FAIM2007 Special Issue: Simulation-based scheduling of assembly operations

Gerald Weigert, Thomas Henlich

FAIM2007 Special Issue: Simulation-based scheduling of assembly operations

Journal:	*International Journal of Computer Integrated Manufacturing*
Manuscript ID:	TCIM-2007-IJCIM-0159.R1
Manuscript Type:	Special Issue Paper
Date Submitted by the Author:	04-Mar-2008
Complete List of Authors:	Weigert, Gerald; Technische Universität Dresden, Electronics Packaging Laboratory Henlich, Thomas; Technische Universität Dresden, Electronics Packaging Laboratory
Keywords:	DISCRETE EVENT SIMULATION, ASSEMBLY PLANNING, GRAPH THEORY, PROCESS PLANNING, MATERIALS MANAGEMENT, PETRI NETS
Keywords (user):	

Simulation-based scheduling of assembly operations

Gerald Weigert[1] <Gerald.Weigert@tu-dresden.de>
Tel.: +49 351 46336439

Thomas Henlich <Thomas.Henlich@tu-dresden.de>
Tel.: +49 351 46331696

Technische Universität Dresden
Department of Electrical Engineering and Information Technology
Electronics Packaging Lab
01062 Dresden, Germany
Fax: +49 351 46337035

Changes in this revision in red (except minor reformatting/rephrasing)

ABSTRACT

The production of large-sized machines or facilities is fundamentally different from the usual manufacturing processes. Mostly it is a fixed-site production, dominated by typical assembly processes instead of linear operation sequences in the known flow shop or job shop problems. To describe these processes, several graph methods were developed in the past, i.e. AND/OR-graphs or modified Petri nets, but most of them are static methods and they are not suitable for scheduling tasks. This paper demonstrates, how Timed AND/OR-graphs can be modeled with a common Discrete Event Simulation (DES) system. The objective is a simulation-based scheduling system for assembly processes which is able to improve the prediction of due date keeping as well as to optimize the workflow of the assembly systems.

KEYWORDS

Modeling, discrete-event simulation, assembly process, assembly graph

1. INTRODUCTION

The production of large-sized machines or facilities, i.e. rotary printing presses or big milling machines, is fundamentally different from the usual manufacturing processes. Not only the fixed-site production is the distinctive feature – this can be found in some manufacturing processes, too – but more essential is the kind of routing. Whereas linear operation sequences dominate the usual manufacturing processes, steps of merging components become more typical for assembly processes. Their most important notes are:

- workflow must be described by complex graphs and not by simple sequences
- the high importance of the bill of material
- exchangeability of technological operations inside a sequence
- a high ratio of manual operations (stochastic influence, spreading of operation time)

The aim of this project is to schedule and to control such an assembly process, so that scheduled delivery dates are kept and personal and material resources are optimally utilized. Until now the project partners predominantly

[1] Corresponding author

have organized their production processes by simple capacity analysis – with the well known disadvantages. Frequently the resources are calculated too optimistic, because the dynamics of the process are not sufficiently considered. That is why the resources are overloaded during critical periods, with the effect of unintentional shifts of due dates.

For the planning of manufacturing processes the simulation method has become one of the most indispensable tools. Simulation-based scheduling of manufacturing processes is state of the art today. But most of these simulation tools use inflexible workflow (sequences of operations) more or less. Tree structures and distinctive interdependencies between articles and products (bill of materials) – which is typical for assembly processes – is a marginal feature here. For developing a suitable simulation tool for planning assembly processes, two steps are necessary:

1. Creation of an suitable graph model for description of assembly processes (mathematical/methodical step)
2. Application of this model with the aid of an existent simulation system

In the following section an overview is given about the known assembly graph models as well as their extensions towards time consumptive activities. It is shown that the so-called AND/OR graphs are very similar to the Petri net models. So both AND/OR graphs and Petri net models are in principle suitable for modeling assembly processes. To simulate the assembly processes, a usual object-oriented Discrete Event Simulation (DES) system is used with its typical simulation units such as machines, queues, working flows etc. This simulation model should be the basis for a process-accompanying scheduling system.

2. Describing Assembly Processes with Graphs

There are diverse methods for describing assembly activities. They differ in approach, detailedness of description, and the degree of abstraction. For example, from an assemblyman's point of view other aspects are important than from that of planning or simulation of the whole process. Therefore all these forms of description have their justification – in their respective field of application. This is the reason some of them are described in this section.

Whereas in manual assembly one gets by with approximate plans (taking human intelligence into account), in the field of robotics an exact specification of the assembly process is strictly necessary. That is why in the following a number of methods originate from this field of research.

2.1 Some general methods

In this section there follows a description of some methods which, even though they are unsuitable or impractical for DES simulation, already have some elements which will be important later on. It is also possible to derive these graphs from the more detailed ones described in section 2.2.

a) Bill of Materials and the Informal Assembly Graph

In industrial practice, especially when the assembly process is performed manually, a formalized representation of precedence relationships between operations is in many cases nonexistent. This is often not such a big problem, as skilled workers are able to plan their activity from previous work experience. What is seen as more important at operational level is the parts list or bill of materials (fig. 1), because this can be used for checking if all required parts are present before an assembly operation can be started.

fig-1.tif

Fig. 1: Bill of materials and an informal assembly description

In case an assembly graph is still required for special reasons, it will in most cases be created in a retroactive way, e.g. by questioning the staff about their way of working. The graph is then very informal, but this is acceptable because it only has a supporting function (e. g. for management decisions).

b) Gozinto Graph

The Gozinto (fig. 2) graph is tightly related to economical production planning and also to the bill of materials. It gives information on the number of items of each sort that go into an intermediate or final product. From the Gozinto

graph any kind of parts list can be extracted by mathematical calculus. Another information to be derived is the manufacturing level at which a certain item is required. That is why the Gozinto graph can also be used for a rough temporal planning. However duration time and resources demand of each operational step are not included in this type of graph.

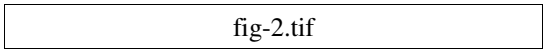


Fig. 2: Gozintograph of product ABC

c) Direct Graph of an Assembly Sequence

In the representation form of a direct assembly graph (fig. 3) each node of the graph contains within itself the total state of the whole product. This includes all the components and their current state assembly. The advantage of this method is the strict formalization, because all the information at each point in time is always contained within each of the nodes. On the downside this is bought dearly by a high number of nodes and a complicated graph structure when the product structure becomes more complex. This method will not be explained in more detail here.

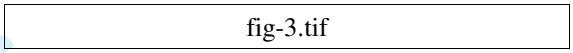


Fig. 3: Direct assembly graph with two alternative routes

2.2 Suitable Methods for Assembly Simulation

In the following a literature survey on AND/OR graph and Petri nets is presented. As they have many features in common, both are relevant to the field of DES simulation. Whereas Timed Petri nets (where a transition takes a certain time to complete) have been investigated in much detail, the contribution of the authors lies in the formulation of a Timed AND/OR graph which in some cases might be an appropriate way of describing and examining the structure of an assembly system.

a) The AND/OR Graph

The AND/OR graph can be seen as a further development of the Gozinto graph: Each node is a (completed) component (fig. 4). The hyperarcs connecting the nodes represent assembly operations. In other words, a hyperarc connects by an AND function multiple items, which together result in a new item. In some cases such a resulting item can be “born” in more than one way. These alternative possibilities are realized by adding more hyperarcs, analogous to an OR function. Strictly speaking one would even need to call it “Exclusive OR” function and “AND/XOR graph”.

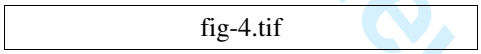


Fig. 4: AND/OR graph for product ABC

De Mello and Sanderson (1990) introduce the AND/OR graph. They use a top-down approach, i.e. start at the final product and progress through all the required parts, and then their required parts and so on. It is shown that disassembly can be seen as the (logical) reverse of assembly (in a logical sense, even if the disassembly process is technically not feasible). The authors note that “the AND/OR graph has substantially fewer nodes than the directed graph of assembly states, and this advantage becomes greater as the number of parts increases.” It is demonstrated that AND/OR graphs can be efficiently used for opportunistic scheduling in robotics.

Fatikow *et al.* (2000) describe a method to generate an assembly graph from an object model of the product. From criteria like freedom of separation and visibility of components, a plan of all possible assembly sequences is generated. These are analyzed to find the best one (in terms of cost of operations). The assembly graph is defined as follows: “Each node of the graph represents one particular assembly operation, and its edges represent the relations between two operations.”

Senin *et al.* (2000) present an assembly graph with the specific assumption: In a single assembly operation, no more than two components are joined together. The authors see assembly planning as a decision problem, and introduce the concept of alternative assembly plans in the form of AND/OR graphs. For analysis it is shown that each disassembly plan is a fact a reversed assembly plan, but can have better performance, because only so-called feasible decompositions need to be evaluated and unfeasible plans can be avoided. Plan execution time (duration) is used as an objective for comparing plans and this is demonstrated in an industrial case study. The conclusion is that genetic algorithms are better than combinatorial approaches with more complex problems and when response time is important.

Tseng and Liou (2000) introduce a combined assembly and machining graph as an approach towards integration of assembly and machining planning. They call it (in its final stage): Assembly-Machining Sequence Tree (AMST). This AMST has "entity nodes" and "operation nodes".

Relange and Henrioud (2001) present an algorithm to generate Assembly state transition diagrams (ASTD) from assembly graphs. An assembly graph is a binary tree: this means that in each step, two components are joined. This approach is similar to that proposed by Senin et al (2000). An ASTD represents all possible states during the assembly of a product and also includes the transitions between them. The presented method is much faster than building a valid ASTD by first generating a complete one and afterwards pruning it.

Nicosia *et al.* (2002) study a simple assembly line balancing problem (SALBP) using an acyclic assembly graph, the nodes of which represent assembly operations. Resources (workstations) are bound to each node (1:1). The solution is found with dynamic programming (DP) by generating a state graph where each node represents a set of finished operations.

b) Assembly Petri Net (APN)

Assembly as a process can be formalized to such an extent that Petri nets become a good modeling instrument. Places in the Petri net correspond to items, transitions equal operations. The presence of a token at a place means that there is an object in the current state of assembly. Generally speaking a Petri net does not only contain static information (about the structure of the assembly process as such) but also dynamic data (about the concrete assembly progress of the product). These properties make APN a powerful tool for the simulation of these processes.

By not imposing a capacity limit on its places, one can describe many instances of a product with one single Petri net at the same time (because multiple tokens can be located at one place). In this case it does not matter if there are 2 or 100 parts in production, it is only important that all items of the same type are identical and interchangeable.

In its most basic form the Petri net does not include timing information for transitions, because it is assumed that all transitions always fire instantly. This limits the usefulness to some extent. To overcome this problem, Timed Petri nets were developed. On the other hand for certain analyses (e. g. reachability graph) the timing aspect is not important at all, so that the same algorithms as for basic Petri nets can be used here.

As has been shown, creating an “artificial” reset or loop transition which connects the output and the input places is essential for certain desirable properties of the net, e. g. liveness (Thomas 1993). Having no practical counterpart it merely exists for the task of restarting the net after having finished the final product. This step is deemed essential to be able to apply Petri net theory to the problem. Fig. 5 shows an example of such a Petri net.

fig-5.tif

Fig. 5: Assembly petri net with reset transition

Kanehara *et al.* (1993) propose a method to find an optimal task sequence in assembly Petri nets by using Linear Programming. In order to achieve this, they use the graph structure of the APN which is mapped from the AND/OR net. This way of solving the problem shows the strong interconnection between AND/OR graphs and Petri Nets, and how both can be combined to describe assembly.

Xiong and Zhou (1997) present a scheduling method which uses a Timed Petri net. This is used in a search for a (best) schedule which does not contain a deadlock marking. The method described in this work can be used for multiple lot sizes and finite buffer sizes, where mathematical programming techniques fail. The time aspect of the operations in the Petri net is the important part here: “Time ... between the start and the end of an operation ... is represented by associating timing delay with the corresponding operation place.”

Zussman *et al.* (1998) propose a disassembly Petri net (DPN) for the modeling of disassembly processes. Conceptually disassembly can be treated the same as assembly, only backwards. For evaluation of alternatives a cost function is introduced, which describe the costs of performing a particular operation. Using a practical example (disassembly and repair of a telephone handset) the introduced concepts are illustrated.

According to Zhou and Venkatesh (1998) the advantages of the Petri net as a tool in manufacturing automation are the ability to detect undesirable properties (e. g. deadlock situations) by mathematically based analysis before doing possibly time-consuming simulations, and the identification of bottlenecks in the system.

Mînzu *et al.* (2001) propose a way of modeling an assembly system as a DES in order to solve a control problem. The reason for this is clear: “As a discrete event system, the assembly workstation is better modeled like a Petri net, because the prerequisites for the execution of any task are more explicit.” The Petri Net is then transformed into a so-

called “Controlled Assembly Petri Net” by introducing controllable transitions. This in turn is converted into an automaton model.

Newer works deal with special types of Petri Nets for on-line modeling: Morandin and Kato (2005) propose a Virtual Petri Net for on-line modeling of flexible manufacturing systems. This method also consideres shared resources and alternative process planning. Chung and Lee (2005) introduce the Augmented Petri Net to model and control assembly tasks in unstructured environments taking into account the uncertainties that such environments bring about. Even though not developed with the focus on discrete event simulation specifically, these last two methods are still useful in this field of application because of their on-line capabilities.

c) Timed AND/OR graph

The Timed AND/OR graph is an enhancement to the regular AND/OR graph. It was designed by the authors to remedy the problem of the missing time information in these graphs. Instead of doing a straightforward attribution of the duration directly to the hyperarc (which would impact clearness and readability of these diagrams), another path was chosen:

1. In a first version the nodes of the graph were enhanced by special “docking areas” for the hyperarcs and attributed the duration of operations to these areas. This means that all docking areas of one node are implicitly ORed. This method is formally unambiguous, but it contradicts the principle of maximum simplicity for graphs (only simple nodes and arcs in their most basic form should be used if at all possible).
2. That was the reason for version two: by decoupling nodes and the docking areas a distinct type of operation node (or “transition” in Petri net speak) was created, which by itself is connected to other item nodes via OR arcs.

The graph has the following properties: It is directed, acyclic, bipartite, and of a mostly tree-like structure. As can be seen, there is an analogy to the structure of Petri nets. Some examples of these graphs follow in section 3.2.

3. Simulation of Assembly Processes

3.1 The Simulation System

The previous remarks have shown that assembly processes can be described by Timed Petri nets as well as by AND/OR graphs. Otherwise Petri nets are suitable to only a limited extend as a basis for simulation models, because their abstract units “place” and “transition” would require a too high effort of modeling. Hence most of the simulation systems which are used in practice are based on more complex and more realistic basic units. The simcron MODELLER used here is a DES system with typical simulation components. In the context of this paper it is an exemplary system and can be replaced by any other simulation system with similar components.

The simcron MODELLER is controlled by work plans which are called technologies here. A technology describes a sequence of operations which are stringently executed one after another. An operation n requires one or more resources as well as a deterministic/stochastic operation time Δt_n. Fig. 6 shows a cut-out of such a technology T which consists of the three successive operations “op n-1” to “op n+1”. Each operation can be executed only then, if its predecessor has been completed and all of the required resources are available (for the example the respective machines). For machines, queues and technologies special units are available in the simulation toolkit. The operations are elements of lists inside the technology objects, which in addition to the operation time and to the required resources have properties such as priority or specific due dates.

fig-6.eps

Fig. 6: Cut-out of a technology consisting of three successive operations

This basic structure is suitable for modeling manufacturing systems like flow shops or job shops, where the jobs have a significant more or less flexible route. In contrast assembly processes can be described with sufficient accuracy only by tree or graph structures. So one has to ask, if assembly processes can be effectively modeled at all by the simulation system introduced just now or if an extended object library would be necessary for it. The so-called need object can solve this problem. These objects were originally designed for synchronization between several operation steps and for resolving bill of materials respectively. Each need object has a reference to a particular operation step and records, if and how many times this operation step was executed in the past. Again, each operation step can contain a list of need objects. There is the rule that an operation step is executable only then, if all

its need objects in its list are satisfied. Because need objects can also be defined between several technology objects, this simply opens up the possibility to model tree-structures as shown in Fig. 7. In this example the operation step *n* of T1 is executable under the condition that – except for its direct precursor inside of T1 – the two operations *A* and *B* of the technologies T2 and T3 are already completed. With the aid of these need objects simple linear work flows can be interweaved to complex tree or net structures, as will be shown in the following section.

fig-7a.eps	fig-7b.eps

Fig. 7: AND-connection of several operations by need-objects (right: resource-oriented view)

3.2 From the Assembly Graph to the Simulation Model

As shown in section 2.2, assembly processes can be described for simulation purposes by means of Timed AND/OR graphs. Now it will be shown how this is done by combining basic graph elements and in a further step these graphs will be transformed into a simulation model. The simulation is then performed by sending jobs through the model.

Simple assembly steps

Fig. 8 shows a simple graph describing the assembly of two components called A and B into a final product which will be called AB after its components. The assembly operation takes 20 minutes and can only begin when both A and B are present. After the operation is finished, the final product comes into existence. The corresponding simulation model is shown in Fig. 8 to the right. Note: Jobs (the driving elements of the simulation) are not shown here.

fig-8a.tif	fig-8b.tif

Fig. 8: A simple assembly graph

In contrast to Senin *et al.* (2000) such an assembly operation might also be performed on more than two distinct components in quite the same fashion.

Branching points: The Case of the Common Components

Now consider another product with two components, to be called BC. It includes the already known part B and a new component C. Instead of simply drawing the graphs for AB und BC next to each other, they are combined in such a fashion that they overlap in B and have this node in common. The result can be seen in Fig. 9 alongside its simulation model.

fig-9a.tif	fig-9b.tif

Fig. 9: A branching point at item B

This also corresponds better to reality, in which there is a shared store or queue from which these common components are fetched on demand. From the point of view of item B, it is unknown beforehand if B is to be made into a product AB or a product BC. This is not decided until the moment the respective assembly operation starts (or is scheduled to start).

Merging points: Alternatives

Looking at more and more complex products one might come across a case where the final product can be obtained on alternative assembly paths. What does that mean? One can imagine the already introduced product AB to be part of another product ABC. This can possibly be produced by assembling AB and C. But maybe technologically there exists a second way: First make an item BC which is then combined with A to form the final product ABC.

This hypothetical situation is depicted in Fig. 10. This means that for the assembly of ABC there are two alternatives. In the assembly process the two subgraphs are merged at node ABC. In Boolean logic it can be written as follows: $ABC = (AB \wedge C) \vee (A \wedge BC)$.

fig-10a.tif	fig-10b.tif

Fig. 10: A merging point for item ABC

The question arises: how to assure the exclusiveness of these alternatives in order to prevent a deadlock situation in the proposed simulation model? For a large-scale production there is a relatively easy answer, because more than

one product will be made and therefore multiple alternatives can coexist at the same time: so no problem here. For the assembly of one single product proceed as follows: As soon as one of the operations (AB+C or A+BC) has started, the job occupies the station ABC, which is given a space capacity of exactly one. This assures that the other operation or operations can not start. For the duration of this operation the job rests now in "busy" state in station ABC. After that it switches to "ready" state (but still occupies the same station).

Now that the basic elements of building the simulation model have been introduced all that remains is to put the building blocks together: The result can be seen in Fig. 11 which shows the complete assembly graph for the proposed product ABC.

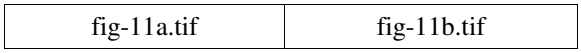


Fig. 11: The complete assembly graph for product ABC, with several branching points and a merging point

4. Summary and Outlook

It was shown that assembly processes have a tree-like structure and explained why the method of simulation can be applied to the scheduling of such processes. Different methods and ways of describing them exist and some of them were analyzed. Especially AND/OR graphs and Petri nets are relevant to building simulation models. The Timed AND/OR graph was introduced as a very compact representation form for assembly plans.

It was presented how the specific structure of assembly graphs can be transformed into a simulation model by using need objects of the DES simulation software. It was demonstrated that potential deadlock situations can be handled by taking advantage of the space capacity limiting feature of the modeling system.

The authors plan to use the methods presented here for scheduling processes of stationary assembly of large milling machines and newspaper presses. Work on this is underway. The process of creating the assembly graph, simulation model and resulting Gantt chart of such a process can be seen in Fig. 12. The complexity of these models is already large enough so that manual model creation (via a graphical usage interface) is unfeasible. That's why it was decided to use automated model creation by means of a relational database from the start (Weigert *et al.* 2006).

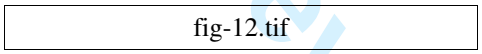


Fig. 12: The simulation cycle: from the real world to the resulting Gantt chart

Further work planned is to apply these methods to more general processes which are similar to or include assembly operations. This includes transport or disassembly problems.

Acknowledgements

The research is funded by the German Federal Ministry of Education and Research.

References

Chung, S.Y. and Lee, D.Y., 2005. An augmented Petri net for modelling and control of assembly tasks with uncertainties. *International Journal of Computer Integrated Manufacturing*, Vol. 18, Issue 2, pp. 170–178.

Fatikow, S., *et al.*, 2000. A Flexible Microrobot-Based Microassembly Station. *Journal of Intelligent and Robotic Systems* 27, pp. 135–169.

Homem de Mello, L.S. and Sanderson, A.C., 1990. And/or Graph Representation of Assembly Plans. *IEEE Transactions on Robotics and Automation*, Vol.6, No.2, pp.188–199.

Kanehara, T., *et al.*, 1993. On Algebraic and Graph Structural Properties of Assembly Petri Net. *In: Proceedings of the 1993 IEEE Intemational Conference on Intelligent Robots and Systems*, Vol. 3, pp. 2286–2293, Yokohama, Japan.

Mînzu, V., Cernega, D., and Henrioud, J.M., 2001. Linguistic Model and a Control Problem for Assembly Workstation. *In: Proceedings of the 4th IEEE International Symposium on Assembly and Task Planning*, Fukuoka,

Japan.

Morandin Jr, O. and Kato, E., 2005. Virtual Petri nets as a modular modeling method for planning and control tasks of FMS. *International Journal of Computer Integrated Manufacturing*, Vol. 18, Issue 2, pp. 100–106.

Nicosia, G., Pacciarelli, D., and Pacifici, A., 2002. Optimally balancing assembly lines with different workstations. *Discrete Applied Mathematics* 118, pp. 99–113.

Relange, L. and Henrioud, J.M., 2001. Systematic determination of Assembly state transition diagrams. *In*: *Proceedings of the 4th IEEE International Symposium on Assembly and Task Planning*, Fukuoka, Japan. May 28–29.

Senin, N., Groppetti, R., and Wallace, D.R., 2000. Concurrent assembly planning with genetic algorithms. *Robotics and Computer Integrated Manufacturing* 16, pp. 65–72.

Thomas, J.P., 1993. Constructing Assembly Plans, *IEEE International Conference on Robotics and Automation*, pp. 515–520.

Tseng, Y.-J. and Liou, L.-C., 2000. Integrating assembly and machining planning using graph-based representation models. *International Journal of Production Research*, Vol. 38, No. 12, pp. 2619–2641.

Weigert, G., *et al.*, 2006: Automated Creation of DES Models in an Industrial Environment. *In: 16th International Conference FAIM'06, Flexible Automation & Intelligent Manufacturing*, pp. 311–318, Limerick.

Xiong, H.H. and Zhou, M.-C., 1997. Deadlock-free scheduling of an automated manufacturing system based on Petri nets. *In: IEEE International Conference on Robotics and Automation*, Vol. 2, pp. 945–950, Albuquerque, USA.

Zhou, M.-C. and Venkatesh, K., 1998. *Modeling, Simulation, and Control of Flexible Manufacturing Systems: A Petri Net Approach*. Singapore: World Scientific.

Zussman, E., Zhou, M.-C., and Caudill, R., 1998. Disassembly Petri net approach to modeling and planning disassembly processes of electronic products. *In: IEEE International Symposium on Electronics and the Environment*, pp. 331–336, Oak Brook, USA.

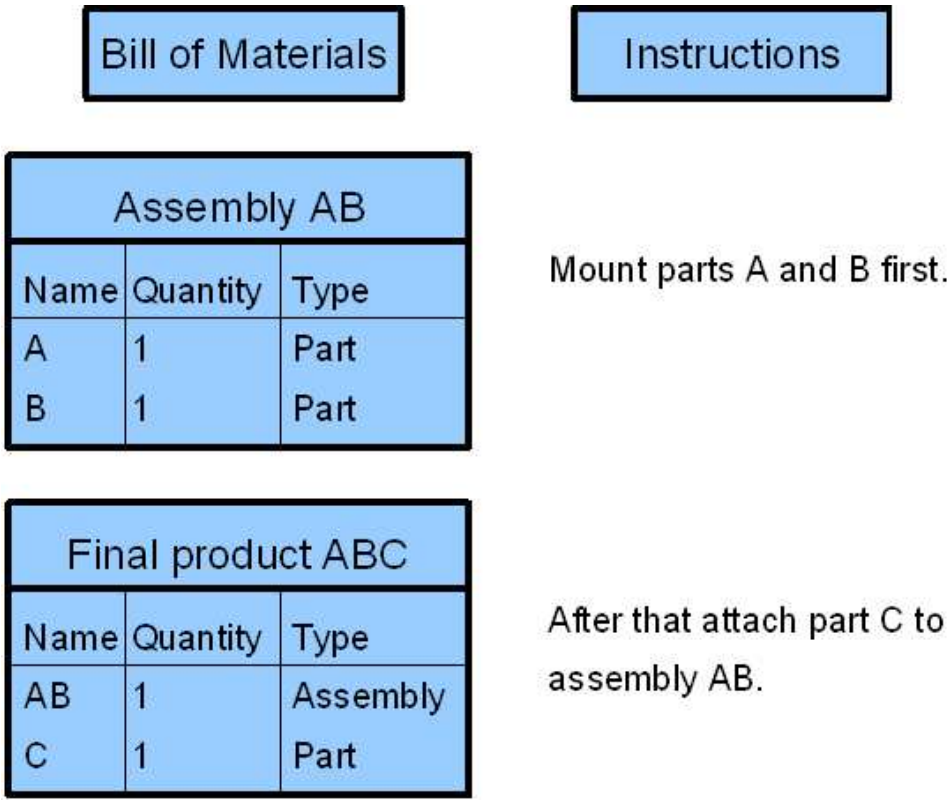


Figure 1: Bill of materials and an informal assembly description

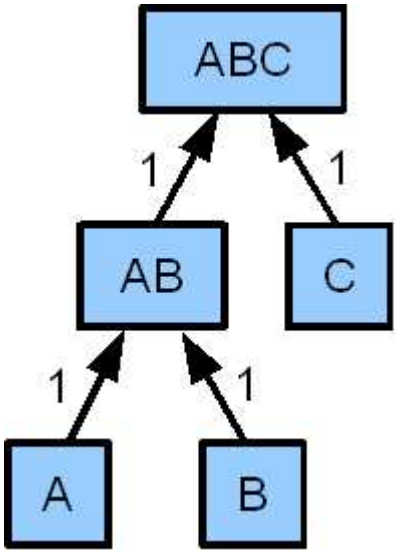


Figure 2: Gozintograph of product ABC

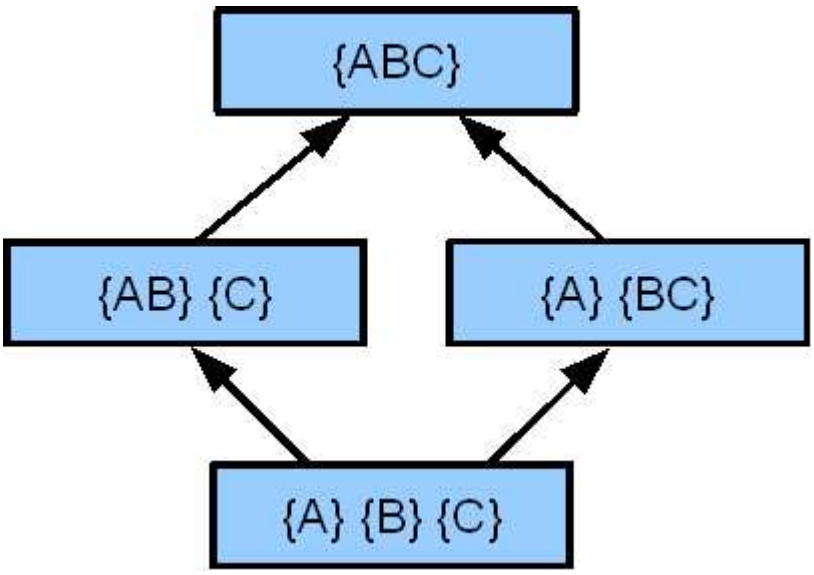


Figure 3: Direct assembly graph with two alternative routes

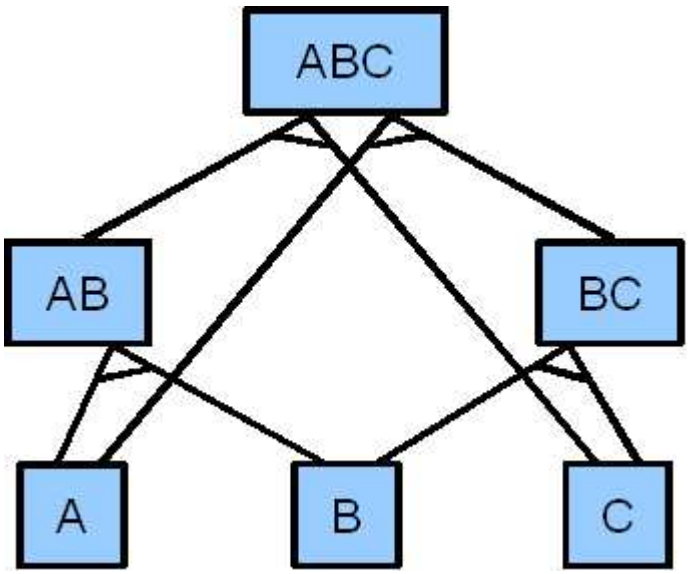


Figure 4: AND/OR graph for product ABC

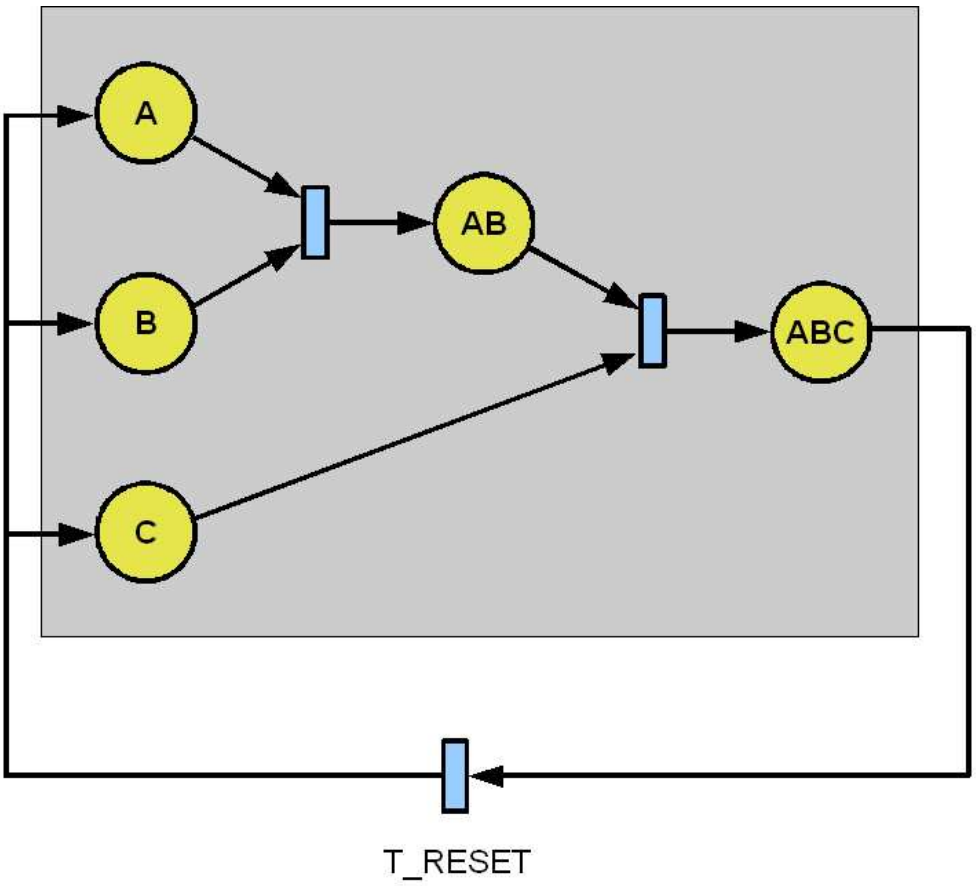
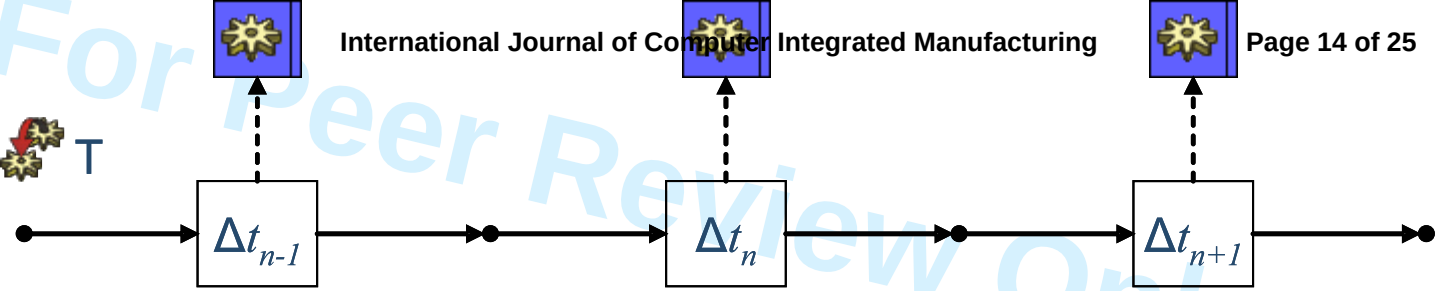
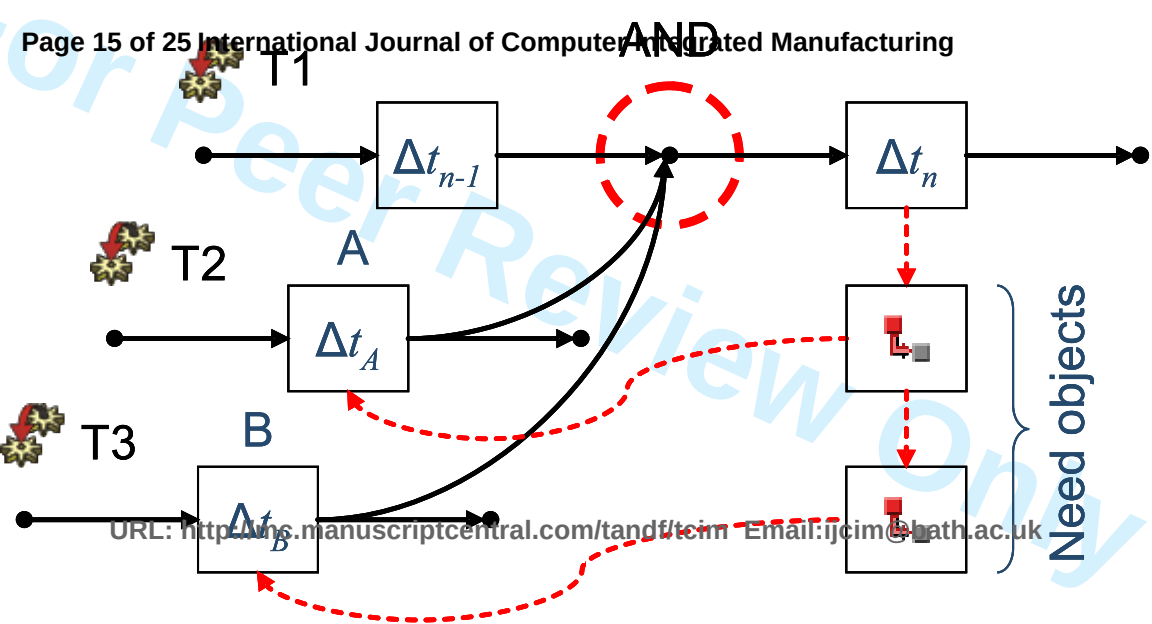
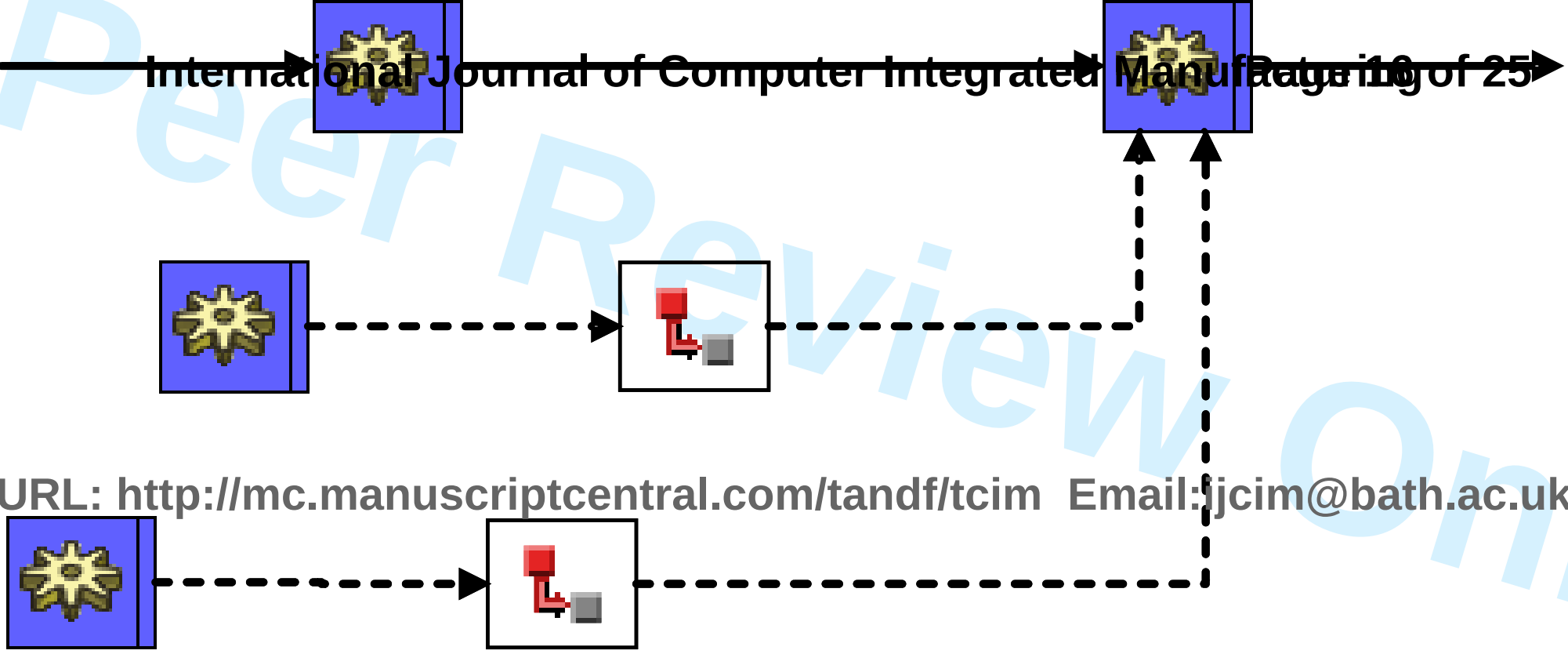


Figure 5: Assembly petri net with reset transition







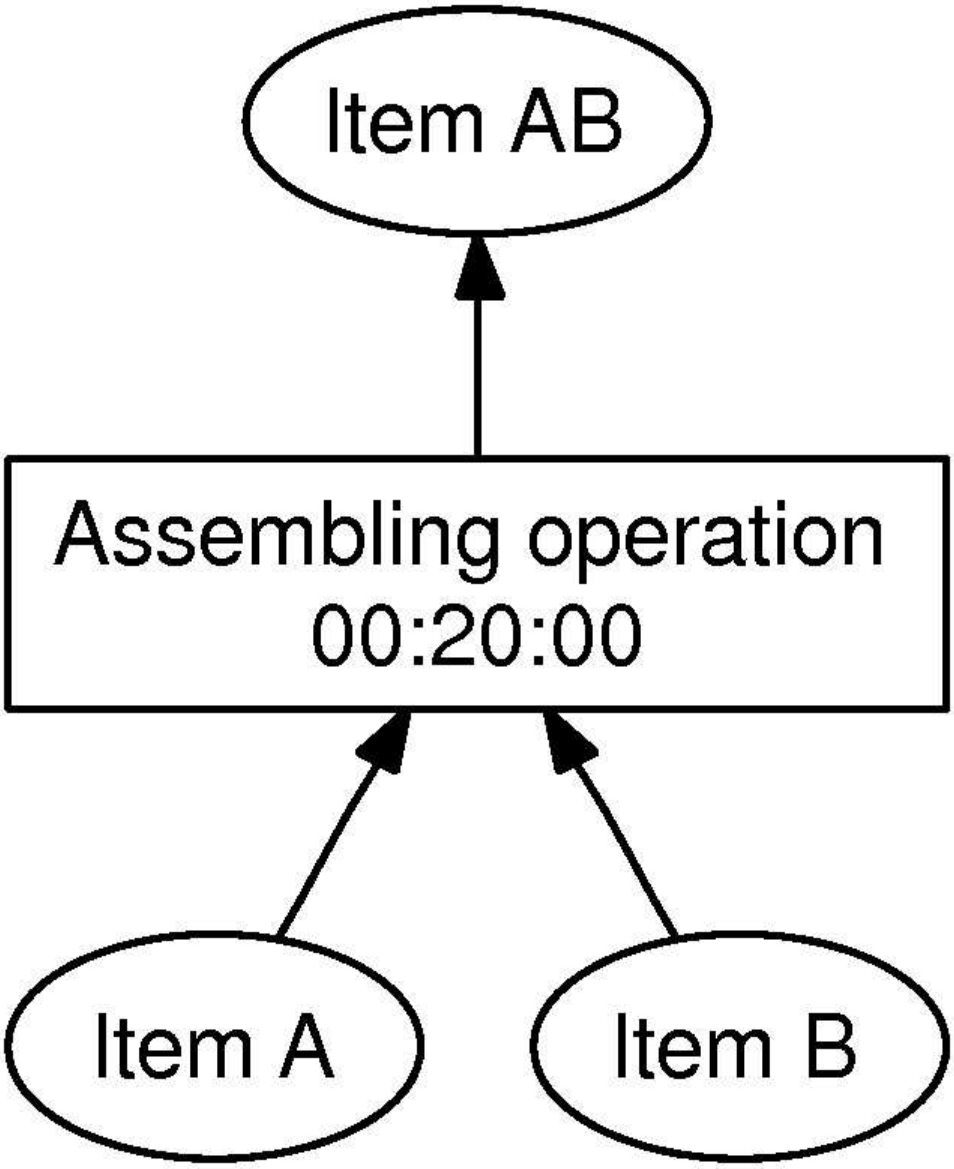


Figure 8: A simple assembly graph
55x67mm (300 x 300 DPI)

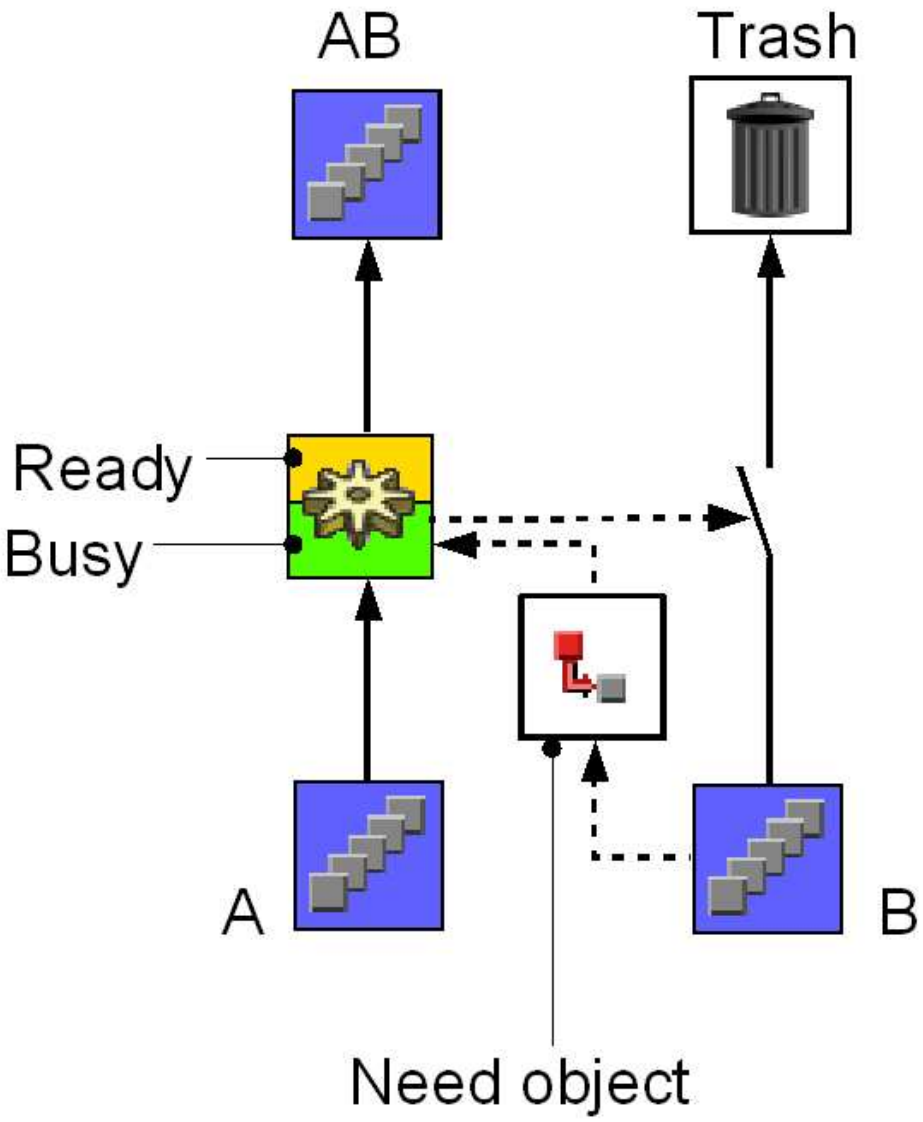


Figure 8: A simple assembly graph
154x175mm (96 x 96 DPI)

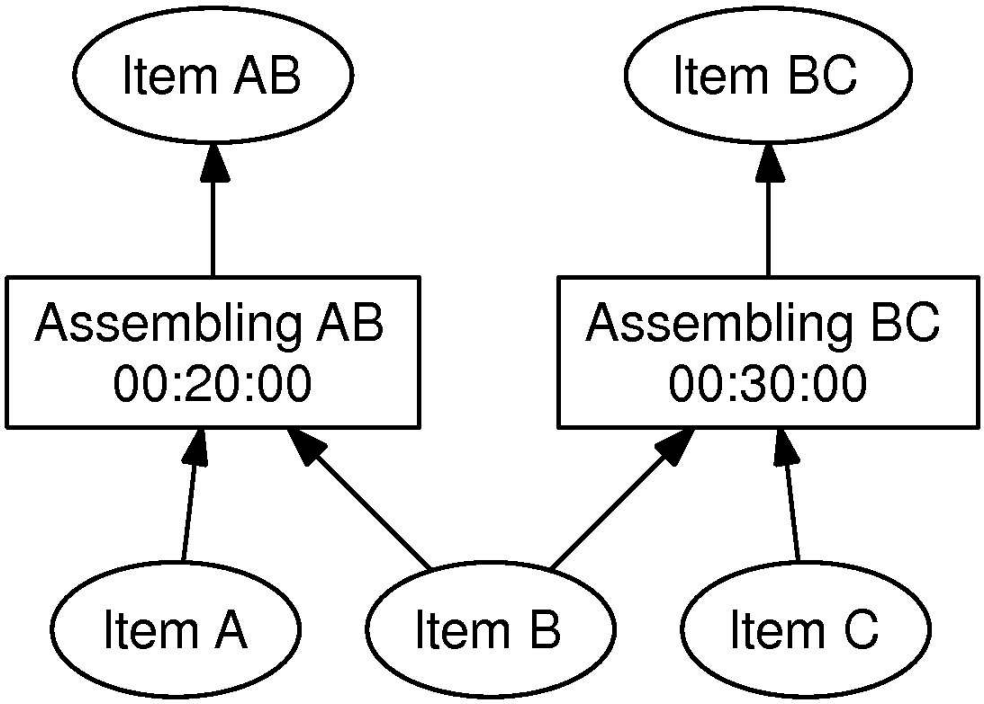


Figure 9: A branching point at item B
94x67mm (300 x 300 DPI)

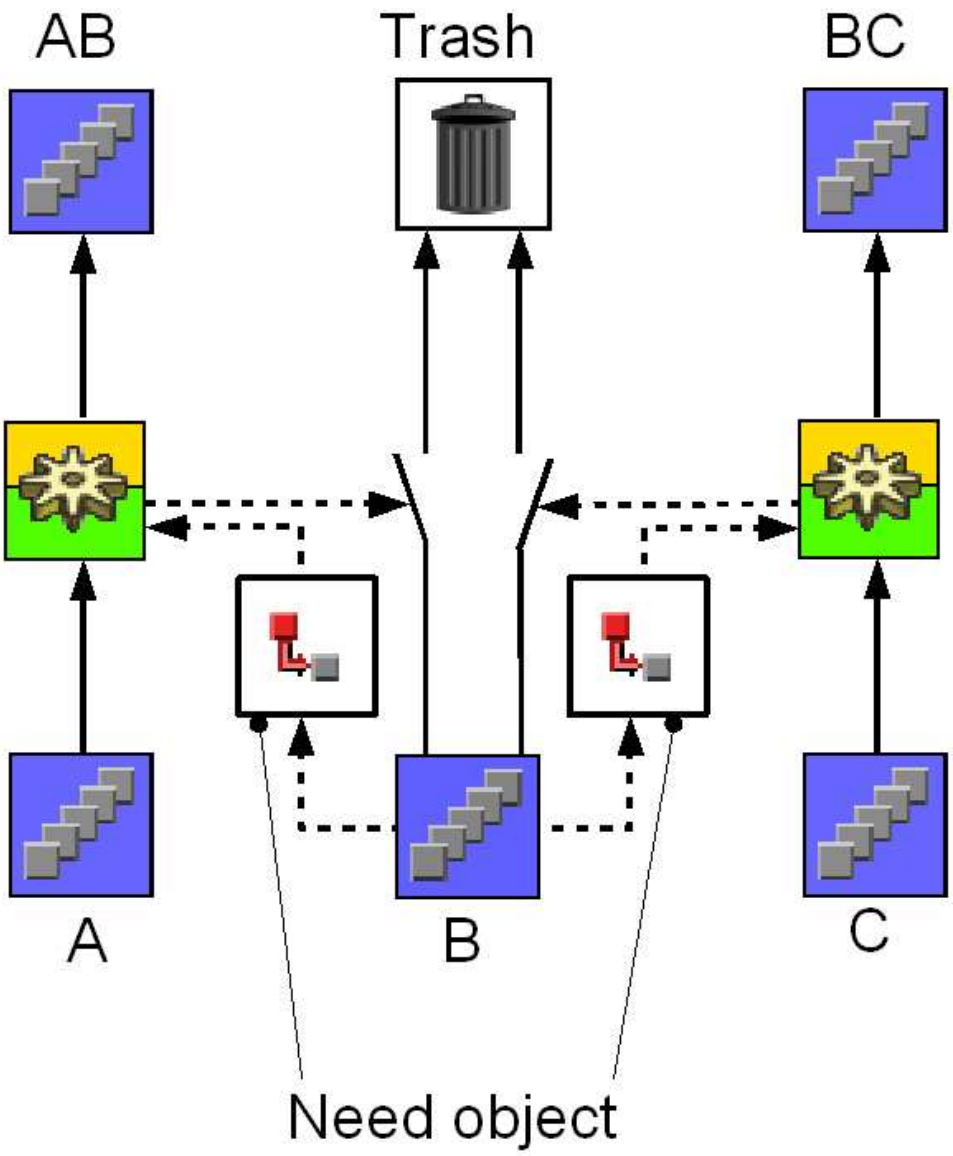


Figure 9: A branching point at item B
160x187mm (96 x 96 DPI)

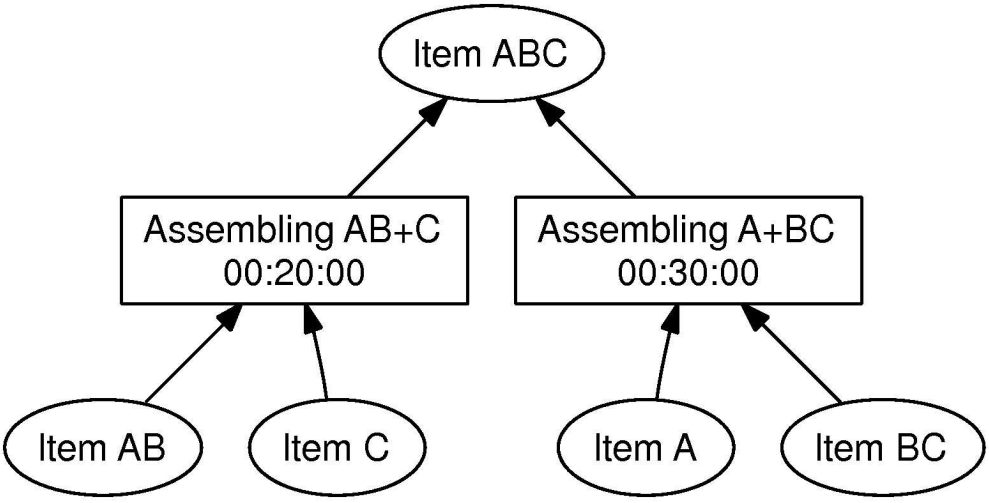


Figure 10: A merging point for item ABC
132x67mm (300 x 300 DPI)

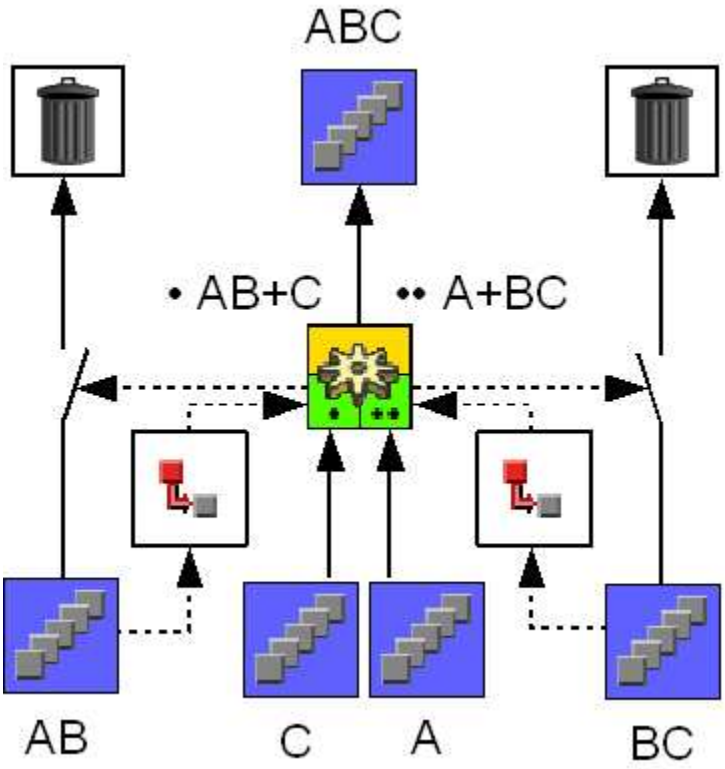


Figure 10: A merging point for item ABC
95x105mm (96 x 96 DPI)

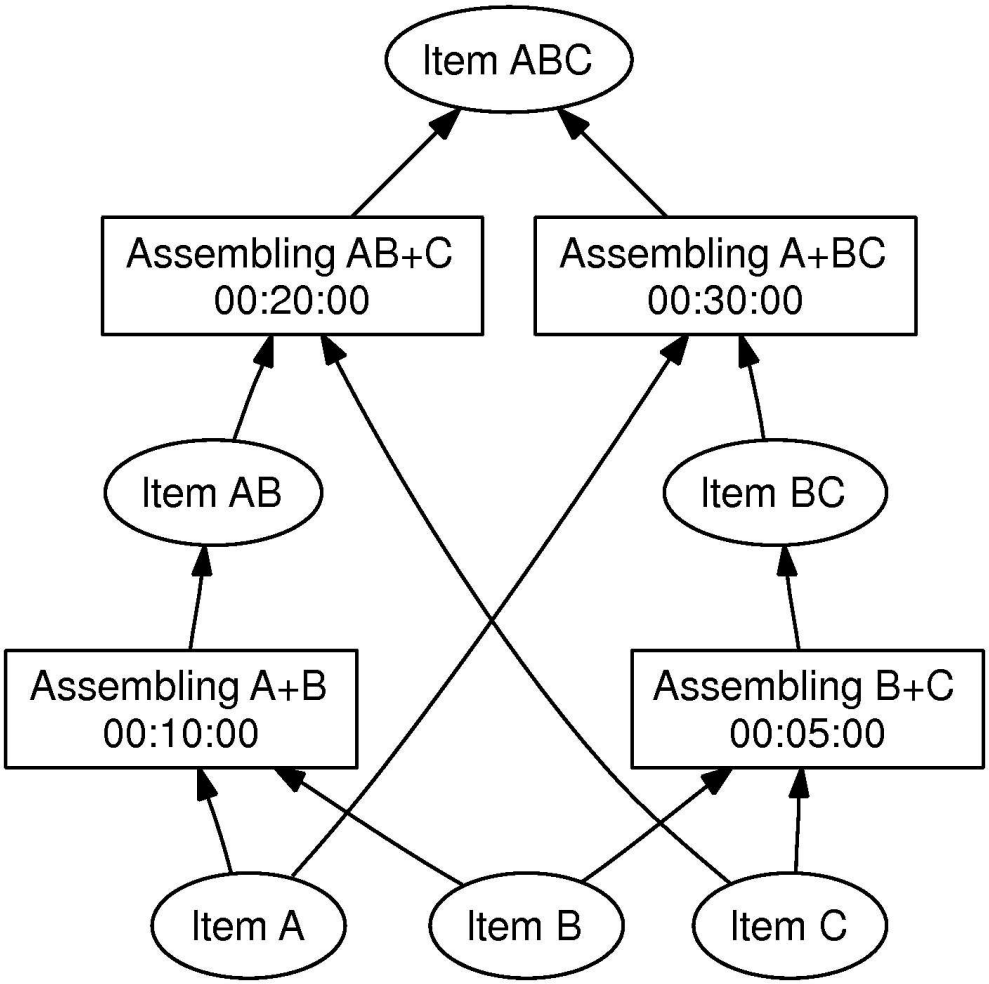


Figure 11: The complete assembly graph for product ABC, with several branching points and a merging point
120x119mm (300 x 300 DPI)

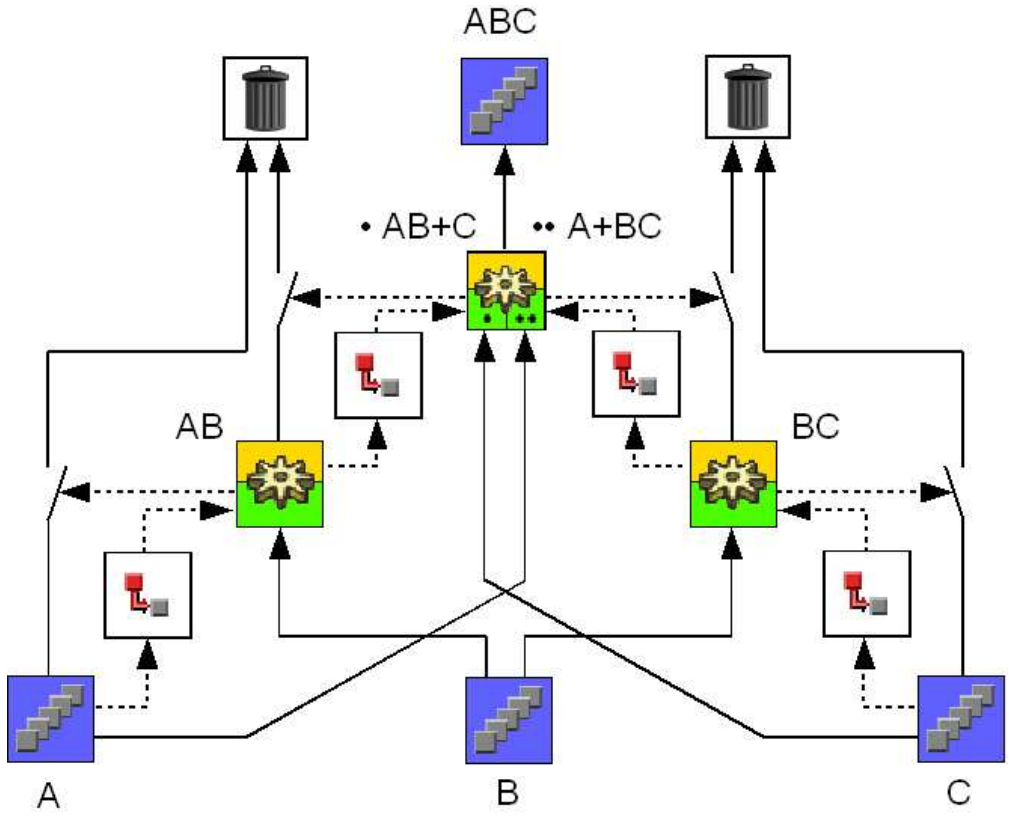


Figure 11: The complete assembly graph for product ABC, with several branching points and a merging point
175x146mm (96 x 96 DPI)

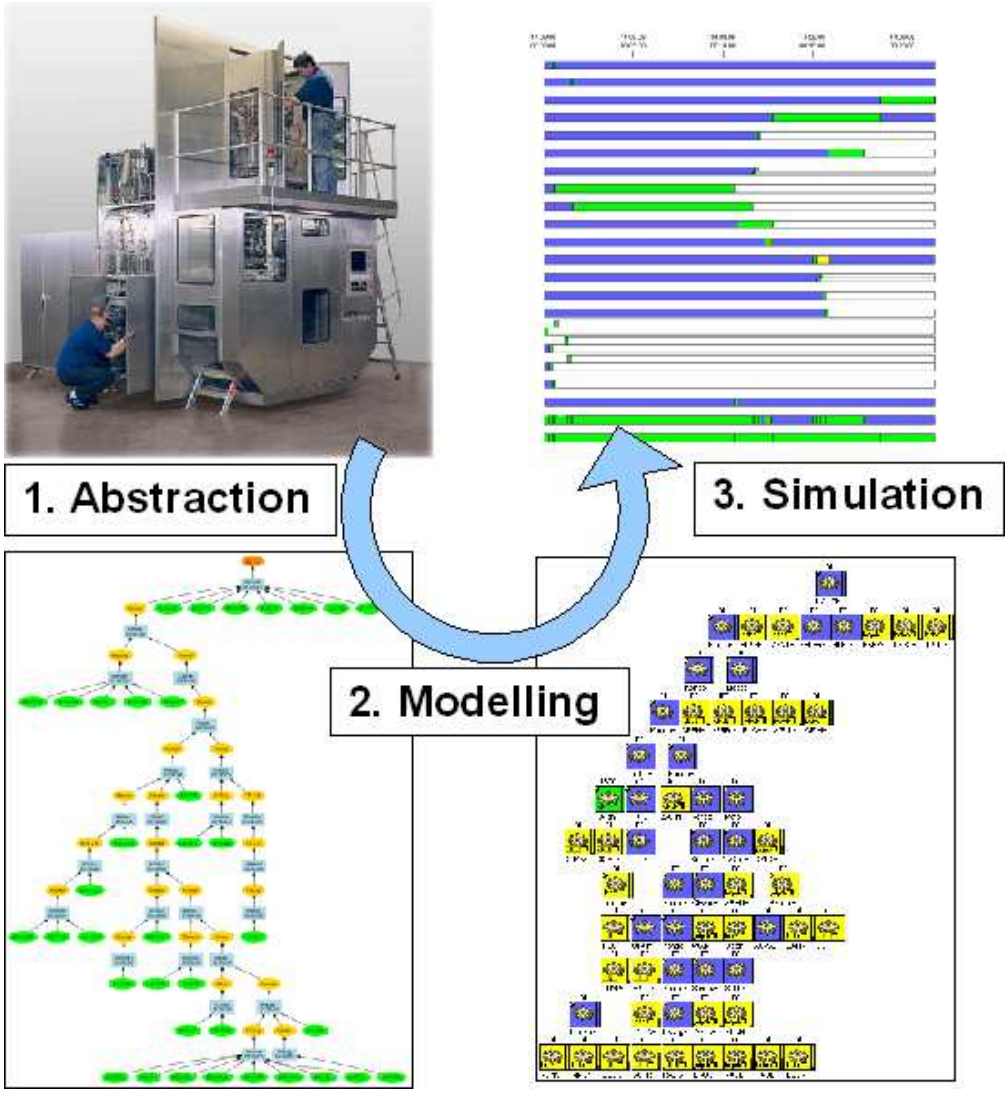


Figure 12: The simulation cycle: from the real world to the resulting Gantt chart