CS405 Introduction to Empirical Modelling
Lab 6
SEMI Aspects of State and Distributed models

1 Introduction

This week we are looking SEMI aspects of state and models that use the distributed version of tkeden, called dtkeden. The dtkeden tool allows us to create models that support multiple agents across two or more computers. For an introduction and manual for dtkeden see "A Not-So-Quick Guide to dtkeden".

2 Digital Watch model illustrating SEMI aspects of state

Note: you may want to do this exercise outside the lab, and do the group exercises now.

Look at Chris Roe's digital watch (digitalwatchRoe2001) which can be found in the EM repository `/dcs/emp/empublic/projects/digitalwatchRoe2001`. Execute the model using `Run.e`. There are some simple questions at the bottom (advance to the next question by redefining `questionNo` to 2, 3 or 4). To make the watch tick - introduce "input = 8;". Carry out the following modifications:

- improve the "updating time" functionality at the interface (as illustrated in the file `example1.e`)
- add the functionality of the extra clock (as illustrated in the file `example2.e`)
- check that you know how to use the watch to time two runners (after inputting the extension file `r2.d`)

The README file should give further details of this model.

3 Distributed models of jugs

To get started with dtkeden we'll look at a simple distributed model of jugs, comprising one or more clients (with the graphical display of the jugs) and a server that monitors the values of several observables. You will need at least two computers for this exercise. Find the model at `/dcs/acad/wmb/public/cs405/EMlabs/lab6/models/jugs`.
To run jugs in client-server mode, you need to follow these instructions. Nominate one computer the "server". You will use this to monitor the jugs models of all the clients.

- Run "`dtkeden -s -c XX`" where XX is a channel number.
- Include `jugs.include`.
- Input "`init_cells();`"
- Input "`ORACLE_LIST = ["capA", "capB", "input", "contentA","contentB"];`"

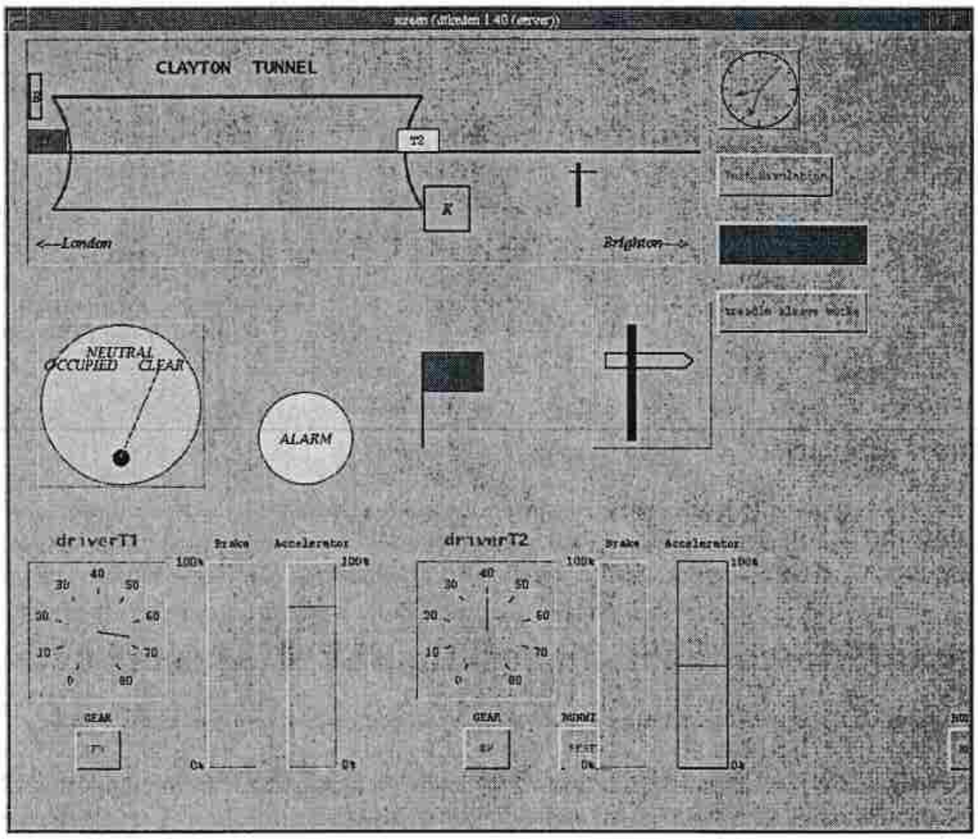

Figure 1: Clayton tunnel model.

- Set type to "private" mode

You should be able to connect multiple clients to the server by following these instructions:

- Run "`dtkeden -a -c XX -h hostname`" (with the correct hostname for the server, and XX as the same channel number)
- Input agent name
- Include `jugs.e`, `jugs.s` and `jugs.button.e`
- Input "`init_game();`"

Use the input window at the clients to change the values of `capA`, `capB` or `input` directly, or exercise the model using the Scout window buttons. Experiment with the Jugs models, and see if changes at the clients are reflected in the server.

4 Clayton tunnel railway disaster

This model is fairly elaborate so it will require several people and computers to run it. The model allows for a re-creation of the Clayton Tunnel railway accident presented in the lectures. Find the model in the repository at `/dcs/emp/empublic/projects/claytontunnelSun1999/`.

You will need several computers. These represent the views of several different agents: (1) God, (2) Brown (although you can run with on the same computer as God), (3) Killick1, (4) Killick2, (5) train driver 1, (6) train driver 2 and (7) train driver 3.

To set up the "server" for the model execute:

```
# dtkeden -s -c 42 God.3
```

To introduce Brown, execute:

```
# dtkeden -a -c 42 -h serverhostname Brown
```

You can do this on the same machine as God.
Then on the other workstations (with *serverhostname* as the name of server), execute:

```
# dtkeden -a -c 42 -h serverhostname Killick1
# dtkeden -a -c 42 -h serverhostname Killick2
```

Finally execute:

```
# dtkeden -a -c 42 -h serverhostname driverT1
# dtkeden -a -c 42 -h serverhostname driverT2
# dtkeden -a -c 42 -h serverhostname driverT3
```

When you have set up the model you should be able to use it to simulate normal use and to re-create the railway accident. Refer to the handouts from the lectures and ask the demonstrators for help.
Some points to consider when interacting with the model:

- You can alter the reliability of the treadle by setting the real-valued Eden variable `treadle_reliability` to a value between 0.0 (always fails) and 1.0 (completely reliable).
- A train can only be put into reverse after it has been brought to rest. It can be started and reset at its initial location by pressing the "START/RESET" button.
- The entire model can be reset using the "Init. Simulation" button.

5 OXO game and the broadcast mode

The dtkeden guide proposes that broadcast mode is suitable for multi-user games. Investigate this by trying to write a distributed OXO model using the broadcast mode. (The resources for such an exercise can be found in `/dcs/emp/empublic/projects/oxoJoy1994` on which the OXO laboratory described in lecture "M2: EM for the single agent" is based. [1])

[1] Answers on a postcard PLEASE!