APPENDIX C: MAPPING SCENARIO INTO PETRI-NET

In this appendix, we present the steps of the algorithm for mapping a structured scenario into a Petri-Net.

```
Algorithm: Transform a Scenario into Petri-Net
Input: Structured Scenario S = {title, context, goal, resource, actor, episodes, alternatives}
Output: Petri-Net PN = {P, T, F, W, M0}
Begin:
1.    Clean scenario S from unnecessary information – pre-processing;
2.    //Transform scenario triggering – Initial Event:
3.    Create an input place – START for denoting S.title and Add a token;
4.    Create a dummy transition – T0 for denoting scenario triggering;
5.    FOR each S.context.pre-condition DO Create an input place and Add a token;
6.    FOR each S.context.constraint DO Create an input place and Add a token;
7.    Create an output dummy place – P0 for T0;
8.    prevNodeToLink ← P0
9.    //Transform Episodes:
10.   FOR each episode e in S.episodes DO:
11.      IF e.sentence starts with "#" THEN:
12.         Create a dummy transition for denoting a FORK;
13.         Create a dummy transition for denoting a JOIN;
14.         Link prevNodeToLink to FORK (Definition 4.1);
15.         prevNodeToLink ← FORK;
16.      IF FORK and JOIN THEN: //Create Input Dummy Place
17.         Create an input dummy place – IDP;
18.         Link FORK to input dummy place – IDP (Definition 4.1);
19.      ELSE:
20.         IDP ← prevNodeToLink;
21.      IF e is SIMPLE OR CONDITIONAL OR OPTIONAL episode THEN:
22.         Create a transition – T for denoting the location of e.sentence;
23.         Link IDP to T (Definition 4.1);
24.         FOR each e.pre-condition DO Create an input place of T;
25.         FOR each e.constraint DO Create an input place of T and Add a token;
26.         Create an output dummy place – ODP of T;
27.         FOR each e.post-condition DO Create an output place of T;
28.      IF e is CONDITIONAL OR OPTIONAL episode THEN:
29.            FOR each e.condition DO Create an input place, Add a token, Link to T;
30.            Create a dummy transition – ELSE for denoting an "ELSE";
31.            Link IDP to ELSE and ELSE to ODP;
32.      IF e is LOOP episode THEN:
33.         Create a dummy transition – BEGIN for starting a "LOOP";
34.         Link IDP to BEGIN;
35.         FOR each e.pre-condition DO Create an input place of BEGIN;
36.         Create an output dummy place – ODPL of BEGIN;
37.         Create a transition – T for denoting the location of e.sentence;
38.         FOR each e.constraint DO Create an input place of T and Add a token;
39.         Create a dummy place – XDPL; //Link to T or WHILE or NEXT
40.         Create a dummy transition – END for ending a "LOOP";
41.         Create an output dummy place – ODP of END;
42.         FOR each e.post-condition DO Create an output place of END;
43.      IF e is LOOP episode with DO-WHILE structure THEN:
44.            Create a dummy transition – WHILE for CHECK CONDITION;
45.            Link WHILE to ODPL; Link ODPL to T; Link T to XDPL;
46.            Link XDPL to WHILE; Link XDPL to END;
47.            FOR each e.condition DO Create an input place, Add a token, Link to WHILE;
48.      IF e is LOOP episode with WHILE-DO structure THEN:
49.            Create a dummy transition – WHILE for CHECK CONDITION;
50.            Link ODPL to WHILE; Link WHILE to XDPL; Link XDPL to T;
51.            Link T to ODPL; Link ODPL to END;
52.            FOR each e.condition DO Create an input place, Add a token, Link to WHILE;
53.      IF e is LOOP episode with FOR-EACH structure THEN:
54.            Create a dummy transition – NEXT for CHECK CONDITION;
55.            Link ODPL to NEXT; Link NEXT to XDPL; Link XDPL to T;
56.            Link T to ODPL; Link ODPL to END;
57.            Create an input place, Add a token for denoting an item, Link to NEXT;
58.      IF FORK and JOIN THEN:
59.         Link output dummy place – ODP to JOIN (Definition 4.1);
60.      ELSE:
61.         prevNodeToLink ← ODP;
62.      IF e.sentence ends with "#" THEN:
63.         Create a output dummy place ODP of JOIN;
64.         FORK ← NULL; JOIN ← NULL;
65.         prevNodeToLink ← ODP;
66.   //Transform scenario completion – Final Event:
67.   Create a final dummy transition - FINAL for denoting scenario completion;
68.   Link prevNodeToLink to FINAL (Definition 4.1);
69.   Link FINAL to input place – START
70.   T ← Find last transition generated form the last Episode;
71.   FOR each S.context.post-condition DO Create an output place of T;
72.   FOR each input place IP created from S.context.pre-condition DO Link FINAL to input place IP;
73.   //Transform Alternatives:
74.   FOR each alternative a in S.alternatives DO:
75.      FOR each solution-step ss in a.solution DO:
76.         Create a transition TX for denoting the location of ss.solution.sentence;
77.         IF ss is the first in a.solution THEN:
78.            IF a.cause is not empty THEN:
79.              FOR each cause in a.cause DO Create an input place of TX and Add a token;
80.            ELSE Create an input dummy place of TX;
81.            T ← Find transition – T from a.branchingEpisode;
82.            IF there exist T THEN:
83.               Link output dummy place – ODP of T to TX (Definition 4.1);
84.            ELSE
85.               Create an input dummy place – XIDP of TX;
86.         ELSE
87.            Create an input dummy place – XIDP of TX;
88.            Link previousTX to input dummy place – XIDP;
89.         IF ss is the last in a.solution THEN:
90.            IF a.post-condition is not empty THEN:
91.               FOR each a.post-condition DO Create an output place of TX;
92.            IDP ← Find Input Dummy Place of Trans T from a.goToEpisode;
93.            IF there exist IDP and T THEN:
94.               Link TX to input dummy place – IDP of T (Definition 4.1);
95.            ELSE-IF a.post-condition is empty THEN:
96.               Create an output dummy place – XODP of TX;
97.         previousTX ← TX;
98.   UNTIL there exist two common places (from Pre-Condition or Post-Condition) within PN REPEAT
         Fuse common places (Definition 4.2);
99.   Return PN;
End
```

Fig. 25. Transforming a Scenario into Petri-Net.