Lecture M1: Overview and Introduction

Overview of the course

The course is based on 4 parallel streams of activity:

- Case studies:
  - Vehicle Cruise Controller Simulation (VCCS)
  - Railway Station Animation
  - Digital Watch Simulation
- Tutorial Introduction to Software Tools
- Practical exercises, based on a cricket simulation
- Seminars

Timetable & Introductions

Nature of the assigments for the course: see Coursework

Sources:

- papers
  - [] and references therein
  - Booch, Deutsch, Backus, Harel, Cantwell-Smith, Baldwin
- books
  - Lamb, Davis
- software
  - public files in the ~wmb/public/demo directory
  - sources for the translators and interpreters
- videos for cricket ??
- documentation
  - eden and scout, theses

Begin by looking at VCCS: use this as a convenient place from which to explore abstract issues (what are we doing? and why are we doing it?) and practical concerns (how do we construct our models? and what tools have we developed for the purpose?). Seminars and analysis of case studies address abstract issues; description of tools, illustrations and exercises address practical aspects.

<Picture of VCCS in relation to top-down and bottom-up perspectives:
why? how? and what??>

Caveat: won't always (often?) seem that we are directly talking about the issues of concurrency or conventional programming. Underlying theme of this module is that the problems of general-purpose parallel programming are deeply bound up with fundamental issues and concepts in concurrent systems modelling. What is more, these issues cannot be satisfactorily addressed in the traditional mathematical manner of "abstracting away from the application". Our subject matter is the relationship between form and content, and content involves real-world illustrations. Hence the electronic cat- flap, the railway station animation, the room layout, the cricket simulation etc. Understanding these examples leads to a new perspective on programming in general and parallel programming in particular.

M1.1. Introduction to the Vehicle Cruise Control Simulation (VCCS)

Software case study initially proposed by Booch [] and further studied by Deutsch. Our version developed by Ian Bridge and Simon Yung []. Available in ~wmb/public/demo/EDEN/

cruise. Run via demo.cruise. Uses family of Eden files referenced by main.e, and visualisation and interface file cruise.s.

Two ways to view. Animation of requirements for reactive system. Exercise in engineering design and simulation.

Animation involves:
- identifying state-changing agents in the VCCS
- describing their roles
- creating an interface that allows the designer to play the role of the driver, simulating operation of brake, accelerator and cruise controller

Construct and animate a model of the concurrent action of the agents.

Components of the animation display ....
- vehicle, with forces acting on it
- speedometer
- cruise control interface
- brake and accelerator
- position of vehicle on road
- clock, displaying time elapsed in simulation
- internal throttle position, indicating effects of use of accelerator and action of the automatic throttle

Functions performed in the simulation
- dynamic model of the vehicle, based on environmental forces and power output of the engine
- controller with simple feedback mechanism
- switches and buttons for the driver to operate

More significant than what simulation does is how it is defined:
- simulation was developed incrementally
- dynamic model and visualisation were independently developed
- designer can make further changes easily
- model has explanatory power, allowing designer to explore the roles of the component agents and the way in which they interact.

Key idea: modelling that is based upon observation and experiment.

M1.2. Nature and structure of the VCCS specification

Propotyping in VCCS is based on 3 software tools developed at Warwick:
eden an evaluator / engine for definitive notations [YWY]
donald a definitive notation for line-drawing
scout a definitive notation for text and window layout [YPY]
eden is an interpreter for a general-purpose programming language.
donald and scout are translators that act as preprocessors for eden.

A **definitive notation** is a simple formal language in which to formulate scripts of definitions of the form:
variable = function of other variables and constants.
[We use the term definition in a technical sense to be clarified in the course, and use (abuse?) the term definitive to mean "definition-based".]
Definitive notations are distinguished by the types of the variables on the LHS, and the operators that can occur on the RHS. The values of the variables typically correspond to observable quantities. E.g. donald variables define points and lines in the plane, and a donald script defines a 2d line-drawing. A definitive script is viewed as defining a state: to change the state, we typically redefine a variable in the script, or else introduce a new definition (see illustration in §1.3 below).

Most of our software prototyping uses a UNIX pipeline:
scout <inputfile> | donald | eden
The eden interpreter interfaces to X windows via an Eden-X interface EX.
eden and EX communicate via the UNIX IPC message queue mechanism.

In the cruise.s file, scout, donald and eden definitions are differentiated via the %scout, %donald and %eden annotations.

Examples of scout and donald extracts from the script:

< the SPEEDO window >

<the speedometer specification>

M1.3. Concepts behind the development of the VCCS

There are several ingredients in the development of an animation:

- analysis of the application to identify the participating agents
- identification of the observations needed to specify the interaction between the family of agents
- representation of the role of agents using definitive scripts to describe the way in which observations are indivisibly linked in change
- animation of observations and agent interactions within a computer model by visualisation and interface creation.

The concept behind the modelling technique resembles conventional engineering: perform experiments on components in isolation, observing how they respond; on this basis, predict how the entire system will behave when all components are combined. Analogy between experimental apparatus and measuring instruments that make observations accessible to the experimentor and computer model that is constructed in VCCS. Harel's observation concerning the importance of visualisation is relevant here [].

The techniques used are

agent-oriented modelling + definitive representation of state.

The agent-oriented modelling process makes use of a special-purpose notation LSD in which to represent the interfaces between agents. Definitive representation of state is expressed via scout, donald and eden definitions. The end-result is an eden program, but conceptually the design is largely specified at a higher-level of abstraction.

The LSD specification in [] is animated in main.e and associated files. This describes the dynamic model of the vehicle.
The screen display is described by the scout and donald scripts in scout.s. These enable the designer to inspect the state of the interface as seen by the driver as well as the behaviour of the cruise controller and the state of the dynamic model.

Important distinction between the VCCS specification and a logical model of the circumscribed behaviour of the system. VCCS specification is not made up of universally valid statements about the entire state-space of the simulation. Instead it is (to some approximation) a representation of a particular state – resembling a set of experimental observations – that incorporates information about the expected effect of changing a parameter in the experiment. This reflects a fundamental distinction between incomplete and uncertain knowledge that *is being acquired* through experiment and comprehensive and reliable knowledge that *has been acquired* from experiment.

M1.4. Use and Development of the VCCS Script

Use the term "observation" to refer to any value in a physical system being modelled that can be measured by some appropriate experiment. Designer has total knowledge of the state of the system in terms of all such observations, and can consider the implications of changing parameters associated with observations. For instance: change sampling rate of speed transducer, modify the layout of the speedometer, or the mode of visualisation etc. In principle, the designer can treat every observation as a parameter, but in practice it doesn't make much sense to do so. For example, don't usually change the force of gravity, or make the length of the vehicle inversely proportional to its speed.

Definitive script for VCCS is a record of all these observations, as represented by variables in the script. Redefinition of a variable is the activity that corresponds to "changing a parameter". In principle, any redefinition is interpretable insofar as it assigns new values to variables recorded in the script, but only certain redefinitions are meaningful.

The VCCS script is a means to express many different views of the vehicle cruise control system. When we refer to the *state* of the system, we generally have the comprehensive set of observations conceived by the designer in mind (the "global state"). The concept of state is much more subtle than even this suggests. State is to be understood relative to a particular agent and to context for observation. For instance, by the *state* of the screen display a designer might be referring only to those aspects of the display that concern layout of the windows and form of the graphical objects. The relevant observations in this context are the scout and donald definitions in cruise.s. On the other hand, the simulation can be observed over time, and the state described in terms of how variables change over time. In this context, the model is enriched by consideration of the analogue variables represented continuously changing quantities, such as the speed and position of the vehicle.

Characteristic set of observations associated with each agent determines the state of the system relative to that agent. The actions of an agent are associated with privileges to change certain parameters. These are modelled by redefinitions.

The actions of agents affect the global state of the system in ways that are modelled by redefinition in the VCCS script. Illustrations:

How the speedometer script is to be interpreted: the designer agent.
How forces act in the model:
    time = time' + 1
    Resultant = f(Resultant')
Driver operates buttons, applies brake

References

(BBY) Agent-oriented Modelling for a Vehicle Cruise Control System
Booch
Deutsch
Harel

YWY EDEN
YPY Scout

Follow up: T1 for more technical details
Programming as Modelling for more discussion of principles

Useful material for elsewhere:

State cf mode (p32 Lamb)

mode = set of states

1. Externally visible behaviour of the system differs from one mode to another
2. The system moves from one mode to another when certain externally visible events happen.