RC 6688 (#28797) 8/18/77
Engineering Technology 19 pages

Research Report

ON THE ARCHITECTURAL REQUIREMENTS OF AN ENGINEERED SYSTEM

N. P. Edwards

IBM T. J. Watson Research Center Yorktown Heights, New York

IBM Research Division
San Jose · Yorktown · Zurich

RC 6688 (#28797) 8/18/77
Engineering Technology 19 pages

ON THE ARCHITECTURAL REQUIREMENTS OF AN ENGINEERED SYSTEM

N. P. Edwards

IBM T. J. Watson Research Center Yorktown Heights, New York

The principles and practices of engineering provide for attributes of control, audit, and changeability. The architecture of such a system must be based on controlled, pre-tested modules and certain controlled interface protocols which permit configurability. This approach also provides the best means known to the author for the control of system complexity. These principles are examined and their application to information system design is proposed. Some areas requiring further development are discussed.

1. INTRODUCTION

Engineers have been building systems which were subject to audit for several thousand years. An engineer who contracts to design and execute a bridge or a space shuttle is held accountable for the quality of his work. NASA requires extensive technical accountability and auditability of its engineering contractors. The AN/FSQ-(SAGE) computer built by IBM in the 1950's for the continental air defense had an extensive system designed for control and audit of the system elements which contributed significantly to the reliability of the system. It also had extensive design features for recoverability. (Ref. 1) . In designing the control, audit and recovery systems for the project, we realized that we were re-inventing techniques in electronics which were state-of-the-art for mechanical engineers. These included such things as functional specifications with tolerance, tested, specified, re-usable modules with standard interfaces and control protocols, redundant function, built in sensing and testing means to name a few.

Since computer logic can be implemented (given enough money), in either hardware, microcode or software, it is logical that the techniques applicable to hardware are also applicable (although not always economical) to computer software and systems. This paper examines the principles and practices of engineering which have been used in some computer projects and which have a history dating back several thousand years. The application of these principles and practices to software and systems is then examined.

In the early days of engineering, in the days of ancient Greece, for instance, the engineering profession must have had great rewards, because the penalties for failure were drastic. Herodotus (2) tells of a bridge which Xerxes had built to span the Hellespont. "But no sooner had the strait been bridged than a great storm swept down and brake and scattered all that work." "When Xerxes heard of that, he was very angry..." and "...he commanded that the sea should be punished, and that they who had been overseers of the bridging of the Hellespont should be beheaded." A second effort was made. The bridge was made of 674 "fifty-oared ships and triremes anchored and held in place with two cables of flax and four of papyrus." This bridge held.

In more modern times, an engineer famous for other reasons put it, "The engineer simply cannot deny that he did it. If his works do not work, he is dammed..." (3) In spite of this, and the occasional dramatic failures of engineered works, engineers have persisted. Some have had genius, most have probably just been competent. Their works "work" because certain key principles and practices have evolved over the years which, when followed without compromise, provide for success.

This paper is based, in part, on some conclusions of a continuing project (see Acknowledgements) which is studying the application of these principles and practices to information systems. It is also based on a continuing study by the author of published results, and several IBM projects which observe these principles in varying degrees. No general purpose data processing system has been studied which has been found to observe these principles and practices to an extent which would justify using the term "engineered system". The conclusions presented are based on traditional hardware (bridges and buildings, as well as computers) engineering practices. The author's confidence that these principles and practices are applicable to data processing and information systems is derived from observed results of their partial application in systems currently under study.

Although the basic principles and practices of the engineering approach have been known and used for hundreds of years, they "have not been adequately articulated..." (4) and infrequently taught. Rather the knowledge and discipline have been acquired by observation and practice in laboratories rather than engineering courses which usually teach techniques and tools only.

2. PRINCIPLES OF AN ENGINEERING APPROACH

The basic principle is that the works of an engineer must work according to agreed to performance specifications. If the engineer is to continue to win contacts, his designs must be economical to implement, operate, and maintain. In addition, the works must be acceptably safe. The design engineer attempts to achieve these objectives by one basic principle: His design must contain no significant unknown factors.

Two well known examples of designs that failed this test are the Tacoma Narrows Bridge, "Galloping Gerty" and the Teton Dam. In the case of the Tacoma Narrows Bridge, the significant unknown factor was the behavior of the roadway in a 125 mile per hour cross wind. It "flew", but not well, and its uncontrolled oscillations broke the support cables and the bridge collapsed. In the Teton Dam, the unknown quantity was the porosity of the volcanic rock underlying the lake bed. Persons acquainted with that kind of rock know that it may contain voids or even tunnels hundreds of feet long, or even longer. There may have been some significant unknown in the rock structure, which the designers assumed was not significant.

A rather extensive search of the literature was conducted by P. O. Crawford, Jr. with participation by the author. Most of the publications found relate to specific techniques for design and implementation. Very limited material was found relating to the principles and practice which underlie successful project design and implementation. Most of this material found relating to the engineering of large systems was published in the late 1950's and early 1960's.

Very few recent publications on the subject were found in the Engineering Societies Library (New York, N.Y.). The principles are perhaps best summarized in the following extract from the definition of an engineer (5) developed by a committee of the council for Professional Development in 1949: "...An engineer is characterized by his ability to apply creatively scientific principles to design or develop structures, machines, apparatus, or manufacturing processes, or works utilizing them singly or in combination, or to construct or operate the same with full cognizance of their design; or to forecast their behavior under specific operating conditions; all as respects an intended function, economy of operation and safety to life and property." The key words are:

"works"
"forecast their behavior"
"intended function"
"economy"
"safety"
"apply scientific principles"

Another author states it as, "The aim of engineering is to create new equipment and systems which perform a desired task at minimum cost." (6) Thus, the principles and practices discussed apply to the design, implementation and operation of systems. Research, the quest for information, and development, the quest for a material, process or structure for a particular use, are not included.

2.1 Principles:

The following are believed to be the key principles of engineering design:

a. Scientific principles are applied creatively.

b. The works of an engineer perform to specification.

c. The performance specifications include the function required to be performed, the quality and amount of performance, and the safety requirements.

d. The performance must be achieved efficiently.

A requirement of (a) above, which will be discussed later is that there exists an underlying body of theory structuring the elements and processes used and their behaviors relating to their use in achieving the desired performance. This body of theory may or may not be formally understood.

3. PRACTICES OF ENGINEERING

Over the years, a set of practices has evolved which implement these principles. These practices are seldom articulated and not generally taught explicitly. Rather, they are acquired by exposure to expert practitioners, and by emulation of successful projects.

3.1 Predictable Performance Through Use of Known Elements

The works of an engineer should use only materials, components, processes and procedures which have been shown by tests to conform to their relevant performance specifications. The design relies only on characteristics which are specified, and the performance required is always within specified bounds. The system should include no significant unknowns. Even the inescapable uncertainty needs to be quantified. To the extent the engineer succeeds in the above, the works perform as designed. (Fig.1)

ENGINEERING DESIGN & EXECUTION

ACHIEVING:	SPECIFIED RESULTS
USING:	PARTS, ASSEMBLIES, MATERIALS OR KNOWN BEHAVIOR
COMBINED:	ACCORDING TO PROTOCOLS AND PROCESSES KNOWN TO PRODUCE PREDICTABLE RESULTS.

Fig. 1

3.2 Use of Common and Standard Design and Test Information

The engineer attempts to achieve his goal of efficient design and execution by utilizing past knowledge and experience in the form of standard specifications and data. Thus, his customer only pays a small (pro-rated) share of the cost of design and test of these component materials, processes and parts.

3.3 Use of Common and Standard Parts

Engineering, primarily through practice, formalized by standards organizations, has sought to classify engineering endeavors such that materials, components, structures and functions common to a class can be identified. This permits the establishment and promulgation of re-usable standard specifications and thus re-usable design information, parts, materials and structures. The attempt is to find "common factors" such that a large variety of structures with different performances can be constructed using mostly standard materials, components, and processes.

Engineering Principles: Techniques and Practices

Performance to Specification	Economy
Performance specified, pre-tested materials, parts, processes	Re-use of design information, (Standard Specifications)
Elemental Functions	Commonality of elemental functions

Common Set of Standard:
materials,
components,
processes,
elemental functions
construct functions

Requires

an underlying concept of structuring; a theory of structure; separation of significant variables.

Fig. 2

3.4 Separation of Variables of Use and Design

In structuring a particular system design, the engineer attempts to identify the system performance parameters which the user will want to treat as independent variables. He also attempts to identify the design factor he will want to treat independently. He then structures his design to preserve the independencies desired by the user as well as those he needs in his implementation and test. The engineered design "separates significant variables" for both implementation and use.

3.5 An Underlying Concept of Structure

Most fields of engineering have an underlying structural commonality such that pre-specified structures can be utilized extensively. System design with LSI circuit modules is an example, as is modular office space. Standard truss and girder designs for bridges and buildings are also examples. Engineering utilizes a stock of pre-specified structures in addition to making use of specified and tested materials, tools, processes and procedures which are available. (Fig.2)

The requirement that the design and execution be efficient, combined with the requirement that all materials, components, structures, procedures and processes have known performance leads to the use of previously designed and tested elements. Thus, there is a continual search for underlying structure, classes and commonality of structural elements. Without these, a practical set of specifications (as for bearings, shafts, fasteners, resistors, valves and switches) would not be possible. Every design engineer would be required to create these specifications for each design, and would also have to verify their correctness.

3.6 Valid Standards Exist

Standards are established and promulgated by recognized standardization bodies, in accordance with established procedures. Good engineering practice requires that any standard have a designated control authority which asures the validity of the standard.

3.7 Engineering Design Proceeds from the User Requirement for the Performance of a Specified Function

"The specification must clearly state the result desired..." (7) The basic system specification is the specification of function. "The first element is function Every system must have a function and if there is one mistake made in system design efforts today, it is to forget the fact that a system must have a function. The function is the mission, aim or purpose of a system. What is the system supposed to be accomplishing, not how it is to be accomplished." (8)

4. SYSTEM ARCHITECTURE AND AN "ARCHITECTURAL SYSTEM"

For the purposes of this paper, the following definition of "system" is used: "A system is an aggregrate of components, sometimes including men and even animals, which act together to perform a desired function." (6) Thus, a system architecture is a particular structure. A building is a particular architecture, and may be in one of a variety of architectural styles, implemented in a variety of architectural systems. Margiloff (9) describes the different kinds of building systems which were developed in the 19th century, including the frame house, curtain wall construction (using cast iron panels), girder sections, and reinforced concrete. An architectural system includes a set of related materials, processes and use protocols which are known to produce predictable results, when used in accordance with good practice.

An architect or engineer, designing and implementing a building or bridge using a particular architectural system is relying on a set of elemental parts which have performance specifica-

tions relevant to the class of use. He procures the parts to these specifications and assures that the contractor uses them for their intended purpose, in accordance with specifications. His contribution is the originality of the particular architecture he designs using the known elements, assembly techniques and procedures. If he needs a new material or process, an engineering development is contracted. The result is used only if the development was successful, that is, the result has been proven to conform to a performance specification.

5. PREDICTABILITY OF THE RESULT

The task of the designer is to proceed from the specification of the desired results to the specifications of the component parts and their interactions. Clearly, if the designer is to be able to predict the behavior of the final design, the result, he must not incorprate any significant "unknowns"in the design. Even the inescapable uncertainties of the real materials must be considered in the specifications and the tests of the component parts. If he has done this, then it should be possible to predict the functional behavior of the implemented design. Although this practice was followed in the early 1950's on the SAGE computer and is required by some contracts, only one concise statement of it was found in recently published engineering literature. M. P. O'Brien (6) says:

"Considering the aggregate of components which act together to perform a desired function, one may analyze the performance of the resulting system by considering only the interactions of the input-output relationships of the components without reference to the internal mechanisms of each component. One links together "black boxes" of known input-output characteristics including in some systems black boxes which measure the output, compare it with a desired value, and send back a command to alter the input. By analysis of the interrelationships between components, one can predict the input-output characteristics of the system as a whole; at least one can do so in principle if all the pertinent characteristics of the components are accurately known." It also seems reasonable to add that the only alternative to this is to build the system and test its output/input transformation to the required level of exhaustion. If the results are not predictable in advance to a very high degree the number of such design and test ("cut and try") cycles required to achieve the specified results could be enormous.

N. H. Taylor (1) described the techniques used in the IBM built SAGE computer, in a paper that won a prize as the best reliability paper of that year (1957). In a complex system, where exhaustive testing on an input-output basis is not feasible, performance can only be predicted or known to the extent that the individual components are known to conform to relevant functional performance specifications, and are known to be used in accordance with specification. Only specified characteristics of specified parts are used, in conformance with specification.

6. CONFIGURABILITY, FLEXIBILITY AND MAINTAINABILITY

Many different variations on the concept of a system architecture composed of functionally specified, pre-tested parts are possible. While the variations can compose a practically continuous spectrum, they are considerd in four classes, for convenience.

6.1 A "fixed architecture".

The component parts which comprise any particular system architecture can be provided with functional specifications, and tested for conformance to specification. No attempt to standardize on common mating faces (interface standards) is made, and thus the basic architecture is fixed. No component part will fit in any other location in the structure. However, if each component part has an implementation independent functional specification, which includes the specification of its mating faces and environment, it can be implemented in any feasible

fashion, tested independently and repaired outside the system. Concurrent design is also possible, providing all funcional specifications and mating face specifications are determined in advance.

6.2 "Customizable architecture".

This is analogous to the customers present view of a mass produced automobile. The basic architecture is fixed, however, you can have a variety of optional features added or substituted. The various optional features are specified, as are their attachment to the system. The mating faces are specified. However, there is still no attempt to standardize on common mating faces.

6.3 "Configurable architecture".

Highway bridges are made with standard girders and forms, AMTRAC stations use standard pre-cast concrete forms for platforms. An automobile manufacturer has interchangeable engines, transmissions, body shells and countless other parts out of which a knowledgeable person can compose orders for a wide variety of different cars. Basic parts such as screw threads, ball and roller bearings, tire sizes and the like were standardized to provide interchangeability during World War I.

A wide variety of individual architectures can be made using a configurable architecture system, and if the rules are followed rigorously, the resulting architecture will "work". That is, it will do what that particular structure should, based on the functional performance of its individual parts, and their interconnections. It may not be the architecture you expected if the design process produced the wrong algorithm or construct, but it will be a "correct" member of the allowable set of architectures of that system. For example, integrated circuit modules are available today which perform complex logical and arithmetic functions, storage, signal processing, signal conversion and the like. These are supplied in "families" with a common set of mating faces, from which you can compose any arbitrary function within the capability of that family. If you have followed the interconnection rules exactly, any algorithm you construct will "work". It may not be the one you want, but it will work. It is not the fault of the architectural system if you happen to build the wrong particular architecture.

Each of these architectural approaches has its own degree of flexibility, ranging from no flexibility to a wide variey of possible configurations. The maintainability of the three systems differs in complexity and in the nature of the test equipment required. In each class of architectural system the individual elements which have functional specifications may be simple or complex. However, in a configurable architecture, the existence of a common set of mating faces means that the inputs needed for testing a system element can be provided by a simpler piece of equipment, as it does not have to cope with different input and output faces for every system component. In a well designed configurable architectural system, there should be multiple occurrence of parts, which further simplifies the test equipment and reduces the number of spare parts which must be stocked. The "eighty-twenty" rule is well known. Typically, eighty percent of the system comprises 20 percent of the part types, the remaining 20 percent of the machine is largely special and has 80 percent of the part types.

7. EFFICIENT REALIZATION OF SYSTEMS

The determination of what is "efficient" depends heavily on the nature of the particular architecture or set of architectures to be realized, and their use. The basic trade-off is between design and test cost on one hand, and production or replication and use costs on the other. Thus, "efficient" may mean the least cost to design, the least cost to produce, or the least cost to use (including maintenance), or any weighted combination of these. The trade-off appears in the amount of time and effort that is appropriately spent on the design and testing

of elements and structures particular to the application, as opposed to using "off-the-shelf" designs and elements. In the absence of a configurable architectural system with its "off-the-shelf" system elements, the designer is compelled to do a complete design and test of each elemental part, whether to do so is economical or not. Only if there is a configurable architectural system with its stock of pre-specified, tested parts, does the designer have a choice.

8. REQUIREMENTS FOR A CONFIGURABLE ARCHITECTURAL SYSTEM

8.1 Elemental functions which recur are necessary.

The essential requirement for a configurable architectural system is an underlying concept structuring the architectures which are within the scope of the system. The architectures to be constructed must be factorable into a largely common set of elemental functions, with a common set of interconnection and use protocols. Thus, the functions performed by the systems to be architected must be largely composable out of a common set of sub-functions. In construction of buildings and bridges there are tension, compression and fastening elements. In digital logic there are gates, registers, drivers, arithmetic elements, code translaters and the like. Any set of functional elements can be configurable, reusable or "common", only if the funcions they implement recur in systems. An appropriate classification scheme, based on the underlying structure permits recognition and cataloging of common functions.

8.2 Specifications of functional elements include function and performance.

Since the designer is required to provide a design which satisfies specified functional performance, he must know and be able to rely on the functional performance of the sub-functions from which he designs his system.

8.3 The functional performance of any "configurable" or reusable element must be independent of its embedment in any particular architecture (10).

If its function is dependent on the structure of the particular architecture, the functional specification of each element would have to consider all possible architectures which might use that element. This is impractical.

8.4 The mating faces of the elements of the configurable system must be specified.

The number of different faces allowed must be small. Adapters which permit functional elements with different faces to be mated can exist, but the number increases rapidly with the increase in number of types of mating faces.

8.5 The elements and their applications must be controlled.

Management must assure that elements and their use conform to specification. A correct set of functionally specified elements and interconnecton protocols is not sufficient to assure that a system using these elements will perform to specification. There must be adequate management controls to assure that the elements themselves perform to specification, and there must be management controls to assure that the use of the elements conform to the use protocols. Incorrect use of a system element is tantamount to use of an unspecified and untested element.

9. ARCHITECTURAL SYSTEM REQUIREMENTS FOR DATA PROCESSING

The idea of applying engineering principles and practices to data processing systems and programming has been around since hardware engineers began to do programming in the 1950's. With one exception, I am aware of no example of a software system which exhibits most, if not all of the essentials of the engineering approach. This one exception is the Advanced Modular Programming System "AMPS" invented by J. Paul Morrison of IBM (11). It is especially notable in that it uses fully independent modules and configurable architecture principles. The need for time buffering between functional units or modules was first brought to my attention by Morrison's work. Myers (12) has examined various levels of module "strength" and "independence", as have Parnas (13) and Kerrighan and Plauger, (14) among others. However, the need for complete independence of individual modules from the program in which they are used does not appear in the general literature. Kerrighan and Plauger say, for example, "...so that no one module has to know much of the total problem, nor deal with more than a handful of immediate neighbors." and, "...the modules are kept as uncoupled as possible, and the coupling that exists is kept visible." In a fully configurable architectural system using "controlled modules", the modules know nothing about the total problem, only about what they do (in any problem), and they know nothing about their neighbors, only that they are a part of an input-output compatible family. These characteristics are discussed in more detail below and in reference 11.

9.1 Function is specified in terms of data state transformations.

The "end user", for whom a data processing system exists, wants to receive the information he wants, where, when and in the form he specifies, with known quality levels including security, privacy, reliability and accuracy. (Fig. 3). The function of the data processing system is to retrieve, derive, assemble, format and present the required information using its internal functions applied to the available data. Any function can be thought of as a set of processes which produce state changes in the things processed. E. V. Kirk, in An Introduction to Engineering and Engineering Design (15) states "A problem arises from the desire to achieve a transformation from one state of affairs to another." Petersen (16) describes an approach to "Data State Design" for characterizing the flow of data through processes in a system. The resulting diagrams look very similar to the "PERT" diagrams used to plan and record the flow of system parts thru the processes which produce a finished system.

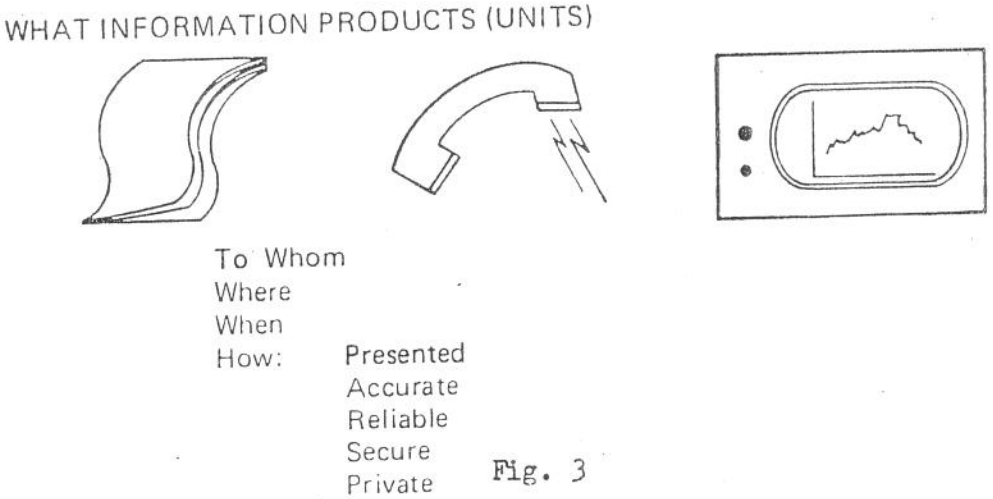

Fig. 3

9.2 Function is independent of structures (programs or systems) which use it. (Fig. 4)

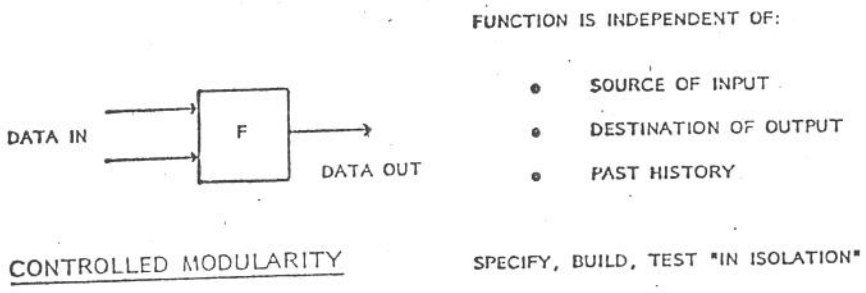

Fig. 4

a. Module function must be independent of the source of its input. This is so because the source of the input is a function of particular programs, not of a reusable module. Four logical options exist which can satisfy this requirement. (Fig. 5)

Input Source	Input: Indirect	Input: Direct
Fixed	A	B
Variable	C	D

Fig. 5

In the case of a fixed input source, the module knows implicitly where to find its input. If the address is indirect, the system provides the real address. No address information needs to be loaded or complied into the module for any use.

In the case of variable input source, the module must be "told" where to find its input and must be supplied this information, for each use (but not for each instance of each use).

"Binding" of addresses can be either early or at run time, depending on the individual choice. If early binding is used, a record must be kept of the address associated with each module to permit changes.

b. Module function must be independent of the destination of its output. A generally useful module cannot know which modules are connected to it as this is a function of the programs that use it. The options are the same as for the input source, (Requirement a).

9.3 Module function must be independent of its past use.

The initial state of the module must be known. If its function is dependent on past use, the initial state of the module is probably unknown to other users.

Corollary 1. A module can only have three states; "Ready", "Busy", or "Inoperative". It is ready to do its function; it is doing its function or it is in trouble. Since each use may want different parameter values in the module, it is essential that no unknown changes of module state or unknown parameter values be left in the module.

Corollary 2. Control must be external. If this is not so, the module must have been previously instructed as to what to do. This means that its function is in part, at least, dependent on its past.

Option a) Appearance of data at the module input turns it on; absence of data turns it off.

Option b) A separate, external on/off control, independent of the data is provided.

Corollary 3. Any parameter in the module must be genuinely constant, i.e., not changed for the life of the module, or it must be entered with each use (but not each instance of a particular use). An external record of the location and value of entered constants must be maintained. The options for entry of constants are the same as "Data in", with the additional option that the constants may be entered by a separate port on the module.

9.4 The data received by the module (from preceding modules) must be appropriate for the receiving module.

Some options are:

Option 1) There is only one fixed data type and format for all modules in the set.

Option 2) All modules of the set have the ability to accept a variety of data types and formats. The modules are instructed as to the type and format of the data (by an external control).

Option 3) Translator modules are provided and instructed by control to make the needed transformations of the data type and format or to signal an error.

The design of a proper set of interconnection protocols depends on the purpose of the architectural system and the particular design approach chosen. A detailed discussion is beyond the scope of this paper; however, some of the factors which have been considered (17) include the code, syntax, symbols, vocabulary, units (used for values of attributes and parameters) and semantic considerations. For example, the input might be in the correct code, format, have the correct vocabulary and syntax, but be about the wrong item. Depending on the particular architectural system chosen, the system might be able to provide protection against errors in any or none of these factors.

9.5 Time synchronization is required.

The results of different functional elements may be required as input to any element. Also, as the performance rate or delay of a functional element may be dependent on the data input or parameters entered, it is necessary to provide means for synchronization. Queues or bins are used in other physical systems; in computers and communications, registers and buffers perform this function. Thus, any general purpose, configurable system architecture requires queues between functions to adjust for different processing rates of the various units.(Fig. 6)

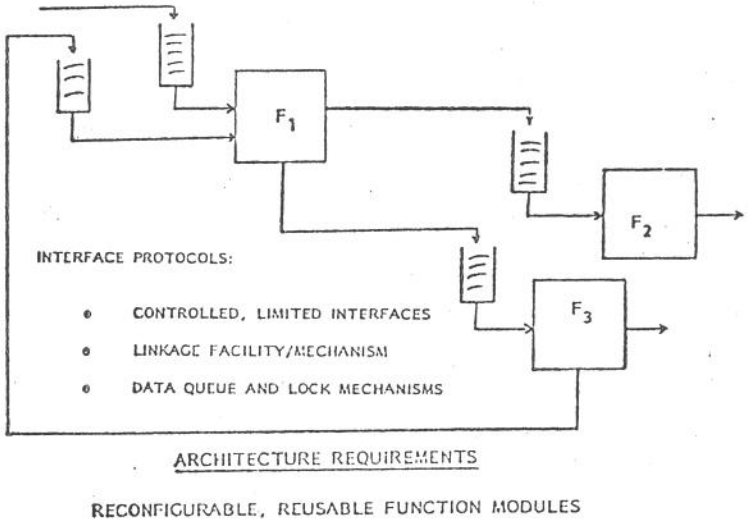

Fig. 6

Particular requirements for queues of given lengths are functions of the particular programs, the available resources, and the system resource manager. Thus, the queue requirements must be unknown to the modules. A module simply receives data at its input. A queue of data is merely data in a "wait" state. Thus, the control of queues is a function of the system resource manager, as data is one of the system resources. A means to connect modules to form programs and to assign data is needed.

9.6 There must exist a linkage facility separate from the modules it invokes.

This facility invokes the modules and assigns data in accordance with the program requirements. It "binds" the modules to modules and data to modules to accomplish a specified result. As mentioned earlier, the binding can be to copies of the modules (and recorded) before run time, or can be at run time. Three options are:

Option a. Early "binding" of real addresses and data. The Linker reads the program, copies modules and links real data to the module copies and makes a load module.

Option b. Early "binding" of virtual addresses. The linker reads the program, enters virtual address of modules and data into modules and creates a load module.

Option c. The linker reads the module sequence, table of data assignments and conditional control at run time and calls the data and modules as needed. It pararmeterizes the modules, assigns the data and runs modules.

9.7 Resource management is separate.

Resource management occurs at several levels of the system. The department manager (of the DP installation) decides what resources to provide, and their nature. This includes decisions as to which functions will be in hardware, microcode, software, or performed by people on or off site. In the automated portion of the system, the primary resources are the hardware, software and the data. Since multiple processes may be running concurrently, the resource managers must be independent of any particular user program on the system. Since the data may be used in a variety of processes, the management of the data resource must be independent of both the individual using programs and the physical resource management.

9.8 Documentation is in accordance with good engineering practices.

Engineering design and implementation practices adequate documentation to provide for producibility, control, auditability and maintainability. (Fig. 7)

DOCUMENTATION - PARTS AND ASSEMBLIES

SINGLE USE (To repair, replace, upgrade)

- PART Specifications:

  Function, Fit
  Environment
  Make - Procure (Procedural)

- ASSEMBLY (ALL OF THE ABOVE)

  Except: Procedural Spec

  = BILL OF MATERIALS

  ASSEMBLY INSTRUCTIONS

MULTI USE PART OR ASSEMBLY

ALL OF THE ABOVE, AND CONTROL POINTS FOR:

SPECIFICATIONS, PROVISION, USE,
(supply)
RECORDS - "WHERE USED", WHERE IS,
CHANGE LEVEL

Fig. 7

a) Producibility: The specifications contain the necessary information for the production of a specified item, and for the verification that the item meets specification. "A specification is a description of an article or a method so complete that it can be bought or built by others to the complete satisfaction of all concerned." (19) Specifications are of two classes: performance specifications which are a formal description of what is to be performed, (function, rates, capacities, timings, qualities); and structure specifications which describe any sort of structure; (organization, thing, process). They provide the information needed to supply an article with a specified performance.

b) Control: Information is maintained regarding who has the responsibility and control of: content of specifications, items produced to specifications, use of items in systems, and the specifications themselves.

c) Auditability is provided by the records of assigned responsibilities and the "where used" files, bills of materials and change level records, as well as by the specifications themselves.

d) Maintainability is facilitated because the functional performance and structure of every part of the system is specified and records exist which identify each part and its documentation. Thus, any functional unit in a specified system can be identified, removed, repaired, the unit tested and replaced in the system. If desired, the functional unit can be replaced with another unit with the same function and performance, but implemented in a completely different way or technology.

9.9 Concepts of structure are needed.

A structuring is proposed which includes three general classes of functions which can be considered as candidates for implementation as "controlled modules". These are:

a. Internal System Functions,

b. Application Independent Modules (AIM), (20)

c. Generalized Application Modules (GAM).

9.9.1 Internal Function Modules.

These include all the system functions which need not be accessible to the end user. Examples of sub-classes are:

a. Resource managers, including data management, security, custody, job schedulers, file management, network control.

b. Data mappers. These map data among families of modules with different data representation.

c System state monitors,

d. System use monitors,

e. Data use monitors, and

f. Editors.

This class of function was proposed by M. J. Marcus in about 1972 as "Implementation Independent Internal Functions," (I^3 F).

9.9.2 Application Independent Modules (AIM)

These are the basic functions the user needs to compose and execute a job. They include the following classes and are discussed in detail in reference 20.

a. Transfer. This class of function permits the user to sign on, enter data or request data from the system. This is accomplished by specifying the data (Fig. 4) and calling the data by name. (18)

b. Convert. This class changes the expression or representation of the data without changing the data meaning.

c. Derive. This class includes the classic computation functions. Data is derived from other data in the system.

d. Sequence. Sort and merge fall in this class. Neither the meaning nor the expression of the data is changed.

e. Logic. This includes conditional and unconditional operations.

9.9.3 Generalized Application Modules.

This class is potentially an extensive one. Work has been done by P. O. Crawford, Jr. (21) and D. C. Burnstine (22). Burnstine has focussed on the structure of business and Crawford has emphasized the structure of information used to manage and operate the business functions. Included are the functions relating to:

a. Registers (Chronology of membership, inventory),

b. Rosters,

c. Journals,

d. Transaction reports

e. Accounts, summaries, analyses, and

f. Specifications.

9.10 A common means of specifying information is needed.

This is fundamental to the approach to "Engineered Systems". Any formal discipline or field of endeavor requires a formal means of communication. The engineering approach could not exist without standards for blueprints, drawings and specifications. "Software engineering" and "engineered" data processing and information systems cannot exist without a commonly accepted, formal means of specifying the results to be produced; the data input, the system function. Without such common specification means, it is not possible to provide the end users with common functional modules which have functional performance specifications and have been tested for conformance to those specifications. Without the common specification means it is not possible for the user or implementer of information systms to take advantage of commonality which exists. Finally, accountability cannot exist for unspecified items. Without a class specification, class membership cannot exist. Most accounts require the concept of class membership.

10. CONCLUSIONS AND SUMMARY

The principles and practices of good engineering design and implementation provide essential ingredients for controllability, auditability, recoverability and for changeability of complex systems. They also provide ingredients essential to the control and audit of the system contents; in this case, the data in the system. That this is so can be summarized as follows.

10.1 Control and auditability.

A system whose contents are specified and under strict accountability provides for requirements essential to accountability, control, and protection.

Management control requires accountability. Accounts are based on specified classes. Unless there are classes (tanks, troops, guns) each item must be specified individually. Specifications are necessary so that class membership can be established. Audits are of account records and existing inventories (memberships), and transactions against those inventories. A proper concept of the nature of the structure of the system and the information in the system is required for proper class specification. The principles of engineering specification must be followed. No generally accepted classification approach for logical data structures now exists.

10.2 Recovery and changeability.

Recovery, relating to either the system itself, or the information in the system requires that functional specifications exist for all system elements and the system itself; that the concepts structuring the system provide the needed separation of variables (controlled modules) in the system function, and that the system architecture obeys the principles of "configurable architecture". These characteristics also provide for changeability. System recovery and changeability are essentially the same. In recovery, the desire is to get from an unwanted state (failure) to a wanted state (working as before). In change, the desire is to go from the present (unwanted) state to a new state of the system. A system composed of specified functional parts, obeying the laws of configurability, can be repaired and modified efficiently.

10.3 Areas requiring development.

The two major areas require development before the principles and practices of engineering can be applied fully to information system design. These are:

a. Establishment of generally useful classification schemes for information system elements and the information itself.

b. Establishment of generally useful specification means for specifying end user information with sufficient precision and accuracy so that the specification is the basis for the system to provide that information requested by the user.

To reiterate:

-- Improved recognition of common user functions cannot occur without improved concepts of classification of these functions so that the underlying commonalities can be recognized.

-- Specification of these functions in an implementation independent way is essential to reusability.

-- Accountability of the kind needed cannot exist without adequate classification and specifications.

ACKNOWLEDGEMENTS:

Much of the material in this paper was developed in meetings and discussions with my co-workers on the project, Perry Crawford, Jr. and Donald H. Myers. I wish to especially note that Perry was primarily responsible for excellence of the literature search and provided several of the key references. Conversations with John Crane and Raoul deCampo have also contributed to shaping my thinking and identifying key concepts of the engineering approach. E. C. Lamb, R. C. Kendall and Y. Tarnawsky have also been most helpful.

REFERENCES:

1. Taylor, N. H., "Designing for Reliability," IRE, June 1957.

2. Herodotus, Vol. III, Loeb Classical Library.

3. Hoover, Herbert; Memoirs of Herbert Hoover, Vol. I. Years of Adventure, The Macmillan Co., New York, 1951, quoted in Whinnery, J. R., The World of Engineering, McGraw-Hill, 1965.

4. Sporn, Philip, Foundations of Engineering, Macilllan Co., 1964.

5. Encyclopedia Britannica, 14th Edition, 1962.

6. O'Brien, M. P., "The Engineering of Large Systems," Ch. 9 of Listen to Leaders in Engineering, David McKay Co., 1965, (Edited by Childers and Love).

7. Jessup & Jessup, Law and Specifications for Engineers and Scientists.

8. Nadler, Gerald, "Engineering Research and Design in Socio-Economic Systems," Engineering: A Look Inward, a Reach Outward, U. of Wisc., 1967.

9. Margaloff, Irwin E., "When Technology Falters," -- an address to the 142nd Annual Meeting of the American Institute of the City of New York, Feb. 4, 1970.

10. Edwards, N. P., "The Effect of Certain Modular Design Principles on Testability," Proceedings, International Conference on Reliable Software, IEEE Press, April, 1975.

11. Morrison, J. P., "Data Responsive Modular, Interleaved Task Programming System," IBM Tech. Disc. Bull., Vol. 13, No. 8, Jan. 1971.

12. Myers, Glenford, J., Reliable Software Through Composite Design, Petrocelli, N. Y., 1975.

13. Parnas, D. L., "On the Criteria to be Used in Decomposing Systems into Modules," Carnegie-Mellon Univ. Aug., 1971.

14. Kernighan, B. W., Plauger, P. J., Software Tools, Addison-Wesley, 1976.

15. Kirk, E. V., An Introduction to Engineering Design, John Wiley & Sons, 1956.

16. Petersen, J. E., "Data State Design," Proceedings Compcon '76.

17. deCampo, R., Internal IBM memos and reports.

18. Edwards, N. P., Tellier, H., "A Look at Characterizing the Design of Information Systems," Proceedings, ACM, Nov., 1974.

19. Ayers, Chesley, P. E., Specifications: For Architecture, Engineering and Construction, McGraw-Hill.

20. Edwards, N. P., Lamb, E. C., "On Classification of Application Independent Modules," (to be published).

21. Crawford, P. O., Jr., Internal IBM memos and presentations.

22. Bernstine, D. C., "BIAIT", presentation to Application Systems Methodology Project, GUIDE 40, May, 1975, by Alistaire Paterson, and Internal IBM memos.