Lecture 6
Operational Specifications

Kenneth M. Anderson
Foundations of Software Engineering
CSCI 5828 - Spring Semester, 2000

Today’s Lecture

• Continue to discuss the Make example
  – It illustrates each of the three specification styles introduced in lecture 5
• Begin to explore Operational Specifications in more detail

The Make Example

• Lecture 5
  – We worked on an example specifying some properties of Make
• However, Make is a specification language itself
  – It specifies dependencies between artifacts
  – It specifies rules for creating new artifacts
  – It specifies actions to carry out the rules

Make Specification Language

• Dependencies are Relational
  – Described according to desired relationships
  – Usually given in terms of multi/hyper graphs
• Rules are Declarative
  – Described according to desired properties
  – Usually given in terms of axioms or algebras
• Actions are Imperative
  – Described according to desired actions
  – Usually given in terms of an execution model
More on Make

• Make is well-integrated into a Unix/C environment
  – Primitive Components are Files
  – Actions are “shell commands”
  – Rules are placed in a file and denote the “specification”
    • Rules make explicit the dependencies of the system and what to do about them

Example “Makefile”

<table>
<thead>
<tr>
<th>Target</th>
<th>Actions</th>
<th>Dependencies</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>T2 T3 T4</td>
<td>A1 A2 A3</td>
</tr>
<tr>
<td>T2</td>
<td>T5 T6</td>
<td>A4</td>
</tr>
<tr>
<td>T3</td>
<td>T5 T7</td>
<td>A5 A6</td>
</tr>
</tbody>
</table>

Rules can have interdependencies

… and shared dependencies!

Questions

• What is the concept of dependence in this system? How is it modeled?
• Why are rules considered declarative?
Hybrid Style Issues

• Consider programming languages
  – They are primarily operational
    • What about them are declarative or relational?
• Most languages will have a chief modeling style
  – Contrast statements in a program with Make’s
    • S1 S2 S3… operational, do these statements in this order
    • Rules in a makefile: declarative, achieve this target
  – One style will lead you to ask different sorts of questions than with another style
    • Is there a unique way to achieve the target? Is a target feasible?

Operational Specification

• Focuses on Control Aspects
  – Here we choose to look at control issues rather than data issues
• Examples
  – Control the flight path of an airplane
  – Control the speed of a car
• Of course, there are data aspects to these problems. However we view them more as parameters that influence the actions of the system

Formalisms and Foundations

• Formalisms
  – Finite State Machines (FSMs)
  – Petri Nets
  – Statecharts - used in UML
  – Communicating Sequential Processes (CSP)
    • Latter three are different attempts to add concurrency to FSMs
• Mathematical Foundations
  – Graph theory, automata theory, modal logic

Preview of Finite State Machines

• Informal Problem Description
  – When turned on by the driver, a cruise-control system automatically maintains the speed of a car over varying terrain. When the brake is applied, the system must relinquish speed control until told to resume. The system must also steadily increase or decrease speed to reach a new maintenance speed when directed to do so by the driver.
Example Continued

• There are seven inputs:
  – System on/off: If on, denotes that the cruise-control system should maintain the car speed.
  – Engine on/off: If on, denotes that the car engine is turned on; the cruise-control system is only active if the engine is on.
  – Pulses from wheel: A pulse is sent for every revolution of the wheel.
  – Accelerator: Indication of how far the accelerator has been pressed. Note: The accelerator does not turn off the cruise-control system, it “pauses” the system.
  – Brake: On when the brake is pressed; the cruise-control system temporarily reverts to manual control if the brake is pressed.
  – Increase/Decrease Speed: Increase or decrease the maintained speed; only applicable if the cruise-control system is on.
  – Resume: Resume the last maintained speed; only applicable if the cruise-control system is on.

In-class Example

• We will now develop a finite state machine to help formalize the problem description
• Method
  – Identify states
  – Identify transitions between states
  – Keep it simple, if it starts to get too complex, we are heading down the wrong path
• <See class video for rest of example.>