A STOCHASTIC NETWORK APPROACH TO TEST AND CHECKOUT

Lawrence J. Watters and Michael V. Vasilik

October 1970

P-4486

A STOCHASTIC NETWORK APPROACH TO TEST AND CHECKOUT

Lawrence J. Watters and Michael V. Vasilik*

The RAND Corporation
1700 Main Street
Santa Monica, California

ABSTRACT

This paper demonstrates the usefulness of GERT simulation for modeling and evaluating policies and processes in the area of test and checkout. Some of the latest developments and extensions to a GERT simulation program are used to model a test plan development process, a general test and checkout process, and specific cases of the latter.

---

*Any views expressed in this paper are those of the authors. They should not be interpreted as reflecting the views of The RAND Corporation or the official opinion or policy of any of its governmental or private research sponsors. Papers are reproduced by The RAND Corporation as a courtesy to members of its staff.

Presented at the Fourth Conference on Applications of Simulation, sponsored by AIIE, ACM, IEEE, SHARE, SCI and TIMS, in New York, December 1970.

A STOCHASTIC NETWORK APPROACH TO TEST AND CHECKOUT

INTRODUCTION

GERT (Graphical Evaluation and Review Technique) is a network analysis technique that has been developed for the formulation, modeling and evaluation of complex systems and processes. It embodies concepts found in stochastic network, signal flowgraph, PERT and semi-Markov theory. Descriptions of the development and applications of the technique may be found in References [5, 6, 10, 15, 16, 17, 18, 23 and 24].

The purpose of the paper is to demonstrate the usefulness of a GERT simulation program for modeling and evaluating various development, diagnosis and repair policies and processes encountered in the test and checkout environment.

Unless otherwise noted, the terminology and network conventions used throughout this paper correspond to those in the GERT Simulation Program II (GERTS II) manual [17]. Briefly, the following list and Figure 1 indicate those characteristics and capabilities of GERTS II and subsequent extensions that are used in this paper.

1. Nodes which are characterized by:
   a. Deterministic node type. All branches emanating from the node are taken if the node is realized.
   b. Probabilistic node type. At most, one branch emanating from the node is realized (the probabilities that determine which one of the branches is taken are specified by the user).
   c. Number of releases. One number specifies the number of releases or times that activities incident to the node must be realized before the node can be realized for the first time; the other number specifies the number of releases required to have the node realized after the first time.
2. Activities (branches) that are characterized by:
   a. Probabilities associated with whether or not the

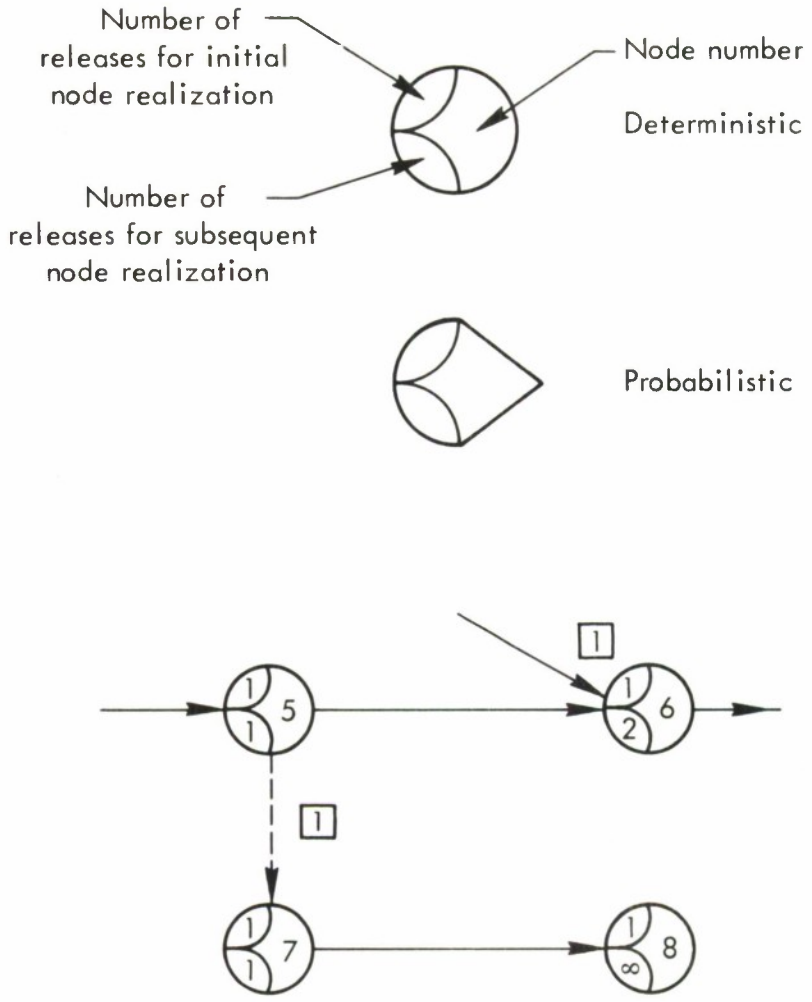

Fig. 1 —Node Characteristics and Network Modifications

branch emanating from a probabilistic node will be taken.

b. Distribution of the time to traverse the branch.

c. An activity number when completion of the activity may cause a modification in the network.

3. Network modifications which occur when completion of a modification activity causes one node to be replaced by another node. The node to be replaced is deleted from the network when the network modification activity is realized. The activities which then result are from the node that is inserted. In Figure 1, when Modification Activity 1 is completed, Node 7 replaces Node 5.

4. Counters which count the number of times a branch or set of branches is realized prior to the realization of a node.

5. "LAGs" associated with any node relative to any other node, where the LAG between any two nodes is the difference between the first realization times of the two nodes.

6. Performance measures and statistics associated with:

a. The probability that a specified node is realized.

b. The average time to realize the specified node.

c. An estimate of standard deviation of the time to realize the specified node.

d. The minimum time observed to realize the specified node.

e. The maximum time observed to realize the specified node.

f. A histogram of the time to realize the specified node.

g. In addition to statistics (b-f above) for single nodes, corresponding statistics are available for the LAGs associated with node pairs. Furthermore, because situations exist where the nodes of a pair are not on the same path in the network, the LAG between these two nodes might be either positive or negative depending upon which node is realized first. Therefore, the positive (POSLAG) and negative (NEGLAG) portions of the LAG are accumulated separately from the total LAG (TLAG).

APPLICATIONS TO TEST AND CHECKOUT

GERT may be applied to a variety of test and checkout problems at various levels of detail. At the gross level, it may be used as a communication aid or to model systems or processes for purposes of investigating overall programs, plans, policies, etc. At the detailed level, it may be used to investigate relevant statistics associated with the operational characteristics of a system or process for purposes of identifying critical components, bottleneck activities, etc.

In the following section a list of possible GERT applications to test and checkout will be presented, followed by several examples of how GERT can be used for modeling specific test and checkout processes.

POSSIBLE GERT APPLICATIONS

Procedures in test and checkout that have been identified as suitable for GERT representation and analysis include:

1. Evaluating tradeoffs between automated and manual checkout including the identification of key factors affecting the degree of automation and computer involvement in testing, and the investigation of man-machine interaction problems.

2. Evaluating the performance of alternative designs and configurations of proposed and existing automatic test equipment (ATE).

3. Evaluating tradeoffs and apportionments between built-in (on-board) and test-bench (ground) checkout.

4. Evaluating tradeoffs between system reliability and maintainability.

5. Modeling the test plan development process, including the design, interface, programming, validation and documentation stages. The model would be useful for

   a. Evaluating the effect of various test philosophies and maintenance policies on test plan design.

   b. Improving the design of the interface between the ATE and the unit under test (UUT).

   c. Determining how various validation procedures affect the time to develop a final test plan.

6. Modeling test and checkout processes for

   a. Evaluating diagnostic testing and retesting policies.

   b. Establishing methodology for setting detection thresholds, determining levels of fault isolation, and selecting and sequencing the individual tests involved in fault location;

   c. Evaluating repair policies;

   d. Evaluating condemnation policies;

   e. Establishing periodic checkout intervals based upon system reliability and maintainability.

A variety of design and performance criteria might be encountered in the above GERT applications. Performance improvement might result, for example, from

1. Increased reliability of automated checkout systems;
2. Faster test plan development.
3. Improved fault diagnostic capability (faster fault location; higher probability of correct diagnosis, reduced number of tests required, etc.).
4. Improved maintainability (reduced repair times, lower probability of an unsuccessful repair, etc.).
5. Improved retesting and repair verification (lower probability of accepting faulty equipment, lower probability of rejecting repaired equipment, etc.).
6. Increased operational readiness.

Several specific examples of how GERT can be used for modeling test and checkout processes now will be discussed.

TEST PLAN DEVELOPMENT PROCESS

GERT is useful as a visual communication aid because it can provide network representations not only of precedence relationships (as depicted by standard PERT-type networks) but also probabilistic branching and feedback loops. The following example demonstrates GERT's use as a communication aid in describing a Test Plan Development Process (TPDP). The aim of the TPDP is to develop a test plan that enables automatic test equipment (ATE) to test a given subsystem or unit under test (UUT). The plan consists of an acceptable test program stored on magnetic or paper tape and a written test document describing step-by-step test procedures. Figure 2 shows a simple activity network model of the TPDP. The process may be divided into six major stages: (1) preparation, (2) design, (3) UUT-ATE interface, (4) programming, (5) validation, and (6) final documentation. The activities associated with these stages are depicted in the GERT network model presented in Figure 3.

Preparation

Preparation of the TPDP starts with the input of a set of requirements including (1) policy considerations that reflect test philosophy and objectives, maintenance policies and inspection standards; (2) UUT information including performance criteria, hardware configuration, failure modes, test point location, etc., and (3) ATE information including hardware configuration, available stimuli, and other test capabilities. These requirements are checked to ensure compatibility of the policy considerations with UUT and ATE operational characteristics in order to establish feasibility of the test specifications. If there are neither major logical inconsistencies nor hardware limitations, the test specifications are approved to proceed to the design stage (in certain instances, interactive feedback with the customer may occur at this point to resolve any incompatibilities).

Design

A test concept is formulated and a design package is prepared. The design provides a test-oriented quasi-English language flowchart with each test step defined in terms of stimuli inputs, stimuli routing, signal conditioning, UUT operating conditions, parameter measurement identification, and decision criteria

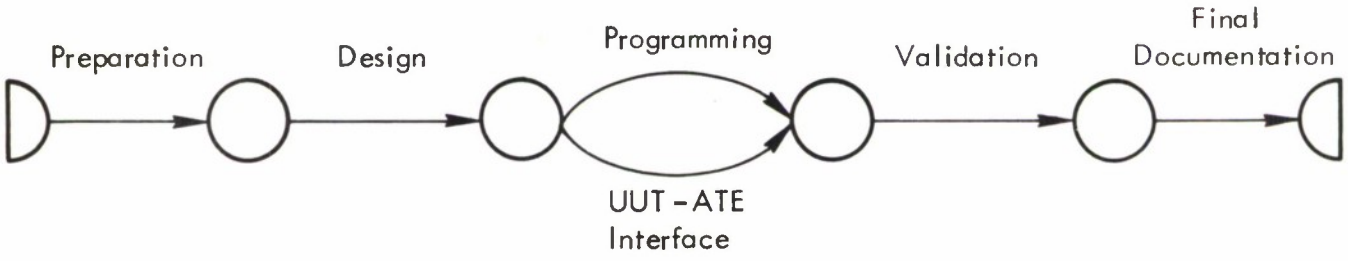

Fig. 2 -- Test Plan Development Process (General)

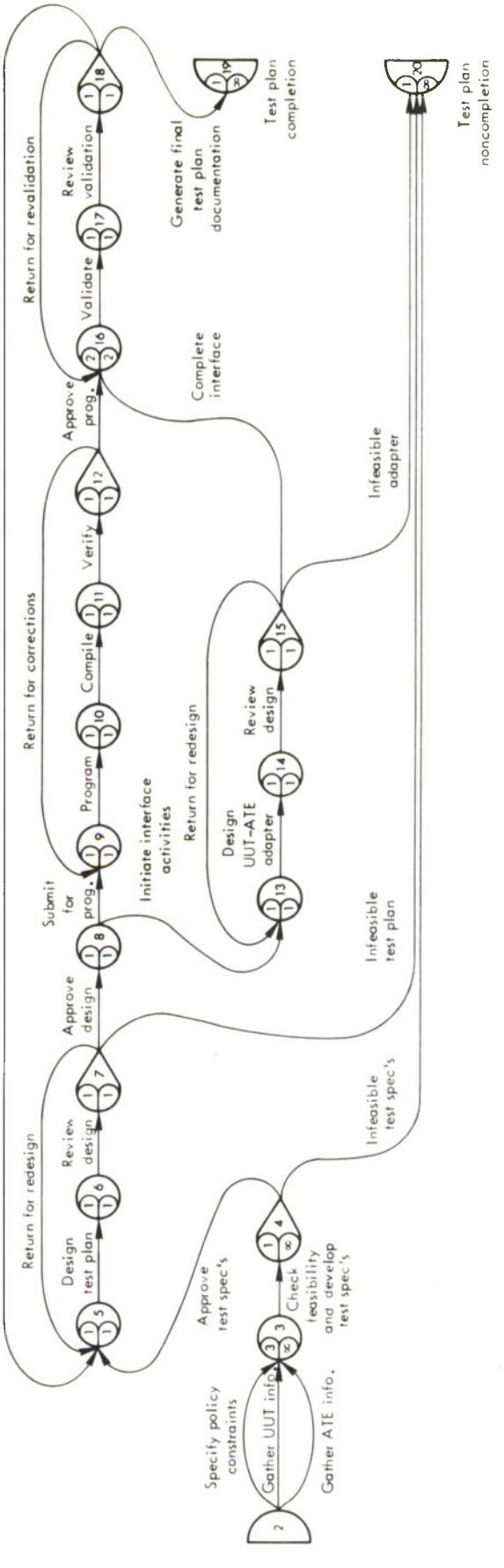

Fig. 3 -- Test Plan Development Process (Detailed)

for test sequencing. The package is then reviewed to determine if the design is still feasible and the anticipated level of test plan performance is acceptable. As a result of review, the test design is either discarded, sent back for redesign, or approved for submission to the next stage of development. The design and review activities are interactive and continue until a detailed test plan design is approved or discarded as infeasible.

UUT-ATE Interface

An adapter may be needed to interface the UUT to the ATE's stimuli generators, measurement devices, and switches. This adapter establishes a testing environment that emulates that experienced by the UUT in its operational environment. The required adapter design specifications may be obtained from the test design flow chart, ATE circuit diagrams, and wiring listings. The interface stage and the test plan programming stage (discussed in the next section) are initiated simultaneously upon completion of the test plan design stage.

Programming

The test programming stage consists of three major activities: (a) programming, (b) compilation, and (c) verification. Programming includes coding the information from the flow chart provided by the design process into a series of programmer-oriented language statements and keypunching cards to form an input source deck.

Compilation involves converting the programmer-oriented language source deck to a machine-oriented object deck. The output of the compilation will vary with the ATE and the programming languages used. For example, a sophisticated compiler may accept a test-oriented meta-language input and, in addition to the normal translating and assembling functions, it may automatically check the legality of source language statements, assign ATE resources (e.g., power supplies), and generate a listing of wire interconnections from the ATE to the UUT.

Verification is concerned with determining whether or not the program accurately represents the intended design. It is a visual process of comparing the flow chart and the compilation output list. Errors are corrected by modifying and recompiling the source deck. These comparisons and modifications are continued until the program is completely verified.

Validation

Validation is concerned with determining whether or not the program adequately represents "real life," viz., whether or not it can detect and isolate the faults of interest. Complete validation requires exercising the UUT in all operational and failure modes under the control of the newly verified program. This is accomplished by first testing a known operational UUT in all operational modes. The program is corrected as required until all tests are GO. The program then is validated for failure modes by forced NO-GO branching, i.e., at each point in the GO, NO-GO chain, a test result is forced out of limits by manual intervention. Next, actual failures are introduced into the UUT to test the ability of the program to detect and isolate the known faults that might not otherwise be revealed by forced branching. Numerous problems may arise during validation, including program coding errors, faulty test logic, improper test limits or test points, adapter interface wiring errors, an undetected faulty UUT, and/or equipment failures. Therefore, considerable time and effort must be devoted to the validation activity before the program is considered acceptable for documentation.

Final Documentation

After validation and final approval of the test program are obtained, final flow charting, coding, compilation, and documentation are completed and copies of the test program are generated.

DIAGNOSTIC TEST AND REPAIR PROCESS

The following example demonstrates GERT's use for modeling a Diagnostic Test and Repair Process (DTRP). Figure 4 shows an aggregated generalized activity network of the DTRP. The main objective of the DTRP is to decide, based on an initial series of diagnostic tests, whether or not a given UUT is defective. A significant feature of the process is that if the UUT is disapproved as defective, an attempt is made to locate the defect or fault and, based on the results of this fault location activity, a decision is rendered to repair or condemn the unit.

Figure 5 elaborates on the process in a more detailed GERT network. For this particular network, the input incorporates any a priori knowledge of the percentage of defective units expected (this would be reflected by the probabilities associated with activities emanating from the source node). Analysis of historical data or exhaustive testing on a sample number of UUT's may provide the basis for estimating this percentage.

The network in Figure 5 is essentially a graphic representation of the hypothesis testing involved in the DTRP. For example, if the null hypothesis assumes the unit is defective and should be condemned, Activity (8, 17) represents the case of approving a defective unit (or in statistical terms, rejecting the hypothesis that the unit is defective and should be condemned, when in reality the hypothesis is true). Similarly, Activity (6, 15) represents the case of condemning a nondefective unit (or in statistical terms, accepting the hypothesis that the unit should be condemned when indeed it should not). Therefore, the probabilities associated with realizing Node 17 and Node 15 (standard GERT statistics) represent the probabilities α and β associated with committing Type I and Type II errors respectively. GERT also can be used to model a confidence test process (e.g., periodically testing units to determine if they are still nondefective). In this case, all units might be assumed to be nondefective unless a priori information to the contrary exists.

Although each of the activities in Figure 5 is relatively self-explanatory, some aspects of the network deserve elaboration.

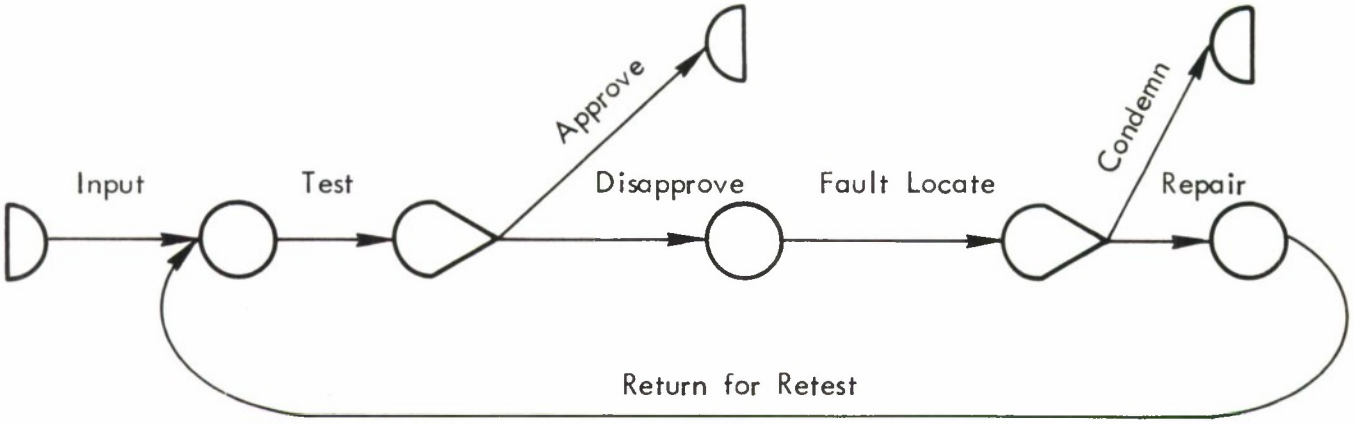

Fig. 4 -- Diagnostic Test and Repair Process (General)

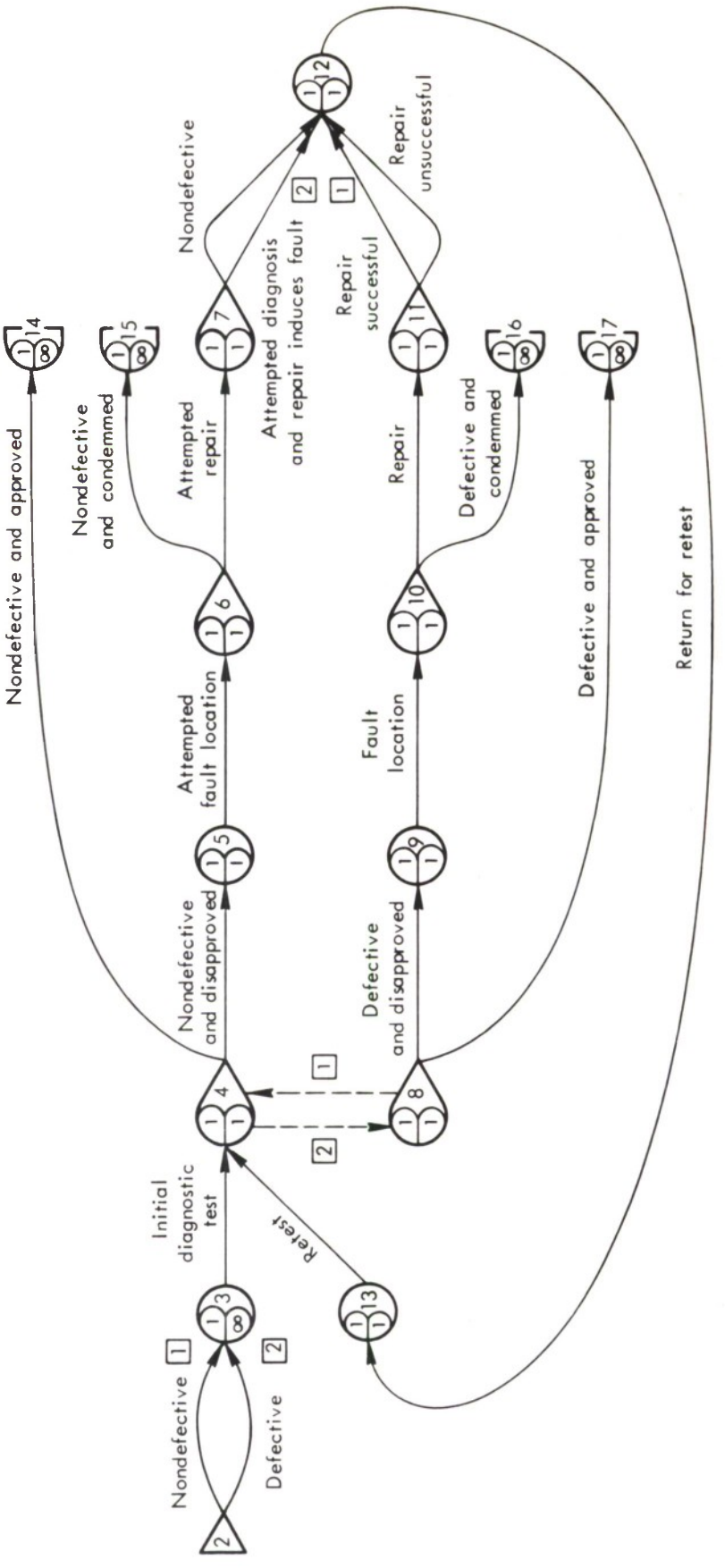

Fig. 5 --Diagnostic Test and Repair Process (Detailed)

Basically the network has two main paths, one representing activities associated with the test process when the UUT is in a nondefective state, and the other when the UUT is in a defective state. It is necessary to differentiate between these two paths because the characteristics of the activities may differ depending upon the state of the UUT. For example, one might be expected to associate (and be able to estimate from the analyses of historical data) a smaller probability of disapproval and a larger attempted fault location time for nondefective UUT's than for defective UUT's. The UUT may enter the test and repair process in either a defective or a nondefective state, and may change state during the test and repair process. The network modification capability of GERT (transposition of Node 4 and Node 8) effects this change of state. The true state is known only when perfect diagnosis and perfect repair exist. More commonly (and as depicted by our network) the approval and disapproval decisions are made when the true state is not known with certainty, thereby creating the attendant risks of committing Type I and Type II errors.

The given network represents only one of a number of possible diagnostic and repair policies. For this case, if a UUT is disapproved, fault location is initiated, and then, based on results of this activity, a decision is made to condemn or repair. (It may be obvious after initial testing that a particular UUT is beyond repair, in which case the fault location time is zero.) An additional policy may dictate, for example, that if the UUT still has not been approved after the n^{th} retest, it is automatically condemned. The network easily may be restructured to represent this new policy, as depicted in Figure 6. For this policy, Node 18 acts as a counter (counting the initial diagnostic test and the number of retests). A $t = 0^{+}$ time is associated with Activity (5'5") and Activity (9'9") to ensure that the replacement has occurred after Activity (18,19) is realized. This could be accomplished by assigning a small positive time, say 0.001, to the activities.

Once a network reflects the desired policies to be evaluated, then the statistics of interest or mreasures of performance can be obtained from the simulation. An example of how this might be done is presented in the next section.

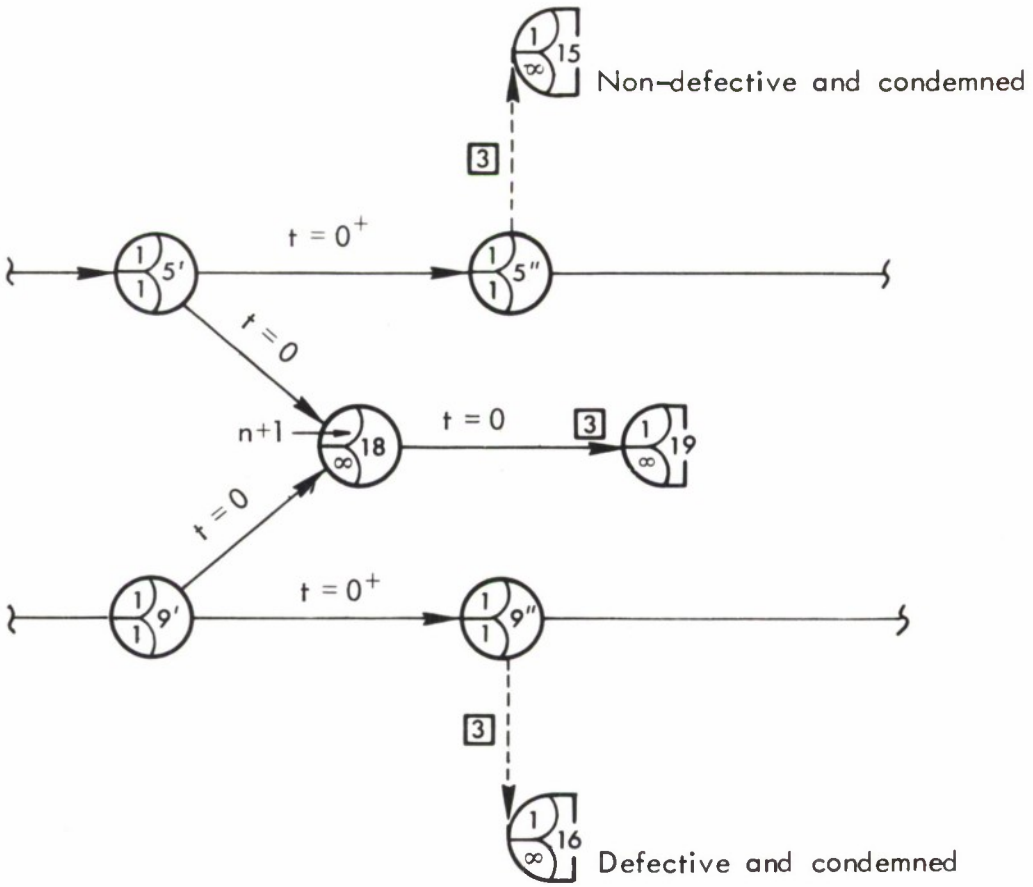

Fig. 6 -- Portion of a GERT Network Depicting Policy of Automatic Condemnation of UUT if Disapproved After n Retests

PERIODIC CHECKOUT AND REPAIR PROCESS

The following example demonstrates GERT's use for modeling and evaluating the general periodic checkout and repair process depicted by the network in Figure 7. This process deals with a type of system that is to be maintained in an operational status, has an exponential failure distribution with a constant failure rate λ, and periodically undergoes a checkout to determine whether or not it is still operational. The system undergoes the checkout of duration t=s every c time periods. The checkout imposes a stress (probability of failure) q on a good system. Furthermore, time-deterioration of a good system starts anew (or because the failure rate is constant, continues) at the end of the checkout. That is, even if one component of the system fails and initiates a repair activity, the system is sufficiently complex that the overall failure characteristics of the system are unaffected.

A special case of the checkout and repair process will be developed here to demonstrate the utility of several features of GERTS (viz., clocks, counters, network modifications, LAG's) for analyzing test and checkout processes. This particular model was selected because of its simplicity, and also because it can be verified analytically. Once verified, the model can be expanded to more adequately represent "real life" processes. (In addition, this simple case can serve as a basis for future development of a cost analysis capability for GERTS.)

For this special case it is assumed that the system undergoes a checkout of duration s=0, and the time between checkouts is constant (i.e., isochronal according to calendar time). The checkout is assumed to uncover any existing failure (i.e., perfect checkout) whether caused by the checkout stress or by natural deterioration. If a failure is detected by the checkout, a repair time r (assumed to be constant) is required to correct the previously identified cause of failure. Table 1 summarizes these assumptions and initial conditions.

Two different aspects of the periodic checkout and repair process have been modeled. The first is a network representation of

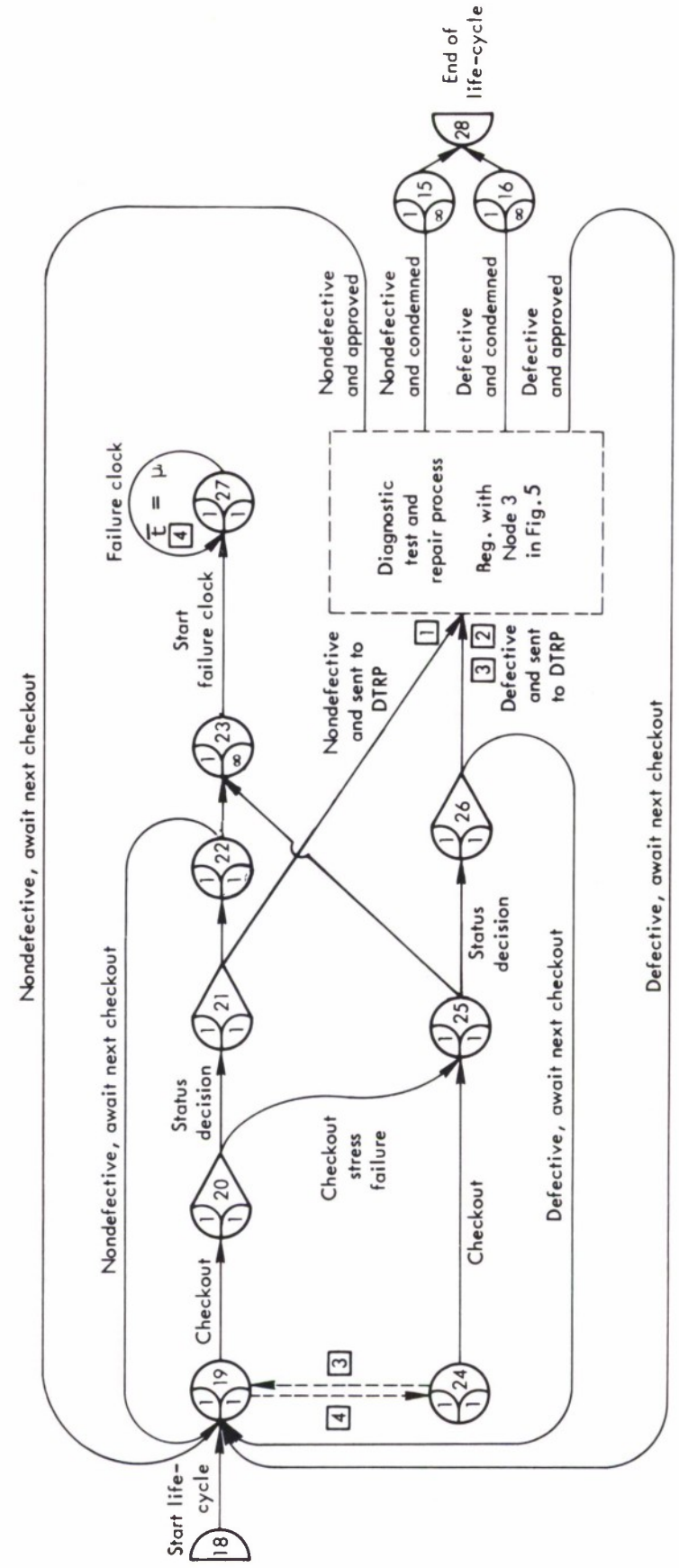

Fig. 7 -- System Life-Cycle With Periodic Checkout

Table 1

ASSUMPTIONS AND INITIAL CONDITIONS FOR SPECIAL CASE OF A PERIODIC CHECKOUT PROCESS

ASSUMPTIONS

1. Systems tested have negative exponential failure distributions with mean $\mu = 1/\lambda$.
2. Checkout is perfect, i.e., all failures that exist are correctly detected; otherwise the system is correctly identified as being operational. Also, checkout time is zero (s=0).
3. Repair is perfect, i.e., upon completion the repaired component is operational. However, other components of the system may have failed during the repair period. Also, repair time is constant and less than the periodic checkout cycle time ($r < c$).
4. Periodic checkout cycle is isochronal, i.e., recurring at regular intervals (calendar time).

INITIAL CONDITIONS

1. System under test is operational at the time of the first checkout, i.e., Node 4 of Figure 8 is "in" the network.

the process as it continues over a large period of time, assuming that the system is initially (at time T=0) in an operational state. The second is a network representation of the process over a single "typical" checkout cycle of duration c, where the initial transient caused by the "operational at T=0" assumption has been neutralized. Simulation of these two networks will be used to estimate a useful performance measure for evaluating alternative checkout and repair policies (such as a policy specifying the time c between checkouts). This measure is the "ready rate" of the system, where ready rate represents the proportion of time the system is "in-commission" (i.e., operational and not being subjected to checkout, repair or other activities that make the system unavailable for immediate use.) Each of the two models (checkout process over time and checkout cycle) and corresponding network representations will now be discussed in more detail.

CHECKOUT AND REPAIR PROCESS OVER TIME

The network depicted in Figure 8 represents the process operating continuously over a period of time ($t=\tau$) and initially operational. The feedback loop on Node 3 represents the checkout cycle clock that periodically (every c time periods) regenerates a checkout activity. Another clock must be established so the network may be realized and the simulation ended. This simulation clock is represented by Activity (2,13) with $t=\tau$, where Node 13 is the (only) sink node in the network.

The purpose of the checkout activity is to discover whether or not the system is nondefective (represented by Node 6) or defective (represented by Node 9). When the system is nondefective prior to checkout, the checkout activity including checkout stress may be thought of as the path containing Nodes 3, 4, 5 and 6 when the system survived the checkout stress, and Nodes 3, 4, 5 and 9 when the checkout stress caused the system to fail. During q x 100 percent of the time, the stress of a checkout will cause a nondefective system to fail. When the system is defective prior to checkout, the checkout activity is represented by the path containing Nodes 3, 8 and 9. Thus, Node 9 represents the end of a checkout cycle when the system

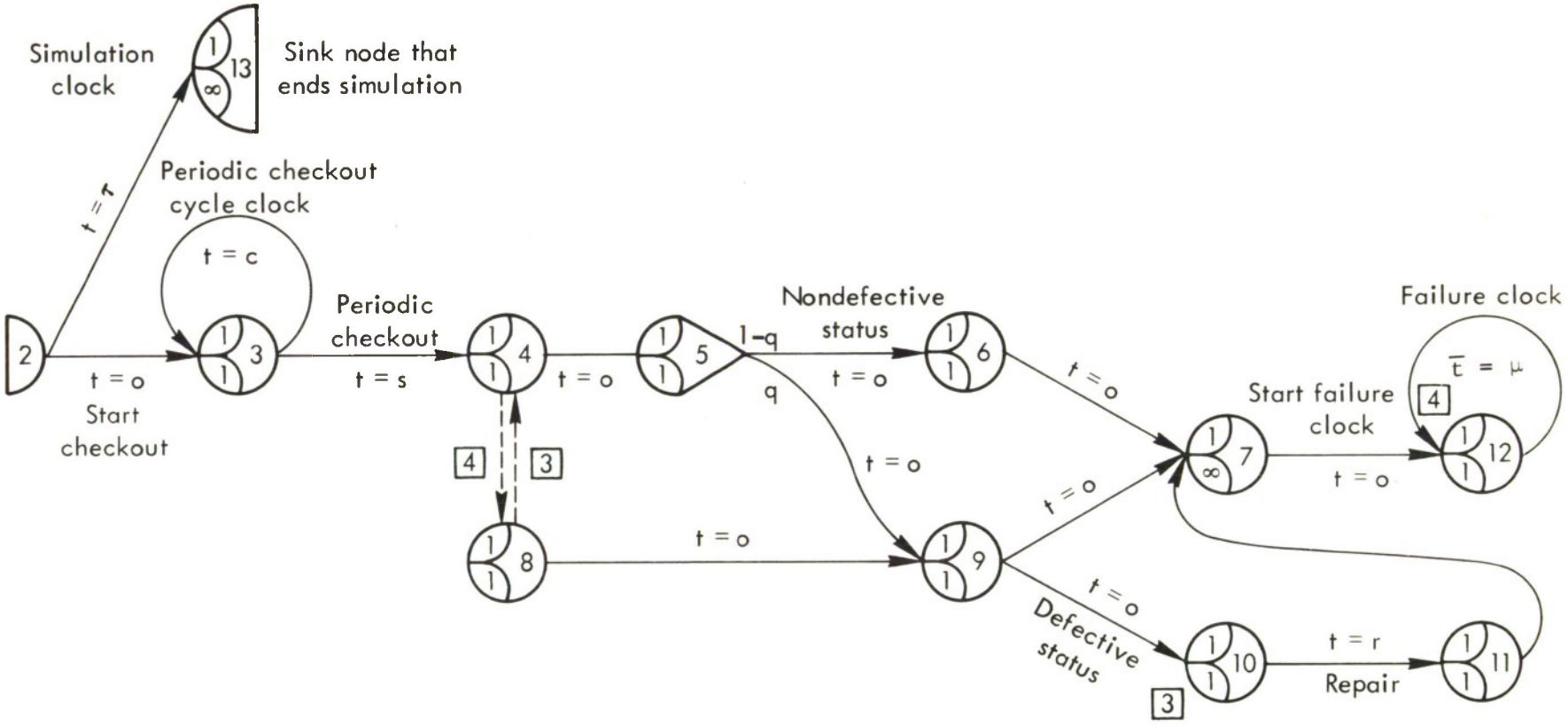

Fig. 8 -- Special Case of a Periodic Test and Checkout Process
(Simulated Over a Period of Time τ)

either was nondefective prior to the checkout and did not survive the checkout stress or had failed due to natural deterioration since the checkout in the previous checkout cycle. At this time the system is known to be defective and repair is initiated on the defective component(s).

On the first cycle, either Activity (6,7) or (9,7) is used to start the failure clock depending upon which condition prevails after the first checkout stress. After the first cycle, Node 7 merely serves as a convenient sink for Activity (11,7). (Otherwise, Node 13 would not be the only sink node in the network.) Once started, the failure clock represented by the feedback loop on Node 12 continues to regenerate component failures randomly, with the mean time between failure equal to μ. Whenever a failure occurs, Modification Activity 4 replaces Node 4 by Node 8 (if it has not already done so), thereby reflecting the fact that the system will be entering the next scheduled checkout as nonoperational. The failed components will not be identified until the end of the cycle (i.e., the end of the next scheduled checkout), at which time Modification Activity 3 returns the network to its original configuration, the system goes into repair and the cycle commences anew.

The approach taken for this analysis was to simulate the network for a period of time considerably greater than the checkout cycle time ($\tau >> c$) in order to neutralize the effect of the initial transcient caused by the assumption that the system is initially operational.* Throughout the simulation a count was taken of the number of times the system was in a defective status at the end of the checkout. This was done by counting the number of times Activity (9,10) was realized. The ratio of this number to the total number of checkouts made was then used as an estimate of the probability (p_d) that the system is defective at the end of any "typical" checkout cycle. (This probability equals q only after the first checkout because of the "operational at T = 0" assumption.)

In particular, the following parameters were used:

---

*The procedures for determining the appropriate length of simulation runs and/or for determining the appropriate number of network realizations for each simulation run are not presented here; rather, we will be concentrating on the modeling aspects of the problem. The reader is referred to Ref. [8] for a discussion of sample-size determination techniques.

Simulation time,	$\tau = 10{,}000$
Cycle time,	$c = 10$
Checkout time,	$s = 0$
Stress probability of failure,	$q = 0.25$
Repair time,	$r = 8$
Mean time to failure,	$\mu = 25$

The output of the simulation indicated that the counter associated with Activity (9,10) was realized 498 times during the $\tau/c = 1000$ cycles. Hence, the estimated probability p_d that the system is defective at the end of a typical cycle (or at the beginning of the next cycle) is 498/1000 = 0.498.* This statistic will now be used as one of the parameters in the checkout cycle model.

CHECKOUT CYCLE

The network of Figure 9 illustrates the checkout cycle where the probability of the system leaving the checkout in a defective state is the p_d obtained from the preceding simulation. Note that because the system has a negative exponential failure distribution, the failure clock can be reinitialized by Activity (4,7) even though the system remains operational from a previous cycle. (Again, Node 7 is used as a convenient sink for Activity (6,8).)

The network contains two sink nodes and both are to be realized before the network is realized. The interval of interest for a single realization of the network, however, is from time T=0 to time T=c and the statistic of interest for this interval is the expected ready rate:

$$E\{\text{Ready Rate}\} = E\{t_i/c\}$$

where t_i is the total time within a cycle when the system is in commission.

---

*The analytic value for this probability is

$$p_d = q\,e^{-c/\mu} + 1 - e^{-c/\mu} = 0.4975.$$

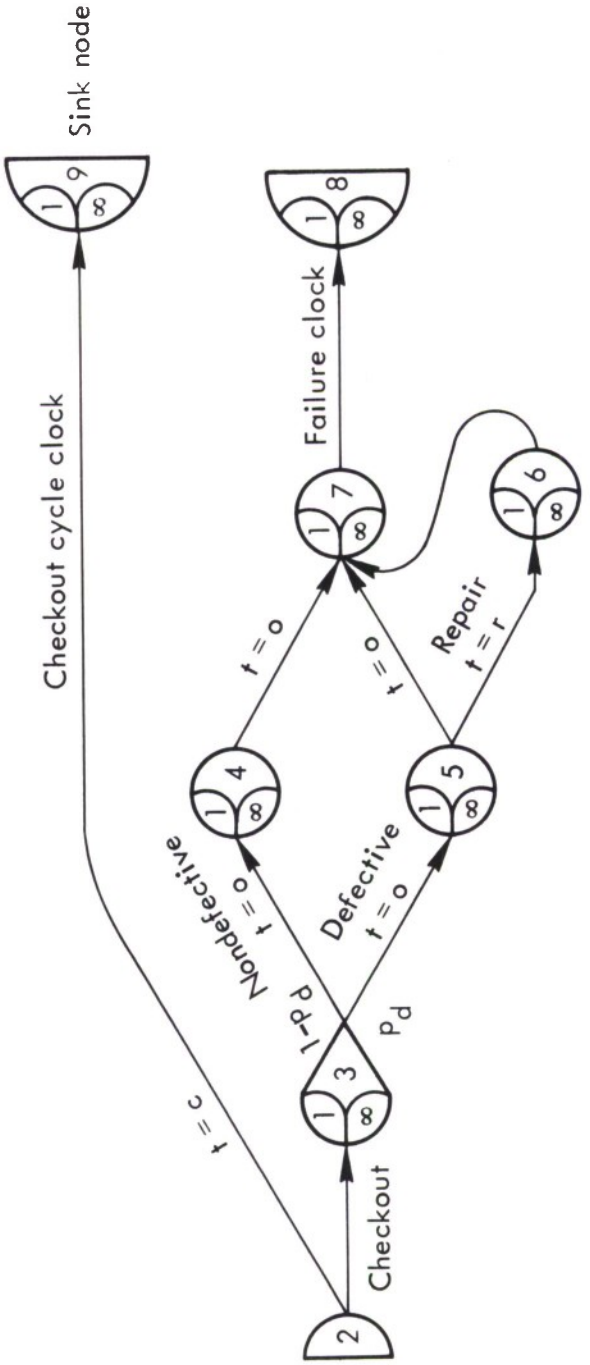

Fig. 9 -- Special Case of a Periodic Test and Checkout Cycle (Simulated Over a Cycle Time c)

The conditional LAGs (that is, the LAG given that it is positive (POSLAG) and the LAG given that it is negative (NEGLAG)) can be used to facilitate the accumulation of in-commission periods of time, where the LAG (i,j) between any two nodes N_i and N_j is the difference between the first realization times of the two nodes; viz., $T_{N_j} - T_{N_i}$. When the system is nondefective after checkout, the system is in-commission during the cycle from time T_{N_4} until $\min \left\{ T_{N_9} = c,\ T_{N_8} \right\}$, i.e.,

$$t_i \mid \text{Nondefective} = \text{POSLAG } (4,9) - \text{NEGLAG } (9,8).$$

(Note that accumulations of LAGs occur only when both nodes are realized, and further that POSLAGs and NEGLAGs are expressed in absolute values. Of course, for this simple case, POSLAG (4,9) = c.) When the system is defective after checkout, it becomes in-commission after repair has been completed unless failures have occurred since the checkout, and remains in-commission until $\min \left\{ T_{N_9} = c,\ T_{N_8} \right\}$. This relationship is expressed as

$$t_i \mid \text{Defective} = \text{POSLAG } (6,9) - \text{NEGLAG } (9,8) + \text{NEGLAG } (6,8).$$

$$\text{Jointly, } E\left\{t_i\right\} = E\left\{\text{POSLAG } (4,9)\right\} - E\left\{\text{NEGLAG } (9,8)\right\} + E\left\{\text{POSLAG } (6,9)\right\} + E\left\{\text{NEGLAG } (6,8)\right\}.$$

The network in Figure 9 was simulated 500 times per simulation run, three replications per cycle time, with the following parameters being used:

Case number		I	II	III	IV	V	VI	VII
Cycle time	c =	8	10	11	12	13	20	40
Prob. of being defective	p_d =	.454	.498	.517	.536	.544	.662	.849
Checkout time	s =	0	0	0	0	0	0	0
Repair time	r =	8	8	8	8	8	8	8
Mean time to failure	μ =	25	25	25	25	25	25	25

To illustrate the calculation of ready rate from our empirical data, selected results from one of the simulation runs are presented in Table 2 (viz., one in which c = 12). From values in Table 2 and the previously mentioned relationship between $E\left\{t_i\right\}$ and LAGs:

Table 2

SELECTED SIMULATION RESULTS FOR c = 12, ONE REPLICATION

	Mean	Std. Dev.	Min.	Max.	Prob. of Occur.
TLAG (4,9)	12.000	0.000	12.000	12.000	0.472
POSLAG (4,9)	12.000	0.000	12.000	12.000	0.472
NEGLAG (4,9)	---	---	---	---	---
TLAG (9,8)	13.313	24.164	-11.989	88.000	1.000
POSLAG (9,8)	26.474	22.959	0.234	88.000	0.602
NEGLAG (9,8)	6.594	3.369	0.095	11.989	0.398
TLAG (6,9)	4.000	0.000	4.000	4.000	0.528
POSLAG (6,9)	4.000	0.000	4.000	4.000	0.528
NEGLAG (6,9)	---	---	---	---	---
TLAG (6,8)	16.233	23.842	- 7.776	92.000	0.528
POSLAG (6,8)	27.005	23.161	0.625	92.000	0.344
NEGLAG (6,8)	3.907	2.355	0.020	7.776	0.184

$$E\{t_i\} = (0.472)(12) - (0.398)(6.594) + (0.528)(4.000) + (0.184)(3.907)$$
$$= 5.871$$

$$E\{\text{Ready Rate}\} = 5.871/12 = 0.489.^*$$

The average of the expected ready rates obtained from the three replications of each cycle time value are plotted in Figure 10. The curve corresponding to the theoretical ready rate is also provided in Figure 10 for comparative purposes. A comparison of the empirical values and the theoretical values indicates that the cycle time chosen by our approach to maximize expected ready rate would not differ markedly from the theoretically optimum cycle time.

An approach similar to the one above for evaluating alternative checkout and repair policies might be useful for more general situations, such as when imperfect checkout exists, imperfect repair exists,

---

*The analytic value is given by the equation:

$$E\{\text{Ready Rate}\} = 1 - \frac{1}{c}\left[\int_r^c (1-e^{-t/\mu})dt + (1-q)e^{-c/\mu}\int_o^r (1-e^{-t/\mu})dt + (1-e^{-c/\mu})\, r + qre^{-c/\mu}\right]$$

$$= \frac{e^{-r/\mu}\,(e^{c/\mu} - 1 + q) - q}{(c/\mu)\, e^{-c/\mu}} \quad .$$

For $c = 12$, $\mu = 25$, $r = 8$ and $q = 0.25$,

$$E\{\text{Ready Rate}\} = 0.4875.$$

An approximation method for analytically obtaining the cycle time that maximizes ready rate is discussed in Ref. [12].

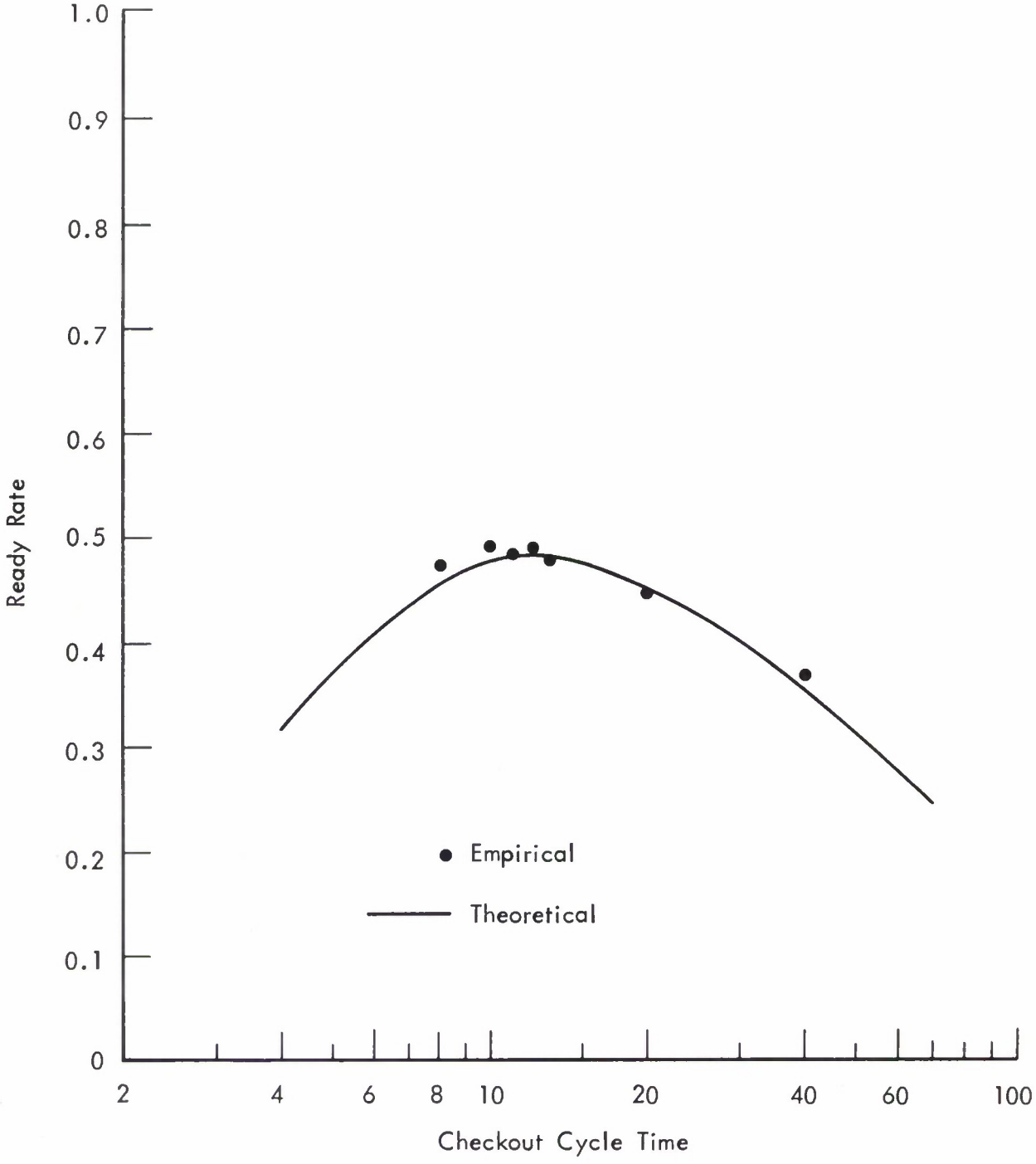

Fig.10--Ready Rate as a Function of Checkout Cycle Time

the activity times are random variables, etc. Keep in mind, however, that the particular approach of isolating a typical cycle may become difficult for certain situations, such as when using a less well-behaved failure distribution than the negative exponential for describing the system failure characteristics. In general, the modeling and analysis strategy may need to be modified depending upon the characteristics of the process being modeled. The versatility of the simulator described in this paper, however, allows such modifications to be made easily.

REFERENCES

1. Ankenbrandt, F. L. (ed.), Electronic Maintainability, Vol. 3, Engineering Publishers, Elizabeth, N.J., 1960.

2. Barbour, A. A., S.I. Firstman, and M. Kamins, Standardization of Automatic Test and Checkout Equipment: A Preliminary Discussion, The RAND Corporation, RM-2685, November 1960.

3. Barlow, R. E., et al., Statistical Estimation Procedures for the "Burn-In" Process, The RAND Corporation, RM-5109-NASA, September 1966.

4. Denby, D. C., "Minimum Downtime as a Function of Reliability and Priority Assignments in Component Repair," The Journal of Industrial Engineering, Vol. 18, No. 7, July 1967, pp. 436-439.

5. Drezner, S. M. and A.A.B. Pritsker, Network Analysis of Countdown, The RAND Corporation, RM-4976-NASA, March 1966.

6. Enlow, R. A. and A.A.B. Pritsker, Planning R&D Projects Using GERT, Technical Report on NASA/ERC Contract NASA-12-2035, Arizona State University, Tempe, Arizona, June 1969.

7. Firstman, S. I., A. A. Barbour, J. R. Brom, N. Jordan, M. Kamins, K. H. Meyer, and B. J. Voosen, An Omnibus of Briefing Papers on Analysis of Automatic Checkout Equipment and Aids to its Design, The RAND Corporation, RM-2750, June 1961.

8. Fishman, G. S., Digital Computer Simulation: Estimating Sample Size, The RAND Corporation, RM-5866-PR, August 1969.

9. Goldman, A. S., and T. B. Slattery, Maintainability: A Major Element of System Effectiveness, John Wiley and Sons, Inc., New York, 1964.

10. Hill, T. W., Jr., "The Improvement of System Performance Through Sensitivity Analysis Using GERT," Industrial Engineering Research: Bulletin No. 3, Arizona State University, Tempe, Arizona, January 1967, pp. 30-48.

11. Jirauch, D. H., "Software Design Techniques for Automatic Checkout," IEEE Transactions on Aerospace and Electronic Systems, AES-3, November 1967, pp. 934-940.

12. Kamins, M., Determining Checkout Intervals for Systems Subject to Random Failures, The RAND Corporation, RM-2578, June 1960.

13. Moon, W. D., "Periodic Checkout and Associated Errors," IEEE Transactions on Aerospace, April 1964, pp. 356-372.

14. Pritsker, A.A.B. (ed.), The Formulation of Automatic Checkout Techniques, Battelle Memorial Institute, Technical Report No. ASD-TDR-62-291, March 1962.

15. -----, GERT: Graphical Evaluation and Review Technique, The RAND Corporation, RM-4973-NASA, April 1966.

16. Pritsker, A.A.B., and W. W. Happ, "GERT: Graphical Evaluation Review Technique, Part I Fundamentals," The Journal of Industrial Engineering, Vol. 17, No. 5, May 1966, pp. 267-274.

17. -----, and P.C. Ishmael, GERT Simulation Program II (GERTS II), Technical Report on NASA/ERC Contract NASA-12-2035, Arizona State University, Tempe, Arizona, June 1969.

18. -----, and G. E. Whitehouse, "GERT: Graphical Evaluation and Review Technique, Part II Probabilistic and Industrial Engineering Applications," Journal of Industrial Engineering, Vol. 17, No. 6, June 1966, pp. 293-301.

19. RCA Automated Support Systems, Technical Journal published by Aerospace Systems Division of RCA, Burlington, Massachusetts, 1968.

20. St. Clair, E., "The Diagnosis Process," Proceedings in Automatic Checkout Techniques Held at Battelle Memorial Institute, September 1962.

21. Stuehler, J. E., "Hardware--Software Tradeoffs in Testing," IEEE Spectrum, December 1968, pp. 51-56.

22. Thompson, W. B., Employment of Launch Site Test Equipment for Maximum System Reliability, General Electric Company, Santa Barbara, California, Report No. SP-71, TEMPO, February 1960.

23. Whitehouse, G. E., and A.A.B. Pritsker, "GERT: Part III-Further Statistical Results; Counters, Renewal Times and Correlations," AIIE Transactions, Vol. 1, No. 1, March 1969, pp. 45-50.

24. -----, and L.J. Riccio, "Application of GERT to Determine Effective Checkout Procedures and Failure Diagnosis in Microcircuits," paper presented at the ORSA 36th National Meeting, Miami, Florida, November 1969.

BIOGRAPHIES

DR. LAWRENCE J. WATTERS received a B.S. in Engineering Physics from Montana State University and a Masters of Business Administration (MBA) and Ph.D. in Industrial Engineering (Operations Research major) from Arizona State University. Dr. Watters is president of Anacomp, Inc. and consultant to The RAND Corporation. His consulting and research experience includes the design and use of management decision models, project selection and capital budgeting techniques, simulation analyses, and resource planning, scheduling and control techniques. Dr. Watters has authored various Rand publications and has published articles in such technical and professional journals as Operations Research, The Journal of Industrial Engineering, and Management Science. His professional and honorary memberships include Operations Research Society of America, The Institute of Management Sciences, Tau Beta Pi (Engineering), Alpha Pi Mu (Industrial Engineering), Sigma Iota Epsilon (Management) and the Society of Sigma Xi (Scientific Research).

DR. MICHAEL V. VASILIK received a B.S. in Chemical Engineering from Newark College. Upon graduation as an Air Force ROTC Distinguished Military Graduate, he was commissioned into the United States Air Force and attended the Air Force Institute of Technology, receiving a M.S. in Astronautics - Space Facilities. He received his Ph.D. at Arizona State University in Industrial Engineering, Operations Research. Formerly engaged in research at The RAND Corporation, Dr. Vasilik currently is associated with the Arnold Engineering Development Center, Tennessee. He is a member of Omicron Delta Kappa, Tau Beta Pi, Arnold Air Society, Omega Chi Epsilon, Alpha Pi Mu, Phi Eta Sigma, Operations Research Society of America, The Institute of Management Sciences, and the American Institute of Industrial Engineers.