[Buck79] F. Buckley, "A Standard for Software Quality Assurance Plans," *Computer*, Vol. 12, No. 8, August 1979.

[Cain75] S.H. Caine and E. K. Gordon, "PDL: A Tool for Software Design", *Proceedings of the National Computer Conference*, 1975.

[Denn81] P. J. Denning, "Eating Our Seed Corn", *Communications of the ACM*, Vol. 24 No. 6, June 1981.

[Dolo76] T. A. Dolotta and J. R. Mashey, "An Introduction to the Programmer's Workbench", *Proceedings of the Second International Conference on Software Engineering*, October 1976.

[Habe80] A. N. Habermann, "An Overview of the Gandalf Project", *Carnegie-Mellon University, Computer Science Research Review 1978-1979*, 1980.

[Kern81] Brian W. Kernigan and John Mashey, "The UNIX Programming Environment", *Computer*, April, 1981

[Moha81] S. N. Mohanty, "On Software Verification & Validation", Software Verification and Validation Symposium, Defense Communication Agency and Mitre, June 9-10, 1981.

[Phis79] M. Phister, Jr., *Data Processing Technology and Economics*, 2nd Ed., Digital Press, 1979

[Step80] D. Stephan, et al., *DoD Digital Data Processing Study: A Ten-Year Forecast*, Electronic Industries Association, 1980.

[Teic77] D. Teichroew and E. Hershey III, "PSL/PSA: A Computer-Aided Technique for Structured Documentation of Information Processing Systems", *IEEE Transactions on Software Engineering*, Vol. SE-3, No. 1, 1977.

[Teit81] Warren Teitelman and Larry Masinter, "The Interlisp Programming Environment", *Computer*, April 1981.

EVALUATION OF SOFTWARE DEVELOPMENT LIFE CYCLE METHODOLOGY IMPLEMENTATION

Prepared by:
Fred van den Bosch
John R. Ellis
Peter Freeman
Len Johnson
Carma L. McClure
Dick Robinson
Walt Scacchi
Ben Scheff
Arndt von Staa
Leonard L. Tripp (editor)

TABLE OF CONTENTS

4.1 TASK 1 - CASE STUDY PREPARATION
4.2 TASK 2 - CASE/SITE SELECTION
4.3 TASK 3 - CASE STUDY AND ANALYSIS
4.4 TASK 4 - IDENTIFICATION OF FUTURE WORK
4.5 TASK 5 - PUBLICATION OF FINDINGS

5.0 PLANS

5.1 CWBS
5.2 PROGRAM SCHEDULES
5.3 RESOURCES

1.0 Executive Summary

The cost of developing, maintaining and enhancing software is a major cost factor in many projects. The inability to understand, on a quantitative basis, what factors affect this process severely limits the ability of an organization to make changes that will have a predictable affect on improving quality and productivity of software products.

In the past decade most software organizations have developed a life cycle approach for their organization. The approaches which describe the actions and decisions of the life cycle phases have been formalized as a methodology. Little has been done, however, to define a basis for comparison of these methodologies or even portions of these methodologies. Therefore, there is little data to guide management to direct its organization on what methodologies should be used in the life cycle phases in order to enhance performance in terms of cost, schedule, and technical quality.

This is a proposal for a project to develop a basis for a standard quantitative and qualitative analysis of a software life cycle methodology. The goals of this project are to define a process by which an organization can monitor its life cycle and develop this process to produce better quality software product at a cheaper and more competitive price. In addition, this project will provide a means by which methodologies can be compared across organizations or phases of the software development life cycle. This would be invaluable to large corporations that have many different software development organizations and large agencies who have their own internal software development agencies as well as funding other organizations for large software development projects. This project would provide data that would enable these corporations to specify methodologies to the suborganizations in order to have a positive control on the quality and price of the software product produced.

This project consists of two phases. Both phases will be discussed by this proposal but the actual funding request will only cover the pilot phase. The pilot phase is a one-year $100,000 project to validate the case study approach to this problem and to redefine the type of questions and methods by which to conduct the interviews and the case study analysis. This pilot project will be followed by a three year project that will begin by studying approximately seven projects and will be the start of establishing the data base to compare methodologies across organizations and phases of a software life cycle.

2.0 Project Definition

This section contains an overview of the problem this project is to address including terminology, significance, focus and benefits.

The sections consists of the following subsections:

- Introduction
- Background
- Project Objective
- Project Benefits and Impacts

2.1 INTRODUCTION

Software Life Cycle Methodology is a collection of tools, techniques, and methods which provide roles and guidelines for ordering and controlling the actions and decisions of project participants during the software life cycle. The Software Life Cycle is the time required to define, develop, test, deliver, operate and maintain a (software) system. It commences at requirements analysis and includes design, coding and checkout, test and integration, installation, and operation and support. It is completed when the (software) system is retired, replaced, or legitimately consumed (e.g., during warfare).

A software life cycle provides a basis for categorizing and controlling project activities. A software life cycle consists of different, but not necessarily distinct activities. Although the activities are continuous and not mutually exclusive, each results in visible and

accepted products. These activities are usually defined in terms of:

- o resulting products
- o required resources
- o required accepted products from previous phases
- o development methodologies used

These products include documents such as:

- o standards to be used in the project
- o specifications (requirements, design)
- o internal documentation (e.g., data dictionaries)
- o experimental results from simulations, prototypes, etc.
- o test data and/or reports
- o verification and validation data/results
- o acceptance criteria, test results, etc.
- o external documents (e.g., user manuals, reference manuals, etc.)
- o program/module listings

Typical Software Life Cycle phases, as identified in Figure 2.1-1, are preceded by an initial planning phase during which the technical, military (if appropriate), and economic basis for the project are established through comprehensive studies, experimental developments, and formal concept formulation.

The software life cycle actually begins with customer acceptance of the system specifications(s). During analysis, the functional performance requirements for the software configuration items are defined. This phase terminates with the successful completion of the Preliminary Design Review (PDR). Further allocation of requirements to software components occurs during design and culminates with the successful completion of the Critical Design Review (CDR). Coding and checkout addresses the reduction of the design to code. Limited checkout of units (modules) of code and corresponding logic and data structures are also done during this phase.

Integration of units (modules) occurs during test and integration, with major emphasis on verifying that overall system requirements have been met and includes the successful integration of hardware and software components.

Installation activity includes the installation of the system, including software, at the customer site with all steps necessary to verify that system (and software) performance has not regressed and that the system operates within a required or specified level of confidence in the operational environment.

Operation and support requires that the system's software operate properly on all input data used in the operational environment. The support aspect of this phase includes all resources and activities required to ensure that the system software continues to meet (or exceed) all required operational capabilities. Incorporation of new software (functions) and/or modifications to an existing system normally requires repeating the previous activities in the software life cycle. For example, integration of previously checcked out modules may be in progress while coding of revised units is taking place (a library or similar mechanism would be used to provide the necessary separation and control of these activities). Hence, the software life cycle is a continuous process throughout the total system acquisition life cycle.

Another key aspect of the software life cycle is the change control method, including the phased levels of control. This is an integral part of configuration management to ensure the integrity of different versions of the software.

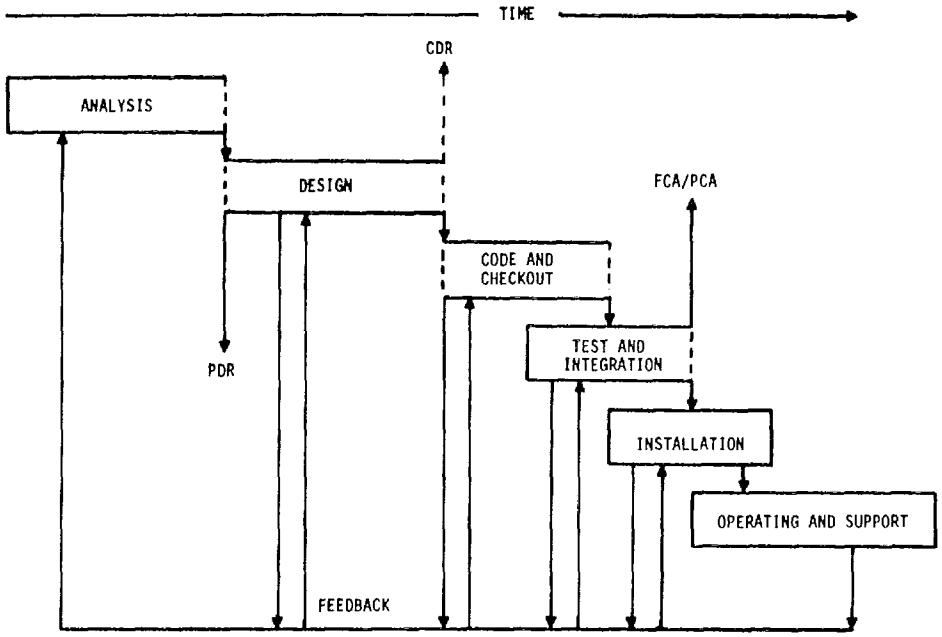
Figure 2.1-1 illustrates the software life cycle, identifying key milestones during the cycle: Preliminary Design Review (PDR), Critical Design Review (CDR), and Preliminary Configuration Audit (PCA)/Formal Configuration Audit (FCA).

2.2 BACKGROUND

Since the origin of "software engineering", numerous tools, techniques, and methodologies have been developed in the name of improving software engineering. Specifically, these tools, etc. have claimed to improve productivity (i.e., making the cost and schedule for software development more predictable and more competitive), to improve reliability (i.e., to reduce the number of "bugs" and speed their discovery), and to increase maintainability (i.e., to reduce the ratio of maintenance costs to the total life cycle costs).

With the criticality of software in virtually all future systems of any reasonable complexity, it is imperative to have tools and methodologies which actually meet these claims. Several recent studies have shown that our universities can supply less than half of the million software engineers that will be required by the end of the decade. The difference must be made-up by improved effectiveness in the tools and methodologies.

Figure 2.1-1



A major unanswered question then is, "How effective are these tools/ methodologies?" Ben Shneiderman, in his book on Software Psychology, advocates the use of controlled experiments to evaluate various methodologies. Some of his examples show that their effectiveness does not approach the initial intuitive estimate.

Other controlled experiments (e.g., Basili and Reiter's complete writing experiment) do not appear to be representative of the problems inherent in large complex systems. There is no valid reason to assume that the conclusions from small (several thousand lines of code) projects developed in a laboratory environment can be generalized to large (over 100,000 lines of code), complex, state-of-the-art software systems. The cost of applying such experimental approaches to the large system environment are prohibitive. These problems may be circumvented by collecting and analyzing meaningful data from large programs and providing analysis of such data. A major effort in this area has been undertaken by the Data Analysis Center for Software (DACS) at RADC. The analysis of the DACS, and other previous data acquisition efforts, has been difficult because of the lack of characterization of key parameters necessary to evaluate the methodologies used. A basic question is:

What methodologies/tools were used and to what extent were they applied?

There is no established framework for the collection of all necessary data to evaluate software development methodologies to determine whether the required efficiences can be realized.

2.3 PROJECT OBJECTIVE

In addition to providing the tools, enforcing the standards and controlling the transition of software through its concept, design, development, operation and maintenance phases, a software methodology must provide accurate predictions of project schedule, cost and quality throughout the software life cycle. The objective of this proposal is to develop the basis for a standard quantitative analysis of software life cycle methodology and will be accomplished through identifying and evaluating, by qualitative analysis, a software life cycle methodology with respect to applicability, value, and obsolescence.

2.4 PROJECT BENEFITS AND IMPACTS

This projects will confirm intuitions on life cycle practices and will reveal new insights into the forces present in the software life cycle process. Benefits can be identified in at least the following three areas:

1. A basis for standard quantitative analysis of software life cycle methodology will provide:

- o An increasingly more accurate predictions of cost, schedule, quality.
- o An information source for future studies.

2. The identification through comparative analysis, of the fundamental principles for developing software life cycle methodology to provide:

- o A framework for studying methodologies.
- o A point of departure for future life cycle system methodologies.

3. To establish the criteria for studying methodologies which will be:

- o Adaptable to many different kinds of projects across organization and method.
- o Utilized to study specific development phases e.g., requirements assessment, design methodology.

The process used to achieve these benefits includes the following impacts and side-effects:

1. The use of the case study as an educational tool providing:

- o A permanent record (history) of methodology use of project.
- o Alternative hypothesis analysis in retrospect.
- o Evaluation capability by different types of teams including both on going and in retrospect.

2. An understanding of how an organization does business providing:

- o A basis for individual company self analysis including the identification of weaknesses in existing approach(s) and providing better predictability of resource needs for projects.
- o Discussions with "proposal team" to allow private feedback.
- o A vehicle for company to improve image and marketability.

3.0 Technical Approach

This section contains an overview of our technical approach to the problem, emphasizing those aspects of the proposal plan which are particularly important and which reflect a distinct approach to the issues raised in this project.

This section contains an overview of:

- o Approach Elements
- o Rational for Approach and Rejected Alternatives

3.1 APPROACH ELEMENTS

The basic strategy of our approach is to conduct comparative case studies. The case studies are performed when an interesting yet poorly understood phenomenon is under investigation. Clearly, this is situation for the implementation of system life cycle methodologies in complex conditions. Usually, case studies are a one-shot affair rich with descriptive analysis though lacking in generalizability. Multiple case studies as part of the same investigation are much less common and generally not controlled for comparative analysis. We propose an extensible research design based on comparative case studies and intend to systematically collect qualitative data in a way that supports comparative analysis and quantitative examination in producing generalizable findings.[1]

There are four elements to case study research: the mode of analysis, the terms of analysis, the unit of analysis, and the levels of analysis. These are used to establish and synchronize the analysis of cases chosen for comparison.

The mode of our analysis is to develop theoretical accounts of the phenomena under investigation. We use a set of motivating questions as the guiding framework for organizing these accounts. The questions are then answered for each case study. A representative set of questions includes (1) why do people in an organization choose to adopt a particular methodology, (2) how do people actually use the implemented methodology in developing software, (3) what does an analyst need to examine to understand how the implemented methodology is being used, and (4) what are the consequences resulting from the use of the methodology as implemented and practiced?

The terms of our analysis reflect a common perspective and vocabulary for evaluating the efficacy of a software life cycle methodology. These terms direct the collection of case data and organize accounts of what transpired. We initially choose "costs,"

---

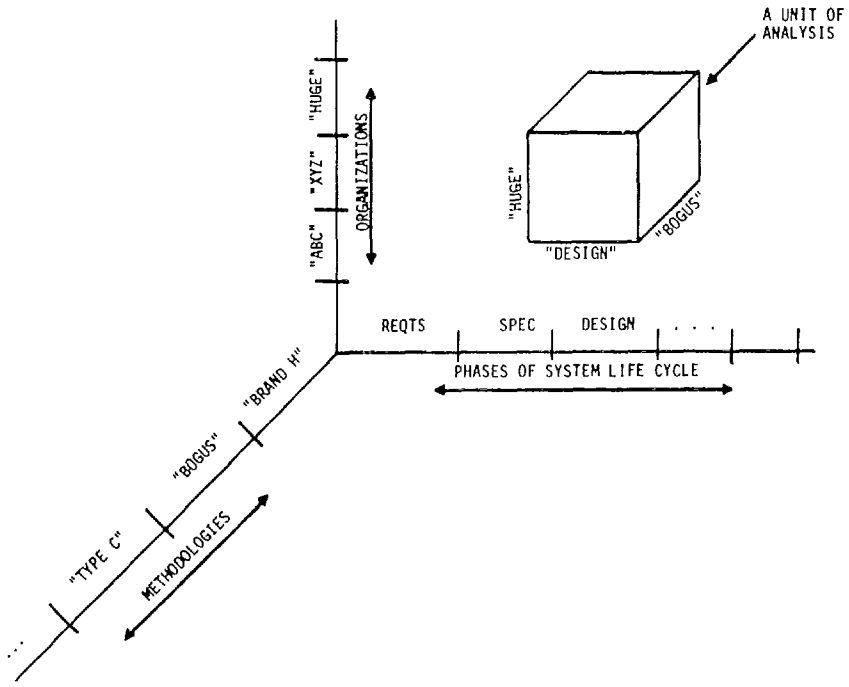
[1] This research design is derived from one used to examine nine cases of innovation in computing. See Walt Scacchi, The Process of Innovation in Computing, Ph.D. dissertation, Dept. of Information and Computer Science, University of California, Irvine, CA (1981).

"organizational impacts," and "benefits" as these terms.[2] We see these terms respectively represent resource commitments or expenditures, shifts in resource allocation or distribution, and new supplies of resources. Clearly, our interest is directed at the ensemble of resources used in implementing and practicing a system life cycle methodology: hardware, software, documentation, personnel, time, money, skills, sentiments, influence, procedures, policies, and so forth. Since the use of most of these resources is interdependent, we are ultimately directing our interest to the infrastructure of computing resources attendant to a methodology, how it fits into the organization, and how resources move from one to the other.

The unit of analysis is the central element in an individual case study. It is the focal item to be examined, analyzed, and contrasted across comparable cases. In studying the implementation of software life cycle methodologies in complex organizations, we focus on two units of analysis: (1) a single phrase or activity in a system life cycle methodology in an organization, and (2) aspects of system quality assurance throughout its life cycle.[3] The unit of analysis also reflects the dimensions of the problem space under investigation. The dimensions we use in this study are "organizations," "phases of the system life cycle," and "methodologies." Figure 3.1-1 portrays the dimensions of our problem space and locates a unit of analysis as a cell within it.

Finally, the levels of analysis characterizes the dimensions across which the unit of analysis is examined. Case studies performed across a single level of analysis are comparable and generalizable only to those at the same level. Yet it is logical to consider comparing case analyses across levels when appropriately chosen. The choice is appropriate when the levels are bound to the unit of analysis, that is, when the unit can be analyzed at each level. We choose four levels of analysis for examining each of our units of analysis: the individual case (a cell), cases within an organization (a row), cases across organizations (a column), and cases within and across organizations (a matrix). Other choices are possible.

Figure 3.1-1



Choosing to examine case studies in this way dovetails with our desire for comparable and generalizable findings. This is achieved in a systematic and straightforward manner. The generalizability of case findings is limited to the level of analysis. Accordingly, findings from an individual case are least general, while all cases simultaneously the most general. What we propose is to work from the mode and terms of analysis for individual cases and progressively move to higher levels. Along the way, we compare our findings for cases at each level then progressively abstract their coverage to account for preceding levels. What results is a set of findings or claims empirically grounded in individual case studies that are compared and generalized in a cross-cutting manner. These findings

---

[2] Other terms oriented to different analytic perspectives of system life cycles are described elsewhere. See Rob Kling and Walt Scacchi, "Computing as Social Action," in M. Yovits (ed.). Advances in Computers, Vol. 19, Academic Press, New York, NY, pp. 249-327, (1980); and Scacchi (ibid).

[3] An example of the first is the "Design Phase" of the CCC development project using "Bogus Design Technique" at the "Huge Corporation." For the second, "Configuration Management" on the CCC Project at Huge Corporation.

are then the final published products of this research.

This constitutes our approach to investigating the implementation of system life cycle methodologies in complex organizations through comparative case studies.

3.2 RATIONALE FOR APPROACH AND REJECTED ALTERNATIVES

When we study the history of computing innovations in complex organizations, it becomes clear that neither the effect on interpersonal relationships nor the impact of particular innovations are understood before they are installed.[4] The problems of implementing an innovation such as a system life cycle methodology are for the most part nontechnical. The major problems are fitting the elements of the methodology to the behavior of participants and embedding them within evolving organizational arrangements. To understand these problems, we require a research method suited to exploring the nature of the methodology, its implementation, the participants behavior, the organizational arrangements, and their interaction.

We propose a comparative qualitative study for our research. Complex organizations operate under the pressures of budge limitations, project deadlines, personnel limitations, and increasingly diverse computing arrangements. Experimental methods that require significant interventions into regular work routines or rigorous experimental controls for reliable quantitative analyses are impractical in such settings. Intuitive or retrospective analyses of such settings are sometimes informing though often ad hoc, noncomparable, and difficult to check. Instead, we propose a careful investigation intended to minimally disrupt the flow of work while observing and collecting data that is focal to the study. Ours is a naturalistic study and our interest is in understanding the flow of work that emerges in settings as they exist. Our study thus stresses learning the patterns of organizational life and technological development whose characteristics are not known or poorly understood when the research begins.

Our qualitative research method is complementary to quantitative techniques. Qualitative and quantitative research methods are not mutually exclusive. We direct our research method in a way where both are appropriate. The qualitative study is appropriate when investigating poorly understood phenomena. It serves as a vehicle by which we can discover and develop theoretical accounts of observed processes. The descriptive reports generated from such a study are assessed for the coverage of observed phenomena, the coherence of the accounts, the examination of alternative accounts which might substantiate or refute the findings, and comparison across the case sample. Survey research methods and quantitative analyses are then appropriate for ascertaining the dimensions of cognitive, organizational, methodological, or technical structures appearing in these processes. Quantitative techniques are also appropriate for measuring the frequency of the states or events that occur as the focal structures are processed. Hence quantitative techniques can be used to test the veracity and degree of relationship among the theoretical accounts (or hypotheses) developed in the qualitative study.

The basis of our research approach is thus to pursue and master the qualitative techniques described in this proposal in preparation for the quantitative analyses which follow.

4.0 Statement of Work

The purpose of SOW is to define the set of tasks to establish the feasibility of using the case study method to evaluate the implementation of a software life cycle methodology. It is anticipated that a similar set of tasks will be appropriate for phase II. The SOW consists of five tasks:

1. Case Study Preparation
2. Case/Site Selection
3. Case Study and Analysis
4. Identification of Future Work
5. Publication of Findings

An explanation of each task follows.

4.1 TASK 1 - CASE STUDY PREPARATION

The first task in applying the case study approach is the preparation activity. This is an important initialization step in performing a case study since it sets the stage and the focus for all the tasks that follow. During the Preparation Task, we define what results/observations are expected from the case study, how the case study will be performed, and what will be the evaluation criteria

---

[4] See Rob Kling and Walt Scacchi, "Computing as Social Action," in M. Yovits (ed.), Advances in Computers, Vol. 19, Academic Press, New York, NY, pp. 249-327 (1980); and Walt Scacchi, The Process of Innovation in Computing, Ph.D. dissertation, Dept. of Information and Computer Science, University of California, Irvine, CA (1981).

and the evaluation processes used for the case study.

The Preparation Task includes the following seven steps:

1. Define Evaluation Criteria
2. Define Mode Framework
3. Define Criteria for Identification of a Sub-standard Software Life Cycle Methodology Implementation
4. Define Analysis Dimensions
5. Define Analysis Terms
6. Preparation Case Study Analysis Plan
7. Characterize Methodology

An explanation of each of these steps follows.

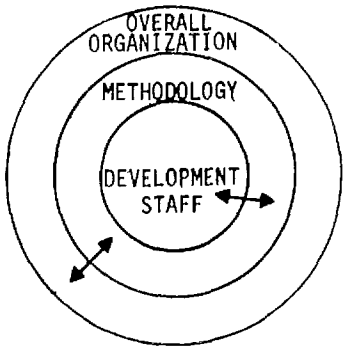
4.1.1 Define Evaluation Criteria

How the effectiveness of the implementation of the software life cycle methodology will be evaluated should be defined before the case study is performed. There are three points to consider when defining evaluation criteria:

1. Fit of the software life cycle methodology.
2. Impact of the software life cycle methodology.
3. Benefits expected from using the software life cycle methodology.

Fit is evaluated in terms of the compatibility of the software life cycle methodology with existing organization (e.g., reporting periods, financial procedures) and with the existing organization infra structure. (See Figure 4.1-1.)

Figure 4.1-1



Some possible impact areas to consider include:

1. Communication among the development staff and with the overall organization.
2. Organizational structure.
3. Software life cycle cost (in particular, development cost).
4. Development staff morale and turn-over rate.
5. Quality of work environment.
6. Visibility of the development project and software system.
7. Management control.
8. Staff skill levels required and hiring practices.
9. Staff training cost.
10. Quality of product.
11. Support services (e.g., software tools, workstations)

Some possible benefits to consider include:

1. Retention/hiring of staff.
2. Visibility with customer.

3. Personnel evaluation.
4. Management control mechanisms.

4.1.2 Define Mode Framework

The case study framework can be defined in terms of questions that should be answered as a result of performing the case study. Examples of such questions are:

1. Why do people in an organization adopt a software life cycle methodology?
2. How is the methodology actually used in practice?
3. What is needed to understand how the methodology is used?
4. What are the expected outcomes of using the methodology (i.e., hypotheses)?

4.1.3 Define Criteria for Identification of a Standard Software Life Cycle Methodology Implementation

As part of determining the validity of case study results/observations, an inadequate or substandard implementation of the software life cycle methodology must be recognizable. Some criteria for identifying this situation are:

1. Inadequate training (i.e., insufficient training time, poor instructors, insufficient training materials).
2. Misapplication of the methodology (e.g., skipping steps).
3. Inappropriate type of application/target system.

4.1.4 Define Analysis Dimensions

The three basic dimensions for analyzing the effectiveness of the implementation of a software life cycle methodology are:

1. Software Life Cycle
2. Organization
3. Software Life Cycle Methodologies

The characteristics of the software life cycle that may be analyzed in the case study include:

1. Definition of phases/subphases in terms of resulting products, required resources, required accepted products from previous activities and development methodologies to be used.

2. Definition of products in terms of confirmation documents, specifications, designs, standards, internal documentation (e.g., data dictionaries), external documentation (e.g., user manuals), experimental results from simulation and prototyping, module/program code, verification and validation data and results, and acceptance criteria.

3. Definition of schedules in terms of time and sequence when products are to be delivered, control points and milestones, schedules of resource requirements and change control methods.

The characteristics of the organization that may be analyzed include:

1. Industry type.
2. Organization size.
3. Project size (in terms of people, deliverables, schedule, time).
4. Resources available to the project.
5. Management style.
6. Prior experience in similar projects and/or with methodology.
7. Project/application type (in terms of geographic separation, matrix approach, parallel hardware development).
8. Customer including procedures and relationships.

4.1.5 Define Analysis Terms

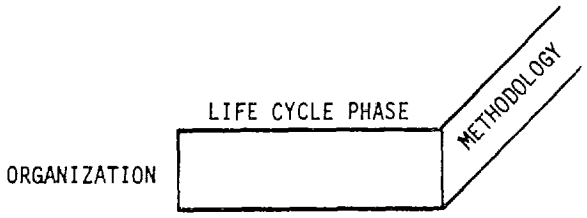
A potential list for terms of analysis using the basic unit of analysis defined in 4.1.4 includes:

1. Resources produced, consumed, and in contention (e.g., skills, software, hardware, policies, influences, time, budget, procedures).

2. Kinds of operational problems encountered and kinds of expected and unexpected successes encountered in using the methodology.

3. External influences of customers/funders, vendors (e.g., change in requirements of product, enhancements to methodology).

Figure 4.1-2



4.1.6 Prepare Case Study Analysis Plan

The Case Study Analysis plan follows from the focus or scope of the case study, that is, the effects, problems, and benefits that are to be observed during the case study. This plan is built around the basic unit of analysis and requires that decisions on each analysis dimension be made. For example, is one methodology to be observed in several different size projects in one organization?

4.1.7 Characterize Methodology

As part of preparation, the characterization of the software life cycle methodology to be studied is defined. Possible terms for characterizing a methodology include:

1. Life cycle phases covered
2. Sub-methodologies included
3. Training requirements
4. Skills needed
5. Types of project/applications appropriate for
6. Benefits to be achieved.

4.2 TASK 2 - CASE/SITE SELECTION

Selection of subjects for study is a critically important aspect to the success of the project. It is important that each of selected projects adhere to the criteria listed below to gain maximum benefit and accuracy from the data that is collected.

The Selection Task includes the following five tasks:

1. Define Selection Criteria
2. Develop Prospectus/Invitation
3. Identify Potential Sites
4. Select Sites
5. Negotiate Site Access and Cost

An explanation of each step follows.

4.2.1 Define Selection Criteria

The analysis plan will define the scope of the case study in the following terms:

1. hypotheses to be studied.
2. number of sites
3. duration of study

These determine the requirements for the selection criteria. There are three points to consider when defining selection criteria:

1. Are any aspects of the object of study (i.e., software life cycle methodology) present at the site?
2. Is the use of the object observable?
3. Is the site willing and able to participate?

Presence is determined by project personnel knowing that the organization has a defined development procedure within a phase structure. The level of documentation could vary from informal project memos to a formal methodology handbook.

Some issues of concern in observing are;

1. Will the subject project progress through the desired phases in the study timeframe?
2. Are the constraints on the subject project such that interference from interviews can be tolerated?
3. Does the site collect and process the desired data?

Some factors in participation include:

1. Formal management support of the study
2. Establishment of clean lines of communication
3. Assigned project personnel to support study with data
4. Availability of support such as secretaries and interview rooms

4.2.2 Develop the Prospectus/Invitation

A prospectus for participating in this case study will be written to provide prospective participants with the details of how the case study will be conducted, and what an organization can expect to gain in utilizing the results of the case study to improve their life cycle models. It will define the resources required in terms of labor resources by type, computing, and facilities. This prospectus will be mailed to all potential sites as identified in the effort described in paragraph 4.2.3.

4.2.3 Identify Potential Sites

Institutions, corporations, and governmental organizations will be canvassed to identify suitable projects in terms of size, duration, start time frame, and other criteria as defined in 4.2.1. Suggestions for candidates will be obtained from the sponsor and other funding agencies. In making original contacts to identify potential projects to use, a reporting framework will be established so that other projects for that follow-on work can be identified without redoing all the front-end work that will be done in this phase of the project.

4.2.4 Select Sites

The list of potential projects will be assessed according to the criteria as specified above and the desired number of projects selected. In addition to the desired number, an additional group will be selected as backup in case the final agreements cannot be reached. The selected sites will be notified at least two months prior to the start of the project such that they can begin the preparations required in terms of informing their people and verifying the resources required to participate in this case study analysis.

4.2.5 Negotiate Site Access and Cost

For the selected sites a letter of agreement will be negotiated and signed by each participant organization and the funding agency. This document will include the schedule of interviews, and a detailed description of required resources. Restrictions and nondisclosure agreements will be specified.

4.3 TASK 3 - CASE STUDY AND ANALYSIS

Two concerns dominate our attention in conducting case studies, data collection and analysis of the data. We are interested in following a systematic procedure for determining what data to collect and how to analyze it. In particular, our goal is to follow a procedure that incorporates built-in cross-checks and controls for data collection and analysis.

The Case Study Task consists of five steps:

1. Collect Case Study Data
2. Code Case Study Data
3. Analyze and Synthesize Case Study Data
4. Prepare Case Report
5. Prepare Composite Case Study Report

An explanation of each step follows.

4.3.1 Collect Case Study Data

There are three forms of data collection in case studies: structured interviews, participant observation, and collection of locally produced artifacts. The principal of these is structured interviews with the participants in the setting. The other two serve as evidence for or against data collected through participant interview. Participant or nonparticipant observation is directed at capturing a glimpse of day-to-day activities from the inside or nearby. Observing how a system designer works with a design

methodology is an example. Locally produced artifacts such as project reports, software documentation, internal performance measures, or administrative memoranda serve as materials which participants reference in their accounts of what is produced and how. Both direct observation and collected artifacts serve to cross-check the data collected during the interviews.

Structured interviews are the main form of data collection. The analyst begins by preparing an initial interview schedule to organize the topics for investigation with the first interviewees. Initially interviews may be loosely structured so as to introduce the analyst to the character of organizational arrangements, workflow, major participants in system development, and so forth. These interviews (the average time is expected to be 40-60 minutes) serve to help the analyst determine which participants to subsequently interview. the choice of data to collect in succeeding interviews depends upon the participant being interviewed, what they do, and the analysis of data previously collected. The interview data is therefore collected according to an interview schedule progressively developed to record who is to be interviewed, when, what topics are to be probed, and why.

Once data collection has begun, the data is analyzed in an iterative and accumulative manner. Data collection and analysis must be procedurally interspersed. Given that we are primarily dealing with qualitative data, we follow three stages for coding data for analysis. The iterative stages are covered in the next three steps.

4.3.2 Code Case Study Data

The interviews are coded according to the interview schedule. That is, the answers provided to interview questions (the data) are recorded either during or immediately after an interview. A fill-in-the-blands questionnaire derived from the interview schedule is a sensible instrument for this purpose.

4.3.3 Analyze and Synthesize Case Study Data

The data is coded and aggregated according to analytic themes (e.g., current hypotheses). These themes emerge interactively from the guiding framework and the data itself. These themes serve to anchor and control the analysis for comparative purposes. Some may change shape as data collection and analysis continue thus implying their iterative development. In addition, the narrative data coded in this form is also amenable to basic quantitative analysis (e.g., frequency of events). The product at this stage an "analytic memo," a narrative theoretical account of the data related to a theme. The interview schedule may be revised to collect supplementary data and insure sufficient analytic coverage. To ensure this coverage, we may bind their scope to the unit of analysis (an individual case).

4.3.4 Prepare Case Report

Finally, after the data collection-analysis procedure has been repeated until saturation, a case report summarizing the experience and the findings accumulated in analytic memos is then prepared. As multiple case studies may be investigated at a site, a case report for each is produced. Data collected and analyzed for one case may directly bear on another. This is both expected and desirable since the nature of the data or phenomena under study is likely to be continuous rather than discrete. Thus it may be appropriate to reference coded data or analytic memos developed for a prior case in the current one.

4.3.5 Prepare Composite Case Study Report

The case report, analytic memos, and coded interviews constitute the data base for an individual case study. In following our approach for comparative multi-level analysis, these case data bases provide the basis for comparing the findings across cases. In this way, we iterate through our comparative analysis across cases at more general levels progressively merging the data bases to produce a more comprehensive final set. These are then the final set of coded data, analytic memos, and case reports which when assembled into a composite report to complete the study.

4.4 TASK 4 - IDENTIFICATION OF FUTURE WORK

Out of the qualitative study of software life cycle methodology implementation will emerge theoretical accounts of the observed process. Then survey research methods and quantitative analyses might be appropriate for determining the dimensions of cognitive, organizational, methodological, or technical structures present in the processes. The purpose of this task is to identify the requirements for further analyses of software life cycle methodology implementation.

The Identification Task includes the following five steps:

1. Determine voids in Existing Life Cycle Implementation Evaluation Technology.
2. Develop Framework for Removing Voids.
3. Define Requirements for Enhanced Life Cycle Implementation Evaluation Technology.
4. Determine Risk and cost for Correcting Voids.
5. Prepare Plan for Next Phase.

An explanation of each step follows.

4.4.1 Determine voids in Existing Life Cycle Implementation Evaluation Technology

This will be accomplished by evaluating the effectiveness of the case study results obtained in Task 3. This will involve comparing the study results against expectations and rationalizing the differences. Sources of differences might include:

1. Inadequate use of existing technology.
2. Application of existing technology to inappropriate situations.
3. Deficiencies in existing technology.

4.4.2 Develop Framework for Removing Voids

It is envisaged that the framework will be a combination of qualitative, survey research, and quantitative techniques. The framework will address the voids identified in 4.4.1. Also the framework will handle the capabilities to analyze the relationships of the theoretical accounts resulting from Task 3. The capabilities will be determined by identifying the both existing and desired quantitative techniques for testing the relationships.

4.4.3 Define Requirements for Enhanced Life Cycle Implementation Evaluation Technology

The requirements will cover needs for changes in methodology for new technique and for technique revisions. The framework from 4.4.2 will provide the structure for defining the requirements. The statement of a requirement will consist of a need statement, development constraint(s) and acceptance criteria.

4.4.4 Determine the Risk and Cost for Correcting the Voids

This will be done by mapping an assessment of State-of-the-Art against the requirements. Then the requirements will be analyzed to determine the points and degree of innovation needed. From this a risk factor will be assigned to the requirements, both individually and collectively. Finally, by requirement using the risk analysis, a cost will be determined.

4.4.5 Prepare Plan for Next Phase

This will result in detailed plan and schedule for conducting a comprehensive qualitative evaluation of several software development life cycle methodology implementations.

4.5 TASK 5 - PUBLICATION OF FINDINGS

The results of this project could seriously impact the productivity of current software development. This potential demands that a deliberate strategy is needed to publish the results.

It is expected that this project will have three classes of results:

1. Description and assessment of utility of evaluation technology.
2. Identification of areas for improvement in life cycle methodology.
3. Description of strengths and weaknesses in the implementation of life cycle methodologies.

The results will be of interest to at least the following four groups.

1. those who participated in the case study
3. Conferences and journals in the areas of: computer science, human 2. users of life cycle methodology
3. creators of life cycle methodology
4. evaluators of life cycle methodology implementation

The Publication Task includes the following four steps:

1. Select Publication Format(s)
2. Identify Publication Sources
3. Prepare Publication(s)
4. Disseminate Publication(s) with Procuring Agency Approval

An explanation of each step follows.

4.5.1 Select Publication Format(s)

The different classes of results and the various audiences will require different publication formats. Some points to consider in choosing formats are:

1. Level of detail to be reported
2. Publication time-frame
3. Durability of publication

4.5.2 Identify Publication Sources

Some considerations in choice of sources would include the use of:

1. Trade newspapers and journals
2. Agency and government publication vehicle (e.g., NTIS)
3. Conferences and journals in the areas of computer science, human factors, and management science.

4.5.3 Prepare Publication(s)

The key here is to produce the results in a timely fashion and assure that the proper agency approvals have been obtained.

4.5.4 Disseminate Publication(s) With Procuring Agency Approval

5.0 Plans

The project is proposed to be done in four phases. Phase I will be a pilot study to validate the use of case study approach on the problem of life cycle evaluation. Phase II will be a broad application of the case study approach to produce generalizable results and a basis for the investigation of a refined analysis of life cycle methodology implementation. Further phases may be needed if the refined analysis looks promising.

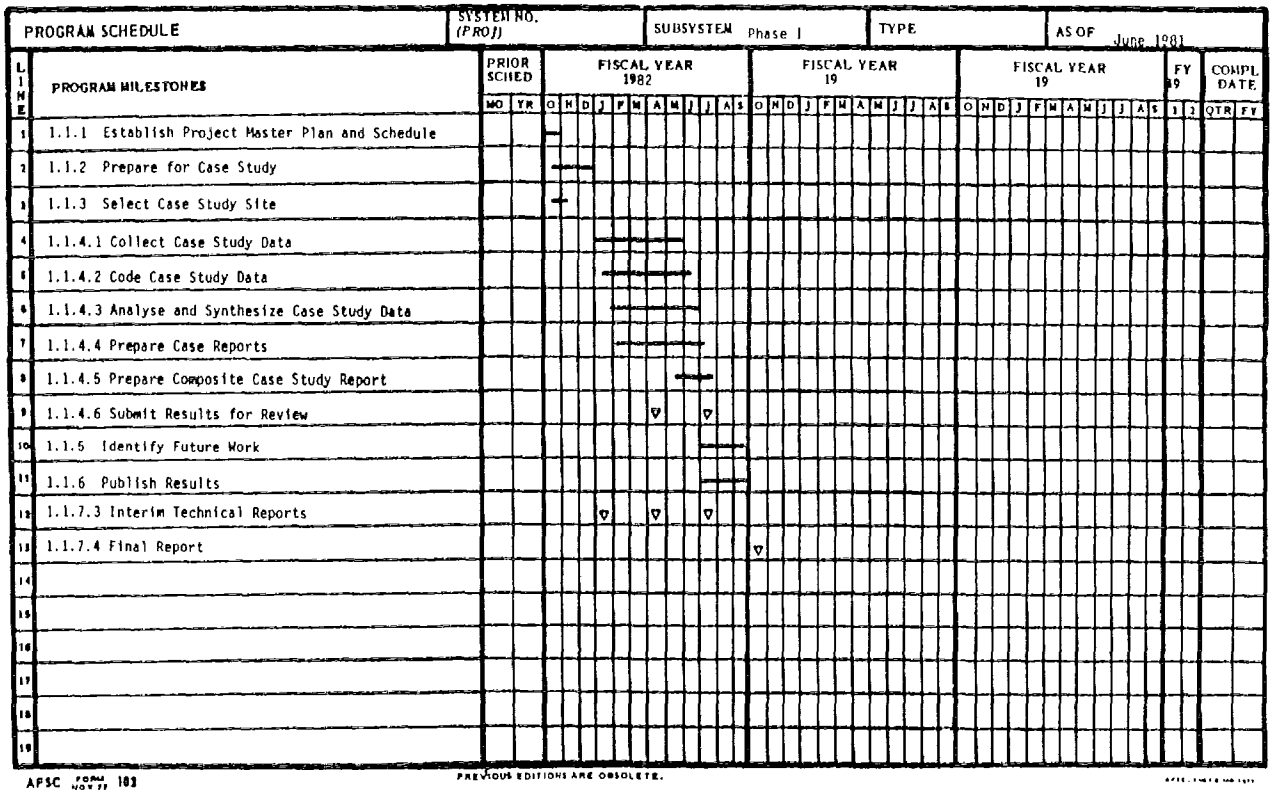
5.1 CWBS (CONTRACT WORK BREAKDOWN STRUCTURE)

```
Element Number      Element Title

1.0  Evaluation of Software Development Life Cycle Methodology Implementation
     1.1  Phase I:  Establish Case Study Analysis Approach
          1.1.1     Establish Project Master Plan and Schedule
                    1.1.1.1  Establish Project Master Plan
                    1.1.1.2  Establish Project Master Schedule
          1.1.2     Prepare for Case Study
                    1.1.2.1  Define Evaluation Criteria
                    1.1.2.2  Define Mode Framework
                    1.1.2.3  Define Criteria for Identification of Standard
                             Life Cycle Methodology Implementation
                    1.1.2.4  Define Analysis Dimensions
                    1.1.2.5  Define Analysis Terms
                    1.1.2.6  Prepare Case Study Analysis Plan
                    1.1.2.7  Characterize Subject Methodology
                    1.1.2.8  Submit Results for Review
          1.1.3     Select Case Study Site
                    1.1.3.1  Define Selection Criteria
                    1.1.3.2  Develop Prospectus/Invitation
                    1.1.3.3  Identify Potential Sites
                    1.1.3.4  Select Sites
                    1.1.3.5  Negotiate Site Access and Cost
                    1.1.3.6  Submit Results for Review
          1.1.4     Perform Case Study and Analysis
                    1.1.4.1  Collect Case Study Data
                    1.1.4.2  Code Case Study Data
                    1.1.4.3  Analyze and Synthesize Case Study Data
                    1.1.4.4  Prepare Case Report
                    1.1.4.5  Prepare Composite Case Study Report
                    1.1.4.6  Submit Results for Review
          1.1.5     Identify Future Work
                    1.1.5.1  Determine voids in Existing Life Cycle
                             Implementation Evaluation Technology
                    1.1.5.2  Develop Framework for Removing Voids
```

```
            1.1.5.3  Define Requirements for Enhanced Evaluation
                     Technology
            1.1.5.4  Determine Risk and Cost for Correcting Voids
            1.1.5.5  Prepare Phase II Plans
      1.1.6 Publish Findings
            1.1.6.1  Select Publication Format(s)
            1.1.6.2  Identify Publication Sources
            1.1.6.3  Prepare Publication(s)
            1.1.6.4  Obtain Agency Review of Publication(s)
      1.1.7 Data
            1.1.7.1  Contract Work Breakdown Schedule
            1.1.7.2  Program Schedule
            1.1.7.3  Interim Technical Reports
            1.1.7.4  Final Report
```

Figure 5.2-1



5.3 RESOURCES

The major resources required for Phase I subtasks appear in this section.

5.3.1 Labor Resource Requirements

CWBS Activity

1.1.1 Establish Project Master Plan and Schedule	80		20
1.1.2 Prepare for Case Study	200	100	40

1.1.3 Select Case Study Site	80		40
1.1.4 Perform Case Study and Analysis	600	250	80
1.1.5 Identify Future Work	100	50	50
1.1.6 Publish Finding	100	50	50
1.1.7 Data	40	50	100
Phase I Total	1200	500	380

5.3.2 Travel Resource Requirements

2 500 mile 3-day trips
2 1500 mile 1-day trips

5.3.3 Support and Machine Requirement

150 hours of word processor time.
$3,000 of computer resources to support database.

EVALUATION OF DESIGN METHODOLOGIES

A Proposal:
Makoto Arisawa - Yamanashi University
G. David Bergland - Bell Laboratories
Earle C. Bowers - Structured Methods, Inc.
John N. Buxton - University of Warwick
Robert A. Kelley - Honeywell, Inc.
Norman L. Kerth - Tektronix Laboratories
Sabina H. Saib - General Research Corp.
Peter D. Ting - Bell Laboratories
Douglas A. Troy - Bell Laboratories (editor)
Stuart H. Zweben - Ohio State University

Abstract

This proposal seeks to evaluate the effects of various design methodologies on the development and maintenance of computer systems. More specifically, an experiment is proposed to test the hypothesis that the cost of making changes to computer systems is influenced by the application area of the system and by the design methodology employed in its development. Proponents of various design methodologies will develop, from user requirements, solutions to selected problems in each of four application areas. After acceptance of the solutions by an independent contractor, requests for changes will be given to each design methodology group. Careful records of the costs of initial development and subsequent modifications will be kept. These costs, other measures taken on the resulting products, and a post activity conference will form the basis for evaluation the experiment and the participating methodologies.

Table of Contents