CLASS COMBINEDSIMULATION

User Guide

BASIC CONCEPTS

Class COMBINEDSIMULATION is a SIMULA class designed to facilitate description and simulation of systems involving both discrete and continuous phenomena.

The following gives a brief description of the class. The reader is presumed to be familiar with SIMULA, especially the class SIMULATION.

COMBINEDSIMULATION builds on SIMULA's process view of simulation. A system is conceived of as a collection of processes which undergo active and inactive phases and whose actions and interactions comprise the system's behaviour.

A distinction is made between two types of processes, namely discrete processes (class PROCESS) and continuous processes (class CONTINUOUS). Discrete processes have instantaneous active phases, called events, and cause discrete changes in the state of the system. In contrast, continuous processes undergo active phases during time intervals and cause continuous state changes.

Any process may be created, activated, de-activated or removed from the system at any time. However, COMBINEDSIMULATION permits more than co-existence of separate processes. It allows processes to communicate with one another and influence one another in a completely general way. Any process may reference and modify any variable in any other process and may affect delimiting and sequencing of active phases.

COMBINEDSIMULATION makes possible simulation of systems having piece-wise continuous state variables (class VARIABLE). The discontinuities, the events, can be time-determined (using HOLD, ACTIVATE etc.) as well as state-determined (using WAITUNTIL).

To facilitate collection of information about the model's behaviour, the class REPORTER is provided.

A sketch of COMBINEDSIMULATION containing the most essential attributes is shown below.

```
SIMULATION class COMBINEDSIMULATION;
 begin
   class CONTINUOUS;
   begin
     procedure START; ... ;
     procedure STOP; ... ;
   end;

   class VARIABLE(STATE); real STATE;
   begin
     real RATE;
     procedure START;
     procedure STOP;
   end;

   class REPORTER;
   begin
     procedure SETFREQUENCY(f); real f; ... ;
     procedure START; ... ;
     procedure STOP; ... ;
   end;

   procedure WAITUNTIL(b); name b; Boolean b; ... ;

   real DTMIN , DTMAX , MAXRELERROR , MAXABSERROR ;

 end;
```

Class COMBINEDSIMULATION is a subclass of class SIMULATION, SIMULA's system-defined class for discrete event simulation. This property can be used to describe discrete state changes. All the concepts of class SIMULATION are available to the user. Thus class PROCESS may be used in describing discrete processes.

Class CONTINUOUS can be used to describe continuous state changes defined by ordinary differential and/or difference equations. The description is given in one or more subclasses which compute the current values or derivatives of state variables. After its START-procedure is called a CONTINUOUS-object becomes active, that is to say, its user-defined actions are executed "continuously". This active phase will cease, when the object's STOP-procedure is called.

Class VARIABLE is used to represent piece-wise continuous state variables which vary between events according to first-order differential or difference equations. The value of such a variable is denoted by STATE, while RATE denotes its rate of change. After its START-procedure is called, a VARIABLE-object becomes active, that is to say, its STATE undergoes "continuous" change between the discrete events. The value of STATE is changed according to the value of RATE, as computed by the active continuous processes. The active phase will cease when the object's STOP-procedure is called.

Class REPORTER can be used to continuous sampling of state-data. After its START-procedure is called, a REPORTER-object becomes active, that is to say, its user-defined actions are executed with the frequency set by SETFREQUENCY. The active phase will cease, when the object's STOP-procedure is called.

Procedure WAITUNTIL can be used to schedule a discrete event to occur as soon as a prescribed system state is reached. WAITUNTIL(B) causes the active PROCESS-object to become passive until the Boolean expression B evaluates to true.

Between the discrete events the state of the model is updated in time-steps of varying size. DTMIN and DTMAX are used to specify the minimum and the maximum allowable step-size, respectively. MAXRELERROR and MAXABSERROR can be used to specify a maximum relative and a maximum absolute error, respectively, allowed in updating the STATE-values of the active VARIABLE-objects.

A SIMPLE EXAMPLE

The following example has to do with the trajectory of a three-stage rocket launched from the earth's surface.

During the rocket's flight its mass, velocity and altitude change according to well known physical laws as described below in class ROCKETMOTION.

```
1.  CONTINUOUS class ROCKETMOTION;
2.  begin
3.    MASS.RATE := -MASSFLOW;

4.      THRUST  := MASSFLOW*FLOWVELOCITY;
5.      DRAG    := C1*AREA*EXP(-C2*ALTITUDE.STATE)*VELOCITY.STATE**2;
6.      GRAVITY := MASS.STATE / (C3+C4*ALTITUDE.STATE)**2;
7.    VELOCITY.RATE := (THRUST - DRAG - GRAVITY) / MASS.STATE;

8.    ALTITUDE.RATE := VELOCITY.STATE;
9.  end;
```

Comments:

line

8 The change in mass is due to the expulsion of burnt fuel and takes place at constant rate: minus MASSFLOW.

4-7 From Newton's second law of motion we can determine the rocket's acceleration, that is its
VELOCITY's RATE, by summing the forces acting upon the rocket and dividing the result by the rocket's mass.

In this example three forces act upon the rocket: THRUST, DRAG and GRAVITY.

THRUST denotes the force generated by the expulsion of burnt fuel.

DRAG is the air resistance. It is dependent on the rocket's cross-sectional area.

GRAVITY is the earth's gravitational attraction acting upon the rocket. Its effect depends on the rocket's mass and altitude.

8 The rocket's change in altitude per unit of time, ALTITUDE's RATE, is equal to its

velocity.

Class ROCKETMOTION is used to define a single continuous process using three differential equations. In this example and in general, it is the user's responsibility to make sure that the equations involved are "evaluated" in the correct order. This means that the variables occurring on the right-hand side of an assignment statement must have values which reflect the current state of the system. In this example the requirements are met because THRUST, DRAG and GRAVITY are computed prior to VELOCITY's RATE.

A program for simulation of the three-stage rocket's flight will now be outlined.

The discrete state changes, that is the separation of stages, are described in the main program below. The continuous state changes between these events are defined by the class ROCKETMOTION.

```
 1.  COMBINEDSIMULATION
 2.  begin
 3.    CONTINUOUS class ROCKETMOTION; ... ;

 4.    ref(VARIABLE) MASS , VELOCITY, ALTITUDE ;
 5.    real THRUST , DRAG , GRAVITY,
 6.         MASSFLOW , FLOWVELOCITY , AREA , C1,C2,C3,C4 ;
 7.    C1:= ... ; C2:= ... ; C3:= ... ; C4:= ... ;

 8.    DTMIN:= ... ; DTMAX:= ... ;  MAXRELERROR:= ... ;
 9.    MASS     :- new VARIABLE(...);
10.    VELOCITY :- new VARIABLE( 0 );
11.    ALTITUDE :- new VARIABLE( 0 );

12.        comment: *** FIRST STAGE *** ;
13.    MASS.START;  VELOCITY.START;  ALTITUDE.START;
14.    new ROCKETMOTION.START;
15.    MASSFLOW:= ... ; FLOWVELOCITY:= ... ; AREA:= ... ;
16.    HOLD(...);

17.        comment: *** SECOND STAGE *** ;
18.    MASS.STATE:= ... ;
19.    MASSFLOW:= ... ; FLOWVELOCITY:= ... ; AREA:= ... ;
20.    HOLD(...);

21.        comment: *** THIRD STAGE *** ;
22.    MASS.STATE:= ... ;
23.    MASSFLOW:= ... ; FLOWVELOCITY:= ... ; AREA:= ... ;
24.    HOLD(...);
25.  end;
```

Comments:

line

8 DTMIN and DTMAX are set equal to the lower and upper bound of the allowable step-size, respectively. MAXRELERROR is set equal to the maximum relative error permitted.

9-11 The three VARIABLE-objects MASS, VELOCITY and ALTITUDE are generated with given initial -values.

13-14 The rocket begins to "fly" when these VARIABLE-objects are STARTed together with an object of class ROCKETMOTION.

17-24 Discrete changes in MASS, MASSFLOW and FLOWVELOCITY take place each time a stage separates from the rest of the rocket.

In this example one type of continuous-discrete interaction is demonstrated; that is, discrete changes of "continuous" variables. Two other types of interaction may also be modelled.

The triggering of a discrete event as a consequence of the fulfilment of a prescribed state condition is one type. This type of interaction can be expressed using the procedure WAITUNTIL. In the example it is known at which times stage separation occurs, so procedure HOLD has been used to schedule these events. If this were not the case, for example if instead stage separation were dependent on altitude, then procedure WAITUNTIL should be used.

The other type of continuous-discrete interaction allows for dynamic STARTing and STOPping of continuous processes. This feature makes possible dynamic replacement of differential equations.

EXECUTION OF A SIMULATION

A simulation is controlled by an object called "the monitor". It is the monitor's responsibility that:

(1) discrete events take place at the right time,
(2) the model state varies "continuously" between the events, and
(3) information about model behaviour is gathered (class REPORTER).

The monitor ensures that all continuous parts of the model operate in true parallel and are fully synchronised with the quasi-parallel discrete processes. An attempt to interfere with synchronisation, either directly or indirectly, is discovered and reported to the user.

The working cycle of the monitor is outlined below:

```
while 'more projected events' do
begin
  DT:=0;
  'execute all ACTIVE CONTINUOUS-objects';
  'execute all ACTIVE REPORTER-objects';

  while 'no event now' do
  begin
    'take an integration step, DT, fulfilling accuracy requirements';

    if 'a state-event was passed'
    then 'determine event time and reduce step accordingly';

    'perform reporting, if requested';
  end;

  'let an event take place now';
end;
```

The default integration algorithm is a fourth-order variable step-size Runge-Kutta-England algorithm (ref. 6). Five fixed step-size algorithms are optionally available: Euler, Trapezoid, Adams, Simpson and Improved Heun. Round-off errors are reduced by quasi double-precision summation (ref. 12).

The monitor causes the events to take place at the right time. The event times of state-events are determined with an accuracy of `DTMIN` using binary search and fifth-order Hermite interpolation (ref. 5).

The monitor controls objects of the class `REPORTER`. Each `REPORTER`-object has its user-defined actions executed with a specified frequency. Hermite interpolation is also used here which gives an efficient and accurate determination of the model's state at the sample times.

USER - ATTRIBUTES

The class outline presented below contains all of the attributes available to the users of class `COMBINEDSIMULATION`.

Their function is for the most part self-explanatory. Each attribute is accompanied by a short comment.

```
1. SIMULATION class COMBINEDSIMULATION;
2. virtual: procedure SIMULATIONERROR,INTEGRATIONERROR;
3. begin
4.   procedure SIMULATIONERROR; ... ;
5.   procedure INTEGRATIONERROR; ... ;

6.   LINK class CONTINUOUS; virtual: procedure PRELUDE;
7.   begin
8.     procedure PRELUDE;;
9.     procedure START; ... ;
10.    procedure STOP; ... ;
11.    Boolean procedure ACTIVE; ... ;
12.    procedure SETPRIORITY(r); real r; ... ;
13.    real procedure PRIORITY; ... ;
14.    PRELUDE; DETACH; EXECUTE:; inner; RESUME(...); goto EXECUTE;
15.  end;

16.  LINK class VARIABLE(STATE); real STATE;
17.  begin
18.    real RATE;
19.    procedure START; ... ;
20.    procedure STOP; ... ;
21.    Boolean procedure ACTIVE; ... ;
22.    real procedure LASTSTATE; ... ;
23.    real RELERROR,ABSERROR;
24.    RELERROR:=MAXRELERROR; ABSERROR:=MAXABSERROR;
25.  end;

26.  LINK class REPORTER; virtual: procedure PRELUDE;
27.  begin
28.    procedure PRELUDE;;
29.    procedure START; ... ;
30.    procedure STOP; ... ;
31.    Boolean procedure ACTIVE; ... ;
32.    procedure SETFREQUENCY(f); real f; ... ;
33.    real procedure FREQUENCY; ... ;
34.    real procedure REPORTTIME; ... ;
35.    PRELUDE; DETACH; EXECUTE:; inner; RESUME(...); goto EXECUTE;
36.  end;

37.  procedure WAITUNTIL(b); name b; Boolean b; ... ;
38.  real WAITPRIORITY; Boolean WAITPRIOR;
39.  procedure CANCELSTATEEVENT(p); ref(PROCESS) p; ... ;
40.  real DTMIN,DTMAX,MAXRELERROR,MAXABSERROR;
41.  real procedure TIME; ... ;
42.  real procedure LASTTIME; ... ;
43.  real procedure DT; ... ;
44.  Boolean EULER,ADAMS,TRAPEZ,SIMPSON;
45.  procedure PAUSE; ... ;
46.  ref(PROCESS) procedure NEXTTIMEEVENT(p); ref(PROCESS) p; ... ;
47.  Boolean REPEATSTEP;
48.  ref(VARIABLE) procedure ERRORVARIABLE; ... ;
49.  ref(CONTINUOUS) procedure ERRORCONTINUOUS; ... ;
50.  ref(REPORTER) procedure ERRORREPORTER; ... ;
51. end;
```

Comments:

line

16 COMBINEDSIMULATION is a subclass of class SIMULATION. The discrete events in the simulated system are described in the usual way; that is, through the class PROCESS.

4 SIMULATIONERROR is a virtual procedure which is called if it is necessary to stop the simulation due to an error. (Information about such errors is given in the next section).

5 INTEGRATIONERROR is a virtual procedure which is called if the requested integration accuracy can not be met.

6-15 Class CONTINUOUS is used to describe the continuous state changes of the system. The model's difference and/or differential equations are expressed in one or more of its sub-classes.

Examples:

The differential equation:

dy/dt = y*t

is expressed by

Y.RATE:=Y.STATE*TIME

where Y is a VARIABLE-object.

The difference equation:

y(i) = y(i-1) + dt*(y(i-1)*t(i-1))

is expressed by

```
Y.STATE:=Y.LASTSTATE + DT*(Y.LASTSTATE*LASTTIME)
```

The LINK-property is at the user's disposal.

8 PRELUDE is a virtual procedure which is called when a CONTINUOUS-object is generated. The procedure is predefined with an empty body.

9-10 START and STOP cause the CONTINUOUS-object to become active and inactive, respectively. That the object is active means that its user-defined actions are executed "continuously".

11 ACTIVE returns true, if the CONTINUOUS-object is active; otherwise, false.

12 SETPRIORITY(R) assigns the CONTINUOUS-object the priority R. The active CONTINUOUS-objects are executed according to decreasing priorities (high-value-first). The procedure can be used to "sort equations".

13 PRIORITY returns the current priority of the CONTINUOUS-object.

16-25 Class VARIABLE is used to represent piece-wise continuous state variables. STATE denotes the current value of such a variable. The LINK-property is at the user's disposal.

18 RATE denotes the current value of the state variable's derivative with respect to TIME.

19-20 START and STOP cause the VARIABLE-object to become active and inactive, respectively. That the object is active means that its STATE changes according to its RATE, as computed by active CONTINUOUS-objects.

21 ACTIVE returns true if the VARIABLE-object is active; otherwise, false.

22 LASTSTATE returns the value of STATE at the start of the current time step (LASTTIME). The procedure can be used to express difference equations.

23-24 RELERROR and ABSERROR denote the relative and absolute error allowed in the STATE increment at each time step. When the Runge-Kutta-England integration method is used, the step-size is automatically adjusted so that the integration error is less than

```
ABS(ABSERROR)+ABS(RELERROR*STATE).
```

At object-generation RELERROR and ABSERROR are automatically given the current values of MAXRELERROR and MAXABSERROR, respectively.

26-36 Class REPORTER can be used to gather information about the model's behaviour. Each REPORTER-object can execute its user-defined actions with a specified frequency. The LINK-property is at the user's disposal.

28 PRELUDE is a virtual procedure which is called when a REPORTER-object is generated. The procedure is predefined with an empty body.

29-30 START and STOP cause the REPORTER-object to become active and inactive, respectively. That the object is active means that its user-defined actions are executed with the specified frequency.

ACTIVE returns true if the REPORTER-object is active; otherwise, false.

32 SETFREQUENCY(F) assigns the REPORTER-object a frequency, F. The meaning of F is the following:

F>0: Execution takes place at uniformly spaced intervals of length F time units and also at event times.

F=0: Execution takes place at the end of each time step (which includes event times).

F<0: Execution takes place only at event times.

When an event takes place, the user-defined actions of all active REPORTER-objects, regardless of frequency, are executed - both immediately before and after the event.

33 FREQUENCY returns the current frequency of the REPORTER-object.

34 REPORTTIME returns the time of the next regular execution of an active REPORTER-object with a positive frequency (cf. the PROCESS-attribute EVTIME).

37 WAITUNTIL(B) causes the active PROCESS-object (CURRENT) to become passive (IDLE) over a period which is planned to last until the logical expression B evaluates to true. However, this passive period may be ended sooner by an activation of the waiting PROCESS-object.

An event scheduled by procedure WAITUNTIL is called a state-event. On the other hand, an event which is scheduled to occur at a pre-determined fixed point in time is called a time-event.

In the case of "simultaneous" time-events and state-events, time-events take priority.

38 WAITPRIORITY and WAITPRIOR can be used to rank "simultaneous" state-events in a priority order (high-value-first). When WAITUNTIL is called, the planned state-event is assigned the priority WAITPRIORITY. In the case of simultaneity, the value of WAITPRIOR at the time of the call determines if the state-event should take place before, (WAITPRIOR is true), or after, (WAITPRIOR is false), the previously planned state-events having the same priority.

39 CANCELSTATEEVENT(P) causes the annulment of a state-event which might have been planned by the PROCESS-object P.

40 DTMIN and DTMAX can be used to specify the minimum and maximum allowable step-size, respectively. The event times of state-events are determined with an accuracy of DTMIN. MAXRELERROR and MAXABSERROR specify default values for the VARIABLE-attributes RELERROR and ABSERROR, respectively (line 24). Note that the initial value of DTMIN, DTMAX, MAXRELERROR and MAXABSERROR is zero.

41-43 TIME, LASTTIME and DT return the current model time, the starting point of the current time step, and the current step-size, respectively. That is to say, TIME = LASTTIME + DT.

44 The Boolean variables EULER, ADAMS, TRAPEZ and SIMPSON can be used to select the desired integration method according to the following table. Note that Runge-Kutta-England is chosen when these four variables are false, which they are initially.

EULER	ADAMS	TRAPEZ	SIMPSON	Method	Order	Step-size
FALSE	FALSE	FALSE	FALSE	RKE	4	Variable
TRUE	FALSE	FALSE	FALSE	EULER	1	Fixed
-	TRUE	FALSE	FALSE	ADAMS	2	Fixed
-	FALSE	TRUE	FALSE	TRAPEZ	2	Fixed
-	TRUE	TRUE	FALSE	HEUN	2	Fixed
-	FALSE	-	TRUE	SIMPSON	3	Fixed
-	TRUE	-	TRUE	No name	3	Fixed

RKE : Runge-Kutta-England (ref. 6).
EULER : First-order Euler (ref. 11).
ADAMS : Second-order Adams (ref. 11).
TRAPEZ : The trapezoid method, improved Euler (ref. 11).
HEUN : The improved Heun method (ref. 11).
SIMPSON : Simpson's quadrature rule (ref. 11).

45 PAUSE can be called by a PROCESS-object and causes the user-defined actions of all active CONTINUOUS- and REPORTER-objects to be executed instantaneously (DT=0). Afterwards the PROCESS-object which called PAUSE resumes its actions.

46 NEXTTIMEEVENT should be used instead of the PROCESS-procedure NEXTEV for security reasons.

47 REPEATSTEP can be used to specify repetition of the last integration step after an eventual call to the user-provided procedure INTEGRATIONERROR. The initial value is false.

48-50 ERRORVARIABLE, ERRORCONTINUOUS and ERRORREPORTER, in the case of an error, can be used to determine which VARIABLE-, CONTINUOUS- or REPORTER-object, respectively, gave rise to the error in question.

ERROR MESSAGES

If during simulation an error occurs, an error message is output after which the virtual procedure SIMULATIONERROR is called and the simulation is stopped.

Error messages have the form:

```
***COMBINEDSIMULATION
***ERROR <message number> <message>
***ENCOUNTERED AT TIME <time>
```

The messages and their associated numbers are listed below with brief comments.

1 THE REQUESTED INTEGRATION ACCURACY CAN NOT BE ACHIEVED

When using the Runge-Kutta-England integration method, the requested accuracy (RELERROR, ABSERROR) can not be achieved without violating the step-size bound set by DTMIN. It should be mentioned that by redefining the virtual procedure INTEGRATIONERROR handling of this type of errors is put under user control.

2 THE CURRENT TIME STEP IS TOO SMALL TO ADVANCE TIME

The time increment, DT, has become so small that model time is no longer changing. This has happened even though that time is advanced using quasi double-precision summation.

3 THERE ARE NO DISCRETE EVENTS SCHEDULED

There are no more planned events; neither time-events nor state-events.

4 DTMIN<0

5 DTMIN>DTMAX

6 FREQUENCY IS TOO SMALL TO ADVANCE TIME (CLASS REPORTER)

For an active REPORTER-object (ERRORREPORTER) with a positive frequency, REPORTTIME+FREQUENCY evaluates to REPORTTIME.

7 TIME IS AT ITS MAXIMUM VALUE AND NO EVENTS OCCUR

TIME has become equal to the computer's largest real-number (MAXREAL) and no events occur. There are still planned state-events, but their conditions are not fulfilled.

8 ILLEGAL CALL OF PAUSE
9 ILLEGAL CALL OF CANCELSTATEEVENT
10 ILLEGAL CALL OF WAITUNTIL
11 ILLEGAL CALL OF SETPRIORITY (CLASS CONTINUOUS)
12 ILLEGAL CALL OF START (CLASS CONTINUOUS)
13 ILLEGAL CALL OF STOP (CLASS CONTINUOUS)
14 ILLEGAL CALL OF SETFREQUENCY (CLASS REPORTER)
15 ILLEGAL CALL OF START (CLASS REPORTER)
16 ILLEGAL CALL OF STOP (CLASS REPORTER)
17 ILLEGAL CALL OF (RE)ACTIVATE
18 ILLEGAL CALL OF PASSIVATE (OR CANCEL(CURRENT))
19 ILLEGAL CALL OF HOLD (OR REACTIVATE CURRENT)
20 ILLEGAL CALL OF CANCEL

The error messages 8-20 signify that the discrete state change in question is not caused by a discrete process.

REFERENCES

1 Helsgaun, K.:
"COMBINEDSIMULATION - a SIMULA-class for combined continuous and discrete simulation".
Proceedings of the sixth SIMULA users' Conference,
Lisboa, 1978.

2 Helsgaun, K.:
"COMBINEDSIMULATION. Speech given at the sixth SIMULA users' Conference, Lisboa 1978".
Roskilde University Center, Roskilde, september 1978.

3 Helsgaun, K.:
"COMBINEDSIMULATION. User's manual". (in Danish)
Roskilde University Center, Roskilde, november 1978.

4 Helsgaun, K.:
"COMBINEDSIMULATION. Documentation". (in Danish)
Roskilde University Center, Roskilde, january 1979.

5 Helsgaun, K.:
"On interpolation in class COMBINEDSIMULATION". (in Danish)
Roskilde University Center, Roskilde, october 1978.

6 England, R.:
"Error estimates for Runge-Kutta type solutions to systems of ordinary differential equations".
Computer Journal, Vol. 12, 1969, pp. 116-170.

7 Forrester, J.W.:
"World Dynamics".
Cambridge, Mass.: Wright-Allen Press, Inc., 1971.

8 Pritsker, A. A. B., and R. E. Young:
"Simulation with GASP_PL/1".
John Wiley & Sons, New York 1975.

9 Sim, R.:
"CADSIM. User's Guide and Reference Manual".
Imperial College. Publ. no. 75/23. London 1975.

10 Hurst, N.R., and A.A.B. Pritsker:
"Simulation of a Chemical Reaction Process Using GASP IV".
SIMULATION, Vol. 21, September 1973, pp. 71-75.

11 Young, D.M., and R.T Gregory:
"A survey of numerical mathematics".
Addison-Wesley, Massachusetts 1972.

12 Møller, O.:
"Quasi double-precision summation in floating point addition".
BIT 5, 1965, pp. 37-50 and 251-255.