Small Project Management: The 3-D Life Cycle

An Honors Thesis (HONRS 499)

by

Robin A. Moschel

Thesis Advisor

Dr. Ramarathnam Ravichandran

R. Ravichandran

Ball State University

Muncie, Indiana

Completed: December 1992

Date of Graduation: December 20, 1992

TABLE OF CONTENTS

ABSTRACT

During the summer of 1992, I worked as a systems intern at a large pharmaceutical company, Eli Lilly and Company. I was assigned what is considered to be a *small project* in this context. This project consisted of creating a computerized tutorial for an auditing program currently in use in the auditing field. Throughout my internship, I spent much time attending seminars and classes. Approximately 50 person days were devoted to the completion of my project.

In order to complete this project, it was required that I first learn the concept of small project management. Because of the limited amount of time and resources, small projects must be managed differently than larger projects. Many of the same procedures are performed, however, on a much smaller scale. In addition, these procedures often have to be adapted to satisfy the distinct needs of each small project. The small project development methodology in which I followed when creating my tutorial can be referred to as the 3-D Life Cycle.

This paper first discusses the major issues relating to project management in general. It will then narrow the discussion to the systems development life cycle methodology in which is most appropriate for small projects. Each phase of the 3-D life cycle will be broken down into the various tasks and examined in greater detail. Finally, the effect that using this methodology had on my project will be evaluated.

After reading this paper, you will have a better understanding of the differences between small and large projects. It should be clear that some project management technique should be applied to all projects, regardless of size. In addition, you should realize the benefits that the 3-D life cycle has to offer the systems developer and how to apply this methodology to your own small projects.

I. PROJECT MANAGEMENT

A *project* is any endeavor for which an objective, a beginning, and an end can be clearly defined (Westney, 1992). Therefore, any job that has a deadline with a set of objectives to be met is considered a project. The person responsible for making sure that a project gets completed is a *project manager*. It is the project manager whose implementation efforts bring the expected benefits of technology to the organization. Therefore, it is vital that a highly competent person be chosen for this position (Wysocki and Young, 1989). Project Management can be defined as a set of principles and methods for effectively planning projects and delivering them on time, under budget and to specifications (Wallace and Halverson, 1992). Managing a project requires planning, decision making, resource scheduling, estimating, performance measurement, and control. Effective project management is essentially the ability to lead those people doing the work, make the right decisions, and to communicate well with all those involved. Utilizing the proper project management techniques is one of the keys to remaining competitive in today's environment.

Successful project management relies on the three sets of skills illustrated in figure 1. Problem-solving is the first of these skills. The second skill is managing the project's effort. Leadership in client, management, and staff relationships is the third important skill for project management. Many problems may be incurred when working with a project. Communication, setting goals, staying focused, organizing, and meeting deadlines are just

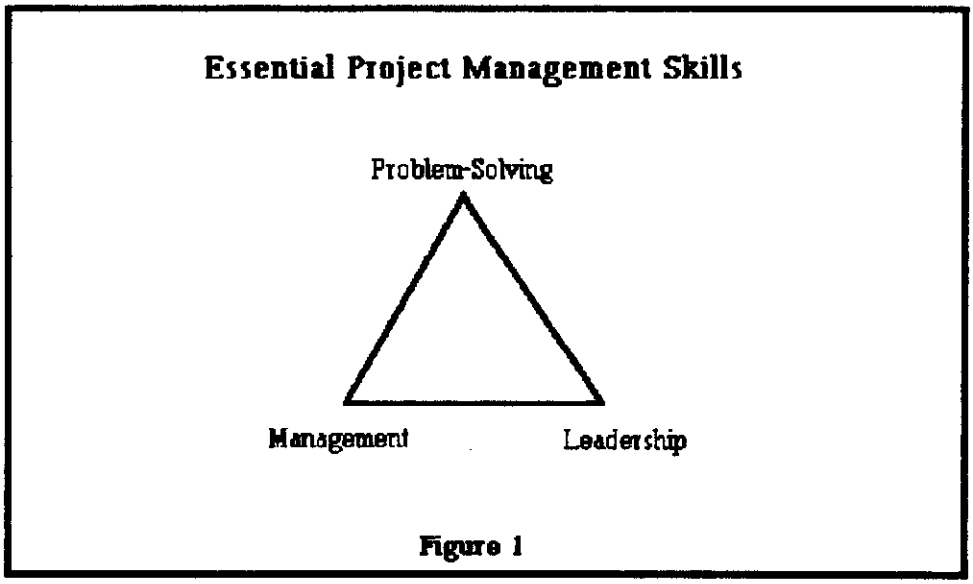

Figure 1

a few of these problems. However, practicing the basic project management techniques described throughout the remainder of this paper and exercising the skills mentioned above will help overcome many of these problems.

Problem-solving skills. Problem-solving skills assure that you are solving the right problem with the right solution. The first step in problem-solving is to define the problem. It is important to discover the real problem rather than the symptoms. Once the problem has been found, an objective must be set for solving the problem. This objective must be specific and feasible. It is pertinent that the existing situation be understood before a solution is contemplated. This requires examining all the elements in the present environment of the problem.

The next step in problem-solving is to identify the requirements for the solution. The requirements are the main focus of the rest of the process. They are absolutely necessary in order

to evaluate what the available alternatives are. Extensive research should be performed to come up with several alternatives to choose from. A decision must be made as to which alternative best solves the problem and that alternative should be implemented. After implementation, one must not forget to evaluate the result (Duarte, 1992). Problem-solving is the foundation of all project management methodologies.

Management skills. Management skills increase project efficiency and the quality of the results. Project management skills can be broken down into various procedures. For instance, agenda-setting is one valuable management procedure to be performed. It requires an examination of the necessary tasks to be performed and the amount of time available. Another procedure is task structuring. The project manager must have the management skills to break the project down into tasks and develop a plan of action to follow.

If the appropriate management skills are not exercised when developing a small project, the result will be a haphazard, unplanned and unstructured system. Loss of assets, customers, and revenue can be the consequences of poorly managed projects (Burch and Grupe, 1991).

Leadership skills. The primary resource in any project is the leader or project manager. The way to get things done through others is by utilizing good leadership skills. The leader is challenged to coordinate clients, managers, technical specialists, and peers with the needed skills. In order to coordinate others,

the project manager must first manage his/her own time better. This can be accomplished by planning effective uses of time, overlapping delay time with other projects, reducing scope to meet deadlines, and doing whatever it takes to get the job done. It is critical that the leader be able to motivate all those involved to do their best work and to cooperate.

In order to be an effective leader, the project manager must provide his/her team members with adequate guidance. In order to do this, he/she must develop a results oriented objective. This objective must be relevant to all parties involved. Likewise, the requirements developed must be well defined and aimed toward success. The team should also be involved in the estimating process. Afterall, only the team members truly know how fast they can work. The priorities need to be clearly understood by all team members. Finally, the project leader must promote team spirit. All successful projects require the direction of an effective leader.

Small projects can benefit from greater consistency and thoroughness in each of these three skill areas. However, most projects gain the greatest improvement by placing additional effort on understanding the problem and managing the effort. Consistently and thoroughly applying these skills need not be expensive in effort or duration. At marginal cost, they can significantly improve quality, efficiency, and responsiveness. The importance of each of these skills should become increasingly apparent as the small project development life cycle is investigated.

II. SMALL PROJECTS VS. LARGE PROJECTS

There are specific criteria that distinguish a small project from a medium to large one. The life span of larger projects can vary from a few months to several years. They demand very substantial financial and human resources. In most cases, large teams of programmers, analysts, technical experts, managers, and consultants work together to complete the project. It is not unusual for many of these personnel to devote all of their time to working solely on the large project. The design and development of a large project may crossover into many departments. The final results of a large project may have a great impact on the organization. Large projects generally require a vast amount of resources and are very important to the efficiency and effectiveness of an organization.

On the other hand, a small project requires only five days to two months of effort. The average amount of time spent on a small project is 30 - 40 person days. A small project is usually highly deadline driven and is of high priority. It must have only a single objective. If it has more than one objective, then it consists of more than one small project. There should be one major decision-maker. Therefore, little delegation is required. Significant expenses should have already been approved. The project should be defined in two days or less. These criteria should be used to understand the risks involved and to help plan around them.

The distinction between a small project and a large project

may also be subject to organizational factors. What is considered to be a small project by a major corporation may actually be a large project to a small business. In addition, the attitudes of the information executives may have an effect on the classification of a project. Another organizational factor is the industry in which the company operates in. Projects may receive a different level of attention according to the importance the industry places on particular projects. Therefore, the attributes of a small project may vary from company to company. However, each organization should establish a set of criteria in order to determine which projects are considered small, mid-sized, and large.

Small projects can be just as important to a company as larger ones. It is important that IS management recognize this fact. The value of successfully completing a small project can be much greater than the cost of the project itself. Likewise, the project should not be treated insignificantly because of its low cost. Managing such user requests as system enhancements and small applications as small projects reduces the need for large costly projects to be developed. Managers should be able to recognize when it is more efficient to treat a situation immediately as a small project rather than waiting for it to grow into a large project (Singer, 1991). Therefore, there is a need to apply project management techniques to small projects as well as larger projects.

III. SMALL PROJECT MANAGEMENT

It is a common belief that, unless you're managing a large project, project management techniques and methodologies have little relevance. However, this simply is not true. Managing a small project's effort requires the scaled-down use of conventional project management techniques. These techniques include: systems' life cycles and standard tasks, initial project sizing, high-low task estimating, prioritizing, staffing and scheduling, tracking, and quality assurance reviews. Each technique will be discussed in more detail in the sections that follow.

Small projects are proportionately more difficult to manage than larger ones. There are several reasons for this fact. Perhaps the most difficult aspect of managing small projects is the problem of dealing with many small projects at once. In addition, everything seems to be needed yesterday. There are many conflicting priorities that change rapidly. Resources are often scarce. There doesn't seem to be enough time for planning. The scope of the project almost always grows. In addition, it is often difficult to gain successful closure of small projects. These challenges can be frustrating and difficult to deal with. Using a small project development methodology can help to improve the organization and consistency of small project work.

There are three characteristics that an effective small project management methodology should possess. First of all, all projects should be handled with a standard approach. This will improve the project's efficiency and consistency. It does this by reducing the amount of time and effort the development process will

take each time. In addition, developers understand the process better and feel more comfortable using a standard approach. The second characteristic is that the development techniques be simple. The reason behind this is that clients are usually very involved with the development of small projects. It is necessary that the process be easy to understand and copy. The last characteristic is a quick result. The short duration of a small project requires that results be produced quickly and effectively. In addition, the dynamic environment in which most small projects dwell demands fast responses. These requirements should be incorporated when developing a systematic, consistent, and effective project management methodology (Westney, 1992).

The result of utilizing proper project management techniques is a faster delivery of a higher quality result with improved satisfaction on the part of the client or customer. This result can be gained by scaling down the methods of large project management techniques to the low-overhead minimum needed for efficient and effective small projects. The systems development methodologies for larger projects usually consist of seven to nine phases. The most common system development life cycle (SDLC) methodology is shown in figure 2. Although this methodology is not appropriate for small projects, it should not be disregarded. It should be considered with flexibility in mind (Kizior, 1991). If the SDLC was applied to small projects, the amount of time required to complete the project would increase significantly. The process would be drawn out to the point where the efficiency and

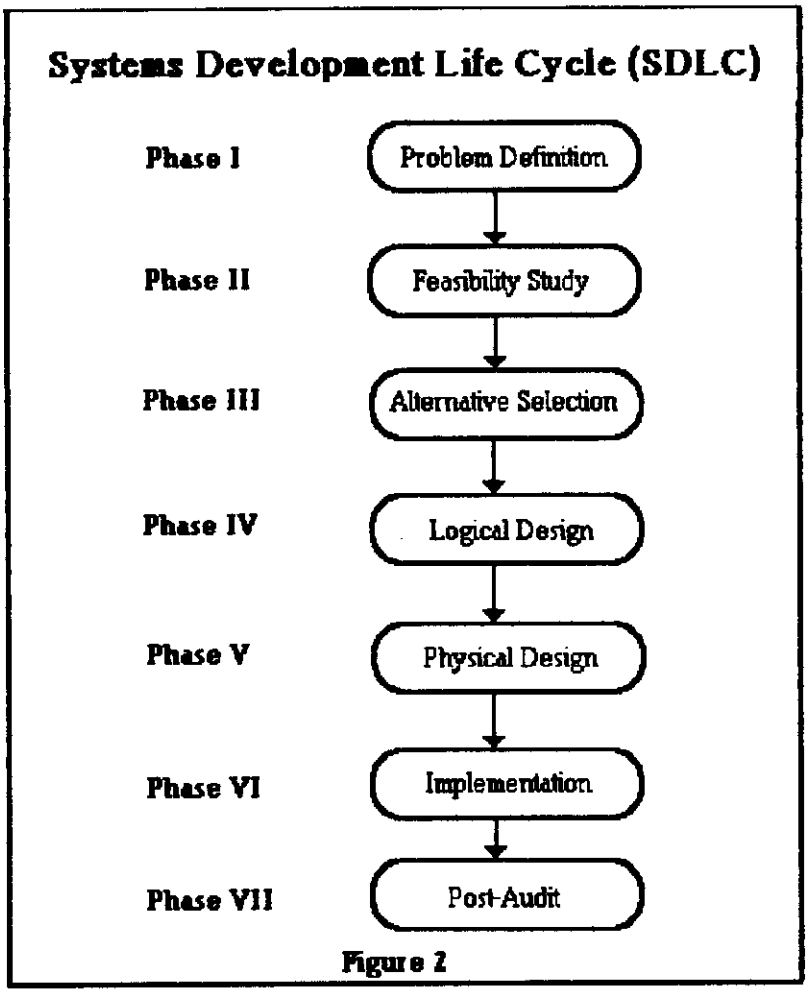

Figure 2

effectiveness of the methodology would suffer. Since most small projects are implemented by a single project manager or a small team, many of the steps in the SDLC would be unnecessary.

A similar systems development life cycle that is suggested for smaller projects is shown in figure 3. This methodology is commonly called the 3-D Life Cycle. The 3-D life cycle consists of the define phase, design phase, and delivery phase. The first phase, define, produces an understanding of the problem, a measurable objective, and requirements for the solution. The design phase identifies alternative approaches and the recommended solution. The delivery phase reflects development and delivery of

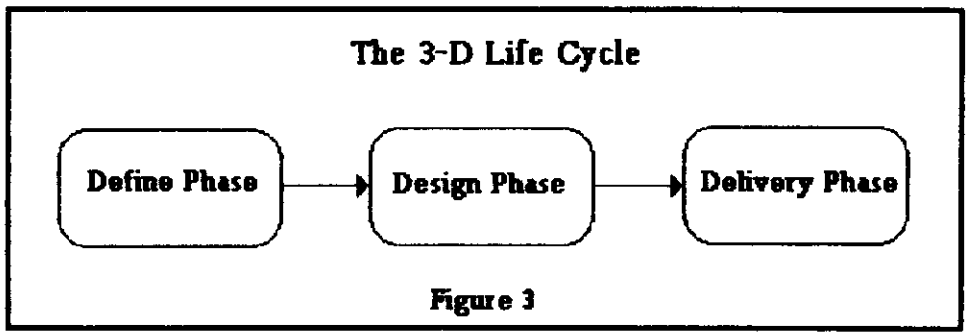

Figure 3

the solution. Each phase can be broken down into several tasks. Although many of the same tasks are performed as in a larger project, the tasks are scaled down and take less time. We will next examine the 3-D life cycle by discussing the individual tasks of each phase.

IV. DEFINE PHASE

The purpose of this phase is to fully understand the problem or opportunity and its environment. The requirements for a successful project must be defined here as well. The project must be prioritized and a schedule must be developed. The tasks of this phase should be planned and managed as one activity rather than as individual tasks. It is recommended that this phase take no longer than two days to complete. If it takes longer than two days, it is probably not a small project and deserves a different systems development methodology. A task template for this phase can be found in figure 4.

Initial project sizing. The first task of any project is similar regardless of the size of the project. The purpose of this task is to determine if the project is indeed small and to set a

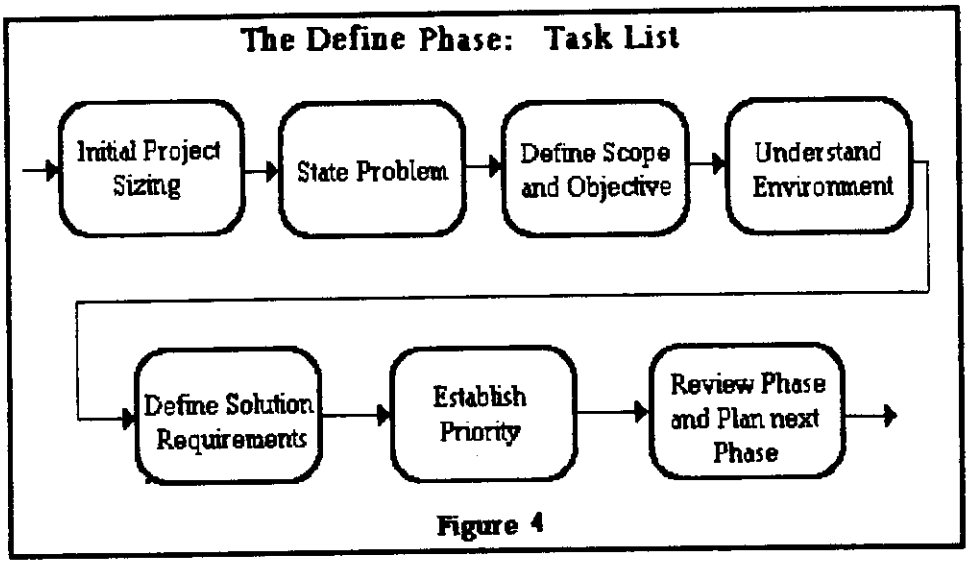

Figure 4

plan for keeping it small. Sometimes a larger project may be masquerading as a small one. It is important to identify that fact early and either reduce the scope or use more industrial-strength methods intended for larger projects.

The first step in project sizing is to make a first guesstimate of the project's size and the amount of effort required. Effort can be defined as the amount of time one skilled person, working full-time, would take to complete the project (Westney, 199) It may be necessary to rely on past experience when making this estimate. If you know of others who have knowledge of the project, it may be helpful to ask their opinion. The best estimates usually come from people with relevant experience in the type of work being estimated. If all else fails, just make a guess according to the facts that you already have. This technique can be referred to as S.W.A.G., a Systematic, Wild, Anatomical Guess based on relevant experience and intuition. Use your estimate of

effort to verify that the project is indeed a small project.

The next step in this task is to evaluate the project against success criteria. The questions that follow will help to evaluate factors that affect the size of the project. They also provide hints for keeping a project small, or reducing its size to make it more manageable. These criteria can be used to understand the risks and plan around them.

How many decision-makers are involved? More than one decision-maker will make a project take longer. The reason behind this is that more negotiations and compromises will have to be concluded. The project manager may have to make presentations to other managers in order to persuade them to agree with his opinion. It will take more time to go through levels of authority to get a decision accepted.

Is the project subject to significant delay? This will, of course, prolong the duration of the project. In addition, the requirements of the project may change in the meantime allowing the scope to expand. A backup plan should be developed for projects of this nature.

How big is it? A small project, by definition, is less than 40 person days of effort. This may vary according to the specifications of the project or the culture of the organization. However, a small project should not exceed 4 months of person days.

How stable is the environment that the project will support? A dynamic environment might make the project take longer and cost more. It may become difficult to develop plans and

schedules. Perhaps another systems development methodology should be sought under these circumstances.

Will the project require unbudgeted expenses beyond your manager's signature authority? Formal approval processes may add costly delay. Most small projects should have low costs. The funds required are usually too small to require upper management approval. Therefore, they should be handled by departmental management. This may, however, depend on the policies of the organization.

Does the project depend on unproven new technology that could cause unanticipated problems or extra effort? This could increase the amount of time needed to complete the project as well as increase the costs. In addition, the scope and requirements of the project may also increase transforming the project into a larger one (Westney, 199).

The last step in project sizing is to choose the appropriate first phase plan. This plan should be based on the results of the previous steps and is subject to change as the project progresses. However, there is a critical success factor for each small project: The first phase should be able to be completed within two person days or less.

The result of the initial project sizing task should be an early understanding of the size of the project, its risks, and the steps needed to complete the first phase plan. These results should be documented in a project file.

State the problem. This task documents your understanding of

the problem, symptoms, causes, parties affected, and benefits of solving it. In order to achieve this task, the developer must work closely with his client. The first step is to determine what the initial problem appears to be. Often, this will only be a symptom of the real problem. However, once this problem has been probed the real problem should become apparent. One way to determine whether the problem is the true problem is to examine the question, "If this problem is solved, will all the symptoms go away?" If any of the symptoms still exist, it is not the real problem.

It is also important to ascertain whose problem it really is. You must discover who is accountable for the problem and who will be impacted by the solution. The next step is to find the source of the problem. Perhaps one of the most important purposes of this task is to determine why the client wants to solve this problem. If the reason does not coincide with the business objectives of the company, there may be no need to go any further with the systems development process. The final result of this task should be an agreement between the systems developer and the client regarding the afore mentioned issues.

Define scope and objective. This task documents how big the project is. It also sets the objectives that a successful project must meet. In the absence of a clear scope and objective, any small project will soon become a larger one.

There are three factors that determine the scope of the project. The first factor is exactly what activities are to be accomplished. How many clients are to be involved is also a factor

that needs to be considered. Having more clients to work with means more needs to satisfy and probably a larger scope. The project completion deadline will also help determine the scope of the project. A small project should be completed within 30 - 40 person days.

A good measurable objective should state why the project is to be solved, rather than how it will be done. It should not take more than five words to describe a small project's objective. A clear objective will be used to evaluate the requirements, manage the scope, determine when the project is complete, and evaluate the success of the project. The developer and client should both agree on the success criteria of the project determined in this task. The client will be the owner of the system to be developed and should take an active role in designing the system. The management of small projects should be based on the mutual understanding of all participants (Gutierrez, 1990).

Understand the environment. This task is very specific to the nature of the project. It involves an early look at all the things that may otherwise be discovered too late. It is extremely important to look at the strengths and weaknesses of the environment the project will be implemented in.

Both the strengths and weaknesses should be evaluated. The developer should strive to maintain the benefits that the current environment has to offer. However, the developer should work to enhance, fix, or replace those aspects of the environment that hamper the situation's effectiveness.

There are certain aspects of the environment that should be examined. The resources available and constraints of the current environment should be quantified. For instance, the developer should be aware of exactly how much money can be spent on the project. The developer should also know which vendors and other employees he/she can consult. Equipment and supplies are two other resources which need to be considered. The most common constraints include time and knowledge.

The last step of this task is to evaluate the risks involved. *What can go wrong?* There is always a chance that not enough time will have been allotted to complete the project. Unstable hardware and software is also a danger. A backup plan must be developed to account for these risks. An understanding of the dynamics of the environment can only result in an improved chance of success. The results of this task should also be documented.

Define solution requirements. Project requirements are the framework for managing scope. These requirements can be used to reduce the scope in order to make a larger project small. As well, they are criteria that can be used to evaluate the success of the project against.

The requirements are very fundamental to the project. The actual functions that the solution should support need to be expressed in a verbal objective form. The developer must know exactly what the project is expected to do. In addition, the developer must know how well the project is expected to perform. This must be in a quantifiable form. A specific guideline should

be developed as to how the result will be measured.

In order to manage the product of the project, it is important to know what information and data is required. This may include data used by machines or personnel. The developer must additionally understand what limitations may be placed upon each function of the solution. This may have an impact on the implementation of the project. There is one other requirement that is often neglected. That requirement is the subjective or qualitative aspects of the project. An example would be the extent of user friendliness the system possesses. The result of the requirements task is an understanding of the criteria in which the solution will be evaluated against and an agreement upon the scope of the project.

Establish priority. Once the requirements have been established, priorities can be set for the project. Prioritization is the key to reliable scheduling. The purpose of this task is to determine which projects warrant the highest priority rating. Prioritization is the key to consistently delivering projects quickly. Lack of prioritization results in insufficient resources, inefficient interruptions and rework (Wysocki and Young, 1989). Projects should be prioritized according to both urgency and importance. Any time that the situation changes, the priorities should likewise be adjusted.

Urgency is measured depending upon how quickly the result is needed. A project's urgency should be classified as immediate, moderate, and when possible (Westney, 199). For example, a project

labeled immediate must begin today. In most cases, the definition phase should be ranked with higher urgency. Importance can be judged according to return on investment, politics, and personal goals. Likewise, importance can be classified as extreme, moderate, and less important. Once these classifications have been made, they should be reviewed with the client. The result of this task is a more efficient use of the developer's time and increased control over priorities.

Review phase results and plan next phases. The purpose of this task is to build and maintain a plan for a successful small project. It should be performed throughout each phase of the project. The deliverables of each task should be examined in the review of the phase. Use a calendar to predict the completion date. Compare the predicted date to the due date. If they differ, make adjustments to scope, priority or resources as needed.

Having defined the project, the developer can next plan the remaining phases. The design phase will now be examined.

V. DESIGN PHASE

The purpose of the design phase is to identify the alternative ways the requirements can be met, to prioritize the requirements for scope cut-off, and select the best alternative that meets the most requirements within the due date and client's cost constraints. The tasks included in the design phase can be found in figure 5. These tasks can be managed as one large activity or as separate tasks. This will depend on the size of the project and the nature of the skilled resources that are tapped.

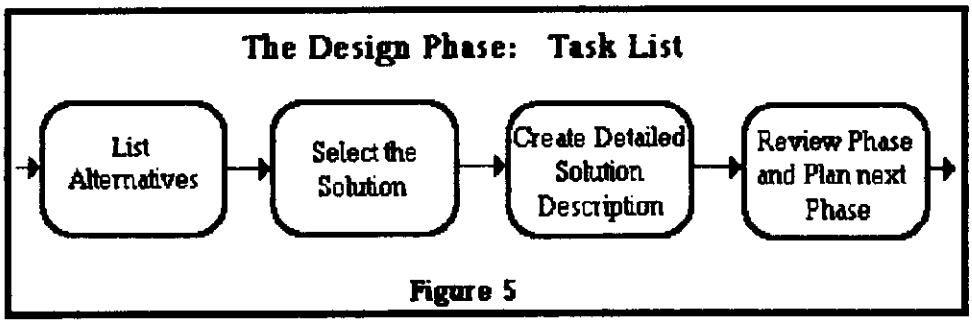

Figure 5

This phase requires access to skilled resources. Prior experience in projects of the same type makes your peers valuable sources of alternative approaches. The design phase entails a great deal of creativity. Therefore, group brainstorming may be beneficial.

List alternatives. The first task of this phase is to list the alternatives. If no alternative solutions are apparent, perhaps a more detailed understanding of the problem is needed. The purpose of this task is to establish clear alternatives. There are always options. One option that always exists is to remain status quo. All of the project options should be considered and documented in the project file.

The first step of this task is to prioritize the functional and performance requirements of the project to determine the possible scope cut-off. The functional requirements are the activities that the system will participate in. For instance, a system may be required to keep track of the sales data of a particular product. The performance requirements relate to how well the system performs its functional activities.

Some alternatives may be eliminated simply because they do not fit within the scope of the project. In order to come up with alternative solutions, it may be helpful to use a creative group process, such as holding a brainstorming session. Alternatives should be generated that will meet functional requirements, performance requirements, and schedule and cost requirements. Also, any work that need be should be delegated in this task. Therefore, the developer must identify assignment options. The developer may chose to do all the work by himself/herself or get help from other employees or consultants. If too much of the work is delegated, the tasks will undoubtedly take longer to perform. The result of this task is a set of clear options for alternative ways to resolve the problem.

Consider alternatives and select one. This is the last chance to cost-effectively reduce the scope of the project or add to it in the development process. Within the cost and schedule constraints, the developer must select the solution alternative that achieves the highest quality results for the most scope. The main objective of this task is to choose the best solution.

When making this decision, the developer must keep in mind the scope cut-off, schedule and cost constraints, and the various trade-offs between alternatives. It should be identified why the selected option is a better alternative than the others considered, including the alternative of doing nothing at all. The developer must decide which tasks will be done and which ones will not. He/she must choose the solution and be able to justify this

decision to the client. The final result of this task is an approach for delivering a doable scope within the project's constraints. The developer should document the strengths and weaknesses of all the alternatives, including the one chosen.

Create detailed solution description. This task produces a documentation of the chosen solution. This is especially important for delegated tasks. In this task, the developer is charged first to plan the tasks and then to work with team members.

The first step is to fully document the chosen solution. The developer must carefully plan the tasks which require assistance. He/she needs to describe the needed skills and write an overview description of each task.

The next step is to work with team members. Each team member must understand the task and skills expected of him/her. The developer should discuss the task purpose with the team member and how it will be evaluated. The team members may have questions. It is the responsibility of the developer to satisfy any queries that the team members have. Estimates and tracking methods must be agreed upon. Finally, quality assurance review points must be established.

It is important to keep in mind that, as more team members are employed, the extent of project management effort begins to dominate the developer's time. When the majority of the work is not being performed by the developer, larger scale project management methods need to be utilized. The result of this task is a complete documentation of the solution and the recruitment of

team members to participate in the project.

Review solution and plan next phase. This task is a continuation of the last task in the define phase. It includes the same steps and will result in an updated plan.

The delivery phase will be studied in the following section.

VI. DELIVERY PHASE

The delivery phase places heavy emphasis upon *doing*. However, the purpose of the delivery phase is to not only to do the work, but to follow through with the often omitted tasks that assure that the work is done well. While this phase often requires the greatest amount of effort, its success is often subject to the quality of the work performed in the prior two phases. The tasks that make up the delivery phase are shown in figure 6. Quality assurance reviews should be performed after each task in the delivery phase.

Build / buy solution. This task requires the bulk of the effort placed upon the deliver phase. It is very specific to the nature of the solution. The selected alternative solution should determine whether the system will be built or purchased. The steps required to develop the solution should have already been identified, estimated, and scheduled in the previous phase. Simply follow the plan laid out in the last task, review and plan the next phase. The end result of this task is an implemented solution. Implementing this task well is the purpose of all the other tasks in the 3-D life cycle.

Test solution. This task proves that the chosen solution

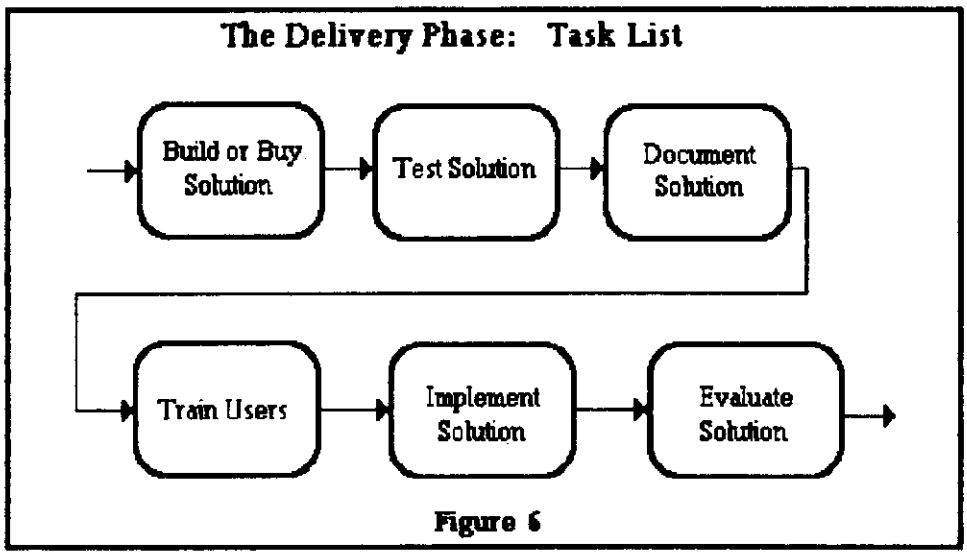

Figure 6

works as planned to meet the requirements. Good requirements are essential input for testing. In addition to assuring that all functional requirements are met as agreed upon, performance requirements should be tested as well.

The following items should be tested as a guideline. The functional requirements that were established in the scope cut-off should be fulfilled. Volume testing can be used to determine whether the performance requirements are met. Use stress testing to determine behavior at the limits of design. The result should be evaluated against information requirements. The developer should also verify that the solution operates within the specified limits or constraints. The client's views regarding the solution's responsiveness to his/her requirements must be evaluated. Once testing has been completed, a plan of action should be established that will correct any weaknesses that were discovered. The result of this task is proof that the solution works as planned.

Document solution. This task provides the documentation needed to use and maintain the solution. It also records the project documentation that shows how the solution was planned and implemented. Some examples of documentation include user's manuals, systems overview information, operator instructions, maintenance programmer background information, and recovery instructions (Wysocki and Young, 1989). Much of this documentation should have already been completed in the first two phases. Documentation should be undertaken throughout all the phases of the life cycle. The benefits of doing so include obtaining the support of management, gaining the acceptance of the clients, and creating a more durable system (Garceau and Jancura, 1990).

In addition to documenting the development of the system, it is important to include user documentation. This documentation informs the user of how to use and maintain the system. Any functional requirement of the system should be documented for these purposes. The result of this task is a document showing how the developer arrived at the chosen solution.

Train users. This task assures that the user or client of the solution has received the proper training to use the solution consistently and effectively. Often systems developers neglect the aspect of training when implementing a new computer system, especially when implementing a small project (El Shazly and Durand, 1991). However, the success of the project may depend on how well the users understand and operate the system.

The first step to be performed in this task is to determine

the extent of the training needed. If the client has worked closely with the developer in designing the system, the client may not need a great deal of training. Once the level of training has been established, the developer must generate the training materials. Of course, the next step is to actually perform the training and to evaluate the results. Training involves controlling or monitoring users to provide for the qualified and proficient execution of tasks (El Shazly and Durand, 1991). Another aspect of training is teaching the user to make informed choices and decisions regarding the system.

It is important to establish a method of ongoing support. This will account for user turnover. The result of this task should be that the clients obtain the maximum use of the solution by learning from the developer.

Implement solution. In this task, the system is fully installed and functions as an experimental working model until closure of the project. This is a client acceptance task that certifies that the solution has been fully completed as specified in the requirements. During this task, it may happen that further functionality is requested by the client. The reason for this may be that the client simply did not realize the need until this point or that the functionality was eliminated during the design prioritization task.

The first step of this task is to verify with the client that the requirements have been met. If it is difficult to obtain client acceptance at this point, chances are that the client was

not properly involved in the earlier phases. The developer must demonstrate that adequate documentation has been produced and that the proper training was performed. The client's business needs and objectives must be met by the solution. If the project meets all the promised deliverables, the client should grant acceptance. The result of this task should be a happy client.

Evaluate solution. This task provides an opportunity to gain closure on the project, to identify what went right and wrong, and to describe how it could have been done differently. A checklist is provided in figure 7 that will help determine the success of the project. Once the project has met the approval of management and the client, the project can be declared finished.

VII. EVALUATION OF 3-D LIFE CYCLE METHODOLOGY

During my internship, I was strongly encouraged to utilize the 3-D life cycle methodology to develop my project. The structure of my project met most of the small project criteria described earlier. It had the single objective to interactively train users to use PHARMA Audit in a user friendly and comfortable environment. Although I worked closely with my supervisor, I had the authority to make all of the design decisions. Little or no delegation was required. At the conclusion of my internship, my project was at the *implement solution* task in the delivery phase. The tutorial had been built, tested, documented, and the users were partially trained.

The scope of my project was somewhat larger than that of the ideal small project described by the 3-D methodology. However, it

Evaluation Checklist

- ☐ Has the objective been met?
- ☐ Was the project completed on time?
- ☐ Did the project stay within the budget?
- ☐ Is the client happy with the final results?
- ☐ Were communication lines kept open?
- ☐ Were alternative solutions evaluated?
- ☐ Was any work deferred to later projects?

Figure 7

was still too small to warrant a full fledged SDLC methodology. My project did take slightly longer than 30 - 40 person days to complete. In addition, I spent approximately 5 days in the define phase. Perhaps the reason for this may have been my inexperience in project management. It could also have been due to the fact that the company in which I was employed has a higher scope cut-off line for small projects than does the average company. Regardless of the reason, I believe that my project was appropriately handled as a small project. It is of my opinion that the time line should be adjusted to meet the particular situation. Not all small projects are going to adhere to the 30 - 40 person day deadline for one reason or another. As long as most of the criteria are met and the project adheres to the standards followed by the organization, the developer should chose to utilize the 3-D life cycle. Otherwise, valuable resources and time may be wasted performing unnecessary tasks.

When I began my internship, many of the tasks in the define phase had already been completed by my supervisor. The initial project size was estimated to be 45 person days of effort. This estimate had been based upon past experience and intuition. The problem had been investigated, the solution requirements established, and many priorities set. I met with users as well as my supervisor in order to fully understand the problem and solution requirements. My supervisor and I worked closely together to define the scope and objective of the project. I examined the current environment and listed the strengths and weaknesses which were later incorporated into the requirements. Once the project was clearly defined, I developed a project plan for the next phases. Since I was integrating new multi-media software, I found that it was particularly hard to determine in advance exactly how long certain activities would take. Periodically throughout each phase, I updated my project plan to reflect any changes in scope that were incurred.

I next entered the design phase. With the input of my supervisor, I listed four alternative solutions. A meeting was held with the group of users to discuss the alternatives. The solution was chosen based upon user preference and the likelihood of success. It fit within the initial scope and constraints of the project. A detailed description of this solution was developed to serve as a guide for the remainder of the life cycle. It was also used to update the project plan to reflect the tasks specific to the selected alternative.

When I reached the delivery phase, it was clear what had to be done. I gathered my tools and begin building the tutorial. The steps required to perform this task had already been outlined in the previous phase. I did not have to take time to deal with scheduling and prioritizing. For the most part, I was able to follow my project plan. Occasionally, I would run into small hardware or software problems which would delay the completion date. However, I feel that many problems were avoided simply because I followed this methodology. The solution was tested by allowing users to test run the tutorial and was found to meet the requirements. A document was created describing the tasks performed, how to use the product, and the specific features of the tutorial. The users were heavily involved throughout the process. Therefore, a vast amount of training was not required. When I completed my internship, the project was ready to be implemented.

Since this was my first small project as a systems analyst, I found the 3-D methodology to be especially helpful. It provided me with a road map. Although there was a lot of learning involved, I felt much more confident in starting my project knowing what specific tasks should be performed. The success of my project is greatly due to the fact that I followed the 3-D life cycle. I received a lot of guidance from my supervisor, who also utilizes this methodology on small projects. It became apparent that this methodology coincides with the attitudes and policies of this organization.

VIII. CONCLUSION

There should be no doubt that small projects do indeed warrant the use of a specialized development methodology. Each organization should define its own criteria for small and large projects and develop a methodology that works best for that company. The 3-D life cycle is an excellent model and may be adapted to meet the circumstances of individual organizations.

Regardless of the method used, it is important to recognize the importance of small projects to organizations. They cannot be ignored or neglected. Small projects lay the foundation for larger projects. They are a part of the everyday functioning of the company. They also encompass a great deal of the systems developer's time and effort. Therefore, they must be properly managed.

References Cited

Burch, John and Fritz Grupe. 1991. "Methods and tools for upgrading systems development." *Information Executive,* Volume 4, Number 3, Summer, p. 49:7.

Duarte, James E. 1992. "Problem-Solving techniques for continuous improvement." *The Management Accounting Magazine,* Volume 66, Number 5, June, p. 13:3.

El Shazly, Hassan E. and Douglas E. Durand. 1991. "User training is necessary throughout the SDLC." *Journal of Systems Management,* Volume 42, Number 8, August, p. 6:5.

Garceau, Linda R. and Elise Jancura. 1990. "Documentation and training as a systems development tool." *The CPA Journal,* Volume 60, Number 11, November, p. 84:5.

Gutierrez, Oscar. 1990. "Design and development of small data management systems." *Journal of Systems Management,* Volume 41, Number 6, June, p. 29:6.

Jones, Mary C. and Leon R. Price. 1991. "A modified SDLC." *Information Executive,* Volume 4, Number 3, Summer, p. 45:4.

Kizior, Ronald J. 1991. "Using a flexible SDLC approach." *Information Executive,* Volume 4, Number 3, Summer, p. 56:2.

Singer, Larry M. 1991. "Evaluate, don't procrastinate: seven steps to sizing up user application requests." *Computerworld,* Volume 25, Number 7, February 18, p. 75:2.

Wallace, Ron and Wayne Halverson. 1992. "Project management: a critical success factor or a management fad." *Industrial Engineering,* Volume 24, Number 4, April, p. 48:3.

Westney, Richard E. 1992. *Computerized Management of Multiple Small Projects.* New York: Marcel Dekkar, Inc.

Wysocki, Robert K. and James Young. 1989. *Information Systems: Management Principles in Action.* New York: Wiley.