\begin{enumerate}

\item Variables

\begin{enumerate}

\item E: an event – failure / repair

\item C: component associated with E

\item M: minimal cutset of which C is a member

\item M\_f: Minimal cutset due to which system failed.

\item N(M): Number of components failed in M

\item T\_m : Mission time

\item T\_e:

* + 1. Failure time if E is failure
    2. Repair time completion if E is repair

\item T\_f: Time of last system failure

\item S: minimal cutset due to which system failed

\item DT: total downtime of the system until missiontime

\item RDT: remaining downime from the current time

\item B\_f: System failed - True or False

\item B\_o : Operating at the end of mission- 1(True) or 0(False)

\end {enumerate}

\item Initialize

\begin{enumerate}

\item S = M = M\_f=None

\item T\_e=DT = RDT = T\_f = N(M) = 0

\item B\_o= 1

\item B\_f=False

\item Definitions

\begin{enumerate}

\item System repair completion

* + 1. E is completion of a component repair

\item System failure

* + 1. B

\end {enumerate}

\item For each event E on the timeline

\begin{enumerate}

\item Set C,T\_e and M based on E

\item If E is a system repair completion

\begin{enumerate}

\item DT = DT + (T\_e-T\_f)

\item B\_f= False

\item RDT=0

\item B\_o= 1

\end{enumerate}

\item If E is a system failure

\begin{enumerate}

\item B\_f= true

\item M\_f=M

\item T\_f= T\_e

\item RDT=RDT+(T\_m – T\_e)

\item B\_o=0

\end{enumerate}

\end{enumerate}

for (int i = 0; i < componentEventList.size(); i++)

{

anEvent = &componentEventList.at(i);

cutsetMemberList = \_sys->GetCutsetAffiliation(anEvent->GetComponent());

for (int j = 0; j < cutsetMemberList.size() ; j++) //For each cutset

{

int cutsetID = cutsetMemberList[j];

int cutsetlen = \_sys->CutsetLength(cutsetID);

increment = anEvent->EventType(); // +1: Failure; -1: Repair; 0: Null Event

//Just for debugging, another component fails when system down

//if (systemFailed && anEvent->EventType() == ComponentEvent::COMPONENT\_DOWN)

//{

// //DisplayEvents(componentEventList);

//}

//if there was a system failure due to this cutset and repaired

//which means that atleast one component of the cutset has been repaired

if (cutsetID == sysFailureCutsetID && eventCount[cutsetID]==cutsetlen && increment == ComponentEvent::COMPONENT\_UP)

{

systemFailed = false; //system repaired, so restore its status

//time since last system failure is MTTR

trialResult.downtime += anEvent->EventEndTime() - TimeOfLastSysFailure;

remainingDT = 0.0;

TimeOfLastSysRepair = anEvent->EventEndTime();

remainingUT = missiontime - anEvent->EventEndTime();

NSystemRepairsEncountered++;

trialResult.operatingAtEndOfMission = 1;

}

eventCount[cutsetID] += increment;

//If all components of this cutset have failed, the system has failed

if (systemFailed==false && eventCount[cutsetID] == cutsetlen) //System failure encountered

{

//DisplayEvents(componentEventList);

//nthFailure++; //Keep a count of system failures

systemFailed = true; //system failure irrespective of the cutset

sysFailureCutsetID = cutsetID;

trialResult.systemFailed = true;

TimeOfLastSysFailure = anEvent->EventEndTime();

trialResult.uptime += anEvent->EventEndTime() - TimeOfLastSysRepair;

remainingDT = missiontime - anEvent->EventEndTime();

trialResult.operatingAtEndOfMission = 0;

if (nthFailure == ComponentEvent::FIRST\_FAILURE) //First occurrence of the failure

{

ttff = anEvent->EventEndTime();

trialResult.uptime = ttff;

if(needOnlyFirstFailure)

{

trialResult.downtime = missiontime-ttff;

trialResult.systemFailed = true;

return trialResult;

}

}

}

}

}

trialResult.downtime += remainingDT;

trialResult.uptime += remainingUT;