Using Resource Constraints to Control the Incremental Development of Large Scale MIS Projects

SULEIMAN K. KASSICIEH and ROBERT S. TRIPP
The University of New Mexico

Incremental MIS design and development methodologies have received limited attention in the MIS literature. This paper describes incremental methodologies appropriate for very large scale MIS developments which can cost millions of dollars and take years to develop. This paper describes a “resource constrained” implementation planning process designed to control both the costs of incremental projects and the functional evolution of the system. The paper also discusses how the process was used in a very large scale system development.

Introduction

Development methodologies for MIS are constantly being enhanced and described in the literature. There is widespread recognition that a rigorous framework is necessary to insure that sound technical design principles are introduced in the MIS development process. To meet this need, the traditional systems development life-cycle (SDLC) method has evolved. The SDLC calls for the development of a system by following a series of rigorous steps in sequence. The shortcomings of following this sequential process have been documented by Appleton (1986), Synder and Cox (1985), Ahituv and Neumann (1984), Juergens (1977), and King (1982), among others. These authors indicate that it is not practical to attempt to establish detailed requirements specifications and detailed system designs before beginning the development of any portion of the system. Tripp and Filteau (1987) discuss how these phases can take years to finish on large scale systems, and that before they can be completed, environmental factors change, requiring a change in the requirements and design of the system. As a way to avoid some of these problems of the SDLC, prototyping has been offered as at least a partial solution (Bally, Brittan, & Wagner, 1977);
In building very large scale MIS which span years of development as well as cost in the millions of dollars, neither of the above approaches is likely to produce desired results. The "build-it-twice" prototype approach is too expensive for these large projects and the level-by-level SDLC approach requires too much time to complete all steps before software products are released to users. As a result, several authors have suggested refinements to the SDLC model to facilitate the development and release of increments of the total system.

Boehm (1981) indicates that the main advantages of the incremental development approach over the prototype and SDLC approaches are that the increments are easier to test than the intermediate products associated with the SDLC and that the use of increments by users provides a less expensive way of incorporating user experience than the "build-it-twice" prototype approach. Using the incremental development approach requires the decomposition of the super-system into subsystems which are of manageable size and can be developed somewhat independently. As those subsystems are developed, the integration of the components into a unified entity must take place.

The remainder of this paper describes an incremental development implementation planning process which was used to: (1) decompose a very large MIS - the Air Force Requirements Data Bank - into subsystems, (2) prioritize those subsystems for development, and (3) control the costs and direction of the multi-year development.

**Incremental Design Methodology**

In multi-million dollar, multi-year MIS developments, it is absolutely necessary to develop and release products to users on an evolutionary and frequent schedule due to the length of the development effort which subsumes that people, conditions and systems that existed at the start of the development are not the same as when it is in the final stages. Moreover, the development team must reinforce users’ perceptions that the development goal is being met. Users cannot and should not be expected to wait for lengthy periods for improved capability. Equally important, users and top corporate management need constant reassurance that investments of great amounts of resources are, in fact, paying off.

The following steps in the iterative development process are recommended:

**Design Step 1**: A “fairly” detailed top-down analysis of the functional requirements of the entire super-system is conducted. The intent of this analysis is to describe the requirements of the entire system in some detail. The goal of this phase is to obtain knowledge of the objectives and boundaries of the system and to determine the feasibility of building the super-system.

**Design Step 2**: The super-system is broken down into smaller subsystems for fairly rapid development and release to users. As Boehm (1981) indicates, each of these increments or subsystems should be developed following the standard steps outlined by the SDLC. Juergens (1977) maintains that the manner in which the development is partitioned is critical to the success of the system. He indicates that large systems can be decoupled vertically and horizontally to form subsystems for development. Horizontal decoupling separates the development into relatively independent subsystems which do not have substantial interaction. These subsystems can then be developed while planning for the eventual interfaces takes place. This process needs to be coupled with an evolutionary database design which insures that the needs of evolving software applications are met (Tripp & Filteau, 1987). Vertical decoupling separates portions of the development within a functional area and allows a portion of the function to be developed and used while the expansion and enhancement of the module is scheduled for later development. This decoupling reduces risks and allows for incremental implemen-
Design Step 3: Enhancements to the basic capabilities of the system are tackled after the initial subsystems are in place. This step helps in keeping the users satisfied. Snyder and Cox (1985) have indicated the importance of being able to include changes in the system during its development. They propose a modification to the SDLC to allow for continuous analysis and modification of the system while it is in the process of development. The adoption of this process is a practical way of insuring the development is on track and achieving its objectives. When adopting an incremental development strategy, it is critical to have a control mechanism to contain costs for these large developments and to provide direction for the functional evolution of the system. The following section describes such a mechanism.

**Resource Constrained Implementation Planning**

In multi-million dollar MIS developments, controlling and retarding cost growth is essential. Investments of this magnitude will receive close scrutiny from the highest levels of corporate management. Even though the literature is rife with examples of MIS developments that have exceeded their budgets, there is surprisingly little written on the subject of how costs can be controlled. This paper is intended to address a portion of this void. The paper demonstrates that a strong “resource constrained” philosophy coupled with the incremental development approach can be used to successfully control the direction and cost of large MIS developments. The paper argues that the cost constrained implementation planning for each subsystem needs to take place in several steps or stages. These steps are outlined below.

**Implementation Step 1:** The first step in the implementation of the system involves completing an initial analysis and determining the preliminary cost/benefits of the MIS. In this stage, the cost of the MIS should be bound. That is, the life cycle estimates for the MIS should be determined and agreements to stay within this cost structure accepted by the parties responsible for the system development.

**Implementation Step 2:** The second step maps the development of specific requirements to macro schedules and costs. This stage involves breaking the processes described in the detailed functional requirements document into partitions which can be implemented, linking costs to these partitions to stay within the design cost agreed upon in the approval cycle.

**Implementation Step 3:** The third step determines the specific project development and release schedules and costs for each major subsystem. The implementation process may involve trade-offs between dropping some functional requirements and scaling down others to stay within the cost constraints. The user must participate in these decisions and should help the developer in staying within the allocated cost baseline for the project. An iterative process for segmenting large functional requirements into a series of smaller subsystems of development is shown in Figure 1. The incre-

![Implementation Planning Process](image-url)
ments are formed by examining all functional automated processes and grouping them into tentative subsystems using criteria such as user needs and technical design considerations. Each process will have a cost and a schedule assigned to it, making it easy to generate the proposed overall cost and schedule documents. The iterative process culminates in a plan to implement the core functions first and a priority list to be used in modifying the costs or schedules subject to the resource limitations.

The intent of the process is to use resource constraints to force users to prioritize their developments requirements. Users must know that this process is supported by top management and that they will be held to fitting the development within the assigned cost boundaries. An analogy, similar to that used by King (1982) of a couple buying a house, depicts the differences between the resource constrained implementation planning process and the traditional requirements specification view. The couple wants a two story house with four bedrooms, three baths, kitchen, and living room. At the beginning of the process, they believe they have enough money to build the house with everything they want. Soon they begin to differ on the accoutrements they should have. They go to the bank, but the banker says they cannot have any more money above that which has been approved. They then agree that their basic plan is sound but that they will build the house in stages. The first priority is to rough in the entire structure. Next they agree to finish the kitchen, downstairs bath, living room, and one bedroom. They have specified basic materials to be used by the contractor. One of the partners wants to have fancy fixtures in the first bathroom and the other wants higher quality carpet in the living room than that which is specified in the contract. The contractor explains that they can upgrade the carpet if they will reduce the amenities in the follow-on bedrooms or bathrooms. If they agree, the contractor needs to be advised in writing to change the development plans and proceed accordingly under the revised plans. If they disagree, plans to complete the first rooms proceed until they can formally agree and notify the contractor of the changes in writing. There are, of course, many other possible outcomes of this process, but the same type of process is needed to control the costs of large-scale MIS developments.

This resource constrained approach runs contrary to the traditional requirements-oriented view which indicates that requirements should be specified in detail without regard to a practical budget. The traditional requirements approach, in the housing example cited above, could lead to the couple specifying their desires for both fancy bathroom fixtures and high quality carpeting in the living room. In this case, the budget of the house is likely to be higher than necessary. In MIS developments, requirements growth is just as likely unless they are viewed from an affordability viewpoint.

The remainder of the paper describes an example of how the “resource constrained” implementation planning process was implemented in the development of the Air Force’s largest distributed processing MIS referred to as “The Requirements Data Bank” (RDB). This process has also been successfully used in the development of the Air Force’s Weapon System Management Information System. The paper discusses how the implementation planning process was used in the development of RDB to control development costs and aid in efficient development and user acceptance of developed products.

The Requirements Data Bank Project

The Air Force Logistics Command is responsible for supporting the material needs of Air Force weapon systems via the Material Requirements Planning (MRP) process. The budget for this endeavor is $15 billion annually. The 22 systems that were used to manage this task were prone and antiquated. The most serious deficiency was that separate sys-
tems were used to make buy and repair decisions for similar components independently without examining the impact on weapon system capability. This practice resulted in optimization of the individual item objectives rather than the overall weapon system availability. Balancing resources to yield optimum weapon system support for the dollars invested was not possible given the diversity of information systems. In addition, even though many of these requirements determination systems used the same information, the information elements were often named differently in the various distributed systems. The result was databases that contained incorrect data, did not allow for sensitivity analysis, and produced errors in projected needs. Changes to the MRP system were warranted and the commitment to systematically analyze and evaluate alternative approaches for improving the MRP process was dubbed the Requirements Data Bank (RDB) (US Air Force, 1982).

The analysis by the AFLC team produced a master functional description (MFD) which identified needed improvements in this process. The MFD was a general document which contained the goals for RDB and broke the development into several modules which could be developed relatively independently.

After a thorough analysis of staffing requirements, AFLC top management determined that it did not have enough skilled systems people to develop RDB with in-house resources. As a result, the analysis cadre worked with contracting specialists to develop a statement of work (SOW) which would be used to solicit bids from professional services contractors to develop RDB. The center piece of the SOW was the MFD. An important feature of the SOW called for each bidder to propose the best computer and functional architecture to meet the needs of the MFD with associated cost and schedule estimates. At this point, AFLC made forecasts of the resources that would be necessary for the RDB development. It was estimated that RDB would cost nearly $300 million to develop and operate over the next ten years (US Air Force, 1982).

To control the risks associated with a development of the scale of RDB, AFLC decided to solicit bids to perform this work using a “cost-plus-award fee” contractual agreement. The award of this fee was based on several criteria, including performance on cost containment, technical design, and schedule maintenance. The fee portion was fixed based upon the bid received from the winning contractor, but the cost portion of the contract was in theory supposed to be allowed to vary depending on the amount of work required to complete the job.

AFLC made financial resources available for the ten-year development which started in January 1985 (USAF, 1982). Following contract award, AFLC assigned its top systems people to staff a System Program Office (SPO) to manage the efforts of the contractor and allocated the best functional specialists with the SPO and contractor for the life of the development. The functional specialists were assigned from the headquarters but had the authority to request the assignment of functional specialists in each area from the decentralized operational sites of AFLC to the development site whenever their expertise was needed.

The scope of the RDB project was enormous. For example, it was estimated that the RDB effort will involve development of nearly 3.7 million lines of code that will replace the primarily batch oriented 22 main data systems. There will be over 5,600 users involved with the RDB using smart terminal interfaces in several geographically dispersed distributed processing sites within the United States. The RDB data environment is estimated to include over 9000 unique elements and an intermix of two data distribution schemes. The first scheme partitions the necessary data to each of the distributed sites, thus ensuring fast response times and immediate data currency. The second
scheme replicates a minimum essential set of
data across all sites and consequently allows
site data visibility, quick response, and minimi-
ization of network overhead. A complete copy
of all RDB databases is also held at two sites for
security backup reasons. The complexity of
synchronization and communication of data
bases within RDB is apparent. The data at any
one site is in excess of 60 billion bytes with
90% of the RDB operations on-line and 10%
batch.

Application of the Resource-Constrained
Implementation Planning Process

This section describes how the three
step resource constrained implementation
planning methodology which was described
earlier was applied to the RDB project.

Implementation Step 1: The MFD and
resultant contractor proposal to meet the needs
of the MFD provided a detailed top-down
analysis of the functional requirements of the
entire RDB super-system. Recognizing the
complexity and the size of RDB, the MFD
called for an evolutionary and incremental
transition to a new architecture. The MFD
outlined a total of 16 major functions that were
needed to meet the RDB objectives. The MFD
also specified response times that had to be met
to accomplish the various functions contained
in the MFD. The MFD further specified that
each of the functions would be developed in an
incremental fashion so that users would be able
to use portions of the system while develop-
ment of other portions were in process.

In addition, the new ways of doing
business not only required the development of
new computer software but required extensive
user education on the direction of change. This
process would require a significant amount of
time to accomplish and incremental develop-
ment allowed this process to take place in an
orderly and efficient manner. Also, the risk of
moving to a new MRP process all at once was
viewed to be too great and the incremental
approach facilitated parallel runs of the new soft-
ware and hardware with the old set so that the
current mission was accomplished with minimum
 disruption to the readiness of the Air Force.

The resultant contractor bid provided a
conceptual architecture and proposed macro costs
and schedules needed to meet the objectives
specified in the MFD. It should be recognized that
because of the scope and uncertainty associated
with the specific design requirements of the RDB,
the contractual vehicle was a cost-plus fee as
described earlier.

This type of contract would appear to be
appropriate for such a development. This view is
supported by Synnott and Gruber (1981) when
they suggest that cost projections to develop MIS
systems be reviewed and updated periodically as
the project moves through various phases of the
MIS development life cycle. They indicate that
corporate managements need to understand that
initial estimates of MIS developments are usually
made with only a portion of the information that is
needed to yield firm project cost estimates. The
initial estimates are usually rendered after a pre-
liminary systems analysis and cost/benefit study
are complete. They indicate firm cost projections
cannot be made at this stage and need to be
adjusted as more detail is added to the project at
the completion of the detailed functional design
stage and again at the completion of the detailed
system design stage of the project. Thus, they
suggest that corporate staffs understand this evo-
lutionary cost estimating approach and expect
cost revisions to major MIS developments and
plan for them to happen. This cost revising ap-
proach, while theoretically sound, will not fare
well for MIS developments in the public sector in
general and in the Department of Defense (DOD)
in particular due to the zero-based budgetary ap-
proach. Budgets for MIS developments within
DOD have to be formulated and defended several
years before the project is to be started to allow for
the allocation of resources needed for the develop-
ment. These budgets are formulated after the
initial systems analysis has been conducted.
While the DOD allocation and budget formulation
processes are flexible enough to accommodate the periodic cost estimation procedures suggested by Synnott and Gruber (1981), neither senior level military and civilian authorities within the DOD nor the public at large understand the need for periodic revisions to the cost estimates for large scale MIS projects. Without this understanding, headlines indicating that a large DOD MIS development is experiencing cost overruns would likely result if this approach were followed in DOD developments. Thus, even though the RDB contractual vehicle was designed to recognize the inherent risks associated with this development, the strategy of designing RDB to the cost approved was adopted early in the project. In fact, the development was treated as a “fixed-price” contract vehicle shortly after the beginning of the development (US General Accounting Office, 1986).

Thus, the first step in the implementation of the cost constrained methodology was recognized early in the system development. In other words, top management recognized the cost of the RDB development had to be bound and methods to develop the system within these constraints had to be initiated. In addition, the incremental development strategy was viewed to be necessary and commitment was given to use this approach.

Implementation Step 2: The second step of the constrained resources implementation approach maps the development of specific requirements to macro schedules and costs. As indicated above, this stage involves breaking the processes described in the detailed functional requirements document into partitions which can be implemented, linking costs to these partitions to stay within the design cost agreed upon in the approval cycle.

The macro level objectives of RDB were contained in the contract which included the MFD and associated target architecture and macro level costs and schedules for each of the 16 major processes. Thus, the contract and associated MFD were viewed by the development contractor, users, and development program office as “the corporate charter” for the development of RDB. It laid out the top-down approved macro level products, costs, and schedules for the development. Changes which affected any one of these parameters had to be approved and the MFD modified in the contract in order to reflect the changes before the contractor would act on them.

The requirements contained in the MFD were broad in scope and not detailed enough to provide specifications for the development of each of the 16 major segments of RDB. As a result, one of the first actions undertaken by joint user-contractor teams was to develop detailed process functional descriptions (PFDs) for each of the 16 major functional processes covered by RDB. To facilitate obtaining user cooperation in writing these PFDs, no attempt was made a priori to restrict their content. The idea was to solicit user inputs on what had to be enhanced in their functional areas to meet the MFD goals. Before acceptance of the PFDs, the contractor had to demonstrate that they were consistent with the intent of the MFD and the detailed process requirements documents. The portfolio of 16 completed PFDs included over 6500 pages that describe more than 2200 processes that were to be automated. As a result of developing these financially unconstrained PFDs, senior user representatives and the development program management team knew some “gold plating” was included and had to be controlled, so the most important basic features of the RDB were developed first.

Perhaps the most important piece of this strategy involved the allocation of specific functional capabilities contained in the PFDs to a product or process group which was capable of being implemented. As shown in Figure 2, each product group contained several processes which could be automated to create a set of capabilities. Each process was considered for inclusion in a product release based on technical criteria and a cost/benefit analysis of each incre-
mental capability. Some of the technical criteria include: grouping processes which share a common portion of the database; grouping processes which need to be performed to complete a useful function; and keeping the product small enough so it could be developed by a development team of between seven and fifteen members in less than one year. The last criterion was viewed by the senior user, contractor, and development program management team as especially important in controlling the software costs and schedules. With products of this size or smaller the team felt they could effectively develop several of these increments and implement them without serious cost, schedule, or quality problems.

This strategy involves active user participation in determining the functional processes which must be grouped to make a meaningful product. In RDB, the users were instructed on the importance of developing the functional capabilities in small increments which could be developed in less than one year of development. In addition, users were asked to place priority on developing products in the first increment which would, if at all possible, incorporate the functions of the current system plus the most necessary enhancements in the first product release. Other releases would contain additional enhancements and features the users believed would be “nice to have” which were contained in the PFDs. Users would then group functions into products in an iterative fashion as shown in Figure 2 until the products incorporated all the function and processes contained in the baselined PFDs. This process resulted in priority groupings of product releases so that the most important functions to the users were grouped in early releases and other “nice to have” or ambiguous features were left to later products.

As an example, one of the 16 major PFDs addressed the requirements for application and program data needed to forecast buy and repair budgets for items used on Air Force weapon systems over a seven year time horizon. The Applications and Programs PFD contained 364 pages that described 144 processes that the users wanted automated in RDB. These processes were allocated to four product releases which were to be developed over a three year process. The first product replaced the current bill of materials process, improved file maintenance for this process, provided a new capability to enter an item indenture structure and determine higher and lower assemblies of that item, and

![Figure 2: Implementation Planning](image-url)
integrated this process with an item cataloging system. Another portion of the first product included future program data which would be used to compute inventory buy and repair requirements for individual items over a seven year time horizon. It provided enhancements over the current system such as trend analysis capability on program data.

The second product or block release for this function included the historical retention of indenture data which were used in a particular computation and the program data that were used to compute requirements for a given period to support audits and other reviews. The product also provided the ability to incorporate aircraft engine indenture and program data into a single integrated data system to provide the ability to make integrated resource buy and repair decisions instead of doing these in a “stovepipe fashion.” The third product provided greater query capability into these products. The four product supported advanced “what-if” capabilities.

The final result of this process is a set of products which have been prioritized by users for development. Each has individual development costs and schedules associated with it. This “first pass” at developing prioritized products, costs, and schedules for each PFD did not constrain costs. In other words, the first iteration produces a good estimate of what the full cost to develop the unconstrained PFD would be. For later discussion, this process is called a stage 2 cost estimate of the PFD. The initial macro estimate which identified the bounding cost for the segment shall be referred to as a stage 1 cost.

Implementation Step 3: Given products, costs, and schedules for the PFDs which came out of Step 2, a reconciliation between programmed funds for these efforts and the requirements of what was needed to develop each product had to be made. Funds were programmed on a yearly basis for each functional area called a Logical Application Group (LAG). A LAG could contain funds to automate the functions of more than one PFD. In addition, the LAGs all covered more than one year for developing the functional capabilities described in the LAG. Thus, each product had to be assigned to a LAG and the availability of programmed funds to cover the products had to

Figure 3: Stage 2 Financial Planning Models
be checked with the requirements for the products within that LAG. If there was a shortage of funds for a given LAG, the users were called upon to make trade-offs between product content between years or by dropping some functional requirements and scaling down others to stay within the cost constraints. Costs have been allowed to migrate across products within a LAG and between products of different LAGs if the users are not willing to reduce some functions in one LAG but are willing to reduce the functions required in another to remain within the overall project cost projections. The output of this process is an agreement on micro schedules and evolutionary or incremental software release plans. Figure 3 shows a schematic of this approach.

The key to making a resource-constrained approach work is for the users to understand that top corporate management is constraining the cost of the project. In the RDB project, this understanding was achieved with the result that the users made the trade-off decisions to keep the project within its cost guidelines. As a result of using cost constraints as a tool to force users into prioritizing functions, the cost of RDB, with very minor variations, was held to the original planning estimates for the development. Thus, with the exception of minor revisions, the cost projections for RDB which were put together three years before the development began bound the project in terms of cost. In a large scale system project such as RDB, the resource constrained strategy is appropriate for public and private sector projects. One could argue that a main reason for substantial cost growth in MIS developments is due to the addition of enhancements that are unnecessary for the sound performance of the system. Without tight cost constraints, an MIS project can keep going indefinitely and the outcome may be no better than a cost constrained project.

Using this development strategy, the RDB system currently has more than ten operational increments. Several of these increments have been added to with numerous block releases which have upgraded the original release with enhancements. About 1.5 million lines of code have been developed and are currently in use with more increments in the pipeline. The current data base is more than 40 gigabytes and growing with additional capability. The users are pleased with the results of the process and support the implementation strategy.
The Cost of Implementing the System

There is a cost associated with implementing the resource constrained incremental development approach, but it is not significantly greater than the ordinary cost associated with imposing cost/schedule/status (CSS) reporting requirements on large scale MIS developments. CSS reporting is necessary on large scale programs and usually supported by a staff to gather and analyze the data. In large programs this data is routinely collected and reported using software packages which have been developed to support this program activity. The only impact the concept has on this reporting function is on the work breakdown structure to facilitate reporting by expenditure of resources by product.

In the case of RDB, the development contractor spent about 48 man-months of effort on this function as well as perhaps as much as another 100 man-months of senior management and product leader time out of a total of more than 3000 man-months of total effort of the project each year. The users spent approximately the same amount of time on these issues. Thus, the rough cost of using CSS, including reporting and breaking the big PFDs into small incremental products, may have been close to 10% of the effort of the project.

Most, if not all, of this cost would be incurred no matter what development strategy was adopted. Senior project managers should focus their time on setting goals and monitoring the progress in achieving those goals. This approach did require the senior project management team including the senior user representative to educate users and development team personnel on the importance of costs in the project and how to use these constraints to determine how to structure the workload so that the best system for the dollars could be built. One could argue that this function should be accomplished in all project development strategies, but resource constraints helped to focus attention on building the essential parts of the system before “nice to haves” were built.

Summary

Figure 4 shows a pictorial of the RDB evolutionary resource constrained methodology. The MFD was viewed by the development contractor as “the corporate charter” for the development of RDB. The MFD laid out the top-down approved macro level products, costs, and schedules for the development. It also shows how the bottom-up cost and schedule inputs influenced the MFD products, costs and schedules. The bottom-up cost and schedule estimates were broken into two stages.

Stage 1 cost and schedule estimates were based primarily on software engineering estimates, and were intended to establish overall long range control of the project. These estimates were made at the time of contract selection and redone each time a PFD for a major segment of RDB was baselined (formally agreed upon for technical content). There were 16 major segments within RDB and these estimates were redone each time a segment functional description was completed. The purpose of redoing these estimates at these baselines was to insure that there were adequate funds to accomplish the goals of the user as specified in the baselined FDs. If resources and desires differed, the user was asked to make trade-offs between the requirements to live within the approved funding and the MFD was changed to reflect new guidance to the development contractor. As each PFD was broken into individual products for development, detailed cost estimates were made by the design teams responsible for developing the product were made. These estimates were referred to as Stage 2 estimates as shown in Figure 4.

As discussed earlier, the content and schedules for each product is an iterative process in which the user plays heavily. The important part of this process is that the Stage 1 costs and schedules govern the expenditure that will be made on these products. If trade-offs are necessary to stay within the budget, users are asked to
help determine which processes would be reduced or rescheduled to remain within the budget. The output of this process is an agreement on micro schedules and evolutionary or incremental software release plans.

The design and implementation approaches taken in the RDB project have been successful in achieving the goals of producing the important portions of the system first, thus maintaining the readiness of the Air Force, of prioritizing enhancements, and of keeping cost contained within the budgeted amounts. For very large scale systems, the methodology works.

References


United States Air Force (1982). The Air Force Requirements Data Bank (RDB) economic analysis, HQ AFLC/LO(RDB), Ohio: Wright-Patterson AFB.


Dr. Suleiman K. Kassicieh is associate dean and professor of Information Systems at the Robert O. Anderson School of Management at the University of New Mexico. His research interests include artificial intelligence modeling, decision support systems and systems development. Previously his articles have appeared in Operations Research, Information and Management, OMEGA, International Journal of Policy and Information, and others.

Dr. Robert S. Tripp is an associate professor of management information systems at the Robert O. Anderson Graduate School of management at the University of New Mexico. Previously he was the program director for the Air Force Requirements Data Bank System Program Office. In addition he has also directed the successful development of two other large scale MIS developments, the AFLC Command, Control, Communications, and Intelligence System and the Weapon System management Information System at Headquarters Air Force Logistics Command.
Related Content

Organizational Memory Information Systems: A Domain Analysis in the Object-Oriented Paradigm
www.irma-international.org/article/organizational-memory-information-systems/51065/

Mapping the Intellectual Structure of the Big Data Research in the IS Discipline: A Citation/Co-Citation Analysis
www.irma-international.org/article/mapping-the-intellectual-structure-of-the-big-data-research-in-the-is-discipline/193611/

International Digital Studies Approach for Examining International Online Interactions
www.irma-international.org/chapter/international-digital-studies-approach-examining/14484/

Wireless Networks for Vehicular Support
www.irma-international.org/chapter/wireless-networks-vehicular-support/14197/

Setting Up to Fail: The Case of Midwest MBA
www.irma-international.org/article/setting-fail-case-midwest-mba/33487/