Appendix B

Definitive models referenced in the thesis

{Car94} The Car History model: The Car History model was developed by James Wood [Wood94] as his final year undergraduate project in 1994. The discussion of this model appears in Chapter 3 and is used as a case study model for the tools implemented by the author in Chapter 7.

{Class95} The Classroom Interaction model: The Classroom Interaction model was implemented by Emma Davis in 1995 [Davis95] in conjunction with the analysis of the interaction between pupils and the interaction between pupils and teacher studied by Sean Neil. The model is referred to in Chapter 3.

{Digital92} The Digital Watch model: This model was originally developed in 1992 by Beynon and further developed by Cartwright in 1995 [BC95] and Fischer in 2000. These three contributors constructed this model in separate phases of work over an 8-year period with minimal communication. The model simulates basic characteristics of a digital watch with an additional statechart diagram [Har87] to give a visualisation of state-change in the internal mechanism of the watch.

{GUI96} The Generic User Interface model: The model was implemented by Yvonne Lui [Lui96] as her final year undergraduate project in 1996. The study of this model in Chapter 3 is entirely based on her original model.

{Heap98} The Heapsort model: The Heapsort model (without the formal specification) was originally implemented by Beynon in 1998. The original version can sort lists of numbers up to seven elements. The author has extended the model based on the original generic script so it can sort more numbers. The version referred to in Chapter 6 is based on sorting numbers up to fifteen elements. Many variants of this version have been implemented by the author and discussed in Chapter 6. These include the Heapsort model with the formal specification.

{Jugs92} The Jugs model: The Jugs model was originally developed by Beynon and Russ to replicate the behaviour of a procedural program for the jugs program written by Townsend [Town]. It was further developed by Yung in 1992 with the use of Scout windows and interfaces. It has been widely used as an illustration of characteristics of modelling with definitive scripts.

For the purpose of the exposition of MWDS in this thesis, many variants of the original Jugs model have been developed by the author (cf. Chapter 2 and Chapter 5).

{Lines91} The Lines model: The Lines model was first developed by Beynon in 1991. The original version makes use of the ARCA notation, but once DoNaLD was developed the ARCA script was transformed into DoNaLD and Eden scripts. The version referred to in Chapter 3 is based on the original version (with using ARCA).

{MBF99} The Monotone Boolean Function (MBF) model: The Monotone boolean function is developed by the author herself to illustrate model building based on a single-agent perspective discussed in Chapter 3.

{Number02} The Number model: The Number model was developed by the author to illustrate model building based on a single-agent perspective. Many variants of the original model have been implemented and discussed in Chapter 2.

{Rail99} The Railway Accident model: The Railway Accident model was originally developed by Patrick Sun in 1999, based on the scenario of the Clayton Tunnel railway accident in 1861 as described in Box B-1 below. It was used as a case study model for the dtkeden tool implemented by him. The model is referred to in Chapter 3.

{Room90} The Room Viewer model: The Room Viewer model was implemented by Edward Yung [Yung90] to illustrate the use of definitive principles in representing simple geometrical elements. The model has been developed into two versions: one with a Scout user interface and the other without. Its script is very simple and easy to understand. The version used in Chapter 3 is the one without a user interface. Two extended versions of the Room Viewer model have been developed by MacDonald [Mac96] and Carter [Carter99] to take account of more realistic observation and interaction. The screenshots of these two models are shown in Figure B-1 and Figure B-2 in this appendix, respectively.

{Ttable00} The Timetable model: The Timetable model was originally developed by Beynon. The original version is very simple and contains generic ideas that have been used in other versions. So far there have been several versions of this model. The one referred to in Chapter 4 is based on the current extended version developed by Chris Keen [Keen00] as his final year Computer Science project in 2000.

{VMC01} The VMC model: The VMC model was developed by the author. Its design is based on a vending machine process described in Hoare's book [Hoare85]. It is referred to in Chapter 5.

The Clayton Tunnel Disaster **August 25th 1861**

Three heavy trains leave Brighton for London on a fine Sunday morning. They are all scheduled to pass through the Clayton Tunnel—the first railway tunnel to be protected by a telegraph protocol designed to prevent two trains being in the tunnel at once. Elsewhere, safe operation is to be guaranteed by a time interval system, whereby consecutive trains run at least 5 minutes apart. On this occasion, the time intervals between the three trains on their departure from Brighton are 3 and 4minutes.

There is a signal box at each end of the tunnel. The North Box is operated by **B**rown and the South by **K**illick. K has been working for 24 hours continuously. In his cabin, he has a clock, an alarm bell, a signal needle telegraph and a handwheel with which to operate a signal 350 yards down the line. He also has red (stop) and white (go) flags for use in emergency. The telegraph has a dial with three indications: NEUTRAL, OCCUPIED and CLEAR.

When K sends a train into the tunnel, he sends as OCCUPIED signal to B. Before he sends another train, he send as IS LINE CLEAR? Request to B, to which B can respond CLEAR when the next train has emerged from the North end of the tunnel. The dial at one end of the telegraph only displays OCCUPIED or CLEAR when the appropriate key is being pressed at the other—it otherwise displays NEUTRAL.

The distant signal is to be interpreted by a train driver either as *all cle*ar or as *proceed with caution*. The signal is designed to return to *proceed with caution* as a train passes it, but if this automatic mechanism fails, it rings the alarm in K's cabin.

The accident

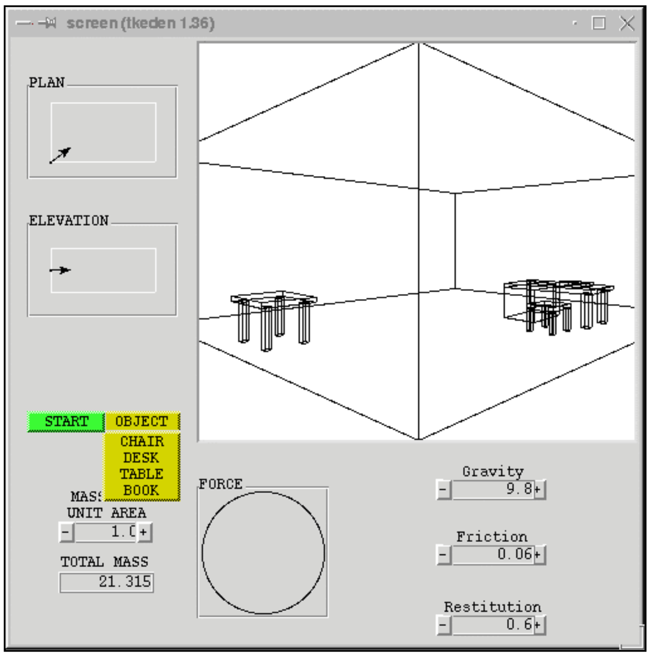
When train 1 passed K and entered the tunnel the automatic signal failed to work. The alarm rang in K's cabin. K first sent an OCCUPIED message to B, but then found that train 2 had passed the defective signal before he managed to reset it. K picked up the red flag and displayed it to Scott, the driver of train 2, just as his engine was entering the tunnel. He again sent an OCCUPIED signal to B.

K did not know whether train 1 was still in the tunnel. Nor did he know whether S had seen his red flag. He sent an IS LINE CLEAR? signal to B. At that moment, B saw train 1 emerge from the tunnel, and respond CLEAR. Train 3 was now proceeding with caution towards the tunnel, and K signalled all clear to the driver with his white flag.

But S had seen the red flag. He stopped in the tunnel and cautiously reversed his train to find out what was wrong from K.

Train 3 ran into the rear of Train 2 after travelling 250 yards into the tunnel, propelling Train 2 forwards for 50 yards. The chimney of the engine of Train 3 hi the roof of the tunnel 24 feet above. In all 23 passengers were killed and 176 were seriously injured.

Box B-1: The scenario of the Clayton Tunnel railway accident (from [Bey99])



MASS
UNIT AREA
1.0
TOTAL MASS
21.315
Gravity
9.8
Friction
0.06
Restitution
0.6

Figure B-1: A screenshot of the Room Viewer model developed by MacDonald

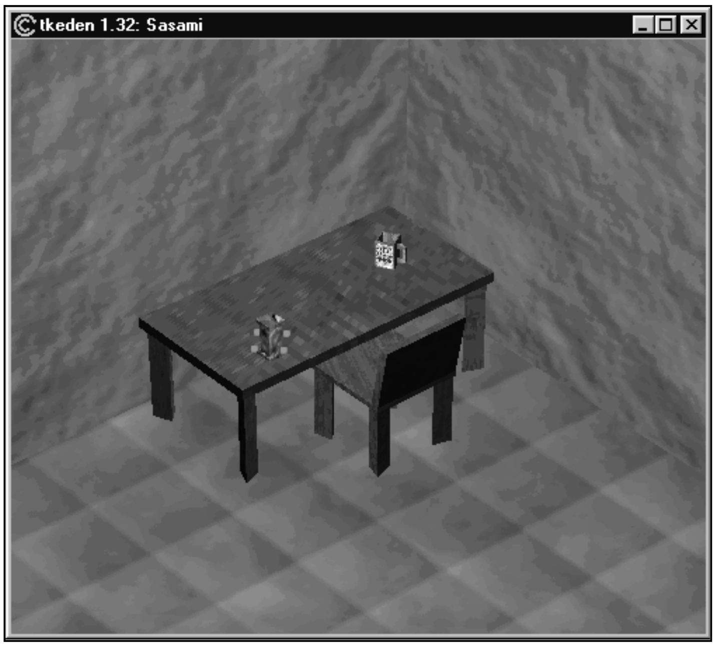


Figure B-2: A screenshot of the extended Room Viewer Model by Carter

A formal specification of the Heapsort program

proc ***Heapsort*** (a)

```
PRE: $N \geq 1$
POST: $sort(1,N) \wedge a = perm(a_0)$
Variant: n
Invariant: $(0 \leq n \leq N) \wedge sort(n+1,N) \wedge (\forall i, j | 1 \leq i \leq n < j \leq N \rightarrow (a(i) \leq a(j))) \wedge heap(1,n))$
```

```
        n := N
                "Establish heap(1,N)"
                PRE: $N \geq 1$
                POST: $heap(1,N)$
                Variant: k
                Invariant: $(1 \leq k \leq N) \wedge heap(k, N)$
                        k := (N+1)/2;
                                DO k $\neq$1 $\rightarrow$
                                        k := k-1;
                                        Shuffle(k ,N)
                                OD
        "Sort extraction"
        DO   n $\neq$ 0 $\rightarrow$
                a := swap(1,n);
                n := n-1;
                "re-establish heap(1,n)"
                    PRE:  $heap(2,n)$
                    POST: $heap(1,n)$
                Shuffle(1, n);
        OD
END
```

A formal specification of the sub-program Shuffle

proc ***Shuffle***(lo, hi)

```
PRE: $heap(lo+1,hi)$
POST: $heap(lo,hi)$
Q': $(lo \leq k \leq hi) \wedge (\forall i, j | ((lo \leq i < j < hi) \wedge (i \neq k) \wedge (i \sim> j)) \rightarrow a(i) \geq a(j))$
```

```
   DO ( (left(k)$\leq$ hi) CAND (a(k)< a(left(k)) ) COR ( (right(k) $\leq$ hi) CAND (a(k) < a(right(k)) ) $\rightarrow$
        IF (right(k)> hi) COR ( a(left(k)) $\geq$a(right(k)) ) $\rightarrow$
                m := left(k);
        [] (right(k) $\leq$ hi) CAND ( a(left(k)) $\leq$ a(right(k)) ) $\rightarrow$
                m := right(k);
        FI
                a := swap(k,m);
                k := m;
   OD
END
```

References

[Brown00] C. D. Brown. Extending the TKEden Interpreter. A final year project report 2000-2001. Department of Computer Science, University of Warwick.

[Carpen95] R. Carpenter. Statecharts in tkeden. A final year project report 1995-1996. Department of Computer Science, University of Warwick.

[Carter99] B. Carter. Sasami project report. A final year project report 1999-2000. Department of Computer Science, University of Warwick.

[Davis95] E. L. Davis. Modelling Human Interaction. A final year project report 1995-1996. Department of Computer Science, University of Warwick.

[Keen00] C. Keen. An Empirical Modelling Timetable System. A final year project report 2000-2001. Department of Computer Science, University of Warwick.

[Lui96] Y. Lui. Generic Application Modelling in Eden (GAME). A final year project report 1996-1997. Department of Computer Science, University of Warwick.

[Mac96] A. R. MacDonald. 3D Physical Modelling with Definitive Notations. A final year project report 1996-1997. Department of Computer Science, University of Warwick.

[Tru95] S. V. Truong. Interfacing Eden with Oracle. A final year project report 1995-1996. Department of Computer Science, University of Warwick.

[Wood94] J. Wood. Carhistory. Time Dependent Data using Eden & Oracle. A final year project report 1994-1995. Department of Computer Science, University of Warwick.