IRM-5231-18A

U.S. Marine Corps

PROTOTYPING STANDARDS

PCN 186 523118 00

UNITED STATES MARINE CORPS
MARINE CORPS COMPUTER AND TELECOMMUNICATIONS ACTIVITY
QUANTICO, VIRGINIA 22134-5010

IN REPLY REFER TO:
5231/18A
CTAS-41

APR 20 1993

From: Director, Marine Corps Computer and Telecommunications Activity

Subj: PROTOTYPING STANDARDS

Ref: (a) MCO P5231.1B
(b) MCO 5271.1
(c) MCO P5600.31

Encl: (1) IRM-5231-18A

1. PURPOSE. To provide guidance and instructions on prototyping as required by reference (a).

2. CANCELLATION. IRM-5231-18

3. SUMMARY OF REVISION. This revision updates the prototyping standards to reflect revised Marine Corps Life Cycle Management policy on automated information system projects and to reflect current prototyping methodologies.

4. AUTHORITY. This publication is published under the auspices of reference (b).

5. APPLICABILITY. The guidance contained in this publication is applicable to all contractors and Marine Corps personnel responsible for the preparation of system prototypes. This standard is applicable to the Marine Corps Reserve.

6. DISTRIBUTION. This technical publication will be distributed as indicated. Requests for changes in allowance should be submitted in accordance with reference (c).

7. SCOPE

a. Compliance. Compliance with the provisions of this publication is required unless a specific waiver is authorized.

b. Waivers. Waivers to the provisions of this publication will be authorized only by the appropriate approval authority as defined by reference (b) on a case by case basis.

8. RECOMMENDATIONS. Recommendations concerning the contents of this technical publication should be forwarded to CMC (MCCTA) via the appropriate chain of command. All recommended changes will be reviewed upon receipt and implemented if appropriate.

Subj: PROTOTYPING STANDARDS

9. SPONSOR. The sponsor of this technical publication is CMC (MCCTA).

D. P. HOUSTON
Colonel, U.S. Marine Corps
Director, Marine Corps
Computer and
Telecommunications Activity

DISTRIBUTION: PCN 186 523118 00

Copy to: 8145001

UNITED STATES MARINE CORPS

Information Resources Management (IRM)
Standards and Guidelines Program

Prototyping Standards
IRM-5231-18A

Enclosure (1)

(This page intentionally left blank)

TECHNICAL PUBLICATION LIBRARY MAINTENANCE

The Information Resources Management Standards and Guidelines Program will be maintained at each receiving activity. Each activity is responsible for ensuring that their set of technical publications is complete, and that all published changes are promptly incorporated.

RECORD OF CHANGES

Change Number	Date of Change	Date Received	Date Entered	Signature of Person Entering Change

(This page intentionally left blank)

PUBLICATION TABLE OF CONTENTS

LIST OF FIGURES

(This page intentionally left blank)

Chapter Table of Contents

Chapter 1

GENERAL

(This page intentionally left blank)

Chapter 1

GENERAL

1.1. INTRODUCTION. As the complexity and costs associated with Automated Information Systems (AISs) increase, and user satisfaction with the development process and delivered product decrease, system developers must consciously work to deliver the right product on time and within budget.

1.2. BACKGROUND. For years, systems developers and management have tried to control the AIS development process, and remake it from an art into a science. To this end, numerous methodologies have been introduced to refine and control the development process. These have achieved varying degrees of success, but all have fallen short of reducing total life cycle costs and consistently meeting or exceeding user expectations. Prototyping is not another new methodology. It is a technique which has existed for some time, and it has long been portrayed as anything from a haphazard, experimental approach to a methodology. This standard views prototyping as a tool with a specific purpose, to be used within the existing Systems Development Methodology (SDM).

1.2.1. Objective. The objective of this technical publication is to provide a framework for the use, development, and control of prototypes within the Marine Corps' SDM.

1.2.2. Purpose. The purpose of this standard is to provide guidelines, procedures, and information in the use of prototypes in the development of AISs. It provides direction for the consistent application of prototypes for systems development, and provides information on how and when they should be used.

1.3. SCOPE. This technical publication covers the prototyping process and its impact on the Definition, Design, and Development phases of the SDM. References utilized for the development of this Technical Publication are located in Appendix A.

(This page intentionally left blank)

Chapter Table of Contents

Chapter 2

BACKGROUND

(This page intentionally left blank)

Chapter 2

BACKGROUND

2.1. GENERAL. A prototype is a dynamic simulation or model of a complex process or system that serves as a communications tool between the requestor and the developer. As a dynamic model, it involves interaction between the system user and data, processes, and outputs. As a communications tool, it physically portrays the ideas and concepts as defined by the requestor and as understood by the developer. It is a tool which aids in the development of the functional requirements, alternative designs, and detailed specifications. It is particularly useful in modeling system-user interfaces.

Prototyping does not eliminate the need for functional requirements or design specifications. It does not replace project or system documentation, or replace the development phase. It is primarily an analysis technique geared toward improving customer satisfaction with the final product. While prototyping will generally decrease the full life cycle costs of a system (development + maintenance + enhancement), it may actually increase time and costs in the development phase, particularly during analysis. The offset to the marginally increased costs is delivery of the product the requestor actually wanted. Quite often in the past, this has not been the system envisioned nor communicated using diagrammatic and textual specifications alone.

2.2. OBJECTIVE. The primary objective of prototyping is to discover what the user wants the system to do, and to provide or at least plan for these capabilities with the initial implementation. In the long run, this will decrease time and full life cycle cost of the system by eliminating some "enhancements" entirely as their functions are incorporated into the initial implementation. Other enhancements will be greatly reduced in complexity when they become true add-ons to the system. All too often, deferred deliverables are referred to as enhancements, when in fact they are actually rewrites to accommodate desired functionality.

2.3. GOALS. As a communications vehicle, the primary goal of prototyping is to ensure delivery of the system the requestor requires, not just to deliver the wrong thing faster. Prototyping has the capability of ensuring that the system which is developed and delivered is the one which is most responsive to the users' needs. In order to effectively meet these goals, the prototype must be developed and modified quickly, easily, and cost effectively.

2.4. PROTOTYPE DEVELOPMENT. Traditionally a linear approach has been used in the standard development of hardware and software systems. This approach has several limitations that can be improved with the use of prototyping. Prototyping is a useful

adjunct to the development process. The linear approach involves, producing, reviewing and freezing the design and then producing the final product. This linear approach is paper based and document driven and it does not allow the users to:

a. Get a hands-on feel for the system prior to acceptance testing, or

b. Develop their requirements while experiencing the new processing environment.

Together these two problems can result in a system that does not meet the expectations of the users. The consequences of this can result in, (1) Expensive changes being made to implemented programs to process transactions correctly, (2) New functions being added to the system, and (3) The system being rejected by the users.

Prototyping addresses two shortcomings inherent in the traditional paper based linear approach: It is a user friendly technique, requiring active user participation throughout the design of a system; and allows the user's requirements to evolve throughout the project. The differences that prototyping brings to systems development is evident in the tools used, skills needed, and procedures followed.

2.5. CHARACTERISTICS. The term "prototype" has been used loosely over the years when applied to the development of ADP systems. Rightly or wrongly, it may evoke negative connotations among those involved with a project, due in large part to prior experiences with any system which did not meet expectations being labeled a prototype. A prototype as used here will have the following characteristics:

a. Exists as a dynamic visual model: It will actually perform or appear to perform functions. It will not be limited to a set of drawings or written specifications.

b. Functional after minimal effort/cost: The prototype must be developed quickly from initial rough requirements without incurring unreasonable costs. This implies turnaround time for development and subsequent changes measured in days or weeks, rather than months or years.

c. Flexibility. The prototype will not be bound by the ultimate technological solution. The system being developed may involve a wide area network connecting micro and mainframe platforms. It can be prototyped on a local area network (LAN) with two workstations.

d. Limited in detail and documentation: In order to easily change to meet and to be able to encourage the changing of requirements, the prototype must not be encumbered by volumes of documentation. Nor must it include any more detail than is

necessary to convey the point. The documentation should be limited to that which is necessary to note changes or additions to the model, keeping in mind that the ability to rapidly and economically change the model requires a similar balance by minimizing the time and effort spent on documenting it. If detail is necessary, it should be limited in scope. The screens needed to demonstrate updating one data element may be required in the prototype, but including all screens for updating every element would impose rigidity on the prototype.

e. Not necessarily representative of the complete system: The model may lack controls, detail modules, exist on a different hardware platform, etc. However, depending on the requirements (functional, control, and performance) of the final system, the prototype **may** contain elements which will be used in the final system.

2.6. ALTERNATIVE PROTOTYPING METHODS. Prototyping has existed in various forms and been applied by numerous professions for years. Likewise, there are several valid (and a number of invalid) prototyping approaches.

2.6.1. Throwaway. This approach assumes that the prototype is strictly a non-implementable model. Whatever code and modules are developed will be scrapped regardless of how well they function. This is particularly valid if the model of a mainframe system is created on a personal computer.

2.6.2. Quick and Dirty. This method has existed since the development of the second program. It involves cloning a system from parts of other programs or systems, making modifications as needed for the current application, and running the composite to see how it works. Unfortunately, patchwork quick and dirty programs are often implemented as production systems since they are written in the target language, are working, and everyone would rather go on to solve new problems than rewrite something that already exists. However, these systems usually contain unnecessary and tightly coupled code, and maintaining them becomes a nightmare.

2.6.3. Detail-Design Driven. This method, associated with structured systems development, involves a rigorous definition and design process, with the prototype being developed only after the initial specifications have been prepared in great detail. The prototype often takes the form of a stub-coded program - the higher logic levels exist to simulate calls, but underlying detail logic is left uncoded until the model has been approved. This approach is valuable for the developer since it ensures that module interfaces are compatible, but the tremendous effort expended in developing and documenting the requirements and design lock the requestor into that which was contractually specified. The requestor still does not have a chance to experiment with a model prior to initiating the development.

2.6.4. Rapid. This is the preferred method for use by systems

developers within the Marine Corps. It is a method of enhancing specification development through the iterative use of a quickly developed, interactive model. This model, or prototype of the desired system, is constructed using those tools which allow the creation of a functional model within a matter of days. By creating and changing the working model of the rudimentary functions of the system quickly, the developer allows the user to evolve the system specifications through interaction. There is no penalty for changing ones mind or adding functions while the prototype is developing - in fact, this is expected and encouraged. The intent is to remove the burden of pre-defining all functions and their interactions without having the opportunity to see what is being specified. Once the prototype has been completed, the formal specifications, etc. are developed as required under the SDM.

2.7. BENEFITS. The proper use of prototyping in conjunction with the SDM can accrue significant long term benefits over the system life cycle.

2.7.1. Delivery of the Right System. The non-prototyping approach to systems development is based on the assumption that the requestor knows and can visualize the entire system being requested. Experience has shown that in the overwhelming majority of cases, the opposite condition exists. This should not be viewed as a weakness of the user, the analyst or designer, or the methodology. It is simply unwise, and unfair, to assume that reams of documentation will convey the workings of the proposed system to the requestor. Plans, requirements, specifications, etc. are all necessary to adequately and accurately describe the system. However, they have limitations as a communications tool between the requestor and the analyst. By quickly providing a working model as a basis for further discussion, the requestor and analyst can work together to define the system with a degree of certainty that what is being requested and what is being constructed are one and the same. This should not be taken as a claim that future requirements will not arise (some of which may be beyond the bounds of the project). Nor should one assume that business needs will not change prior to delivery of the system. These situations will continue to occur, and must be handled in accordance with the SDM.

2.7.2. Avoided Costs. As shown below, initial costs for development of the system may exceed those estimated using non-prototyping methods. Two points should be considered before dismissing prototyping as an unneeded added expense which will do nothing to reduce development time. First, estimates of project development time have historically been grossly understated. Second, considering the entire Life Cycle of a system, many development costs are currently hidden as future enhancements. This is especially true of methodologies which contractually lock-in the requestor at a point in the development phase. Any changes beyond that point are carried as future enhancements to avoid impacting the development schedule, causing a new system to

bring a dowry of enhancements with it upon implementation. Assuming that prototyping may increase short term development costs by 6% - 10%, there is still the potential for achieving overall Life Cycle savings by eliminating the need for future enhancements. In fact, assuming that 50% of the life cycle costs accrue during maintenance, and further assuming that prototyping can reduce the need for enhancements by 20% to 50%, net savings can still be 7% to 20% (Connell). If the prototype is promoted to production after meeting all performance and documentation requirements, the savings will be even higher, since savings will accrue in the development phase. These additional net savings would accrue from the difference between the traditional coding costs and the more rapidly coded prototype additions, and could approach the 20% of the life cycle costs normally spent on coding. Similar figures are quoted in most literature on prototyping, but they are arguably difficult to document. The difficulties arise because few if any systems have been developed both without and with prototyping, full life cycle costs are usually not available nor accurate, and the benefits of avoiding enhancements are difficult to quantify. The point here is that an increase in the needs assessment and requirements definition costs may still provide a long term payback, and have a high probability of increasing user satisfaction with the delivered system.

2.8. COSTS. As with any tool, there are costs associated with prototyping. The costs incurred are both direct - the cost of the tool, and indirect - potentially longer analysis/design phases. The costs listed below should be weighed against the potential value to be gained by implementing the right system correctly the first time.

2.8.1. Direct Costs.

a. Prototyping Tool Cost. This is a one time cost for a reusable package. Costs will vary greatly by vendor and type of package purchased. In general, a micro-based relational database package with a fourth generation language (4GL) programming tool is recommended. The database will permit rapid creation and maintenance of simulation files, while the 4GL provides the user interface aspects of the prototype. These costs can be mitigated if the prototyping tool is compatible with or included in a Computer Aided Systems Engineering (CASE) tool which has the ability to generate functional code from the prototype.

b. Training/Learning Curve. Although the learning curve for using a 4GL is typically much less than for a third generation language (3GL), training is still recommended, and a period of familiarization should be anticipated. The cost of the training will also vary by vendor and product, but it is typically a one time cost. Depending on the availability and skill of in-house resources, follow on training for advanced use or initial training for new members may also be required.

2.8.2. Indirect Costs. These costs will not be identifiable as

separate entities. Rather, they will add to the estimated development time for the project as additional time required to complete specific phases of the Life Cycle.

a. Development of Mission Needs. Time spent in this area has historically not been charged to the project or has been heavily user driven and will now appear as an automated data processing (ADP) cost. This is due to the involvement of the analyst in assisting with the needs assessment and requirements definitions through prototyping.

b. Iterative Development of Functional Requirements. Additional time and resources may be required in developing the functional requirements since the user will actually experiment with and refine a model of the system being requested.

c. Development of Alternative Designs. Whereas the traditional, detail-design driven approach encourages the development of alternative methods, all too often the process itself is so cumbersome that the only alternative considered is "Do nothing." Prototyping may appear more costly because several models may actually be created to test alternative designs. Thus, a true cost comparison between development of non-prototyped and prototyped systems may not be available.

Chapter Table of Contents

Chapter 3

PROTOTYPING AND THE LIFE CYCLE MANAGEMENT PROCESS

(This page intentionally left blank)

Chapter 3

PROTOTYPING AND THE LIFE CYCLE MANAGEMENT PROCESS

3.1. RELATIONSHIP OF PROTOTYPING TO THE LIFE CYCLE MANAGEMENT (LCM) PROCESS.

a. Prototyping follows and is used in conjunction with the Marine Corps' Life Cycle Management process. Within each phase of LCM, prototyping functions as a miniaturized version of that and dependent phases. In effect, the prototype is also a simulation of the Life Cycle operating in a time-lapse mode.

b. The initial prototype corresponds to the high level Requirements Statement or Functional Requirements Definition.

c. The prototype adds control points, it doesn't eliminate them. This is often a misunderstanding associated with the seat-of-the-pants or if-it-works-its-productional schools of project development. The lack, or disregarding, of controls lends itself to these approaches, and those who espouse these techniques as quicker/better/cheaper than the LCM will often erroneously identify an error-laden version of a program or system as a prototype.

d. The ratios of time spent in various phases of LCM to the total system life often change, as noted above, in costs. Generally, proportionally more time is spent in the analysis and design phases, somewhat less in development, and considerably less in enhancement and maintenance than with traditional methodologies.

e. When using prototyping, all milestones which occur under the LCM without prototyping are still relevant. However, additional milestones are created to control the development and approval of the prototype. These are integrated into the SDM to augment and enhance the traditional development.

f. Walk-throughs are as or more critical to the success of a project with prototyping as to one without. The iterative development of the prototype model is, in fact, a series of small develop and review or walk-through iterations. The user is heavily involved in these, and it is important that the standard rules for walk-throughs (limited to one hour; peers only, not management; document results) be followed. The group size for these walk-throughs may be even smaller than those found for traditional walk-throughs, which often defeat their intended purpose by including such a large group that they cease to function properly. With prototyping, the group may be as small as two -- the requestor and the prototyper, although it is usually advisable to include one or two additional requestors, and another analyst to assist with recording reactions, changes, comments, etc. The intent of these walk-throughs is still to flush out changes, not to immediately resolve them, which may be

a tendency. Trying to respond to requests by altering the prototype during the walk-through may be tempting due to the ease and speed with which the simulation can be changed. However, this practice should be avoided since it detracts from the ability to review all aspects of the current version of the prototype. This often leads the prototypers off on their own tangents which may not be meaningful to the requestor.

3.2. USING PROTOTYPES WITHIN THE SYSTEMS LIFE CYCLE. Prototyping is a communications tool which complements the SDM, but does not replace it. It is a method to enhance communication, but it is not a development methodology. As a process, prototyping can be applied to improve and enhance the SDM by clarifying user needs, verifying the feasibility of a proposed system, developing alternative designs, and validating the final product (system). Prototyping applies within the SDM phases as shown below:

3.2.1. Concepts Development Phase. Using a software prototype to clarify requirements provides a common reference point between the user and analyst. It encourages (even demands) heavy user involvement in this phase, and in so doing encourages interaction and evolution of the requirements. As opposed to the conventional approach to systems development which takes a static view of the users' environment, this approach provides the flexibility to incorporate the changes to the manual process brought by the introduction of the system. Only after an iterative model building process are the requirements and specifications developed and finalized. In this way, prototyping is itself a recognition of the development and change process which has occurred all along in systems development, not a radical change. The difference is that tools now exist which permit inexpensive "experimentation" prior to the development of rigorous specifications and production systems. In effect, the prototype itself serves as a microcosm of the development process.

3.2.2. Design Phase. Prototyping assists in this phase by enabling the designer and the user to develop and evaluate several alternatives. The dynamic development of prototypes can assist in this phase by:

a. Validating and refining the functional requirements of the AIS. Through several iterations of the prototyping process, the model will draw closer to what the user actually desired, not necessarily what was requested or what was perceived by the analyst.

b. Developing working models of the alternatives. This process may uncover additional alternatives which might not have been apparent without the interaction between the user and the model.

c. Providing a basis from which resource budgets can be extrapolated. By identifying the universe of functions which the

system will encompass, the developer can more accurately estimate the overall impact.

d. Determining general and detailed design specifications. By providing a medium through which different approaches can be modeled, much the way an aircraft wing is tested in a wind tunnel, prototyping can assist in the development of specifications and for exploring technical issues. In this regard, one must not overlook the possibility of using a software prototype to model a hardware component.

e. Validating the design against the functional requirements, and the specifications against the design. Even within the same phase, having a physical model provide the opportunity to match written documentation against a model provides a check and balance opportunity.

3.2.3. Development Phase. While reference is made to "throwing-away" the prototype and coding the production system from scratch once the specifications are complete, the model is not actually discarded. Just as the requirements and specifications are matched against the prototype, the prototype serves as a model against which the final product can be evaluated.

(This page intentionally left blank)

Chapter Table of Contents

Chapter 4

PROTOTYPING IMPACT ON THE SYSTEMS DEVELOPMENT METHODOLOGY

(This page intentionally left blank)

Chapter 4

PROTOTYPING IMPACT ON THE SYSTEMS DEVELOPMENT METHODOLOGY

4.1. COMPARISON. The following paragraphs offer a comparison of various aspects of development with and without prototyping.

4.2. SYSTEMS DEVELOPMENT METHODOLOGY (SDM). The SDM is a detail-design driven methodology. It imposes certain artificial constraints in the development of systems by its sequential, or waterfall, process orientation. There are valid reasons for this step-by-step approach, particularly in the tightly regulated environment in which military business systems are developed. However, a problem arises in that the requestor is bound by the contractual nature of the SDM even when the system is not developing as was originally intended or is currently desired. The SDM is tied closely to 3GLs and their inherent inflexibility, and the cost of changing a 3GL system once it has been designed is high. However, the cost of changing the same system after it has been programmed is sometimes prohibitively higher, since the specifications and possibly the requirements, as well as the already developed code, must change. In fact, the cost of change has been demonstrated to increase geometrically as the project passes through various stages of development.

4.3. CONVENTIONAL APPROACH. This approach does not involve prototyping (beyond elementary screen and report formatting), and is the approach most often used under the Marine Corps' System Development Methodology. The following steps are executed sequentially by the persons indicated:

a. Mission Need Statement (MNS) - Defines the need and states preliminary requirements. It is developed by the requestor and approved by the approval authority.

b. Project Management Plan (PMP) - The working document that assists the Project Manager in managing the AIS project.

c. Requirements Statement (RS) - Identifies system functions and performance objectives based on the MNS.

d. Functional Requirements Definition (FRD) - A formal presentation of the user's needs and a hierarchical presentation of requirements.

e. General Design Specification (GDS) - Within the Design process, the first physical grouping of functions for the system.

f. Data Base Plan (DBP) - The full data architecture defining the data structure, access points, and navigation.

g. Detailed Design Specifications (DDS) - Addresses specific software, programming, and hardware environment issues.

4.4. PROTOTYPING APPROACH. The following steps indicate changes to the conventional approach to accommodate prototyping:

a. MNS - Provides basis for initial prototype. This document is still prepared by the requestor. Costs of prototyping during the concepts development phase may be a factor affecting projected costs as shown in this document.

b. PMP - Is modified to include additional prototyping related milestones, staffing, tools, etc. Comments relative to the prototyping process should include its objectives, the CASE and/or 4GL prototyping tools to be used, and the relationships between the final approved prototype and the production system. Depending on the size of the project, a separate Prototyping Plan, as described below, may be required.

c. RS - Expect to add 20% to both duration and cost over a non-prototyped project (see earlier discussion on costs and benefits), more if this is the first prototyping effort and purchase of a 4GL and training must be included. Although the sign-off document for this phase is still the Statement, emphasis is shifted to developing and approving the prototype. Preparation of the Requirements Statement becomes more a matter of defining and documenting what it is that the prototype is showing, rather than interpreting the user's needs and desires (both known and anticipated). Those are clearly demonstrated by the prototype.

In this phase, the initial prototype model is constructed, and proceeds through a series of change/review iterations until it accurately depicts what the users envision their system accomplishing. The Requirements Statement itself is not written until after the user approves the prototype. However, changes to the prototype are documented as the prototype is developed. This documentation serves as the basis of the Requirements Statement. It also serves to control the prototype process to prevent unintentionally rehashing already resolved issues.

d. FRD - Development of this document involves additional change/review iterations of the prototype as the full set of the user's functional requirements unfold. Again, the document is not written until the prototype has been approved. At this time, the prototype should be demonstrating all of the business functions required of the system by the user.

e. Prototyping Plan - Bearing in mind the desire to keep the prototype as inexpensive, flexible and responsive to change as possible, the Prototyping Plan should focus on the goals to be attained, milestones, timeframes, and schedules without being unduly restrictive. Since prototyping may be used in several phases, the Prototyping Plan should document its intended use in all applicable LCM phases. A suggested Prototyping Plan content description is provided in Appendix C.

4.5. IMPACT ON SYSTEMS DEVELOPMENT METHODOLOGY. Prototyping is used in conjunction with the standard SDM to augment the user's participation in and understanding of the requirements, conceptual and detailed design stages. However, the use of prototyping in this manner tends to blur the distinction between the traditional stages. The three levels of prototyping respectively bring forward components from the conceptual design to requirements definition, detailed design to conceptual design and development to detailed design.

With the traditional linear design methodology, screens and reports are designed during the conceptual design. However, the first level of prototyping introduces the design of screens and reports as an integral component of the requirements definition stage.

In the linear approach the requirements definition stage is concerned with what should occur while, the conceptual design stage focuses on how it should be done. Prototyping combines the design of the system's externals with the development of user requirements. The functions of the system are not developed separately from the methods of performing them. The defining of business processes is impacted by the introduction of on-line screens and is dependent on the user's interfacing with and operation of the system.

The second level of prototyping introduces components of the traditional detailed design stage to the conceptual design stage. This level augments the design of the file structures through the user's manipulation of screens and screen flows. The simulation of and interaction with files requires a level of design detail not normally encountered until detail design.

The third level of prototyping merges detailed design and development. It requires that a working model be developed. The prototyping of a subset parallels the phased design and development for a large system.

4.5.1. Impact on Development Personnel. Prototyping provides an opportunity for programmers to become involved in the design stages. In the development of a large system following the traditional development cycle, the programmers are often not the same personnel who designed the system. Prototyping provides an opportunity for the programmers to become more familiar with the system through active participation in earlier stages.

4.5.2. Impact on User Acceptance Testing. Through their involvement in the design and development of the system, the users will have gained significant knowledge of the system and its capabilities. The users will not be facing an unknown entity at acceptance testing. Since they will understand how the system operates, they will be able to test the operations of the system, rather than first having to familiarize themselves with its concepts. With prototyping, the users confirm the operations of a system for which they designed the user interfaces.

4.5.3. Impact on Documentation. The documentation required for Life Cycle Management under MCO P5231.1B still applies in a prototyping environment. The model is meant to provide the correct input to these documents to ensure development of the correct system. Prototyping cannot be viewed as a loophole to avoid structured design techniques and requirements. This is not to say that prototyping might not impact more than just the quality of the documents produced. The possibility exists that prototyping, by reducing the need for enhancements, might reduce total life cycle costs enough to place the program in a lower cost threshold category. This would eliminate the requirement for certain documents. For example, an Abbreviated Systems Decision Paper would replace SDP I and the associated required documentation if the life cycle cost can be reduced below $1 million through the use of prototyping.

4.5.4. Impact on Timing of Deliverables. There is much less emphasis on finalized, complete requirements and specifications while developing the prototype. However, upon approval of the finalized version of the prototype and prior to proceeding with the next milestone, documentation specified under the SDM must be completed, and required approvals obtained. This is necessary due to the contractual nature and high cost of developing the production system. To reemphasize, the purpose of the prototype is to help ensure development of the correct system, not to avoid documentation.

Delaying the development of completed documents until the corresponding version of the prototype is approved is necessary for the following reasons:

a. Changes are minimized. Documentation which has not been finalized until after the prototype has been approved does not create the burden of constantly updating and keeping versions of the changing specifications. This is not to say that nothing is written concerning the model - quite the contrary, each change should be noted and checked off when added to the model. The difference being the speed at which the prototype is developed both encourages change and discourages a lengthy approval process. The proper amount of documentation at any point is that amount which will not inhibit changes to the prototype by requiring changes to the documentation.

b. Emphasis is on rapid turnaround. A primary goal of prototyping is to take advantage of the requestor's knowledge and expertise at a time when interest in the project is high. The most enthusiasm for a project is usually generated at startup and, if things have gone well, at implementation. With traditional development methods which isolate the requestor from the development process, interest usually wanes during the long analysis and development phases. Rapid turnaround is intended to make the most effective use of the requestors time and to generate and maintain momentum while interest is still high. The emphasis is on spending time quickly developing something which the users can touch and feel, not on the laborious process of

formal, contractual type documents which are then packaged and delivered for approval, isolating the requestor from the process.

c. Development of the prototype is interactive. Prototyping involves the users with analysts in an interactive development mode while creating the model. Historically, this has been recognized as a key factor critical to project success. Without this approach, the requestor is expected to visualize all implications and ramifications deriving from a set of written (and/or diagrammed) requirements and specifications. In the interactive mode with a dynamic prototype, the user can actually work with a model derived from the requirements, and the programmer can see the intent of the analyst's specifications. In each instance, the opportunity to correct errors and misunderstandings before they are cast in concrete is obvious.

4.6. INTEGRATION OF PROTOTYPING INTO THE SYSTEMS DEVELOPMENT METHODOLOGY. Prototyping is an iterative process which builds an intentionally incomplete model and iteratively develops it to demonstrate the business functions intended for the system. This section will provide a synopsis of the recommended prototyping process. Since this process is antithetical to a conventional SDM, this section will also show the integration of prototyping steps into the Marine Corps SDM. Lastly, differences in project management when using prototyping will also be discussed.

4.7. TOOLS. Prototyping requires software tools that allow designers or programmers to create a working system in a very short time. The following types of tools are used:

a. Data Management Systems (DMS) with an Accompanying Non-Procedural Language: A DMS allows the user to specify what needs to be done rather than how the work must be performed. A non-procedural language provides this user interface. It is an internally driven prototype which requires a database and data elements to be created prior to developing screen scenarios. The iterative process requires modifications to be made to the data-base and elements while the screen formats and flows are refined. Additionally, a DMS can have many useful functions already coded, the user need only specify the parameters of the operation and the files to be used. Numerous functions can be performed quickly and easily to set up a working prototype. These functions include allocating storage space for the data, creating data definitions, select and sort, retrieving answers to queries, and report generation. Some DMS also provide facilities for defining formats for data entry on a cathode ray tube (CRT) screen. The designer need only enter the appropriate titles in the location where they should appear on the screen and then indicate data entry field locations, field lengths, and field validation criteria. DMS also have file management facilities so that updating the files is done easily and usually without writing code. The DMS may also take care of the security and recovery aspects of the database so that the programmer is left with more time to concentrate on the design of the prototype, rather than coding routine functions.

b. Application Development Systems with an Accompanying Procedural Language: The term 'application development system can be confusing because it is used to refer to different types of tools. These are tools designed for programmers--tools that would rarely be employed by typical end users because they require a procedural language, such as COBOL. It is used on an externally driven prototype that is dependent upon user interface. They do not require a data base to be established in order to develop the scenarios. The iteration process requires changes be made only to screen formats and flows, and not to an internally supporting database or data element. A DMS, on the other hand, is designed so that it can be used by end users. However, a DMS can also be, and often is, used by programmers as well. Application development systems can provide many of the same facilities as the DMS, though use of a procedural language is generally required.

c. Application Generators: Program generators have been around for some years, in fact numerous user companies have written their own. An application generator has a broader scope than a program generator. Many of the vendors have enhanced their products to include: a data dictionary, on-line interactive use, screen format generator, and other DMS-like facilities for developing on-line applications. Generators were originally designed for data processing professionals but some have been enhanced to the point where end users can now use them. These generators produce procedural language source code. Unlike the DMS, the resulting application does not run under them. The programmer selects from a menu what type of module he wishes to create. The system then initiates a question-and-answer dialogue, in which the programmer supplies the variable information. When this form of coding is complete, the generator produces the actual source code. It can generate 100% of the code for standard file entry and update programs, including formatted input screens and output reports. For more specialized applications, logic source code can be hand coded and added to the generator's code; therefore, generators can also provide a means for creating a prototype quickly.

d. Libraries of Re-usable Code: One of the points that has been brought out by programmers is that many of the DMS and other prototyping tools encourage the use of re-usable code. Modules can be quickly retrieved from storage, changed if necessary, and then used in a new program. By creating an on-line library of such modules, programmers can more rapidly create a prototype.

4.8. PROTOTYPING SKILLS. The prototyping environment does require different skills for the data processing professional. With prototyping, interviewing skills are not as important as they are with conventional methods. This is because user specifications are no longer based on how well the analyst interprets the user's spoken requirements; instead, the specifications are based on the demonstrated working prototype. Prototyping also requires the data processing professional to spend more time with users and less time coding; hence, the

professional must be more people-oriented. Programmers who prefer the "art" of programming might not particularly enjoy that.

4.9. AVOIDING PROBLEMS. There are potential problems associated with the use of any tool, and prototyping is no exception. The following paragraphs describe some of the more common problems and pitfalls which may befall a prototyper, especially one relatively new to the process.

4.9.1. Promoting the Prototype to Production. Due in large part to bad experiences with cost overruns and poorly functioning systems developed under the guise of prototyping (e.g. without documentation), there is strong resistance to this practice. In the absence of overwhelming reasons to the contrary, promotion to production should not be viewed as the developmental approach of choice for the following reasons:

a. By definition and semantics, a prototype cannot be promoted to production since the prototype does not include the required LCM documentation. The prototype is a model which helps develop requirements and specifications. It is intentionally kept lean in terms of formal documentation and control mechanisms to encourage changes and enhancements to the requirements. Although it serves as a working model of what the final system will be, and is an aid in developing the Requirements Statement, the Functional Requirements Definition, and the General and Detailed Design Specifications, the prototype itself does not contain the documentation required by reference (a), nor will it contain those controls necessary to ensure system integrity. To add these elements to the prototype in order to make it productional makes it inflexible, and therefore not a prototype. Once the requirements and specifications have been included, the model is too inflexible to be considered a prototype.

The major issue here is that prototyping (i.e., quick and dirty programming) has been used as an excuse to avoid formal documentation of requirements, specifications, design, and coding in the past. While this occasionally shortened the initial development times, the savings were often short-lived and paid for by future generations of programmers forced to patch work-arounds into undocumented, 3GL spaghetti code for enhancements which should have been, but were not, foreseen.

b. Evolving model code into production after documentation requirements are met is not inherently bad. However, this practice is more difficult to control. It may impact on the ability to rapidly turn around prototype changes when the intent is really to produce a mini-production system, not a prototype. Conversely, prototyping may become nothing more than unstructured development as stated above.

c. Once the documentation is complete and accepted, if it can be conclusively shown that the 4GL code used in the prototype meets all functional, performance, and control requirements, a

waiver to promote the 4GL to production may be requested.

4.9.2. Premature Evolution to Production. Even if it has been decided that the prototype cannot satisfy all of the functional needs and performance requirements, there will be a tendency to promote the model to production immediately. That is, the user will want to begin using it on a limited basis. This usually occurs under certain conditions:

a. The user likes the model too much to give it up. This is a particular problem if the prototype has stimulated business process changes which have been implemented within an area, and the model has been integrated into the normal flow of business. This is an argument against turning a prototype over to the user for ongoing evaluation, and for tight controls governing the development process. An analysis of the ability to meet all requirements, including performance requirements and the ability to incorporate enhancements, will generally discourage this idea. Discussion of the risks involved, especially in exceeding capacity and degraded response times, as more productional demands are made on the prototype, may also serve to discourage this practice.

b. The developer has demonstrated a solution to the problem and wants to move on to other challenges. The likelihood of this occurring is slim on large projects with multiple member teams, but has been a problem associated with 4GL's and should be considered. Proper goal and expectation setting can help avoid this problem.

c. The user is not satisfied with the results of the prototype, or with the prototype itself. There may be a tendency to avoid scrapping the prototype and starting over, or dropping the project entirely even though it is unworkable. Care must be taken to differentiate between characteristics of the model (e.g. response time, lack of controls), and of the functions being demonstrated. If the user is not satisfied now when the prototype can be changed easily, there should be no reason to expect satisfaction later when the system has been developed and is much more costly to modify.

d. Issues previously resolved resurface. This might be an indication that they were resolved to the prototyper's but not adequately resolved to the users' satisfaction initially. Another possible cause is that a resolution does not exist. If a review of the requested prototype changes indicates a loop of this nature, it may indicate that there is no resolution, and the project cannot be completed, at least to the users' satisfaction. Better to recognize this early on in the development, than at the very end after considerable expenditure of development time and money. In fact, identifying the limitations early on may lead to the users acknowledging them and altering their expectations.

Chapter Table of Contents

Chapter 5

PROTOTYPING PROCEDURES

(This page intentionally left blank)

Chapter 5

PROTOTYPING PROCEDURES

5.1. PREMISES TO PROTOTYPING. When deciding whether to include prototyping as part of the systems development life cycle, the systems analyst needs to consider what kind of problem is being solved, and in what way the system presents the solution. There are three general conditions that should exist before a prototyping development process begins:

a. The Number of Users is Limited. Ideally there will be a single user or a small group of users tasked with evolving the prototype. This group should be localized in the sense that the work in the same organizational group and within the same physical location, yet it should be representative of those who will ultimately use the system. It is less likely that prototyping will succeed on a project with diverse users in different organizational groups and different physical locations.

b. The Data Model Exists or can be Easily Created. This assumption is safe in an environment where (1) the user wants a new system that will interact with an existing database, and (2) the MIS organization has already developed an information model of that database, complete with data dictionary definitions, entity-relationship diagrams, and so forth. For a completely new system with a completely new database, it may still be possible to derive the information model fairly quickly and create a data dictionary that can serve as the foundation for the prototype.

c. The Application Scope is Finite. With prototyping, as with any development, the bounds of the effort must be established, communicated, and understood. The purpose of the prototype is to define the system within the established bounds. Care must be taken when developing the prototype that the scope is not intentionally expanded.

d. The Application is Modular. Prototyping will work best when applied to small to medium applications, or discretely defined functions of a large application. It is the developer's responsibility to determine the optimum size of the function to prototype. The guidelines for this determination are that a prototype: a) Model a complete, recognizable function or set of functions; b) Incorporate only as many functions as can be easily viewed, understood, and demonstrated in a single session. If a system is so large that a single prototype becomes unwieldy, several smaller prototypes should be used to demonstrate the functions. Partitioning the prototype in this manner may also serve to identify logical boundaries for the phased implementation of a large AIS.

5.2. PROTOTYPING PROCESS. Prototyping involves a somewhat different outlook from that of the conventional development cycle. In the conventional cycle, user requirements are defined

in a specification document which is approved and signed by the user. Only after that phase is completed and approved does the design of the system begin. In prototyping, the requirements and in some aspect the design, evolve together through the iterative prototype development process within the Requirements Definition Phase. The development of the prototype deals with both the requirements from the user's view and the design from the designer's view, allowing the user to view how the designer interprets the stated requirements. There are four steps in the prototype development process, depicted in Figure 5-01, with the last two repeating until agreement is reached that the prototype accurately represents the intended AIS.

5.2.1. Step 1: Identify Users Basic Requirements. The first step involves uncovering only the user's most basic and evident requirements. While the size and complexity of the project determine the amount of time spent on this step, the objective is always to develop a first cut prototype within a matter of days of meeting with the user to establish initial requirements. For smaller projects, the designer generally talks with a few users for a short time to find out what the problem is and what they expect the system to do. Together they may define the information to appear on certain reports and data input screens. The designer might also want to get some statistics about the expected volumes of the various types of transactions the system will process. This initial requirements study should be informal and need not involve formal written specifications. For larger systems a design team may need to spend some time interviewing several users to more fully understand the overall scope of the system. However, regardless of final system size, the objective of this step is to determine just enough about required system functions to permit rapid development of a very basic prototype. Remember that the objective is to evolve the prototype, not pre-specify all of its functions.

5.2.2. Step 2: Develop a Working Prototype. The important point in prototyping is that the designer takes the notes developed in the user discussions and quickly creates a working system. For example, when using DMS, the designer can use the default values in the report generator to create standard report formats, rather than define the layouts of each report. Also, the prototype need only perform the most important functions. The designer creates a relatively small file of data, with just enough differentiation to allow the users to experiment with some sample screens. The prototyper should be able to develop this first prototype, which is the beginning of the first of three levels, within a few days. Each level can be used successively to contribute to the design and development of the system. In developing the prototype data files, the objective initially is ~~to~~ get the model up and running, not to model a database. However, later on in the development process, the prototype can certainly be used to evaluate performance characteristics of alternative database designs.

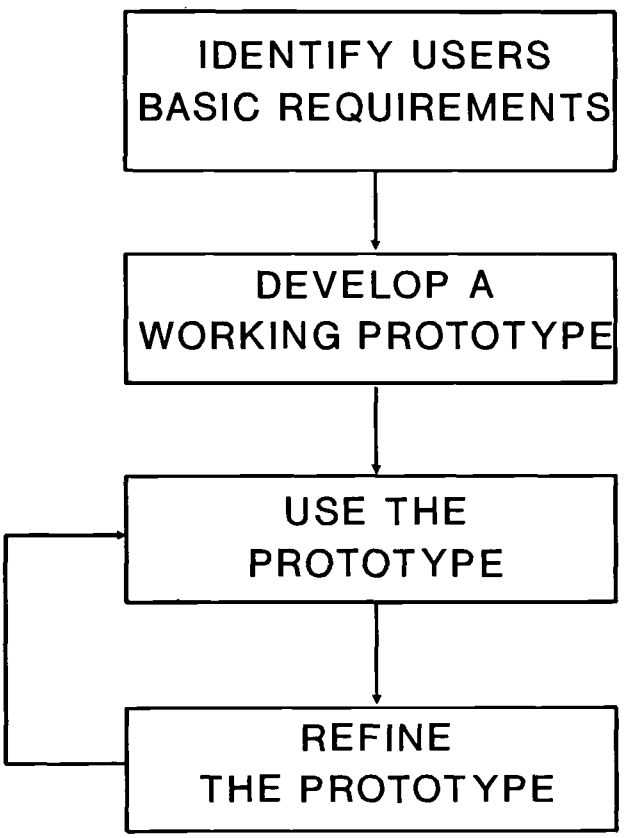

FIGURE 5-01
Four Step Prototyping Process

a. Levels of Prototyping. There are three levels at which prototypes can be developed. While it is not always necessary to develop all three levels, they logically fall into the following order from a user perspective:

(1) Level 1: On-line screens. The objectives of this level are to determine the screen and report layouts and contents, the screen flow sequence and the method for sequencing the screens. The systems analyst would develop scenarios to illustrate the basic processes of the application. Several iterations would be performed until the users were satisfied with the screen representations and screen logic flow.

(2) Level 2. Introduction of database interactions and data manipulation. The main purpose of this level is to demonstrate the operation of key areas of the system. Users would enter specific sets of transactions, upon which simulated processing would be performed. Features such as error handling could also be incorporated. Level 2 prototyping assists in verifying the first level screen design and screen flows, and clarifying application processing logic.

(3) Level 3. An incomplete working model of the system. It is a subset of the system in which application logic transactions and database interactions operate with real data. The objective at this level is to develop a model that will serve as the basis for the systems specifications. This objective is attained by allowing the users to experience the interaction with a live system. Level 3 is distinguished from Level 2 by the inclusion of processing logic. At this level, prototyping can be used to present the entire environment including business systems, manual procedures, system interaction and training. It is useful in determining and correcting problems associated with: 1) awkward, incomplete or wrong procedures; 2) disruption of the work flow; 3) interfacing and interaction with the system; and, 4) preparing staff for the system. The use of the third level prototyping within the System Development Methodology is encouraged.

b. Determining the Appropriate Level. Some level of prototyping is recommended for every project. Many systems analysts have used Level 1 for years, (i.e., Output mockups of printed reports and/or on-line screens). By using either level 2 or 3 prototyping, the traditional, linear development approach can be dramatically enhanced. Both levels 2 and 3 resolve uncertainty during system design. The appropriate level is that which achieves the minimum uncertainty in the specifications.

5.2.3. Step 3: Using the Prototype. This step begins with the designer demonstrating how the prototype works to a small group of users. During the presentation, the users may request some changes. Some of these may be made right then--others can be reserved for later. If the system is not really what these users need, the designer may discard the prototype and start over again. While initial iterations should be demonstrated by the

developer, at some point the developer may teach the user how to use the prototype. This will permit experimentation and detailed evaluation. This period of experimentation should have a pre-determined time period, since the objective is to promote interactive development, making changes as they are requested.

5.2.4. Step 4: Refining the Prototype. The designers and users discuss the desired changes, and decide which ones should be included in the next version of the prototype. The designer then creates that version--either by starting over again or by extending the current version. After the changes have been made, the cycle returns to Step 3--the designer demonstrates the new version, instructs the users on its operation, and lets them use it for awhile. Steps 3 and 4 are repeated until the system fully achieves the requirements of this small group of users.

5.3. OBTAINING APPROVAL OF THE PROTOTYPE. Once the prototype satisfies the primary user, approval should be obtained before proceeding with finalization of the specifications. The recommended approach is to first introduce the prototype to a larger group of users for their experimental use to see if additional requirements come to light. Once a representative sample of users is satisfied with the prototype, it can be demonstrated to management to gain approval for the production version. The recommended approach to this demonstration is to have managers bring questions to the demonstration on what they would like to see the system do, rather than presenting an in depth demonstration of all functions. In this session, the primary user contact, or those already familiar with the prototype, should conduct the demonstration, rather than having the managers operate the system themselves.

(This page intentionally left blank)

Appendix A

REFERENCES

1. SecNav Inst 5231.1B, "Life Cycle Management (LCM) Policy and Approval Requirements for Information System (IS) Projects," 8 Mar 1985.

2. MCO P5231.1B, "Life Cycle Management for Information Systems (LCM-IS) Projects," 22 Mar 1990.

3. Connell, J. L. and Shafer, L. B., "Structured Rapid Prototyping: An Evolutionary Approach to Software Development," Yourdon Press 1989 ISBN 0-13-853573-6.

4. Canning, R. C., "Developing Systems by Prototyping," EDP Analyzer, Sep 1981, Vol 19, No 9 Canning Publications, Inc.

5. Melde, J. E., Overmyer, S.P., and Trowbridge, T. L. "Simulation and Display Prototyping in the Design Of Man-Machine Systems," TRW Defense Systems Group, Colorado Springs, Co.

6. Harrison, R., "Prototyping and the Systems development Life Cycle," Journal of Systems Management, Aug 1985 pp 22 - 25.

7. Fisher, G. E., "Application Software Prototyping and Fourth Generation Languages," NBS Special Publication 500-148, U.S. Dept of Commerce, May 1987.

8. Johnson, J. R., "A Prototypical Success Story," Datamation, Nov 1981 pp 251 - 256.

9. Davis, W. S., "Systems Analysis and Design: A Structured Approach," pp 210 - 221.

10. Housman, D., "Rapid Prototyping," Federal Computer Week, 14 Aug 1989 pp 32 - 34.

(This page intentionally left blank)

Appendix B

GLOSSARY

3GL - Third Generation Languages (e.g. ADA, COBOL)

4GL - Fourth Generation Languages (e.g. Natural, FOCUS)

ADP - ADPE Support Plan

ADPE - Automated Data Processing Equipment

AIS - Automated Information System. A combination of information, computer, and telecommunications resources, and other information technology and personnel resources which collects, records, processes, stores, communicates, retrieves, and displays information.

CDPA - Central Design and Programming Activity

CRT - Cathode Ray Tube

DBMS - Database Management System

DDS - Detailed Design Specification

DMS - Data Management Systems

FRD - Functional Requirements Definition

GDS - General Design Specification

IRM - Information Resources Management

IP - Implementation Plan

LCM - Life Cycle Management. A management discipline for acquiring and using AIS resources in a cost-effective manner throughout the entire life of an AIS.

MCDN - Marine Corps Data Network

MNS - Mission Need Statement

PRS - Prototyping Standard

RASC - Regional Automated Service Center

SDM - System Development Methodology - A collection of methods, procedures, and activities associated with developing an automated information system.

(This page intentionally left blank)

Appendix C

PROTOTYPING PLAN CONTENTS DESCRIPTION

Section 1. USE OF PROTOTYPING

A discussion of how prototyping will be used throughout the development of this AIS.

Section 2. PROTOTYPING TOOL

Specification of the particular tool selected for use to develop the prototype. If the tool is not already available to the developer, this section should contain justification for the choice of this particular tool over another, especially if another is readily available.

Section 3. TRAINING

Plans for providing training to developers using prototyping or this prototyping tool for the first time.

Section 4. GOALS

A description of the specific goals which the prototype is expected to achieve. This section is intended to provide direction to define the scope of the prototype, help keep the process on track, and help set the user's expectation level. The latter may be formed in a discussion of how the model developed in prototyping will relate to the final product.

Section 5. USER INVOLVEMENT

A statement of the level of involvement required by the user in developing the prototype. It is extremely important that the user commit the appropriate resources to this process to ensure success of both the prototype and the AIS.

Section 6. PRELIMINARY MILESTONES AND SCHEDULE

Particularly valuable where a large AIS has been segmented to facilitate prototyping, this section serves to identify known functions to be prototyped, and provides a plan for coordination and control of the process. It must be realized that this plan will undoubtedly change as the prototype develops and expands to include more functions (within the scope of the project).

(This page intentionally left blank)

COMMENTS/REVISIONS

Technical publications under the Information Resources Management (IRM) Standards and Guidelines Program (MCO 5271.1) are reviewed annually. Your comments and/or recommendations are strongly encouraged.

IRM Tech Pub
Name: _____

IRM-_____-___ (Number) Date of Tech Pub: _____

COMMENTS/RECOMMENDATIONS:

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

Name/Rank:
_____(optional)

Unit:
_____(optional)

Mail To: Director
Attn CTAS
MARCORCOMTELACT
3255 Myers Ave
Quantico VA 22134-5048

ELMS - GICISP:MQGMCCTA