National Research Council Canada
Conseil national de recherches Canada

Institute for Information Technology
Institut de technologie de l'information

NRC · CNRC

Software Cost Estimation and Control

M.R. Vigder and A.W. Kark
Software Engineering
February 1994

Canada

NRC No. 37116

Additional copies are available free of charge from:

Publication Office
Institute for Information Technology
National Research Council of Canada
Ottawa, Ontario, Canada
K1A 0R6

Des exemplaires supplémentaires peuvent être obtenus gratuitement à l'addresse suivante:

Bureau des publications
Institut de technologie de l'information
Conseil national de recherches du Canada
Ottawa (Ontario) Canada
K1A 0R6

Table of Contents

0. Executive Summary

The Software Engineering Laboratory (SEL) of the Institute for Information Technology of the National Research Council was asked by the Chief Research and Development (CRAD) of the Department of National Defence (DND) (contract number 220792NRC08) to perform a study of software cost estimation methodologies used within defence and industry, recommend improvements in the process and, if necessary, identify research directions in the area of software cost estimation.

The initial approach taken by SEL was to concentrate on a few projects within the DND, both from the procuring agency and supplier point of view. All aspects of the projects were to be studied. These included original estimates and methodology used to arrive at the software cost estimates, changes in the project as defined by contract amendments, and comparison of the result with the original estimates. On the basis of the data gathered, SEL was to evaluate methodologies within the domain of specific projects' types. As part of the data gathering, the comparison between cost estimates for hardware- and software-oriented projects was to be made.

The selected approach was very quickly proven not to be viable, as the required data were simply not available. The focus of the study has therefore changed. We decided to look at the process of procuring and developing systems -- but with software cost estimation in mind. We significantly increased the number of the DND and industry interviews we carried out. As a result of these interviews we were able to determine the definite patterns in which projects are procured, costed, and executed. These patterns show that

- Software cost estimation is done too early in the procurement process and is based on -- usually -- wrong specifications;
- Once established, the estimates are very difficult to change;
- Parametric models for software cost estimation are rarely used within either the military or the industry developing systems for the military;
- Business-oriented software producers are more likely to use some parametric models;
- There are no historical data on which to base the estimates for the new software projects;
- Very often the price for the software development for the DND does not reflect the actual estimates for the software as a function of the requirements, but is a function of many others non-technical factors;

This state of disarray is not unique to the DND and DND contractors. From the studies conducted by the Software Engineering Institute (SEI) - associated with the Carnegie Mellon University it is clear that, at the present time only a small percentage of software developers are able to effectively handle the full

software life cycle from the requirements through the development to the support.

Clearly, many of these problems can be corrected through "low-tech, common-sense" actions. Only after the basic process problems are resolved, can we start talking about sophisticated methodologies aimed at substantially improving the estimation accuracy.

The recommendations outlined in Chapter 5 are divided into two parts: those, that we believe are applicable to the software development industry in general and those that are DND specific. The general recommendations are based on the assumptions that- there is in fact room for improvements in the accuracy of the cost estimates and despite the possibility of improvements there will continue to be a large margin for error in these estimates.

Recommendations that can be applied within the software development industry include:

- Improvements in the software development process by:
    - Formalizing when and how cost estimates are performed. This should include definitions of a company-customized methodology, a process by which the methodology is to be applied, when the estimates and re-estimates are performed, and what is being estimated for a project.
    - Permitting effective monitoring and control of the software costs. If there are no effective measurements during the development process the accuracy of the estimates cannot be established. But the measurement process cannot be a burden to the project personnel, must be performed to a proper level of detail, and must contain an objective measure of completeness.
    - Analyzing problems reported during the development process. This applies to the process itself and is used to improve future performance of the organization.
    - Effective control over functional requirements. The most practical way of achieving this goal is to introduce effective change control process, as it is very difficult if not impossible to produce perfect requirements specification. Parties involved in the development process on both the customer and developer sides must realize and accept that the requirements will change during the development.
- Collecting and analyzing the historical data for the projects. This is a vital step towards improvement of the accuracy of software development costs. The collected data should include information on both products (projects) and process by which these products were built. These metrics must be customized for each organization;

however, one can generalize the areas necessary for cost estimation improvements:

- Basic characteristics of the development process in order to understand the context in which the system was developed;
- All estimates and re-estimates for the project.
- Actual costs of the final system. This metric must clearly indicate what was included.
- Characteristics of the completed product. This must include size and complexity metrics, description of the final product, classification of software, and other information that can be useful in developing cost estimation models for a given organization.
- The data must be collected in a form that is easy to access and analyze and in enough detail to be useful to the future estimation process.

• Changes in the project management procedures. Unless proper management procedures are in place within an organization, then cost estimates, no matter how accurate, are irrelevant. At any point in time, project managers must be aware of the state of the project. This includes incurred costs, level of completion, and estimates of effort required to complete the project.

DND is involved in software development in two different roles: software procurer and software developer and maintainer. Where DND is involved in software development and maintenance the same recommendations as for the industry will apply. There are, however, additional recommendations that are DND specific:

• From a Procurement Agency perspective, DND must recognize the problems existing with the preparation of the requirements specification. In fact, it must be recognized that the Procurement Process itself causes unrealistic or inaccurate estimates. The possible approaches to a solution of this problem include

- Different procurement processes for systems that are relatively standard versus those which are innovative.
- The procurement process must have provision for modifying the requirements and the corresponding cost estimate in order to arrive at the correct and final requirements as early in the development process as possible.

- Improve the relations between the procuring agencies and the contractors. To that end, "Common Purpose Procurement" should be investigated and experimented with.

- Effective monitoring of the project should be based on the assumption that the contractor is a technically competent and therefore the reviews should focus on

    - Functionality.

    - Process.

Large software systems have a long life span. It has been shown that the majority of the costs of the system are incurred after the initial development has been completed. Given that fact, only a few organizations make conscious tradeoffs between development and maintenance costs. The DND should therefore

- Investigate and record the current proportion of maintenance to development costs.

- Introduce full life-cycle costing for software products to permit tradeoff of the development costs for reduced maintenance costs.

- Sponsor research into re-engineering and reverse engineering for software. Many current systems are delivered without proper documentation and are given to the maintenance centers without the necessary knowledge. This increases the cost of maintenance and reduces the quality of work performed.

One key observation is that there is a difficulty within DND in building up a core knowledge of software expertise. We believe that DND must strive towards building such an expertise and to that effect it is recommended that

- DND investigate the development of a group with the mandate of researching, developing, retaining, and disseminating software expertise throughout DND. To ensure the effectiveness of such an organization it must have a stable work force not affected by the customary military rotation process. Such an organization would

    - Assist project managers in various areas of the software development process.

    - Have the mandate and the power to collect, analyze, and disseminate data on software procurement, development, and maintenance throughout the DND.

    - Monitor and evaluate contractors.

    - Perform research in the area of the development and procurement processes.

- Participate in the definition of procurement standards.
- Promote transfer of knowledge between various organizations within DND.

- To initiate and facilitate the creation of such an organization it is recommended that a pilot project be started. Such a project would initially focus on data collection and transfer of knowledge.

Clearly, implementation of any of the above recommendation will contribute to the improvement of the predictability and accuracy of the software cost estimates.

1. Introduction

This report summarizes a study which the National Research Council (NRC) of Canada performed for the Chief Research and Development of the Department of National Defence (DND). The study focused on issues and research directions relating to cost estimation of systems that contain significant amounts of software. The primary long term goal of the project is to develop improved methodologies for forecasting and controlling software development costs. The immediate objectives of this study were to

- Determine the current "state of the art" in use in Canada for estimating the costs of military and civilian software systems.
- Analyze the (in)accuracy of software cost estimates, e.g., how inaccurate are they and why are they inaccurate.
- Identify factors affecting the cost of software systems.
- Identify steps for improving software cost estimating practices within DND, based on current technology.
- Identify research directions for estimating and controlling software costs.

In Chapter 1 of this report we will define the terms used in the report and describe the general approach and methodology used during the project. Chapter 2 describes the data gathering phase of the project. Chapter 3 talks about current cost estimation methodologies in general, as well as issues specifically pertaining to the study.

Chapter 4 contains our observations regarding the problems with the cost estimations and software development process as collected during our study. This chapter describes differences in problems faced and approaches used between the development and maintenance organizations. Chapter 5 summarizes our findings and makes recommendations.

The project can be divided into three distinct phases.

First, we familiarized ourselves with the organization of DND and clearly formulated the objectives of our study. As part of this phase of the project we analyzed data available to us through the Industry Research Assistance Program (IRAP) on 21 completed projects. This analysis was used not only to define the types of data we expected to collect during the subsequent phases of the project, but also to provide a comparison between cost estimation models and effectiveness of project management techniques of research projects in the hardware and software domains. The results of this study are briefly described in [Vigd-94].

The *data gathering* phase of the project involved preparing the questionnaire and visiting a number of organizations involved in software development projects, either as developers or procurers. The purpose of these visits was to

obtain information about software costs from a representative cross section of Canadian organizations involved in software development. These interviews provided us with information regarding the current state of the art of software cost estimation and control, information on cost drivers for software development, and indications of whether there is an actual problem in accurately estimating software costs. The questionnaire and the interview process are described in Chapter 2.

The *data analysis* phase of the project put into context the information gathered from the visited organizations both in the DND and industry. The analysis suggested solutions to the problems and pointed to possible further research.

1.1 Definition of Terms

Despite the terminology, *software cost* does not refer directly to the dollar figure associated with software development. Such a figure, as we have discovered, is almost impossible to arrive at and not always useful. The questions people are more interested in are "*What's the effort involved?*" and "*How long will it take?*" The answers to these two questions can then be translated to the dollar figure. This leads to the following definition of software cost.

Software Cost

Software cost consists of the following three elements:

- *Manpower loading* is the number of engineering and management personnel allocated to the project as a function of time.
- *Effort* is defined as the engineering and management effort required to complete a project, usually measured in units such as person-months. The types and the levels of skills for the resources will come into play here.
- *Duration* is the amount of time (usually measured in months) required to complete the project.

Webster's (Ninth New Collegiate Dictionary, 1985) defines a "cost estimate" as "...to arrive at an often accurate but usually only approximate statement of the cost of a job to be done." Arriving at a cost estimate involves using a number of different factors to try to determine the overall cost of a system. Deciding which factors to include and combining them to arrive at the estimate make up the *software cost estimation process* that is defined as follows:

Software Cost Estimation Process

A *software cost estimation process* is the set of techniques and procedures that an organization uses to arrive at a software cost estimate. Generally there is a set of inputs to the process (e.g., system requirements) and an output of effort, manpower loading, and/or duration.

We discovered very early in our study, that it is very difficult to examine the software cost estimation process without the overall context of the software development process in use within a given organization.

Software Process

The set of procedures, techniques, and standards that an organization uses for organizing, managing, and controlling software development projects is called the *software process*.

Organizations have different software processes, depending on the type of software they are developing. For many organizations, the development process is very informal; in other cases it is well documented and stringently monitored.

1.2 Scope of Work

In identifying software development work to include in this study, we purposely chose a very broad cross section of classes of software and classes of organizations. Our objective was to identify common problems and trends within organizations involved in the development of software. The types of software development projects studied included

- Management Information Systems (MIS);
- Embedded systems;
- One of a kind systems and mass produced systems;
- New development work;
- Maintenance of legacy systems;
- Procurement agencies that contract out the development of software.

We were interested specifically in *software* development as opposed to *system* development. When looking at system development projects, we isolated and focused on the software component of the system, at how the software was estimated and controlled. The fact that the software was embedded within a larger system being developed affected the constraints on the software cost estimation and software development processes, and these are discussed further in Section 3, but the focus of the research was limited to the software itself.

Items included in "software costs" vary considerably from organization to organization. Direct costs include items such as analysis, design, coding, testing and integration. Depending on who is doing the development and why, software cost may also include a number of other items such as training, customer support, installation, level of documentation, configuration management, and quality assurance. As part of the study we looked at what

organizations included within their definition of “cost estimate” and how this affected their estimation technique.

1.3 Cost Estimates and the Software Process

The software process in use by an organization defines *why*, *when*, and *how* cost estimates are made.

There are a number of reasons why an estimate is being done:

- *Planning and budgeting*. Senior managers make strategic decisions based on the accuracy of estimates. Such decisions include whether to proceed with a project, how much to bid, etc.
- *Project management*. Project managers use estimates to plan, monitor, and control the implementation of a project.
- *Communication among project members*. A cost estimate frequently includes a complete work breakdown structure (WBS), which project team members use as a basis for understanding their roles within the team.

The reason for performing an estimate will dictate the outputs and the accuracy required of the cost estimate process. For example, for management to make a decision of whether to proceed with a feasibility study, all that may be required is a rough order of magnitude effort estimate. Project planning requires a much more detailed (and hopefully accurate) estimate.

Estimates and re-estimates are performed throughout the software development process; the software process defines when these estimates are to be done. Re-estimations of a job serve a number of purposes. First, since an estimate is “often accurate but usually only approximate”, re-estimates are done to improve accuracy once more information regarding the job is known. The new information that is used in the re-estimate can be, for example, the overall Work Breakdown Structure (WBS) or the progress of the project during implementation. Second, re-estimates are done because outputs different from the ones generated in previous cost estimates are needed. For example, an initial estimate may give a level of effort for the entire project; a re-estimate can be used to provide levels of effort for the individual tasks of within the overall project.

2. Data Gathering

The longest phase of the software cost estimation study was collecting data from different organizations involved in software development. The objectives of this phase were to determine the current state of the art of software cost estimation across a broad spectrum of organizations and to identify any software cost estimation problems common to a large number of organizations.

2.1 Initial approach to the study

Our initial expectation was to proceed as follows. First, we wished to identify a set of development projects that had recently been completed. To achieve this, we met with a number of key DND personnel. These initial meetings were used, not only to get the list of projects that we should be looking at, but also to familiarize ourselves with the procurement process used within DND as described in An Introduction to the Defence Program Management System (C Prog 500). Those meetings were used to help us to design our questionnaire.

Having prepared the questionnaire and collected the names of the projects within DND, we approached the organizations involved in these projects. We also selected a number of companies within the software industry that we believed would provide us with an insight into non-military software project cost estimation. We suspected, and were subsequently proven right, that the motivation for the estimates and the key cost drivers would be quite different in that environment.

2.2 The questionnaire

The questionnaire (Appendix A) was designed to cover not only software cost estimation but also most of the software development life cycle. This proved to be very valuable during the data gathering process. We found quite quickly that most of the organizations were not prepared to be interviewed unless and until we ensured them of full anonymity and confidentiality. The questionnaire has an introduction and eight sections. The introduction to the questionnaire (which was expanded from the original short paragraph) reflects our attempt to deal with this problem. Only one company approached by us did not agree to participate in the survey. The content of the questionnaire reflected our belief that we would be able to gather the hard data in various areas, which we would use to perform statistical analysis of the information and draw our conclusions. It was clear to us, however, that the numbers themselves would not be meaningful. We therefore asked many "environment questions" in order to put the numbers in a proper context and be able to normalize them in a reasonable fashion.

Section 1 of the questionnaire refers to the project properties. We needed to know where the project was being executed (procuring agency, development organization), the type of system that was being developed (weapon system, Command and Control, MIS ...), and whether it was primarily a software project or if the software part of a larger system. We also needed to know the size of the project in terms of the organization’s perception of the size.

Section 2 of the questionnaire was intended for the organizations contracting (procuring) software. Here, we could not expect to ask questions regarding overall software development cycle, but only look at the "end points." We asked about the difference between expected bids and contracted price, perceived reasons for the difference between actual and estimated costs and who ended up paying the difference.

Section 3 was intended to determine what types of estimates were performed during the project and what were the goals of those estimates (obtain budgetary approval, man-power loading). We wanted to know in what phase of project life cycle the estimates were done, what was estimated (duration, man-power loading, or effort), and who was the customer for those estimates. For completed products we wanted to know the accuracy of these estimates (i.e., the comparison between estimates and actuals). This section also asked questions about items included in the estimates (software, hardware, testing, training, etc.).

The people doing the estimates are an important factor in the accuracy of the estimates. Section 4 of the questionnaire dealt with that aspect of the estimation process. We inquired about experience level of people involved, their positions and status within the organization, and the level of training of those people.

Section 5 of the questionnaire dealt with the estimation process itself. We wanted to know the process used to arrive at the estimates, what were the inputs, to the process and what was the quality of these inputs (in terms of completeness, architectural design, details of User Interface, and so on). We were also interested in whether and how parametric models and tools were used in the estimation process.

Sections 6 and 7 dealt with project tracking and metrics. We assumed that most of the organizations would track the progress of the project in an orderly fashion and that there would exist a set of metrics for both the product produced by the organization and the process by which that product is produced. We wanted to know what these metrics were and how the knowledge gained during a project is transferred and applied to the subsequent projects in which an organization is involved.

The last section of the questionnaire was the basis for an open discussion of software cost drivers. We wanted to gather both DND and the industry perception of the main problems of software cost estimation.

As can be seen, the format of the questionnaire assumed that we were seeking information on a specific project. It was thus “vertical” in nature in that it assumed we would get detailed information on a specific project. We could then use these data as a basis for determining the current state of the art in software cost estimation.

2.3 Interviews

Using the questionnaire as a basis, we gathered data from the different organizations involved in software developing. The data gathering focused on historical data relevant to cost estimation.

Our initial approach was to make contact with an organization, explain our study, and ask it was willing to participate. With only a few exceptions, all groups we contacted felt that participating in such a study would be useful. We then sent the questionnaire to the contact person within the organization and followed up with a meeting at the company premises. We initially expected that the company would prepare written answers, which would serve as an anchor for the discussions. As it turned out, in most cases we did not receive written responses to the questionnaire (there were only two exceptions). We therefore used the questionnaire as a basis for our discussions with the organization. These discussions would generally last about half a day, although with some large organizations we spent up to two days interviewing various people.

We immediately ran into a major problem with this approach: lack of available data. Despite our numerous interviews, there were only a few organizations that could claim that they owned any kind of the historical project data in a form that we could collect. There were primarily two reasons for the lack of hard data:

- The data were not recorded or were not recorded in a readily available form.
- Private companies with "hard" information, did not wish to release it.

By far the main reason for not being able to collect the data was the fact that the data were not recorded. Most of the organizations to whom we talked recorded only a very minimal amount of data about a project. Even some of the most basic information that we required for any reasonable study on cost estimation, such as “what was the original estimate” and “what was the actual cost”, was not always known. In some of the organizations the data that were recorded were often kept in formats not suitable for our purposes, such as paper files of time sheets collected for accounting purposes. These time sheets seldom made a distinction as to the specific task being done by the individual. Only a few organizations provided us with detailed historical data on projects, others claimed to have such information but did not provide it for proprietary reasons. However, these organizations were few and far between.

In most cases we had to depend on peoples' memories as to what happened during a project. In DND, which has regular rotations of personnel, there was often no one left to interview about a project, and the corporate memory of the project had essentially disappeared.

Because there were so few project data available, it became evident to us very quickly that the focus of our study would have to change quite dramatically. Thus, rather than gathering hard data on individual projects and using these as the basis for our analysis, we ended up gathering “soft” data about the organization itself and about different projects and processes within the

organization. As mentioned previously, the structure of the questionnaire allowed us to make such a switch without major delays in our study. The fact that hard data were not available (either existent or not analyzable) was one of the most crucial findings of the study.

2.4 Organizations Interviewed

During the course of this work, we talked to 37 organizations and individuals involved in the development or procurement of software. To get a broad understanding of the issue, we attempted to interview organizations involved in the following aspects of software development:

- *Procurement*. Procurement agencies do not develop software directly. Rather, they identify a need for a system requiring a significant amount of software development and then contract out the development of the software. Their interest in estimating is to "ball-park" the expected contract values when going to tenders and to "sanity-check" the received contract.

- *Contract.* These organizations estimate software costs to establish the price to bid on a project and to control software costs during the duration of the project.

- *Commercial software development.* These organizations perform software cost estimates primarily for estimating time to completion, as "time to market" is their most critical issue.

- *Maintenance.* Maintenance organizations are involved in the upgrading of large existing systems. Their interest in estimating is to determine timing and content of maintenance releases.

3. The Estimation Process - Current State of the Art

How do organizations concerned about software development actually perform their estimates? When are the estimates done? Why are they done? How are they done? To improve the accuracy of software estimation, it is necessary to provide answers to these questions.

3.1 Modeling the Cost Estimation Process

Organizations rely on a wide array of techniques and approaches to arrive at a cost estimate. To gain a better understanding of the different techniques and to contrast and compare different approaches, it is necessary to develop a “model” of the cost estimation process. It is useful to view the estimation process model from three perspectives:

- *Estimation and the software process*. What is the role of estimation in the software process?
- *Inputs and outputs to the software process*. What data are available to an estimator doing the estimate and what information is generated by the cost estimation process?
- *The process itself*. How does the estimator use the information available to generate the actual estimate.

3.1.1 Estimation and the software process

Cost of a project can be estimated for a number of reasons. Why it is done is an important factor in determining when and how it is done. The reasons why a cost estimation process is undertaken include the following:

- *Project approval*. For every project there must be a decision by the organization to undertake the project. Such a decision requires an estimate of the money and resources required to complete it.
- *Project management*. Project managers are responsible for planning and control of projects. Both activities require an estimate of the activities required to complete a project and the resources required for each activity.
- *Project team understanding*. For members of a project team to work together more efficiently on a project, it is necessary that each one understand his/her role in the project and the overall activities of the project. A project task definition, which can be used for this purpose, is generated by a cost estimate.

The "why" of the cost estimation process can be any of the above reasons and is one of the factors determining when the estimate is done. Project approval requires estimates to be performed very early in the project life cycle, often before requirements have been clearly specified. The project approval process

typically has a number of points where a “go/no go” decision must be made. At each of these points, an estimate may be required to permit management to make the decision. Early in the project life cycle, these may be rough order of magnitude estimates sufficient to allow the organization to determine whether they should continue to look at a project. Late in the project, management can get much more detailed estimates of cost to completion in order to decide whether to cancel an ongoing project.

For managing and understanding a project, an estimate must be done early in the development of the project to arrive at an initial estimate, and then repeated on a regular basis during development to keep the estimate current. For these estimates the prime concern is not necessarily the absolute “cost," but the estimated set of tasks required to complete the project, the results of each of these tasks, how these tasks fit together, and the resources required to complete each task.

Re-estimates are required throughout the development cycle regardless of why the estimate is done. As a project progresses, more information is available on the product and the process being used to develop it. This information can be used to increase the accuracy and detail of the estimate.

3.1.2 Inputs and Outputs to the Estimation Process

The software cost estimation process computes a set of outputs as a function of a set of inputs. The inputs to the estimation process depend on when the estimate is being performed. Very early estimates are necessarily based on sparse and incomplete data regarding the project and the development process. Preliminary estimates are needed before requirements are known or architecture has been defined. Such estimates will necessarily be based on sketchy data and will not have a high degree of accuracy.

Estimates performed late in the development cycle are based on a much wider set of information. Computing cost to completion late in the development cycle allows a great deal of project and process information to be used. Given that more information is available, more detailed estimates can be made, which have a much greater degree of accuracy than the initial estimates.

Most models of cost estimation view the estimation process as being a function computed from a set of cost drivers. These drivers are assumed to be the characteristics of a system that determine the final cost of production. In most of the advocated cost estimation techniques, the primary cost driver is assumed to be the software requirements [Boehm, Albrecht, De Marco]. In this model of software cost estimation (illustrated in Figure 1), the requirements are the primary input to the process and form the basis for the estimate. The estimate is then adjusted according to a number of other cost drivers (such as experience of personnel and complexity of system) to arrive at the final estimate.

In this classical view, the effort, duration, and loading are computed as fixed numbers (perhaps with tolerances), or a set of relationships between the values is given, allowing managers to trade off costs in order to minimize any of the three values.

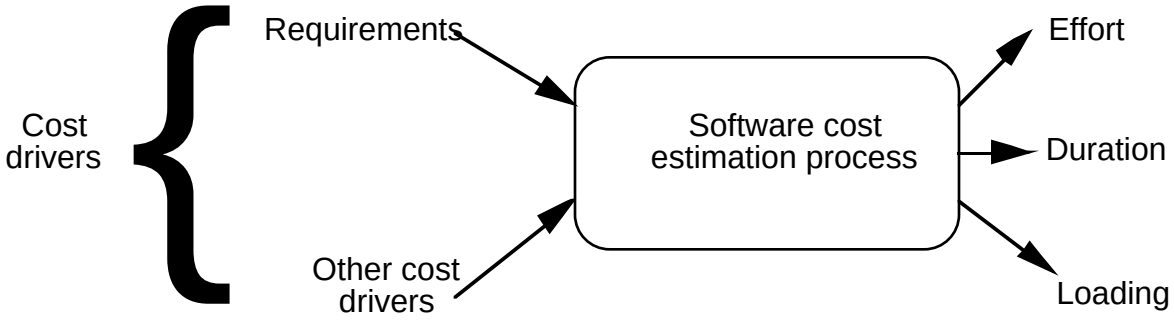

Figure 1. Classical view of software estimation process.

In fact, the cost estimation process can be much more complex than that portrayed in Figure 1. There is an interdependency between many items of information, all of which are relevant to the cost estimation process (Figure 2). Many of the data items that are inputs to the cost estimation process are modified and output by the process. Thus, rather than viewing the cost estimation process as a function of the requirements, it is often more accurate to view this process as trying to satisfy a set of constraints. The inputs to the system are a set of constraints on the requirements, software architecture, financial resources, etc., while the outputs are a cost estimate and a set of assumptions that satisfy all the constraints.

This view allows the constraints to be imposed on any of the factors that affect the cost. These factors range far beyond requirements to include issues such as delivery date, finances and software process.

Requirements are viewed as constraints that must be satisfied. In a few cases, these requirements are fixed, complete, and correct. In most cases, however, during estimation the estimator detects inconsistencies and ambiguities in the requirements. As part of the estimation process, the estimator will resolve some of these ambiguities by imposing new constraints on the requirements. In other cases, the problems with the requirements remain, with a corresponding affect on the accuracy of the estimate.

Financial, calendar, manpower, architectural, and software process constraints are also significant to the cost estimation process.

Financial, calendar, and manpower constraints limit the amount of resources that can be allocated to a project. Financial constraints limit the amount of money that can be budgeted for the project; calendar constraints specify a delivery date that must be met; and manpower constraints limit the number of people that can be allocated to the project. For example, if a fixed amount of money is available for a project, then the estimated cost should satisfy this financial constraint, perhaps by varying the functionality.

The software architecture defines the different components used to construct the system and the interrelationships between these components. The stage in the development life cycle determines whether the software architecture is a factor

for the estimation process. For example, maintenance organizations that are working with an existing system are constrained to use the existing architecture and can base their estimates on this architecture. The cost estimation process for new development may not make any assumptions on the software architecture and base the estimate entirely on the basis of system functionality.

For many larger contracts, the software process becomes one of the constraints that must be satisfied by the estimating process. Many organizations have within their software process a standard Work Breakdown Structure (WBS), which defines the tasks to be performed to complete a project. Frequently, the estimating process will be working under the constraint that the standard WBS must be used for a project. The estimating process will then tailor the WBS to the specific project, adding sufficient detail.

For example, one situation where constraints to the software process affect the estimation process is the requirement to develop according to the 2167a standard. Significant cost is incurred by adhering to this standard; for small changes, 2167a can actually be the dominant cost factor. When estimating a system developed to this standard, estimators must be aware of the cost incurred by use of the standard.

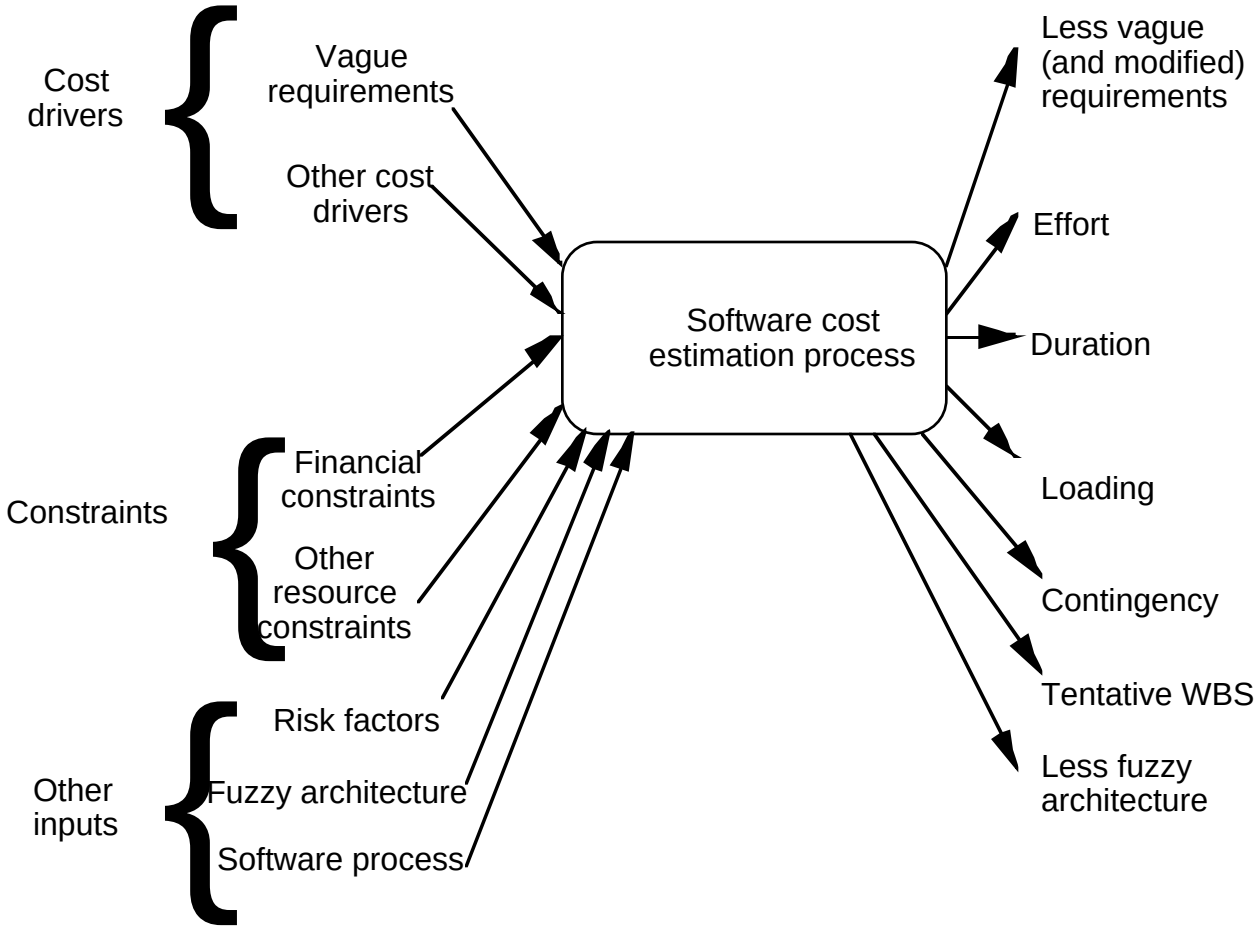

Figure 2. Actual cost estimation process.

Aside from the various constraints, other factors that must be included as part of the estimation process are the risks associated with the project. These risks could include, for example, dependency on outside contractors, integration of new CASE tools into the development process, lack of experience in the application domain, etc. These risk factors should be identified as early as possible in order include them in the decision making and project management processes.

The effect of risk on the cost estimate affects the tolerances of the estimate and the allocation of contingency funds. An estimate for a high risk project will have a very high tolerance, for example, with an accuracy expected of ±100%. Management may then decide to include a very high contingency fund for the project.

3.1.3 The Estimation Process

An estimate is arrived at by taking the identified constraints, applying the estimation process, and generating results that satisfy all the constraints. A variety of techniques are used by different organizations to arrive at these estimates. The processes used can be classified as either **model based** or **analogy based**.

Model-based estimation builds a costing model of system development based on the characteristics of the system being built, the process being used to build it, and its the development environment.

A model can be a formal mathematical model or a set of informal guidelines used by an estimator. Informal models are used by experienced developers who have gained sufficient knowledge about system development by working on previous projects. The informal model used by such an estimator is expressed as a set of “rules of thumb” or, at an even more primitive level, as a “gut feel." When questioned as to how they developed their model and how they apply it, estimators are usually unable to say exactly what it is they do. It appears to be an issue of gaining the required experience in order to arrive at accurate estimates.

Formal models attempt to quantify all inputs to the cost estimation process, and then apply a set of equations that describe the relationships between the inputs and the outputs of the cost estimation process. The equations are developed through analysis of historical data and must be calibrated to each individual development environment. The best known formal models are Boehm’s COCOMO [Boeh-81], Albrecht’s function points [Albr-83], and Putnam’s application of Rayleigh curves to the development process [Putn-92].

The usual method of applying the formal model is to transform the requirements into a measure of the “size” of the system. This size measure, which can be either SLOC (Source Lines of Code) or FPs (Function Points), is used as the basis for creating the cost estimates. The estimator can also quantify a set of other cost drivers, examples of which include

- Product attributes, e.g., required reliability, product complexity, etc.

- Computer attributes, e.g., memory constraints.
- Personnel attributes, e.g., applications experience, programming language experience.

These cost drivers become multipliers that can be used to increase or decrease the initial estimate.

The bulk of the current literature and research on cost estimation is devoted to formal models, particularly as relates to new system development. As discussed further in Section3.2.3, we found that formal models are not in general used by estimators as a primary tool for cost estimating.

Analogy-based estimating processes estimate costs by comparing the current development project with previous development projects undertaken by the organization. An analogy-based technique requires maintenance of a history of past projects; this information can be used as a reference point. Past projects with properties similar to the current project are identified and their costs used as a basis for estimating the current project.

At the most informal-level of analogy based techniques, the history of past projects is maintained in the estimator's memory. Finding past projects with properties similar to the current project involves the estimator thinking "Which projects does this remind me of and how much did they cost?" Such an approach is highly dependent on the memory of the individual estimators and a very low employee turnover.

The analogy-based approach can be made more rigorous in a number of ways. The history of past projects can be maintained as a computerized database, with detailed metrics and descriptions of characteristics recorded for each project. Using a historical database, an estimator can query the database searching for projects with similar characteristics and then base the estimate on actual costs and process of the previous projects. Such an approach avoids the fallibility of human memory and provides a much more detailed historic record of what occurred in the course of a project [Cowd-89].

3.2 Current Practices

This section summarizes the practices that we encountered at different organizations for producing cost estimates and controlling the costs of projects.

A wide range of organizations was involved in the study. The characteristics, purposes, and constraints of these organizations vary quite dramatically, making it difficult to analyze and generalize the data. To allow comparisons across projects it was necessary to categorize the organizations; the data were then analyzed within each category. For the purposes of this study, the organizations were classified in the following way:

- *Commercial development* is primarily concerned with developing new products that can be sold to more than one group without significant modification.

- *Contractors* are organizations that agree to build a system for a procuring agency, to the procuring agency's specifications.

- *Procurement agency* is an organization responsible for obtaining a system. For this study, we are interested only in procurement agencies that contract for the development of new software.

- *Maintenance organizations* are organizations that maintain large existing systems.

3.2.1 Timing of the estimates

Estimation is not a task done once, at the beginning of a project. Rather, estimates and re-estimates are undertaken throughout the life of a project. The success of an estimator is not necessarily the accuracy of the initial estimates, but rather the rate at which the estimates converge to the actual costs.

The timing of estimates depends on the type of organization involved and why the estimate is being performed.

Contractors usually perform two estimates early in the development life cycle. The first is done to prepare a bid for the contract, usually in a relatively quick fashion, with the objective of arriving at a winning bid. The timing of this bid is very much dependent on the procuring agency that issues the RFP. The contractor is required to generate an estimate at this point, basing it on information within the RFP and obtained informally from the contracting agency.

Upon winning a bid, most contracting organizations immediately undertake a second, more detailed, estimation process. The objective of this estimate is to develop a more accurate and detailed cost estimate and project plan which are based on the previous estimate and WBS. Frequently, much discussion between the contractor and the agency is necessary to deal with previously undetected issues and problems in the requirements.

During the course of the project, contractors may or may not perform detailed estimates of cost to completion. The current practice varies, with some organizations having regular and formal re-estimates of cost to completion, while other organizations perform informal and ad hoc re-estimates as required when initial cost estimates are clearly not going to be met. All organizations interviewed have some type of regular review of current status and expected cost to completion, but there was sufficient variation between the organizations that it is difficult to generalize about when and how it is done.

All the procurement agencies involved in the study were DND organizations. These military procurement organizations have a standard estimating procedure leading up to budgetary approval. Larger projects are initiated by a Project Director (PD) and move through different levels of project status (E, D, C, and B), reflecting different levels of approval. When a project has reached the top level status (A), Treasury Board approval for spending has been obtained.

For each project status the Project Director provides an estimate of the overall project cost with accuracy increasing through the various status levels. The

estimates are based on different levels of specification and have different accuracy requirements. A class A estimate, which is typically found in contracts, is expected to achieve an accuracy of ±10%. Once budgetary money has been approved, it is difficult to revise this figure; the estimate is assumed to be correct.

An individual project within DND can have many phases and many associated contracts. Although monitoring and re-estimates are provided in all projects, the method and timing for performing these re-estimates is difficult to generalize across all projects. Regardless, because the A-level funding is fixed and difficult to alter, there is strong pressure upon the estimators not to increase cost estimates during the monitoring and re-estimation of projects. When the project is over budget, there is no easy mechanism within the current procurement process by which the PM can reduce functionality or increase the budget of the system.[1]

Accurate estimates by DND often require estimates done by contractors or potential contractors. Contractors can be paid for detailed price estimates or "price and availability" may be used. The general feeling was that you get what you pay for and it was worth an investment of money to receive an accurate estimate from a potential contractor.

Maintenance organizations organize their work around individual change requests and this affects when and how they estimate. When a change request is first received, normal practice is to perform an impact analysis on the change request. The impact analysis includes an estimate of the impact of implementing the change, including estimating the effect on the software architecture and the cost of implementing the change. These estimates are usually performed by the individuals directly involved in implementing the change. The time taken to perform an impact analysis depends on the complexity of the change being proposed. For simple changes, the effort involved in the impact analysis could be as short as a few hours; for complex changes this analysis could stretch into days or weeks.

The second stage of a maintenance organization's estimation process is to determine the releases. All maintenance organizations involved in the study went through a process of defining a new release, which is essentially an estimate of what will be released, and when, based on the effort estimates within the impact analyses and the priorities of each of the change requests.

Re-estimates during development are performed by the different maintenance organizations, although when and how these re-estimates are performed varies

---

[1]A number of approaches to providing a mechanism for altering functionality during development are being looked at. The first example we encountered was *common purpose procurement,* which is being defined by DSS. The objective is to define a problem and then to work with a contractor to define and implement a solution. Another example that we encountered was the ARDS R&D project, where the complete functionality is defined early in the project, but the actual functionality implemented is decided much later in the project. Both of these initiatives are in preliminary stages, and thus it is too early to attempt to define their relative successes.

among the organizations. As each re-estimate is performed, the estimated delivery date and functionality of the proposed release are updated.

3.2.2 Estimation Constraints

An estimation process involves arriving at an estimate that satisfies the constraints. These constraints vary depending on the timing of the estimate and the organization performing the estimate, but can include:

- System requirements.
- Delivery date.
- Financial.
- Manpower resources.
- System architecture.
- Software process.

Contractors. When preparing a bid to develop new software, a contracting organization is usually faced with constraints on system requirements, delivery date, manpower resources, and software process. Depending on the system under construction, constraints may be placed upon the architecture.

The constraints on the requirements of the system vary considerably among projects. Some projects have requirements which are well understood and well documented within the Request for Proposal (RFP). In these cases, the constraints on the requirements are well understood by all parties involved. However, in many cases, requirements are not clearly understood up front, or are flexible in terms of the actual functionality to be delivered as part of the end product.

Delivery date and financial resources are constraints that are very firm and have a large impact upon a contractor's preparation of a bid for estimation purposes. There are two reasons that these constraints are imposed upon contractors. First, the procuring agency has a budget and timetable, which they are under pressure to meet and which they are not willing to exceed. Second, there will be competing bids submitted.

Almost all contractors indicated that they knew (or felt they knew) the budget of the procuring agency while they were preparing the bid. They also considered this to be vital information for preparing a competitive bid. Although not usually included within the RFP, contractors would learn (or guess) this information through informal contacts with the procuring agency. Any estimate was required to satisfy the financial constraint.

If, while preparing a bid, the estimator arrived at an estimate that exceeded the financial constraints, it was necessary to alter the bid in some way to satisfy the constraints. In many organizations, the bidding team would put pressure on the estimators to reduce their estimates (often while still trying to maintain all the

functionality). The more realistic approach is to perform a trade off between functionality and cost; this trade off is frequently done by contractors while preparing the bid.

Once the bid has been won, the contractor performs another more detailed estimate. This estimate is in many ways more realistic because there is less pressure to satisfy financial constraints; it is usually done by the project manager to determine how much the system is really going to cost. Although financial constraints affect the process, the manager usually defines in much more detail the functionality of the system and the process used to develop the system. This results in a more accurate estimate and can determine whether the system may be built for the contracted price.

Re-estimates done by contractors during development involve modifying the duration, effort, and functionality. As understanding of the tasks increases, more accurate estimates can be made regarding effort and duration. As the requirements of the system are better understood, they can be re-estimated and appropriate modifications made to the effort and duration estimates.

Procuring Agency. From a procuring agency's perspective, estimates are performed under a different set of constraints. Project Directors try to balance the following constraints while getting approval for the project:

- *Financial*. How much money is the organization willing to put into this project?
- *Calendar*. When do I have to show results to keep management satisfied?
- *Requirements*. What is the functionality required of the system?

Each of these constraints has a different level of priority, depending on the particular project.

Once project development begins, control of the project passes from the Project Director to the Project Manager (PM). At this point budgetary approval has been received and all previous estimates are considered to be cast in stone. Thus, there is great pressure on the PM not to change any of the previous estimates.

When it is clear that the original estimates will not be satisfied, PMs must re-estimate. The PM must decide in what order to sacrifice the financial, calendar, and requirements constraints. Different PMs have different approaches; generally they try to maintain the functionality of the system, but let either the calendar or financial constraints slip. In reality, however, it appeared that if the original estimates were incorrect, all of the constraints were affected.

Commercial organization. A commercial organization's overriding concern is maintaining market share, and this defines its constraints. It is required to develop products with a sufficient number of features in as short a time period as possible. The delivery date of the product is the overriding concern, since a product coming in late may miss a market window. Thus, the main constraint on commercial developers is the duration of the project. Other factors, such as

functionality, reliability, and maintainability, can be sacrificed in reasonable ways to maintain the release date. Commercial organizations involved in the study usually have rules that allow them to perform these trade-offs in a deliberate and conscious way.

Maintenance. Maintenance organizations are usually the most constrained in what they work with. They generally have a work force whose size is fixed before the maintenance work is estimated, the system architecture is fixed, and frequently a release date for the new version of the software is fixed. The estimation process comes down to the maintenance organization estimating what functionality can be included in the next release given fixed loading, effort, and duration. The input to this process is the set of change requests, together with the estimated level of effort for each change request.

The initial estimates performed during the impact analysis are the least affected by the constraints. These estimates, performed by a technical person, are concerned primarily with determining the effort required to implement an individual change request. At this stage, the main constraints on the estimator are those imposed by the defined architecture of the system and the defined development process. The estimator must determine the amount of effort necessary to implement the change. Depending on how well defined the change request, the estimator may be able to trade off functionality and effort.

The second estimation stage in a maintenance organization is to perform the estimates regarding the next release. Defining the functionality of the next release is not “cost estimation” by the strict definition of the term. Maintenance organizations are stable organizations with a fixed amount of manpower devoted to the maintenance activity. Thus, their effort and manpower loading are fixed. As well, maintenance organizations are frequently given a fixed release date. Thus effort, duration, and loading, are all fixed before the release is defined. The estimation process reduces to determining which change requests will be included in which releases. Because most other constraints are fixed the estimator can make very few trade-offs, other than altering the number of change requests to include in the next release.

During the course of the maintenance activity, procedures vary among organizations as to when re-estimates are performed. The most rigorous approach we encountered was to have the implementor perform a quick re-estimate of work remaining after each milestone achieved for each change request. For example, once design of the change request was complete, the implementor would re-estimate the amount of work required for implementation, testing, and integration; the inputs for this re-estimate would be a detailed knowledge of the change to be done and the implications on the software architecture; output would be effort and time until completion.

Maintenance units also re-estimate the next version release. This re-estimation is usually done needed, when it becomes clear that the original version estimate is wrong. The original version estimate can be determined to be wrong because the change requests on which it was based are found to be wrong, or because new change requests arrive that are deemed to be sufficiently critical that they must be included in the release. At this point, the estimator has

available not only all the information from the original estimate, but also the re-estimates for each of the change requests. The estimator has two factors that can be output: modify the duration by extending the release date to include all the required change requests; or remove functionality in order to meet the release date.

3.2.3 Estimation Process

In Section 3.1.3, the process by which estimators arrive at an estimate was categorized as either *model based* or *analogy based*, with these categories having different levels of formality.

By an overwhelming majority, informal analogy was the most commonly used estimating method for all types of software and for all organizations. Estimators used their past projects as a basis for estimating the cost of future projects. This has two implications for the estimating process:

- The “training” provided to estimators consists of having them work on projects within the organization for a number of years before they perform the estimates.
- The estimates are almost always done by the people who will be responsible for the implementation.

In general, estimators did not refer to any historical database in any significant way in order to use informal analogy as an estimating tool. The “database” consisted of their memories or the memories of their colleagues. Estimators often admitted that some information was available on past projects, but it was either in a form too difficult to access, or they did not believe accessing the information would improve the accuracy of the estimate.

Only three organizations made any attempt to formalize the analogy process by systematically comparing the current project being estimated to past, completed projects. Two of these had the advantage that they were producing families of systems, where all members of a particular family were similar in terms of the architecture and the subsystems from which the system was constructed. This allowed the estimators to compare the current project with previous systems on a subsystem by subsystem basis to determine similarities and differences. The third system was a DND weapon system procurement, and the contractor had some data gathered from previous similar projects. The analogies were done at the subsystem level to arrive at costs for the individual subsystems. In this case, the data regarding previous projects were not nearly as extensive as for the previous two cases. Since the project is in its early stages it is too soon to determine the success of the estimate and whether the analogies are sufficiently close and have been applied in the proper manner.

Formal models were not used extensively for cost estimation. Almost all estimators had developed a set of “rules-of-thumb," which can be viewed as informal modeling. The lack of formal models were notable by the exceptions:

- Two organizations involved in MIS development made extensive use of formal models based on Function-Point-like measures, either as the primary estimating tool or as a backup to other estimation techniques. Both organizations had at least two years of data available that could be used for calibrating the model.

- One DND procurement organization made extensive use of formal models to construct estimates of software development for a large scale embedded system. The estimates were performed for individual subsystems of the overall software system.

- One organization made minimal use of a very simple model based on Source Lines of Code (SLOC) to provide a "sanity check" on estimates developed by other means.

A "non-traditional" use of models was used by DND to estimate the size of software maintenance units required for major weapons systems. COCOMO was used to estimate manpower requirements for maintenance centers based on knowledge of the size of the system and predicted volume of change requests. This was being done for maintenance units being set up, so the accuracy of these estimates cannot be determined at this point.

Two of the organizations that made use of formal models on a regular and ongoing basis shared two characteristics: they recorded detailed historical data relevant to their cost estimation models; they produced a number of systems within a limited application domain. These organizations claimed they could arrive at accurate estimates using these models (within 10% to 25%).

There were two major reasons given for organizations not using formal models. First, there was a lack of confidence in the ability of a model to outperform an expert. Managers felt that these models were expensive to implement and provided little benefit. One perspective was summed up by the comment "if I know how many lines of code are in the system I don't need a model to tell me how much it is going to cost."

The second problem with the models is the lack of historical data available to calibrate the model. Proponents of models emphasize the fact that models are not transferable between organizations and that there is a great deal of effort required to calibrate a model for a particular organization. Without calibration, values produced by the model can fluctuate radically. For example, within the COCOMO model simply selecting different values for the multipliers can vary the minimum and maximum estimates by 700 times. Without historical data, it is impossible for an organization to determine the correct values for these multipliers.

The most difficult estimation problems encountered were those in which an organization was developing a software system outside its immediate domain of expertise. In this case, historical data of the organization, even if available, are not particularly relevant. Undertaking a project in a new application domain is generally considered to be a high risk by an organization. Although

occasionally an organization arrived at an accurate estimate, in most cases the cost was severely underestimated regardless of the estimation process used.

3.2.4 Personnel involved

There are two opposing views regarding who should perform estimates within an organization. The first states that estimates should be performed by people who are directly involved with and have a stake in the implementation of the system. Thus, the persons responsible for development will estimate the amount of work required.

A second view is that the estimate should be performed by an independent authority who can provide a completely unbiased estimate [DeMa-82, Putn-92] by using historical data on past project developments, to estimate the current project. The estimator would have no personal stake in the development of the project and his performance would be based on his ability to develop estimates that converged quickly to actual costs of a project.

The consensus view which we found was that people involved in the estimation were the people responsible for implementation. Almost all organizations surveyed required one of the managers in charge of developing software to come up with the cost estimates. Software managers would do this either on their own, if their knowledge was sufficiently broad, or in consultation with technical people who would be working on the development. For contracting organizations, we occasionally found the situation where one manager would perform the estimate while another would do the implementation, but this was often due to manpower constraints rather than a deliberate effort to have an estimator who was independent of the implementation process.

The people who performed the estimates were not, in general, provided with any training in software cost estimation. They were expected to develop the necessary skills through their work on project development. In many cases, organizations did not provide any training to their project managers.

There was one exception to the model of the person(s) responsible for implementation doing the estimating. This was an organization which depended heavily on formal models for cost estimation purposes. The organization had available a computer tool that implemented the formal model. A few of people were trained in the use of the model and the tool and performed most of the estimates.

3.2.5 Data gathering

It seems obvious that without knowledge of the past, it is impossible to predict what may happen on future projects. (Even with knowledge of the past, there is still no guarantee that the future can be predicted.) A corollary is that if an organization wants to improve its cost estimation process, it must gather relevant data on previous projects.

Virtually all the organizations surveyed recognized the benefits that could be gained by gathering historical data to use in estimating. However, very few

organizations had an effective means of gathering data on their processes and their projects; even fewer organizations were able to apply the data gathered to improve their estimation accuracy.

The simplest way to gather data is to have a stable work force so that project and process data are maintained in the memory of the individuals of the organization. The individuals can then use this information to estimate costs of other projects. However, relying on individuals' imperfect memories is barely sufficient for small projects; for large projects it is completely inadequate.

To overcome the limitations of relying on individuals' memories to record project data it is necessary to have a more rigorous approach to data collection. A few organizations tried to write a summary report of each project upon completion of the project. These reports would contain a summary of the product and the process used, and any "lessons learned" associated with the project. The few organizations that attempted this type of historical record found that the record was very infrequently referred to by future project managers, and so there was little incentive to actually write these records upon completion of the project.

Metrics are quantitative means of recording the history of a project. All organizations were trying to gather, or recognized the need to gather, metrics on their development work. Very few organizations had yet found useful techniques for gathering and applying metric information to cost estimation. All organizations had some form of data gathering, but in many cases the data gathered were not in a form useful to estimators.

It is not possible to estimate the cost of future projects unless the cost of previous projects is known. The easiest way to determine the effort expended on previous projects is through time sheet data, which records the person-hours expended on a project. For many organizations, this is the only "metric" gathered. However, such raw data alone do not necessarily assist the estimator. Even if this information is gathered, it is often done for financial purposes and is not used by software managers to estimate the cost of future projects. There are a number of reasons why these data may not be useful:

- *The data are not accurate*. If the primary perceived purpose of time sheets is to monitor the staff, the accuracy of the figures in the time sheets must be questioned.

- *The data are not accessible*. Often time sheets are gathered for the benefit of the financial department rather than to assist estimators. Thus, they are kept on systems not easily accessible to estimators, or worse, are simply stored as masses of paper files.

- *The data are not broken down in a useful way*. The overall cost of a project has a limited usefulness. What is usually of more interest to an estimator is how the project was broken down into activities and the cost of each of these individual activities. Organizations broke down activities differently for time sheet data collection, but many gathered time sheet data in very broad categories.

Only three organizations we encountered had a metrics program in place that was sufficiently mature to be an assistance to estimators in trying to project future projects. These were programs where there was a clearly defined goal in gathering the metrics and for which sufficient data had been gathered from previous projects to be useful.

Many other organizations were in some stage of implementing a metrics program to gather historical data. The level of implementation ranged from organizations that had a complete metrics plan but did not yet have sufficient data to make valid projections of future projects, to those organizations beginning to define their metrics plans. The problems most often cited by these organizations included:

- *Usefulness of metrics*. Many organizations were not clear on how to use the metrics gathered. One had instituted an extensive and expensive metric gathering program, but as yet had no idea how these metrics were to be applied to future projects. It seems that before beginning a metrics program, organizations should ask how the metrics are to be used, rather than gathering the metrics and then trying to determine what to do with them.

- *How to gather metrics*. Depending on which metrics are gathered and how they are gathered, metrics can be expensive. Organizations with successful metrics programs had found a straightforward and inexpensive means of gathering the metrics through the use of automated tools. These included intelligent use of time sheets, automatic recording of problem reports and their status, coding conventions which allow SLOC and change histories to be computed automatically, etc.

3.2.6 Cost Estimate and Cost Control

One of the purposes of performing a cost estimate is to have a means by which the development costs can be monitored and controlled. One of the outputs of the cost estimation process is a development plan with a defined set of tasks. If the tasks have a well-defined output and cost associated with them, the progress of a project can be monitored by determining the actual completion and costs of tasks and comparing this with the estimates.

The monitoring can be performed at a macro or a micro level. At the macro level, the monitoring agent is concerned with schedule and progress, and monitors them primarily by reviewing major milestones. This, for example, is the role assumed by a procurement agency whose primary means of project monitoring is a review at major milestones, such as the preliminary design review and critical design review. The reviewing agent typically has sign-off authority at these reviews, and can measure completion of the project by when the milestone reviews are completed. This is the role assumed by DND when monitoring a project required to follow the 2167a standard.

Monitoring projects by reviewing critical milestones frequently assumes a waterfall model of development. As the project moves through the waterfall, the monitoring agent can see the project fall by means of the milestone reviews.

For major projects, the critical reviews by procuring agencies were not always an adequate means of monitoring the progress of a project. Reviewers cited the following major problems in monitoring projects by this mean:

- *Document to review is too large and incomprehensible*. Regardless of how diligent a reviewer is, a manual review of a large technical document can never assure that the document is complete and correct.

- *Contractors modify review documents without proper configuration control*. Therefore, the final system does not correspond to the reviewers understanding of what was to be delivered.

Two other views could be discussed here: reviewer should review the *process* not the *product* (Watts Humphrey); and the reviewer should review the only the *functionality* not the design or code.

Contractors involved in large development projects, consistently complained to us about the difficulty in getting a proper review from the reviewing agency. They said that it was difficult to obtain review meetings, too many people were involved in reviews (leading to too long a review period with too many contradictory comments), and staff turnover at the reviewing agency led to reviewers who were unfamiliar with the project.

At the micro level, monitoring is performed by a manager to determine the day-to-day progress of the project. One of the outputs of the cost estimation process is a detailed WBS identifying the activities required to complete project development and the effort, loading, and duration of each task. The size of activities defined within the WBS is dependent on the level of detail of the estimate and size of the project. Smaller projects would normally deal with work activities of, at most, a few man-weeks. Large projects would deal with activities of up to a few man-months.

All companies we talked to constructed a WBS during the early cost estimates. However, early estimates (e.g., during bid preparation) would not necessarily have the level of detail in their WBS that later estimates would have. For large systems, the level of detail in the WBS regarding software may be minimal, with a single WBS item called “software," with no further indication of the software subsystems required or even what was included within the item “software” (e.g., did it include documentation, training, etc.)

Most organizations would use the WBS as an input to the cost estimation process, to estimate the cost of each activity. The sum of the individual activity costs gives the overall project cost. A couple of exceptions to the above were noted, whereby an overall project cost would be determined, without constructing a WBS (e.g., using parametric models or the costs of previous projects). Given the overall cost, the different WBS items were then identified

and the costs allocated to each WBS activity. If estimates were done in this fashion, there tended to be a relatively standard formula for breaking the cost down into WBS activities (e.g., requirements analysis - x%; coding - y%, etc.).

For management (or a procuring agency) to use cost estimates as a tool for monitoring and controlling costs it is necessary to have a mechanism in place monitoring a project and to be able to maintain profiles of “actual costs” versus “estimated costs” for the project. Some general observations regarding how organizations monitor costs and create profiles of estimated and actual costs are

- Organizations that developed software on a contract basis almost always knew the exact value of their development costs and how much they made or lost on each contract.

- Commercial organizations were generally not as concerned with overall cost. Customer demand and market forces required them to deliver working software systems regardless of the cost. Although they were interested in what the cost was and in minimizing it, they were much more concerned with time-to-market than final development cost.

- For software development work done internally, there was a split between those groups who knew the precise cost of a development project and those who did not have sufficient data to determine software development costs.

At a more detailed level, it is necessary to determine during the course of a project how the project is progressing relative to the original estimates. This requires being able to track the progress relative to the original WBS, and to re-estimate the cost-to-completion as required.

Tracking progress requires being able to determine the completion of the WBS activities. We asked the participants in the study how they measured “completion” and received a number of different answers. Most organizations conceded that they have difficulty in measuring completion. Certain activities can be easily measured, e.g., if a set of integration tests has been correctly defined, then completion of integration can be measured by the number of tests completed. Means used to measure completion include the following:

- *Inspections and reviews*. Each activity in the WBS should have a set of well defined outputs. An activity can be said to be completed when all the outputs of the activity have been inspected and approved by the inspection team. For major milestones, there may also be a review by the procuring agency (e.g., for the CDR).

  Inspections and reviews are often not adequate in and of themselves to determine activity completion. Often there is insufficient time to perform complete inspections of every activity output. If the output is extremely long or complex, there will inevitably be errors and omissions regardless of how closely it is inspected. In such cases, to

determine whether an activity has really completed when it said it did, it is necessary to track all faults and problems that result from the outputs of the activity being incorrect. Very few organizations perform such fault analysis.

- *Blind faith*. Most organizations used the blind faith method for determining when an activity was complete. Whenever the person(s) performing the activity claimed it was complete, it was assumed to be complete. Within these organizations, the later activities of the development cycle seem to be severely underestimated.

- *Recording problem reports filed on the output of each activity.* In order to measure whether a WBS activity has really been completed, it is necessary to have a well-defined output for each activity and a means of determining whether the output is complete and correct. One means of measuring the completeness and correctness of the output of an activity is to measure how many problem reports generated during development can be traced back to a specific WBS activity. If there are a significant number of problem reports related to an activity, it can be assumed that the activity was not correctly completed at the time it was declared completed. All the organizations surveyed had some means of determining filing problem reports during development. Only one organization, however, actually traced each problem report back to its original cause and thus had a measure of the error rate for each activity.

4. Problems with the Cost Estimation Process

Is there a problem with generating accurate software cost estimates and, if so, what is the cause of the problem?

All organizations agreed that there was a problem with software cost estimates, although the degree and nature of the problem varied among the organizations. Unfortunately, because of the nature of the data we managed to collect, it is not possible to give statistics on the level of accuracy of estimates.

A few organizations claimed to consistently achieve estimates within a 10% accuracy, although without hard data we cannot verify that the actual costs were this accurate relative to the initial costs, nor can we verify that the system produced satisfied all the requirements and met the customers' expectations. Most organizations did not claim that they could estimate within a 10% accuracy, particularly in projects outside of the application domains in which they had extensive experience. Most of the organizations could identify at least one significantly underestimated project in their past and a few organizations had consistent problems in their estimation accuracies.

The situation may or may not be worse than in other fields of engineering. For example, comparisons of hardware and software projects within the NRC IRAP database indicated that software development projects are not the only ones that encounter difficulties in cost estimation [Vigd-94]. All organizations however felt that something better was necessary and possible.

What factors make software cost estimation difficult? There were situations where we found a high level of accuracy in cost estimation; many of these situations were identified by the following characteristics:

- The users are experienced in the system, know what they want, and can express what they want.
- The requirements are clear, precise, correct, and complete.
- The project duration is short .
- The manpower loading is small.
- The people doing the estimation are experienced in the application domain and have developed similar systems.
- The development environment and development process are familiar to all people involved.
- Staff turnover is low both among the developers and the users.
- No unfamiliar software or hardware from outside suppliers is to be integrated with the final product.

A project satisfying the above characteristics frequently resulted in accurate cost estimates. However, most of the projects did not satisfy the above conditions and therefore the estimates produced were not accurate. The characteristics needed for accurate estimates can be reversed in order to enumerate problems leading to inaccurate estimates:

- Problems with the requirements.

- Issues in maintenance.
- Procurement process.
- System size.
- Software process and process maturity.
- Monitoring progress of the project.
- Lack of historical data.
- Lack of application domain expertise.
- Embedded software.

These issues are discussed in the following sections.

4.1 Problems with Requirements

Almost universally and without exception, organizations blamed problems with the requirements as a major reason why cost estimates were inaccurate. This was true regardless of whether we talked to development or maintenance organizations, large or small organizations, contractors or procurement organizations, MIS or real time organizations. The problems cited were numerous: incomplete, ambiguous, inconsistent, incorrect, incomprehensible.

Why is it so difficult to write correct, clear, and complete requirements? There is no definitive answer to this problem. We are not the first to recognize this problem, nor will we be the last. Through discussions with the various organizations involved in software development, we have identified the following as some of the reasons for a lack of adequate requirements.

Users do not understand their requirements during the early stages of the project. Software projects are often undertaken when there is a recognition that a problem exists but no clear idea of the real problem, or the solution to the problem. Yet at this early stage of problem recognition, someone is expected to be able to write a requirements specification at a sufficiently detailed level such that an accurate cost estimate can be made. It is clearly not possible. Estimates can be made at this stage, but it must be recognized that they are inherently inaccurate.

The problem of users not understanding the requirements existed for all types of systems and all types of developments. For new development projects, users would request systems (and quotes) before there was a complete understanding of the problem or the solution. For maintenance organizations, estimators were expected to cost out features to be added to the system before the features had been fully defined. Both cases result in highly inaccurate estimates.

Cost estimates can be made without a clear understanding of the requirements of the system being built; it must be accepted that these estimates have a very high likelihood of error.

Requirements creep. As projects progress and the knowledge of the problem increases, it seems inevitable that users (and developers) request more and more features and changes to be included in the product. Thus, over the development of the project, new features work their way into the requirements,

leading to “requirements creep” (or, as one participant described it, “requirements gallop”). New feature requests come from many sources and for many reasons, but the problem seems to be universal.

Correct and complete requirements for complex systems are impossible to achieve. A fact that must be accepted is that a complete statement of the requirements cannot be defined before development begins [Hump-89]. This has nothing to do with the competence of the users or the developers but rather is inherent in the nature of complex computer system applications. Unless the system being developed is almost identical to a previously developed system, the requirements will invariably be wrong and/or incomplete. As a project evolves, users and developers gain a better understanding of the problem and of the solutions. As people gain a better understanding of the problem being solved the requirements evolve.

One frequent assumption is that the requirements will be firm before development begins. Anyone working under this assumption will meet serious problems when trying to estimate software costs accurately. Since the requirements are probably wrong or incomplete, it is unlikely that the estimates based on those requirements will be accurate.

If the requirements are included as part of the RFP put out by a procurement agency and a contractor is expected to submit a firm bid based on those requirements, a frequent result later in the development stages is confrontation between the contractor and the agency as they argue over the meaning of each requirement and the cost associated with the changing requirements.

Requirements written by people who do not know how to write requirements. Both DND and some of the larger development organizations said that they ran into problems with requirements because they did not know how to write requirements.

Long development time, leading to requirements that are obsolete before the system is delivered. The rate of change in technology is so fast that any attempts to predict what the technology will be in a few years are doomed to failure. As the technology changes, so do the range of solutions to problems, and the users’ expectations of the solutions. Projects with a long time between initiation and expected delivery suffer in that the solution is usually obsolete by the time it is delivered. The customer is dissatisfied because the product does not satisfy the new requirements.

Large staff turnover for end users, resulting in changing requirements as new people arrive. Developing software systems requires a consistent users’ base throughout the development cycle. If the users’ base changes too frequently, requirements continually change, and it is difficult for developers to obtain consistent answers and comments from the end users. This problem was mentioned by a number of contractors, particularly those who are involved with DND, where military rotation may require multiple turnover of personnel during the course of a project.

4.2 Maintenance organizations

Little attention has been paid to software maintenance costs despite the fact that, over the life of the product, maintenance costs may be significantly higher than development costs. Despite the fact that maintenance is expensive, little or no attention is paid to life cycle costing of software. In only a single organization did we find that maintenance costs were considered during development, with conscious trade offs made between development costs and maintenance costs.

There are factors that make cost estimation an easier task for maintenance organizations, including:

- Maintenance organizations are concerned with software systems for which the software architecture is well defined (though not necessarily well understood) and does not change significantly. Since design of the architecture and subsequent modifications to it are not significant cost factors, a major unknown variable is fixed. Assuming that this architecture is sufficiently well documented and well understood by the maintenance team allows the resulting estimates to be made much more accurately.

- Estimates are done on the basis of change requests. Because an individual change request is small relative to the entire system, it is easier to get an accurate estimate for an individual request. The cost estimate for the release can then be computed as a function of the cost of the individual change requests.

- Manpower resources and calendar release date are frequently fixed constraints, reducing the number of variables to be considered during the estimation process.

There are problems that a maintenance organization can inherit from the development organization. One issue is whether the development organization has designed the system to be maintainable. Issues affecting maintainability include software design, documentation, coding standards, etc. Many of these issues require a conscious decision by the development team and a cost in implementing them. If pressure is on a development organization to satisfy cost and/or release date constraints, there is a tendency to ignore issues relating to maintainability.

Another development issue that affects maintenance organizations is the level of completion of the software when the development team declares it to be finished. If the pressure to satisfy cost and calendar constraints is intense, a development organization may declare an incomplete piece of software as “done” in order to claim that cost constraints have been satisfied, even though the software contains major deficiencies. The maintenance organization then inherits an incomplete piece of software, and their first “maintenance” task is to finish the software.

Both of the above stated problems are amplified in the DND environment where the development organization and its budget are completely separate from the

maintenance organization and its budget. The project manager's responsibilities end when the completed system is delivered; he/she has no stake in the maintenance effort. Thus, there is no incentive to guarantee that an easily maintainable piece of software is delivered to the maintenance group. Rather, the major incentive is to complete the project on time and under budget.

The major problem encountered by maintenance organizations when they try to arrive at accurate software estimates is the issue of requirements creep. Maintenance organizations are frequently calendar driven. Given a set of change requests and a delivery date, the organization estimates which change requests can be completed by the scheduled release date. However, during the course of the maintenance work there is a great temptation to add new change requests to the scheduled release without modifying the corresponding release date. This causes a great deal of frustration as the users do not get the promised changes, and developers are required to work to impossible schedules trying to meet the scheduled release as originally planned while including functionality not anticipated when the original schedule was created.

The second problem with maintenance estimating is the large amount of overhead that is often associated with making a relatively small change. Typically, each change request is allocated to a designer to perform an impact analysis, which includes a cost estimate and a description of the impact on the design of the system. For minor changes, the effort to actually perform the estimate is most of the cost of designing and coding the change. If the designer has opened the source code and determined how the change is to be implemented in order to perform the estimate, most of the work of designing and coding is complete before it has been determined if or when the change is to be implemented.

4.3 The Procurement Process

In an "ideal" software costing process, a set of system requirements is input to the process and a cost estimate is output. In reality, this is far from what actually occurs. In many cases, constraints other than system requirements drive the cost estimation process and distort this process. Nowhere is this distortion more evident than in the procurement process used by DND. The effects of the procurement process are felt by both the procuring agency, and the contractor.

Within the DND, a project requires approval at different project status (D-capital, C-capital, B-capital, A-capital). Once a project has attained A-capital status, authority for spending money has been obtained [Prog500]. It is difficult to alter the associated cost estimate for which spending authority has been received. This has a number of implications for the procuring agency.

First, the estimates associated with A-capital status are generally established before the complete requirements of the system are known and almost certainly before any architectural design has been performed. This forces PDs to create accurate cost estimates on a minimum amount of information. This results in estimates that can be modified only with great difficulty, but that are based on very preliminary information.

Second, the Program Managers (PM) are given these estimates as “the law” and are under great pressure to deliver the product within the estimated time and under the estimated budget. Because the PM is under such budgetary pressure, and because his responsibilities end upon product delivery, there is no incentive for the PM to worry about maintenance costs. Since maintenance costs of major systems can be much greater than the original procurement costs, it can be very cost effective to increase the cost of the original development cycle if this results in more efficient maintenance. The current DND procurement process provides no incentives for PMs to make such trade offs. It is interesting to note that, of all the surveyed organizations, only one made a conscious budgetary decision as to whether development costs should be traded off for reduced maintenance costs.

From the perspective of the contractor, the procurement process forces the contractor to make cost estimates that are not necessarily computed as a function of the requirements of the system to be developed. The procuring agency has a budget for the procurement; if a contractor wishes to win a contract the bid must be within this budget. Although the budgets are not normally published as part of the RFP, contractors to whom we talked considered it very important to have a good idea of what those budgets are. This information is generally determined through informal contacts between the procuring agency and the contractor. Once the budget of the procuring agency is known (or guessed), the contractor can then construct an estimate within this budget.

Most contractors had a two-stage estimation process: the pre- and post-contract estimates. The pre-contract estimate was used as a basis for the formal bid for the contract. The strategy used by most companies was generally a “bid to win” approach. Based on the Statement of Work (SOW), the system requirements, and the budget of the procuring agency, a company would use some method to arrive at a price for the work. Such bids were often prepared in a rather hurried manner (due to the limited time to prepare the bid) from requirements which were often vague, contradictory, or wrong, with little or no time to prepare a proper architectural design. To win the bid, the company is forced to come in as low as possible. This frequently involves management putting pressure on the engineering department to lower level of effort estimates for the various activities.[1]

Once a company is awarded the contract, it frequently performs another more detailed estimate. This is the “real” estimate in that the post-contract estimate is what the company believes it is really going to cost to build the system. If the “real” estimate is higher than the “bid to win” estimate, the problem for the company can be addressed in a number of ways:

---

[1]Of all the companies we talked to, only one claimed that the bidding team must accept the effort estimates of the engineering department “as is” when preparing the bid. All other companies stated that pressure was put on engineering to reduce estimates of effort.

- Negotiate for more money from the procuring agency. This is usually done later by trying to suggest enhancements or finding problems with the requirements. Contractors realize that procuring agencies generally have about a 10% “contingency fund” and will often try to increase the actual price by at least this amount for any contract.

- Reduce the functionality of the system. Requirements issued as part of the RFP often have enough vagueness and inconsistencies to allow the contractor to modify the final deliverable while still satisfying the SOW.

- Accept a loss on the job. This is a last resort for the company; a company cannot accept a loss on many jobs.

4.4 System Size

A number of the projects we investigated were large-scale systems, involving more than four years duration and more than 100 person-years of effort. Without exception, the costs of all of these projects were seriously underestimated. There were numerous reasons why large projects were consistently found to be exceptionally high risk.

First, there is the sheer complexity of large-scale systems. As reflected in most of the parametric cost models, complexity does not increase linearly with lines of code, but rather exponentially [Boeh-81, Broo-87]. There is no way to eliminate this complexity and it must be accepted as one of the risks of developing large-scale systems.

The larger a system and the further into the future its delivery, the more difficult it is to correctly and completely specify all the requirements. As discussed in Section 4.insufficient requirements is a major reason for cost overruns in all systems. These cost overruns seem to be amplified for large systems. As one would expect, the more complex the system, the more complex the requirements, the more likely that the requirements cannot be stated correctly up front. Another reason for the changing requirements is the length of time until delivery. The longer the duration between initial requirements and delivery, the more likely that there will be changes to the initial requirements. This can occur due to changing user expectations, changes to the environment in which the system is to be installed, or new personnel with different views on what the requirements should be, becoming involved.

A long project duration also means that technology advances may outstrip the initial requirements. We encountered more than one large project in which expensive hardware upgrades were required to modernize the system before the system had even been delivered.

4.5 The Software Process and Process Maturity

An organization cannot hope to achieve accurate cost estimates if it has no clear idea of what it is doing and how it is doing it. A small, knowledgeable,

experienced, and stable development team with competent management has undoubtedly developed a working software process and, even if it is not formalized, written down, and monitored, this process will be followed by the team.

For most of the larger development organizations we interviewed, the state of the software process was bleak. It is clear that a major “software engineering awareness program” is needed. Some of the companies are already embarking on the improvement process, likely driven not by the recognition of the problem but by the fear that certification will soon be demanded by customers. Regardless of the cause for this movement it should in fact be encouraged and helped along. Software development organizations should be made aware of the value of collecting historical data, formalizing the development process, and trying to assess that process periodically.

All of the organizations with whom we talked (with one exception) were aware of SEI and the maturity model. Many had gone through some type of assessment. Almost all could be categorized as “level 1” organizations.

Why are so many of the organizations stuck with an ineffective software process? One of the major reasons is simply the difficulty in putting in place an effective software process. It is relatively easy to write down an effective software process. It is much more difficult to achieve the level of commitment and experience within management and staff that translates a plan into an effective working process.

For DND and large DND contractors, the lack of process maturity was very noticeable. Within DND, there are two problems with trying to implement an effective software process. First, any individual branch of DND only rarely makes major acquisitions. The result is that within DND the knowledge and experience acquired in the previous acquisition has often vanished by the time the branch is due for its next acquisition. There does not appear to be an effective mechanism for translating knowledge and experience between major projects, particularly when those projects are controlled by different branches of DND.

A second reason DND has problems maintaining experience for major projects is the policy of military rotation. The military personnel involved in multiyear projects may go through a number of rotations, resulting in three or four people being rotated through each position within the Project Management Office (PMO). Each time a person is rotated out, the experience of that person is lost. As the experience is lost, so too is the possibility of implementing a mature software process.

From the perspective of the contractor, we also found little collective experience and maturity in dealing with very large, complex systems. Few companies have built up the extensive expertise required to implement a mature and effective software process.

Another problem related to software process that impacts cost estimation is the use of CASE tools. CASE tools can be effectively used within a software

process, and their effect on development costs can be factored into the estimate. The problem we observed was with the introduction of new CASE tools into an organization. Estimators included training and application of the tool when generating an estimate. The problems were caused by

- The training required by the developers is underestimated.
- The effectiveness of the tool in increasing productivity is not as great as expected.
- The tool does not work as expected.

These factors combine to make the cost of introducing the tool into the organization much higher than expected.

4.6 Monitoring Progress of the Project

Software costs cannot be controlled unless the software costs and the development process are monitored and progress is measured. There are at least two different perspectives to monitoring a project. From a project manager's perspective, there is a need to monitor the day-to-day progress and costs of a project. From a procurement agency's perspective, there is a need to monitor progress in order to have confidence that the delivered system will be on time and satisfy requirements.

From a project manager's perspective, there was a notable lack of techniques for objectively measuring the level of completeness of a project. Testing a particular software item against a well-defined test suite can be done objectively. Most of the tasks of software development are not quite so concrete. Most organizations would declare a task complete when the person responsible for the task declared it to be complete. Some would augment this by having selected reviews of material to try to verify that the item really was complete when it was declared to be. However, most managers agreed that this was not sufficient and that it was extremely difficult to determine the actual status of a project.

Milestone reviews and technical and progress reviews are the typical techniques used by procurement agencies to gain visibility into and control over the development process. These agencies often complained that milestone reviews are necessary, but are not by themselves sufficient to monitor progress on a project. The complaints included:

- The contractor changed items without the proper controls because of a lack of configuration management; there were frequent changes required to items that had previously been reviewed but were later found to be inadequate.
- Documents to be reviewed were too large and complex to allow real confidence that the reviewer could identify all problems.

- Items reviewed and signed off required frequent changes due to changing requirements.

There are a number of open issues regarding effective monitoring of contractors by procurement agencies:

- *Data access*. What are the contractor data to which the procuring agency should have access? The data available to the agency should have the maximum amount of relevance to the procuring agency, while imposing the minimum cost and inconvenience to the contractor.

- *Tool use by contractor*. Does tool use by the contractor help or hinder the procuring agency in monitoring the contractor? For example, if a contractor uses a particular CASE tool, does the procuring agency require access to and training with the tool in order to effectively monitor the project? If a reviewer is using the same tools as the contractor, will the same mistakes be made?

- *Tool use by reviewer*. What tools can be used by different reviewers (e.g., procurement agency, IV&V) to make their reviews easier and more effective.

4.7 Lack of Historical Data

An organization needs information about previous projects to estimate accurately what will happen in its next development project. For small organizations working with stable work forces and smaller projects, this may be achievable by relying on the knowledge and expertise of key people in the organization, although there is an inherent danger to this approach if these key people became unavailable. For larger organizations where the projects are more complex, the knowledge distributed among larger numbers of people, and the data may be years old and voluminous, relying on people's memories is not sufficient.

Most of the organizations we talked to had a serious lack of data about past projects. Many organizations could not easily put their hands on such basic data as the initial cost estimates or the actual costs, or in some cases, even the cost of the current project. If an organization does not know its initial cost estimates or its actual costs it is difficult to imagine that it can ever improve its cost estimation accuracy. Even being able to perform a systematic analysis by analogy cannot be done without a detailed record of past projects.

4.8 Lack of Application Domain Expertise

There was a very direct correlation between the application domain knowledge of an organization and its ability to accurately estimate software development costs. It was obvious from our study that organizations that developed similar systems over and over were generally able to arrive at accurate cost estimates. The farther a system was from the application domain knowledge of the

organization, the less accurate the estimates tended to be. Many of the contractors gave anecdotal evidence of a contract undertaken in a new application domain area on which they severely underestimated the work required.

4.9 Software Within a Larger System

A number of the projects were not simply software projects, but rather were larger systems in which a significant software element was embedded. A number of problems were noted in trying to estimate the cost of the software for these systems.

At the point at which the estimates were being made, there was frequently little or no architectural or system design Thus, it was not clear even how much software was to be included within the delivered system. Estimates were performed for the system as a whole with only minimal information as to the estimated cost of the software. Frequently, it was only after development was well underway (and budgets had been fixed) that all of the software items were identified and detailed cost estimates created for each of these items.

If software was identified as an item within the original estimate, it was not always clear what was included within this item. Often, software modules were not properly identified during the initial estimates, and it was not clear whether the line item “software” included training, documentation, etc.

5. Conclusions and Recommendations

There is no silver bullet. It is not possible to detail a cookbook solution to accurate software cost estimation. Estimation accuracy can be improved for the organizations that consistently or occasionally generate bad estimates; but the methods are not simple.

This chapter is divided into three sections. Section 5.briefly lists the conclusions of the study. Section5.2 contains recommendations for all organizations. Since organizations are so diverse, these recommendations are general in nature; it is not possible to provide specific recommendations that apply to all organizations. Section5.3gives more specific recommendations, which are applicable to DND procurement and maintenance of software.

5.1 Conclusions

Experience and informal analogy are the primary cost estimation methods. The vast majority of organizations relied on individuals' expertise and experience to arrive at cost estimates. Managers received little or no training in estimation. Estimators were expected to arrive at accurate estimates by relying on their knowledge of the software process used within the organization and recollections of their previous projects

Few organizations have sufficient historical data to be used meaningfully for cost estimation. With a few notable exceptions, organizations did not have information regarding past projects recorded in a manner that was useful and accessible to estimators. However, a number of organizations had recently implemented programs to gather and store these data, but it will be a few years before the impact of the data gathering on estimation accuracy can be determined.

Estimation cannot be improved without a well-defined and well-controlled software process. Organizations without a defined and controlled software process cannot achieve consistency in their software development. Without consistency in software development, consistently accurate estimates are not possible. Using the SEI CMM as a measure or software process consistency, a clear majority of the surveyed organizations were level 1 maturity.

Requirements creep is a major reason for cost overruns. It can be minimized, but cannot be eliminated. Changing requirements was the reason most often cited for cost overruns. Two conclusions are drawn. First, if cost estimates are to be accurate, the initial software requirements must be as complete and correct as possible. Second, for complex systems, it is impossible to generate requirements that are 100% complete and correct. Thus, one must accept the fact that complete accuracy for estimates of complex systems is not possible.

The impact of modifications to the software process are underestimated. If an organization modifies its software process in any way,

the cost of software development, at least in the short term, will increase. The amount of the increase is severely underestimated. The changes to software process we observed were the introduction of new tools into the development process and the use of 2167a to an organization that had not applied it previously.

The DND procurement process forces DND project managers and contractors into a situation where accurate cost estimates are difficult. A contractor who wishes to win a contract is often forced into the situation of submitting a bid lower than the initial estimated cost of building the system. During development, the contractor then looks for ways to extract more money from DND or to cut corners in areas of functionality, testing, reliability, etc. From a DND Project Manager's perspective, when budget costs start escalating, the pressure to deliver a system under budget is an incentive to accept an inferior product and to let the maintenance group worry about fixing it.

Parametric cost estimation models are rarely used as the primary cost estimation technique. With one exception, parametric models were not used as a primary means of cost estimation. A few organizations used parametric models as "sanity checks" on estimates. DND also used parametric models to arrive at gross estimates of manpower requirements for new maintenance centers.

Life cycle costing is ineffective in DND. A significant percentage of the cost of any major project is the maintenance of the system after delivery. DND does not appear to have any means of reliably estimating these costs or of trading off maintenance and development costs.

5.2 General Recommendations

The solution to improving estimation accuracy is not a high technology issue. No existing fancy tools, models, or methodologies can be brought to bear on the problem that by themselves, will have a significant impact. Rather, the problem is one of applying simple technologies, an effective software development process, and proper management and control to achieve a consistency in development, which allows more accurate cost estimation. Solutions to the cost estimation problem must address the issues in all of these areas or they will not be effective.

The major recommendations of this report are intended to provide mechanisms and techniques for a better understanding and control of software costs. Primarily they are intended to point out how software systems can be installed without any major surprises as to the final cost, although an overall reduction in software costs will also be a likely result.

The cost estimation problem varies considerably among organizations that do their estimation under very different constraints. The recommendations are general in nature and must be tailored to the individual organizations needs, depending on whether they are maintenance groups, procurement organizations, commercial developers, etc.

The following recommendations are based on two assumptions:

- There is significant room for improvement in the accuracy of cost estimates for software intensive systems.
- Although there is room to improve the level of accuracy of software cost estimates, there will continue to be a large margin of error; organizations must adapt to accept this fact.

5.2.1 Software Process Improvements

Improving software cost estimation accuracy must begin with a solid and effective software development process. An effective software process can be used to increase accuracy in cost estimation in a number of ways.

•*Formalizing when and how cost estimates and re-estimates are performed*. A critical aspect to estimation accuracy is to have a well-defined process that defines when and how cost estimates are performed. This fact is well recognized by process improvement organizations such as SEI, which require an organization to have a formal technique for cost estimation in place before moving beyond maturity level 1 [Paul-93]. The software process should define

- When cost estimates and re-estimates are performed.
- The process used to perform the estimates, including who performs the estimate and who has sign off authority on the estimate.
- What specifically is estimated, e.g., effort, loading, software size, etc.

•*Permit effective monitoring and control of software costs*. No cost estimate will be accurate without effective monitoring and control of software costs. If there is no effective technique for monitoring and controlling the project, there is an increased risk of the costs of the project escalating without management being able to recognize or identify the problem at a time when action can be taken to minimize the effect. Monitoring and control of a project must satisfy the following criteria.

- *Measured to a proper level of granularity*. Managers must be able to monitor a project at a sufficiently detailed level so that a complete understanding of the process and the product can be attained. This could involve for example, measuring each activity of a Work Breakdown Structure or measuring the process associated with every Software Change Request. Care must be taken, however, that data gathered is not so detailed that the expense of gathering the data impacts accuracy and the ability to analyze the data.
- *Objective measure of completeness*. Each WBS work item should have clearly identified output items and an objective means of determining the completeness of these items.

•*Analyzing problems reported during the development process*. Every reported problem with either a product or a process should be traced back to its cause.

This requires determining which Work Breakdown Structure activity, and which work item of that activity, was the cause of the problem. This is a prerequisite to determining whether a particular activity within the organization is a cause of the problem.

•*Control and minimize the effects of requirements creep*. As stated in Section 4.3 requirements creep was the single most often cited cause of inaccurate cost estimates. Any attempt to improve cost estimation must address this problem. Two approaches can be taken to the problem: make sure that the requirements are correct and immutable before development begins; change the development process to accept the fact that the initial requirements are wrong.

Much has been written over the years about the need for a thorough and complete requirements analysis and specification. This has been well documented and well argued, and we agree with it. What is perhaps less well understood is that regardless of the dedication and competence of the people involved, the initial requirements analysis inevitably produces a requirements definition that is incomplete and ambiguous and will be changed significantly throughout development. It is in fact not usually possible to get the requirements correct the first time. This has a number of implications upon how cost estimation is performed and interpreted as part of the software process.

- The development process must be something other than waterfall in order to allow for significant modification of the requirements during development.

- Management must recognize that cost estimates based on the initial requirements are wrong because the requirements are wrong. This means there must be a provision within the software process to re-estimate costs as requirements are changed. The re-estimation depends on the constraints under which the system is being developed. For example, if money is the main constraint, then the estimate of the functionality of the system must be modified to satisfy the cost constraints; if functionality must be maintained, then cost must be altered in order to satisfy the functionality. The worst solution (and one sometimes adopted) is to assume that both functionality and cost requirements are fixed; this results in irrational decisions being taken, such as reducing testing time when a project is over budget and trying to rationalize that this will have no effect on the product reliability.

5.2.2 Maintaining a Historical Database

Consistently accurate software cost estimates require that an organization have an accurate and objective record of previous development projects. For many organizations, only a minimal record keeping is undertaken, and often the data recorded is not in a form or of a type useful to software cost estimators. In the majority of cases, the only useful history maintained on projects is that which individuals can remember. For large projects and large organizations, relying on individuals' memories as a historical database is insufficient.

Organizations should maintain a database, which can be used as a basis for estimating costs of future projects. The database should include both *project* metrics (which describe the features of the system built) and *process* metrics, which describe features of the process used to build the system.

It is impossible to identify specific metrics that should be recorded and used by every organization; each organization and each situation are unique. However, metrics recorded for the purpose of improving cost estimation should be able to satisfy the following:

- Basic characteristics of the development process. To understand the context of the data gathered and to know whether they are applicable to other projects, the basic characteristics of the process must be recorded. This includes, for example, number of people involved in development, the development methodology used, and organizational structure.

- Actual cost of the system development. Unless the actual cost of the system development is known, it is impossible to determine the accuracy of the estimates.

- All estimates and re-estimates are recorded. To determine the accuracy of estimates, and the rate of convergence of the estimates to the actual cost, a complete record of all estimates must be maintained.

- The characteristics of the completed product. This includes the size measured in some suitable units (e.g., Source Lines of Code, Function Points), a description of the functionality of the system, classification of type of software, and any other information that characterizes the system. This information is required if any rigorous estimation by analogy is to be performed or if any costing models are to be developed.

Almost all organizations recorded some types of information about a project; many, however, were not able to apply this recorded information to cost estimation of future projects. To apply the information is a useful way, the following conditions must be met:

- The information must be stored so that it is readily available to an estimator. Information stored in raw paper files or not properly indexed will not be helpful.

- The information must be recorded at a level of detail useful to the estimator. Actual costs recorded for the project as a whole are of minimal use to an estimator who is performing cost estimates based on WBS activities or on the basis of software modules. The information recorded should be broken down in whichever of the following ways is most useful: WBS activity, module, or feature. Recording the cost data at the level of the WBS activity and module is critical, since this is how many of the cost estimates are performed.

For maintenance organizations, where estimation is performed on the basis of individual features, the data must be recorded on an individual feature basis.

5.2.3 Project Management

If an organization has not instituted a proper project management discipline, cost estimates, regardless of how carefully they have been constructed, may have little validity. Project managers must be aware at all times of the current costs of a project, cost overruns incurred, problems that have arisen as soon as they are identifiable, estimated cost to completion, and level of completion of the project. Without detailed and objective means of making all these project characteristics visible to the project manager, it is unlikely that accurate estimates can ever be made.

The key factors that a manager must be able to monitor are the following:

- Percentage completion. A manager requires an objective measure of the level of completion of a project. Subjective measure, such as “90% of the coding is done for module X”, are useless. Rather, there must be a clear and objective method for determining when a particular activity has completed. This could be, for example, a successful inspection by an inspection team or a successful test completed.

- Productivity. Productivity refers to the rate of progress of the project. There are many ways of measuring progress, e.g., rates of SLOC produced, rate of completion of tests, etc. No single method is sufficient.

- Problem recognition. To control costs, managers must recognize as early as possible during development when a problem is occurring. Various techniques can be applied. For example, analyzing problem reports and tracing the cause back to individual modules or WBS activities will help identify which processes or modules are causing problems to the project.

5.3 DND Recommendations

DND is involved in two roles: as a software procurer and as a software maintainer.

5.3.1 Cost Estimates from a Procurement Agency’s Perspective

The concerns of a procurement agency are somewhat different than the concerns of a development organization when it comes to software cost estimation. However, as described in Section 4.3 the procurement process itself often introduces problems into the software cost estimation process. The government and military procurement process forces contractors and procurers to try to agree on the requirements and cost of the system very early in the software development life cycle. As has been emphasized already, arriving at

firm requirements and cost estimates early in a project is not always possible. Just as the software process must be altered to recognize this fact, so must the procurement process. Possible approaches to this problem include

- Different procurement processes for standard versus innovative systems. With innovative systems, requirements are generally not known and costs cannot be accurately estimated. Different solutions, such as greater use of prototypes, phased development, and executable specifications, must be explored.

- More emphasis put on correct and detailed requirements. Although correct requirements cannot be produced until the end of a project (and sometimes not even then), every attempt must be made to arrive at a set of requirements as early as possible. The procurement process must then have provision for modifying these requirements and the corresponding cost estimate.

- Investigate and experiment with “Common Purpose Procurement”.

Cost estimates will never be accurate without a successful relationship between the procuring agency and the contractor. The most wonderful project plans submitted in a proposal are meaningless when the project breaks down into shouting matches between the contractor and the agency. A necessary condition is that the procuring agency have confidence in the capability and confidence of the contractor.

Procuring agencies can monitor a project in one of two ways: they can inspect and sign off on specific project milestones, such as the Preliminary Design Review and Critical Design Review; and they can monitor the contractors process to verify that the contractor is developing the system according to well accepted and reliable development techniques. Monitoring the project is what is currently done; for example, it is the philosophy of the 2167a standard. However, since milestone reviews by procuring agencies are not always effective (Section *4.6* , [Hump-89]), procuring agencies should explore further how they conduct these reviews and what specifically they are reviewing. Assuming that the contractor is technically competent (otherwise the project is doomed regardless of the reviewing process), the procuring agency reviews should focus on the following rather than on technical content:

- *Functionality*. Will the system meet the requirements of the procuring agency?

- *Process*. Has the contractor developed the system according to well-accepted software development process standards?

5.3.2 Life-Cycle Costing

Large software systems have a long life span. During the life span of a system, it is continually upgraded. The cost of developing these upgrades is often much greater than the initial development costs. Studies within the Department of

Defence in the U.S.A. indicate that maintenance costs can be three or four times the initial costs of the system.

Given the high percentage of development costs devoted to maintenance, in many cases it would be cost effective to increase development costs if there was a resulting benefit in reduced maintenance costs. Unfortunately, few organizations have the ability to perform such tradeoffs.

Managers are judged within an organization based on whether they keep their costs within their allocated budgets. If a technique can be applied during development to reduce long-term costs at the expense of increased development costs, it may be in the long-term interest of the organization to do so, but it may not be in the long-term interests of the manager; he will be given credit for minimizing development costs now, but will not necessarily be given credit for reducing maintenance costs 5 or 10 years into the future.

To understand better the implications of long-term software support, a number of issues should be further investigated by DND:

- What is the current ratio of maintenance costs to development costs? We are not aware of any studies done by DND to determine the cost of software maintenance for large-scale weapons systems.

- Full life-cycle costing to permit tradeoff of development cost for reduced maintenance costs.

- Research into re-engineering and reverse engineering. Many systems are delivered without adequate documentation. Without a clear idea of what is being maintained, it is difficult to arrive at accurate cost estimates. Reverse engineering and re-engineering systems will give maintenance units a better understanding of the architecture with which they are working, and as a result they will be able to arrive at more accurate cost estimates.

5.3.3 Maintaining Software Expertise Within DND

One observation that was made is that there is a difficulty within DND in building up a core knowledge of software expertise. As discussed in Section 4, the factors that lead to this problem are the following:

- Military rotation, which guarantees a large turnover of personnel for long-term projects.

- The infrequency with which large weapons systems are acquired.

- A lack of communication between different units within DND.

These three characteristics of DND make it difficult to build up and retain software expertise.

To build up the software expertise required and to transfer this expertise to the appropriate units within DND, it is recommended that DND investigate the

development of a group with the mandate of researching, developing, retaining, and disseminating software expertise throughout DND. Among the roles the group could perform would be the following:

- *Assist project managers*. The software expertise group would act as consultants to project managers and maintenance personnel to assist them in arriving at accurate cost estimates, improving development processes, evaluating bids, recommend appropriate data gathering and monitoring methods, and provide techniques and tools for analyzing data.

- *Data collection*. DND does a significant amount of software procurement, maintenance, and development. In each of these areas it is important to be able to study the process used in order to improve it and to determine which techniques are successful. The software expertise group should have the power and the mandate to collect information on the software development, procurement, and maintenance throughout DND.

- *Monitor and evaluate contractors*. The software group would be able to evaluate the capability of contractors. This could be done both through an accurate historical record of the performance of different contractors and knowledge on how to monitor the ongoing progress and process of a contractor during development.

- *Research on the development and procurement processes*. The software expertise group should be able to perform research, particularly relating to new development processes and new procurement processes.

- *Definition of development and procurement standards*. The software expertise group should be involved (along with other units) in the definition of development and procurement standards for software.

- *Promote transfer of knowledge between groups*. One observation we have made is the lack of interaction between groups within DND who perform similar functions. This was most striking with the weapons maintenance units, which were all performing similar functions in different ways but seemed to lack a means of transferring “lessons learned” between the units. The software expertise group should provide a mechanism for exchanging knowledge between these units.

To be effective, such a software expertise group must have a stable work force and not be subject to military rotation every few years.

To initiate the formation of such a group within DND, it is recommended that a pilot project be undertaken. The purpose of the pilot project would be to focus initially on data collection and transfer of knowledge between groups. The other roles listed above cannot be fully satisfied until the data are collected. Since DND is involved primarily with maintenance and procurement, it is suggested

that the pilot project include a small number of maintenance units and a small number of organizations initiating a procurement. The data to be collected during this pilot project would include:

- Cost estimates during the course of the project and actual costs.
- High-level architecture of the system being developed/maintained.
- Characteristics of the system, such as functionality and size.
- Changes or clarifications to functionality during the course of development.
- Costs associated with different development phases.

The pilot project would help determine what data to collect and how to collect the data.

5.3.4 Software within a System

There is ample anecdotal evidence that for major weapons systems, software is a significant percentage of the development cost, development problems, and maintenance costs. Despite this fact, there are generally few attempts made up front to identify all software components within a system and to create a strategy for developing and maintaining the software part of these systems.

One of the recommendations is to identify as early as possible all software items within the system and to identify clearly the cost and development process for these items. The software items should be a clearly identified item within the budget of a project. The budget item “software” should also clearly define the full extent of the software items and what is being included as software.

A development process and a WBS structure should also break out each major software item as a separate entity, identifying clearly the process to be used to develop the software.

Appendix A

Introduction

The Institute for Information Technology (IIT) within the National Research Council (NRC) Canada is currently conducting a study to investigate the process used to estimate the cost of developing and maintaining systems involving significant amounts of software. This study is commissioned by the Chief Research and Development (CRAD) within the Department of National Defence (DND). The objectives of this study are to determine the current state of the art in software cost estimation techniques in order to help identify methods and research directions for improving software cost estimation and control. As part of the study we wish to look at and compare software development in both commercial and military applications.

Our approach is to identify a set of development and maintenance projects which have been completed (or are close to completion), determine when and how the software costs were estimated, and then look at how the estimated costs relate to the actual costs. We are not limiting ourselves to a specific kind of system; we are interested in all systems which contain software. Nor are we limiting ourselves to a particular development group or organization. We will be looking at military and commercial systems, maintenance and development projects, embedded and information systems.

This document contains a set of questions and topics for discussion. It is intended as a starting point for discussion to allow us to gather the information about the projects from the appropriate people. If you have the time to provide us with any responses to these topics beforehand, it will allow us to be better prepared for the discussion.

Please note that we are gathering this information as NRC employees for a research project within the IIT, and the specific company information will not be available outside the Institute. Any reports resulting from this project will not contain specific references to companies or organizations within the body of the report other than a list of the organizations with whom we have discussed the issues.

This document, which originated out of a series of interviews with commercial software developers, DND personnel, and DND contractors, is continuously evolving as we learn more about the software cost estimation process currently in use. If you have any suggestions as to how to modify this document, and the questions and issues which it raises, your input will be greatly appreciated.

Forward any comments or questions to:

Mark Vigder
Institute for Information Technology
National Research Council Canada
Ottawa, Canada
K1A 0R6
Phone - (613)991-6972
Fax - (613)952-7151
email - vigder@iit.nrc.ca

or

Anatol Kark
Institute for Information Technology
National Research Council Canada
Ottawa, Canada
K1A 0R6
Phone - (613)991-6973
Fax - (613)952-7151
email - kark@iit.nrc.ca

Topics for Discussion

These topics and questions are intended to gather information about a specific project which has been undertaken by your organization. During our discussion, we will also be interested in determining how similar (or different) the development process of this project is in relation to other projects in which you or your organization have been involved.

The questions cover a wide range of topics and not all questions will be applicable to all projects nor to all situations. If a question does not apply to your situation, please note that the question is “not applicable”.

We also recognize that in many cases the information which we are requesting will not be available, either because it is too difficult to gather the information, or because it is information which you do not wish to release to us. Answer such questions as “not available” or “cannot be released”.

This study is interested primarily in cost estimates relating to effort and duration :

- Effort. The manpower required, measured in some unit such as “person-years” or “person-months”.
- Duration. The length of the project, measured in units such as months or years.

Where possible, discuss the cost estimates in terms of the estimated effort and estimated duration of the project.

1. Project Properties

These questions are intended to determine the type of project which is being investigated.

1-1. Was the project developed:

a) Primarily within your organization.

b) Primarily contracted out.

1-2. Describe the project, e.g., what was being built, what was expected to be delivered at the completion of the project, etc.

1-3. What type of system was developed, e.g., embedded system, information system, etc.?

1-4. Was the project:

a) Primarily software

b) System (involving significant combined hardware and software development).

1-5. Was the project:

a) A new system.

b) A major upgrade. A currently existing system was upgraded, requiring extensive amounts of new software to be developed.

c) A minor upgrade. A currently existing system was upgraded. The upgrade was done primarily by modifying existing software.

2. For Contracted Projects.

For projects which were primarily contracted to other developers:

2-1. What was the contracted price?

2-2. If you were the organization managing the acquisition, how close was the contracted price to the expected bids?

2-3. What was the final cost paid to the contractor?

2-4. If the price paid was different from the contracted price, what is the reason for the discrepancy:

a) New requirements determined by contracting organization.

b) Initial requirements were vague.

c) Contractor underestimated work required.

d) Technical problems.

e) Contractor suggested enhancements.

f) Other (Please explain).

2-5. If you were the contractor, what were the non-contractual costs incurred which were absorbed by the company?

3. The Cost Estimates

These questions are intended to determine what type of estimates were being performed, and what the goals of the estimates were.

3-1. During the course of the project, identify when formal cost estimates occurred. For each estimate, identify:

a) At what stage of the project lifecycle it occurred.

b) What was being estimated (e.g., total effort, duration, manpower buildup, etc.)

c) For what purpose was the estimate being done, e.g., obtain budgetary approval, determine manpower requirements, etc.

3-2. Were re-estimates of costs performed during development?

3-3. If the answer to question 3-2 is yes, then identify when the re-estimating procedures were invoked (see list below). If possible, indicate when (in the development life cycle) the re-estimates occurred, and the result of the re-estimates

a) Informal re-estimates performed during development.

b) Formal re-estimates performed at pre-defined milestones.

c) An amendment changed the system being built and a re-estimate was required.

d) Other (please explain).

3-4. How accurate were the cost estimates? In your opinion, what were the reasons for the discrepancy between the actual cost and the estimates.

3-5. The estimation problem can be classified into three main categories:

a) New Project Development. A new project (or major upgrade) is being developed. The estimation requirement is to determine the cost of the development.

b) Ongoing maintenance requirements. A new system is being implemented and it is necessary set up an ongoing maintenance and support unit. An estimate is being made of the ongoing resources required to maintain the software.

c) System upgrade. A set of System Change Requests (SCR) have been assembled, and the requirements for a new version are being defined. A fixed set of resources are available for implementing the upgrade. The problem is to estimate which SCRs can be included in the new version of the software, given the constraints of the fixed resources.

Is the estimation problem you were faced with one of:

a) New product development.

b) Ongoing maintenance requirements.

c) System upgrade.

d) Other (please explain).

3-6. What items were included in each of the estimates, e.g.,

a) Software

b) Hardware

c) Number of delivered units

d) Testing

e) Documentation

f) Training

g) Other (please explain)

3-7. Life cycle costing is an approach to cost estimation which looks not only at the cost of developing and delivering a system, but also the cost of maintenance during the life of the system. Do you consider life-cycle costing preferable to the costing of development and maintenance as separate items and estimated at separate times? Why (or why not)?

3-8. Should software be included as a special line item within an overall system project budget? Why? When should the estimates be done?

4. People involved in estimation.

Cost estimation for larger projects generally involve a number of people. These questions are to determine who was involved in the estimation, and the role which they played.

4-1. How many and what was the positions of the people involved in the cost estimation process?

4-2. How were the individuals involved in the cost estimation included as part of the process:

a) Informal consultation, e.g., person in charge of estimation walked into a managers office to solicit an opinion.

b) Formal consultation with project staff, e.g., a meeting was organized for the purpose of performing the cost estimate.

c) Sign off authority.

d) Other (explain).

4-3. Did the people involved in the estimation process have any previous experience in software cost estimation before they were called upon to estimate for this project? If yes, was their experience useful in arriving at accurate estimates?

4-4. Was any training in project management provided? Is such training required? Did the training include cost estimation?

4-5. Do you think the training did (or would) improve your ability to arrive at accurate cost estimates?

4-6. Would an outside group trained in cost estimation have been a useful resource to you?

5. Estimation process

The cost estimation process is the set of steps which are performed in order to arrive at a cost estimate.

5-1. For each of the cost estimates which were performed during the project, describe the process your organization went through in order to arrive at the estimate.

5-2. What were the inputs which were used for each of the estimates:

a) An incomplete set of requirements.

b) A detailed set of requirements.

c) Details of the user interface, including for example, prototype screens.

d) Architectural design.

e) System engineering.

f) Other.

5-3. What was the level of accuracy which the estimator was trying to achieve for each of the estimates?

5-4. How would you classify the software estimation process which was used for each of the estimates? If more than one method was used, list all methods in order of significance.

a) Ad hoc. Cannot be categorized.

b) Informal analogy and rules-of-thumb. Based on experience, the estimator arrives at an estimate by informally comparing this project with previous projects, and applies a number of rules-of-thumb to arrive at an estimate.

c) Formal analogy. A database is maintained of previous projects. New projects are estimated by comparing new projects with previous projects in a formal and rigorous way, identifying differences and similarities and basing the cost estimate on the similarities and differences with previous projects.

d) Formal model. A formal model (such as COCOMO) is used to provide a cost estimation model for the project.

e) Other (explain).

5-5. Was there a formal procedure used during estimation which tried to identify and assess the risks which could jeopardize the successful completion of the project?

5-6. How were the cost estimates done for each estimate performed:

a) Estimate was done at the system level, for the complete development process.

b) The development process was broken down into tasks at the system level (e.g., requirements analysis, design, coding, testing, etc.). An estimate was done for each task, and these estimates were combined into an overall system estimate.

c) An architectural design was performed decomposing the system into subsystems and components. The cost of each subsystem was estimated and these were combined into an overall system estimate.

d) For upgrades, the cost of the changes required were estimated relative to the current system architecture.

e) Other.

5-7. Often estimators are constrained to produce estimates based on available resources, e.g., limited dollars, limited manpower, limited duration. Were there constraints put on the estimation process? If yes what were the constraints? How restrictive were the constraints (e.g., minor, major, cast in stone)?

5-8. What effect did the constraints have on the functionality of the system and the reality of the estimates? For example, was a fixed amount of resources available and was the functionality of the system defined by what could be built with these resources?

5-9. In your opinion, how could the software cost estimation process be improved in a practical way?

6. Tracking costs

Improving estimates and controlling costs requires that initial estimates are made of the development process, and that the actual values gathered during the development process are compared to the original estimates.

6-1. Were profiles maintained of actual versus estimated cost over time?

6-2. Were staffing profiles maintained of actual versus planned staffing?

6-3. Were profiles maintained of actual versus planned software units designed, tested and integrated over time?

6-4. What are the mechanisms in place for tracking the development process? Are any of these mechanisms automated?

7. Metrics

Metrics are measures which quantify properties of a system. Product metrics measure properties of the delivered system (e.g., number of lines of code, number of configuration items, etc.). Process metrics measure properties of the process which is used to build the system (e.g., number of faults found at each phase of the development, development effort, etc.). Metrics provide a means of

capturing historical data about a project. This historical data can be used to assist in cost estimation and control for future projects.

7-1. Is any information regarding past projects recorded and maintained in a formal way to be kept in a historical database by your department or organization?

7-2. If a historical database is maintained, what information is recorded and what is the intended use of the database?

7-3. Were any process or product metrics maintained during development? Examples of metrics include (but are not limited to) the following:

- Process metrics

  a) Project development effort (per phase).

  b) Project development time (per phase).

  c) Number of changes (classified by fault correction, requirements change, etc.).

  d) Slippage. Difference between estimated and actual values, e.g., effort, duration, etc.

  e) Staffing profiles (e.g., experience and knowledge of staff -- subjective).

- Product metrics

  f) System Size.

  g) Number of Document Pages (per phase)

  h) Source code size.

  i) Structure metric, e.g., complexity of data structures, number of modules, etc.

  j) Project type (defines external constraints, i.e., required reliability, system complexity, interconnections with other existing systems, embedded software, commercial software products, etc.)

7-4. Did the process of gathering the metrics tend to distract from the development process?

7-5. If a historical database of projects is to be maintained with the goal of better estimating and controlling development costs, what information should be recorded as part of the database?

8. Cost drivers

There exist a number of cost drivers for projects involving significant amounts of software development. Estimating and controlling costs can be improved by better understanding these cost drivers.

8-1. Which of the following do you consider to be significant cost drivers for software development:

a) Manpower turnover.

b) Requirements changes.

c) External system variations (e.g., OS keeps changing).

d) New technologies raise user expectations.

e) System complexity.

f) Configuration management.

g) Testing.

h) System size

i) Distribution of software to users.

j) Other

Bibliography

[Albr-83] A.J. Albrecht and J.E. Gaffney Jr., “Software Function, Source Lines of Code, and Development Effort Prediction” in *IEEE Transactions on Software Engineering, SE-9, 6,* 639-648

[Boeh-81] Barry Boehm, *Software Engineering Economics*, Englewood Cliffs N.J., Prentice-Hall Inc. 1981

[Broo-87] F. Brooks, “No Silver Bullet: Essence and Accidents of Software Engineering”, in *IEEE Computer*, **20**(4) pp 10-19

[Cowd-89] A.J.C. Cowderoy and J.O. Jenkins, “New Trends in Cost-Estimation” in *Measurement for Software Control and Assurance*, Elsevier Applied Science, New York, 1989

[DeMa-82] R. DeMarco, *Controlling Software Projects: Management, Measurement, and Estimation*, Yourdon Press, Englewood Cliffs, NJ., 1982

[Hump-89] W.S. Humphrey, *Managing the Software Process*, Addison-Wesley, Don Mills Ont., 1989

[Putn-92] L.H. Putnam and W. Myers, *Measures for Excellence, Reliable Software on Time, Within Budget*, Yourdon Press, Englewood Cliffs N.J., 1992

[Paul-93] M.C. Paulk, B. Curtis, M.B. Chrissis, C.V. Weber, *Capability Maturity Model for Software, Version 1.1*, Software Engineering Institute, CMU/SEI-93-TR-24, February, 1993.

[Vigd-94] M.R. Vigder, A.W. Kark, *A Comparison of Project Management for Hardware and Software System Development*, NRC Report Number NRC37117