1994-09

A comparative study of commercial and Department of Defense strategies for developing software applications

Clancy, Gregory A.

Monterey, California. Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

A COMPARATIVE STUDY OF COMMERCIAL AND DEPARTMENT OF DEFENSE STRATEGIES FOR DEVELOPING SOFTWARE APPLICATIONS

by

Gregory A. Clancy

September 1994

Thesis Advisor: **James C. Emery**

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE September 1994	3. REPORT TYPE AND DATES COVERED Master's Thesis, Final

4. TITLE AND SUBTITLE A COMPARATIVE STUDY OF COMMERCIAL AND DEPARTMENT OF DEFENSE STRATEGIES FOR DEVELOPING SOFTWARE APPLICATIONS	5. FUNDING NUMBERS
6. AUTHOR(S) Clancy, Gregory A.	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited	12b. DISTRIBUTION CODE A

13. ABSTRACT *(maximum 200 words)*

A focus on information system application development is on the rise as users become more familiar with the computing environment and the business advantages it gives the organization. Enormous software development backlogs and increasing demand for application software is forcing information system managers to look at new and innovative ways to develop and maintain software. High-level languages and tools are being introduced into organizational information system development environments. Software languages and tools that are being used to build systems quickly and effectively by leading-edge organizations are fourth-generation languages, computer-aided software engineering tools, and object-oriented technologies. Results of a survey of 23 information system executives that accompany this thesis provide evidence that organizations are moving rapidly toward these languages and tools, and continue to shift their emphasis away from older conventional development methodologies and line-by-line coding of procedural programming languages. The Department of Defense should revise its own policies and practices where appropriate to conform to the clear trends emerging in leading private-sector organizations.

14. SUBJECT TERMS Application Software Development	15. NUMBER OF PAGES 66
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

Approved for public release; distribution is unlimited.

A COMPARATIVE STUDY OF COMMERCIAL AND DEPARTMENT OF DEFENSE STRATEGIES FOR DEVELOPING SOFTWARE APPLICATIONS

by

Gregory A. Clancy
Lieutenant Commander, United States Navy
B.S., South Dakota State University, 1981

Submitted in partial fulfillment
of the requirements for the degree

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY MANAGEMENT

from the

NAVAL POSTGRADUATE SCHOOL
September 1994

Author: [redacted]
Gregory A. Clancy

Approved by: [redacted]
James C. Emery, Thesis Advisor

[redacted]
William B. Short, Second Reader

[redacted]
David R. Whipple, Chairman
Department of Systems Management

ABSTRACT

A focus on information system application development is on the rise as users become more familiar with the computing environment and the business advantages it gives the organization. Enormous software development backlogs and increasing demand for application software are forcing information system managers to look at new and innovative ways to develop and maintain software. High-level software languages and tools are being introduced into organizational information system development environments. Software languages and tools that are being used to build systems quickly and effectively by leading-edge organizations are fourth-generation languages, computer-aided software engineering tools, and object-oriented technologies. Results of a survey of 23 information system executives that accompany this thesis provide evidence that organizations are moving rapidly toward these languages and tools, and continue to shift their emphasis away from older conventional development methodologies and line-by-line coding of procedural programming languages. The Department of Defense should revise its own policies and practices where appropriate to conform to the clear trends emerging in leading private-sector organizations.

TABLE OF CONTENTS

I. INTRODUCTION

A. BACKGROUND

One of the problems in software development is the creation of specifications for complex systems. Specifications that define user requirements are normally full of inconsistencies, ambiguities, and omissions. The conventional systems development life cycle (SDLC) is a software development process that demands user requirements be specified up-front, before development of an information system can continue. DoD continues to emphasize a version of the conventional (i.e., "waterfall") model to develop information systems. This conventional model for software development is not flexible enough and is not adaptable to changing user requirements.

The literature shows that despite the efforts to modify the conventional process, information systems are still delivered late, over budget, and at variance with users need. Data collected for this paper indicates that Chief Information Officers (CIOs) do not view the conventional model as a viable option for application development. CIOs are expecting a prototyping method of software development or a new development paradigm to give them adaptability and flexibility in software development. They expect applications to be developed quickly, using high-level languages and tools.

The trend in application development has moved toward using more powerful programming languages and development tools such as fourth-generation languages, CASE tools, micro-based development tools, and object-oriented technology. To replace specific business applications of organizational information systems, organizations are purchasing application packages that contain the required functionality to assist them in performing their business tasks.

Organizations are looking to train and hire software development personnel with additional skills beyond the traditional third-generation (3GL) programming skills and systems analysis/design experience. Organizations are seeking data communications

specialists, specialists in PC-based products, database specialists, and specialists skilled in specific business areas.

B. THESIS OBJECTIVE

The objective of this research has been to study the DoD and commercial industry strategy for software development by examining software development methodologies, languages, and tools used for software development, as well as the organization of software development. The overall objective of the thesis is to provide DoD with views and practices of application development that are now being used by the private sector.

C. METHODOLOGIES

A literature search was conducted to provide insight into background information on application development methodologies and software technologies. A survey, given in the Appendix, was developed and distributed to senior executives who participate in the strategic planning of application development for their organizations. The senior executives, who are all serving (or have served) as CIOs or equivalent executive officers, were specifically selected because they are recognized as leaders in information technology. The confidentiality of the survey participants has been reserved in order to obtain the maximum number of object responses.

The survey was sent by fax to 31 executives; 23 responses were received. Approximately one-half of the completed surveys were faxed back, and the other half of the responses were obtained over the telephone. Obtaining the survey responses over the telephone provided a valuable opportunity to interview some of the respondents and allow them to expand on the reasoning behind their responses to questions in the survey.

D. SCOPE AND LIMITATION OF STUDY

This research was designed to obtain trends in application development. The number of completed surveys was not enough to perform a traditional statistical analysis. The questionnaires do, however, provide valuable clues about the views of top-level

executives to determine trends in software development methodologies, application development, professional skills necessary for software development, and the role of legacy systems in their organizations.

E. ORGANIZATION OF THESIS

Chapter II gives a brief history of software development in DoD information systems. The chapter then continues with DoD's strategy for exploiting information technology as described by the Corporate Information Management (CIM) initiative and the Technical Architecture Framework for Information Management (TAFIM). The chapter concludes with a discussion on software development environments.

Chapter III discusses issues in the software development environment: software development methodologies and the software organization. The Software Development Life Cycle (SDLC), prototyping, and object-oriented programming methods of development are analyzed. A section of this chapter discusses options for dealing with the hundreds of legacy systems in DoD's inventory.

Chapter IV surveys application development from 23 senior executives in the information technology field. The results provide tabulated data in the areas of software development methodologies, legacy systems, developing and maintaining information systems, and professional skills needed for the software development organization. Relatively little information is available about software development strategies of leading private-sector organizations. Because of DoD's professed intention to emulate best practices in industry, the survey data should provide valuable new information for DoD policy makers.

Chapter V provides conclusions to the collected survey data and recommendations for further research.

II. LITERATURE REVIEW

Developing and maintaining information systems in DoD has been a difficult process that often results in a product that is not acceptable to the user. DoD has recognized its shortcomings in applying information technology management, and has established new programs to make better use of information technology and eliminate duplication--especially the Corporate Information Management initiative (CIM). The Technical Architecture Framework for Information Management (TAFIM) has also been formulated to guide managers in the development of technical architectures to meet DoD mission requirements.

In order to gain a better insight about DoD's current position on software application development, it is important to have a broad perspective of DoD policy and guidelines in information technology development and acquisition. This chapter briefly discusses the CIM initiative and the TAFIM, as DoD's policy and guidelines for DoD development and acquisition of information system resources. The chapter then continues by discussing trends in information systems and concludes with a discussion on software technologies that have been used to create information systems by DoD and the private sector.

A. HISTORY

Over the past three decades, there have been rapid technological advances in computer hardware. The benefits of using the hardware, however, have been hindered by problems of software development and maintenance. Throughout this period, the rate of improvement in software development and maintenance has fallen far behind the rate of advance in hardware technology, continually widening the gap between the power of the hardware and our ability to use it effectively (Jones 1991).

There are a number of reasons for this gap: software is difficult to specify, debug, and maintain; there has been little progress in new methodologies for developing software; projects have become increasingly larger and more complex; and organizations do not

know or understand how to deal with the organizational and cultural change required to successfully introduce new technologies and methodologies.

DoD, as well as the private sector, have had their share of problems specifying the requirements necessary to build an effective information system. As a result, the delivered system is often delivered late and over budget, and is not what the user needs.

1. DoD Automated Information Systems Development

The exploitation of information technology within DoD has not been without its problems. Requirements from users for additional functionality and integration have grown at a more rapid rate than can be satisfied by existing software engineering practices. As a result, software now constitutes by far the most serious bottleneck to exploiting information technology. Without major improvements in the process of developing and maintaining software, the "software crisis" can only grow worse. (Emery, McCaffrey 1991)

Adding to problems associated with the development of software, DoD has had its share of difficulties in Automated Information Systems (AIS) acquisition. A report to the 101st Congress (1989) reviewing eight DoD AIS reported that AIS projects:

> ...experienced significant cost growth, some in the hundreds of millions of dollars. Four of the eight systems have been in development for the last 8 years and two of the system's development efforts were abandoned after $237 million. ...and have been delayed by 3 to 7 years and none of the systems are scheduled to be fully deployed until the 1990s.

Solving these problems within DoD is a difficult task at best. The Corporate Information Management (CIM) initiative is a start at formulating a strategic plan that addresses these issues.

2. CIM Strategy

The critical role of information technology in DoD is widely recognized in the development of AIS and weapons systems. Information technology will play a pivotal role as a "force multiplier" during a time when the force structure is being reduced dramatically. Given the austere budgeting environment and the exploding cost of AIS

development, military leaders are challenged with accomplishing more with less. To accomplish this, they must identify and eliminate waste and make improvements in quality and efficiency.

To deal with the problems of information technology development and acquisition, DoD in 1989 initiated CIM to improve its business practices, make better use of information technology, and eliminate duplicate information systems. The program is designed to eliminate redundant information systems and software from distributed administrative areas, and consolidate common applications into centers that operate under standardized data usage.

DoD's CIM strategic plan lays out six basic goals to ensure that DoD is able to exploit information technology in an era when funding has been reduced and consolidation is an organization norm: (Endoso 1994)

- Re-invent and re-engineer DoD's processes.
- Couple DoD organizations together through common, shared data.
- Minimize duplication and enhance DoD's information systems.
- Implement a flexible, worldwide information infrastructure.
- Apply CIM to integrate DoD-wide operations.
- Establish a CIM policy and management structure.

These are formidable tasks. DoD has hundreds of information systems that support a myriad of business practices, supported by an entrenched culture that adheres to long-standing practices and traditions. For many years within DoD, individual services and subunits have developed and operated their own AIS to perform equivalent functions. As an example, DoD had more than 30 different automated systems to support civilian payroll. Consolidating and integrating those systems will be a major undertaking.

3. Information System Trends

DoD and commercial businesses have historically used information technology in an attempt to make the organization more productive and efficient. Computers and application software were brought into the organization without managers understanding the role the information systems would perform for the business. It was believed that adding information technology to the business environment would make the organization more productive and effective. In essence, the organization was expected to adapt to a host of technological improvements for the sake of technological advance, and many AIS projects did not help the organization achieve its desired goals. The emphasis of information technology was placed on how well the information systems functioned with respect to processing efficiency and performance reliability, regardless of the particular functions the information systems performed.

The 1980s brought a new business vision that looked at how information technology could be used to support the organization. Customized management reports generated by management information systems (MIS) and expanded database management systems (DBMS) are examples of systems that have been used to give management better information and consolidate data for the organization.

This new vision focuses on the strengths and capabilities of the organization, along with the business opportunities that the organization encounters, and assesses how information technology can be used to bring about that vision (Senn 1990).

Organizations now look at their business processes and tailor information systems to support their business objectives. This process, which has been called "business re-engineering," allows organizations to identify the functions that are important to the success of the organization and then apply information technology to assist the organization in meeting those functions.

Because of this trend in applying information technology to help organizations accomplish their business goals, the complexity of delivering and maintaining information systems has increased. The following are some of those trends (Emery, McCaffrey 1991):

- Organizations increasingly view MIS as an integral contributor to their business strategy and as a primary vehicle for implementing improvements in critical operational activities.
- There seems to be few limits to the growth in the functional requirements demanded by the users, except for the organization's ability to deliver the supporting software; as a result, mainline MIS applications are growing rapidly in size, with programs in excess of a million lines of code not uncommon.
- Managers increasingly demand that the MIS be designed in such a way that it can adapt to organization learning and environment changes.
- MIS designers increasingly put the primary design emphasis on a shared database rather than on individual applications.
- Hardware is declining rapidly as a significant design issue; software issues should almost always dominate design decisions (unless constrained by existing hardware or acquisition regulations).
- Low programmer productivity, both in the initial development and throughout the continuing maintenance cycle of an application, constitutes a major impediment to the successful use of management information systems.

B. DOD INFORMATION TECHNOLOGY STRATEGY

1. DoD Technical Architecture Framework for Information Management

Current DoD information systems infrastructure consists largely of "stovepiped," single-purpose, and inflexible systems that are difficult and costly to maintain. In an attempt to provide guidance for development of better quality information systems, DoD provides a strategic vision for information technology, which emphasizes integration, interoperability, flexibility, and efficiency through the development of a common, multi-purpose, standards-based infrastructure. DoD's Technical Architecture Framework for Information Management (TAFIM) implements CIM initiative concepts by providing services, standards, design concepts, components, and configurations that can be used to guide the development of technical architectures that meet mission requirements. The rest

of this section focuses on specific issues from the TAFIM that provide background for the study of this thesis (TAFIM 1993).

a. Technology

Information technology within DoD will eventually extend from the foxhole to the office, in fixed and mobile locations. Platforms are expected to adhere to a common set of interface standards that make it possible to configure software across distributed environments and tailor the software to support specific function processes. Low-cost platforms, coupled with rapid and responsive software development, will enable effective implementation of continuous functional process improvements.

b. Product availability

Commercial software products, supplemented (when necessary) by Government-developed re-usable components, will provide DoD with tools to enhance productivity and decision making. Users also will be provided with the tools to tailor screens, menus, and applications so that they can be more productive, innovative and effective in the performance of assigned duties.

c. Open systems environment

DoD is fully committed to implementing an open systems environment. DoD is establishing a standards-based framework for defining a technical architecture to provide interoperability, portability, and scalability. The TAFIM uses Federal and National standards adopted by industry and international standards accepted worldwide by U.S. allies. Also, the guidelines will provide transition strategies on how to evolve baselines and legacy systems to the target open environment.

d. Streamlined life cycle

A streamlined life cycle will be used to compress the time needed to deliver the capabilities to the field and to reduce total life cycle costs. The process will emphasize the use of integrated computer-assisted methodologies and tools such as shared utility services, software re-use, and use of commercial products. Ad hoc system development efforts will not be permitted. System developments will be organized and engineered to be

repeatable and reliable to achieve rapid production of quality, efficient, and effective software.

e. Modeling and Prototyping

Data modeling will be fully integrated with computer-assisted development and maintenance environments to rapidly capture process models, data models, and other requirements and transform them into applications and databases that adhere to DoD standards for data elements and software. Rapid prototyping will be a built-in aspect of the systems development cycle so that incremental changes that support improved business processes can be accomplished in days and weeks versus months and years.

f. Streamlined acquisition

A streamlined acquisition process will be functioning in a way to ensure that the DoD information system infrastructure can be implemented on schedule and within budget.

g. Education and Training

Education and training of the DoD information management community in new methods, tools, and practices will be centrally managed. The goal will be to create technically literate users with a renewed emphasis on enhancing individual skills, productivity, professional growth, and job satisfaction.

C. SOFTWARE TECHNOLOGY

One key aspect of the software development environment has been the selection of available software technologies to develop and maintain information systems. With the many number of development languages and tools available in the market, selection for application development products can be a difficult task. To understand the goals that the CIM initiative and TAFIM have specified for application development, a discussion follows on the software technologies that are being used to develop information systems.

1. The 3GL Environment

Programming languages of the third generation are often described as procedural languages because programmers give detailed, step-by-step, instructions on *how* a task is to be accomplished. The most widely used third-generation languages (3GLs) include FORTRAN, primarily for scientific languages, and COBOL, primarily for business languages; C/C++ and Ada are more recently-developed 3GLs that are considered to be superior to the older 3GLs in a number of respects. All 3GLs are intended almost exclusively for use by professional programmers.

Some of the characteristics of a 3GL development process are as follows:

- Formal requirements specifications are needed to define user requirements.
- A development cycle model like the waterfall model is used to define and control the development process.
- Programs are formally documented.
- Application development time can take months or years to complete because of inefficient line-by-line coding.
- Maintenance can be slow and very expensive.

McPharland (1993) discusses some of the problems in developing applications using a purely 3GL approach:

- Developers are encouraged to write every application from scratch, believing that, because the low-level detail of their applications is unique, no portion of an applications is reusable.
- Large 3GL applications are difficult to maintain because the maintainers are often presented with a large body of 3GL code without any design information to help them identify the small portion of the code that must be changed.
- Development teams are often split into programmers and analysts to encourage consideration of end-user requirements. Ensuring that the two groups work as a coherent team is a major source of problems in many projects.

- The concentration on low-level programming concerns, the lack of re-use, and the need for programmers and analysts, leads to long development times.

2. The 4GL Environment

A concern among computer specialists and managers is the amount of time and development effort devoted to creating computer programs. Fourth-generation languages (4GLs) were created to speed up the application building process; make applications easy and quick to change, thus reducing maintenance costs; minimize debugging problems; generate bug-free code from high-level specification statements; and make languages user-friendly so that end users could solve their own problems and put computers to work (Martin 1985).

The distinguishing feature of 4GLs is in their *nonprocedural* nature that allows a programmer to specify *what* is to be done rather than *how* it is to be done. The processing task can be specified with significantly fewer lines of code compared to use of a 3GL (often a tenth of the instructions). (Senn 1990)

Fourth-generation languages help developers achieve this productivity by providing easy-to-use code generators for developing screens and reports, and by providing a comprehensive library of verbs, or subroutines, for commonly used program functions. Developers spend more of their time with the user, ensuring that the screens and reports match the user's needs. In some cases, the developers spend so little time writing detailed code that the need for separate analysts and programmers disappears (McPharland 1993).

Some installations have reported bad experiences using 4GLs or few benefits in productivity were achieved. The reasons include the following: (Martin 1985)

- Much learning is needed to handle some 4GLs with skill. Problems with development result when the organization has not invested the necessary money and time in building a team with sufficient skill and practice.
- Some application generators are limited in what they can generate. When applied to an inappropriate system, they can cause problems or fail to achieve the required results.

- To achieve a major reduction in development time, a major change in the management techniques and controls is needed.
- A design methodology appropriate for the 4GL was not employed. For nontrivial systems, computer-aided design is essential to achieve full productivity and maintenance benefits.
- The interactive prototyping features of the tool are not used; traditional up-front specifications (which are usually inadequate) are adhered to, regardless of the tool's ability to accommodate change.
- Some 4GLs are oversold and do not have the capabilities needed for complex systems.

3. The CASE Environment

Computer-Aided Software Engineering (CASE) is a tool-based technology that assists the software developer through all the stages of software development. CASE tools were developed for organizations to increase the productivity of their development staffs, improve the cost effectiveness of the development process, and ensure the quality and reliability of the system produced. CASE environments allow computer specialists to develop and validate designs and specifications -- in effect, eliminating the manual methods that were used to perform the same functions.

A CASE environment provides the analyst or systems developer with facilities for drawing a system's architectural diagrams, describing and defining functional and data objects, identifying relationships between system components, and providing annotations to aid project management (Chikofsky, Rubenstein 1988).

CASE technology does not stop at the traditional boundaries of analysis, design, and construction of software. Integrated CASE (I-CASE) products provided an integrated package of CASE tools that incorporates such functions as analysis and design, code generation, and project management tools.

Despite the flexibility and power of I-CASE tools, they still have some shortcomings, particularly in DoD use: (Emery, Zweig 1993)

- They are proprietary, requiring relatively long-term commitment to a single vendor, and with little likelihood of being adopted as an open industry standard.
- Many of them (particularly the older ones) lack the functionality to define a complete system within the product's specification language.
- Many are not well integrated with other software products, such as database management systems and communications monitors.
- The more powerful products are very expensive in terms of hardware requirements and/or software license fees.
- Some of the products are relatively inefficient in the use of machine resources, making them inadequate for high-volume production systems.
- In order to get the most effective use of the tools, organizations must undergo a significant change in the entire software development process. Such change generally involves a steep learning curve, and encounters a number of serious organization and behavioral barriers.

4. Additional Software Technologies

Besides 3GLs, 4GLs, and CASE tools, the software development environment can be enhanced by the use of other software technologies such as software re-use, visual programming aids, commercial off-the-shelf (COTS) software, micro-based development tools, debugging tools, utility programs, "middleware" products for interfacing new applications with existing legacy systems, and various products for managing software testing, program release, and software distribution. (Emery, Zweig 1993)

III. THE SOFTWARE DEVELOPMENT ENVIRONMENT

A highly productive development environment goes a long way toward correcting the deficiencies in many current information systems. The advantage is not simply a matter of reducing the time and cost of developing and maintaining a system; it can also significantly enhance the effectiveness of the system. (Emery, McCaffrey 1991)

Success in implementing information technology (IT) in an organization requires a realistic strategic plan for exploiting the IT support environment for software development. Developing quality AIS is increasingly important in DoD as continuing reductions in DoD budgets force re-allocation of resources. DoD's focus on software development methodologies has been the traditional "waterfall" model. This chapter discusses that method, plus the prototype methodology and the characteristics of a new paradigm that is emerging for software development.

Another aspect of the software development environment is the personnel who develop the information systems. A portion of the chapter discusses the software organization and the professional skills that private sector organizations look for in personnel to develop their information systems. Finally, DoD has hundreds of legacy systems in their inventory. In order to deal with these legacy systems and abide by the CIM initiative of eliminating duplication, a discussion is presented on how to deal with legacy systems.

A. SOFTWARE DEVELOPMENT METHODOLOGIES

Achieving more successful information systems calls for some major improvements in the software implementation process. This objective can be attacked along two lines: 1) refining the existing process, or 2) making fundamental changes in the process. (Emery, McCaffrey 1991)

Many IS observers feel that a fundamental change is necessary to keep up with the growing demands of software development. Despite considerable time and effort to improve the process of the conventional *systems development life cycle* (SDLC), it is still

prone to significant failures, schedule delays, and budget overruns. The rest of this section discusses some of the shortcomings of the SDLC for application development. Prototyping and an emerging software development paradigm will also be discussed as alternative approaches to application development.

1. Conventional Systems Development Life Cycle

Accepted methodologies for developing information systems were developed more than 30 years ago and are still being used today. The most widely used methodology has been the SDLC. Although there have been some improvements to the process--analysis and design techniques, CASE tools to automate the different stages of development, and software products that assist the software programmer in the development process--the fundamental development process remains the same.

The SDLC development process (also called the "waterfall" model) is divided into several relatively independent phases: requirement specifications, systems analysis and design, coding, testing, and implementation. Each phase is completed prior to continuing to the next stage of development. The rationale for this model is that successful software development is accomplished by meeting subgoals or stages prior to continuing to the next phase of development.

The SDLC process introduces problems in the development cycle at the first stage: requirement specifications. In order to follow the sequential process of the SDLC model and proceed to systems analysis and design, careful planning of user requirements must be incorporated in the requirement specifications stage. The later stages of the development process then implement the "frozen" elements that define the user requirements. User requirements are difficult to define, and often times the user is not aware of the type of system that needs to be developed.

This approach to software development is generally characterized by:

- Hand coding in a third-generation language (e.g., COBOL, Ada)
- A structured programming development methodology
- Programming by professional programmers

- Modest user participation, mainly in requirements definition phase.
- Low productivity generally using a 3GL, requiring large development teams
- Large development projects have a long delivery cycle
- Maintenance consumes most of the available technical resources

This methodology has worked well for user requirements that are well understood and where user requirements are not likely to change during the life cycle of the system being developed. However, the reality is that program sponsors continue to submit additional requirements throughout the system design process. This results in time schedule delays and adds costs exponentially.

Adding to the problem of defining user requirements, developers using the SDLC model began to realize that the longer an error persists throughout the development cycle, the more costly it is to correct.

Boehm (1983) noted that if correcting an error in the requirements stage costs $1, the same error will cost $5 to correct in the design phase, $10 to correct in the programming stage, and $100 in the implementation phase of the development life cycle. Developers thus need to catch the errors early in the development stage in order to avoid a drastic rise in costs of developing the system.

Productivity alone would be a powerful reason for moving from conventional programming to more automated forms of application creation. There is, however, another reason that is often more powerful: in many situations the conventional development process *does not work*. (Martin 1982) The literature is full of examples of information systems that have been delivered after years of development effort, only to find that they are not acceptable to the users.

Martin also goes on to say that a common reaction to this unfortunate situation is to blame the problem on failures to specify requirements thoroughly. As a result, more elaborate procedures have been devised for requirement specifications, sometimes resulting in voluminous documentation. But still the systems have been unsatisfactory.

The mere act of implementing a user-driven system changes the requirements for that system. The solution to a problem changes the problem.

2. Prototyping

An alternative to the SDLC is application software prototyping. Prototyping is an iterative development process that builds quick software models as the means to solicit and validate user requirements. It reduces cost and adds value to the application development cycle (Boar 1993).

A software prototype has been described as a live, working system. It can be evaluated by the developer and the end user as it is being used in an operational environment. The prototype's purpose is to test out assumptions about users' requirements, or about the design of the application, or perhaps even about the logic of a program. (Sprague, McNurlin 1993)

Prototypes are built quickly, tested, and returned for further development in an iterative process. The initial prototype may be a simple program that performs basic functions. The system designer and the end user then discover new requirements as they use the system. As the user operates the prototype version of the system, valuable feedback can be given to the developer. Each additional version of the prototype then incorporates the additional functionality that is learned from operating the prototype. An advantage to using this development process is that user requirements do not have to be well understood ahead of time.

Unlike the SDLC model for development, the prototype method of development can be used when the user requirements for a systems are not well understood because concrete specifications do not have to be identified before building the prototype. Prototyping eases the communications between the developer and the user by allowing the user to interact with the prototype, experience its functionality, learn from it, and convey that information to the developer for further iterations of the prototype. Resistance to implementation of the system by the user is greatly reduced, significantly reducing

implementation and training costs. With user involvement throughout the development process, there is a greater likelihood that the system will be accepted and used.

DoD has used a special version of the prototype methodology as a means to better identify requirement specifications in the SDLC. As discussed in the SDLC method, requirements must be well-planned and understood prior to continuing to the next stage of development. The prototype has been used as a way to solidify the requirements specification stage of the waterfall model. This method has been described as prototyping, but it really is just another adaptation of the traditional conventional model that incorporates all the limitations of the SDLC.

Important to the prototype methodology is the type of application language used to develop the prototype. The time and cost of an initial version of a prototype application can be substantially reduced using higher level-languages like non-procedural 4GLs (Carey, Mason 1983). In fact, without the use of a high-productive language, prototyping is not feasible.

There is a difference of opinion by some software developers on which methodology is better for software development--the SDLC or prototyping. Sprague and McNurlin write that some (software evolutionists) contended in the early 1980's that 4GLs and prototyping really only affected a small portion of the development effort, mainly coding, and thus provided only marginal benefits. The largest productivity problems came from errors introduced during system analysis and design, and by the paperwork associated with large projects. Therefore, the evolutionists argued, the new techniques should be merged into the traditional life cycle methodology and meaningful productivity increases would come from improving the proven conventional techniques, such as structured programming. These proponents of the "conventional approach" believe that fine-tuning the development process through automation and re-use of code and other development products is the real key to big gains in programmer productivity.

Another point of view says that 4GLs increase programmer productivity but only when applied to programming in a new way, not just used as another language. These

tools allow people to work differently, not just faster, and that was the key to using them successfully.

Fourth-generation language proponents argue that programmers who develop systems in traditional procedural languages have a difficult time adapting to the rapid, iterative programming allowed by a 4GL. In the 4GL environment, the application framework is the primary concern in development and details like requirement specifications are ignored until later in the development process. The argument is that the conventional programming mind-set that requires fully specifying a system beforehand, programming the static set of requirements, and concern about code detail and exactness, is actually a disadvantage, not an advantage, to using 4GLs for application development.

3. Emerging Software Development Paradigm

Emery and Zweig suggest that the difficulties in software development stem from intrinsic flaws in the conventional implementation process. They suggest a change is needed in the development process, rather than refining the conventional development process. They provide common "themes" of a new paradigm that is emerging for software development:

- The process tends to be relatively continuous throughout the development life cycle and corrections and modifications are continuously fed back to earlier stages based on learning.
- Line-by-line coding in a 3GL is avoided as much as possible. Developers use such products as commercial-off-the-shelf software (COTS) to provide components for a complete application package, and 4GLs and integrated CASE products that provide the analyst with high-level specification languages that require many fewer statements than a 3GL.
- The use of large development teams is avoided as much as possible. Small development teams are possible because of the use of higher-level languages and the avoidance of 3GL programming.

- A small team working on a project over a relatively short development cycle permits greater team continuity.
- Rather than discouraging changes in the requirements like the SDLC, this paradigm depends on continual interaction with the users to solicit their knowledge into an evolving application.
- End-user tools, such as report generators and query languages, permit the user organization to make simple, quick changes to information outputs.

4. Object-Oriented Technology

Object-oriented technology is increasingly credited with huge development productivity gains, development-cost savings, on-time or early completion of projects, increasing quality of initial products, and easier maintenance of the finished systems (Amaru 1993).

Object-oriented programming appears to be one of the emerging methodologies of the 1990s for faster application development time, re-usability of code, and increased programmer productivity. However, Amaru warns that improved productivity comes at a price. Object-oriented programming requires a radically different way of thinking. Programmers and analysts who are accustomed to doing their jobs in the traditional procedural way--using structured 3GLs or 4GLs and traditional system analysis techniques--will need substantive retraining in object-oriented programming and analysis techniques.

In conjunction with object-oriented technology, several methodologies have been developed to assist developers in system analysis and design: (Amaru 1993)

- Shlaer-Mellor method allows software engineers to examine abstract data types to find objects. The objects are then used to create models representing process, state, and information.
- Yourdan system analysis methodology developed by Edward Yourdan concentrates on object-oriented systems analysis.

- Booch method attempts to create a logical and a physical model of the system being developed.

Object-oriented development is most important in three type of applications: (Sprague, McNurlin 1993)

- Graphical applications. The inheritance of the objects provides consistent behavior among the objects allowing easy-to-remember user interfaces.
- Multimedia applications. "No other database management technique can handle a variety of data, such as voice, data, images, text." (Sprague, McNurlin 1993)
- Complex systems. Objects manage complexity better by reducing the dependencies among functions.

Fichman and Kemerer (1993) note that object-orientation is a new development model and requires new skills in analysis, design, and programming that replace, rather than build on, those associated with conventional development. It has also been observed that to maximize re-use, developers must learn to assemble applications using objects developed by others.

A software development organization must select among an almost limitless number of possibilities and combinations of management practices, development techniques, and automated support in order to create and evolve whatever software methodology it uses. The choice of methodology is influenced by a wide variety of such considerations as the size and skills of the software development organization, applications being developed, the number of places in which applications will be used, the criticality to the applications, and the projected needs for maintenance and modification. (Wasserman, 1981)

B. DEALING WITH LEGACY SYSTEMS

The costs of operating and maintaining DoD's application software for Automated Information Systems (AIS) has been estimated to be $10 billion annually. Despite this level of spending, DoD has been unable to obtain correct information from the data stored in various databases due to the lack of standardized data and data structures across systems. Many of these systems use proprietary software and hardware making it difficult

or impossible to interoperate with other DoD information systems. The DoD legacy systems inventory includes obsolete electronics, technology, and systems designs up to 30 years old and they have been poorly documented. (Aiken, Muntz, Richards 1994)

For many years, DoD has developed, operated, and maintained unique AIS that often accomplish the same task. Valuable resources are often wasted maintaining these obsolete or outdated systems. As described in Chapter I, the redundancy of these systems has forced DoD to attempt to consolidate systems with similar functional capability. DoD's approach to eliminating the redundant information systems is to select the "best of breed" AIS systems and use them as "migration" systems, with the goal to integrate them into target systems by selective reengineering.

Most information systems executives feel trapped by the past. They have hundreds or even thousands of old legacy programs and data files that they would love to replace. But with a backlog of perhaps two or more year's worth of new work already in the queue, they see no way or replacing these legacy systems (Sprague, McNurlin 1993).

Today, there are choices available for dealing creatively with legacy systems. Instead of replacing a legacy system with an application package or totally re-writing it with a procedural language such as COBOL, there are several alternatives, made possible by tools, new development techniques, and new programming languages.

Sprague and McNurlin (1993) present several tactics for dealing with legacy systems:

- *Re-write legacy systems*. In some cases, a legacy system may be too far gone to rescue. If the code is convoluted and patched, if the technology is antiquated, and if the design is poor, it may be necessary to start from scratch . An option many companies are choosing is to "downsize" legacy system applications by re-writing them for a smaller platform, such as a midrange machine or network server.
- *Replace legacy systems with purchased packages*. Many commercial packages today have the functionality and versatility to replace older legacy systems that perform the same functions. Another reason for purchasing a commercial package

is to provide connectivity between different operating systems or distribute an application's workload among heterogeneous systems. Some application packages, often called "middleware," support an open-system strategy in that the database networking infrastructure is independent of front-end and back-end vendors.

- *Refurbish legacy systems.* If the legacy system is operating well and is still maintainable, some extensions can be added. These extensions can support new inputs, outputs, and make new uses of the data.
- *Restructure legacy systems.* If a legacy system is running efficiently, the system can be restructured using automated tools to turn "spaghetti code" into more structured code.
- *Re-engineer legacy systems.* A step beyond restructuring is re-engineering, which is extracting the data elements from an existing file and the business logic from an existing program, and moving them to new platforms. (Sprague, McNurlin 1993)

C. THE SOFTWARE ORGANIZATION

Companies that are attempting to manage the full range of system development are creating a spectrum of systems groups. PRISM, which stands for Partnership for Research in Information Systems, is a joint research service that investigates subjects suggested by its corporate sponsors. In its study of information systems human resources, PRISM identifies three types of information system professionals needed for the 1990s (Sprague, McNurlin 1993).

1. The High-level Business Analysts

The high-level business analyst is the professional in an organization who identifies ways that the organization can benefit from using information technology. This analyst is someone who has both company and general business knowledge and generally is a functional specialist skilled in a specific business area such as inventory control.

2. The Multidisciplinary Team Leaders

Multidisciplinary team leaders lead small, highly-skilled teams of system developers with diverse backgrounds. The work of these teams differ from that of the current practice in systems development where the work is done in relatively independent stages or pieces. Multidisciplinary teams perform the entire application development and are involved throughout a project's life cycle.

3. The Super Technical People

Super technical people know one technology very well and have some knowledge of several contemporary technologies.

4. Information Technology Specialists

Martin (1982), defines various specializations for today's information technology environment:

- *Information center representatives* work with end-users to assist them in obtaining the information they require and in generating user-driven applications or prototypes.
- *Data-base administrators* collect the logical view of the data that different applications need and synthesize them to create logical database structures and maintain a data dictionary and logical model of the data.
- The *Data-base designer* is the technical expert of the database systems that are being used by the organization and is concerned with database hardware, software, performance, and how the data is distributed.
- A *Software specialist* is a specialist in one software package for application generation.
- *Network specialists* ensure that network services provide the services required by the organization.

The Bureau of Labor Statistics (BLS) predicted that employment in the computer industry will grow at an average annual rate of 3.7% through 1995 (BLS 1986). The need for computer industry personnel, along with the rapid rate of change in computer

technology, will add pressures on today's organizations to find the quality personnel with the critical skills necessary to develop information system applications.

Cheny, Hale, Kasper (1989) surveyed senior IS professionals to identify the importance of various skills for programmers, systems analysts/designers, and project managers for the 1990s. The predictions are based on expert opinion of 79 information systems managers and 58 organizations of various sizes. Results of the survey for the 1995 workforce, compared to a prior survey conducted in 1987, are as follows:

	1987*	1995**
COBOL Programmers	11, 151	6,480
FORTRAN or Basic	300	300
PASCAL and C	2,050	4,180
Database Management	1,480	2,180
4th Generation Languages	1,150	7,401
Systems Programmers	401	600
Data Communications	401	1,400
Systems Analysts	2,840	4,150
Operators	850	400
Data Entry	330	0
Information Center Personnel	2,400	4,100

* actual employment figures
** projected employment figures

Reproduced from (Cheny, Hale, Kasper, 1989)

The data shows that respondents expect net increases in personnel requirements for all categories except COBOL programmers, FORTRAN or Basic programmers, data entry personnel, and computer operators. The data also indicates that the number of personnel limited to some traditional third-generation languages such as COBOL or FORTRAN skills will decrease and the demand for workers with knowledge of non-procedural languages (4GLs) is expected to increase dramatically.

The need for 4GL specialists appears to be tied to the movement toward using microcomputers for an increasing number of small applications. Some 4GLs, such as PowerBuilder and Visual Basic, are used as development tools and are replacing procedural languages such as COBOL.

The data also indicates there will continue to be a substantial demand for database and system programmers. The cost of converting existing applications has prevented many companies from fully utilizing this technology.

Many of the firms surveyed are using a phase in/phase out approach--that is, when new systems are designed to replace existing ones, database management system (DBMS) technology is utilized, but modification of existing applications to employ DBMS technology is not actively pursued. As more software applications utilize DBMS technology, the demand for database specialist will increase. (Cheny, Hale, Kasper 1989)

The IS professionals also report a continued demand for data communication specialists, probably a result of continued integration of systems and a tendency of moving their technology toward distributed systems.

Collectively, the results of this survey show a shift from a centralized, mainframe environment to a distributed, micro-based systems and from procedural to non-procedural languages.

D. DATABASE INTEROPERABILITY

In order to allow different computers, using different operating systems, to work together on cooperative tasks, interoperability will have to be achieved. Interoperability will allow the exchanging of information in standard ways without any changes in the command language or in functionality of the information system.

Open systems and standards refer to using products based on standards that promote interoperability and portability between heterogeneous vendor environments. Interoperability and transportability can be achieved by using various strategies: using a single vendor, standards, 4GLs, middleware, and CASE tools.

IV. PRESENTATION OF DATA

Contemporary technology presents policy makers with a bewildering array of alternatives, what policies and strategies for software development should be employed, given the huge variety of available alternatives? A review of best industry practice can provide valuable insights for dealing with this problem. This was the purpose of a questionnaire used to assess current industry practices.

A survey questionnaire was faxed to senior executives who play a vital role in strategic planning for their organizations in information systems. Each organization was telephoned to establish contact with the Chief Information Officer or equivalent executive officer to solicit his or her input to the survey. Respondents received a packet of the survey, along with a cover letter explaining the purpose and confidentiality of the survey. Of the 31 respondents selected for the survey, 23 responded--a gratifying high response rate.

The survey was designed so that the majority of the questions provided a list of responses to select. The survey participants were then asked to select the response that fit their views most closely. In compiling the results of the survey, it is recognized that the sample size is too small to perform traditional statistical analysis. However, the results do add valuable insight into emerging trends in software development. A copy of the questionnaire is in the Appendix.

A. THE ROLE OF LEGACY SYSTEMS

The respondents were asked three questions about the role of legacy systems in their organizations:

1. What role do you envision for legacy systems in your information strategy over the next 5 years? (Total number of responses are in parenthesis.)

   (2) A continued central role

   (10) An important but declining role

   (11) A sharply declining role as we aggressively replace legacy systems.

The response to this question depended largely on the function the legacy systems perform for the organization, and the investment the organization has made in its legacy systems. As an example, one respondent that selected a continued central role for legacy systems in his organization stated that his organization invested more than $250 million in legacy systems over the last eight years. With such a large investment, it will be difficult for his company to scrap the legacy systems. Besides, the legacy systems still play a vital role in the organization.

2. Rate the importance of continuing to invest in enhancement of their legacy systems. (Total number of responses are in parenthesis.)

   (11) Not important to enhance legacy systems

   (11) Somewhat important to enhance legacy systems

   (1) Very important to enhance legacy systems

The responses to this question follow the first question closely. The respondents that viewed legacy systems as playing an important but declining role rated their systems as somewhat important to enhance; those that see a sharply declining role for their legacy systems, do not want to invest in enhancements.

3. The respondents were asked to select the relative importance of a list of options to enhance their organization's legacy systems. The data depicted in Table 1, shows that re-writing a legacy system in a procedural language and restructuring a legacy system, is not an important option.

	Not Important	Somewhat Important	Very Important
Re-write in a procedural language	18	5	0
Re-write for client/server	6	7	10
Replace with application packages	2	4	17
Refurbish by adding extensions	10	8	5
Restructure	16	5	2
Integrate using "middleware"	5	12	6

Table 1. Legacy System Options

B. DEVELOPING AND MAINTAINING INFORMATION SYSTEMS

Respondents were asked to indicate the relative importance of software languages and tools in developing and maintaining applications over the next five years. The results appear in Table 2 through Table 10.

Table 2 shows the results for using a 3GL such as COBOL for developing and maintaining applications. It is interesting to note that a majority of the respondents viewed 3GLs as relatively important in developing and maintaining applications. Later tables will indicate their strong support for higher-level languages and tools. When asked about their responses, the respondents felt that the question really addressed two separate issues: software development and maintenance. Because the question was stated as, "In developing and maintaining information systems....," the respondents argued that a 3GL such as COBOL may not be important for developing applications, but still is very important for the maintenance of systems that have been written in 3GLs.

	Not Important	Somewhat Important	Very Important	Essential	Don't Know
3GL	10	8	3	3	

Table 2. Importance of 3GLs

Table 3 displays the respondents' views of programming in 3GLs with a heavy use of re-usable components for application development. The results of the data correspond fairly closely with the results in Table 2.

	Not Important	Somewhat Important	Very Important	Essential	Don't Know
Program 3GL with re-use	8	11	3	1	

Table 3. Program in a 3GL with heavy use of re-usable components

The data in Table 4 supports the conclusion that 4GLs are expected by the respondents to be important in application development in the near future. A strong majority of the respondents feel that 4GLs are very important or essential for application development.

	Not Important	Somewhat Important	Very Important	Essential	Don't Know
4GL	1	4	14	4	

Table 4. Importance of 4GL

Table 5 lists the results of the respondents view on I-CASE tools in developing applications. Conclusions can be drawn from the data that the majority of the respondents do not strongly support I-CASE as important to application development.

	Not Important	Somewhat Important	Very Important	Essential	Don't Know
I-CASE	3	10	7	2	1

Table 5. Importance of I-CASE products

Table 6 lists the respondents' views on the relative importance of using application packages in developing information systems. The data supports the conclusion that the respondents feel application packages will be used in the future to develop and maintain information systems. When interviewed, some of the respondents indicated that the commercial application packages that are available today are sufficient to take care of some of their business needs. Again, a strong majority of the respondents selected very important and essential as responses.

	Not Important	Somewhat Important	Very Important	Essential	Don't Know
Application packages	2	1	11	9	

Table 6. Importance of Application packages

The respondents view micro-based development tools as playing an important role in the development of applications. In Table 7, a strong majority of the respondents expect

these tools to be very important or essential over the next five years for development of applications.

	Not Important	Somewhat Important	Very Important	Essential	Don't Know
Micro-based development tools	1	2	12	7	1

Table 7. Importance of Micro-based development tools

Table 8 shows that a majority of the respondents view "middleware" as very important or essential in the development of software applications. It is interesting to note that four of the respondents did not understand the role of middleware or did not know if middleware will play a role in application development.

	Not Important	Somewhat Important	Very Important	Essential	Don't Know
Middleware	1	3	12	3	4

Table 8. Importance of Middleware

The responses in Table 9 are more dispersed than the responses for some of the prior software tools. The respondents were asked to select the relative importance of object-oriented technology in the development of applications. Conclusions can be drawn from the data that many of the respondents do not feel strongly about object-oriented technology for application development, although many view it as very important or essential.

	Not Important	Somewhat Important	Very Important	Essential	Don't Know
Object - oriented technology	1	7	8	6	1

Table 9. Importance of Object-oriented technology

The respondents were asked to select the relative importance of prototyping in application development. Table 10 indicates that all the respondents feel prototyping is important in developing and maintaining system applications over the next 5 years.

	Not Important	Somewhat Important	Very Important	Essential	Don't Know
Prototyping	0	3	8	12	

Table 10. Importance of prototyping

C. PROFESSIONAL SKILLS

Table 11 displays the respondents' views of the professional skills they will be looking for to train or hire in their software development personnel over the next five years. The respondents were given a list of professional skills and asked to rate their organization's relative need for those types of personnel. The table lists the skills and the responses to their organization's relative need.

	Little or no need	A significant need	A critical need
3GL specialists	18	4	1
4GL or I-CASE specialists	6	14	4
Database specialists	1	13	9
Data communications specialists	1	7	15
Systems analysts/designers	2	14	7
Specialists in PC-based products	6	12	4
Functional specialists	4	10	8

Table 11. Professional skills to train or hire

A majority of the respondents feel their organizations have enough 3GL specialists to fulfill requirements for software development. A strong majority of respondents view data communications specialists as a critical need for their organization. The results also correlate well with Table 4, where a strong majority viewed 4GLs as very important or essential in application development.

D. SOFTWARE DEVELOPMENT METHODOLOGY

The respondents were asked to select their relative agreement with four statements about revising the software development process. The available response for each statement was: strongly agree, somewhat agree, somewhat disagree, strongly disagree, and no view. (Total number of responses are in parenthesis.)

1. The conventional approach is basically the right one; any flaws can best be remedied by imposing greater software engineering maturity over the process. (Total number of responses are in parenthesis.)

(3) Somewhat agree

(8) Somewhat disagree

(12) Strongly disagree

2. An adaptive approach may be useful for less critical applications but is too risky for "mission-critical" applications. (Total number of responses are in parenthesis.)

(5) Somewhat agree
(5) Somewhat disagree
(13) Strongly disagree

When asked about the response to question 2, the respondents agree that an adaptive approach is useful for less critical applications but disagree that an adaptive approach is too risky for "mission critical" applications.

3. An adaptive approach appears to offer a good long-term approach to software development, but the development tools and human skills do not yet exist to make this a practical approach over the next 5 years. (Total number of responses are in parenthesis.)

(4) Strongly agree
(5) Somewhat agree
(6) Somewhat disagree
(8) Strongly disagree

The responses are more dispersed than the prior responses. Several of the respondents when asked about their response, agree that an adaptive approach is a good approach to software development, but disagree that the development tools and human skills will not exist to make it a practical approach in the next five years.

4. An adaptive approach provides the most practical way of responding to the rapidly changing business environment that we face; therefore, we are trying to move to such a methodology as rapidly as practicable. (Total number of responses are in parenthesis.)

(13) Strongly agree

(9) Somewhat agree

(1) Strongly disagree

E. STRATEGIC PLANNING

A series of three questions was asked of the respondents to determine their views on strategic planning. The questions centered on goals for improving the software development process and the relative importance for accomplishing long-term organizational goals.

1. When asked what their strategy is for "open architecture" over the next five years, 96% responded that it will be given considerable weight in their information strategy. They view open architecture and standards as vital to competing in the business environment.

2. When asked how important it is for their organization to make a substantial improvement in the software development process, 17 responded that it is one of their high priority goals and 6 responded that it is important, but not at the top of their list in priorities.

3. The respondents were given a list of organizational goals and asked to rank them from 1 to 5, with 1 being most important and 5 being least important. The responses are listed in order by weighted average:

- Developing a more effective infrastructure **2.2**
- Identifying and developing new applications of strategic importance to the organization **2.3**
- Rapid response to user requests **2.4**
- Lower cost of application development **4.1**
- Managing the maintenance of legacy systems and migrating them to new systems **4.2**

V. CONCLUSIONS AND RECOMMENDATIONS

It is hoped that the survey data and conclusions of this thesis will provide DoD with additional knowledge of private sector practices for developing application software, the use of software languages, and the type of skills that the private sector is looking for in their software development personnel. Even though the data is not statistically significant, it is felt that the views of the executives surveyed, are relevant in determining the future direction of application development.

A. RESPONDENT PROFILE

The data from each questionnaire was analyzed individually and also in aggregate form to develop a profile of the IS executives that responded to the survey. Hopefully, this will give the reader an overall view of the information technology tactics used by the majority of the respondents.

The survey data was too limited in scope to separate the executives into specific categories, but an attempt was made to "label" the executives as IS managers of the "status quo" or managers on the "leading edge" of information technology innovation. In order to accomplish that, three categories in the survey were looked at closely: view of legacy systems, software languages and tools used for application development, and view of the conventional approach versus an adaptive approach to software development methodologies.

Even though the views of the author are open to debate, they do provide a starting point for identifying a profile of the survey respondents. The categories were chosen for the following reasons:

1. Organizations that have operated their legacy systems feel comfortable with the services they provide. Abandoning these systems for new technology can be a major task in change management skills, even though abandoning the systems for better technology may mean better productivity for the organization. Also, management may not be willing to take the risk to move to another information system or direct the necessary funds

toward their development. CIOs of these organizations feel comfortable with the "status quo" and generally feel comfortable with the way things are.

2. CIOs that are forward-looking will seek software languages and tools that incorporate the latest technological advances. These CIOs will view 4GLs, I-CASE products, and object-oriented technologies as very important for development of applications in their organizations.

3. In today's era of budget constraints, application development backlogs, and user demands for new software applications, some CIOs will view the conventional approach to software development as not important and will demand a more adaptive approach as vital to reliable and effective application development.

Twenty of the executives surveyed are viewed as being on the leading edge of information technology innovation. They view legacy systems as not important in their organizations and do not see the need to enhance them. At the same time, they view 4GLs, I-CASE products, and object-oriented technology to be very important in application development. They also feel an adaptive form of software development is key to the successful implementation of information systems.

Interestingly, one respondent's view of the three categories selected to profile the IS executives was quite different from the majority of survey views. His view is that it is very important to enhance legacy systems and integrate them with middleware products, or restructure the code by reducing the code into subcomponents and modules. His view on languages and tools is that 4GLs, I-CASE products, and object-oriented technology are not important. A telephone interview with the respondent revealed that he is very skeptical of the newer products on the market and feels that vendors over-sell their products. In his view these products have not proved to be effective for application development and are too immature. His philosophy is that the "old" ways of developing and maintaining information systems still work better. This IS executive appears to be happy with the status quo.

It has not been the intent here to determine which approach to managing information technology resources is the correct one. Organizations manage technology in different ways depending on their business functions and their established goals.

B. LEGACY SYSTEMS

DoD, as well as the private sector, have many legacy systems that continue to be maintained. Some experts in the IS field have estimated that maintaining an information system can absorb more than 60% of the system's life cycle costs. Clearly, if a legacy system does not provide the necessary functionality for an organization to accomplish its business goals, then questions can be raised as to why organizational IS budgets continue to pour valuable resources into maintaining these systems.

Because of the CIM initiative, DoD has taken steps to reduce the inventory of its legacy systems and consolidate them by taking the "best of breed" systems and migrating them toward more centrally operated systems. Their short-term goal is to re-engineer the systems by identifying common processes and restructuring them into common systems operated by the individual services.

In the private sector, legacy systems are being aggressively replaced rather than rebuilding them or restructuring them. Viable options cited for replacing legacy systems are replace with application packages, integrate with middleware, and re-write for client/server platforms. The majority of respondents view legacy systems as too costly to maintain and are looking for new and innovate ways to replace them.

C. INFORMATION SYSTEM LANGUAGES

In developing applications for the next five years, some of the languages and tools the private sector is expecting to use are fourth-generation languages, application packages, micro-based development tools, and middleware. Some of the CIOs interviewed for the survey feel that application packages are now being developed that can easily replace older systems in the organization, especially in the business applications.

They see no need in developing new applications, when proven and tested products are available from vendors that contain the functionality they desire for their business.

DoD encourages the use of commercial off-the-shelf (COTS) software for developing information systems. When necessary, the COTS must be supplemented by government-developed re-usable components.

DoD also mandates the use of Ada for the development of all applications unless it can be shown that it is cost effective to use some other software language. It is interesting to note that none of the executives responding to the survey has software development personnel trained in the use of Ada. The majority of the organizations almost exclusively use COBOL as the 3GL of choice, with FORTRAN and BASIC as secondary choices.

Moreover, the private sector does not view developing information systems with a procedural language or a procedural language with re-usable components as feasible options.

D. SOFTWARE DEVELOPMENT PERSONNEL

The private sector is looking for a diverse group of specialized software development personnel to support their organizations. For 3GL programmers, these organizations use personnel that have been trained mainly with COBOL, but feel that 3GLs are not important in applications development. Most will be employing personnel with skills in 4GL programming, CASE tool development experience, and specialties in object-oriented technologies. It appears that many of the organizations will be employing fewer personnel with procedural language background.

An interesting question arises as how these organizations are going to pay the cost for training and developing the skills of these personnel. Also, some organizations will have less development personnel because they intend to purchase more application software rather than develop applications themselves.

E. RECOMMENDATIONS FOR FURTHER RESEARCH

As the research for this thesis continued to develop and the survey was administered to the IS executives, it was realized that the scope of the thesis could be expanded. However, time constraints did not allow for further research and modification of the questionnaire. Therefore, this paper is viewed as a general starting point for follow-on research. Areas of additional research are:

1. Determine how private sector organizations plan to transition to new development environments and high-level languages and tools. It would be interesting to find out the plan for training personnel in these new skills and to determine what the costs will be.

2. Assemble several focus groups of IS management personnel to determine their strategies for application development. The survey is a valuable tool for obtaining some information but it is limited in the information it provides. A better understanding can be obtained through interviews and focus groups.

3. Observe several IS departments in organizations to determine how they deal with information technology issues. Develop a case study where recommendations can be made for the best high-level languages and tools to be used in application development.

4. Expand the research of this paper by developing a more comprehensive survey and correlating the data to provide more specific information on why CIOs select specific responses.

APPENDIX. SURVEY QUESTIONS

Survey Questionnaire

1. Some information systems that were developed 10 or 20 years ago are still being used by organizations. The term often used to describe these systems is *legacy systems*. What role do you envision for legacy systems in your information strategy over the next 5 years?

_____ A continued central role
_____ An important but declining role
_____ A sharply declining role as we aggressively replace legacy systems.

2. Over the next 5 years, how would you rate the importance of continuing to invest in the enhancement of your organization's legacy systems (versus the development of new systems and applications)?

_____ Not important to enhance legacy systems
_____ Somewhat important to enhance legacy systems
_____ Very important to enhance legacy systems

3. In your opinion, what do you see as viable options for dealing with legacy systems? For each option listed below, please rate its importance using the following scale:

1 = not important 2 = somewhat important 3 = very important

_____ Re-write in a traditional procedural language (e.g., COBOL)
_____ Re-write for a client/server architecture
_____ Replace them with off-the-shelf application packages
_____ Refurbish by adding extensions (e.g., add a new graphical user interface)
_____ Restructure (e.g., use tools to turn "spaghetti code" into more structured code)
_____ Integrate them using "middleware" products
_____ Other approaches (please describe briefly) _____
_____
_____

4. In developing and maintaining information system applications over the next 5 years, please indicate the importance of the following software languages, tools, and programming techniques. (Please check one for each category.)

	Not Important	Somewhat Important	Very Important	Essential	Don't Know
(a) 3GL (e.g., COBOL)	____	____	____	____	____
(b) Program in a 3GL with heavy use of re-usable components	____	____	____	____	____
(c) 4GL (e.g., FOCUS, PowerBuilder)	____	____	____	____	____
(d) I-CASE products (e.g., TI's IEF)	____	____	____	____	____
(e) Application packages (e.g., payroll)	____	____	____	____	____
(f) Micro-based development tools (e.g., Visual Basic)	____	____	____	____	____
(g) Middleware (e.g., IBI's EDA/Link)	____	____	____	____	____
(h) Object-oriented technologies (e.g., C++, Ada 9X)	____	____	____	____	____
(i) Prototyping	____	____	____	____	____

5. Approximately how many information system software development personnel (full-time professional programmers, analysts, etc.) are employed by your organization (i.e., the part you manage and on which your answers are based)?

_____

6. What percentage of your software development personnel have skills in the use of the following procedural programming languages? (Note: The percentages need not add up to 100%; the sum can be less than or greater than 100%.)

COBOL _____%
C/C++ _____%
PASCAL _____%
Ada _____%
BASIC _____%
FORTRAN _____%
Other _____% (please list) _____
_____
_____

7. In developing software applications for information systems over the next 5 years, how many software development personnel will your organization have compared to software development personnel you have today?

_____ Significantly fewer than today
_____ About the same
_____ Significantly more than today

8. Over the next 5 years, which types of professional skills will you be looking for to train or hire in significant numbers? Please rate your need on the following scale:

1 = little or no need 2 = a significant need 3 = a critical need

_____ 3GL specialist
_____ 4GL or I-CASE specialist
_____ Database specialist
_____ Data communications specialist
_____ Systems analysts/designers
_____ Specialists in PC-based products (e.g., word processing, spreadsheets)
_____ Functional specialists (i.e., skilled in a specific business area such as inventory control)
_____ Other (please specify) _____
_____
_____

9. There has been considerable attention given to revising the way we develop software applications. One possible direction would be to move from the "conventional" approach to a more adaptive one. The following (admittedly somewhat simplistic) descriptions are provided for the purpose of explaining the questions given below.

Characteristics of the conventional methodology:
- The process is divided into relatively independent stages (e.g., requirement analysis, design, coding, testing, and conversion).
- Careful attention is given to the up-front requirement specifications, with a heavy effort made to minimize the number of subsequent revisions in the specifications.
- The primary emphasis in improving the development process is focused on achieving greater control in order to improve the predictability and reduce the variation of the process.

Characteristics of an adaptive methodology:
- Initial requirement specifications are viewed as a first approximation rather than an attempt to determine the "final" specifications.
- Prototyping is used to demonstrate to "customers" working examples of screens, reports, and application logic.
- Feedback is obtained from users during the entire development process, with suggested changes incorporated into a working prototype.

Please indicate the strength of your agreement with the following statements about such a shift in the development methodology.

a. The conventional approach is basically the right one; any flaws can best be remedied by imposing greater software engineering maturity over the process.

Strongly Agree	Somewhat Agree	Somewhat Disagree	Strongly Disagree	No View
_____	_____	_____	_____	_____

b. An adaptive approach may be useful for less critical applications (e.g., "end-users" programs), but is too risky for "mission-critical" applications.

Strongly Agree	Somewhat Agree	Somewhat Disagree	Strongly Disagree	No View
_____	_____	_____	_____	_____

c. An adaptive approach appears to offer a good long-term approach to software development, but the development tools and human skills do not yet exist to make this a practical approach over the next 5 years.

Strongly Agree	Somewhat Agree	Somewhat Disagree	Strongly Disagree	No View
_____	_____	_____	_____	_____

d. An adaptive approach provides the most practical way of responding to the rapidly changing business environment that we face; therefore, we are trying to move to such a methodology as rapidly as practicable.

Strongly Agree	Somewhat Agree	Somewhat Disagree	Strongly Disagree	No View
_____	_____	_____	_____	_____

10. Over the next 5 years, what is your strategy with respect to the use of an "open" architecture (i.e., one based on widely used standards)?

_____ The use of an open architecture will be given considerable weight
_____ "Openness" is not likely to be given much weight in making development decisions
_____ We have no strategy
_____ No opinion

11. In looking at your overall management goals, how important do you consider it for your organization to make a substantial improvement in the software development process?

_____ One of my high priority goals
_____ Important, but not near the top of my priority list
_____ Not very important

12. In looking at strategic planning for information systems development over the next 5 years, please rank the following choices in terms of importance to you:

(1 most important, 5 least important)

_____ Rapid response to user requests
_____ Lower cost of application development
_____ Developing a more effective infrastructure (i.e., network, shared databases)
_____ Managing the maintenance of legacy systems and migrating them to new systems
_____ Identifying and developing new applications of strategic importance to my organization
_____ Other (please specify) _____
_____
_____

13. Would you like to receive a summary of the research findings from this project?

_____yes _____no Name _____

LIST OF REFERENCES

Aiken, P., Muntz, A., Richards, R., *DoD Legacy Systems Reverse Engineering Requirements, ACM*, 1994.

Amaru, Christopher, *Where Object Technology Fits In*, Digital News & Review, v. 10, no. 9, October 1993.

Boar, Bernard H., *The Art of Strategic Planning for Information Technology*, John Wiley & Sons, Inc., 1993.

Boehm, Barry W., *Software Engineering Economics*, IEEE Transactions on Software Engineering, pp. 4-21, January 1984.

Bureau of Labor Statistics, *Occupational Projections and Training Data*, Bulletin 2251, Washington, D.C., U.S. Government Printing Office, 1986.

Burke, John P., *Nip & Tuck for Legacy Systems: Give your HP 3000 System a Face Lift, But Use the Right Instruments*, HP Professional, v. 7, no. 12, December 1993.

Carey, T.T., and Mason, R.E.A., I*nformation System Prototyping: Techniques, Tools, and Methodologies*, The Canadian Journal of Research and Information Processing, v. 21, no. 3, pp. 177-191, 1983.

Cheny, Paul H., Hale, David P., Kasper, George M., *Information Systems Professionals: Skills for the 1990's*, Proceedings 22nd Annual Hawaii Int'l Conf. on System Sciences, pp. 331-336, 1989.

Defense Information Systems Agency Center for Architecture, *Department of Defense Technical Architecture Framework for Information Management*, Ver. 2.0, November 1993.

Emery, James C., and Zweig, Dani, T*he Use of Ada for the Implementation of Automated Information Systems Within the Department of Defense*, Naval Postgraduate School, December 1993.

Emery, James C., McCaffrey, Martin J., *Ada and Management Information Systems: Policy Issues Concerning Programming Language Options for the Department of Defense*, Naval Postgraduate School, Monterey, Ca., June 1991.

Endoso, Joyce, *DoD Releases Long-awaited Vision Plan for CIM Initiative*, Government Computer, June 1994.

Fichman, Robert G., and Kemerer, Chris F., *Adoption of Software Engineering Process Innovations: The Case of Object Orientation*, Sloan management Review, Winter 1993.

GAO, House Report 101-382, *DoD Automated Information Systems Experience Runaway Costs and Years of Schedule Delays while Providing Little Capability*, 1992.

Jones, Capers, Applied Software Measurement, McGraw-Hill, 1991.

Martin, James, *Application Development Without Programmers*, Prentice-Hall Inc., 1982.

Martin, James, F*ourth-Generation Languages, Volume I, Principles*, Prentice-Hall, Inc., 1985.

McParland, Patrick, *From GLs to GTIs*, EXE, v. 8, no. 5, October 1993.
Senn, James A., *Information Systems in Management*, 4th Ed., Wadsworth Publishing Co., 1990.

Shikofsky, Elliot J., and Rubenstein, Burt L., *CASE: Reliability Engineering for Information Systems*, IEEE Software, pp. 11-16, March 1988.

Sprague, Jr., Ralph H., and McNurlin, Barbara C., *Information Systems Management in Practice*, 3rd Ed., Prentice-Hall, Inc., 1993.

Wasserman, Anthony I., The Ecology of Software Development Environments, IEEE Computer Society Press Technology Series, pp. 28-33, IEEE Computer Society Press, 1989.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
   Cameron Station
   Alexandria, Virginia 22304-6145

2. Library, Code 52 2
   Naval Postgraduate School
   Monterey, California 93943-5002

3. Professor James C. Emery 1
   Systems Management (SM/EY)
   Naval Postgraduate School
   Monterey, California 93943-5002

4. Commander William B. Short 1
   Systems Management (SM/SH)
   Naval Postgraduate School
   Monterey, California 93943-5002

5. Lieutenant Commander Gregory A. Clancy 1
   VQ-4
   7791 Mercury Road
   Tinker AFB, OK 73145-8704