Techniques for requirements-oriented design

by KENNETH J. THURBER
Sperry Univac Defense Systems Division
St. Paul, Minnesota
and
University of Minnesota
Minneapolis, Minnesota

ABSTRACT

The purpose of this paper is to discuss how requirement studies can be performed for computer systems. There are many different types of requirement studies that can be used to define a system and a number of these are described.

INTRODUCTION

The architectural process and its relationship to requirement studies

This paper is about the determination of requirements for the architecture of computing systems. Webster defines architecture as follows:

architecture: the art or science of building; specifically: the art or practice of designing and building structures.

As we proceed, we will find the connotation of architecture as art to be particularly appropriate in the context of computing systems and their requirements. While there certainly is a large and rapidly growing body of knowledge concerning the engineering aspects of computing system design, there has been little real emphasis on techniques for determining the requirements to be used as the basis for the design of computing systems.

In the current context, architecture includes additional activities to those delineated in the Webster definition. In particular, the design activity must include requirements analysis and specification as an integral part. In fact, in many computing system applications, initial marketing activity involves creation of customer demand as a first and often overlooked step of the complete architectural process. Of course no one of the integral activities of architecture stands alone. The requirements analysis is closely tied to the design which is iteratively associated with logic design, and software design and implementation. All these activities are associated with evaluation which must take place within each activity and between activities. Consider, for example, the restricted architectural problem of exactly duplicating a competitor's computing system. Presumably a marketing analysis entered into the decision to duplicate and the requirements specification is largely provided by the complete specification of the extant system. Other requirements include projected costs and additional features. The design activity here could concentrate on efficient implementation, and some evaluation activity could be required to determine how accurately implementation costs matched those specified as the target during the requirements analysis. Implementation of the copy system (which could proceed from the design and detailed evaluation) could be required to determine that the implemented system met the specified requirements (both technical and financial). Typical architectural activities include: requirements synthesis and analysis, requirements specification, configuration and subsystem design, detailed design, implementation, and evaluation. Trade-offs cross hardware, software, and firmware boundaries at each level and the design process will proceed iteratively. Some decisions that are made are simply a matter of style. If the designer does not have a total feel for the complete architectural problem there may be no way to show that a given proposed feature is worth the cost of implementation. Based upon the designer's experience the resultant system style will be developed.

Systems

In a paper about systems it is important to discuss what constitutes a system and how the view of a system can be different depending upon whether we are the user or designer of the system. There are no real, precise definitions of what constitutes a system. Companies that sell memories speak of memory systems; mainframe manufacturers speak of computer systems; software manufacturers speak of software systems; etc. Notably, the word "sys-
tem” seems to mean the end product produced by the manufacturer.

We will define a system to be a hierarchical, dynamic collection of hardware and software entities. A system is composed of an application-defined environment together with a set of software and hardware that hosts the application. The application environment delimits and specifies the system.

The software and hardware which supports the system has hierarchical, dynamic relationships, which together form the basis to support the application. The hardware and software are hierarchical because of the identifiable levels; i.e., microcode, register level, CPU level, etc. They are dynamic in that in processing the application, the various levels interact to support the application.

There are many different views that can be taken of a system. We will briefly discuss several major views of a system. First, there is the view we have of a multi-end user system. An example of such a system is a computer utility. In this case, we see general-purpose hardware and general-purpose software. Another system type is the single-end user system. Examples of such systems are integrated command and control systems, and single-owner computers centers. Such systems have general-purpose hardware and specific software tailored to the system’s function. A third view of a system is that of a single-owner system. In such systems, we see specific hardware and software. The last view of a system is that of the designer. In the past, since designers were primarily hardware designers only, the designer saw specific hardware and either specific, general, or no software. The usual case was no software. In the future, it is important that computer architects not be solely hardware designers, but computing system architects capable of making realistic, relevant and user-responsive global system design trade-offs. Only then will systems be more responsive to the user and customer.

SYSTEM REQUIREMENTS

The system design process

In designing a system, we must satisfy a need. This need must reflect the environment as well as the objectives for the system. For example, a system designed for a deep space environment would include consideration of very stringent environmental and reliability constraints on the entire system. The design process starts with determination of requirements; that is, the user’s needs. From these needs, the design is specified. It is the extension of this need that makes a market for a product. The user’s needs may be broken into two categories. These categories are discussed further under the headings of Requirements and Attributes.

Requirements are the constraints which the system must satisfy. The requirements specify what the system must do. That is, any system concept which meets the requirement is a candidate solution to the customer’s problem.

Attributes, on the other hand, specify either options or evaluation criteria for qualitative comparisons of competing systems that meet the system requirements. There may be many concepts which satisfy the architectural requirements. Attributes may be used to evaluate the competing architectures to obtain a feel for the “goodness” of the architecture in solving the customer’s problem or as a set of factors used to optimize system designs. Attributes may also specify options which the user desires but does not necessarily demand.

The problem statement

In designing a system, the customer’s needs must first be determined. This involves a problem statement. Usually we will not be given a specific set of requirements. Rather, we will be given a statement of the user’s problem in general terms. This problem statement describes the user’s need. It is from this basic need and the concurrent design constraints based upon industrial affiliation that the specified system will be derived.

Requirements

Requirements are constraints placed upon a system concept. Requirements are the “musts” that any candidate system shall satisfy in order to be a viable, potential solution to the user’s problem. However, depending on the level of definition of the requirements, there may be many systems which meet the requirements but are not acceptable. For example, if the sole requirement were to meet a certain throughput rate, there would be many architectures which could meet the required throughput. However, due to connectivity of processing elements, basic structure, or machine repertoire, a large number of proposed concepts may not actually be applicable. It is extremely important that the requirements be very carefully defined.

The source of requirements is a problem definition. It is from the translation of the customer’s functional problem definition that the requirements are derived. There are typically two types of requirement studies that are performed. These are requirements analysis and requirements synthesis. Requirements analysis typically is involved with special purpose systems or market analysis after the fact. The requirements of a system or an application are typically analyzed in terms of what is known about the problem a priori from previous solutions to describe the problem environment and obtain a feel for how new technology could better be applied to the problem. Requirement synthesis, on the other hand, is involved with trying to project what types of systems will be useful in new application environments; i.e., to synthesize or create problems that

* Bell calls requirements “wants.”

** Bell calls attributes “objectives” and describes their evaluation as a relative maximization or minimization process.
could be solved with new technologies, or define areas in which new technologies will allow a more cost-effective solution to the created problems. As an example of the requirements analysis, the AMNCS (Advanced Multiplatform Naval Computer Study) is a classic example of a functional requirements analysis of current Navy problems. The AMNCS analysis tried to determine what Navy tactical requirements were established in the past and how new technology may be applied to those functions. The Hewlett-Packard hand-held calculators, are a classic example of requirement synthesis. In the Hewlett-Packard example the designers asked, "If we were able to build such a device, would a market develop?" Obviously, the market did develop since large numbers of people own pocket calculators.

There are numerous varieties of requirements. Some of these may be generally categorized as marketing requirements, economic requirements, technical requirements, and political requirements.

Examples of marketing requirements may be that the system being designed must be capable of solving a certain list of designated problems. Other marketing requirements may include price goals, goals specifying the production cycle, product life goals, and logistics goals.

Typical economic requirements deal with the financial constraints imposed on the system developer. Examples of economic requirements may be that the non-recurring development costs do not exceed a certain dollar figure; that the spares cost for repair does not exceed a certain figure; and that the manufacturer has been in business a specified number of years.

Political requirements are viewed mainly by the designer of systems that require the use of off-the-shelf equipment and who may be trying to use piece parts that are available through his own company's manufacture versus piece parts that are available from other vendors. In this case, a typical requirement may be that we use our own microprocessor if we're a semiconductor vendor.

At best, the political, economic, and marketing requirements may be extremely vague. In a real sense, however, technical requirements may be made very precise. The types of items that may be considered for technical requirements include system organization, word/byte/bit organization, hardware and software expansion capability, data path widths, word size, memory hierarchy and memory sizes, availability of software, availability and capability of the operating system, assembly and compilation speeds, throughput speeds on specified benchmarks, type and availability of peripherals, interrupt structure and support features, instruction repertoires, and utility packages. Further technical requirements may include alternative throughput capabilities, I/O capabilities, percentage utilization of resources, and cost performance ratio. One real requirement that is typically not considered but is very important is the type of performance evaluation and monitoring features available. This equipment must be available to allow the user to judge how well the system not only initially meets his procurement requirement, but continues to meet his job requirements in the future.

Attributes

Since system requirements cannot be totally comprehensive and precise unless they spell out a unique system architecture concept, attributes are introduced. Generally, it is not in the customer's best interest to define a precise system for if he does, he runs the risk of purchasing a system which is not cost-effective. If you buy a system in a non-competitive environment, you run the risk of paying more for that system than if there are two systems which can satisfy the requirements and, consequently, two manufacturers bid on the requirements. It is therefore to the user's advantage to make the requirements as definitive as possible, but allow for the introduction of various manufacturers' equipment to obtain the best price possible.

There are times when the architecture of the system will not be totally precise and there are features which are not specifically required, but are desired depending on their cost. Attributes are the wants (options) and evaluation criteria used to determine which characteristics make one system more desirable than another system even though they may both meet the requirements. Attributes usually deal only with the detailed technical aspects of a system. Attributes in the form of desired options deal with specific system features. For example, as an option to a processor which requires an interrupt structure, you may ask that the structure not only include enable/disable by class, but that the interrupt structure is desired to have, but not require, the ability to arm, disarm, enable and disable interrupts by level, and that a number of levels (such as 32) should be furnished. However, the requirement for such features may have actually been stated in the following manner: That the computer at least furnish four classes of interrupt with enable/disable characteristics. There are many more machines which satisfy the requirement than would satisfy the attribute. Depending on how systems rank on the option list and on other general attributes, the final selection of the design will proceed. Attributes are the intangibles of design.

The following list of criteria can be used to evaluate candidate architectures. These, then, are the alternatives that must be ordered to provide a list of the attributes and their relative importance in system design. These ranked attributes may then be used to trade off alternative machine architectures: flexibility, expandability, bus complexity, executive complexity, availability, adaptability, partitioning, modularity, reliability, maintainability, manufacturability, production cost, development cost, technical risk, logistics, programmability, support software cost, software adaptability and transferability, compatibility, and service. Having the ability to precisely quantify these attributes so that they may be measured against each other is difficult but necessary to insure a good system selection.

Options which are detailed technical desires may also be rank ordered and included in the attribute evaluation analysis. Typical options may be inclusion of a maintenance processor, ability to upgrade to a virtual memory system, or disk operating system availability. Obviously, detailed technical options are easier to measure than general attri-
requirements-oriented design

There are a number of steps involved in designing a system to a set of requirements. However, there are specific sets of steps involving the use of the requirements and attributes. These steps will be summarized and discussed herein. The actual design step will be simply described as "system design" such that at this point we will be able to see how the requirements and attributes actually lead to the detailed system design step.

The first step in requirements-oriented design is the problem analysis step. This step consists of determining, from the customer, what are the user's functional requirements. This may involve studying the applications as they are currently implemented, trying to project what the application is, or will be in the future, or trying to determine, based on technology issues, what needs could be generated in the marketplace for certain types of products. After the problem analysis step, we will have a detailed statement of the exact problem and its functional characteristics.

The second step is the determination of the requirements and attributes. This step involves taking the problem analysis functional description and translating it into a detailed set of requirements and a detailed set of attributes; the requirements specifying what the product system must do, the attributes specifying what detailed technical options are desired for the system, and a set of attribute priorities ranked for use as architectural trade-off parameters. In the case that more than one system satisfied the requirements, then, the attributes will be used in optimizing system selection. The attributes should be ranked in priority order at this point to insure objective trade-off and comparison of competing architectures.

The third design step is the determination and description of the requirements and attributes in a specification. Two documents should be generated, one which describes the detailed translation of the problem statement into the requirements and how the requirements were derived from the problem statement. This document then ends with a detailed specification of all machine requirements. Analogously, an attributes specification is generated. The attributes document should rank general attributes for trade-off usage and enumerate all detailed technical options.

The system design process (step) consists of designing delimited architectural choices. It is very difficult to translate a set of requirements onto a set of architectures, or to use a set of requirements to select a set of architectures. To achieve an accurate evaluation of the concepts under consideration they all may have to be designed to a level of detail that enables comparison to the requirements specification. Therefore, the design process really consists of using the requirements document and attributes document to narrow the scope of allowable choices and to narrow the number of concepts that satisfy the requirements. After all systems have been screened and those that do not satisfy the requirements eliminated, architectural descriptions are generated and furnished to the attribute evaluation step for all remaining architectures.

Using the attribute ranking, an attribute evaluation step is further used to list the architectures in descending order of their satisfaction of the attributes. Then, a number of systems which all satisfy the requirements and which are ranked in order of priority of satisfaction of the attributes may be selected and bids solicited from manufacturers. Alternatively, if we are designing a system from scratch, a composite concept which satisfies all the requirements and is one of the systems which rank high on the attributes should be selected for detail design. This process is probably iterative and best performed in a trial and error fashion. The main difficulty is that attributes cannot be precisely measured and thus the "optimal" design is always illusionary.

Requirements and attributes make the design process manageable. They cut down the number of choices we must make by clearly delimiting our choices. Obviously, the process must be iterative. Changes in technology, cost considerations, etc., may cause changes in the requirements or attributes specifications. Further, the use of requirements and attributes splits the design goals clearly into "musts" and "options". The effect of the process is to continually narrow the choice so that the designers can quickly focus on the problem. Thus, the requirement and attribute, and problem analysis continues to refine the general spectrum of architectures down to a single architecture best suited for the application. This is the architecture which is then designed.

Goals, policies, and product specifications

To help with project management, after the requirements and attributes are determined, a set of system goals which describe objectives and policies which point the direction for the achievement of the goals (how the goals can be achieved) can be determined. This information is useful from the management viewpoint, but does not get into the detailed system requirements as described in the product specification document which is a result of either system synthesis or system selection.

TYPES OF REQUIREMENT STUDIES—ANALYSIS VERSUS SYNTHESIS

Requirements seem to emanate from two important contexts: (1) Analysis of existing system concepts or (2) synthesis of new system concepts. The basic premises behind analysis efforts are the ideas of either (1) building a better copy of a competitor’s product, (2) upgrading a current product, or (3) integrating the best of many concepts into one new design. In these cases, extensive analysis of both the market and technology can be performed with the resulting analysis used to drive the design effort.
Analysis is also quite useful for the selection of a system (end-user perspective).

On the other hand, requirements synthesis involves the conception of a new product or new market. This environment may be intuitive and thus the issues may not be as clear as in applications in which only analysis is performed. In some circumstances, synthesis may only involve the projection of current analysis results to account for technology improvements or enhancements.

Requirements can fall along a spectrum. The design of a system from a requirement thus also can fall on a spectrum. At one end of the spectrum we have market-only requirements. At the other end of the spectrum is the technology-driven requirements. In this case, the problem is to build the highest performing product given a particular technology. In between there are many variations of the two extremes, most of which can be generically categorized as an attempt to design a product which optimizes the cost/performance ratio. Figure 1 summarizes the spectrum of possible requirement studies.

**USE OF ANALYSIS**

As previously indicated, analysis can be projected and used either by designers of products or by buyers of products. In this fashion, analysis can become a form of product concept synthesis.

**IMPORTANT EXAMPLES**

Requirement studies for a number of important systems or application groups have been documented in the literature and are briefly listed below. Some of these studies are important due to their actual resulting product; whereas, others are important not because they resulted in products, but because they illustrate specific design techniques. The studies listed below are illustrative of the types of efforts that have been performed and documented. This list of references does not attempt to be a complete bibliography.

**Bell**

C. Gorden Bell\(^1\) provides some interesting insight into the design process in this book.

**IBM Series 360**

Amdahl et al.\(^3,4\) in these papers presents the design strategy and requirements behind the 360 as well as insight as to how the system was impacted by the requirements.

**DEC PDP-11**

Bell\(^5\) provides the PDP-11 requirements and their design impact in this paper.

**Military systems**

The AMNCS (Advanced Multiplatform Naval Computer System) study\(^2\) provides significant insight into the requirements analysis of a large user group. In a related study, Punj provides a survey of the requirements and capabilities of a large number of Naval tactical operating systems. Further, related work includes: the E-2B requirements study\(^7\) of a specific aircraft, the MCF (Military Computer Family) design goals,\(^8\) and analysis of specific Army system requirements for a large number of systems.\(^9\) The Air Force has also performed significant requirements work and technology projections, as noted in References 10, 11, and 12. Further, interesting military studies include References 13 and 14.

**Distributed processing**

Kilpatrick\(^15\) presents a unique approach to the analysis of specific application for use with a distributed processor.\(^16\)

**Software systems**

A detailed analysis of many operating systems can be found in Reference 17. Analysis to support specific operating systems with hardware primitives is available in Reference 18.

**DISCUSSION**

In the following, four examples will be considered. Examples one and two will discuss general purpose computer systems. Example three will discuss the functional computation requirements for a large user application base. The
last example will discuss the detailed I/O, computation and memory requirements for a specific system.

Comments on general-purpose computers

For general-purpose, data-processing systems, the functional requirements usually cannot be well defined. The competitive business nature has kept the analyses that have been done in a proprietary vein. Currently, the most important commercial requirement seems to be software compatibility.

Although few general-purpose computing system requirement studies have been discussed in the literature, there are two classic examples of such requirement studies—studies were conducted on the IBM 360 Series, and the DEC PDP-11 Computer. Both articles contain a general description of the design objectives, the major architectural decisions, and some of the reasons for those architectural decisions. Since the machine to be designed is general purpose, the data from the detailed requirement studies has been filtered and the results indicate a series of design goals for the new computer system.

IBM system 360 requirement study

There were four major innovations in the IBM System/360: (1) a flexible storage concept which provided variable capacity; a hierarchy of different speed memories; storage protection and program relocation, (2) an I/O system which provided concurrent operation; large amounts of channel capacity; an integrated design between the hardware and software and CPU interaction; and a standard channel interface, (3) a general-purpose machine organization with very powerful operating system; logical processing operations; and many different instruction and data formats, and (4) machine-level language compatibility over a series of models with a performance range of over 50.

In performing the architecture development of the 360 System, several important systems concepts and trends were noted by Amdahl, et al.: (1) the adaption of business data processing to scientific data processing equipment, (2) the total system concept including I/O, (3) the use of program translators, (4) the development of large, secondary storage mechanisms such as tapes, drums and discs with many order-of-magnitude larger storage capabilities than seen in previous media, and (5) real-time and time-sharing system development.

Based on these general technical trends, a number of different concepts were provided for in the 360 System. The major requirements for the system could be grouped into five major areas: (1) provide for advanced system concepts, (2) provide an open-ended design, (3) ensure a general-purpose functional capability, (4) provide a cost-effective performance range, and (5) produce complete intermodule software compatibility. In each of these five requirements there was a number of subrequirements.

In the advanced concept area it was recognized that a major break would have to be made with existing products even though this would result in some software incompatibility. The break would establish the new family of machines. Therefore, the following subrequirements of the advanced concept requirement were considered: (1) that the computer provide for a family capability to provide growth, and to allow for a succession of product lines, (2) that a high-performance, general I/O technique be developed which would allow I/O devices tailored for applications to be used with any machine even though the I/O devices differed in rate, access times, or functionality (also, that the input/output channel and input/output control program had to be designed to be compatible with each other), (3) to utilize the throughput of a machine not to obtain high-speed processing, but to obtain high-speed problem solution by making a complex machine and programming system that are easy for the user to manipulate, (4) to increase CPU utilization for computing by providing for addition of compilation, I/O management, etc., (5) to provide a comprehensive operating system which includes extensive interrupt facilities, and good storage protection, (6) to provide a fail-safe/failsoft capability in systems with more than one CPU, (7) to provide a large storage capability rather than the 32,000 words normally required and furnished at that time, (8) to provide for large word lengths to accommodate large fixed- and floating-point words, and (9) to provide detailed hardware maintenance and diagnostic aids to reduce system downtimes and make identification of individual malfunctions easier.

The open-ended design requirement was an attempt to ensure customers that when they made the break with the previous software concepts, they would have a long-term, viable computer system which would continue to use the same architecture but be upgraded for speed and performance over a long time. This then enabled IBM to satisfy their customers so that when they made the switch to the new machine, they would not immediately, in three or four years, be required to make another switch. In this area, a number of subrequirements were identified: (1) that the new design must provide customer programming capability for over a decade; thus, the machines would have to remain with the same architecture for at least a decade, (2) that the design permit asynchronous operation of major subsystems so that subsystems may be updated technologically without impacting the total system configuration, (3) that many decisions be made to ensure that the functions of the machine are general; that is, that spare bits, etc., be carefully placed in the words to ensure that new techniques or new functions that came along did not obsolete the new product line, (4) that hardware and software control be embodied in the machine such that it could directly sense control and respond to other equipment modules via techniques which are outside the "normal techniques." This would provide for the construction of "super systems" that could be dynamically managed from the basic system. It would also provide for the construction of special systems designed for specific applications and would allow for the construction of systems where some short-sightedness of the original design had been encountered.
In order to meet varying requirements such as those encountered in commercial, scientific, time-sharing, data reduction, communications, and other types of processing, the 360 CPU would have to be capable of hosting these different applications. Thus, different types of facilities may have to be offered as options, but must appear as integral features from the viewpoint of the system's logical structure. In particular, the general-purpose objective dictated: (1) that manipulation of words or bits be such that the operation depends upon the general representation rather than on any specific selection of bits, (2) that operations be code independent, i.e., all bit combinations are acceptable as data and no data can exert any control function on the machine, (3) that bits be addressable, (4) that the addressing structure be able to address directly the unit used for character representation, i.e., addressability to the byte.

In the performance area, the main consideration is that the various products in the product line have a consistent cost-performance ratio that decreases or remains stable as the system performance increases. However, due to the compatibility constraint, there is a large problem in this area.

The last 360 requirement was for intermodel compatibility. At least six models were anticipated with a performance range of 50. Intermodel-compatible really meant program-compatible. Program-compatible meant that any valid program whose logic did not depend implicitly upon time of execution, or other side effect programming which would run on Configuration A, would also run on Configuration B if B contained at least the required storage, I/O devices, and optional features. A hedge clause was placed in this description such that any invalid program which violated the programmer's manual was not constrained by the manufacturer to yield the same results and thus was not strictly program-compatible. Therefore, if the user adhered to the programmer's manual, since the architecture of all the machines was identical, he could run on any 360 structure regardless of the speed differences between models. However, the program can run at different rates.

The article by Amdahl, et al. continues to describe how some of the decisions were made for the machine design which resulted in the 360, based on the previously summarized requirements. However, this paper is really only interested in the requirements and thus will not delve into the 360 architecture. But, we will make one point; that is, in doing a requirement study, regardless of the type, whether a general purpose study such as that for the IBM 360 or a detailed special-purpose requirement study, the design step must begin somewhere. It is difficult to break out of the mode of determining the requirements and starting the design. Therefore, it is irrelevant as to where we begin the design, except that we must break the requirements open from some position and say, "If we make this selection, how does this impact the other requirements?" In the 360, for example, this was accomplished by considering the basic addressing structure and first determining what the data format should be. The first decision was to go with an 8-bit byte. Once this decision is made, the design could proceed and the architect could lay out the formats, work on the field specifications, the instruction decisions, and the system architects could go to work synthesizing the various configurations. The architects will thus know how large the machine will tend to be; what kind of addressing modes are envisioned; how the memories have to be addressed, etc.; the point being, that we can get into a circuitous mode during the requirement study. A decision must be made! After making that first decision, we can then assess the impact upon the requirements and then change the decision, if necessary. But most importantly, we can begin the design process and cut down the amount of information we are required to deal with in general terms. The specifics we decide upon can be traded off against each other so that the system design may progress.

The DEC PDP-11 requirement study

The PDP-11 requirement study is slightly different than the IBM 360 requirement study. Whereas IBM decided to make a major break from their architectures, the PDP-11 was designed without regard as to whether or not it would be a major break. Rather, it was designed in order to solve certain technical problems that had been encountered by the customers of the DEC Corporation. Their customers were using four different machines at the point that the PDP-11 was conceived, a PDP-5, LINC, a PDP-4, and a PDP-8. Furthermore, these models were being used in communication control environments, instrumentation environments, preprocessors and communication processors for large systems, data acquisitions, etc. The PDP-11 was designed to overcome weaknesses that had been encountered in the current DEC mini-computers based on the customer's application experience. The weaknesses that the PDP-11 was to overcome include: (1) limited addressing space, (2) too few registers, (3) lack of hardware stack capability, (4) slow context switching among multiple processes, (5) lack of byte string manipulation capability, (6) lack of read-only memory storage facilities, (7) elementary I/O concepts, (8) lack of ability to upgrade users to a higher performance model, and (9) high programming costs due to lack of high-level languages and their associated software support.

The new machine family was to take advantage of new integrated circuit technologies that were becoming available, contain enough machine models to span a range of functions and performance, update the DEC product lines into what is considered classical, third-generation machines, work equally well in the addressing mode mechanisms 0, 1 or 2 address machine, and present the user with a very sophisticated connection system which later became known as the Unibus.

Notably, the PDP-11 requirements tended to be much simpler than the 360 requirements. There are a number of reasons for this including the size of machine, the size of the corporation and its market base, and the context in which the machines were used. However, one will note that the requirements which were used to define the PDP-11 are very distinct and direct to the points that describe what
changes must be made to be successful in the minicomputer business.

Functional requirements analysis for a large user group

In this discussion we will first consider the problem definition taken from the Advanced Multiplatform Navy Computer Systems Project (AMNCS). The problem statement or objective of the AMNCS Project was to provide information and guidance on requirements for the Advanced Multiplatform Naval Computer System. The specific project objectives were: (1) to identify a set of common functions for tactical data systems, (2) identify major, common subfunctions for the functions identified, and (3) identify computational functions for the subfunctions. Additional objectives of the AMNCS study dealt with computer architecture and technology projection, however, they are not pertinent to the objectives of this paper.

In identifying a set of common functions, two specific groups of functions were identified. These were functions which all tactical data systems tended to perform, and specific mission capabilities required of tactical data systems. Typical functions performed by all tactical data systems are: (1) data collection, (2) data measurement, (3) data processing, (4) data correlation, (5) data display, and (6) system executive control. Typical mission application functions that are contained in most tactical data systems are: (1) ‟track management, (2) ‟air interception computations, (3) ‟air traffic control computations, (4) ‟strike control computations, (5) ‟electronic warfare, and (6) ‟weapons allocation and fire control. Therefore, when analyzing a particular tactical data system, we may examine or specify a particular system in terms of its general common functions which exist in all tactical data systems and its specific mission-oriented functions. The main difference, then, between tactical data systems in this limited kind of environment will be in the functions included in the concept, rates of computation of the various kinds of functions, complexity associated with the computations; namely, decision requirements, number of variable requirements, etc., along with the distribution of the functions in the computing system. The system concepts of the tactical data system can cause these changes in requirements on a specific function basis due to factors such as degree of computation accuracy, amount of input data, type of computers to be employed in the system, volume and type of communications, physical dynamics of the physical systems, and the types of display information that must be generated. Looking at the list of common and mission type functions, Shen next generated a list of major functions for tactical data systems. Shen provides a list of representative United States Navy tactical data systems and a list of major functions that appear in these tactical data systems. Figure 2 indicates which major functions appear in these tactical data systems. Figure 2 is representative of the type of tabulation performed in a requirements analysis study. From Figure 2 we can also determine all the functions that must be computed for any given system. However, each of the systems have potentially different throughput requirements, input data rates, etc. Based on these data rates and the break out in terms of major functions, we are able to obtain an estimate of the computational complexity of any given system. That is, with rate information we now have the global description of any given system in terms of its system functions. Using the system functions, we can continue to refine each of the functions until we are able to determine the exact computation rates to be used in the design of the computer system. Therefore, the next step would be to take each of the functions, such as tracking, and break it down in terms of subfunctions. In doing this for the AMNCS Study, Shen broke the major functions down into a set of 72 specific computational requirements and tabulated these 72 computational requirements against the 27 major functions. Typical computational modules (requirements) used by Shen included: (1) matrix operations, (2) sensor calibration, (3) triangulation and trilateration, (4) \( R, \theta \) conversion—range and bearing to rectangular conversion (and inverse), (5) data encrypting, (6) track correlation, (7) vector operations, and (8) trigonometric functions.

With the list of subfunctions, Shen was able to generate a chart that compared the subfunctions to the major functions. This allowed a more detailed definition of typical rates for the functions, interaction between functions, parameters that must be passed between each other, difficult

![Figure 2: System vs. major functions](https://www.computerhistory.org)
computations, etc. This level of detail provides a basis for the type of computation rates that must be performed in developing a particular architecture. At this point, we have not received any information, other than rate information, that would tend to give us a feel for the type of detailed computations to be performed. The subfunctions, we have determined, tend to be very small, such as coordinate conversion type functions. At this level of complexity we are able to break down these functions even further in terms of the system model. Having the subfunctions, we are able to break out and construct a model of any given particular system. We could start in a tactical data system and construct a figure which consisted of, on the left side, a set of all the sensors available, a set of all the functions included in the system, and a set of the outputs in the system. Using such diagrams, we can determine the parameters to be passed between subsystems. Each function or subfunction may now be described in terms of its given inputs into the function, computations to be performed, outputs to be returned, and special comments. For example, coordinate conversion, as a function, is given a range and a bearing. It computes the \( x, y \) coordinates based on the equations 
\[
x = r \cos \theta, \quad y = r \sin \theta
\]
and returns as outputs \( x \) and \( y \). We can now develop a good description of all functions that must be actually computed in the system. A phase diagram indicating all the parameters and I/O data that has to be passed between functions of a particular system and the timing of all computations could be generated for estimation purposes.

A specific application requirement analysis

Kilpatrick\(^{15}\) considered detailed requirements for a number of limited systems. Most of the systems considered would compare to one of the smallest systems considered by Shen. In particular, we will consider Kilpatrick's air-to-ground attack system concept. A summary of this concept is shown in Figure 3. It consists of electro/optical functions for target acquisition, threat warning function, fire control function, three variations of navigation functions, an air data function, flight control function, digital data link function, and display function. These functions are all sensor processing functions that connect to a system via some types of I/O links. In this case, the system concept depicted in Figure 3 was a distributive processor memory system and is therefore listed as DPM (Distributed Processor/Memory) Processor. DPM does the appropriate computations on the data and performs the appropriate functions associated with the functional concepts. A detailed system description is then generated. It breaks out not only the detailed subfunctions, but additionally all the signals between the I/O system and the computer system.

Let us review each of the primary subfunctions provided by the system. The primary functions are:

- Target Acquisition and Electro/Optic—a means for target recognition.
- EW/ECM—the Electronic Warfare, Electronic
- Counter Measures function. It is used for threat warning location and neutralization of enemy vehicles.
- FLR—a Forward Looking Radar function that is used for target acquisition, the navigation function, and weapon delivery.
- LORAN/Inertial—a part of the navigation function.
- Flight Control—the Stability Augmentation Function (SAF) and the Attitude Flight Control System (AFCS) function.
- Digital Link—provides two-way communication between the aircraft and the ground.
- Vertical Situation Display—provides a pilot with flight direction, sensor imagery, and weapon delivery information.
- Horizontal Situation Display—provides navigation information, threat and target location along with sensory image functions to the pilot.
- Control Unit/Data Entry—provides for subsystem mode of operation and data insertion into the computer from the pilot.

From the subfunction definitions and detailed equations, and the block diagram which shows the interaction of all the functions, the requirements analysis is begun. Kilpat-
rick first generated the I/O data transfer requirements of which there are some important points to be noted. The system block diagram provided a listing of all primary information that must be transferred within the machine and processes within the system. Each of the subsystems has a number of other associated functions. These are the preflight type of functions of on/off power test and various kinds of status signals that do not impact the actual operational system.

With assumptions and derived information, the total I/O now can be specified for the system. Furthermore, taking a more detailed look at each of the functions associated with each of the subsystems, we can determine the data parameters that must be passed between each of the functions. These data parameters are tabulated as input data requirements versus computational functions.

With the I/O information and now the information detailing the interaction of data between the various functions, the requirements definition process can proceed by taking each of the separate functions and examining their algorithms. Thus, a set of total processing-time, memory and I/O requirements, by function, can be determined for each particular function.

Figure 4 lists the functions and memory cycles in thousands per second for the complete system shown in Figure 3. Since Kilpatrick made the assumption of a single-address machine with two memory cycles required for the execution of simple instructions such as Load or Add, a total computer memory cycle or instruction rate can be determined. The memory requirements for this concept are given in Figure 5 and the I/O requirements are summarized in Figure 6. With the I/O requirements, the memory requirements, and the system cycle speeds established, along with the detailed block diagrams and the detailed functional block diagrams, the requirements analysis is complete for this system. We could now proceed with the synthesis of the actual computing system.

SUMMARY

This paper has presented a description of the use and generation of computer system requirements and a description of the relationship of the design process to system requirements. Two types of criteria (musts—requirements and wants—attributes) were introduced. Four important example requirement studies were discussed and a brief bibliography of important requirement studies was given.

REFERENCES
