Proceedings of the 2003 Winter Simulation Conference
S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, eds.

DESIGNING REUSABLE SIMULATION MODULES FOR ELECTRONICS MANUFACTURING SYSTEMS

Phani S. Mukkamala
Jeffrey S. Smith
Jorge F.Valenzuela

Industrial and Systems Engineering Department
Auburn University
Auburn University, AL 36849, U.S.A.

ABSTRACT

Developing simulation models for related problems in the same domain is generally a repetitive process. Such simulation models are similar in many aspects and have only minor differences. Modeling efforts can be reduced to a great extent through the development of domain specific modules or templates that encapsulate the domain-specific logic and hide many of the modeling details. This paper describes the development of such a domain-specific template for electronics assembly. In particular, the template focuses on the automated assembly of printed circuit boards. The template encompasses the complexity of the target domain and simplifies the model-building process. While the paper focuses on a language-neutral description of the template, specific experience with Arena is described.

1 INTRODUCTION

Simulation modeling is one of the most popular and useful system analysis techniques. Most of the systems we come across in the real world are complex and stochastic in nature with a number of input and output variables. Analytical models for such systems often require simplifying assumptions causing concern over the applicability of the solution. Simulation is often the preferred choice to model complex and stochastic systems. Simulation models help us in understanding the interactions going on inside the complex systems which are otherwise difficult to consider in other analytical models.

Simulation modeling is designed to provide numerical results to the decision makers, who use these results to make conclusive design and operational decisions. In general, the simulation specialists and the decision makers have different backgrounds and more often than not, the simulation specialists lack the domain-specific knowledge required to model the system. As a result, the simulation specialists and the decision makers must work closely to develop usable models.

Models are often developed from the scratch to solve a specific problem. Within a given domain, many of these models are very similar and could be generalized for use in similar environments. Unfortunately, the models are often discarded once the project is finished, ignoring the opportunity to leverage the time and efforts that went into the models' development. Such problems can be avoided to some extent by simplifying the simulation modeling efforts by choosing one of the following options (Paul and Taylor 2002): a) reuse of modeling components b) reuse of model subsystems c) reuse of a similar model.

Templates are essentially reusable components with model subsystems encapsulated and hidden inside. They occur in three tiers (Valentin and Verbraeck 2002a). A queue or a resource with user interface to ease modeling can be considered as building blocks (or modules) at the lowest level. Building blocks with focus on a specific domain come at the middle level. Building blocks that can be extended according to the needs of the user come at the highest level. Their common characteristics are: a) independence, b) reusability, c) replaceablity, d) adaptability, e) effective user interfaces and f) internal structure encapsulation (Valentin and Verbraeck 2002b).

The use of templates comes with many advantages. The time taken to develop domain-specific models can be reduced considerably as the templates can be used in more than one simulation study. They help in easy conceptualization and understanding of the model due to their domain-specific names and visual characteristics. The templates are verified and validated, simplifying the overall model verification and validation. Significant complexities of the model are hidden within the templates (i.e., encapsulated within the templates). The use of templates also simplifies model maintenance and extension.

This paper describes the development of a domain-specific template for the automated assembly of printed circuit boards. This paper is organized as follows. Section 2 provides a brief review of existing literature relating to the development of templates and generic models. Section

3 describes the target domain of electronics manufacturing and the general concept for the template. Section 4 describes the details of the template. Finally, Section 5 presents the conclusions.

2 LITERATURE REVIEW

Simulation model reuse and template development have been described for many years. We discuss only a small subset of these in this paper. Thesen (1990) has developed a generic model called *template based simulator* (TBS) which requires the user to input only the coefficient values for each defined element. The templates defined in TBS include: work centers, routings, parts families, material handling systems and scheduling rules. Prasad (1990) discusses the development of a generic simulation model to characterize photolithography area in a semiconductor FAB facility at Intel. Ozdemirel and Mackulak (1993) describe the development of generic modules for manufacturing systems using a group technology (GT) classification scheme. Paul and Kuljis (1995) developed a simulation modeling package called *CLIMSIM*, which is a general purpose model for addressing the issue of waiting times in outpatient clinics and can model any hospital.

King and Kim (1995) developed an object oriented simulation tool called *AgvTalk* consisting of a library of classes used for the design and analysis of automated guided vehicle (AGV) systems. Farrington et al. (1996) introduce software tools for the rapid modeling of electronics manufacturing systems. The proposed architecture consists of three elements: a line definer, a static analyzer and a code generator; connected through data transfer links and feedback loops. The code generator consists of modules capable of generating input files for simulation languages like WITNESS and ARENA. The templates described in this paper are similar in concept to the work described by Farrington et al. (1996). AbouRizk et al. (1999) describe the special purpose simulation template developed in Simphony, useful for the simulation of tunneling operations in the construction industry.

Son, Jones and Wysk (2000) discuss the development of neutral libraries of simulation components and model templates useful in discrete event simulation of flow of jobs through a job shop. Ho et al. (2002) discuss the development of a general purpose simulator consisting of a library of modules for modeling various subsystems within the railway system. Anglani et al. (2002) present an object oriented procedure named *UMSIS* (UML Modeled SIMAN Implemented Simulation software) to develop flexible management system simulation models.

Thus there has clearly been a significant amount of research and development devoted to the development of generic models and templates where repetitive modeling efforts are involved. However, despite this research, there are few commercially-available templates specifically focused at electronics manufacturing systems.

3 ELECTRONICS MANUFACTURING

3.1 Overview

Electronics manufacturing involves the design, development, assembly and testing of electronic components, parts and tools (Hollomon 1989). There are two primary categories of electronics manufacturing - *surface mount technology (SMT)* and *insertion mount technology (IMT)*. SMT offers some significant advantages over IMT, which has resulted in a rapid growth of its market share in recent years. The current version of the template focuses on SMT. SMT manufacturing styles are classified broadly into *three* kinds based on the soldering approach used. *Type 1 SMT* is a pure surface mount process and uses only SMCs. Parts can be populated on both sides of the board. In *Type 2 SMT*, both SMCs and IMCs are populated together on at least one side of the board and both sides of the board can be populated. In *Type 3 SMT*, all SMCs are glued to one side and the IMCs are inserted onto the other side.

The general SMT manufacturing process consists of the following steps: i) application of solder paste or adhesive ii) component placement iii) curing iv) reflow soldering and v) cleaning. In addition to these processes, automated or manual inspection processes can be inserted between the individual processes steps. Note that these general processes are the same regardless of the specific type of board being manufactured. It is this commonality of processing steps that makes electronics manufacturing particularly well suited for a common template.

From a modeling perspective, the following four issues account for almost 95% of the design/operational analysis problems associated with PCB manufacturing (Kiran, Kaplan and Unal 1993):

1. *Capacity analysis*: This is done to either analyze the capacity of the system keeping the number of machines fixed or to analyze the number of resources required to achieve a target capacity.
2. *Operator analysis*: To analyze the number of operators required for successful operation of the system.
3. *Material handling*: To determine whether the current material handling equipment is sufficient for current and/or future needs.
4. *Process improvement and automation*: To study the impact of potential process improvement ideas and automation procedures.

These are precisely the types of analysis problems for which simulation is particularly applicable.

3.2 Typical Machine: Description

A typical machine encountered in electronics manufacturing has three successive stages through which each of the boards passes – a) an *input area,* b) a *processing area,* and c) an *output area*. Boards move into the input area to await processing. Boards are then automatically moved from the input area into the processing area. If the processing area has a capacity of more than one, the boards may be processed in batches. Once processing has been completed, the boards are automatically moved into the output area. As such, the input and output areas provide buffer space for the processing area and typically have unit capacity.

The boards are moved between the machines using conveyors. A PCB reaches the input area from a *preceding* conveyor and is removed from the output area by a *succeeding* conveyor. It is these conveyors that provide the automated material handling between processing machines. In addition, the conveyors are generally *accumulating*-type conveyors and can provide additional buffering capacity between the machines.

During operation, machines frequently go down necessitating manual intervention by the line operator. The two common classes of downtime include *machine failures* and *part exhaust*. Machine failures occur when a machine encounters a problem that requires operator intervention. Part exhaust occurs when the supply of one or more of the components being placed on the board is exhausted, requiring the operator to replenish the supply. These downtimes represent critical modeling components that add to the complexity of the simulation models. The downtimes are characterized by the distributions of the times to occurrence, the distributions of the duration, and the resource(s) required for attendance (e.g., the operator or technician). These parameters and their implementations are described in more detail in Section 4.

The total time spent by a board inside the machine can be split into smaller categories based on the state of the machine. This is done to help us explain deviations in the time taken from the expected values. The state of the machine describes the condition of the machine. The machine begins in an idle state. With the processing step, the machine changes into busy state. If the machine goes down, it waits for an operator to arrive and fix the machine. This state is called as failure state. When the raw material in the machine is exhausted and the machine goes down for replenishment, its state is called as exhaust state. The state while the operator or the technician fixes the problem is either repair state or replenishment state. When the problem is fixed and the machine starts working, its state is changed to busy again. At the end of processing, the board leaves the machine if there is capacity available in the output area. If capacity is unavailable, the board remains on the machine and this state is called as blocked. Once the board leaves the machine, its state is turned back to idle. Thus, the seven states defined for the machine are: i) Idle, ii) Busy, iii) Failure, iv) Repair, v) Exhaust, vi) Replenish, and vii) Blocked. For the operator(s) and the technician(s), two states are defined: Idle and Busy. One of the primary simulation output categories is the relative time that the machines and operators spend in each state.

3.3 Typical Machine: Modeling

The modeling aspect of this typical machine has been discussed here. The machine's processing logic and failure logic are attached in the form of block diagrams for easier understanding. The processing logic, shown in Figure 1, is discussed first. The transporting elements preceding and succeeding the machine are conveyors. The input and output areas are assumed to have a capacity of at least one. The boards enter the input area (block 2) before exiting the preceding conveyor (5). The boards then seize (6) the resource before forming a batch (8). There is a delay block (10) to account for the processing time of the board. At the end of processing, the board batch is split (12) before the boards enter the output area. The boards leave the output area (17) if space is available on the following conveyor. Some of the blocks are for assisting us in the collection of statistics like the time spent by an entity in the process. The states of the machine have to be changed continuously as the entities flow across the machine.

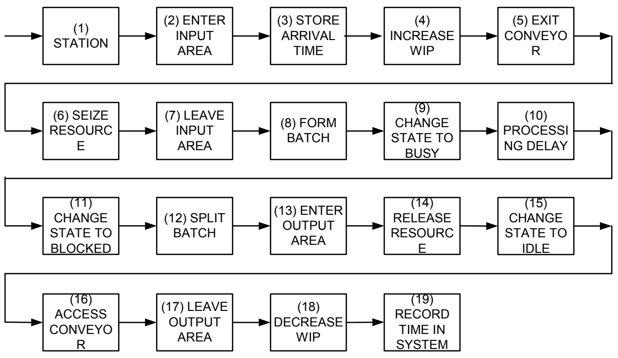


Figure 1: Processing Logic

The failure and exhaust processes of the machine are similar in most of the aspects except for the minor difference in their states and hence just the failure logic is discussed here. The failure logic is shown in Figure 2. Failures can be either time dependent or board dependent. We discuss only the time dependent logic here. The failure process runs with the creation of failure entities according to a specified distribution of failure time (1). The failure process has a queue (2) and a seize block (3) for seizing the resource. This queue has a higher priority than the regular queue from the processing logic above. After the machine is seized, the operator (and/or other required resource) is seized by a separate seize block (5). Once the operator responds and reaches the machine, he checks if the failure is of major type (6). If the

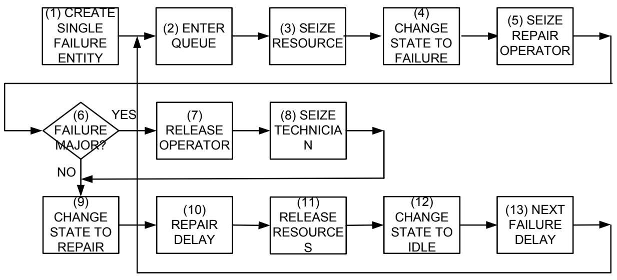


Figure 2: Failure Logic

failure is major, the operator is released and a technician is requested (8). When either the operator or the technician starts fixing the machine, the state of the machine is changed to repair (9) and there is a delay (10) accounting for the repair time. Once the repair work is finished, the operator/technician and the machine resource are released (11) and their states are changed to idle (12).

The typical machines we come across in electronics manufacturing include screen printers, placement machines, soldering ovens, automated inspection machines, and conveyors – corresponding to the general process described above. Section 4.3 provides more detailed descriptions of the machine types. The manufacturing lines generally consist of some or all of these machines connected in some fashion by conveyors. Each of these machines is similar to the typical machine described above, but with different processing times, failure rates and exhaust rates. A module developed for this machine would encapsulate all this logic in a subsystem. Thus the modeling efforts would be considerably reduced by the development of a template with reusable modules. The following section describes such a template.

4 TEMPLATE DEVELOPMENT

4.1 Inputs

This section discusses in detail the inputs used in the design of modules, categorized based on their common characteristics.

4.1.1 Conveyor

The inputs related to conveyor are addressed below:

i) Capacity of the conveyor: The number of boards that can simultaneously travel on the conveyor segment.
ii) Length: The length of the conveyor.
iii) Velocity: The constant velocity of the conveyor.
iv) Entry point: The conveyor has two end points. The entry point is the point from where the boards enter the conveyor.
v) Exit point: The exit point is the end point on the conveyor from where the boards leave the conveyor.
vi) Preceding conveyor: The name of the conveyor from which the boards enter the module. This is required in order to release the conveyor length occupied by the board entering the module. The conveyor cannot be released until the resource is available for service.
vii) Succeeding conveyor: The name of the conveyor, which moves the board from the current module to the following module. This is required in order to occupy sufficient space on the conveyor before releasing the resource. If sufficient space is not available on the conveyor, the resource is blocked and is not released for serving the next board.

4.1.2 Resource

The inputs related to resource are addressed below:

i) Resource: The resource representing the machine in the module.
ii) Capacity of the resource: The number of machines available for this module. Default value is one. If the capacity is more than one, the boards may be processed in batches.
iii) Indexing time: Indexing involves setup of the board before the actual processing is started. It is provided as an option and is in the form of a statistical distribution or a constant value.
iv) Processing time: It is the length of time for which the board is processed. It has the form of either a statistical distribution or a constant value.

4.1.3 Failures

The inputs related to failures are discussed below:

i) Failures: It is provided as an option to the user. Failures can be in two forms - time dependent failures (failures that occur at regular time intervals) or board dependent failures (failures that occur after every specified number of boards). The value can be in the form of a statistical distribution or a constant value.
ii) Resource for repairs: This is valid only if failures exist. This is the resource which fixes the machine when it is in failure state and is referred to as an operator.
iii) Repairs: This is valid only if failures exist. This is the time taken for the operator for checking the failure and fixing the machine. It is also in the form of a statistical distribution or a constant value.
iv) Probability of major failures in total failures: Provided as an option, major failures are the failures which cannot be fixed by the operator and a tech-

nician is required. These are different from the normal failures and may require a different repair resource, which we call as technician, with a new repair time distribution.

v) Resource for major repairs: This is valid only if major failures exist. This is the resource which fixes the machine when it has a major failure and is referred to as a technician.

vi) Major repairs: This is valid only if major failures exist. This is the time taken for the technician for checking the failure and fixing the machine. It is also in the form of a statistical distribution or a constant value.

4.1.4 Exhausts

The inputs related to exhausts are discussed below:

i) Exhausts: The discussion is similar to that of failures, but there are no major exhausts analagous to major failures. As exhausts may not be applicable for all the machines (e.g., inspection processes), it should be provided as an option.

ii) Resource for replenishment: It is similar to resource for repairing. This is valid only if exhausts exist.

iii) Replenishments: Valid only if exhausts exist. It is similar to repair time distribution.

4.1.5 Miscellaneous

The other inputs generally required to model a line are as follows:

i) Batch size: The boards that move together form a batch. The number of boards in a batch is the batch size.

ii) Capacity of the operator: The number of operators available for the production line. The default value is one.

iii) Capacity of the technician: The number of technicians available for the line. Default value is one. Valid only if major failures exist.

4.2 Outputs

The outputs we are interested in collecting are as follows:

i) The utilization of the machine, which is the percentage of time spent by the machine in busy state out of the total running time.

ii) The percentage of time spent by the machine in each of the other defined states.

iii) Time in module, which is the total time spent by an entity from its entry to exit in the module.

iv) The average number of boards on each of the conveyors defined.

v) The utilization of the operator, which is the percentage of time spent by the operator in the busy state out of the total time.

vi) The utilization of the technician, which is the percentage of time spent by the technician in the busy state out of the total time. This is valid only if major failures exist and they are fixed by a technician different from an operator.

All these outputs have to be obtained for each replication as well as over all the replications for which the model is run.

4.3 Modules

The following are the machines which we frequently come across in the electronics manufacturing industry:

i) *Board Destacker*: A device that removes boards from a stack and feeds them onto the line one at a time.

ii) *Screen Printer*: It is the machine where the attachment media in the form of solder paste or adhesive is applied on the board. This module is based on the module designed in Section 4.4.1 with both failures and exhausts. The machine suffers from exhausts when it runs out of solder paste or adhesive. The processing time is the time taken for solder paste to be applied on the board.

iii) *Placement*: Here the components are placed on the machine. This module is based on the module designed in Section 4.4.1 with both failures and exhausts. The machine suffers from exhausts when it runs out of the components. The processing time is the time taken for the components to be placed on the board.

iv) *Board Inverter*: If the components are to be placed on both sides of the board, the board has to be inverted after the components are placed on one side. This is the machine where it is done. This module is based on the module designed in Section 4.4.1, with only failures.

v) *Curing machine*: Machine where the attachment media is cured. It is either in the form of a machine or conveyor.

vi) *Soldering machine*: Machine with a conveyor where either reflow soldering or wave soldering is done.

vii) Cleaner: Machine where the boards are cleaned by passing them through a conveyor or a machine.

viii) *Tester*: Here the boards undergo a testing procedure. The processing time is the time taken for the testing process of a single board. The boards that fail the test are checked by the operator before they are either discarded or sent for rework.

ix) Board *stacker*: Place where the boards coming out of the line are stacked together and packed.

x) *Conveyor*: The transportation of boards between the machines is done by conveyors, which move at a constant speed. This is based on the module designed in Section 4.4.2.

4.4 Modeling of Modules

This section discusses the modeling of the following machines:

1. Machine with(out) failures and/or with(out) exhausts.
2. Conveyor with(out) failures.

4.4.1 Machine with Failures and Exhausts

The modeling involves mainly three aspects – data input, collection of output and development of logic. The input data required for this module is as follows:

i) Name
ii) Preceding conveyor
iii) Succeeding conveyor
iv) Resource
v) Capacity of the resource
vi) Indexing time
vii) Processing time
viii) Failures
ix) Repairs
x) Resource for repairs
xi) Probability that a failure will be a major failure (potentially requiring a different repair resource)
xii) Major Repairs
xiii) Resource for major repairs
xiv) Exhausts
xv) Replenishments
xvi) Resource for replenishments

The outputs to be collected for this module are as follows:

i) The utilization.
ii) The percentage of time spent by the machine in each of the other defined states.
iii) Time in module.

The logic for this machine involves developing of processing logic, failure logic and exhaust logic. The logic is same as discussed before in Section 3.3. The only important aspect is the proper transfer of data entered in input dialog into the logic blocks. The seize, delay and release blocks in the logic use the resource entered in the input dialog. All the other data like the processing time, indexing time, etc has to be referred in the proper blocks. Output statistics like the utilizations and the percentage of time spent by the resource in each of its states have to be collected appropriately.

4.4.2 Conveyor with Failures

This section discusses the design of a conveyor module with failures. The inputs required are as follows:

i) Name
ii) Entry point
iii) Exit point
iv) Velocity
v) Length
vi) Capacity
vii) Failures
viii) Repairs
ix) Resource for repairs
x) Probability of major failures in total failures
xi) Major Repairs
xii) Resource for major repairs

The outputs to be collected for this module are as follows:

i) The average number of entities on the conveyor.
ii) The percentage of time spent by the conveyor in each of the defined states.

The number of boards that can travel simultaneously on a conveyor is its capacity. When a conveyor fails, the boards moving on it are stopped until the problem is fixed by an operator. The design of a conveyor in terms of resources and queues is described here. The conveyor flow logic is shown in Figure 3. The conveyor is designed in the form of a resource with the capacity equal to the capacity of the conveyor. The length of the conveyor is split into equal lengths of length/capacity. The boards enter a queue (1) and seize (2) a unit of resource. The boards undergo appropriate delay (3) before releasing the resource (4). Here the assumptions are: i) the delay (3) is less than the interarrival time of boards onto the conveyor so that the boards maintain minimum distance between each other and ii) there is an infinite queue at the end onto which the boards are released from the conveyor.

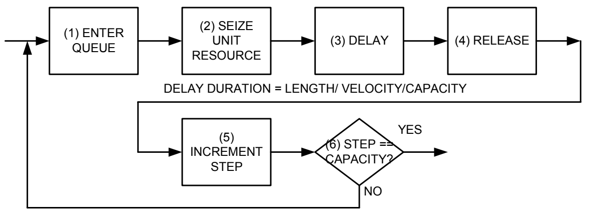


Figure 3: Conveyor Flow Logic

The failure logic of the conveyor is shown in Figure 4. The failure entities are created (1) as a batch with size equal to the capacity of the resource. Each entity enters a queue (2) whose priority is higher than that in the flow logic and seizes (3) a single unit of resource. The entities are again batched (4) so that the full conveyor fails at a time. The remaining logic is similar to that of a resource.

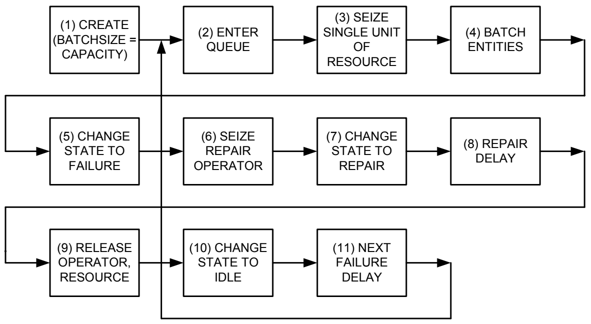

(1) CREATE (BATCHSIZE = CAPACITY)
(2) ENTER QUEUE
(3) SEIZE SINGLE UNIT OF RESOURCE
(4) BATCH ENTITIES
(5) CHANGE STATE TO FAILURE
(6) SEIZE REPAIR OPERATOR
(7) CHANGE STATE TO REPAIR
(8) REPAIR DELAY
(9) RELEASE OPERATOR, RESOURCE
(10) CHANGE STATE TO IDLE
(11) NEXT FAILURE DELAY

Figure 4: Conveyor Failure Logic

4.5 IMPLEMENTATION

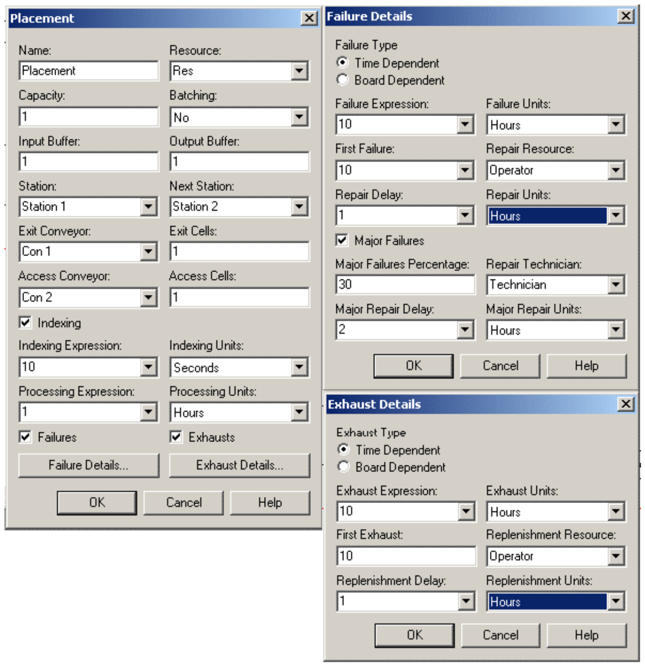
The implementation of the modules discussed above is done in the simulation tool Arena, which is based on SIMAN simulation language. The module development in Arena involves the development of a template, which is a collection of modules. A module development process in Arena involves the following five steps (Rockwell Software 1999):

1. *Input dialog*: The input dialog is the dialog shown for data to be entered into the module. The input dialog should display all the input items designed for a module.
2. *Logic*: It consists of the simulation logic of the module. The data required by the blocks is obtained by referencing the data entered into the input dialog.
3. *Switches*: Switches help in defining variations of input dialog, module logic, etc. based on certain conditions.
4. *Panel icon*: The icon for the module as seen in the template panel.
5. *User view*: It is the display of the module as seen when an instance of the module is placed in the model window.

The development of modules for placement and conveyor machines is discussed below. Only the input dialog and logic are discussed here.

4.5.1 Placement Machine

The placement machine is a machine with failures and exhausts. The logic, the input data and the output statistics are similar to those for the machine discussed in Section 4.4.1. The input dialog for the placement module is shown in Figure 5. Indexing, failures and exhausts are provided as options with the help of checkboxes. Secondary dialog boxes are provided for failure details and exhaust details. The data entered into the input dialog is referenced by the Arena blocks used in the logic. The processing logic is displayed in Figure 6. The output statistics like the utilization and percentage of time spent in different states have to be collected for the resource.


Placement

Name: Placement
Resource: Res
Capacity: 1
Batching: No
Input Buffer: 1
Output Buffer: 1
Station: Station 1
Next Station: Station 2
Exit Conveyor: Con 1
Exit Cells: 1
Access Conveyor: Con 2
Access Cells: 1
☑ Indexing
Indexing Expression: 10
Indexing Units: Seconds
Processing Expression: 1
Processing Units: Hours
☑ Failures
☑ Exhausts

Failure Details

Failure Type
[✓ Time Dependent] Board Dependent
Failure Expression: 10
Failure Units: Hours
First Failure: 10
Repair Resource: Operator
Repair Delay: 1
Repair Units: Hours
☑ Major Failures
Major Failures Percentage: 30
Repair Technician: Technician
Major Repair Delay: 2
Major Repair Units: Hours

Exhaust Details

Exhaust Type
[✓ Time Dependent] Board Dependent
Exhaust Expression: 10
Exhaust Units: Hours
First Exhaust: 10
Replenishment Resource: Operator
Replenishment Delay: 1
Replenishment Units: Hours

Figure 5: Placement Module Input Dialog

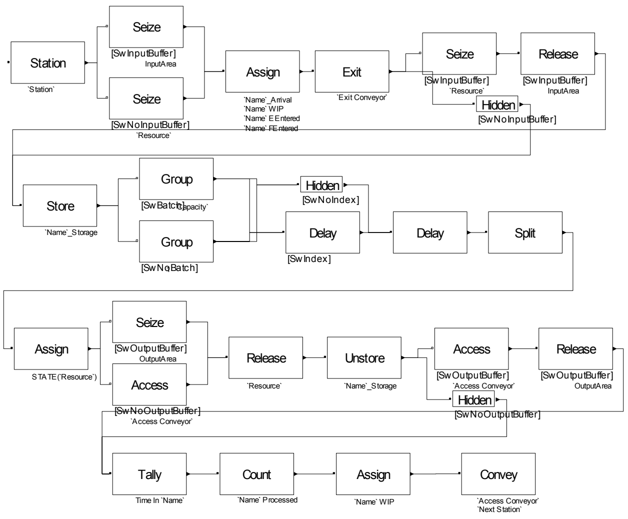


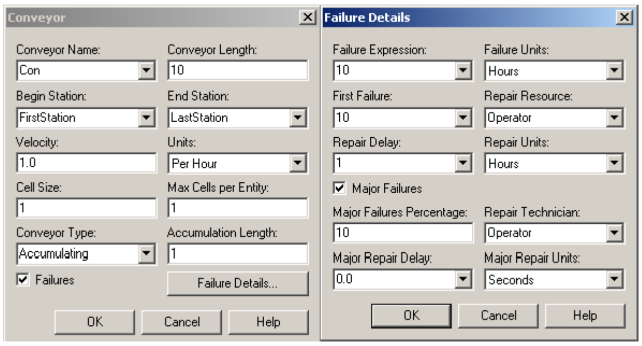
Figure 6: Placement Module Processing Logic

4.5.2 Conveyor

The conveyor module is modeled based on the conveyor data module provided by Arena. The input dialog for the conveyor is shown in Figure 7. The logic development is quite simple and involves referencing the data items in the input dialog from the conveyor data module.

4.5.3 Additional Details

In addition to the definition of modules, the following things should be done before doing the simulation. The



Conveyor

Conveyor Name: Con	Conveyor Length: 10
Begin Station: FirstStation	End Station: LastStation
Velocity: 1.0	Units: Per Hour
Cell Size: 1	Max Cells per Entity: 1
Conveyor Type: Accumulating	Accumulation Length: 1
✓ Failures	

Failure Details

Failure Expression: 10	Failure Units: Hours
First Failure: 10	Repair Resource: Operator
Repair Delay: 1	Repair Units: Hours
✓ Major Failures	
Major Failures Percentage: 10	Repair Technician: Operator
Major Repair Delay: 0.0	Major Repair Units: Seconds

Figure 7: Conveyor Module Input Dialog

capacities of the operator(s) and technician(s) defined have to be set to their actual capacities. This is done from the *Resource* data dialog of *Basic Process* template panel. The output statistics like the utilization and percentage of time spent in different states have to be collected for these resources. This is done from the *Statistic* data dialog of *Advanced Process* template panel.

5 CASE STUDY

This section discusses the project involving the analysis of a PCB manufacturing facility which motivated us towards the development of generic modules. The manufacturing process consisted of the following machines: i) screen printer ii) inspection iii) placement iv) reflow soldering oven v) coolers and vi) auto visual inspection (AVI). The boards undergo testing at the inspection machines. Coolers are conveyors fitted with fans at the top and they bring down the temperature of the boards as they come out of the soldering oven. All the machines are interconnected by limited capacity conveyors. A single operator oversees the operation of the system and is responsible for the repair and replenishment of the machines. The objective of the project was to determine the capacity of the system and to suggest steps for potential improvements.

The first simulation model was developed using the templates provided in Arena. Processing, failure and exhaust logics were developed for screen printer and placement machines, while only processing and failure logics were done for inspection and AVI. Reflow oven and coolers were modeled as conveyors. Data was collected from the production facility about the cycle times, defect probabilities for inspection machines, failure and repair rates, exhaust and replenishment rates of the machines. To analyze the system, the following outputs were collected: a) utilization of the resources b) the percentage of time spent by a resource in each of its states c) the time spent in system by a board d) WIP in the system and e) the capacity of the system.

A second simulation model has been developed using the modules. Screen printer, placement, soldering oven and inspection modules are selected to represent their respective machines. Appropriate number of conveyor modules are selected to represent conveyors. The modules are filled with the relevant data collected before. These steps complete the model development. The model is then run and the results are analyzed.

It can be concluded that the model development for PCB assembly process is simplified significantly by the use of templates. The processing, failure and exhaust logics are hidden inside the modules, requiring us to just understand the process and fill the data appropriately.

6 CONCLUSIONS

The modeling of machines for electronics manufacturing involves significant effort due to the existence of failures, part exhausts and their dependence on operators and other resources. The complexity involved in modeling a typical machine has been described in this paper. The design and development of the placement module (with failures and exhausts), and the conveyor module (with failures) has been discussed in detail. These reusable modules encapsulate the modeling logic and significantly reduce the efforts required in future for modeling a system with these machines.

REFERENCES

AbouRizk, S. M., J. Y. Ruwanpura, K.C. Er, and Siri Fernando. 1999. Special purpose simulation template for utility tunnel construction. In *Proceedings of the 1999 Winter Simulation Conference*, ed. P. Farrington, H. Nembhard, D. Sturrock, and G. Evans, 948-955. Piscataway, N.J. : Institute of Electrical and Electronics Engineers.

Anglani, A., A. Grieco, M. Pacella and T. Tolio. 2002. Object-oriented modeling and simulation of flexible manufacturing systems: a rule-based procedure. *Simulation Modeling Practice and Theory* 10:209-234.

Farrington, Phillip A., John S. Rogers, James J. Swain, John L. Evans. 1996. Developing reusable modeling capabilities for simulating high volume electronics manufacturing systems. *IEEE Transactions on Components, Packaging, and Manufacturing Technology, Part C* 19:89-97.

Ho, T. K., B. H. Mao, Z. Z. Yuan, H. D. Liu and Y. F. Fung. 2002. Computer simulation and modeling in railway applications. *Computer Physics Communications* 143:1-10.

Hollomon, Jr., James K. 1989. *Surface mount technology for PC board design*. Indianapolis, Indiana: Howard W. Sams & Company.

King, Russell E. and Kyung Sup Kim. 1995. AgvTalk: An object-oriented simulator for AGV systems. *Computers and Industrial Engineering* 28:575-592.

Kiran, Ali S., Celal Kaplan and A. Tamer Unal. 1993. Simulation of electronics manufacturing and assembly operations. In *Proceedings of the 1993 Winter Simulation Conference*, ed. G. W. Evans, M. Mollaghasemi, E. C. Russell, W.E. Biles, 773-779. Piscataway, N.J. : Institute of Electrical and Electronics Engineers.

Ozdemirel, N. E. and G. T. Mackulak. 1993. A generic simulation module architecture based on clustering group technology model codings. *Simulation* 60:421-433.

Prasad, K. 1990. A generic computer simulation model to characterize photolithography manufacturing area in an IC FAB facility. IEEE/CHMT Eighth International Electronic Manufacturing Technology Symposium, p 66-72. Piscataway, N.J. : IEEE Service Center.

Paul, Ray J. and Jasna Kuljis. 1995. A generic simulation package for organising outpatient clinics. In *Proceedings of the 1995 Winter Simulation Conference*, ed. C. Alexopoulos, K. Kang, W. R. Lilegdon, and D. Goldsman, 1043-1047. Piscataway, N.J. : Institute of Electrical and Electronics Engineers.

Paul, Ray J. and Taylor, Simon J. E. 2002. What use is model reuse: Is there a crook at the end of the rainbow. In *Proceedings of the 2002 Winter Simulation Conference*, ed. *E. Yücesan, C.-H. Chen, J. L. Snowdon, and J. M. Charnes*, 648-652. Piscataway, N.J. : Institute of Electrical and Electronics Engineers.

Rockwell Software. 1999. *ARENA Professional Edition Reference Guide*. Rockwell Software, 504 Beaver Street, Sewickley, PA 15143.

Son, Y. J., Albert T. Jones, and Richard A. Wysk. 2000. Automatic generation of simulation models from neutral libraries. In *Proceedings of the 2000 Winter Simulation Conference*, ed. J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, 1558-1567. Piscataway, N.J. : Institute of Electrical and Electronics Engineers.

Thesen, Arne. 1990. Template based simulators: An example from manufacturing. In *Proceedings of the 1990 Winter Simulation Conference*, ed. Osman Balci, Randall P. Sadowski, Richard E. Nanee, 547-550. Piscataway, N.J. : Institute of Electrical and Electronics Engineers.

Valentin, Edwin and Verbraeck, Alexander. 2002a. Simulation using building blocks. In *Proceedings of AI, Simulation and Planning in High Autonomy Systems*, ed. F.J.Barros, N.Giambiasi, 65-71. Available online <http://www.tbm.tudelft.nl/webstaf/edwinv/articles/AIS2002.SimulationBuildingBlocks.pdf> [accessed January 15, 2003].

Valentin, Edwin and Verbraeck, Alexander. 2002b. Guidelines for designing simulation building blocks. In *Proceedings of the 2002 Winter Simulation Conference*, ed. *E. Yücesan, C.-H. Chen, J. L. Snowdon, and J. M. Charnes, 563*-571. Piscataway, N.J. : Institute of Electrical and Electronics Engineers.

AUTHOR BIOGRAPHIES

PHANI S. MUKKAMALA is an M.S. student in the Department of Industrial & Systems Engineering at Auburn University. He received his B.S. degree in Civil Engineering from Indian Institute of Technology, Madras, India, in 1999. His current research interests are in discrete event simulation and modeling for complex systems. His email address is <mukkaps@eng.auburn.edu>.

JEFFREY S. SMITH joined Auburn University as an associate professor in the Industrial & Systems Engineering department in September of 1999. Prior to joining Auburn, Dr. Smith was an associate professor in the Industrial Engineering Department at Texas A&M University. He received the B.S. in Industrial Engineering from Auburn University in 1986 and the M.S. and Ph.D. degrees in Industrial Engineering from Penn State University in 1990 and 1992, respectively. In addition to his academic positions, Dr. Smith has held industrial positions at Electronic Data Systems and Philip Morris U.S.A. Dr. Smith is an active member of IIE, SME, and INFORMS. His email and web addresses are <jsmith@auburn.edu> and <http://sim.auburn.edu/~jsmith>.

JORGE F. VALENZUELA is currently an assistant professor in the Industrial & Systems Engineering department at Auburn University. Dr. Valenzuela received his B.S. in Electrical Engineering from Northern Catholic University in Chile in 1984. He obtained an M.S. degree in Statistics at CIENES and an M.S. degree in Industrial Engineering at Northern Illinois University in 1985 and 1996, respectively. In 2000, he received the Ph.D. degree in Industrial Engineering from University of Pittsburgh. Dr. Valenzuela is a member of IIE and INFORMS. His email address is <jvalenz@eng.auburn.edu>.