Asserted predicates,

1. environment\_instance(Environment\_name)
2. component\_instance(Component\_name)
3. event\_schema(Event\_id,Event\_schema\_list)
4. event\_advertisement\_entry(Component\_name,Environment\_name,Event\_id)
5. event\_subscription\_entry(Component\_name,Environment\_name,Event\_id,Content\_filter\_expression,Context\_filter\_expression)
6. component\_context\_variable(Component\_name,[Key,Value])
7. environment\_component(Environment\_name,Component\_name)
8. environment\_context\_variable(Environment\_name,Component\_name,H)

The above list is the list of predicates that are used to represent the different entities of the DEBS in the database.

The below predicate is used to simulate an event origination,

**simulate\_event\_environment(Environment\_name,Event\_id,Event\_data\_list):-**

environment\_instance(Environment\_name),

% TYPE FILTER START event\_advertisement\_entry(Environment\_name,Environment\_name,Event\_id,Context\_filter\_expression),

% TYPE FILTER END

We now validate the event with one on the known event schemas, the validation includes the matching of the structure of the nested list but also the type checking of all the data elements present in the event.

[Event\_type|Et] = Event\_data\_list,

event\_schema(Event\_id,Event\_schema),

!,unify\_event(Event\_data\_list,Event\_schema),

We now enter a recursive method that traverses the tree of entities,

get\_reachable\_environments(Event\_id,Event\_data\_list,[Environment\_name],Environment\_name,[],Reachable\_environment\_list),

nl,write('Event '),write(Event\_id),write(' Visited List -->'),write(Reachable\_environment\_list),nl.

This is one of the terminal definitions of the recursive predicate, it is used when an environment is encountered without any components,

get\_reachable\_environments(Event\_type,Event\_data\_list,[],Calling\_component\_name,I,I).

This definition is used to iterate the list of components in an environment,

get\_reachable\_environments(Event\_type,Event\_data\_list,[Rh|Rt],Calling\_component\_name,I,O):-

get\_reachable\_environments(Event\_type,Event\_data\_list,Rh,Calling\_component\_name,I,O2),

get\_reachable\_environments(Event\_type,Event\_data\_list,Rt,Calling\_component\_name,O2,O).

get\_reachable\_environments(Event\_type,Event\_data\_list,Component\_name,Calling\_component\_name,I,O):-

environment\_instance(Component\_name),

not(member(Component\_name,I)),

write('Visiting Environment - '),write(Component\_name),nl,

% write('\_-> Already Visited - '),write(I),nl,

append(I,[Component\_name],I2), process\_event\_environment(Component\_name,Event\_type,Event\_data\_list,Calling\_component\_name),

findall(X,environment\_component(X,Component\_name),Environment\_name\_list),

% write('|-> Parent Environments - '),write(Environment\_name\_list),nl,

get\_reachable\_environments(Event\_type,Event\_data\_list,Environment\_name\_list,Component\_name,I2,I3),

% write('@@'),write(I3),nl,

findall(X,environment\_component\_event\_subscription(Component\_name,X,Event\_type),Sub\_component\_name\_list),

% write('|-> Sub of '),write(Component\_name),write(' - '),write(Sub\_component\_name\_list),nl,

get\_reachable\_environments(Event\_type,Event\_data\_list,Sub\_component\_name\_list,Component\_name,I3,O).

get\_reachable\_environments(Event\_type,Event\_data\_list,Component\_name,Calling\_component\_name,I,O):-

not(environment\_instance(Component\_name)),

component\_instance(Component\_name),

not(member(Component\_name,I)),

append(I,[Component\_name],O),

% TYPE FILTER START

event\_subscription\_entry(Component\_name,Environment\_name,Event\_type,Content\_filter\_expression,Context\_filter\_expression),

% TYPE FILTER END

write('##Matching## Component Found - '),write(Component\_name),nl,

% CONTEX VARIABLE START

findall(X,get\_context\_variables(Calling\_component\_name,\_,X),Environment\_context\_variable\_list),

write('Context filter expression - '),write(Context\_filter\_expression),nl,

write('Context filter data'),write(Environment\_context\_variable\_list),nl,

compute(Context\_filter\_expression,Environment\_context\_variable\_list),

write('Context filter passed'),nl,

% CONTEX VARIABLE END

% CONTENT FILTER START

can\_pass\_event(Event\_data\_list,Content\_filter\_expression),

write('\*\*Passed\*\* Component Content Filter - '),write(Component\_name),nl,

% CONTENT FILTER END

write('Visiting Component - '),write(Component\_name),nl,

trigger\_component\_event\_response(Component\_name,Event\_type,Event\_data\_list).

get\_reachable\_environments(Event\_type,Event\_data\_list,Component\_name,Calling\_component\_name,I,O):-

not(environment\_instance(Component\_name)),

component\_instance(Component\_name),

not(member(Component\_name,I)),

append(I,[Component\_name],O),

write('Visiting Component - '),write(Component\_name),nl,

not(event\_subscription\_entry(Component\_name,Environment\_name,Event\_type,Content\_filter\_expression,Context\_filter\_expression)).

get\_reachable\_environments(Event\_type,Event\_data\_list,Component\_name,Calling\_component\_name,I,I).