B. J. Dooley

A Model for the Prediction of Assembly, Rework, and Test Yields

The increase in density and complexity of computer components used as field-replaceable units of the IBM 3081 processors has required more sophisticated and capital-intense manufacturing and test lines than heretofore seen. To help in the manufacturing planning effort, a simulation model based on the concepts described in this paper was implemented as a means for studying the behavior of different assembly/test methodologies and to predict the throughput capabilities of various manufacturing configurations. By adopting a different simulation philosophy, the model was greatly simplified, and by taking advantage of the matrix processing features of APL, modeling changes required due to procedure or test-equipment changes could easily be implemented. This paper reviews the philosophy of the simulation of computer part assembly, takes a new look at modeling, and describes the architectural concepts embodied in the implementing program. It also details some of the more important concepts needed to complete the simulator.

Introduction

The modules, called TCMs, used in the new generation of IBM computers represented by the IBM 3081 processor models are complex, hybrid assemblies of high-density integrated circuit chips on multi-layer ceramic substrates that normally contain one hundred chips and match the circuit count and logical power of a typical central processing unit of a large system of mid-1970s vintage [1]. The circuit boards into which up to nine of these TCMs can be plugged, called TCM boards, contain many layers of buried circuitry plus module sockets which are characterized by close tolerances and stringent alignment requirements [2]. The density and complexity of these assemblies have offered unique challenges for manufacturing planning. Despite low and well-controlled defect densities for the chips, substrates, boards, and other components, the assembled modules and circuit boards could contain at least one defect when first assembled. The rework of these assemblies could, in turn, introduce further defects.

As part of the packaging design, practically all defects which can affect system operation are both testable and repairable, so that scrapping of assemblies is seldom necessary. The manufacturing lines, therefore, were planned to contain re-entrant test/rework loops, driven by the defect densities present in the incoming parts and the defect densities introduced by testing, assembly, and handling. Since the process tools needed for assembly, testing, and rework are very much more sophisticated and expensive than in previous technologies, the need for a simulation model to plan the right number and mix of processing areas (sectors) for a targeted manufacturing throughput was readily apparent. This model could be directed to the prediction of gross workloads at all main sectors in the assembly line, in terms of assemblies processed, components placed, and tests performed.

The design of such a model is not as straightforward as it would seem. Each sector in the process line would require a complex probability function which could be changed in response to the particular process sequence in use for each product type at each period in calendar time. This would necessitate an exhaustive treatment of the probability of the assembly being routed from one manufacturing sector to another. Thus, each change in the process, however small, which had not been planned for in the creation of the function, could alter the conditions in the model beyond the

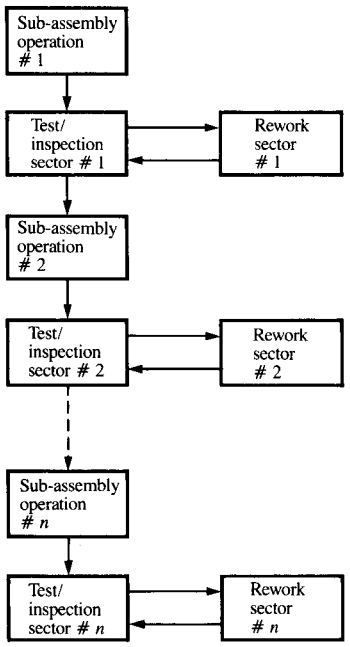

Figure 1 Conventional manufacturing process flow model.

intent of its design and lead to incorrect conclusions. For these reasons, we decided to re-examine the fundamental philosophy of modeling.

This paper reviews the philosophy of the simulation of computer part assembly. It takes a new approach for the simulation process and describes the architectural concepts embodied in the implementing program. It also details some of the more important concepts needed to complete the simulator. The new model, called PARTY (prediction of assembly, rework, and test yields), is written in APL and has been used as a basis for predicting significant aspects of workload and tool requirements for manufacture of modules and boards used in the IBM 3081 processor line. Data derived from the model are used by several IBM manufacturing locations as the basis for assembly line planning.

Approaches to assembly simulation

It is customary to visualize a manufacturing process in terms of a flow of parts, some of which are defective (Fig. 1). This concept works very well, provided that one has only a general interest in the nature of the defects and little interest in their combinational aspects. This is usually the case with fairly simple assemblies.

Modern electronic assemblies with a dense heterogeneous population of components mounted upon a multi-layer substrate are not simple assemblies. Even though the density of any selected type of defect may be extremely low, the potential range of defect types is so wide that there is a high probability that many assemblies may contain at least one defect when first built; many could contain more than one. Rework of parts to correct observed defects can often introduce new defects, not necessarily of the same type.

If one possesses detailed information as to the types and densities of defects which can be expected, and the tester coverages associated with these defect types, one can fairly readily compute an estimate of the yield to be expected at the first test pass after initial assembly. The yield which is to be anticipated at the second test pass is not so readily predicted, since some defects which were masked by other defects may now have become detectable. New defects may have been added in the course of the rework, some rework may have been unsuccessful, and some errors may have been made in the first test and/or rework operations. In addition, the isolation of defects in the course of diagnostic testing may have necessitated removal of components. Restoration of such assemblies to their original design may have been incomplete or may have introduced new problems. From this discussion it is readily seen that the difficulties in prediction become progressively worse for subsequent test passes.

Yield, of course, is not the only item affected. Calculation of the workload to be expected at each of the rework sectors becomes extremely difficult. The quantities and types of repair actions and their combinational aspects cannot be predicted without some form of simulation. In the absence of such a prediction, construction of adequate and appropriate tooling plans for the manufacture and testing of the product becomes a matter of chance.

The form of simulation to be used must be chosen carefully. For example, it may not even be appropriate to use an *average* assembly as the basis for prediction since there may be too great a difference between the assembly types, in terms of the combinations of defects (and therefore of rework actions) which can be expected to occur during manufacture. Simulation must then be applied on an individualized type-by-type basis.

Application of a Monte Carlo approach would clearly be inappropriate in such a case, since the amount of time needed for simulation of the manufacture of many different assembly types would become prohibitive. This is especially true when many of the processes display improvement over time. In such cases, many separate simulations are required for each assembly type to represent the changes occurring in successive vintages of product. In addition, the range of possible defect types is large, with different defect densities and tester effectiveness for each defect type.

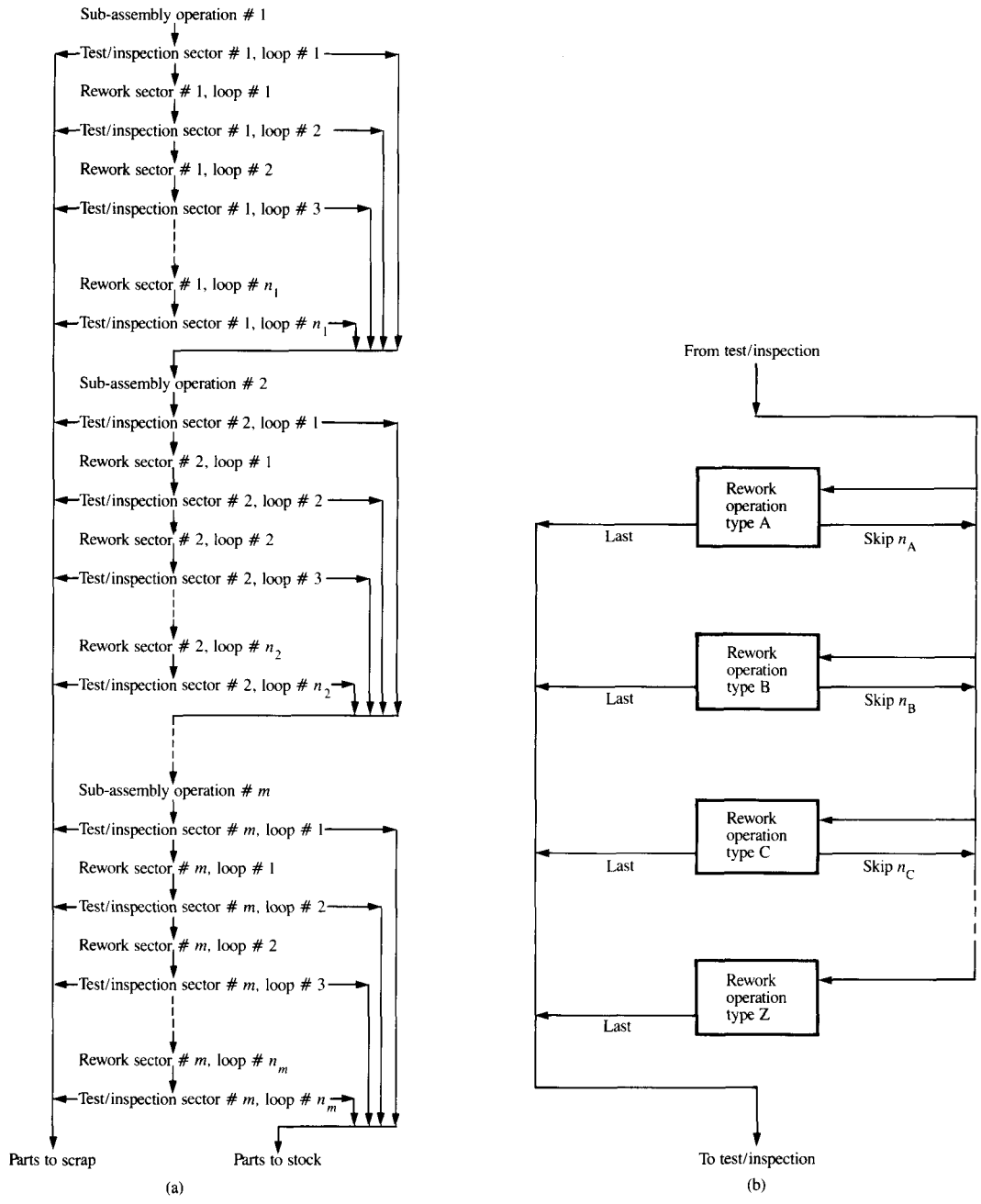

Figure 2 Modified manufacturing process flow model showing the test/rework loops unraveled. (b) Model of an individual assembly/rework sector showing the sequence of assembly/rework steps.

The numbing complexity of the problem just presented is derived, in part, from the concept used in its statement. That concept is the conventional visualization of the manufacturing line in terms of a flow of assemblies, some of which are defective. But the potential quantities and combinations of varieties of defects on those assemblies have rendered this concept of little value in predicting the most probable outcome.

The assembly line can be depicted in quite a different fashion by visualizing the loops in the manufacturing line as having been unrolled, so that the sectors in any given loop are replicated successively as many times as necessary to accomplish all the work required (Fig. 2). Any assembly which is free of defects when it reaches a test stage is allowed to bypass all the remaining sectors which belong to the loop, and to re-enter the assembly line at the first sector beyond

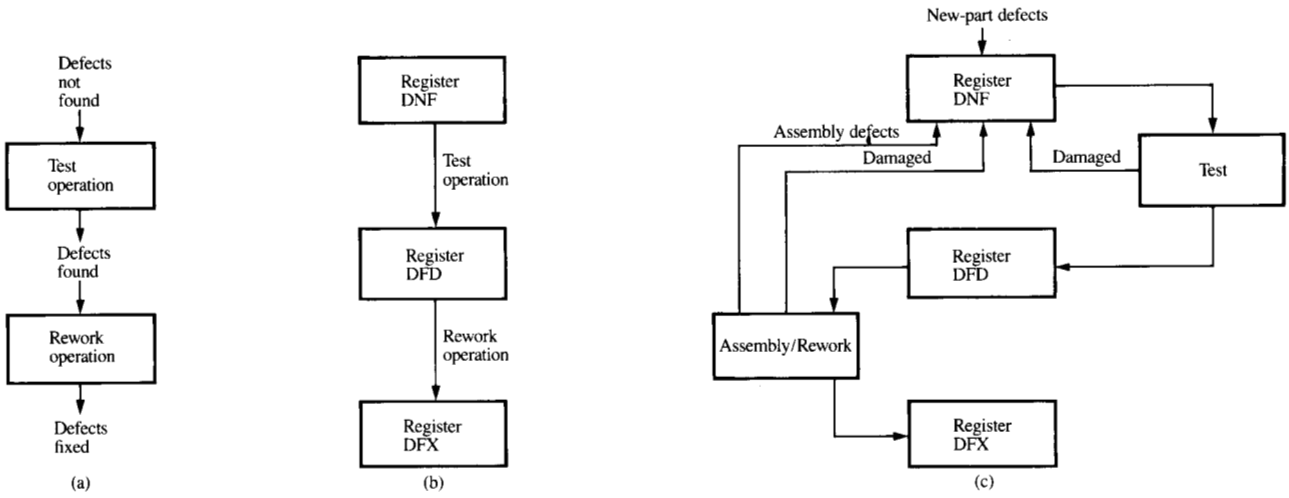

Figure 3 (a) Conceptual flow of manufacturing defects. (b) Simplified data flow of PARTY model. All defects entering the manufacturing line are recorded in register DNF (defects not found) which contains the list of defects which have not yet been discovered by inspection or testing. When a defect is discovered, its record is transferred to register DFD (defects found) which records the defects found but not yet remedied. When a defect is repaired its record is transferred to register DFX (defects fixed), which retains a list of all repair actions which have been performed. (c) The same model modified to show the effects of adding test and assembly/rework sectors.

the limit of the unrolled loop. Assemblies are also allowed to bypass all rework sectors for whose services they have no need. When viewed in this fashion, the line is seen to be composed of a continuous linear succession of assembly and testing sectors, bypassed by a mass of shunt paths.

Viewed from this standpoint, the flow of assemblies through the manufacturing line may still appear chaotically complex since all that has been achieved is the substitution of a multitude of parallel paths for the previously re-entrant loops. It will remain chaotic as long as the *assembly* is considered to be the unit by which flow through the paths is measured. If one chooses instead to have the detectable *defects* become the unit of measure, a dramatic change is observed: the majority of the shunt paths are now seen to be empty, and the entire traffic passes through every test sector in the loop. Knowing the quantity of assemblies which enter the first sector of the unrolled loop, one can apply simple probability methods to determine the flow through each succeeding sector. If the defects can be treated separately by type, their flows also can be predicted separately and superimposed on one another. This merged flow can in turn be superimposed upon the flow of defective assemblies which has been predicted from the defect population.

The benefits of this approach are large. Since the existence and point of origin of the various types of defects are individually determined, it becomes a relatively trivial probability exercise to derive, for example, an estimate of how many of those assemblies which failed at the first stage would undergo a given combination of rework actions before re-entering testing. A totally deterministic approach becomes possible, in which all assemblies can be considered as having entered the manufacturing line simultaneously. The calculations for the various defect types can be performed in parallel by array processing. This enables a large range of defect types to be considered separately, with all aspects of association and relative significance being properly addressed.

Defect processor model concepts

This then is the central concept in PARTY: An assembly line can be modeled as a *defect* processor rather than as an *assembly* processor. Defects flow into the line, borne by the substrates, boards, or other components. To these are added defects induced during assembly and testing. Each time that defects are subtracted by rework of those assemblies recognized as defective, there may be further defects added with the replacement parts.

Thus, there is a "fund" of defects residing in the manufacturing line which is waiting to be discovered and which is vulnerable only to the testing process. There is also a fund of defects residing in the line comprised of defects waiting for remedial action and/or replacement, which is vulnerable only to the rework process. In addition, there is a record of defects which have already been both discovered and remedied.

In Fig. 3(a) these three groups are designated as "defects not found," "defects found," and "defects fixed," respectively, to demonstrate the conceptual flow. These three groups are represented in the PARTY model by registers DNF, DFD, and DFX, as shown in Fig. 3(b). The flow between the groups is then simply a matter of appropriate transfers of records between the registers.

All defects entering the manufacturing line are recorded in register DNF, which thus contains the list of defects which have not yet been discovered by inspection or testing. Any defects introduced by the rework and test operations fall, of course, into this category. When a defect is discovered, its record is transferred to register DFD, which thus contains the list of defects which have been found but which have not yet been remedied. When a defect is repaired, its record is transferred to register DFX, which thus contains a list of all the repair actions which have been performed.

Adding a typical test stage and a typical assembly/rework stage to this simple diagram makes clear the principal flow paths of records within the simulation model [Fig. 3(c)]. When a test stage is simulated, the DNF register provides a list of the defects which should be detectable at that tester. The records representing these defects are then removed from DNF and placed in DFD. When a rework stage is simulated, the DFD register provides a list of defects which should be addressed by that type of rework. These records are removed from DFD and placed in DFX. All records generated to represent additional defects stemming from processing or damage are, of course, placed in DNF. Incorporation of these flow lines into the diagram of the model leaves it still fairly simple.

As is immediately evident, register DNF is the hub of the model. It is in this register that the distinctions are made between the menus of defects which are detectable at the various types of tester available. It must also maintain the up-to-date status of all types of defects, in terms of the quantities of defects present in each process. Since defects which are detectable at one type of tester may be undetectable at another, a full accounting must be kept of all defects, detectable or not, in the knowledge that any defects still present in this register at the end of a simulation are going out undetected into the next higher level of assembly.

Register structure

At this point it is appropriate to give a general account of the register structure of the PARTY model. There are eight main registers, of which one is for input, four are for process simulation, and three are for output. For the sake of clarity, the process-simulation registers are addressed first.

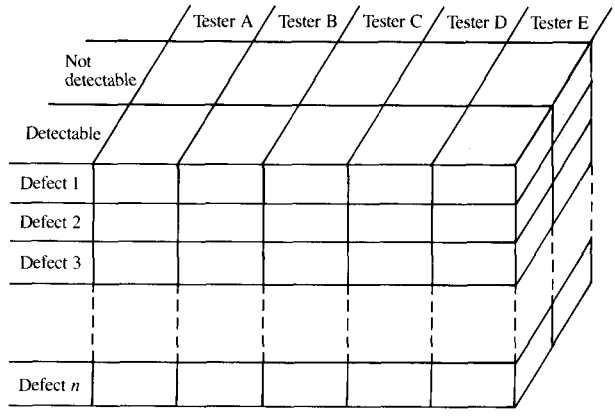

Figure 4 Structure of the DNF register for the case where there are five tester types, designated testers A through E. Each horizontal slab in the diagram represents a variety of defect. Thus, for a given defect, if there are d such defects present, the totals for the various testers will be the same, but the way in which the d divides between the detectable and the non-detectable categories will be a function of the coverage afforded by each of the tester types for the particular defect mode involved.

- *Process-simulation registers*

In addition to registers DNF and DFD, registers MISSES and AVAILABLE are defined in this category. Register MISSES is required as a repository for those records of defects which have escaped detection because the postulated testers are not 100% effective. These defect escapes reside upon assemblies which have been routed into a rework stage. Their records are presented, along with those from DNF, when these assemblies next reach a test sector. Register AVAILABLE does not relate to defects. It is used instead to control the quantities of assemblies which enter the various process sector types. A major part of the housekeeping procedures in PARTY relates to the updating of register AVAILABLE.

The structure of register DNF is fundamental to the action of the model. This is the register in which are maintained the counts of defects which have not yet been detected by a tester. But there may be many types of defects and many types of testers, and some testers are not able to detect some of the defects. In order to accommodate these differences between testers, the DNF register is constructed as a three-dimensional array, $t \times d \times 2$, with one column for each type of tester t (see Fig. 4), one row for each type of defect d, and two planes representing the tester-detectable and the tester-undetectable defects, respectively.

Registers DFD and MISSES are constructed just like the first plane of register DNF; they are shown in Fig. 5. When

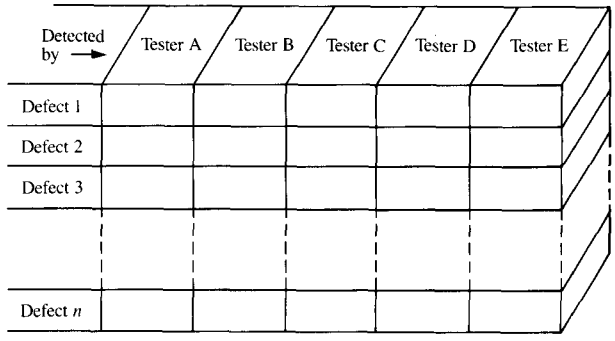

Figure 5 The DFD and MISSES registers are identical in structure to the DNF register, except that they contain only one plane.

defect counts are transferred into DFD from DNF, they move from their location in the first plane of DNF to the identical position in DFD. This register, therefore, is two-dimensional, with t columns to reflect the t types of tester, and as many rows as there are types of defect. The structure of MISSES facilitates the return of missed defects into DNF in the event that the next test stage encountered differs from the stage at which they were not detected.

It should be clear from the description of AVAILABLE, given earlier, that it is structured as a string of numbers representing quantities of assemblies which are available to enter various types of sectors for which the model has been set up. The indices of this register match the indices in a literal matrix named SECTORS in which are stored the names of the modular functions which simulate the respective sectors of the assembly process.

- *Input register*

The structure of the input register, called BASE, is identical to that of DNF. BASE is initialized at the start of each simulation to contain the defect densities for all defect types, as split between detectable and undetectable portions at the various testers. When a quantity of components is being added to the assemblies in the line, the appropriate slabs of BASE are multiplied by the incoming quantity to generate the associated defect menu. All other slabs of BASE are multiplied by zero. The three-dimensional matrix of defect quantities so produced is added into DNF. The BASE register, of course, is left unaltered throughout the simulation as source material for all calculations.

- *Output registers*

DFX is considered an output register, and so are two others called ESCAPES and TAB. The DFX register contains a record of the quantities of defects which were addressed by the rework sectors during the simulation. It is a numeric vector containing a position for every type of defect addressed by the model. It thus provides the capability for printing out a list of defects which were encountered, and also summaries by component type. In the course of each rework sector simulation, the quantities of affected defects are summed across register DFD and transferred into this register to provide a cumulative count by defect type.

The ESCAPES register is similar in structure to both DFD and MISSES. It is into this register that records are moved from MISSES to reflect totals that evaded detection at the testers, although test coverage does exist for the defect types concerned. Records placed in this register represent defects which should have been caught but which will go on to cause problems at later levels of assembly.

The TAB register is the sector-by-sector record of what occurred during the simulation. It is two-dimensional, containing one row for each process sector encountered. Each row summarizes the activity in its associated sector. It becomes apparent that the model concepts lend themselves to easy implementation in APL [3], and many APL features are readily embodied in the model. The versatility of PARTY, in terms of its ability to simulate any reasonable (i.e., convergent) process sequence, is achieved by a totally modular structure. Each type of process sector has an identifying reference number. At this number index, in the AVAILABLE register, are found the quantities of assemblies which should be routed to a sector of this type. At this same index, in the SECTORS register, is found the name of the modular program which provides a simulation of such a sector. The APL *execute* operator provides a simple and elegant means of executing the functions as needed, without the use of a large multiple-branch routine.

The parallel processing of data relating to many different types of defects, which is fundamental to the concepts of PARTY, is most conveniently achieved by matrix arithmetic. The size of the arrays which must be manipulated varies considerably, depending on the type of manufacturing sector which is to be simulated, because of the varying *menus* of defect types involved. Here again APL is an appropriate language, as it has great power in generalized array manipulation.

Basic architecture

The power of the PARTY model resides in its architecture rather than in its mathematics. The calculations are all simple and direct applications of standard probability manipulation methods, with nothing more sophisticated than the occasional use of a Poisson distribution. The programming architecture of the model is modular; i.e., each line-sector is represented by a program which, when executed,

finds all the inputs it needs stored as global variables within the workspace and updates these variables as appropriate during its execution.

The names of the modular programs are stored as lines of a literal matrix. The process sequence is stored as a numeric vector, in which each number is the index of the appropriate program name within the literal matrix. Changes in process sequence can thus be invoked by changing the input numeric vector which represents the process routing.

Each completed process step is represented by a set of six numbers (Fig. 6), of which the first identifies the nature of the activity in the process step and the remaining five indicate the quantities of assemblies entering, the quantities of assemblies leaving the step as good, the quantities of assemblies scrapped, the quantities of components added, and the quantities of removal or other non-addition activities performed. These six stored numbers enable the generation of other desired statistics as needed, such as percentage of the original starting assemblies which enter the step, tests or other activities performed per assembly within the step, and so on.

Usage	*Contents*
Sector type	3 (= Test)
Total assemblies in	1000
Good assemblies out	654
Scrapped assemblies out	3
Total components added	0
Total other activities	1000

Figure 6 Each completed process step consists of a set of six numbers, as shown. Note that the numbers shown correspond to the first test stage in the example illustrated in Figs. 7–9.

Data flow within the model

Apart from some special cases, discussed in another section of the paper, the flow within the model is as follows:

1. Determine how many assemblies are to be built, and how many components will be added to each in the assembly operation.
2. Using the defect densities projected for these new parts, determine the resulting types and quantities of defects which enter the process flow. Add these to register DNF.
3. Using the error rates and defect densities projected for the assembly process concerned, determine the types and quantities of defects introduced by the process. Add these also into register DNF.
4. Determine which type of tester is to be used (from the process sequence). Using the test coverage appropriate to this tester, compute the types and quantities of defects in register DNF which will be detected, and transfer these from register DNF into register DFD.
5. Using the error rates and defect densities projected for this test process, determine the types and quantities of defects introduced by testing. Add these into register DNF.
6. On the basis of the defects detected (vs the assembly quantity entering testing), determine the quantity of assemblies sent to be reworked.
7. Simulate the rework operation by transferring the records stored in register DFD into register DFX. However, many of these represent additional new components being installed as replacement parts on the assemblies, so the simulation re-enters at Step 2. It continues until all the assemblies which entered at Step 1 have been accounted for, either as having been shipped to stock or as having been scrapped.

The data created by PARTY yield, for each type of assembly, test, or rework activity, the quantities of assemblies which must be processed and the number and nature of the actions which must be performed during the processing. When coupled with the standard operational data for the processes involved (time needed to place an assembly into the fixture; process time per action performed, etc.), the PARTY data provide a full account of the total time needed at each step in the manufacturing process, with each step through the re-entrant test/rework loops accounted for separately.

Special cases

There are several special cases requiring treatment which deviates from the general flow shown in the previous section. The most interesting of them are the following:

- Masking of defects—cases in which the presence of one defect prevents the detection of another.
- Testing *over-kill*—cases in which one defect produces the symptoms of two or more defects, resulting in the unnecessary replacement of good parts.
- Test error—cases in which a defect is detected correctly but the wrong component is marked for replacement.
- Sequential rework—the development of wiring requirements to restore circuit function following the deletion of a substrate short.
- Secondary damage—inadvertent damage to chips which, while not involved in the actual rework operation, are closely adjacent to the site.

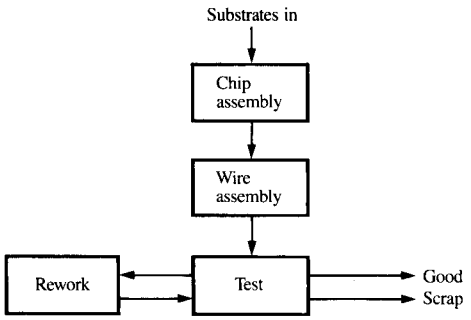

Figure 7 Simple manufacturing sequence for use in the simulated example.

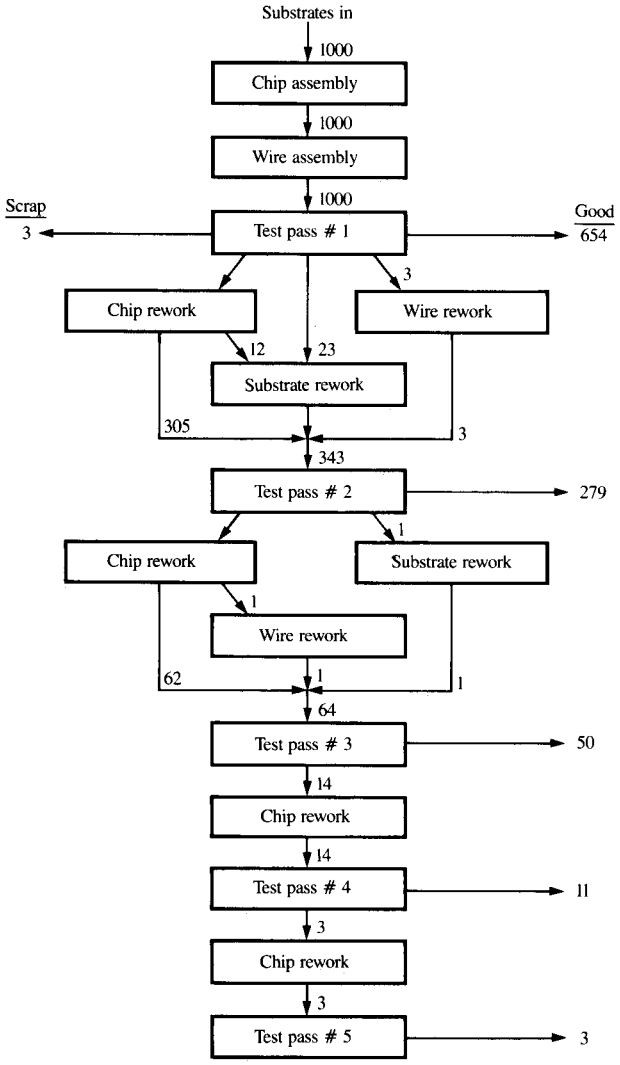

Figure 8 Interpretation of a hypothetical sample simulation explained in the text. Of the 1000 parts entering the assembly process, three parts are scrapped after the first testing pass, and three require five rework processes before the entire batch is accounted for. The data for this flow is obtained from the PARTY output listing of Fig. 9.

These, and most other extensions of the test/rework scenario, can generally be accommodated by the provision of blank positions in registers DNF and DFD into which additional records can be placed as needed, causing appropriate action sequences to be adopted by the model as the simulation proceeds. For example, even the requirement for a rinse operation following certain types of processing can be encoded into the model by the creation of dummy records in DFD.

Input data

As is well known, a model is only as good as the information which is fed into it. The design of the PARTY model presupposes the availability of a large fund of detailed input data. These data have to be carefully selected for use by the model from the many types of information that are available—some firm and well understood, and others very much based on engineering judgment. They fall into three general categories: 1) bills of material for all assembly types, 2) testing plans and projections of tester coverages, and 3) defect density estimates.

Though many of the estimates are necessarily based on intuition, they can be revised routinely on the basis of experience and process control measurements; and it is encouraging to note that as the model becomes more refined, some helpful optimizing effects can be expected from mixing of overestimated and underestimated values.

Simplified example

To demonstrate the capabilities of the model, consider a hypothetical module assembly. With defect rates set high enough to cause traffic through the test and rework loops, and using a manufacturing sequence illustrated in Fig. 7, PARTY enables the user to break out the traffic flow as illustrated in Fig. 8.

The defect categories shown in Fig. 9 were provided as input to the model, together with their respective probabilities of occurrence. In order to keep the model simple for this example, we defined four defect types (AA–AD) for circuit chips arriving from a vendor; four defect types (BA–BD) for chip joining when new, and three defect types (CA–CC) during rework; three defect types for substrate wiring when new (DA–DC), and one defect type during rework (EA); also, three defect types for new substrates (FA–FC), and one defect type during rework (GA). With this information, the model provides the user with a complete account of the defect types and quantities which caused this traffic flow.

In this particular simulation of the 1000 assemblies entering the test, 654 passed the first time, 343 were sent to rework (enter the second column of Fig. 9), and three assemblies could be categorized as not economical to repair,

and consequently would be scrapped. This simulation further shows that three assemblies would go through rework five times until they finally passed. A convenient summary of the rework routings is also provided.

Summary and conclusion

The model is flexible enough to simulate any reasonable process sequence and provide full accounting of how many parts passed through any stage in the process. It is capable of providing a full description of every defect which was predicted as being detected or reworked at any process stage; and it can accommodate the use of several different types of test equipment, both alone and in combinations. The test coverages provided by each of these testers will differ, but the degrees of redundancy between two testers may, in some cases, vary considerably from the simple probabilistic overlap [4]. The model is also capable of predicting the types and quantities of defects which can be expected to escape detection at the assembly line testers and thus enter the next level of the manufacturing process.

The model simulation is unconventional in that its emphasis is on *defect* flow rather than *assembly* flow. It is deterministic in nature, providing an assessment of the total workload imposed upon every sector in the manufacturing process for a given quantity of parts assembled, without reference to throughput constraints or processing time.

The output of the model is usually normalized, to enable ready application against scheduled requirements for completed assemblies in more than one manufacturing location. It has obvious applications as an input to an iterative simulation of the manufacturing process and has in fact been used for that purpose; but it is generally applied directly to the creation of tool plans, manpower plans, space plans, and plans for the computer support of manufacturing operations.

Acknowledgments

The author acknowledges the following contributors to the work reported: C. J. Kraus, for validating the original model; A. J. Andres and W. H. McAnney, for discussions leading to the final model; L. E. Euvino, for providing the input data base for tuning and the production model; P. H. Bardell and K. M. Dooley, for support, suggestions, and encouragement in writing this paper.

References and note

1. A. J. Blodgett and D. R. Barbour, "Thermal Conduction Module: A High-Performance Multilayer Ceramic Package," *IBM J. Res. Develop.* **26,** 30–36 (1982).
2. Donald P. Seraphim, "A New Set of Printed-Circuit Technologies for the IBM 3081 Processor Unit," *IBM J. Res. Develop.* **26,** 37–44 (1982).
3. L. Gilman and A. J. Rose, *APL: An Interactive Approach,* John Wiley & Sons, Inc., New York, 1976.

DEFECT CATEGORY TYPE DEFECT QUANTITIES BY TEST PASS

		1	2	3	4	5	6	7
CHIP (NEW)	AA	73						
	AB	133	2					
	AC	77						
	AD	21						
CHIP JOIN (NEW)	BA	29						
	BB	10						
	BC	10						
	BD	10						
CHIP JOIN (REWORK)	CA		65	14	3			
	CB		4	2	1			
	CC		1					
WIRES (NEW)	DA	1						
	DB	1						
	DC	1						
WIRES (REWORK)	EA		1					
SUBSTRATE (NEW)	FA	30						
	FB	3						
	FC	3						
SUBSTRATE (REWORK)	GA		1					
DIAGNOSED-REMOVED	HA	85	15	3	1			
TOTALS		487	89	19	5			

TEST SUMMARIES:

ASSEMBLIES ENTERING TEST	1000	343	64	14	3		
ASSEMBLIES ACCEPTED	654	279	50	11	3		
ASSEMBLIES SENT FOR REWORK	343	64	14	3			
ASSEMBLIES SCRAPPED	3						

REWORK ROUTINGS:

CHIP REWORK ONLY	305	62	14	3			
WIRE REWORK ONLY	3						
SUBSTRATE REWORK ONLY	23	1					
CHIP AND WIRE REWORK		1					
CHIP AND SUBSTRATE REWORK	12						
WIRE AND SUBSTRATE REWORK							
CHIP/WIRE/SUBSTRATE REWORK							

Figure 9 PARTY output listing of the hypothetical example. The listing consists of three parts: details of defect discoveries by test/rework passes, summaries of defect discoveries, and summaries of rework routings.

4. Note that in some cases, one type of tester may provide a 90% discovery effectiveness for a particular defect category, vs 75% for another type of tester, yet the 75% might be contained entirely within the 90%.

Received September 8, 1981; revised October 14, 1982

Brian J. Dooley *IBM Data Systems Division, P.O. Box 950, Poughkeepsie, New York 12602.* Mr. Dooley is an advisory engineer currently working on technical strategy in the manufacturing engineering organization associated with the advanced sub-products used in the new generation of IBM computers. Prior to joining IBM in 1962, he spent ten years with the English Electric Company, Ltd., working on the design and development of high-voltage insulation systems. He received a B.Sc. in physics and mathematics from Liverpool University in 1952, followed by extensive postgraduate studies in statistics and electrical engineering at the Liverpool College of Technology. Since joining IBM, Mr. Dooley has specialized primarily in data analysis and prediction, and was for several years manager of a group studying the reliability of electrical components in computers.