

PART B
A SIMPLE CALCULUS FOR DISCRETE SYSTEMS

PART B

CONTENTS

	<u>Page</u>
I. THE NEED FOR A CALCULUS	131
II. THE CALCULUS	134
III. USE OF THE CALCULUS	154
REFERENCES	159

I. THE NEED FOR A CALCULUS

The complex utility systems and weapon systems that are built out of public resources are basically problem solving systems; society buys them in the hope that they will serve to solve problems that broadly affect society. In recent times such systems have been developed at an accelerating rate. To some extent each new system builds upon the systems that have been developed in the past, and, in this manner, the overall size and complexity of systems tends to grow. Although this growth in size and complexity has made these new systems more difficult to produce, society has nevertheless produced many of them when urgent pressures have been broadly recognized. Thus, in recent years society has been able to solve problems that were thought impossible of solution a few years ago. The importance to society of the new complex systems is great and there are strong pressures to continue to solve those problems that are shifted into the realm of solvability by advancing technology.

With increasing complexity and with increasing importance of the problems encountered, there has also been a trend toward increasing cost of systems. Thus, in our time, new systems to serve society, such as waste management systems, power supply systems, and transportation systems require for their development such a significant proportion of our total resources that we cannot undertake them all at once, even though all are clearly within the scope of technology to build. Given as opposing factors high cost in terms of resources needed for development and great importance in achieving success, there is need for capability to predict, to design, and to control development processes for the complex systems needed, so that our resources can be used most effectively to solve as many problems as possible.

In recognition of the importance of control over the development process, in recent years there has been increasing use of one tool that is useful for

this purpose, the Program Evaluation and Review Technique called PERT. PERT was developed specifically to help solve the problem of gaining control of the development process and it does provide a partial answer to the need. However, the successful use of PERT techniques for the control of a given development cycle depends upon having an adequate description of the development process to be controlled. Given an adequate description, PERT techniques can be employed to redescribe the process in terms of resource requirements, time requirements, and contingencies. But without any description of the steps in the development process to start with, PERT is of no use. By the same token, a good PERT description cannot offset a bad process description upon which it is based. To date it appears that there is no generally available method for generating an adequate description of a development cycle so that a PERT description may be generated in turn and used to full advantage. There is a need for such a descriptive method.

There have been attempts to describe or model the process of complex system development. The most significant undertaking has been sponsored by the Air Force. With the support of the Department of Defense (ref. 3), the Air Force has prepared an horrendously detailed description of the process by which the systems built under its aegis should be developed (ref. 1). The Air Force documentation, however, does not lend itself to adaptation for solving the development cycle problem in general. It is tailored specifically for the management conventions and hierarchical relationships of the Air Force. It presents a model for system development in great detail, but the model is not one from which general principles may be extracted, nor is it one that is amenable to evolution by means of rigorous public discussion. Several authors writing in the general area of system engineering have recently presented models of what the system development process is like (ref. 2, 6, 5); none of these contains sufficient detail nor adequate rationale for it to be useful for solving the problem of gaining control of the system development process.

Although existing documented descriptions of the development process are not adequate to enable the prediction, design, and control of development cycles for complex systems, they all demonstrate that the business of designing

a development cycle is essentially that of finding a defensible strategy for the sequence and relationships of events that must take place in the course of developing a complex system. In order to be able to talk about development cycle strategies without ambiguity, and in order to promote the comparison of alternatives in the course of evolving good strategies, we need a special language; specifically, there is need for a language whose terms and concepts are public and precise and whose symbology is well defined so that there can be an exchange of precise ideas among the specialists interested in the development process. Given such a language, there would be a good basis for communicating and improving development cycle models which exhibit useful strategies.

The objective of this paper is to present a language which satisfies the needs outlined above.

The vehicle for talking about development systems that is presented here was generated within certain ground rules. A basic rule was, of course, that the language be useful for talking about development cycles. Another ground rule was that the language be presented as a calculus according to the conventions of mathematics in order to take advantage of the established methods of the mathematical community as a way of providing for the orderly improvement of the language (ref. 7). Yet another ground rule was that the calculus should articulate with PERT and with probability calculus, such that it would permit building models of development cycles which could be translated into probability equations (models) on the one hand, or into PERT models on the other (ref. 4). This ground rule was compatible with our objective that the language make it possible to utilize computers for testing and manipulating detailed development cycle models which might result from the use of the language. Finally, we hoped to provide a language rich enough to enable the evolution and elaboration of relatively complex models of the system development process, should such elaboration prove to be necessary and fruitful.

What follows then, is the presentation of a simple calculus which is a language for talking about development cycles. It is called a simple calculus

for discrete systems, because we believe that any development cycle may usefully be treated as a discrete system.¹ In this manner, we have avoided the complexity which would have been necessary had we chosen to attempt the development of a calculus for systems whose individual outputs must be described over an interval of time, or whose outputs are distributed over time. Only the test of application will reveal whether or not this was a good decision. Following the presentation of the calculus in the next section, there is a brief discussion which attempts to provide a partial justification for the specific coinage and syntax chosen for the calculus. The method of justification is to introduce the reader to the use of the calculus for describing development cycles.

II. THE CALCULUS

We begin the presentation of the calculus with a discussion and definition of the key concept, State.

State

In the definition of state, we shall employ the intuitive concept, "public method of measure." By public method of measure we mean a set of instructions which is available to a target population of people, and which, when used by members of this population, is capable of reliably guiding their actions in obtaining information about the real world. We call the information obtained (that is, the result of using the method of measure) a "symbolic statement" (e. g., 26 grams). For our purposes, a symbolic statement is a sequence of symbols from an appropriate underlying alphabet.

¹ A discrete system is one whose operation can satisfactorily be described as a finite sequence of events moving forward in time and whose terminal output state is fully described at a point in time after which no further events occur. Such a system must be one whose condition at any point of time can satisfactorily be described by stopping the clock and by identifying the complete condition of the system at that point in time. (Output state is precisely defined in the following section of this paper.)

For the purposes of the people who would obtain a symbolic statement by measurement, it would hopefully convey some information about the real world, which would be of use to them. We shall assume that there is a basic encyclopedia, \mathcal{E} , of methods of measure which is publicly available, and from which precisely defined methods of measure may be drawn.

We call the use of a public method of measure "an act of measurement." We assume that an act of measurement occurs at a particular point in time. In fact, measurements are not made at a point in time, but in most applications of the calculus there is no penalty for pretending that a symbolic statement is associated with a point in time, and to do so avoids need for undue complexity in the calculus.

We are now prepared to begin a rigorous definition of state. Roughly speaking, we want our definition of state to carry the idea that a state is the symbolic statement resulting from an act of measurement. As it turns out, as soon as we get state pinned down to this idea, we will want to expand the notion of state to include other ideas as well. Therefore, let us call this preliminary notion an atomic state; we will save the word state for the expanded notion which will come a little later on. Thus, we define atomic state rigorously as follows:

Definition: — An atomic state is an ordered triple (S, M, t) , where S is a symbolic statement, M is a public method of measure taken from the basic encyclopedia \mathcal{E} , and t is a symbol (frequently a real number). We interpret this triple as follows: S is the symbolic statement which results from the use, at time t , of the public method of measure M .

As already stated, for our purposes a symbolic statement is a sequence of symbols from an appropriate underlying alphabet. Likewise, M may be thought of as a sequence of symbols; namely, that sequence of symbols which makes up the set of instructions of which M is composed.

Suppose that M_1 and M_2 are public methods of measure. Then a combined method of measure might for didactic purposes be thought of as devised by tagging the set of instructions for M_2 at the end of the set of instructions for M_1 . Then as one finished complying with the instructions

for M_1 , it would still remain to comply with the instructions for M_2 . If people from the target population may reliably use M_1 and also reliably use M_2 , they may reliably use the combined public method of measure (denoted by M_1M_2). If S_1 and S_2 are the symbolic statements resulting from the use of M_1 and M_2 at some common instant in time, then we denote by S_1S_2 the symbolic statement resulting from the use of M_1M_2 at that same instant in time. We may now rigorously define the expanded notion of state which we need.

Definition: — We define state recursively in terms of atomic state as follows:

1. Every atomic state is a state.
2. If (S_1, M_1, t_1) is a state, and (S_2, M_2, t_2) is a different state but with $t_1 = t_2$, then (S_1S_2, M_1M_2, t_1) is a state.

Thus, our general notion of state, is that a state is something which may be composed of atomic states or other states. Notice that the recursive definition above permits states of the following structure: $(S_1S_2 \dots S_n, M_1M_2 \dots M_n, t)$. Therefore, we may think of a state as being composed of many states. That is, the state $(S_1S_2 \dots S_n, M_1M_2 \dots M_n, t)$ may be thought of as being composed of (S_1, M_1, t) , and (S_2, M_2, t) , and... and (S_n, M_n, t) . We shall call the set of states $\{(S_1, M_1, t), \dots (S_n, M_n, t)\}$ a subdivision of the state $(S_1S_2 \dots S_n, M_1M_2 \dots M_n, t)$. Notice that a given state may have many subdivisions. Thus if we let $S' = S_2 \dots S_n$ and $M' = M_2 \dots M_n$, then $\{(S_1, M_1, t), (S', M', t)\}$ is also a subdivision of $(S_1S_2 \dots S_n, M_1M_2 \dots M_n, t)$.

Instead of writing a triple each time we wish to refer to a particular state, we shall often use a single lower case letter (sometimes with a subscript) to denote a state. Thus, for example (S, M, t) might be denoted by b , and (S_1, M_1, t_1) might be denoted by b_1 . In addition, we shall sometimes denote the state which results from combining $b_1, \dots b_n$ as $\{b_1, \dots b_n\}$. (Notice that $b_1, \dots b_n$ must all have the same time t associated with them, else they cannot legally be combined into a single state.) Furthermore, if b is composed of the states $b_1, \dots b_n$, then of each b_i we shall say that b_i is an element of b . Finally, we shall sometimes use the

convention of referring to the time t associated with the state (S, M, t) as "the time at which the state occurs."

Primitive Functions

We do not use the word function as it is used in the world of mathematics. Rather, our use of the word derives from its everyday use in systems analysis. We define primitive function precisely as follows:

Definition: — A primitive function is an ordered pair of states (a, b) such that if t_a is the time associated with a , and t_b is the time associated with b , then $t_a < t_b$ (read, " t_a earlier than t_b ").

The first coordinate in the pair is called the input state, and the second coordinate is called the output state.

Isomorphic Primitive Functions

Let (a, b) be a primitive function and suppose a and b are subdivided as follows: $a = \{a_1, \dots, a_n\}$ and $b = \{b_1, b_2\}$. Eventually we would like to be able to make sense of such questions as:

1. "What is the probability that b_1 and b_2 occur, given that all of a_1, \dots, a_n occur?"
2. "What is the probability that b_1 or b_2 or both occur, given that all of a_1, \dots, a_n occur?"
3. "What is the probability that b_1 occurs given that a_7 or a_8 or both occur?"

To answer such questions as these, we must define an appropriate sample space over which to compute probabilities. This sample space will be defined in the following section, and will consist of a collection of primitive functions which are in some way "similar." This notion of similarity is contained in the following definition of isomorphic primitive functions.

Definition: — Let $F^1 = (a^1, b^1)$ and $F^2 = (a^2, b^2)$ be primitive functions, which are subdivided as follows: $a^i = (a_1^i, \dots, a_n^i)$ and $b^i = (b_1^i, \dots, b_m^i)$ where $i = 1, 2$. Then F^1 and F^2 are isomorphic if:

1. $t_{b^1} - t_{a^1} = t_{b^2} - t_{a^2}$, where t_{b^i} is the time associated with b^i , and t_{a^i} is the time associated with a^i , for $i = 1, 2$.
2. For each a_k^1 , the method of measure for a_k^1 is the same as the method of measure for a_k^2 .
3. For each b_k^1 , the method of measure for b_k^1 is the same as the method of measure for b_k^2 .

Roughly speaking, for F_1 and F_2 to be isomorphic, there must be a subdivision of the input state of F_1 and a subdivision of the input state of F_2 , which both have the same number of substates. (Likewise there must be subdivisions of the output states of F_1 and F_2 with the same number of substates in them.) In addition, the time difference between the input and output of F_1 must be equal to the time difference between the input and output of F_2 . Furthermore, the method of measure in each substate in the input subdivision of F_1 , must be the same as the method of measure for the corresponding substate in the input subdivision of F_2 . (A similar condition holds for the output subdivisions of F_1 and F_2 .)

Probability Tables for Primitive Functions

Let F be a primitive function (a, b) , where a and b are subdivided in some desired manner: $a = \{a_1, \dots, a_n\}$, $b = \{b_1, \dots, b_m\}$. With this primitive function F and its subdivisions, we associate a class of sample spaces. Each sample space X in this class is a finite set of primitive functions $\{F_1, \dots, F_n\}$, each of which is isomorphic to F . In addition we assume that X contains F . The probability measure, P , on X is defined by the number of elements in the subsets of X . Thus if Y is a subset of X , then $P(Y) = \frac{m}{n}$, where m = number of primitive functions in Y , and n = total number of primitive functions in X . Notice that $P(X) = 1$.

Example: — Let F be the primitive function $((S, M, t), (S', M', t'))$.

Then the following set of primitive functions is a sample space for F :

$$\begin{array}{l}
 F_1 (=F): \quad ((S, M, t), (S', M', t')) \\
 F_2: \quad ((S_1, M, t + \Delta), (S'_1, M', t' + \Delta)) \\
 F_3: \quad ((S, M, t + 2\Delta), (S'_2, M', t' + 2\Delta)) \\
 F_4: \quad ((S, M, t + 3\Delta), (S', M', t' + 3\Delta))
 \end{array}
 \left. \vphantom{\begin{array}{l} F_1 \\ F_2 \\ F_3 \\ F_4 \end{array}} \right\} \begin{array}{l} \text{Where } S \neq S_1, \\ \text{and } S', S'_1, \text{ and} \\ S'_2 \text{ are all} \\ \text{distinct.} \end{array}$$

In this example, neither the input nor the output of F is subdivided further than one state apiece, whereas in the general case the subdivision may be quite extensive in each state. Notice however that the input and output states of the other primitive functions F_2 , F_3 , and F_4 in the sample space are subdivided to the same extent that the input and output of F is. Thus in this example the inputs and outputs of F_2 , F_3 , and F_4 each have only one state in them, in consonance with F . Notice also that the method of measurement in each input state of F_2 , F_3 , and F_4 is the same method of measurement as in the input state of F . Likewise the method of measurement in each output state of F_2 , F_3 , and F_4 is the same as the method of measurement in the output state of F . Finally, observe that the time differences between the input and output states of the primitive functions in the sample space X are:

$$\begin{array}{llll}
 \text{time difference} & & = t' - t & \text{for } F, \\
 \text{time difference} = t' + \Delta - (t + \Delta) & = t' - t & & \text{for } F_2, \\
 \text{time difference} = t' + 2\Delta - (t + 2\Delta) = t' - t & & & \text{for } F_3, \text{ and} \\
 \text{time difference} = t' + 3\Delta - (t + 3\Delta) = t' - t & & & \text{for } F_4.
 \end{array}$$

Thus all the time differences are identical. In summary, then, we have shown that all the conditions for isomorphism exist between F , F_2 , F_3 , and F_4 , and hence X is indeed a sample space for F . Actually one would probably want a sample space to contain a very large number of primitive functions in it, rather than a mere four primitive functions as in this example. Therefore in practice, a sample space so simple as this one would not be used.

Special Notation: — If a is any state, then let S_a be the generic symbol for the symbolic statement within a . Thus, for example, if b_i is a state, then S_{b_i} denotes the symbolic statement in b_i .

Suppose F is subdivided¹ in some way, and let a be any substate in F (in either the input or the output of F). Let X be a sample space for F . Then we shall define a special subset X_a of S as follows. Let F' be any primitive function in X . Since F and F' must be isomorphic, there is a state a' in F' which corresponds to the state a in F . Then we shall let F' be a member of X_a if and only if $S_a = S_{a'}$. Evidently then F itself is a member of X_a . We call the set X_a the occurrence set for the state a . For each F' in X_a , we shall say a occurs in F' .

In the example above, the occurrence set for the state (S, M, t) is the set of functions $\{F_1, F_3, F_4\}$. Or, alternatively, we may say (S, M, t) occurs in $F_1, F_3,$ and F_4 ! Notice that F_2 is not included because the symbolic statement in F_2 which corresponds to S , is S_1 , and we assumed in the example that $S_1 \neq S$.

For any sets W and Z , $W \cap Z$ represents the set theoretic intersection of W and Z , $W \cup Z$ represents the set theoretic union, and $-Z$ represents the set theoretic complement of Z . We shall define a collection of subsets, Γ , of X as follows:

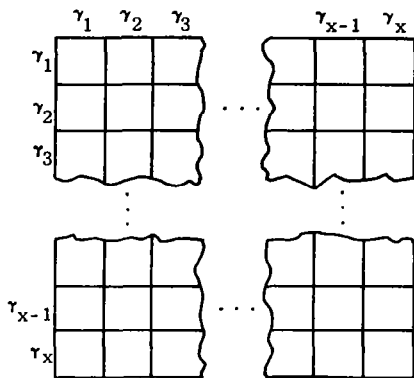
1. Γ contains X_c , for every state c in the subdivision of the input and output states of S .
2. If X_1 and X_2 are in Γ , then $X_1 \cap X_2$, $X_1 \cup X_2$, $-X_1$, and $-X_2$ are all in Γ .
3. Γ contains X .

Thus Γ is the closure under the operations \cap , \cup , and $-$, of the sets $X_{a_1}, \dots, X_{a_n}, X_{b_1}, \dots, X_{b_m}$. Let us assume that $\gamma_1, \gamma_2, \gamma_3, \dots, \gamma_x$,

¹ When we say that a function is subdivided, we employ the same idea as in the subdivision of states. Thus, a subdivided function is one whose input states and output states are expressed in subdivided form.

are the sets in Γ . Thus γ_1 might be X_{a_1} , and γ_{26} might be $X_{a_1} \cup (X_{a_3} \cap (-X_{b_2}))$.

Definition: — We define the probability table for the primitive function F to be a chart:



The labels along the top of the chart correspond to the sets in Γ . The same labels are used along the side of the chart. The entry¹ in the chart at the intersection of row γ_i with column γ_j is the ratio: $\frac{\#(\gamma_j \cap \gamma_i)}{\#(\gamma_i)}$ (see footnote 2 below). Thus this entry is the probability of γ_j given γ_i ! The unconditional probability of γ_j (i.e., $P(\gamma_j)$), is the probability of γ_j given X which is the ratio: $\frac{\#(\gamma_j \cap X)}{\#(X)}$ or, since X contains γ_j the ratio: $\frac{\#(\gamma_j)}{\#(X)}$. This unconditional probability of γ_j also appears in some entry in the chart, since X is in Γ and therefore X corresponds to one of the labels. (Notice that all the probabilities along the main diagonal of the chart are 1, since for every γ_j , the probability of γ_j given γ_j is simply 1.)

Now we can answer the sort of question that was posed in the preceding section. In that section, we considered a primitive function (a, b) , with subdivisions $a = \{a_1, \dots, a_n\}$, and $b = \{b_1, b_2\}$. The problem posed

¹ It is unnecessary to prescribe every entry in the probability table. Using $P(A \cup B) = P(A) + P(B) - P(A \cap B)$ and its generalizations all entries may be computed given relatively few.

² If γ is any finite set, then $\#(\gamma)$ means "the number of elements in γ ."

there was to make sense of such questions as:

1. "What is the probability that both b_1 and b_2 occur, given that all of a_1, \dots, a_n occur?"
2. "What is the probability that b_1 or b_2 or both occur, given that all of a_1, \dots, a_n occur?"
3. "What is the probability that b_1 occurs given only that a_7 or a_8 or both occur?"

To answer these questions we first construct a sample space X for (a, b) . Then the required probability for question number 1 is interpreted to be:

$$P = \frac{\# \left[(X_{b_1} \cap X_{b_2}) \cap (X_{a_1} \cap \dots \cap X_{a_n}) \right]}{\# \left[X_{a_1} \cap \dots \cap X_{a_n} \right]}$$

In this expression¹ for P , the set $X_{b_1} \cap X_{b_2}$ is the intersection of the set of primitive functions in which b_1 occurs with the set of primitive functions in which b_2 occurs. That is, $X_{b_1} \cap X_{b_2}$ is the set of primitive functions in which both b_1 and b_2 occur. Likewise, $X_{a_1} \cap \dots \cap X_{a_n}$ is the set of primitive functions in which all of a_1, \dots, a_n occur. Finally, $(X_{b_1} \cap X_{b_2}) \cap (X_{a_1} \cap \dots \cap X_{a_n})$ is the set of primitive functions in which b_1 and b_2 and all of a_1, \dots, a_n occur. Thus, P is the number of primitive functions (in $X_{a_1} \cap \dots \cap X_{a_n}$) in which b_1 and b_2 occur, divided by the total number of primitive functions in $X_{a_1} \cap \dots \cap X_{a_n}$. This then is how we define the probability that b_1 and b_2 occur given a_1, \dots, a_n .

¹ Notice that each set which occurs in this expression corresponds to one of the γ_i 's in Γ , by the way Γ was defined.

The required probability in question number 2 is interpreted to be:

$$P = \frac{\# \left[(X_{b_1} \cup X_{b_2}) \cap (X_{a_1} \cap \dots \cap X_{a_n}) \right]}{\# \left[X_{a_1} \cap \dots \cap X_{a_n} \right]}$$

Finally, the required probability in question number 3 is interpreted to be:

$$P = \frac{\# \left[X_{b_1} \cap (X_{a_7} \cup X_{a_8}) \right]}{\# \left[X_{a_7} \cup X_{a_8} \right]}$$

Special Notation: — For a primitive function with even a moderate number of states in a particular subdivision, the construction of a complete probability table would be wholly unfeasible. There are simply too many entries in the chart to actually fill them all in. Ordinarily, just a part of the probability table is filled in. What results, in the working situation, is a "short" probability table with only those entries of particular importance in the given situation being filled in. Perhaps the shortest, and also the most commonly used probability table is the following. Let $F = (a, b)$ be a primitive function with subdivisions $a = \{a_1, \dots, a_n\}$, $b = \{b_1, \dots, b_m\}$. A short probability table for this case is:

	$X_{b_1} \cap \dots \cap X_{b_m}$
$X_{a_1} \cap \dots \cap X_{a_n}$	P_1
$-(X_{a_1} \cap \dots \cap X_{a_n})$	P_2

(†) $\left\{ \begin{array}{l} \text{Thus } P_1 \text{ is the probability that } b_1, \dots, b_m \text{ all occur, } \underline{\text{given}} \text{ that} \\ a_1, \dots, a_n \text{ all occur. Likewise, } P_2 \text{ is the probability that } b_1, \dots, b_m \\ \text{occur given that } \underline{\text{not all}} \text{ of } a_1, \dots, a_n \text{ occur! A brief notation for this} \\ \text{table is ordinarily employed:} \end{array} \right.$

	Probability of b
a	P_1
• a	P_2

Here, we say that P_1 is the probability that b occurs given that a occurs, and P_2 is the probability that b occurs if a does not occur. When we use this terminology, it is to be understood that we are just using a shorthand terminology for what is said in the sentences above marked with a dagger (\dagger). Finally, in many cases, we shall want P_2 to be zero. Then we shall use the even shorter table:

	Probability of b
a	P_1

and it will be understood that the probability that b occurs given that a does not occur is zero. This is the only case when leaving off part of a chart allows us to deduce something about one of the entries in the part of the chart which is deleted. Ordinarily, if part of a chart is deleted, it simply means we are not interested in those entries in the deleted part.

Function

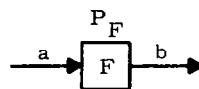
Definition: — A primitive function $F = (a, b)$ along with its complete probability table is called simply a function. A function is sometimes denoted by the symbology:



If the short table

	Probability of b
a	P_F

is what is being used, then the function may be denoted by



Probability of Output

Definition: — In the short table above, P_F is called the probability of output of the function F . Thus, P_F is the probability of b , given a .

Primitive Array

Definition: — Let t_1, t_2, \dots, t_n be the times associated with the output states of some sequence of primitive functions $\mathcal{A} = F_1, F_2, \dots, F_n$, each of which is subdivided in some pertinent way. Then \mathcal{A} is called a primitive array if:

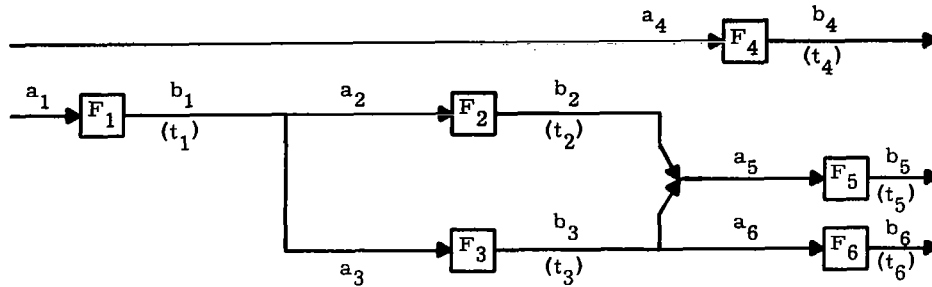
1. $t_1 \leq t_2 \leq \dots \leq t_n$
2. Every function F_j , whose output state occurs earlier than t_n has each element¹ of its output state occurring as an element in the input state of at least one of the other functions in \mathcal{A} .

Roughly speaking, any sequence of primitive functions where all the output states occur at the same time for one reason or another, is by definition a primitive array. But if two or more output states of a sequence of primitive functions occur at different times, then that sequence is an array only if all the elements of the earlier output states occur as elements in the input states of some of the other primitive functions. Thus, the only primitive functions in an array whose output states do not "feed into" other primitive functions, are those whose output states occur at the latest time. Notice that nothing is said which indicates that the elements of input states have to come from the output states of some of the other primitive functions. Our only restrictions are on output states.

Example: — Let $\mathcal{A} = F_1, F_2, F_3, F_4, F_5, F_6$ be a primitive array where: $F_1 = (a_1, b_1)$, the time associated with b_1 is t_1 ,
 $F_2 = (a_2, b_2)$, the time associated with b_2 is t_2 ,
• • • • •
 $F_6 = (a_6, b_6)$, and the time associated with b_6 is t_6 .

¹ An "element of a state" which has been subdivided into other states, is any of the states in that subdivision.

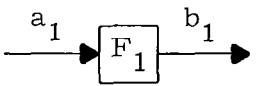
Suppose that $t_1 < t_2$, $t_2 = t_3$, $t_3 < t_4$, and $t_4 = t_5 = t_6$. This array might be represented diagrammatically¹ as follows:



We have represented \mathcal{A} in such a way as to indicate that $b_1 = \{a_2, a_3\}$, $a_5 = \{b_2, b_3\}$, and $b_3 = a_6$. With these final provisions, then \mathcal{A} is indeed a legitimate primitive array.

Isomorphic Primitive Arrays

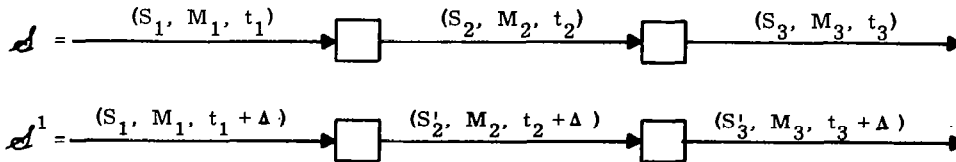
We may define isomorphism between two primitive arrays \mathcal{A} and \mathcal{A}' in a manner which is completely analogous to the manner in which isomorphic primitive functions were defined. The definition looks more complex only because in general there is more than one primitive function in a primitive array, and we must carefully correlate the states of each primitive function in \mathcal{A} with the states of each primitive function in \mathcal{A}' . A precise definition of isomorphic primitive arrays may be given as follows:

¹ Notice that in the diagram alluded to, symbols such as  occur, which look suspiciously like our symbol for a function. But in our discussion to this point, we have spoken only of primitive functions, and indeed have said nothing about any of the things (such as sample spaces or probability tables) which are required to bring in the notion of function. Later on we shall give a definition of array, in which symbols of this sort occur and are intended to be functions. Until that time, however, we shall

Definition: — Consider two primitive arrays $\mathcal{A} = F_1, \dots, F_n$, and $\mathcal{A}' = F'_1, \dots, F'_m$, with appropriate subdivisions in the input and output states of each F_i and each F'_i . Then \mathcal{A} and \mathcal{A}' are isomorphic if $n = m$, and there is a one-to-one correspondence G between the elements of the subdivisions in \mathcal{A} and those in \mathcal{A}' such that:

1. For every F_k in \mathcal{A} , if b is a state in the input (output) state of F_k , then $G(b)$ is a state in the input (output) state of F'_k .
2. For every F_k in \mathcal{A} , if (S, M, t) is a state of F_k (either in the input or in the output state) and if $G((S, M, t)) = (S', M', t')$, then $M = M'$.
3. There is a real number Δ , such that if (S, M, t) is any state in \mathcal{A} , and if $G((S, M, t)) = (S', M', t')$, then $t' = t + \Delta$.

Example: — The following two primitive arrays are isomorphic:



Probability Tables for Primitive Arrays

With each primitive array \mathcal{A} , we associate a class of sample spaces. Each sample space X associated with \mathcal{A} is a finite set of primitive arrays $\mathcal{A}_1, \dots, \mathcal{A}_n$, each of which is isomorphic to \mathcal{A} . Thus the sample space

use the symbology $\xrightarrow{a_1} \boxed{F_1} \xrightarrow{b_1}$ to indicate a primitive function, or if you will, a function without its probability table. This sort of diagram is useful because it helps us keep track of the input and output relations between the primitive functions in \mathcal{A} .

for a primitive array is no longer a set of primitive functions (as was the case for the sample space for a primitive function) but rather is a set of primitive arrays. The probability measure, p , on X is defined by the number of elements in the subsets of X . Thus, as before, if Y is a subset of X , then $p(Y) = \frac{m}{n}$, where m = number of primitive arrays in Y and n = number of primitive arrays in X . Again notice that $p(X) = 1$.

Now the probability table for a primitive array is exactly analogous to the probability table for a primitive function. A slight change (but a natural one) occurs in the definition of the collection Γ of subsets of X . Here, we define Γ as follows:

1. Γ contains X_a , for each state a , in any subdivision of any primitive function in \mathcal{A} .
2. Γ is closed under \cap , \cup , and $-$.
3. Γ contains X .

From this point on, the probability table is defined exactly as it was defined earlier for the primitive function.

Array

Definition: — A primitive array together with its complete probability table (over an appropriate sample space) is called simply an array.

Component Arrays

Definition: — Let \mathcal{A} be an array with a sample space X . Suppose \mathcal{A} is the sequence of primitive functions F_1, F_2, \dots, F_n . Let $\mathcal{A}' = F_{i_1}, \dots, F_{i_k}$ be a subsequence of \mathcal{A} . Then \mathcal{A}' is a component array of \mathcal{A} if \mathcal{A}' is itself an array, and if the sample space, \bar{X} , for \mathcal{A}' is defined as follows:

For each array $\mathcal{A}^j = F_1^j, F_2^j, \dots, F_n^j$ in X , we define an array $\bar{\mathcal{A}}^j$ as the subsequence $F_1^j, \dots, F_{i_k}^j$ of \mathcal{A}^j . The set of subsequence arrays generated in this way is the sample space \bar{X} .

This condition on the sample space of the component array guarantees that the probability tables associated with the component array are consistent with the original array \mathcal{A} .

Component Function

Definition: — A component function of \mathcal{A} , is a component array of \mathcal{A} which contains only one function.

Now that we have the notion of component function, we may speak of an array as a sequence of functions rather than as a sequence of primitive functions.

Partitioning/Adding

Definition: — Let \mathcal{A} be an array with n functions F_1, \dots, F_n . Let \mathcal{A}' be an array whose sequence of functions may be divided into n disjoint subsequences of functions $\mathcal{A}'_1, \mathcal{A}'_2, \dots, \mathcal{A}'_n$. Then \mathcal{A}' is a partitioning of \mathcal{A} if:

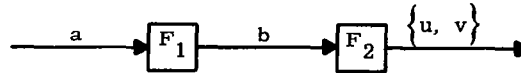
1. All the $\mathcal{A}'_1, \dots, \mathcal{A}'_n$ are component arrays of \mathcal{A}' .
2. The array inputs¹ of \mathcal{A}'_i contain all the states in the input state of function F_i in \mathcal{A} (this being true for each $i = 1, \dots, n$).
3. The array output² of \mathcal{A}'_i is equal to the output state of function F_i in \mathcal{A} (for each $i = 1, \dots, n$).

¹ The array inputs (of an array \mathcal{A}) are simply all those states in \mathcal{A} which occur in the input subdivision of some function in \mathcal{A} , but not in the output subdivision of any function in \mathcal{A} .

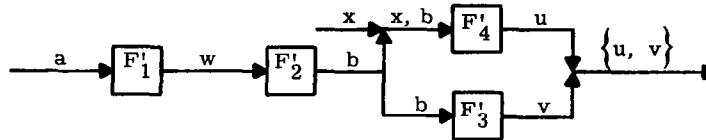
² Array outputs are defined in an analogous way to array inputs.

Intuitively, we may think of a partitioning of an array \mathcal{A} as being another array \mathcal{A}' , which is gotten by "cutting up" each function of \mathcal{A} into a bunch of functions. This cutting up process, however, must take care to preserve the states in the input and output of the function (although it is permissible to introduce new input states).

Example: — Let \mathcal{A} be the following array:



Then a partitioning of \mathcal{A} might be the following array \mathcal{A}' :



In the example, the component array $\mathcal{A}'_1 = F'_1, F'_2$ of \mathcal{A}' corresponds to the function F_1 , and the component array $\mathcal{A}'_2 = F'_3, F'_4$ corresponds to the function F_2 . The array inputs of \mathcal{A}'_1 are the same as the inputs of function F_1 (likewise for the outputs). The set of array inputs of \mathcal{A}'_2 contains the inputs of function F_2 , and the array output of \mathcal{A}'_2 is equal to the output of function F_2 . Therefore, \mathcal{A}' is indeed a partitioning of \mathcal{A} .

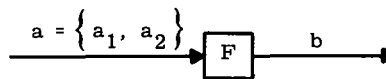
We shall want the probability tables for a partitioning \mathcal{A}' of \mathcal{A} , to be consistent with the probability tables for \mathcal{A} . We may ensure this consistency by placing conditions on the sample space for \mathcal{A}' , just as we placed conditions on the sample space of a "component array," when we defined it. We shall assume that such conditions are placed on the sample space of \mathcal{A}' as part of the definition of a partitioning. Thus for a partitioning \mathcal{A}' of \mathcal{A} , it will always be the case that the probability tables of \mathcal{A}' are consistent with those of \mathcal{A} .

Convention: — If \mathcal{A}' is a partitioning of \mathcal{A} , we shall sometimes use the phraseology: " \mathcal{A} is obtained from \mathcal{A}' by adding."

Three Special Types of Functions

There are three important types of functions which are distinguished by their probability tables. We shall adopt a special notation for those functions which employ these tables. This notation will be used in our diagrammatical representations of arrays in order to circumvent the necessity of writing out these tables whenever these functions occur in an array.

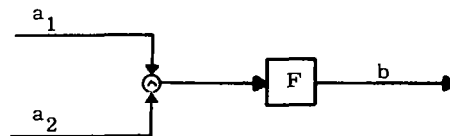
The AND Table: — Consider a function F :



If the probability table¹ for F is:

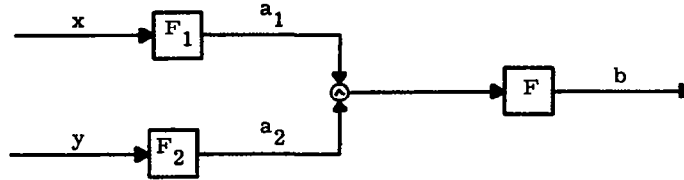
Given:	Probability of b
a_1 and a_2	P_F
a_1 and \dot{a}_2	0
\dot{a}_1 and a_2	0
\dot{a}_1 and \dot{a}_2	0

then the function F sometimes is denoted by:



¹ In this probability table, the labels a_1 and a_2 , are supposed to stand for $X_{a_1} \cap X_{a_2}$, and the label \dot{a}_1 and a_2 is supposed to stand for $(-X_{a_1}) \cap (X_{a_2})$. Likewise the label "probability of b" is supposed to stand for X_b . The use of these new labels above, make for a quicker interpretation of the entries in the table.

The symbol \wedge is called the AND symbol, and the table is called the AND table. Consider for example the array:



This array may seem to convey the idea that somehow the outputs of F_1 and F_2 are "joined" so that they can be fed into F . Actually, what this symbology tells us is that the input to F is precisely the state $\{a_1, a_2\}$, and the probability table for F is the AND table.¹ No "joining" function is implied.

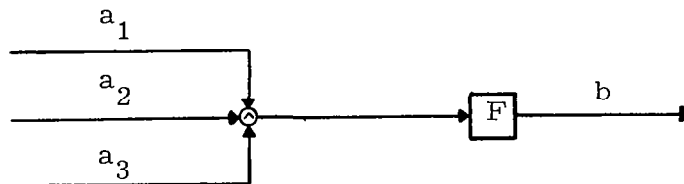
The Exclusive OR Table: — Consider the function F :



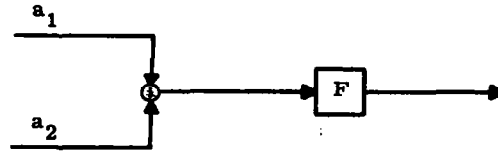
If the probability table this time is:

Given	Probability of b
a_1 and a_2	0
a_1 and \bar{a}_2	P_F
\bar{a}_1 and a_2	P_F
\bar{a}_1 and \bar{a}_2	0

¹ The AND table above is for an input state with two component states. The table may be easily generalized to input states with three or more component states. The notation would not change. Thus, in a pictorial representation, the AND table for three components would be implied by:



then the function F will be denoted by:

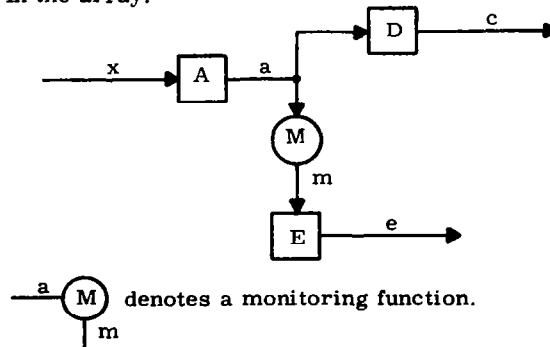


The symbol \oplus is called the exclusive OR symbol, and the table is called the exclusive OR table. This symbol tells us that the input to F is precisely $\{a_1, a_2\}$, and the probability table for F is the exclusive OR table.

The Monitoring Table: — Consider the function $C = (a, b)$. Suppose this function is accompanied with the following probability table:

	Probability of b
a	0
\bar{a}	P_C

When such a probability table is associated with (a, b) , then (a, b) is called a monitoring function. The table is called a monitoring table. In a pictorial representation of an array, a monitoring function will be denoted by a large circle. Thus in the array:



Roughly speaking, M responds when the output of A fails to occur.

III. USE OF THE CALCULUS

Thus far we have not employed the term system. That is because system is simply a term of convenience within the calculus. Its most frequent use will be outside of the calculus when making application of it. Indeed, the use of the term system in the title of this paper is an extramathematical usage.

Within the calculus it is useful to have a term to refer to the "most comprehensive" array that will be considered in a given discourse. Thus, suppose $\mathcal{S}_1, \dots, \mathcal{S}_n$ are all the arrays under consideration in a given discourse. A very typical situation is that one of these arrays (say \mathcal{S}_1) is a "parent" array to the others. By this we mean that the other arrays may be divided into two classes C and C' where each \mathcal{S}_i in C is a partitioning of \mathcal{S}_1 , and each \mathcal{S}_j in C' is either a component of \mathcal{S}_1 or a component of some \mathcal{S}_i in C. We say that this parent array is the most comprehensive of all the arrays under consideration. In a natural way we may associate a unique function (a, b) with this parent array. The input state a of this function is the earliest of the array inputs of \mathcal{S}_1 , and the output state b is the array output of \mathcal{S}_1 . We use the word system to refer both to the function (a, b) and to any partitioning of it.

System is used in engineering and in everyday communication with a wide variety of connotations. In fact, one might suspect that the number of connotations is somewhat greater than the number of users. In a situation in which the calculus is being applied, it is suggested that the use of the word system to refer to real world objects be restricted to application to a collection of objects that may be set in correspondence with the parent function in the discourse.

The calculus that we have presented, then, is intended to be useful for talking about real world systems. The systems to which it may be applied will be discrete systems or they will be systems which may reasonably be treated as discrete systems. Thus the calculus may be used only to talk about real world systems all of whose system outputs occur at the same time. Our principal interest is in development cycles.

In the introductory section of this paper, and in the above paragraph, we employed the term "development cycle" with the expectation that it would be understood in the everyday sense. Without a calculus, that is about the best that we can hope for. We are now in a position to identify more precisely how we wish to use the term.

A development cycle is a discrete system whose input is a Primitive Need Statement¹ and whose output is a collection of means for implementing an operational system² that will satisfy the need which occasioned the Primitive Need Statement. Not all processes in the real world that we would like to call development cycles are discrete, and thus not all development cycles in the everyday sense of the phrase match the definition above. However, we believe experience will show that virtually all development cycles (in the everyday sense) may be treated as discrete systems for the purpose of designing and controlling them.

Once a development cycle that is of practical interest has been described as a system by identifying its input and output states, we may employ the calculus as an aid to determining an appropriate basic strategy for carrying out the development process. Thus, a development cycle can be partitioned to define a prime function array³ in a manner that precisely identifies a chosen strategy.⁴ A strategy that is described in this manner can be presented for public inspection and, as the result of such inspection, may be corrected,

¹ By Primitive Need Statement, we mean the first verbalization which has the effect of calling attention to a real world problem which requires a new system for its solution (ref. 2 , page 18).

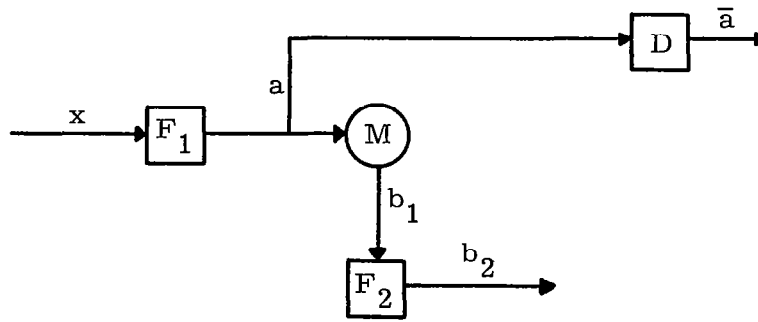
² A man-made system which is built to satisfy a need in another system is called an operational system.

³ A prime function array is an array in which the probability of the array output (given the first array input) is equal to the product of the output probabilities of all the component functions. Recall that an "array input" is a state in the array which does not occur as an output of any of the functions in that array. Recall also that states which occur at the same time may be considered as a single state or as multiple states, whichever is the most convenient.

⁴ By a strategy we mean here a sequence of steps for carrying out the development process which can be justified where justification is in terms of cost of

accepted, or rejected. An example of a basic strategy for carrying out the development of complex aerospace systems is presented in Part A of this report.

Not only does the calculus permit the precise description of a strategy in terms of the prime functions essential to the development cycle, but it also permits the elaboration of such a prime function array to provide for high probability of success of the development process being described. To achieve such a goal, we use monitoring functions to build additive loops. Thus, a monitoring function, M , is often used in an array as follows:

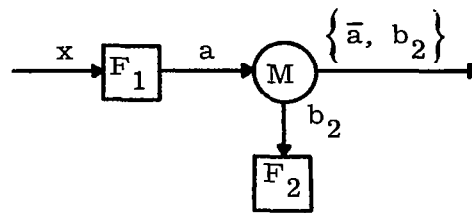


Here \bar{a} denotes the same state as a , except that $t_{\bar{a}} > t_a$ (see footnote 1). We interpret the purpose of the monitoring function in this array as follows: M responds only if the state a fails to occur. If a does not occur, M produces b_1 , which is an input to function F_2 which in turn produces output b_2 . The probability that \bar{a} or b_2 , or both occur, is greater than the probability that \bar{a} occurs, if P_M and P_{F_2} are both greater than zero. The role of M in the array is defined precisely only by the probability table associated with M .

development and in terms of the quality and cost of use of the system that is produced by the development cycle. See Report I, A Simple Model of a Man-Machine Development Cycle.

¹ The function D is inserted to preserve time relationships such that $t_{\bar{a}} = t_{b_2}$.

An abbreviated notation for the above array is sometimes employed:



In such an array, the set of functions $\{M, F_2\}$ is called a first-order additive loop.

By means of additive loops we can provide for "management" to ensure the high probability of the success of a development cycle. Chapter 5 of Part A suggests some of the ways in which this may be done.

By the manner in which the concept of function and array are defined in the calculus it is articulated with probability calculus such that one may readily derive a system description in terms of a probability equation given a system description in terms of the calculus. One may also readily translate a system description in terms of the calculus into a PERT description. In this translation, each function becomes a PERT activity, and time relationships are preserved.

Any attempt to recapitulate the rationale underlying the selection of the specific basic concepts and syntax which make up the calculus presented above would be both incomplete and tedious. It would be incomplete because much of the rationale is difficult to retrieve. The calculus was employed in its earliest form as an informal working tool; initially there was no intention to formalize it. It evolved as a working tool over several years as it was used to help solve a wide variety of system problems. By the time the decision was made to formalize it, the calculus was well shaped as an intuitive method and the many specific motives underlying it, like most evolutionary forces, were no longer identifiable. The best that can be done now is to test whether the calculus can be used to begin the task of describing the system development

process; one attempt is presented in Part A. Whether or not the adjustments made over the years of use have indeed generated a calculus that is broadly useful (or which is amenable to modifications so that it will be useful) can only be determined if others attempt to use it as an aid to solving real problems. No amount of rationalization will make it any better than it is.

Serendipity Associates

Chatsworth, California, October 1966.

REFERENCES

1. Air Force Systems Command: Configuration Management During Definition and Acquisition Phases. Air Force Systems Command Manual, Systems Management, AFSCM 375-1, 1 June 1964.
AFSCM 375-2 — System Program Management Surveys and Industrial Management Assistance Surveys. 25 June 1963.
AFSCM 375-3 — System Program Office Manual. 15 June 1964.
AFSCM 375-4 — Systems Program Management Manual. 16 March 1964.
AFSCM 375-5 — System Engineering Management Procedures. 14 December 1964.
AFSCM 375-6 — Development Engineering (DE). 14 August 1964.
2. Asimow, M.: Introduction to Design. Prentice-Hall, (Englewood Cliffs, New Jersey), 1962.
3. Department of Defense: Initiation of Engineering and Operational Systems Development. DOD Directive Number 3200.9, July 1, 1965.
4. Department of Defense and National Aeronautics and Space Administration: DOD and NASA Guide PERT COST. Joint Services, Office of the Secretary of Defense and National Aeronautics and Space Administration, June 1962.
5. Goode, Harry H. ; and Machol, Robert E.: System Engineering: An Introduction to the Design of Large-Scale Systems. McGraw-Hill Book Co., Inc., (New York), 1957.
6. Gosling, W.: The Design of Engineering Systems. John Wiley & Sons, Inc., (New York), 1962.
7. Whitehead, Alfred N.: Mathematics as an Element in the History of Thought. In The World of Mathematics, Vol. 1, James R. Newman, Simon and Schuster, (New York), 1956, pp. 402-416.