of a program that effects what the programmer is trying to understand
  - Weiser’s idea seemed simple but leads to surprising complexities
    - Due both to computational complexity issues and problems from aliased variables
      - Alias: two names for the same memory location
      - Associated with pointers, arrays, parameters
Causes in Discrete Event Simulations

- For discrete event simulations, each event is either explicitly scheduled or is the result of a state change
  - The scheduler of each event easily recorded in a trace; only one scheduler for each event (usually)
  - State-induced events may depend on values of several variables & have at least one immediate cause but may have several other causes
    - An immediate cause must have occurred before this event and without the simulation clock changing
    - Other causes that occurred earlier in simulation time can be harder to identify and require data flow analysis
      - Aliases can make this tricky to determine

Summary

- We believe that allowing modelers, users of models and programmers to interactively explore the reasons for some model actions can be beneficial to each group, with different benefits for each
- Causes for many model actions can be easily obtained; others raise some interesting computational issues