

Hybrid Modeling and Simulation of Failures in Robotic Assemblies Using Hierarchical Time-Extended Petri Nets

David Gentry III , S. Ramaswamy

School of Computer and Applied Sciences

Georgia Southwestern State University

Americus, GA 31709

Phone: (912)-931-2100, Fax: (912)-931-2270

email: srini@gswrs6k1.gsw.peachnet.edu

Preferred Technical Area: Petri Nets and Applications / Factory Modeling and Automation

Abstract

In this paper, Hierarchical Time-Extended Petri Nets (H-EPNs) are used for the modeling and simulation of complex automated manufacturing system functionalities including failures and complex assembly operations. H-EPNs allow for integrating the top down and bottom up modeling techniques to effectively construct models of automated manufacturing systems through the use of activator arc extensions [3]. This flexibility allows for the reuse of Petri Net models of independent subsystems to be easily integrated into any top-down system decomposition.

Fourth International Conference on Control Automation, Robotics and Vision

Westin Stamford

Singapore

Dec 4-6, 1996

Hybrid Modeling and Simulation of Failures in Robotic Assemblies Using H-EPNs

David Gentry III, S. Ramaswamy

School of Computer and Applied Sciences, Georgia Southwestern State University, Americus, GA 31709, USA

Phone: (912)-931-2100, Fax: (912)-931-2270, email: dlg@canes.gsw.peachnet.edu,
srini@gswrs6k1.gsw.peachnet.edu, www: <http://canes.gsw.peachnet.edu>

Abstract - In this paper, Hierarchical Time-Extended Petri Nets (H-EPNs) are used for the modeling and simulation of complex automated manufacturing system functionalities including failures. H-EPNs allow for integrating the top-down and bottom-up modeling techniques to effectively construct models of automated manufacturing systems through the use of activator arc extensions. This flexibility allows for the reuse of PN models of independent subsystems to be easily integrated into any top-down system decomposition.

I. INTRODUCTION

Automated manufacturing systems are usually characterized by complex interactions between a variety of components that include distributed and parallel software, hardware, and communication components. In addition they increasingly have a large number of interfaces (both internal and external) that aid in sensing the operating environment as well as other subsystems. The number of interconnections between their corresponding components tends to be large and complex. Moreover, these components evolve over time, their logical and physical interactions change, and the operational semantics of the system change accordingly, often leading to increased system complexity. The major barrier in building such complex systems tends to be the overall coordination and integration of these subsystems with their conflicting requirements specification, rather than individual subsystem components. Such subsystems also interact in a complex manner with respect to shared system resources, data, etc. Moreover, as the complexity of the system increases, it becomes more difficult to acquire enough knowledge about the system parameters due to the complexity and the dynamic nature of the system operating environment. To this end, the capturing of dynamic system behavior by a system model becomes an invaluable advantage. Typical applications areas include manufacturing, process control, aerospace, defense, transportation, communications, energy, utilities, medical, health, and commercial data processing, etc.

In terms of software development for such systems, the above means that (i) the system requirements are not always exhaustively specified at the beginning of the development process, (ii) the process of adding more complex functionality and increasing / modifying output requirements increases the possibility of introducing catastrophic errors, (iii) the means of communication between personnel from different specialistic domains is not well defined, (iv) the design technique adopted for a particular subsystem design must be independent of the overall system design, thereby providing more flexibility to the terms in charge of particular subsystem / module development, (v) it should be easy to integrate such diverse design / development approaches into the overall system design, requiring minimum effort in testing, validation and integration.

H-EPNs have been developed to efficiently capture design constraints, thereby allowing for the easy development of PN system models and utilizing already developed system models to be integrated within a top-down decomposition of an incremental / new system design. In this paper, the H-EPN system model is easily derived using H-EPN modules that may be generically defined for any automated manufacturing system. Moreover, the H-EPN model also allows for studying system failures that are known apriori, thereby building error detection and recovery processes into the H-EPN system model. Such an approach will allow for tradeoffs to be established during the system implementation phase, wherein certain simple (trivial) system errors may be automatically handled by the system software.

This paper is structured as follows: In section II, H-EPN extensions to PNs are discussed. Sections III and IV present the example assembly process and the H-EPNs model of the complex assembly operations in detail. Section V concludes the paper.

II. PETRI NET EXTENSIONS

Petri nets (PNs) are a graphical formalization for the modeling and specification of dynamic systems. Ordinary PNs consist of three basic elements: a set of places, a set of transitions, and a set of arcs. Places represent system processes and transitions represent system conditions and events. Tokens in a place denote the state markings of a PN. In ordinary PNs, a place can contain any number of tokens. Arcs interconnect places and transitions and arcs may be weighted. A transition is enabled if all the input places contain the correct number of tokens. When a transition is enabled, it can possibly fire. If it fires, tokens are removed from the input place(s) and placed in the output place(s) according to the arc weights in the PN model. Figure 1 shows a simple PN example. Its operations can be described as below: Places P1 and P2 are input places to transition T1 and P3 is the output place to transition T1. Since P1 and P2 contain input tokens, the transition T1 is enabled. When T1 fires, tokens in places P1 and P2 are removed and an output token placed on place P3.



Figure 1. Example Petri Net

While ordinary PNs are quite useful in representing and understanding complex system interconnections and operations, they are not suited to model real world applications that require a complex decision making structure. Ordinary PNs cannot resolve conflicts or make decision choices. PNs extended with inhibitor arcs provide the ability to model decision behavior, however, they are still inhibited by their inability to represent depth in a complex PN model. To this end, many PN extensions have been defined and used in various applications. A summary of such extensions to

manufacturing systems may be found in [1-6] and references therein.

H-EPNs have been defined to model and derive the coordination level of a three level intelligent machine organization [1]. The place extensions include status, action, subnet, decision, and source-sink places. A status place is an ordinary PN place. An action place represents some system action. A subnet place is an abstraction for an independent system component that minimally interacts with other system components. A decision place represents a binary / multi-choice decision. A source-sink place represents a tokens' entrance or departure from a PN. This place is useful in testing the system for boundedness properties. H-EPNs are discussed in greater detail in [1, 3]. Tokens in a H-EPN are distinguished as either control or flow tokens. A flow token describes the flow of system resources in a dynamic system operation, and are useful in representing the input conditions for subnets. A control token is used in representing actual system resource and the presence or absence of these tokens either denotes resource availability or other system information such as messages etc. Arc extensions include the definition of two zero weighted arcs: inhibitor and activator arcs. Similar to the definitions in other PN extensions, an inhibitor arc allows a transition to be enabled only if there are no tokens in the place belonging to the arc. An activator arc allows a transition to be enabled only if there is a token in the preceding place. In situations such as conflicts and message passing to multiple areas of the subsystem, the activator arc is used effectively to enforce priorities and to duplicate data over multiple channels without increasing the need for a large number of tokens and associated complexities, respectively. Figure 3 depicts the behavior of inhibitor and activator arcs.

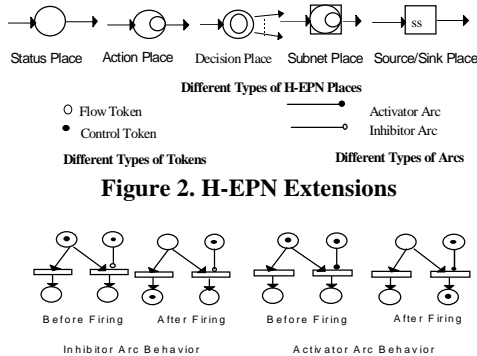


Figure 2. H-EPN Extensions

H-EPNs allow for the modeling of systems as a collection of interacting nets. From a H-EPN system modeling point of view, the operations of a large system is represented by the execution of loosely related subnets, called interacting subnets that closely relate to a bottom-up synthesis of a system model. H-EPNs allow *distribution of control information* and *encapsulation of subsystem resources* and thereby facilitate the integration of top-down design and bottom-up design philosophies. The activator arc extensions allows for the easy transformation of a set of bottom-up synthesized subnets (MIMO modules) to be converted to a SISO net structure and thereby be used in a top-down decomposition of a system model. This technique allows for the creation of an arbitrarily large net systems while preserving the essential net properties of boundedness, liveness and reversibility [3]. The H-EPN approach therefore, allows for better definition and

understanding of system structure and organization by means of visually explicit semantics.

III. EXAMPLE SYSTEM

The robot assembly workstation is shown in Figure 4a. The workstation assembles a product from four different parts: A, B, C, and D respectively. The final part S4 is formed from a combination of parts A, B, C and D. There are three possible subassemblies as illustrated in Figure 4b. Parts A, B, C, and D are moved along the input conveyer belt. The four parts may arrive in random order, but the parts always come together. When a part is in front of the camera, the camera identifies the part and the manipulator (robotic arm) is requested to fetch the part. When the manipulator fetches the part, the part is either buffered in the correct buffer or mated. Part A must always be buffered. The other parts (B, C, and D) may or may not be buffered.

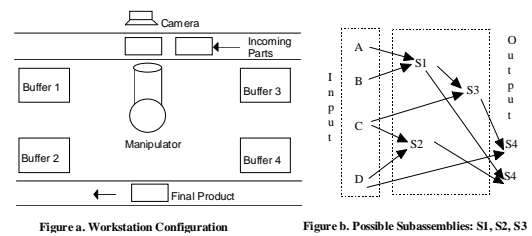


Figure 4. The Assembly

There are two basic methods of assembly. The first method requires part A and part D be buffered. If part A is already in buffer A, part B is fetched from the buffer B or the conveyer, and mated with part A to form subassembly S1 which is stored in buffer 1. S2 is produced in a similar manner using parts C and D and stored in Buffer 4. If S1 and S2 are assembled and stored, S2 is fetched and mated with S1 in buffer A. This produces part S4. Then S4 is moved to the end conveyer belt. The assembly of S1 may be done before or after the assembly of S2.

The second method of assembly consists of assembling S1 as stated above, then part C is fetched from buffer C or the conveyer and mated with S1 to form part S3. Then part D is fetched from buffer D or from the conveyer and mated with S3 to form part S4, which is later moved to the output conveyor belt.

IV. H-EPN SYSTEM MODEL

The H-EPN model is split up into subnets and the collective functioning of these H-EPN subnets defines the assembly process. Table 1 provides a description of the places and transitions in the H-EPN system model.

A. Move Subnet

Figure 5 shows the move subnet. This subnet is responsible for the manipulator moving a part from the conveyor belt to a buffer, or from a buffer to another buffer, or from a buffer to a conveyer belt. When a token enters the move subnet, the part is moved a small distance (P60). The token then advances to P61, where the position of the part is checked. If the part is at the destination, the token moves on to P62. At P62, the part is placed at the destination. Then the token goes out of the subnet at P63. If the part is not at the destination, the token moves to P64. At P64, the robotic manipulator checks for system state changes due to

dropping of a part¹. If there are no changes, the token goes back to P60. Else, the manipulator checks to see if the part was dropped. If it was not dropped, the token goes back to P60. If the part has been dropped, the token advances to P66. Here the manipulator checks to see if the part can be retrieved. If the part can not be retrieved, the token is sent to P67, that causes a system alert. If the part can be retrieved, the token is passed to P69 which locates the part. Then the token is passed to P70 and P71, which picks the part up. After the part is retrieved, the token is sent to P60.

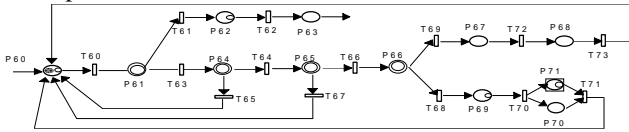


Figure 5. Move Subnet

B. Request/Release Subnet

The request/release subnet is shown in Figure 6. The request/release subnet is used to process the request and release of resources such as the camera or the manipulator. When a request for a resource is made a token enters place P20. Transition T20 is enabled only if P26 contains a resource token, thereby creating a queue at place P20 for the corresponding resource. A token in P26, indicates that the resource is free. Place P21 checks to see if the resource is working. If it is not working, a token enters P24. From P24 an error message signaled. When the error is fixed, the resource is entered back into the system through place P25.

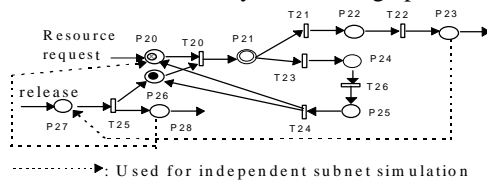


Figure 6. Request/Release Subnet

C. Identify Subnet

The identify subnet in Figure 7 uses the camera to identify the part. It is assumed that there occur no input errors and that an input group contains all the four distinct parts, A, B, C and D respectively. Therefore, these parts are uniquely identified. Once the parts are identified four tokens corresponding to the four input parts are placed in place P44 that is used as an input to the assembly subnet.

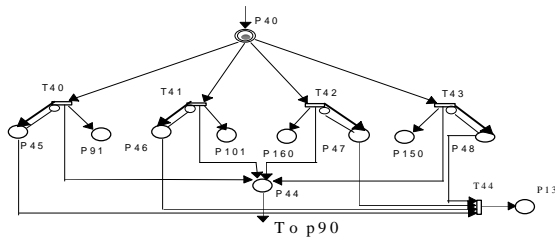


Figure 7. Identification of Part Subnet

D. Pick-Up Subnet

Figure 8 shows the pick up subnet. This subnet uses the manipulator to pick up the part. A token enters the subnet at P80. The token advances to P81, causing the arm to advance toward the part. Then the token goes to P82, where it is determined if the part is within reach. If it is not reachable, the token goes back to P81.

If it is reachable, a token goes to P83. Here the arm determines if the part is retrievable. If the part can not be retrieved, an error message is realized in P84. If the part can be picked up, it is retrieved in P85. Then the token is passed out of the subnet.

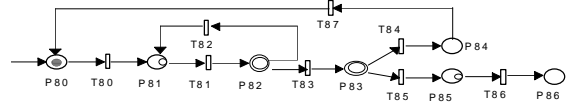


Figure 8. Pick-Up Subnet

E. Assembly Subnet

The assembly process is illustrated by Figures 9-13. These figures represent one large subnet, but for simplicity they have been broken down in to distinct figures. After entering the subnet, a token enters place P90, which correspond to parts A, B, C, or D entering the assembly line.

In the assembly of S1 (Figure 9), part A or B is identified. If part A is identified, it is fetched, and moved to the buffer 1. If part B is identified, it is fetched and moved to either buffer 1 or buffer 2. If part A is not in buffer 1, part B is moved to the buffer 2 and the manipulator is released. Otherwise, part B is moved to buffer 1 (rather than buffer 2) and mated with part A to form the subassembly S1. After the mating the manipulator is released. If part B is buffered in buffer 2, the part waits for part A to arrive in buffer 1. When part A is buffered, the manipulator is requested. Then part B is fetched and moved to buffer 1. Here it is mated with part A and the manipulator is released. The assembly of part S2 is done in the same manner. Part D would mimic part A and part C would mimic part B. The assembly of S2 is shown in Figure 10.

Figure 11 shows the formation of part S3. Part C is identified, fetched and moved. If part S1 has been assembled and buffered in buffer 1, part C is moved to buffer 1 and mated with part S1. Then the manipulator is released. If S1 is not in buffer 1, part C is buffered in buffer 3 and the manipulator is released. When S1 is assembled, the manipulator is requested. Then part C is fetched and moved to buffer 1. Here it is mated with S1 and the manipulator is released. Part S3 is formed and stored in buffer 1.

The mating of parts S1 and S2 is shown in figure 12. If parts S1 and S2 are in buffers 1 and 2 respectfully, the manipulator is requested. Part S2 is fetched and moved to buffer 1. Then mated with S1 to form S4. After the mating the manipulator is released.

The matting of parts S3 and D is shown in figure 13. If part D is identified, it is fetched, and moved. If part S3 is in buffer 1, D is moved to buffer 1 and mated with S3. The manipulator is then released. If S3 is not in buffer 1, part D is buffered in buffer 4 and the manipulator is released. When buffer 1 contains S3, the manipulator is requested. Part D is fetched and moved to buffer 1. Here it is mated with S3. The manipulator is released and Part S4 is created.

F. System Model

Figure 14 shows the entire system. A part is on the conveyer belt (P1). The conveyer belt advances (P2). Now the part is in front of the camera on the conveyor belt (P3). The camera is requested (P4, P5). The part is identified (P6, P7). When the part is identified, the camera is released (P8, P9). After the release of the camera, the conveyer advances, and the identified part is assembled or buffered (P10, P11). After all the parts are assembled, S4 arrives on the end conveyor belt (P12).

¹ It is assumed that the robot uses a 4 parameter trajectory planning algorithm using the (x, y, z, w) coordinates, where w represents the part weight.

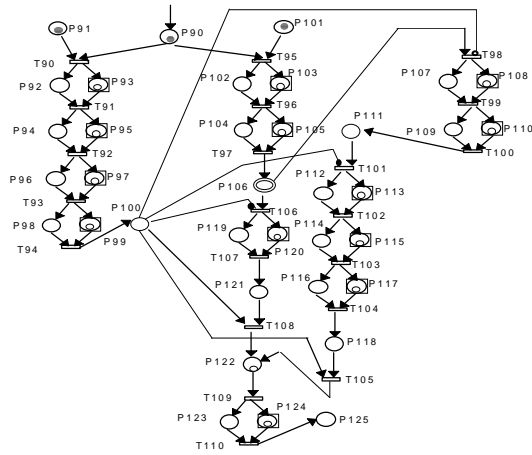


Figure 9. Subassembly S1 / Storage of Parts A, B

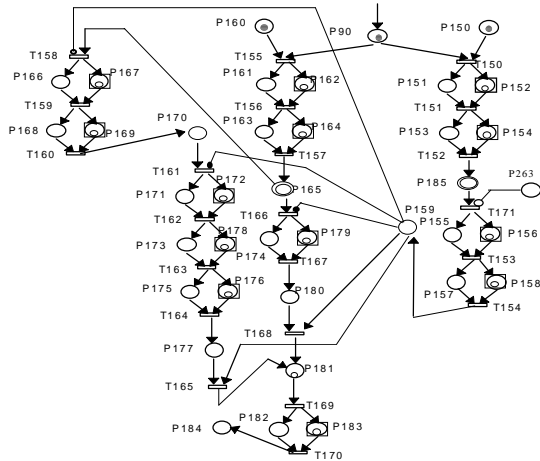


Figure 10. Subassembly S2 / Storage of Parts C, D

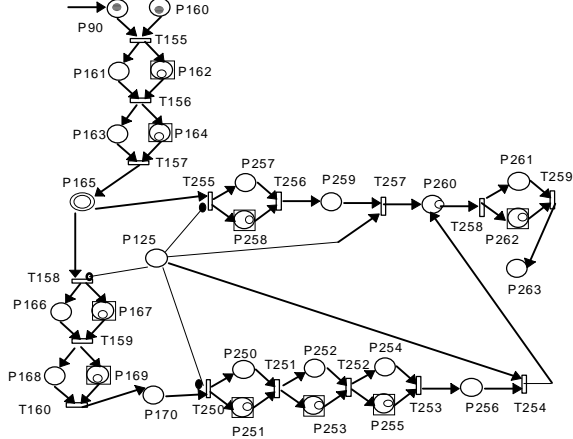


Figure 11 Subassembly S3 / Assembly of Part C, S1

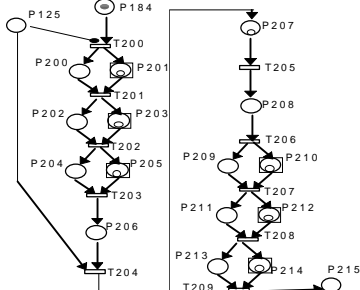


Figure 12. Subassembly S4 / Assembly of Part S1, S2

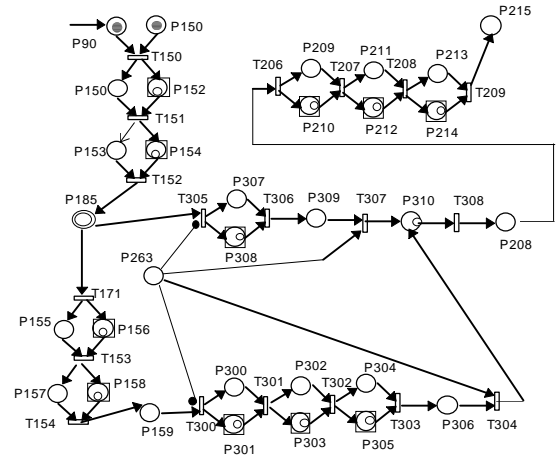


Figure 13. Alternate Subassembly S4 / Assembly of Part S3, D

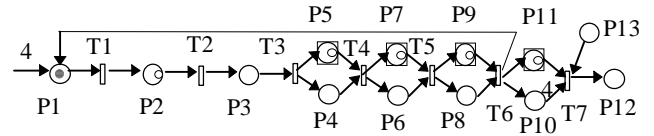


Figure 14. Overall System Model

V. CONCLUSIONS

The H-EPN system model was simulated using Visual Simnet [7]. The system was tested for 100,000 incoming parts which were used to make 25,000 outgoing components. The important system properties of boundedness, liveness and reversibility were preserved.

VI. REFERENCES

1. S. Ramaswamy, K. P. Valavanis, "Hierarchical Time-Extended Petri Nets Based Error Identification and Recovery of Multilevel Systems", *IEEE Transactions on Systems Man and Cybernetics*, Vol 26, No. 1, Feb 1996, pp. 164-175.
2. S. Ramaswamy, and K. S. Barber, "A Design Architecture for the Modeling, Analysis and Design of Manufacturing Control Software", *Submitted to the Journal of Intelligent Automation and Soft Computing*, July 1996.
3. S. Ramaswamy, K. P. Valavanis, "On the Construction and Analysis of Multiple Input Multiple Output Subnet Structures", *Third International Conference on Control, Automation, Robotics and Vision*, Singapore, Nov. 1994.
4. S. Ramaswamy and K. P. Valavanis, "Modeling, Analysis and Simulation of Failures in a Materials Handling System With Extended Petri Nets," *IEEE Transactions On Systems, Man, And Cybernetics*, Vol. 24, No. 9, September 1994, pp. 1358-1373.
5. K. P. Valavanis, S. Ramaswamy, and Steve P. Landy, "Extended Petri Net Based Modeling, Analysis and Simulation of an Intelligent Materials Handling System," *Journal of Intelligent and Robotic Systems*, No 10, 1994, pp. 79-108.
6. Adao Murata, "Petri Nets: Properties, Analysis and Applications," *Proceedings of the IEEE*, Vol 77, No. 4, April 1989. Pp. 541-580
7. G Wolf, *Visual Simnet V1.33*. Dresden, Germany. (<http://www.crim.ca/se/Petri/visualsimnet.html>)

Places	Transitions	Places	Transitions
P1: Part on conveyor	T1: Part on conveyor	P123: Release arm	T168: Ready to mate
P2: Advance conveyor	T2: Conveyor is advanced	P124: Release arm subnet	T169: C and D mated
P3: Part in front of camera	T3: Begin id part	P125: Part S1 in buffer 1	T170: Arm released
P4: Request camera	T4: Camera requested	P150: Part D identified	T171: Where to move?
P5: Request camera subnet	T5: Part identified	P151: Request arm	T200: Part s are ready
P6: Id Part	T6: Camera is released	P152: Request arm subnet	T201: Arm requested
P7: Id Part subnet	T7: Parts all assembled	P153: Pick-Up	T202: Part picked up
P8: Release camera	T20: Get resource	P154: Pick-Up subnet	T203: Part moved
P9: Release camera subnet	T21: Resource working	P155: Move	T204: Ready to mate
P10: Assembly	T22: Out of request	P156: Move subnet	T205: Parts mated
P11: Assembly subnet	T23: Error in resource	P157: Release arm	T206: Ready to exit
P12: Final assembly complete	T24: Error fixed	P158: Release arm subnet	T207: Part picked up
P13: Flush id tokens	T25: Release resource/exit	P159: Part D in buffer 4	T208: Part moved
P20: Enter request subnet	T26: For simulation	P160: Part C identified	T209: Arm released
P21: Is resource working?	T40: Part A identified	P161: Request arm	T250: Parts ready
P22: Resource working	T41: Part B identified	P162: Request arm subnet	T251: Arm requested
P23: Exit request subnet	T42: Part C identified	P163: Pick-Up	T252: Part picked up
P24: Error in resource	T43: Part D identified	P164: Pick-Up subnet	T253: Part moved
P25: Error fixed	T44: All parts identified	P165: Where to move?	T254: Parts ready to mate
P26: Control token place	T60: Arm moved	P166: Move	T255: Parts are ready
P27: Enter release subnet	T61: At destination	P167: Move subnet	T256: Part moved
P28: Exit release subnet	T62: Part released	P168: Release arm	T257: Parts ready to mate
P40: Camera identifies part	T63: Part not at destination	P169: Release arm subnet	T258: Parts mated
P44: Part identified/exit	T64: Change in weight	P170: Part C in buffer 3	T259: Arm released
P45: Part A identified	T65: No change in weight	P171: Request arm	T300: Parts ready
P46: Part B identified	T66: Part dropped	P172: Request arm subnet	T301: Arm requested
P47: Part C identified	T67: Part not dropped	P173: Pick-Up	T302: Part picked up
P46: Part D identified	T68: Part is retrievable	P174: Pick-Up subnet	T303: Part moved
P60: Move small distance	P69: Part is not retrievable	P175: Move	T304: Parts ready to mate
P61: Check position	T70: Part located	P176: Move subnet	T305: Parts are ready
P62: Release part	T71: Part retrieved	P177: Part C moved to buffer	T306: Part moved
P63: Exit move	T72: Part not retrievable	P178: Move	T307: Parts ready to mate
P64: Check weight	T73: For simulation	P179: Move subnet	T308: Parts mated
P65: Is part dropped?	T80: Prepare to move arm	P180: Part C moved to buffer	
P66: Can part be retrieved?	T81: Arm moved	P181: Mate parts C and D	
P67: Part unretrievable	T82: Part not in reach	P182: Release arm	
P68: An error is realized	T83: Part in reach	P183: Release arm subnet	
P69: Locate part	T84: Part unretrievable	P184: Part S2 in buffer 4	
P70: Pick-Up part	T85: Part is retrievable	P185: Where to move?	
P71: Pick-Up subnet	T86: Part retrieved	P200: Request arm	
P80: Enter pick-up subnet	T87: For simulation	P201: Request arm subnet	
P81: Move arm	T90: Part A enters assembly	P202: Pick-Up	
P82: Is part in reach?	T91: Arm requested	P203: Pick-Up subnet	
P83: Can part be picked up?	T92: Part picked up	P204: Move	
P84: An error is realized	T93: Part moved	P205: Move subnet	
P85: Retrieve part	T94: Arm released	P206: Part S2 ready to mate	
P86: Exit pick-up subnet	T95: Part B enters assembly	P207: Mate S1 and S2	
P90: Enter assembly	T96: Arm requested	P208: Part S4 in buffer 1	
P91: Part A identified	T97: Part picked up	P209: Pick-Up	
P92: Request arm	T98: Buffer part B	P210: Pick-Up subnet	
P93: Request arm subnet	T99: Part B moved	P211: Move	
P94: Pick-Up	T100: Arm released	P212: Move subnet	
P95: Pick-Up subnet	T101: Part A is ready	P213: Release arm	
P96: Move	T102: Arm requested	P214: Release arm subnet	
P97: Move subnet	T103: Part picked up	P215: Part on conveyor/Exit	
P98: Release arm	T104: Part moved	P250: Request arm	
P99: Release arm subnet	T105: Ready to mate	P251: Request arm subnet	
P100: Part A in buffer 1	T106: Mate part B	P252: Pick-Up	
P101: Part B identified	T107: Part B moved	P253: Pick-Up subnet	
P102: Request arm	T108: Ready to mate	P254: Move	
P103: Request arm subnet	T109: A and B mated	P255: Move subnet	
P104: Pick-Up	T110: Arm released	P256: Part C ready to mate	
P105: Pick-Up subnet	T150: Part D enters assembly	P257: Move	
P106: Where to move?	T151: Arm requested	P258: Move subnet	
P107: Move	T152: Part picked up	P259: Parts ready to mate	
P108: Move subnet	T153: Part moved	P260: Mate S1 and C	
P109: Release arm	T154: Arm released	P261: Release arm	
P110: Release arm subnet	T155: Part C enters assembly	P262: Release arm subnet	
P111: Part B in buffer 2	T156: Arm requested	P263: Part S3 in buffer 1	
P112: Request arm	T157: Part picked up	P300: Request arm	
P113: Request arm subnet	T158: Buffer part C	P301: Request arm subnet	
P114: Pick-Up	T159: Part C moved	P302: Pick-Up	
P115: Pick-Up subnet	T160: Arm released	P303: Pick-Up subnet	
P116: Move	T161: Part D is ready	P304: Move	
P117: Move subnet	T162: Arm requested	P305: Move subnet	
P118: Part B moved to buffer	T163: Part picked up	P306: Part S3 ready to mate	
P119: Move	T164: Part moved	P307: Move	
P120: Move subnet	T165: Ready to mate	P308: Move subnet	
P121: Part B moved to buffer	T166: Mate part C	P309: Parts ready to mate	
P122: Mate parts A and B	T167: Part C moved	P310: Mate S3 and D	

Table 1: Description of Places and Transitions