Lecture M1: Overview and Introduction

Overview of the module

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

Administrative issues

Timetable

Introductions to Personnel

Assignments

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

• documentation
  eden and scout, theses

Definitive Principles for Modelling and Programming?

top-down

Seminars

WHY? motivation & philosophy

issues

VCCS as a case study

WHAT?

techniques

HOW? software prototypes

Tutorials

bottom-up

Themes of the Module

problems of general-purpose parallel programming
bound up with fundamental issues and concepts
in concurrent systems modelling

can't be properly addressed
in the traditional mathematical manner by
"abstracting away from the application"

concerned with relationship between form and content
and content involves real-world illustrations:
the electronic cat- flap, the railway station animation
the room layout, the cricket simulation etc.

get new perspective on programming in general
and parallel programming in particular

BUT – warning!

- won't always directly address
  concurrency / conventional programming

M1.1.

Introducing the Vehicle Cruise Control Simulation (VCCS)

Software case-study considered by Booch and Deutsch

Our version due to Ian Bridge and Simon Yung

Source available in ~wmb/public/demo/EDEN/cruise

Run via

```
demo.cruise
```

Dynamic model of vehicle <--> family of Eden files (ref main.e)
Visualisation and interface file is cruise.s.

Two ways to view:

Animation of requirements for reactive system.

Exercise in engineering design and simulation.

Construct and animate agent-oriented model by:

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

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 evolving:**

- simulation was developed incrementally
- dynamic model and visualisation were independently developed
- designer can make further changes easily
- model has explanatory power

  designer can explore the roles of the component agents and the way in which they interact.

Key idea: modelling based upon observation and experiment.

M1.2. Nature and structure of the VCCS specification

Prototyping 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 = interpreter for a general-purpose programming language.

donald and scout = translators that act as preprocessors for eden.

A **definitive notation**

is a simple formal language

used for expressing scripts of definitions of the form:

variable = function of other variables and constants.

Warning! *definition* ia a technical term in this module

and we (ab?)use the term *definitive* to mean "definition-based"

Definitive notations characterised by

- the types of the variables on the LHS
- the operators that can occur on the RHS

Values of the variables <--> observable quantities

E.g. in donald – a definitive notation for line drawing

- variables define points and lines in the plane
- donald script defines a 2d line-drawing

Basic principle

A definitive script is viewed as defining a **state**:

to change the state, typically

redefine a variable in the script, or else

introduce a new definition

The prototyping tools

Most of our software prototyping uses a UNIX pipeline:

```
scout <inputfile> | donald | eden
```

eden interpreter interfaces to X windows via Eden-X interface EX.

eden <--> EX link uses UNIX IPC message queue mechanism.

In the cruise.s file, scout, donald and eden definitions

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

Ingredients in the development of an animation:

- analyse application to identify the agents
- identify observations needed to specify the interactions amongst the family of agents
- represent the role of agents using definitive scripts to describe the way in which observations are indivisibly linked in change
- animate observations and agent interactions within a computer model by visualisation and interface creation.

cf conventional engineering prototyping:

- perform experiments on components in isolation,
- observe how they respond

then predict how system will behave when compts are combined.

Nature of the VCCS computer model ....

computer model in VCCS

analogous to

experimental apparatus and measuring instruments

that make observations accessible to the experimentor

Techniques for construction

Techniques used

agent-oriented modelling + definitive representation of state.

agent-oriented modelling .....

uses a special-purpose notation LSD

to represent the interfaces between agents

definitive representation of state .....

uses scout, donald and eden definitive scripts

..... VCCS model = eden program, but conceptually

design is specified *at a higher-level of abstraction*

About the VCCS

LSD specification 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 driver interface
- the behaviour of the cruise controller
- the state of the dynamic model.

What is the VCCS specification?

VCCS specification
≠ logical model of the circumscribed system behaviour

VCCS specification

= an approximate representation of a particular state –

resembling a set of experimental observations –

incorporating information about the expected effect

of changing a parameter in the experiment.

Concerned with

incomplete and uncertain knowledge

that *is being acquired* through experiment

and not

comprehensive and reliable knowledge

that *has been acquired* from experiment.

M1.4. Use and Development of the VCCS Script

"observation"

= value in a physical system being modelled

in principle measurable by some appropriate experiment.

Designer

- knows state of the system in terms of all such observations
- considers effect of changing parameters <--> observations

[Important distinction between

what agent can **observe**, and what agent can **change**]

Designer experiments:

change sampling rate of speed transducer

modify the layout of the speedometer

change mode of speed visualisation etc.

An important parallel

In principle – even in real engineering:

designer can treat *every* observation as a parameter

In practice 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

= a record of all observations of the system,

represented by variables in the script.

Redefinition of a variable = "changing a parameter".

In principle, any redefinition is interpretable
but only certain redefinitions are meaningful.

Views and States

VCCS script is a *means* to expressing knowledge
    developing the script is not a completed process

Can be used and viewed in many ways, not nec preconceived

e.g. many different views of the vehicle cruise control system.

Default: *state* of the system (the "global state")
= comprehensive set of observations conceived by the designer

But
    state to be understood relative to
        a particular *agent* and to *context for observation*

e.g. IS *state* of the screen display
    = layout of the windows and form of the graphical objects?

    IF SO, relevant observations are scout / donald definitions

OR is state of the display conceived via behaviour over time

    IF SO, consider speed and position of the vehicle etc.

Agent-oriented modelling for views

Characteristic set of observations associated with each agent
determines the state of the system relative to that agent.

Modelled by set of variables in a definitive script

Actions of an agent <--> privileges to change certain parameters

Modelled by redefinitions

Illustrations:

How the speedometer script is interpreted: the designer agent.

How forces act in the model:

time = time' + 1

Resultant = f(Resultant')

Driver operates buttons, applies brake

Picture the course from the perspective of

agent-oriented modelling	definitive representation of state
LSD / ADM	SCOUT, DONALD
animation	visualisation
concurrent systems modelling	EDEN

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.

Some of the ideas so new they haven't been thought up yet

In my experience the fact that Simon and Ian can do it doesn't prove that it's possible

Simon's idea of documentation is giving you the BNF grammar