A Knowledge-Based System to Predict Software Development

Jason D. Thompson
University Undergraduate Fellow, 1995-1996
Texas A&M University
Department of Computer Science

APPROVED

Fellows Advisor [illegible]

Honors Director [illegible]

A KNOWLEDGE-BASED SYSTEM TO PREDICT SOFTWARE DEVELOPMENT.

Jason D. Thompson (Dr. Dick B. Simmons), Computer Science, Texas A&M University.

Effective software development is critical in order to corporations to survive in today's software markets. Not only do they need to produce quality software, but they need to be able to do it on a limited budget and schedule. This problem is exacerbated by ineffective means of measuring the software process. Models have been developed to help quantify the intangible software process, but taken alone they are often ineffective. Manager's estimates need to be modified by both a model and historical project analysis in order for them to be meaningful. I have developed a software system that attempts to help managers make meaningful predictions about the software process in order to make better software. It takes manager estimates on program size and type, and uses previous projects to help the COCOMO model create a better estimate than the model could come up with by itself.

Table of Contents

List of Figures

Introduction

Since the first person opened a shop in some small village and tried to sell his goods, competition has been the rule of the game. For every person that can produce some good at some price, there is someone else that is trying to make the same product better and for less money. This simple rule of the marketplace forces businesses to constantly improve their production facilities. This includes getting the best equipment as well as the newest ideas for management. The industry of software engineering is no exception. In an industry where late products are delivered well over the original budget, any type of an edge one business can achieve over another can be critical. In other engineering fields, models and formulas have been developed that allow accurate prediction of time and cost. However, because of the extremely fast paced development of faster computers, the software engineering community has been unable to keep up. Therefore, any model that could accurately predict cost and schedule before a project has been undertaken would greatly help developers in the software industry. The COCOMO model is one such model that attempts to predict the amount of effort that a project will require. The manager can take this value and use it to determine staffing needs. Once staffing has been computed, the cost and duration of the project would follow. The COCOMO model can be an extremely accurate model that can be used to make some accurate predictions, especially once it has been tailored to the company in which it will be used. Before discussing one possible implementation of the COCOMO model in a computer program, this paper will turn its discussion toward the origins of the need for such a model.

Background of the Problem

Origins of Software Engineering

The term "software engineering" was first introduce in the late 1960s. It was coined at a conference that was organized in order to discuss what was being call the "software crisis." The crisis was simple; the creation of new technology made possible the development of complex projects that would require large software systems. However, early attempts at developing these large software systems showed that the existing methods for development were insufficient to handle the new demands. Projects were often years late, way over projected costs, unreliable, difficult to maintain, and performed poorly. New techniques were obviously required (Sommerville 3). The phrasing of this new term was interesting. Because it contained the word 'engineering,' developers looked at the software process as they would look at other engineering processes. This spawned the creation of numerous models to help organize the process. The idea of a model to reflect the software development process was readily accepted by managers as a means of making the operation more visible (Sommerville 5). These attempts to apply some of the tactics of other engineering disciplines to the software production industry have caused the creation of more advanced technology such as better compilers and computers (Abdel-Hamid 1426). So while better techniques have been developed because of the creation of this new phrase, software engineering is not an easy concept to define. It is an extremely complex issue and there are many different definitions for it. However, each of these definitions has similar components. Software engineering is concerned with software built by teams and not individuals. The teams use engineering principles while they develop the software, and the projects have both technical and non-technical aspects. Software engineers not only need to be able to communicate in computer languages, but they also need to be able to communicate ideas orally and in written correspondence. They need to understand the importance of good management and user involvement in the development process (Sommerville 2). All of these ideas are very different from the methods for developing software in earlier years. One last point that needs to be made concerning software engineering is the use of the term 'software.' When discussing software engineering, the term software goes far beyond the computer programs that make up the application. Instead, the term software refers to the documentation necessary to install, use, develop, and maintain the program. For large systems, the effort in developing these aspects of the project may be as great as the effort to complete the coding aspect (Sommerville 2). Obviously, the creation of this new branch of computer science has revolutionized the software industry by changing the way that developers think about the process.

Problems with Software Management

While a new branch of computer science was created at the "software crisis" conference, a solution to the crisis was never found. The software industry has continued to face its share of troubles as it has grown. The record shows that it is plagued with cost overruns, late deliveries, poor reliability and user dissatisfaction (Abdel-Hamid 1426). One of the key problems is management. Managing and overseeing large software projects is extremely difficult. Today, 50-70% of software projects are still late, over budget, and full of defects. Several attempts have been made to overcome these problems, but few have been successful (Putnam 105). So while over the last 20 years, there have been improvements in software engineering techniques, the industry is still in a crisis. The reason is that the demand for software is increasing at a faster rate than the techniques used to produce it. More people are being brought into the industry and many are still repeating mistakes make in the 1970s (Sommerville 3). The problem must lie with the people involved since the equipment has increased at a rate much higher than anyone could have hoped. So while the tools and machines have been improving over this time, management skills have not kept pace. Instilling new management techniques and discipline into workers has not increased as quickly as technical improvements (Putnam 106).

The reason for this discrepancy is simple. Everyone has been focusing on the process itself and how to make the equipment better. Managerial techniques research has drawn much less attention. Authors have speculated that it is due to the fact that computer scientists believe that management is not their concern. Conversely, management professionals believe that computer scientists should have to take care of the problem (Abdel-Hamid 1426). This separation within the field could account for the difficulties encountered while producing software systems. A chief concern is that software engineers still lack a basic understanding of the software development process. Without this understanding, making any gains in management is unlikely (Abdel-Hamid 1426). Managers are not completely in the dark, however. They would like to deliver products on-time, within the budget, and with few defects. However, it is impossible for them to accurately steer the project in the right direction because there are no accurate ways for them to measure where the project is, at a given point in time. They have no decent metrics that can tell them when the project is going astray. Thus, they have no way to know when to initiate corrective action until it is too late (Putnam 105).

This difficulty in quantifying software is no easy matter to resolve. Managers like to consider such concepts as quality and productivity when developing a product. However, these concepts are not so easy to measure when discussing computer software. Productivity is difficult to measure in software projects. Since there are so many factors that help determine productivity, it is almost impossible to know how productive a software team is, at a given point in time (Abdel-Hamid 1431). The solution is intuitive even if the way to achieve it is not. The fields of software engineering and management need to be made more quantitative in order for them to reach the full level of maturity that other disciplines have reached (Putnam 105). However, this level of quantification has not yet been reached and so software managers often have no way of gauging how far along a project is since the software process is intangible (Abdel-Hamid 1427).

One other problem that managers face is that they have an overly simple understanding of the software process. The model of this process can be seen in Figure-1. Stage 1 of the model represents the resources that can be applied to the project. As these resources are used to accomplish work (2), it is reported to some project control system (3). These reports are analyzed to predict completion time (4). If these results differ from the scheduled completion data (5) then resource changes are made(6) and the process starts all over again (Abdel-Hamid 1427). Unfortunately, things do no work out so nicely.

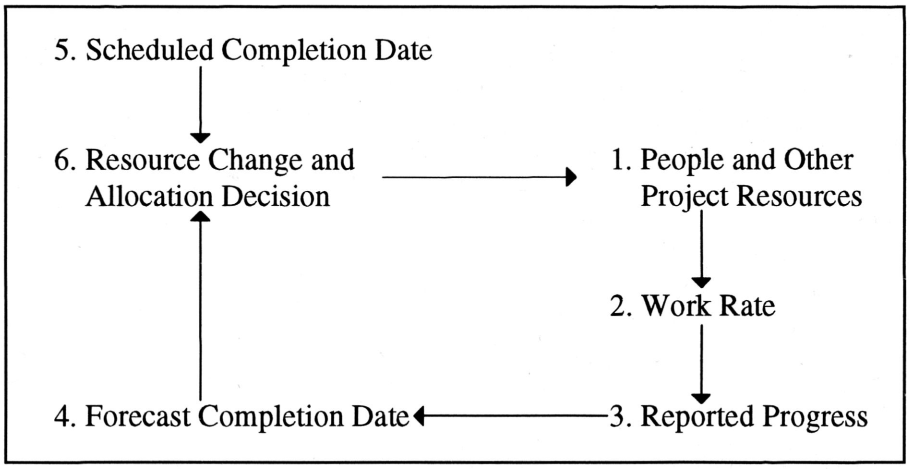

5. Scheduled Completion Date
6. Resource Change and Allocation Decision
1. People and Other Project Resources
2. Work Rate
3. Reported Progress
4. Forecast Completion Date

Figure-1: A Model of Software Project Management
Abdel-Hamid, Tarek, and Stuart E. Madnick. "Lessons Learned from Modeling the Dynamics of Software Development." *Communications of the ACM.* December, 1989. p.1427

In actuality, the software development process has more complex interrelationships that interact in non-linear ways. Such a simple model fails to take into account all necessary information (Abdel-Hamid 1427). Additionally, the simple model depicted in Figure-1 fails to take into account scheduling pressure on developers. When a deadline approaches, developers will often put in longer hours to get everything finished. This often results in a higher error rate. Therefore, more rework will be required later in the cycle (Abdel-Hamid 1427). As can obviously be seen from the above discussion, the work of a software manager is no easy task.

The Goal - Producing Good Software

While the task of a software manager is extremely difficult, their goal is a simple one: to produce quality software that people will use. Good software is easy to define. It should possess four key attributes. First, it should be maintainable. Software products are often used for long periods of time which will cause them to require regular changes.

Therefore, the code should be written in such a way so as to make updating it easy and inexpensive. Second, the software should be reliable. This means that the software should perform as the user expects and should not fail more than is acceptable. Next, the software should be efficient. This simply means that the product should not waste system resources. Finally, the software should provide an appropriate interface with the user. A lot of software is not used because it does not provide a good interface to the user (Sommerville 3-4). The goals are simple to state, but hard to achieve. However, companies that focus on improving the development process through education of their employees, instillment of management techniques to measure and control the process, and investment of money do improve the quality of their software products. They produce products faster, cheaper, and with fewer people. However, the commitment to the new process must be long-term (Putnam 106). The production of higher quality software in a shorter period of time has obvious benefits. Software development is an extremely expensive process. Precise figures are hard to gather, however it has been estimated that in 1985, worldwide software costs were greater than $140 billion. These costs have been growing at a rate of about 12% per year. By 1995, this would put the cost at over $445 billion. Therefore, it is obvious that even small improvements in the development process would yield huge results (Sommerville 2). Trying to achieve these goals is something that every software development team has tried to do.

Tools to Reach the Goal - Models

Since reaching these goals is the objective behind every software project, tools have been created to make this task easier. One of these tools, the model, attempts to quantify and describe the software life cycle. These software process models are important for the guidance that they provide. They detail the order in which the steps of a software development process should be carried out. Many software projects have failed because they did not follow the development process in the correct order (Boehm 61). One of the primary functions of a software model is to determine the order of the stages in software development and evolution. It also needs to establish the transition criteria for movement from one stage to the next. Thus a model can be thought to answer two questions: What will we do next and how long will we continue to do it? (Boehm 61). One of the first software life cycle models created was called the code-and-fix model. This model was the most basic model used in the early days of software development. It consisted of two basic steps: write some code and then fix the problems with the code. Thus, all thought about the requirements of the system, how best to design it, how best to test it, etc. came after the program had been written. This model had three problems. First, after a number of fixes had been implemented into the system, the code became poorly structured, making future fixes even more expensive. This underscored the need for a design stage implemented prior to coding. Second, even well structured programs often failed to meet users' needs. This caused the projects to be rejected or forced the need for expensive rework. This underscored the need for a requirements stage prior to coding. Finally, problems with the code were expensive to fix because of poor plans for testing. This made it clear that a specific stage for testing with clear guidelines needed to be established (Boehm 61-63). The next model, the stagewise model, was created in an attempt to overcome these problems. This model was developed by large software companies as early as 1956 in response to the shortcomings of the code-and-fix model. This model established stages of software development and said that code development should proceed from one stage to the next. The stages were: operational planning, operational specifications, coding specifications, coding, parameter testing, assembly testing, shakedown, and system evaluation (Boehm 63). While this model was a definite improvement over earlier models, it still had some problems.

The Waterfall Model

The solution was the creation of the most widely used model since its inception, the waterfall model. This model consists of five stages as seen in Figure-2. The first stage

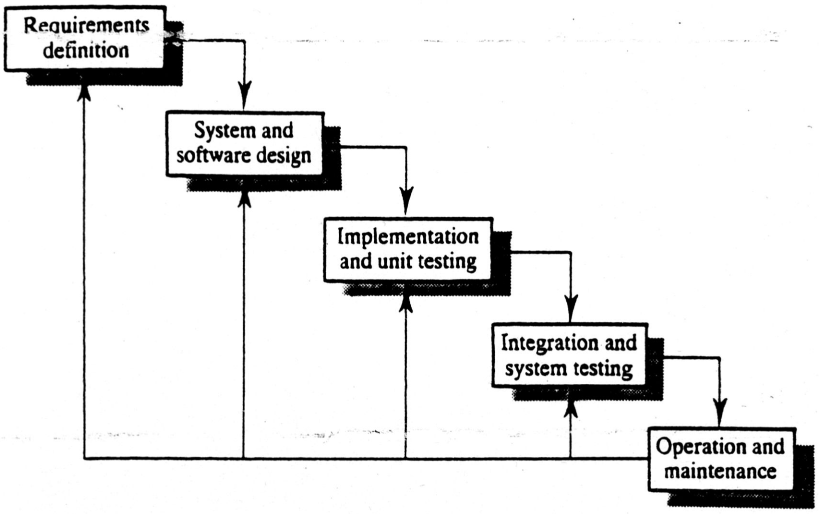

Requirements definition
System and software design
Implementation and unit testing
Integration and system testing
Operation and maintenance

Figure-2: The Waterfall Model
Sommerville, Ian. *Software Engineering*. England: Addison-Wesley Publishing Company, 1992. p. 9.

consists of requirements analysis and definition. During this stage the "system's services, constraints, and goals are established by consultation with system users" (Sommerville 7). The second stage is the system and software design stage. The system's tasks are classified as hardware or software considerations. The software factors are then represented in a form that is more easily converted into one or more executable programs. The third stage is the implementation and unit testing stage. This stage has each of the program units implemented and tested individually. Once all of the units work independently, they are placed together in stage four, the integration and system testing stage. This stage tests to make sure all units work as well together as they did separately. Finally, the operation and maintenance stage is reached. Maintenance involves making changes to the program over time and correcting errors that were previously undiscovered (Sommerville 7). Each of these stages has been outlined so as to appear separate from one another. However, these stages often overlap and feed each other information. Indeed, there are frequent iterations between stages (Sommerville 7). The system testing stage represents the final validation stage in the process. At this time, the user must be convinced that the system that has been implemented meets his needs. However, this is not to say that the system should not be validated at each stage of the process. Instead, this stage is simply the ultimate realization of this process (Sommerville 7-8). During the operation and maintenance stage, information is fed into earlier parts of the life cycle. Errors and new functionality come to light and the system must be reworked. This maintenance may result in changes in requirements, design, or implementation which could result in more testing. Therefore, the entire process could be repeated once the maintenance stage has been entered (Sommerville 8).

This waterfall model was created in 1970, and it was simply a modification of the stagewise model. It had two improvements over this earlier model. First, it recognized feedback loops between stages. It established guidelines for movement backwards, in order to avoid the more expensive problem of having to move back several stages. Second, it incorporated the concept of building a prototype during the life cycle. This model helped to eliminate many of the difficulties encountered by software developers. However, this model is not perfect. One of the problems with this model is its reliance on documents. In the early design stages, it emphasizes the completion of elaborate documents. This approach is well suited to some software projects, but not to others. This document-driven approach has led to the failure of some software projects by forcing them to complete the stages in the wrong order (Boehm 63). This iteration concept has created some problems as well. Frequent iteration makes it difficult for management to establish checkpoints to be used for planning and reporting. Therefore, managers tend to freeze parts of the development after a small number of iterations have been completed. Problems are either left for later, ignored, or worked around. The premature freezing of a stage could result in a system that does not completely satisfy the user and one that is poorly designed (Sommerville 7). However, there is no other obvious solution to this problem.

Because of the intangibility of software, managers cannot simply look at the program to determine the status of the project. This is different from other engineering disciplines. Therefore, management prefers models that give them factors to measure at each stage. They prefer a model that frequently creates visible documents and reports. The waterfall model is well suited to be applied to this delivery-model approach. Table-1 shows one possible way of dividing the waterfall model up into deliverables (Sommerville 12-13). Another way in which the waterfall model is well suited to the needs of management is that it allows an easy breakdown of costs. The cost of each stage can be computed and used to predict future projects' costs. One of the aims of software engineering is to reduce software cost. Costs could be established for each stage of the model. However, costs can vary dramatically for different applications. It is often difficult to get a good understanding of software costs because commercial companies are unwilling to publish their information. Some guidelines have been established. Table-2 shows the costs of some projects, broken down by stage (Sommerville 9). This information can be very useful to management.

Activity	Output Documents
Requirements Analysis	Feasibility Study Outline Requirements
Requirements Definition	Requirements Specification
System Specification	Functional Specification Acceptance Testing Specification Draft User Manual
Architectural Design	Design Architecture Specification System Test Specification
Interface Design	Interface Specification Integration Test Specification
Detailed Design	Design Specification Unit Test Specification
Coding	Program Code
Unit Testing	Unit Test Result Report
Module Testing	Mode Test Result Report
Integration Testing	Integration Test Report Final User Manual
System Testing	System Test Report
Acceptance Testing	Final System

Table-1: Documents From the Waterfall Model
Sommerville, Ian. *Software Engineering*. England: Addison-Wesley Publishing Company, 1992. p. 13.

System Type	Phase Costs %		
	Requirements/Design	Implementation	Testing
Command and control systems	46	20	34
Spaceborne systems	34	20	46
Operating systems	33	17	50
Scientific systems	44	26	30
Business systems	44	28	28

Table-2: Life Cycle Cost Distribution
Sommerville, Ian. *Software Engineering*. England: Addison-Wesley Publishing Company, 1992. p. 9.

Problems with the Waterfall Model

While the waterfall model overcomes many of the problems demonstrated by earlier models, it is far from perfect. It has faced a great deal of criticism. Part of the problem is that it fails to recognize the importance of iteration by forcing premature freezing of stages. Another related problem with the freezing of early stages is that the software does not meet the user's needs. Other models have been developed to meet these needs, but none is as accepted as the waterfall model (Sommerville 10). One such model is the evolutionary model. The problems with a document-driven approach like the waterfall model led to the development of an evolutionary model. Its "stages consist of expanding increments of an operational software product, with the directions of evolution being determined by operational experience" (Boehm 63). It allows the user to rapidly see a product and make changes in it. Thus it is well suited to fourth generation languages. However, it also has some problems. It is very similar to the code-and-fix approach of earlier days and leads to unstructured code (Boehm 63). Document-oriented approaches have several other drawbacks. First, management demands documents produced at specified intervals. Since these intervals may not correspond to actual stage completion, artificial documents could be produced. Second, since the documents need to be approved, process iteration is constrained since costs with approval are high. Inelegant solutions to problems are often used to avoid the hassle of iteration. Third, the use of the documents from one stage for the next stage creates the flawed impression of a linear system. Fourth, the time to review a document is high and so the transition between stages is often not very smooth. Finally, some problems cannot be solved with this method which does not allow much freedom for adaptation (Sommerville 14). One model that was developed to overcome these problems is the spiral model. However, this composite model is no where near as used as the waterfall model.

The Spiral Model

In 1987, the Defense Science Board Task Force Report on Military Software said that traditional software process models were preventing the use of more effective approaches to software development such as software reuse and prototyping (Boehm 61). A new model was obviously needed. The spiral model developed out of the waterfall model in attempt to refine this earlier model so as to apply large software projects. The radial dimensions seen in Figure-3 represent the cumulative cost incurred to date. The

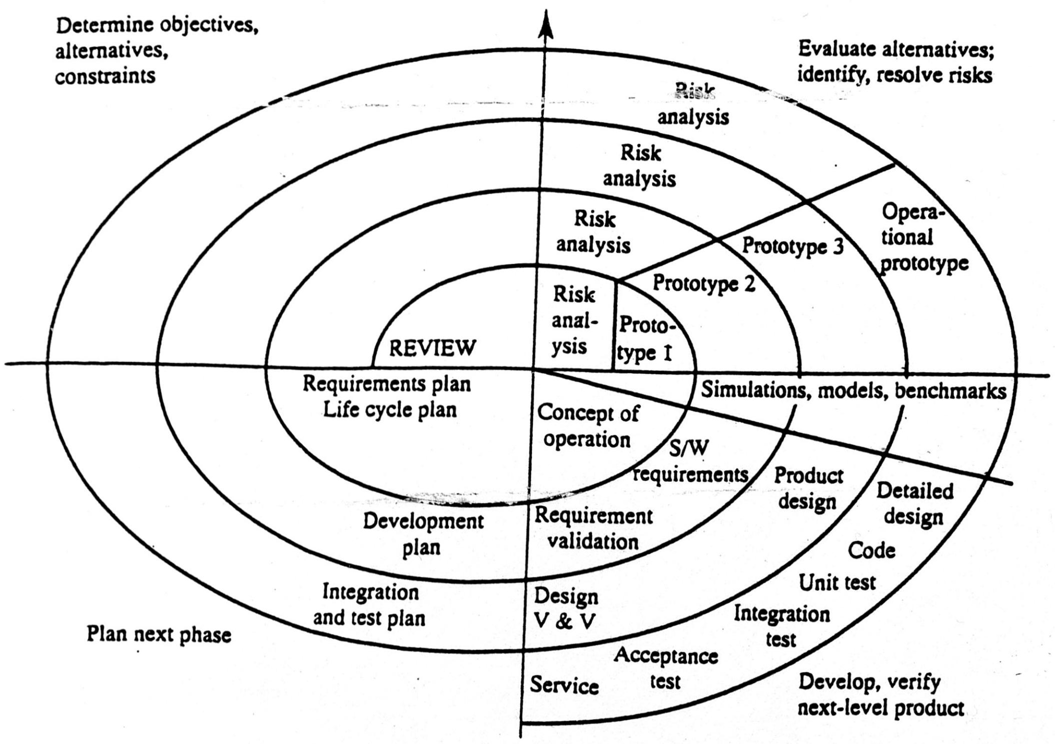


Figure-3: Boehm's Spiral Model
Sommerville, Ian. *Software Engineering.* England: Addison-Wesley Publishing Company, 1992. p. 15.

angular dimension represents the progress made in completing each cycle of the model. Each cycle begins with several considerations. First, the objectives of the cycle are outlined. Next, the alternative means of implementing the cycle are discussed. Finally, the constraints imposed on the application of the alternatives is considered. Also, the risk of the system is measured in order to determine how well it would meet users' needs. Once risk is within an acceptable level, the model would proceed like the waterfall model until project completion. If risk is too high, then another cycle is undertaken. This risk-driven approach is very helpful in meeting users' needs (Boehm 64-65). The spiral model creates a risk-driven approach to the software process rather than relying on documents or source code to drive it (Boehm 61). Risk measurement stated simply is an attempt to determine what could go wrong in the upcoming cycle. At each stage a new model can be adopted for the next stage since one model might not be good for all stages (Sommerville 15-18).

Making Estimates About the Software Life Cycle

Current Methods Used to Make Estimates

Making estimates about the software project before it has even begun is very important to software managers. These estimates can be derived using a variety of methods. When planning a project, the manager will first consider constraints on the project. These constraints include the required delivery date, the staff available, and the budget. The manager will also make estimates about such things as project size and structure. Then he will define the deliverables and milestones. Then a schedule is constructed. After a few weeks, the progress is measured and the schedule is revised. As more information becomes available, the manager will revise his initial estimates and make them better and more accurate (Sommerville 496). Estimation techniques can require a number of different components. First, there could be an historical database that includes previous project data. Second, there could be an estimation model that would predict events based upon a software model. Next, there could be an estimation process that would run through a series of algorithms, modifying the initial estimate. On top of this process could be a tool that would allow the user to view the database and graphically see the estimates. Finally, a report would be created discussing the results (Lehder 12-13). There are several different types of projects that can be considered when making estimates. First, completed prototypes of the system can be used to make estimates about the entire system. Second, similar software projects can be considered in order to make estimates. Third, similar software projects developed by different groups using different tools could be considered (Belova 950).

Now that a general idea about how estimates can be made has been achieved, a more thorough understanding is necessary. There are three stages to the estimate process and management must pass through all three stages for the estimates to be meaningful. First, there are primary estimates very early in the process in order to estimate effort and schedule. Once part of the project has been completed, refined estimates can be made. Once sufficiently into the life cycle, current estimates can be taken at any time to judge where things are in development (Belova 950). Managers often use a model such as the COCOMO model to create an initial estimate. Then they pad their estimate with a substantial buffer since they do not want to look bad. This makes the estimates meaningless and thus useless (Abdel-Hamid 1433). Metrics are very valuable in the software process. They can be used to develop quality models such as the COCOMO model. They can also provide the designer with valuable information concerning the characteristics of the software system (Khoshgoftaar 979).

Metrics can be critical to making accurate predictions. There are three different types of metrics. Code metrics are those that measure an attribute like length to number of tokens. Structure metrics try to determine the connectivity of program parts. Hybrid metrics combine these two (Henry 37). Predictions can be made at the design stage by considering these metrics in light of the design specification. Code metrics have been found to depend strongly on the specification's refinement level. However, structure metrics are independent of this. In this case, a highly refined specification would contain specifications that were very code like. Low level specifications appear to be natural language (Henry 39). Therefore, structure metrics required only low level refinement of specifications in order to be effective. However, code metrics require at least a moderately high level of refinement (Henry 41).

When estimating, developers often use a best-fit line to make predictions from existing data points. However, there are many different approaches that can be used to determine the best line that can be drawn through a series of data points (Khoshgoftaar 979). When using metrics to determine quality of a project, these types of linear fit methods are often employed. In this case, the metrics will be the independent variable and the measure of quality will be the dependent variable. Linear regression models try to choose the best subset from the independent variables that explain as much variation in the dependent variable as possible (Khoshgoftaar 980). In order to evaluate the fit of a line to the data, two considerations need to be taken into account. First, the model must accurately represent the linear dispersion of the data. Second, the model must make meaningful predictions. It is possible to nicely fit a curve to data that has no accurate predictive qualities (Khoshgoftaar 981-82).

Now that a precise way of measuring the accuracy of estimates has been outlined, the usefulness of predictions will be covered. Estimates are made at the beginning of the software project and are continuously revised. If a software project is perceived to be late, then more people can be hired or the schedule can be slipped (Abdel-Hamid 1431). When determining staffing needs of a project, the manager will consider project completion date. He will then hire the correct number of people necessary to meet this deadline (Abdel-Hamid 1430). At the early stages of development, progress is often determined by budgetary expenditures. However, over time the project team begins to produce tangible products, and management can use these artifacts to determine how productive the software team has been (Abdel-Hamid 1431). It is possible to develop systems that can help this process become automated. If human judgment calls or direct intervention are necessary to make estimates about software projects, then an expert system would be ideal (Abdel-Hamid 1437). Expert systems can be developed that can learn the behavior of the system being developed as more data is entered. This will allow the manger to predict such things as when the remaining milestones will be reached and how many people should be assigned to the project (Putnam 107).

project will be completed sooner (Abdel-Hamid 1436). One factor that is important to consider when adding people is their experience level. Newly added team members are normally less productive (Abdel-Hamid 1429-30).

Traditionally, making estimates about software is highly inaccurate. The reason is that the software process is very malleable and invites changes late in the design process (Lutz 110). Few other engineering disciplines would allow changes during the implementation stage, however this is common in software development (Lutz 110). Additionally, managers have a limited source of information. One of their primary means of information is the programmer. However, when asking programmers about completion of the project, they often suffer from what is called 90% completion syndrome. Programmers make broad jumps in the completion rate of their program until they reach 90%. Then their estimates begin to make very small jumps until the product is finally finished (Abdel-Hamid 1432).

There are some problems with trying to define product quality at the beginning of a software project and then estimating it while in the process of development. This has resulted in large errors in planned completion dates, in estimating effort, and in projecting cost. These all lead to considerable completion delays and cost overruns. The problem is not due to just the subjective nature of the estimates, it depends on some objective considerations as well (Belova 949). Designers often fail to modify their estimates even though they are known to be wrong. The reason is that they do not want to be perceived as doing a poor job. They do not want to have to change the numbers only to have to change them again later. This would result in their looking bad twice (Abdel-Hamid

1432). The best approach might be to use independent estimators since they are not subject to pressure to make estimates more favorable. Also they are detached from the project and thus are not as biased (Lutz 111). It is obvious that less subjective methods need to be developed.

One Estimation Technique

As has been demonstrated, managers have many options available to them in order to help them make predictions. One of the keys to this type of management strategy is to use statistical techniques coupled with metrics throughout the development process (Putnam 106). This allows the manager to carefully monitor the progress of the project with real project data. This notifies him immediately of a schedule slip or overrun. It replaces the inaccurate guess work of other models with actual project statistics. By using these techniques, unacceptable predictions can be corrected early enough in the process to make a difference (Putnam 106). The basics of this type of software control are milestone completion, effort expenditure, code production and defect rate. These metrics should be gathered on a monthly basis and compared to the planned approach. Unfavorable slippage or overruns can be gauged using adaptive forecasting (Putnam 106). In order to make it work, the company must take quality seriously, take product improvement seriously, measure progress with the correct metrics, and set realistic goals (Putnam 106). Companies that make these commitments are able to reduce costs by 25% per year, shorten schedules by 10% per year, and increase their capacity to handle complex and large software projects. This allows them to produce better products that satisfy the customer (Putnam 106). The analysis behind this method is more advanced than simple extrapolation. Instead it uses curve-fitting techniques that take into account project behavior (Putnam 107). This approach is a good one, however building upon it, a more thorough estimate can be achieved.

Another Approach - Models

Just as models can be used to represent the software life cycle, they can also be used to predict the software life cycle. Estimating schedules and staffing incorrectly in the early stages of a software project can result in low quality projects and dissatisfied customers. However, staff and cost estimation models can suggest staffing levels and reasonable schedules (Lehder 10). There is an increasing demand to be able to quantify software quality. Since, in most cases, quality measures cannot be taken until late in the design process, there is increasing pressure to develop linear models that can predict software quality from early in the software process. The idea is to base these estimates on metrics that be gathered early in the process such as complexity (Khoshgoftaar 979). There are several different techniques that can be used to predict cost and effort for a software project. First, algorithmic cost modeling can be used. This involves the creation of a model which relates project cost to a software metric such as size. Once an estimate of the metric is received, an effort estimation can be taken from the model. Second, expert judgment can be used. Experts on similar applications are consulted and their estimates compared. Third, estimates can be made based upon similar projects. Fourth, Parkinson's Law says that work expands to fill the time available. This approach looks at resources instead of an objective assessment of the project. Fifth, the cost of the project is set to whatever the customer has to spend. Sixth, cost estimates are based upon the number of functions required in the program and how complex their interaction is. Finally, a bottom-up approach is taken in which the cost of every sub-module is computed and then summed for the entire project (Sommerville 513). The first approach is the one that will be discussed in this paper. There are two types of models. Micro models make estimates starting at the bottom and work their way up. However, macro models take a top-down approach (Lutz 111).

Estimation techniques for software products are based on a model for software development that breaks the life cycle into several stages. A simple life cycle is three simple stages of planning, development, and maintenance. Each of these phases is basically independent of the other phases and concludes with the accomplishment of a project milestone (Lehder 11). Historically, software producers have worked with the waterfall model where the development process falls through stages of development. Prediction techniques based on this type of model work in two parts. The first part creates a base estimation based upon some high level estimate such as project size. The second part refines this estimation based upon environmental factors such as programmer experience (Lehder 11). There are several models that can be used to predict schedule and staffing needs. In 1981, Barry Boehm created the COCOMO model. Other models have been created by Jensen, Putnam, Rubens, and Jones. Many of these algorithms have been commercially implemented and can be purchased. The models have continued to be modified and adapted over time, leading to more accurate predictions. As these models become more accurate, their use increases, leading to better estimates in the future (Lehder 11).

Figures obtained from algorithmic modeling can often yield extremely diverse answers. The discrepancies do not say that the model is bad. Instead it suggests that the model and its parameters need to be tailored to the organization developing the software. Thus, after a model has been tailored to the specific application for which it will be used, the estimates are much better (Sommerville 514). The problem with these models is that they rely on the quantification of some attribute of the completed project such as the number of lines of code. Often, managers must makes guesses long before coding has even started for a project. Often, these guesses can be very inaccurate(Sommerville 514-515). Additionally, the lack of understanding of the precise relationship among the many possible software metrics prevents their widespread use in software modeling. More research needs to be done in this area (Khoshgoftaar 979). Continued change in the software development process such as reusable code modules and greater use of object-oriented languages will change the way that models can be applied. Adaptation of these models to new techniques will require newer and better models (Lehder 18).

The COCOMO Model

The Basic Model

Barry Boehm developed the COCOMO model in order to make predictions about the software life cycle. It is an example of an algorithmic model that can be tailored to a specific organization. This model exists in three forms: the simple, intermediate, and detailed forms. The basic model gives an order of magnitude estimate of software effort. It looks only at the size of the program and the type of project (Sommerville 517). The basic COCOMO model breaks projects into three different types. The first type is the organic mode. These are small projects in which the teams are working in familiar environments developing well understood and familiar projects. Communication overhead is low and efficiency is high. The second type of project is the semi-detached mode. In these projects, the teams have both experienced and inexperienced members. Team members are experienced with parts of the project, but not all. The final type are embedded mode projects. These are projects in which the software will be tightly coupled with hardware. Because of project diversity, team members will not likely have much experience with similar projects (Sommerville 517). The formula for the basic COCOMO model is:

$\text{Effort} = A\ (\text{KDSI})^b$

Where KDSI is the number of thousand of delivered source instructions. A and b are constants determined by the project type. For this model, a source statement is defined as an actual line in the program regardless of how many instructions are actually on that line. Comment lines are excluded from consideration (Sommerville 518). The actual value of the parameters for A and b should be tailored to the corporation using information about previously completed applications. The Effort is computed in terms of person-months. Boehm defined a person-month as 152 hours which takes into account employees missing work whether for holidays or illness (Sommerville 518).

The Intermediate Model

The intermediate COCOMO model takes the rough estimate made by the basic model and adds in other considerations in order to make the estimate more accurate.

These other factors are multiplied against the prediction made in the basic model. A normal value for the multiplier is attributed a value of one. Outstanding values are less than one and bad values are greater than one. These factors can be broken into four areas:

- Product Attributes
    - Reliability
    - Database Size
    - Product Complexity
- Computer Attributes
    - Execution Time Constraints
    - Storage Constraints
    - Virtual Machine Volatility
    - Computer Turnaround Time
- Personnel Attributes
    - Analyst Capability
    - Application Experience
    - Virtual Machine Experience
    - Programmer Capability
    - Programming Language Experience
- Project Attributes
    - The Use of Software Tools
    - The Development Schedule
    - The Use of Modern Programming Practices

(Sommerville 522-523).

Since factors have changed since Boehm originally proposed the nominal values, in order for the multipliers to be effective, they must be tailored using historical records of previous projects. One of the problems with both this model and the basic model is that they treat the entire software process as a single entity. However, most large systems are made up of several smaller systems which are not homogeneous (Sommerville 523).

Using the COCOMO Model

The COCOMO model can be very useful for a manager. While planning the project, the manager establishes milestones. These are end-points in the software process activity. Upon the reaching of a milestone, the manager should receive some type of formal report. A good milestone is characterized by a completed document. Usually, milestone delivery should be scheduled every two or three weeks. The waterfall model readily lends itself to the establishment of milestones (Sommerville 497-498). The COCOMO model can be tailored after the fact by comparing actual cost with computed cost and then using a least squares fit to change the parameters (Sommerville 524). In order to make accurate estimates, data needs to be accumulated on completed software projects. The data should be processed and analyzed to create a statistical database of completed projects for a company (Belova 949).

Program Description

The Goal

The goal of the program that I developed was to help software managers make meaningful predictions about the software that they are developing. I wanted to design a program that could become more accurate over time, as the parameters were tuned to meet the demands of the corporation in which they would be used. In order to realize this, I knew that an historical database of previous projects needed to be maintained. This database in conjunction with currently undertaken projects would allow the software to make useful guesses. I wanted the guesses to be based on some figures that the manager could estimate, but I also wanted a system that was not completely dependent on the manager's predictions. The software should try and smooth out the error caused from really bad estimates.

The Method

The program that I have developed is written for the Windows operating system. It was developed using Borland C++. Originally, I had thought to use a knowledge based tool called Clips in order to more effectively handle the rules for the system. However, the project did not develop as quickly as I had hoped and so a large rule base in order for the system to make its estimates was not needed. I started the project with the simple goal of using the COCOMO model to make predictions about software projects. My goal was to get a simple system working and then add more detailed analysis as time allowed. This would allow the system to be frozen at any stage and still be functional.

The Program

The Estimation Program resembles a standard windows program when it is loaded (Figure-4). The user is presented with several different choices. The first menu that the user can choose from is the "Project" menu item. This will drop down a list box that presents the user with several different choices, all concerning storing and retrieving project information. First, the user is presented with a "New Project" option. This option will prompt the user to enter a project name(Figure-5). If the user enters a name that has already been entered as a project name, then an appropriate error message will inform the user of their error and then it will allow them to correct the error. In Figure-5 a project

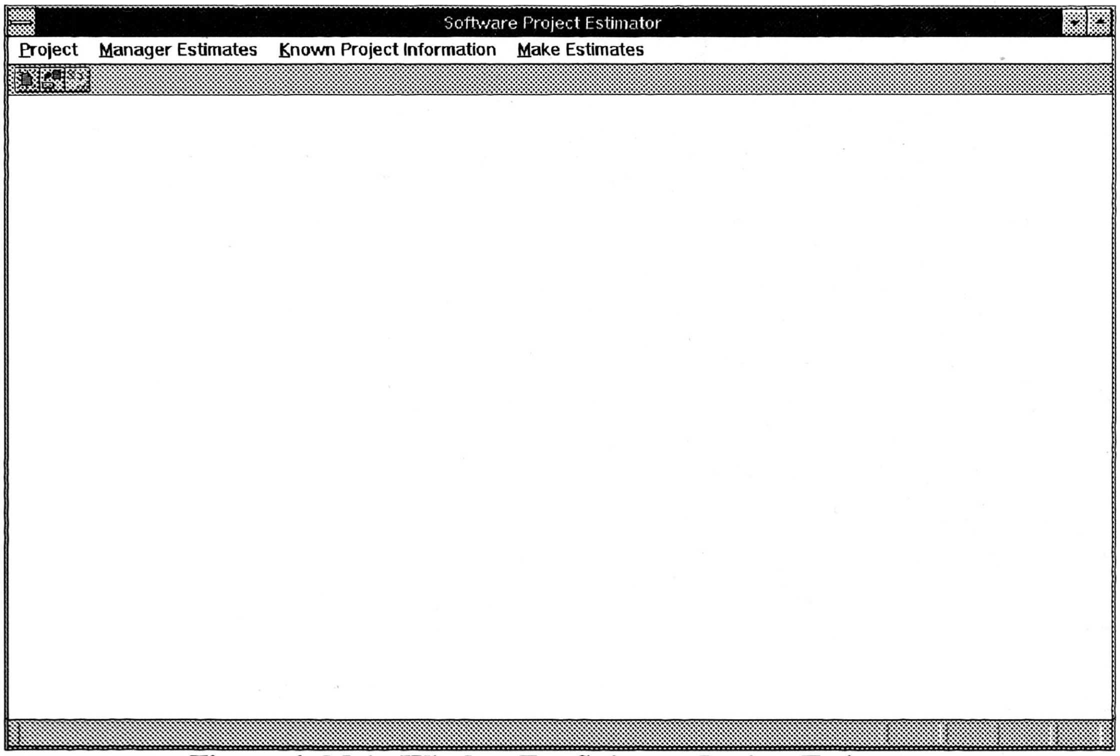
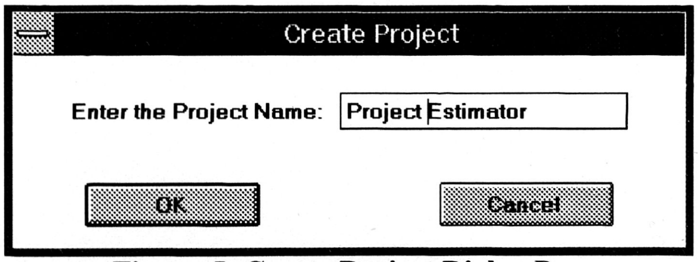


Figure-4: Main Window For Software Project Estimator

called "Project-Estimator" has been entered. A similar function is performed by the next option under the "Project" menu. This second item is the "Open Project" option. This option presents the user with a drop-down list box that contains a list of all of the projects



Enter the Project Name: Project Estimator

Figure-5: Create Project Dialog Box

that have been created(Figure-6). As can be seen, the new project that was just created, "Project Estimator" appears in the list box. The user would then click on the item and then choose OK. All relevant information that has already been entered about this project would be loaded so that it could be modified. Since this project was just created, there is no information to load. The next option under this menu is "Save Project." The function that this item performs is obvious. It writes all of the information that has been entered about the current project out to disk. This allows the information to be recalled at a later date. The final option under this menu is to "Exit." This closes the program and saves all current project information.

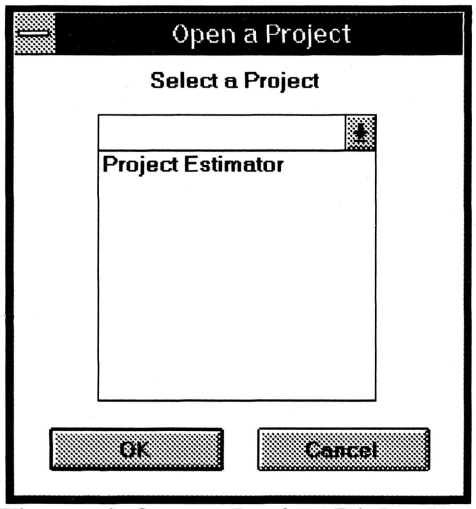
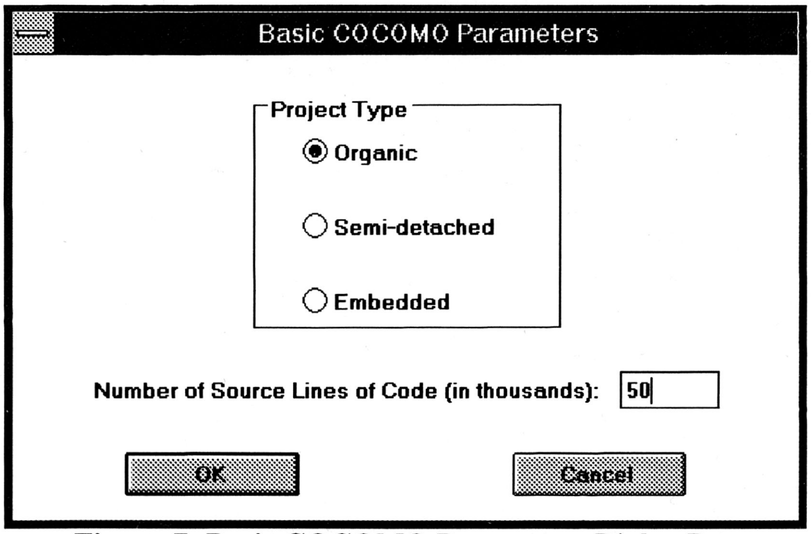


Figure-6: Open a Project Dialog Box

All of the functions performed under the first menu item are pretty mundane and every program has similar functions. However, the next item seen on the menu-bar, "Manager Estimates" performs some more interesting operations. Each of the items under this menu are concerned with gathering information from the manager. This information will be estimates made by the manager concerning the project. For example, the first item is called "Basic COCOMO Model Parameters." This item loads a dialog box that is interested in gathering information about the basic COCOMO model(Figure-7). This

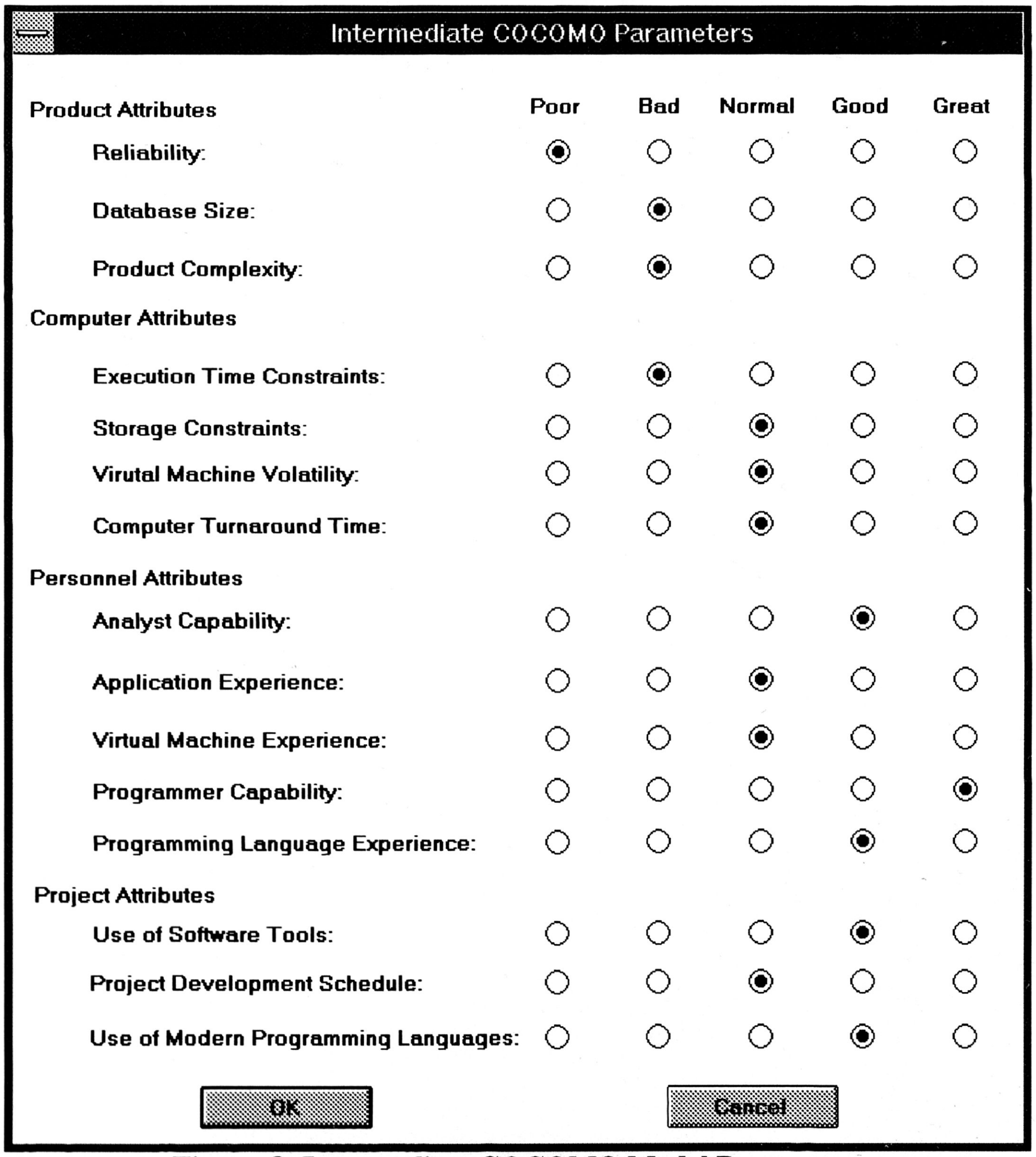

Basic COCOMO Parameters

Project Type
[✓ Organic]
Semi-detached
Embedded

Number of Source Lines of Code (in thousands): 50

Figure-7: Basic COCOMO Parameters Dialog Box

information includes the project type. As was discussed earlier, organic projects are the simplest, semi-detached slightly more complex, and embedded extremely difficult. This dialog box also asks the manager to estimates the number of source lines of code. In this case, an organic project is predicted which will generate 50,000 lines of code. This is a relatively small project. Once the manager has entered this information then he is ready to proceed to the Intermediate COCOMO Model Parameters screen, which is the next choice from the main menu(Figure-8). This screen presents a list of the 15 multipliers, broken up



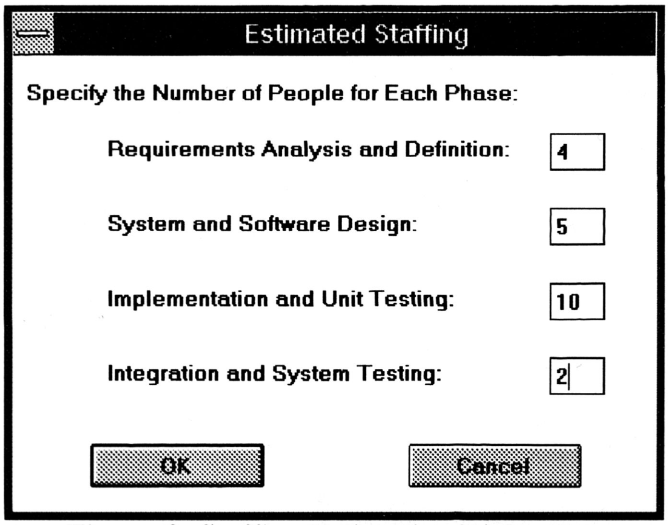
Intermediate COCOMO Parameters

	Poor	Bad	Normal	Good	Great
Product Attributes					
Reliability:	●	○	○	○	○
Database Size:	○	●	○	○	○
Product Complexity:	○	●	○	○	○
Computer Attributes					
Execution Time Constraints:	○	●	○	○	○
Storage Constraints:	○	○	●	○	○
Virutal Machine Volatility:	○	○	●	○	○
Computer Turnaround Time:	○	○	●	○	○
Personnel Attributes					
Analyst Capability:	○	○	○	●	○
Application Experience:	○	○	●	○	○
Virtual Machine Experience:	○	○	●	○	○
Programmer Capability:	○	○	○	○	●
Programming Language Experience:	○	○	○	●	○
Project Attributes					
Use of Software Tools:	○	○	○	●	○
Project Development Schedule:	○	○	●	○	○
Use of Modern Programming Languages:	○	○	○	●	○

Figure-8: Intermediate COCOMO Model Parameters

by category. The manager would specify whether the quality of the multiplier is poor, bad, normal, good, or great. As can be seen by the data that has been entered, this project has some poor storage considerations. However, there are some good people working on the project. The final set of data that the manager needs to estimate is the staffing needs for each sub-cycle of the project. The menu option is called "Staffing Estimates" (Figure-

9). This screen asks the manager to enter a value for each phase of the project. As is typical in projects, fewer people are used on the design part of the project, and more people are brought in for the implementation part. All of this information together is the extent of the information that the manager is asked to guess. The rest of the information comes for empirical observation.


Estimated Staffing

Specify the Number of People for Each Phase:

Requirements Analysis and Definition: 4

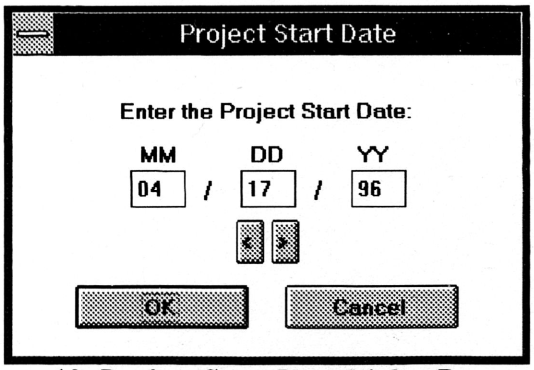
System and Software Design: 5

Implementation and Unit Testing: 10

Integration and System Testing: 2

Figure-9: Staffing Estimation Dialog Box

The next menu option on the menu bar is called "Known Project Information." This is empirical information about the project that becomes known as time progresses. The first option is to enter the "Project Start Date" (Figure-10). This dialog box is fairly

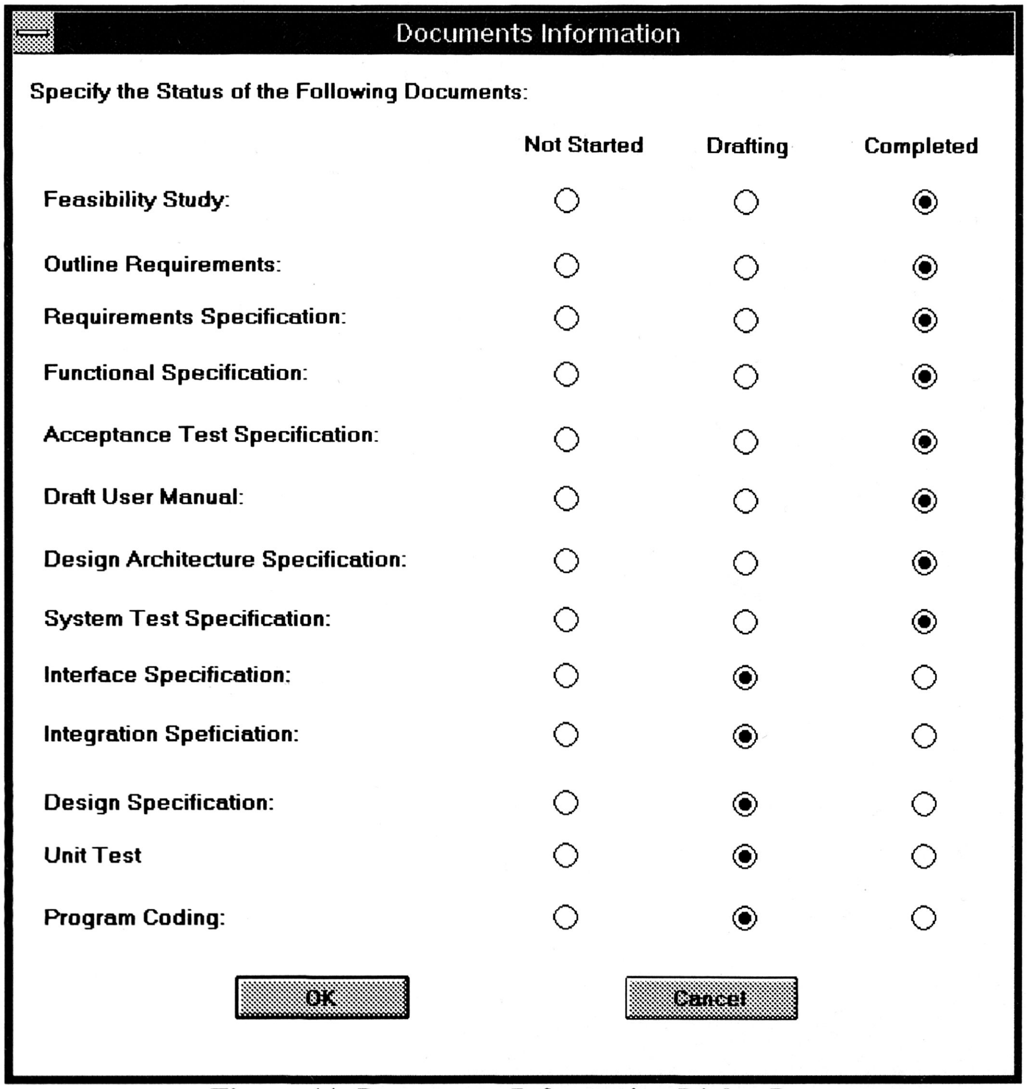

Project Start Date

Enter the Project Start Date:

MM 04 / DD 17 / YY 96

Figure-10: Project Start Date Dialog Box

self explanatory. It asks for the start date in a MM/DD/YY format. When the screen comes up, the current date is automatically filled in. The left and right arrows decrease and increase the date respectively by one day at a time. When the correct date has been entered, the manager would press OK. The next menu item that deals with actual project data is called "Document Information" (Figure-11). This screen gathers the status information on the various documents that can be produced on a project. Ideally this screen would be update frequently. If, when the user presses "OK", a new phase has been entered into than previously achieved, then another screen would pop up asking for information about the completed phase (Figure-12). This new screen gathers some concrete information about the completed phase, so that future predictions will be more accurate. The information that this screen asks for is similar to earlier information. First,

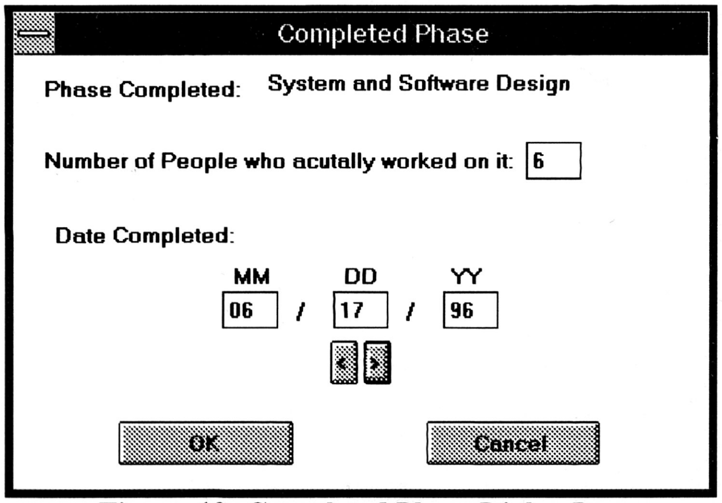

Documents Information

Specify the Status of the Following Documents:

	Not Started	Drafting	Completed
Feasibility Study:	○	○	●
Outline Requirements:	○	○	●
Requirements Specification:	○	○	●
Functional Specification:	○	○	●
Acceptance Test Specification:	○	○	●
Draft User Manual:	○	○	●
Design Architecture Specification:	○	○	●
System Test Specification:	○	○	●
Interface Specification:	○	●	○
Integration Speficiation:	○	●	○
Design Specification:	○	●	○
Unit Test	○	●	○
Program Coding:	○	●	○

Figure-11: Documents Information Dialog Box

it asks for the user to specify the number of people that actually worked on the project. This is a refinement of the earlier value which was just a guess on the part of the manager. It also asks for the date on which the phase was completed. According to the documents screen, the implementation phase has been entered, even though the earlier phase still has some lose ends. Since, the implementation phase had not previously been entered, the program assumes that the previous stage was just completed. It pops up the dialog box to gather more information on this recently completed phase. The date pops up as the current date, however, as this user has entered the information, the design stage was completed two months after the start of the project. Since the document screen only goes up until the completion of the coding phase, there is no way to tell when the integration and testing stage has been completed. Therefore, the last dialog option on the main menu is to enter the information on this stage. The dialog box is exactly the same as the one seen in Figure-12.



Phase Completed: System and Software Design

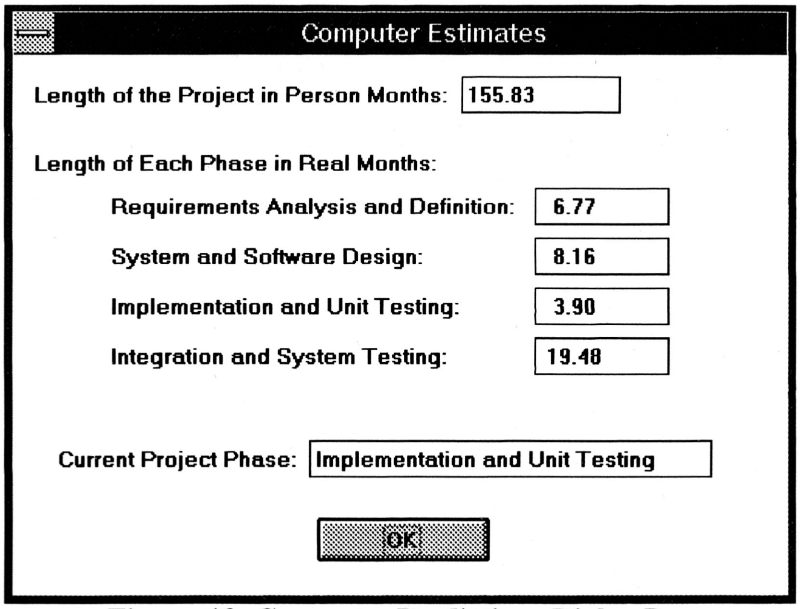
Number of People who acutally worked on it: 6

Date Completed:

MM		DD		YY
06	/	17	/	96

Figure-12: Completed Phase Dialog Box

The last option on the menu bar is called "Make Estimates." This is one of the most important buttons on the menu. This is the button that takes all of the information that has been gathered, and it uses it to make predictions about the software. When this option is chosen, the program returns the screen seen in Figure-13. This screen shows the


Computer Estimates

Length of the Project in Person Months: 155.83

Length of Each Phase in Real Months:

Requirements Analysis and Definition: 6.77

System and Software Design: 8.16

Implementation and Unit Testing: 3.90

Integration and System Testing: 19.48

Current Project Phase: Implementation and Unit Testing

Figure-13: Computer Predictions Dialog Box

estimated length of the project according to the COCOMO model. Then it uses staffing information and completion dates in order to calculate the actual number of months for the project. Finally, it shows what stage the project is in.

Future Improvements

The program has a great deal of room for improvement. The error trapping is at a minimum. This means that the user has to know exactly what they are doing or else it would be very easy to mess up the system. Also, there is no cost estimations involved. The reason that management wants to be able to make estimates about a software product is to know how much it will cost. Additionally, it is not possible to enter the date when the manager would like to have the project complete and then have the computer tell him how many people he should hire. All of these are features that would be easy to implement if more time were available. Additionally, more estimates and predictions could be built into the system.

One other area for improvement for the program is in how it divides work amongst the various phases. At first, it divides the work evenly between all of the phases. As it gathers information on projects, it looks to see how much of the time is actually spent on each phase. The algorithm that it uses to fit this information could be modified so that an average of all values is taken over time. Additionally, the parameters for the COCOMO model should also be changed to reflect the needs of the company using the software. However, right now, these values are static. They are being stored in a database, so it would be a simple matter to have the computer modify them as needed.

Conclusion

The potential for a project like this is overwhelming. Managers understand the importance of making accurate predictions and yet most of them fail to use even a simple model. Instead they rely upon their own padded guesses with a huge margin for error. A program like this would provide software managers with a tool that would allow them to make changes in the project far enough in advance that all benefits would be collected. For example, if it is known that five more programmers will be needed in the implementation stage, then they can be brought in before coding even begins. This is much better than trying to bring these programmers in once coding has started. This creates almost as many problems as it solves. Even the simple algorithm that I have developed is more accurate to use than a raw guess. In fact, since this system will modify itself to meet the needs of the specific company, it is much better than a wild guess.

Appendix

Database Files

dbparam.h

```
#ifndef DBPARAM_H
#define DBPARAM_H

#include "..\..\srcpool\dbbase.h"

#if defined DB_DLLBUILD
        #define _DBCLASS _export
#elif defined DB_DLL
        #define _DBCLASS _import
#else
        #define _DBCLASS
#endif


class _DBCLASS DBParam : public DBBase
{
        public:
                DBParam(char * sFile="paramtrs");

                FIELD4 *rPROJTYPE;
                FIELD4 *rCOC_A;
                FIELD4 *rCOC_B;
                FIELD4 *rPERPHS1;
                FIELD4 *rPERPHS2;
                FIELD4 *rPERPHS3;
                FIELD4 *rPERPHS4;

                char *ProjectType()     {return get(rPROJTYPE);  };
                char *CocomoA()         {return get(rCOC_A);    };
                char *CocomoB()         {return get(rCOC_B);    };
                int  PercentPhase1()    {return getInt(rPERPHS1);};
                int  PercentPhase2()    {return getInt(rPERPHS2);};
                int  PercentPhase3()    {return getInt(rPERPHS3);};
                int  PercentPhase4()    {return getInt(rPERPHS4);};

                void ProjectType(char *str)       {set(rPROJTYPE, str);};
                void CocomoA(char *str)           {set(rCOC_A, str);  };
                void CocomoB(char *str)           {set(rCOC_B, str);  };
                void PercentPhase1(char *str)     {set(rPERPHS1, str); };
                void PercentPhase2(char *str)     {set(rPERPHS2, str); };
                void PercentPhase3(char *str)     {set(rPERPHS3, str); };
                void PercentPhase4(char *str)     {set(rPERPHS4, str); };
};
#endif
```

dbparam.cpp

```
#include "dbparam.h"

static FIELD4INFO Fields[]=
        {
                {"PROJTYPE", 'C', 10, 0},
                {"COC_A",    'C',  6, 0},
                {"COC_B",    'C',  6, 0},
                {"PERPHS1",  'C',  2, 0},
                {"PERPHS2",  'C',  2, 0},
                {"PERPHS3",  'C',  2, 0},
                {"PERPHS4",  'C',  2, 0},
                {0,0,0,0}             // null entry at end
        };

static TAG4INFO Tags[]=
        {
                {"PROJTYPE_","PROJTYPE","",0,0},
                {0,0,0,0,0} /* null entry at end */
```

```
		};

/////////////////////////////////////////////
DBParam::DBParam(char *sFile)
				:DBBase(sFile, Fields, Tags)
{
		rPROJTYPE=fieldRef("PROJTYPE");
		rCOC_A     =fieldRef("COC_A"    );
		rCOC_B     =fieldRef("COC_B"    );
		rPERPHS1  =fieldRef("PERPHS1" );
		rPERPHS2  =fieldRef("PERPHS2" );
		rPERPHS3  =fieldRef("PERPHS3" );
		rPERPHS4  =fieldRef("PERPHS4" );
}
```

dbproj.h

```
#ifndef DBPROJ_H
#define DBPROJ_H

#include "..\..\srcpool\dbbase.h"

#if defined DB_DLLBUILD
		#define _DBCLASS _export
#elif defined DB_DLL
		#define _DBCLASS _import
#else
		#define _DBCLASS
#endif

class _DBCLASS DBProj : public DBBase
{
		public:
				DBProj(char * sFile="projects");

				FIELD4 *rPROJNAME;
				FIELD4 *rPROJTYPE;
				FIELD4 *rESTMSIZE;
				FIELD4 *rINTPRM1;
				FIELD4 *rINTPRM2;
				FIELD4 *rINTPRM3;
				FIELD4 *rINTPRM4;
				FIELD4 *rINTPRM5;
				FIELD4 *rINTPRM6;
				FIELD4 *rINTPRM7;
				FIELD4 *rINTPRM8;
				FIELD4 *rINTPRM9;
				FIELD4 *rINTPRM10;
				FIELD4 *rINTPRM11;
				FIELD4 *rINTPRM12;
				FIELD4 *rINTPRM13;
				FIELD4 *rINTPRM14;
				FIELD4 *rINTPRM15;
				FIELD4 *rSTRTDTMN;
				FIELD4 *rSTRTDTDY;
				FIELD4 *rSTRTDTYR;
				FIELD4 *rDOC1;
				FIELD4 *rDOC2;
				FIELD4 *rDOC3;
				FIELD4 *rDOC4;
				FIELD4 *rDOC5;
				FIELD4 *rDOC6;
				FIELD4 *rDOC7;
				FIELD4 *rDOC8;
				FIELD4 *rDOC9;
				FIELD4 *rDOC10;
				FIELD4 *rDOC11;
				FIELD4 *rDOC12;
				FIELD4 *rDOC13;
				FIELD4 *rPHS1DONE;
```

```
FIELD4 *rPHS2DONE;
FIELD4 *rPHS3DONE;
FIELD4 *rPHS4DONE;
FIELD4 *rPHS1STFF;
FIELD4 *rPHS2STFF;
FIELD4 *rPHS3STFF;
FIELD4 *rPHS4STFF;
FIELD4 *rPHS1MNTH;
FIELD4 *rPHS1DAY;
FIELD4 *rPHS1YEAR;
FIELD4 *rPHS2MNTH;
FIELD4 *rPHS2DAY;
FIELD4 *rPHS2YEAR;
FIELD4 *rPHS3MNTH;
FIELD4 *rPHS3DAY;
FIELD4 *rPHS3YEAR;
FIELD4 *rPHS4MNTH;
FIELD4 *rPHS4DAY;
FIELD4 *rPHS4YEAR;

char *ProjectName()                 {return get(rPROJNAME);};
int  ProjectType()                  {return getInt(rPROJTYPE);};
long  EstimatedSize()               {return getLong(rESTMSIZE);};
int  IntermediateParameter1()       {return getInt(rINTPRM1);};
int  IntermediateParameter2()       {return getInt(rINTPRM2);};
int  IntermediateParameter3()       {return getInt(rINTPRM3);};
int  IntermediateParameter4()       {return getInt(rINTPRM4);};
int  IntermediateParameter5()       {return getInt(rINTPRM5);};
int  IntermediateParameter6()       {return getInt(rINTPRM6);};
int  IntermediateParameter7()       {return getInt(rINTPRM7);};
int  IntermediateParameter8()       {return getInt(rINTPRM8);};
int  IntermediateParameter9()       {return getInt(rINTPRM9);};
int  IntermediateParameter10()      {return getInt(rINTPRM10);};
int  IntermediateParameter11()      {return getInt(rINTPRM11);};
int  IntermediateParameter12()      {return getInt(rINTPRM12);};
int  IntermediateParameter13()      {return getInt(rINTPRM13);};
int  IntermediateParameter14()      {return getInt(rINTPRM14);};
int  IntermediateParameter15()      {return getInt(rINTPRM15);};
int  StartDateMonth()               {return getInt(rSTRTDTMN);};
int         StartDateDay()          {return getInt(rSTRTDTDY);};
int         StartDateYear()         {return getInt(rSTRTDTYR);};
int         Document1()             {return getInt(rDOC1);};
int         Document2()             {return getInt(rDOC2);};
int         Document3()             {return getInt(rDOC3);};
int         Document4()             {return getInt(rDOC4);};
int         Document5()             {return getInt(rDOC5);};
int         Document6()             {return getInt(rDOC6);};
int         Document7()             {return getInt(rDOC7);};
int         Document8()             {return getInt(rDOC8);};
int         Document9()             {return getInt(rDOC9);};
int         Document10()            {return getInt(rDOC10);};
int         Document11()            {return getInt(rDOC11);};
int         Document12()            {return getInt(rDOC12);};
int         Phase1Done()            {return getInt(rPHS1DONE);};
int         Phase2Done()            {return getInt(rPHS2DONE);};
int         Phase3Done()            {return getInt(rPHS3DONE);};
int         Phase4Done()            {return getInt(rPHS4DONE);};
int         Phase1Staff()           {return getInt(rPHS1STFF);};
int         Phase2Staff()           {return getInt(rPHS2STFF);};
int         Phase3Staff()           {return getInt(rPHS3STFF);};
int         Phase4Staff()           {return getInt(rPHS4STFF);};
int         Phase1Month()           {return getInt(rPHS1MNTH);};
int         Phase1Day()             {return getInt(rPHS1DAY);};
int         Phase1Year()            {return getInt(rPHS1YEAR);};
int         Phase2Month()           {return getInt(rPHS2MNTH);};
int         Phase2Day()             {return getInt(rPHS2DAY);};
int         Phase2Year()            {return getInt(rPHS2YEAR);};
int         Phase3Month()           {return getInt(rPHS3MNTH);};
int         Phase3Day()             {return getInt(rPHS3DAY);};
int         Phase3Year()            {return getInt(rPHS3YEAR);};
int         Phase4Month()           {return getInt(rPHS4MNTH);};
```

```
                int         Phase4Day()            {return getInt(rPHS4DAY);};
                int         Phase4Year()           {return getInt(rPHS4YEAR);};
                void ProjectName(char * str)       {set(rPROJNAME, str);};
                void ProjectType(int num)          {set(rPROJTYPE, num);};
                void EstimatedSize(long num)       {set(rESTMSIZE, num);};
                void IntermediateParameter1(int num)        {set(rINTPRM1, num);};
                void IntermediateParameter2(int num)        {set(rINTPRM2, num);};
                void IntermediateParameter3(int num)        {set(rINTPRM3, num);};
                void IntermediateParameter4(int num)        {set(rINTPRM4, num);};
                void IntermediateParameter5(int num)        {set(rINTPRM5, num);};
                void IntermediateParameter6(int num)        {set(rINTPRM6, num);};
                void IntermediateParameter7(int num)        {set(rINTPRM7, num);};
                void IntermediateParameter8(int num)        {set(rINTPRM8, num);};
                void IntermediateParameter9(int num)        {set(rINTPRM9, num);};
                void IntermediateParameter10(int num)       {set(rINTPRM10, num);};
                void IntermediateParameter11(int num)       {set(rINTPRM11, num);};
                void IntermediateParameter12(int num)       {set(rINTPRM12, num);};
                void IntermediateParameter13(int num)       {set(rINTPRM13, num);};
                void IntermediateParameter14(int num)       {set(rINTPRM14, num);};
                void IntermediateParameter15(int num)       {set(rINTPRM15, num);};
                void StartDateMonth(int num)                {set(rSTRTDTMN, num);};
                void StartDateDay(int num)                  {set(rSTRTDTDY, num);};
                void StartDateYear(int num)                 {set(rSTRTDTYR, num);};
                void Document1(int num)                     {set(rDOC1, num);};
                void Document2(int num)                     {set(rDOC2, num);};
                void Document3(int num)                     {set(rDOC3, num);};
                void Document4(int num)                     {set(rDOC4, num);};
                void Document5(int num)                     {set(rDOC5, num);};
                void Document6(int num)                     {set(rDOC6, num);};
                void Document7(int num)                     {set(rDOC7, num);};
                void Document8(int num)                     {set(rDOC8, num);};
                void Document9(int num)                     {set(rDOC9, num);};
                void Document10(int num)                    {set(rDOC10, num);};
                void Document11(int num)                    {set(rDOC11, num);};
                void Document12(int num)                    {set(rDOC12, num);};
                void Document13(int num)                    {set(rDOC13, num);};
                void Phase1Done(int num)                    {set(rPHS1DONE, num);};
                void Phase2Done(int num)                    {set(rPHS2DONE, num);};
                void Phase3Done(int num)                    {set(rPHS3DONE, num);};
                void Phase4Done(int num)                    {set(rPHS4DONE, num);};
                void Phase1Staff(int num)                   {set(rPHS1STFF, num);};
                void Phase2Staff(int num)                   {set(rPHS2STFF, num);};
                void Phase3Staff(int num)                   {set(rPHS3STFF, num);};
                void Phase4Staff(int num)                   {set(rPHS4STFF, num);};
                void Phase1Month(int num)                   {set(rPHS1MNTH, num);};
                void Phase1Day(int num)                     {set(rPHS1DAY, num);};
                void Phase1Year(int num)                    {set(rPHS1YEAR, num);};
                void Phase2Month(int num)                   {set(rPHS2MNTH, num);};
                void Phase2Day(int num)                     {set(rPHS2DAY, num);};
                void Phase2Year(int num)                    {set(rPHS2YEAR, num);};
                void Phase3Month(int num)                   {set(rPHS3MNTH, num);};
                void Phase3Day(int num)                     {set(rPHS3DAY, num);};
                void Phase3Year(int num)                    {set(rPHS3YEAR, num);};
                void Phase4Month(int num)                   {set(rPHS4MNTH, num);};
                void Phase4Day(int num)                     {set(rPHS4DAY, num);};
                void Phase4Year(int num)                    {set(rPHS4YEAR, num);};
};
#endif
```

dbproj.cpp

```
#include "dbproj.h"

static FIELD4INFO Fields[]=
        {
                {"PROJNAME", 'C', 20, 0},
                {"PROJTYPE", 'C',  1, 0},
                {"ESTMSIZE", 'C',  7, 0},
                {"INTPRM1",  'C',  1, 0},
                {"INTPRM2",  'C',  1, 0},
                        {"INTPRM3",  'C',  1, 0},
                        {"INTPRM4",  'C',  1, 0},
                        {"INTPRM5",  'C',  1, 0},
                        {"INTPRM6",  'C',  1, 0},
                        {"INTPRM7",  'C',  1, 0},
                        {"INTPRM8",  'C',  1, 0},
                        {"INTPRM9",  'C',  1, 0},
                        {"INTPRM10", 'C',  1, 0},
                        {"INTPRM11", 'C',  1, 0},
                        {"INTPRM12", 'C',  1, 0},
                        {"INTPRM13", 'C',  1, 0},
                        {"INTPRM14", 'C',  1, 0},
                        {"INTPRM15", 'C',  1, 0},
                        {"STRTDTMN", 'C',  2, 0},
                        {"STRTDTDY", 'C',  2, 0},
                        {"STRTDTYR", 'C',  2, 0},
                        {"DOC1",     'C',  1, 0},
                        {"DOC2",     'C',  1, 0},
                        {"DOC3",     'C',  1, 0},
                        {"DOC4",     'C',  1, 0},
                        {"DOC5",     'C',  1, 0},
                        {"DOC6",     'C',  1, 0},
                        {"DOC7",     'C',  1, 0},
                        {"DOC8",     'C',  1, 0},
                        {"DOC9",     'C',  1, 0},
                        {"DOC10",    'C',  1, 0},
                        {"DOC11",    'C',  1, 0},
                        {"DOC12",    'C',  1, 0},
                        {"DOC13",    'C',  1, 0},
                        {"PHS1DONE", 'C',  1, 0},
                        {"PHS2DONE", 'C',  1, 0},
                        {"PHS3DONE", 'C',  1, 0},
                        {"PHS4DONE", 'C',  1, 0},
                        {"PHS1STFF", 'C',  2, 0},
                        {"PHS2STFF", 'C',  2, 0},
                        {"PHS3STFF", 'C',  2, 0},
                        {"PHS4STFF", 'C',  2, 0},
                        {"PHS1MNTH", 'C',  2, 0},
                        {"PHS1DAY",  'C',  2, 0},
                        {"PHS1YEAR", 'C',  2, 0},
                        {"PHS2MNTH", 'C',  2, 0},
                        {"PHS2DAY",  'C',  2, 0},
                        {"PHS2YEAR", 'C',  2, 0},
                        {"PHS3MNTH", 'C',  2, 0},
                        {"PHS3DAY",  'C',  2, 0},
                        {"PHS3YEAR", 'C',  2, 0},
                        {"PHS4MNTH", 'C',  2, 0},
                        {"PHS4DAY",  'C',  2, 0},
                        {"PHS4YEAR", 'C',  2, 0},
                        {0,0,0,0}               // null entry at end
            };

static TAG4INFO Tags[]=
            {
                        {"PROJNAME_", "PROJNAME", "", 0, 0},
                        {0,0,0,0,0} /* null entry at end */
            };

//////////////////////////////////////////////////////
DBProj::DBProj(char *sFile)
                        :DBBase(sFile, Fields, Tags)

{
            rPROJNAME=fieldRef("PROJNAME");
            rPROJTYPE=fieldRef("PROJTYPE");
            rESTMSIZE=fieldRef("ESTMSIZE");
            rINTPRM1 =fieldRef("INTPRM1" );
            rINTPRM2 =fieldRef("INTPRM2" );
            rINTPRM3 =fieldRef("INTPRM3" );
            rINTPRM4 =fieldRef("INTPRM4" );
            rINTPRM5 =fieldRef("INTPRM5" );
            rINTPRM6 =fieldRef("INTPRM6" );
            rINTPRM7 =fieldRef("INTPRM7" );
            rINTPRM8 =fieldRef("INTPRM8" );
            rINTPRM9 =fieldRef("INTPRM9" );
            rINTPRM10=fieldRef("INTPRM10");
            rINTPRM11=fieldRef("INTPRM11");
            rINTPRM12=fieldRef("INTPRM12");
            rINTPRM13=fieldRef("INTPRM13");
            rINTPRM14=fieldRef("INTPRM14");
            rINTPRM15=fieldRef("INTPRM15");
            rSTRTDTMN=fieldRef("STRTDTMN");
            rSTRTDTDY=fieldRef("STRTDTDY");
            rSTRTDTYR=fieldRef("STRTDTYR");
            rDOC1    =fieldRef("DOC1"    );
            rDOC2    =fieldRef("DOC2"    );
            rDOC3    =fieldRef("DOC3"    );
            rDOC4    =fieldRef("DOC4"    );
            rDOC5    =fieldRef("DOC5"    );
            rDOC6    =fieldRef("DOC6"    );
            rDOC7    =fieldRef("DOC7"    );
            rDOC8    =fieldRef("DOC8"    );
            rDOC9    =fieldRef("DOC9"    );
            rDOC10   =fieldRef("DOC10"   );
            rDOC11   =fieldRef("DOC11"   );
            rDOC12   =fieldRef("DOC12"   );
            rDOC13   =fieldRef("DOC13"   );
            rPHS1DONE=fieldRef("PHS1DONE");
            rPHS2DONE=fieldRef("PHS2DONE");
            rPHS3DONE=fieldRef("PHS3DONE");
            rPHS4DONE=fieldRef("PHS4DONE");
            rPHS1STFF=fieldRef("PHS1STFF");
            rPHS2STFF=fieldRef("PHS2STFF");
            rPHS3STFF=fieldRef("PHS3STFF");
            rPHS4STFF=fieldRef("PHS4STFF");
            rPHS1MNTH=fieldRef("PHS1MNTH");
            rPHS1DAY =fieldRef("PHS1DAY" );
            rPHS1YEAR=fieldRef("PHS1YEAR");
            rPHS2MNTH=fieldRef("PHS2MNTH");
            rPHS2DAY =fieldRef("PHS2DAY" );
            rPHS2YEAR=fieldRef("PHS2YEAR");
            rPHS3MNTH=fieldRef("PHS3MNTH");
            rPHS3DAY =fieldRef("PHS3DAY" );
            rPHS3YEAR=fieldRef("PHS3YEAR");
            rPHS4MNTH=fieldRef("PHS4MNTH");
            rPHS4DAY =fieldRef("PHS4DAY" );
            rPHS4YEAR=fieldRef("PHS4YEAR");
}
```

dbbase.h

```
#ifndef DBBASE_H
#define DBBASE_H

#define S4MDX

#include <d4all.h>

#if defined DB_DLLBUILD
            #define _DBCLASS _export
#elif defined DB_DLL
            #define _DBCLASS _import
#else
            #define _DBCLASS
#endif


//////////////////////////////////////////////////////////////////////
// DBBase CLASS                                                     //
//////////////////////////////////////////////////////////////////////
```

```
class _DBCLASS DBBase
		{
public:
		DBBase(char *sFileParm,FIELD4INFO *FieldParm=NULL, TAG4INFO *TagParm=NULL);
		DBBase(char *sDirParm, char *sFileParm,FIELD4INFO *FieldParm=NULL, TAG4INFO *TagParm=NULL);
		~DBBase();
		FIELD4 * fieldRef(char *sFieldName)		{return(d4field(db,sFieldName));};

		void set(FIELD4 *field, char * str){f4memo_assign(field, str );};
		void set(FIELD4 *field, char byte);
		void set(FIELD4 *field, int num);
		void set(FIELD4 *field, long num);

		void set(char * field, char * str)	{f4memo_assign(d4field(db,field), str );};
		void set(char * field, char byte);
		void set(char * field, int num);
		void set(char * field, long num);

		char*get(FIELD4 *field)				{return(f4memo_str(field));};
		char getChar(FIELD4 *field)	{return(*(f4str(field) ) );};
		int  getInt(FIELD4 *field)			{return( atoi( f4str(field) ) );};
		long getLong(FIELD4 *field)	{return( atol( f4str(field) ) );};

		char*get(char * field)		{return(f4memo_str(d4field(db,field)));};
		char getChar(char * field)	{return(*(f4str(d4field(db,field)) ) );};
		int  getInt(char * field) {return( atoi( f4str(d4field(db,field)) ) );};
		 long getLong(char * field) {return( atol( f4str(d4field(db,field)) ) );};


		void go(long rec);
		void top()				{d4top(db);};
		void bottom()				{d4bottom(db);};
		void skip(long nRecs=1L)	{d4skip(db,nRecs);};
		int  seek(char *sSearch)	{return(d4seek(db,sSearch));};
		int  seek(char *sSearch,char * sTagName)
										{tag(sTagName);
		return(d4seek(db,sSearch));};
		void tag(char *sTagName)	{d4tag_select(db,d4tag(db,sTagName));};

		long count()			{return(d4reccount(db));};
		long recno()			{return(d4recno(db));};
		int  eof()			{return(int(d4eof(db)));};
		int  bof()			{return(int(d4bof(db)));};

		void append()			{d4append_blank(db);};
		void save()			{d4write(db,d4recno(db));};
		void erase(long rec)	{go(rec);erase();};
		void erase()			{d4delete(db);fDeleted=1;};

		void open();
		void close()			{d4close(db);};

		virtual void init(CODE4 *cbstruct);

		int relate(void);
		void query(char * expr);
		int rtop(void);
		int reof(int rc);
		int rskip(long num = 1L);
		void rclose(void);


protected:
		DATA4 *db;
		CODE4 cb;
		RELATE4 * relation;

private:
		char *psFile;
		char sIntBuf[20];
		FIELD4INFO * fields;
		TAG4INFO * tags;
		int fDeleted;
};

typedef DBBase* PDBBase;


#endif
```

dbbase.cpp

```
#include "..\..\srcpool\dbbase.h"

#ifdef unix
#include <unistd.h>
#else
#include <io.h>
#endif

////////////////////////////////////////////////////////////////////
DBBase::DBBase(char *sFileParm,FIELD4INFO *FieldParm, TAG4INFO *TagParm)
{
		psFile = new char[strlen(sFileParm)+1];
		strcpy(psFile, sFileParm);

		char *psPeriodDelim = strchr(psFile, '.'); // find filename extension
		if(psPeriodDelim)
				*psPeriodDelim = '\0'; // cut off extension

		fields=FieldParm;
		tags=TagParm;

		fDeleted = 0; // flag to see if file needs to be packed

		init(&cb); // virtual function for setting CODE4

		open();

}
////////////////////////////////////////////////////////////////////
DBBase::DBBase(char *sDirParm,char *sFileParm,FIELD4INFO *FieldParm, TAG4INFO *TagParm)
{


		psFile = new char[strlen(sFileParm) + strlen(sDirParm)+2];

		strcpy(psFile, sDirParm);

		if( *sDirParm != NULL )
				{
#ifdef unix
				strcat(psFile,"/");
#else
				strcat(psFile,"\\");
#endif
				}

		strcat(psFile, sFileParm);

		char *psPeriodDelim = strchr(psFile, '.'); // find filename extension
		if(psPeriodDelim)
				*psPeriodDelim = '\0'; // cut off extension

		fields=FieldParm;
		tags=TagParm;

		fDeleted = 0; // flag to see if file needs to be packed

		init(&cb); // virtual function for setting CODE4
		open();
}

//////////////////////////////////////////////////////////////////////
DBBase::~DBBase()
{
		if( fDeleted)
			{
			d4pack(db);
			d4memo_compress(db);
			}

		d4close(db);
		  d4init_undo(&cb);	// NEW LINE ADDED
		delete []psFile;
}

//////////////////////////////////////////////////////////////////////
void DBBase::set(FIELD4 *field, char  byte)
{
char *str=" ";
		str[0]=byte;
		f4assign(field, str );
}


//////////////////////////////////////////////////////////////////////
void DBBase::set(FIELD4 *field, int num)
{
		sprintf(sIntBuf,"%d", num);
		f4assign(field, sIntBuf);
}


//////////////////////////////////////////////////////////////////////
void DBBase::set(FIELD4 *field, long num)
{
		sprintf(sIntBuf,"%d", num);
		f4assign(field, sIntBuf);
}


//////////////////////////////////////////////////////////////////////
void DBBase::set(char *field, char  byte)
{
char *str=" ";
		str[0]=byte;
		f4assign(d4field(db,field), str );
}


//////////////////////////////////////////////////////////////////////
void DBBase::set(char *field, int num)
{
		sprintf(sIntBuf,"%d",num);
		f4assign(d4field(db,field), sIntBuf);
}


//////////////////////////////////////////////////////////////////////
void DBBase::set(char *field, long num)
{
		sprintf(sIntBuf,"%d",num);
		f4assign(d4field(db,field), sIntBuf);
}


//////////////////////////////////////////////////////////////////////
void DBBase::go(long rec)
{
```

```
            if(rec<1)
                    {
                    rec=1;
                    d4go(db,rec);
                    }
            else if(rec>d4reccount(db))
                    {
                    rec=d4reccount(db)+1;
                    d4bottom(db);
                    d4skip(db,1L);
                    }
            else
                    d4go(db,rec);
}


////////////////////////////////////////////////////////////////////
void DBBase::open()
{
char *psFileName = new char[strlen(psFile)+5];
int fDatabase, fIndex;
            // check for dbase file and open/create
            strcpy(psFileName, psFile);
            strcat(psFileName, ".dbf");
            fDatabase = ! access(psFileName,0);

            // check for index file and open/create
            *(psFileName+strlen(psFile)) = '\0';
            strcat(psFileName, ".mdx");
            fIndex = ! access(psFileName,0);
            delete [] psFileName;


            // if database is gone, create new one with production index
            if( ! fDatabase)
                    db=d4create(&cb, psFile, fields, tags);
            else
                    {
                    if( ! fIndex)  // if index file is missing
                            cb.auto_open = 0; // dont auto open index because it aint there

                    db=d4open(&cb, psFile);

                    if( ! db)
                            return;

                    // if we have an open database, the tag information, and no index
                    if(tags && ! fIndex)
                                    i4create(db, NULL, tags); // create production index
                    }

            d4tag_select(db, d4tag_default(db));
            d4top(db);
}



////////////////////////////////////////////////////////////////////
// this is a virtual function that gets called from the constructor.
// if you want to set any of the CODE4 settings, override this function
// in your derived class, call the base function and then change the
// CODE4 settings.
void DBBase::init(CODE4 *cbstruct)
{
            d4init(cbstruct); // initialize codebase structure
}

int DBBase::relate(void)
 {
            relation = relate4init(db);
            if (relation)
```

```
              return 1;
            else
              return 0;
  }

 void DBBase::query(char * expr)
  {
            relate4query_set(relation,expr);
  }

 int DBBase::rtop(void)
  {
            return (relate4top(relation));
  }

 int DBBase::reof(int rc)
  {
            return (rc == r4eof);
  }

 int DBBase::rskip(long num)
  {
            return (relate4skip(relation,num));
  }

 void DBBase::rclose(void)
  {
            relate4unlock(relation);
            relate4free(relation,0);

  }
```

Dialog Class Files

```
#if !defined(__tcmpltpd_h)              // Sentry, use file only if it's not already included.
#define __tcmpltpd_h

/*  Project fellows
    Texas A&M University
    Copyright © 1994. All Rights Reserved.

    SUBSYSTEM:    fellows.apx Application
    FILE:         tcmpltpd.h
    AUTHOR:       Jason Thompson


    OVERVIEW
    ========
    Class definition for TCompletedPhaseDlg (TDialog).
*/

#include <owl\owlpch.h>
#pragma hdrstop


#include "fllwsapp.rh"            // Definition of all resources.


//{{TDialog = TCompletedPhaseDlg}}
class TCompletedPhaseDlg : public TDialog {
private:
  char                          cBuf[100];
            int                           cDonePhase;
            ProjectData         *cData;
            TEdit                         *MonthEditField;
            TEdit                         *DayEditField;
            TEdit                         *YearEditField;
            TEdit                         *PeopleEditField;
```

```
        TEdit                   *PhaseEditField;

public:
   TCompletedPhaseDlg (char *sBuf, int donephase, ProjectData *data, TWindow* parent, TResId resId = IDD_PHASE_COMPLETE,
TModule* module = 0);
   virtual ~TCompletedPhaseDlg ();

//{{TCompletedPhaseDlgVIRTUAL_BEGIN}}
public:
   virtual void SetupWindow ();
//{{TCompletedPhaseDlgVIRTUAL_END}}

//{{TCompletedPhaseDlgRSP_TBL_BEGIN}}
protected:
   void CmDecreaseDateHandler ();
   void CmIncreaseDateButtonHandler ();
   void CmOkButtonHandler ();
//{{TCompletedPhaseDlgRSP_TBL_END}}
DECLARE_RESPONSE_TABLE(TCompletedPhaseDlg);
};   //{{TCompletedPhaseDlg}}


#endif                          // __tcmpltpd_h sentry.




/*  Project fellows
   Texas A&M University
   Copyright © 1994. All Rights Reserved.

   SUBSYSTEM:    fellows.apx Application
   FILE:         tcmpltpd.cpp
   AUTHOR:       Jason Thompson


   OVERVIEW
   ========
   Source file for implementation of TCompletedPhaseDlg (TDialog).
*/

#include <owl\owlpch.h>
#pragma hdrstop

#include "tcmpltpd.h"
#include <time.h>


//
// Build a response table for all messages/commands handled
// by the application.
//
DEFINE_RESPONSE_TABLE1(TCompletedPhaseDlg, TDialog)
//{{TCompletedPhaseDlgRSP_TBL_BEGIN}}
   EV_BN_CLICKED(IDC_COMPLETE_BDOWN, CmDecreaseDateHandler),
           EV_BN_CLICKED(IDC_COMPLETE_BUP, CmIncreaseDateButtonHandler),
   EV_BN_CLICKED(IDOK, CmOkButtonHandler),
//{{TCompletedPhaseDlgRSP_TBL_END}}
END_RESPONSE_TABLE;


//{{TCompletedPhaseDlg Implementation}}


TCompletedPhaseDlg::TCompletedPhaseDlg (char *sBuf, int newphase, ProjectData *data, TWindow* parent, TResId resId, TModule*
module):
        TDialog(parent, resId, module)
{
        // INSERT>> Your constructor code here.
        strcpy(cBuf,sBuf);
```

```
		cDonePhase = newphase;
		switch(newphase)
		{
				case 1:	strcpy(cBuf, "Requirements Analyis and Definition"); break;
				case 2:	strcpy(cBuf, "System and Software Design"); break;
				case 3:	strcpy(cBuf, "Implementation and Unit Testing"); break;
				case 4:	strcpy(cBuf, "Integration and System Testing"); break;
		}

		cData=data;
		MonthEditField = new TEdit(this, IDC_COMPLETE_EMONTH);
		DayEditField = new TEdit(this, IDC_COMPLETE_EDAY);
		YearEditField = new TEdit(this, IDC_COMPLETE_EYEAR);
		PeopleEditField = new TEdit(this, IDC_COMPLETE_ENUMPEOPLE);
		PhaseEditField = new TEdit(this, IDC_COMPLETE_SPHASE);
}


TCompletedPhaseDlg::~TCompletedPhaseDlg ()
{
		Destroy();

		// INSERT>> Your destructor code here.
		delete MonthEditField;
		delete DayEditField;
		delete YearEditField;
		delete PeopleEditField;
		delete PhaseEditField;
}


void TCompletedPhaseDlg::SetupWindow ()
{
  TDialog::SetupWindow();

		// INSERT>> Your code here.
		time_t	now;
		char			sBuf[10];

		now = time(NULL);
		strftime(sBuf, 3, "%m", localtime(&now));
		MonthEditField->SetText(sBuf);
		strftime(sBuf, 3, "%d", localtime(&now));
		DayEditField->SetText(sBuf);
		strftime(sBuf, 3, "%y", localtime(&now));
		YearEditField->SetText(sBuf);

		PhaseEditField->SetText(cBuf);
}


void TCompletedPhaseDlg::CmDecreaseDateHandler ()
{
		// INSERT>> Your code here.
		struct tm	*tp;
		time_t			newtime;
		char				sBuf[10];

		tp = new struct tm;
		tp->tm_sec=0;
		tp->tm_min=0;
		tp->tm_hour=0;
		MonthEditField->GetLine(sBuf, 9, 1);
		tp->tm_mon=atoi(sBuf)-1;
		DayEditField->GetLine(sBuf, 9, 1);
		tp->tm_mday=atoi(sBuf);
		YearEditField->GetLine(sBuf, 9, 1);
		tp->tm_year=atoi(sBuf);

		tp->tm_mday-=1;
		newtime = mktime(tp);
		strftime(sBuf, 3, "%m", localtime(&newtime));
		MonthEditField->SetText(sBuf);
		strftime(sBuf, 3, "%d", localtime(&newtime));
		DayEditField->SetText(sBuf);
		strftime(sBuf, 3, "%y", localtime(&newtime));
		YearEditField->SetText(sBuf);
		delete tp;
}


void TCompletedPhaseDlg::CmIncreaseDateButtonHandler ()
{
		// INSERT>> Your code here.
		struct tm	*tp;
		time_t				newtime;
		char						sBuf[10];

		tp = new struct tm;
		tp->tm_sec=0;
		tp->tm_min=0;
		tp->tm_hour=0;
		MonthEditField->GetLine(sBuf, 9, 1);
		tp->tm_mon=atoi(sBuf)-1;
		DayEditField->GetLine(sBuf, 9, 1);
		tp->tm_mday=atoi(sBuf);
		YearEditField->GetLine(sBuf, 9, 1);
		tp->tm_year=atoi(sBuf);

		tp->tm_mday+=1;
		newtime = mktime(tp);

		strftime(sBuf, 3, "%m", localtime(&newtime));
		MonthEditField->SetText(sBuf);
		strftime(sBuf, 3, "%d", localtime(&newtime));
		DayEditField->SetText(sBuf);
		strftime(sBuf, 3, "%y", localtime(&newtime));
		YearEditField->SetText(sBuf);
		delete tp;
}


void TCompletedPhaseDlg::CmOkButtonHandler ()
{
		// INSERT>> Your code here.
		char sBuf1[10];
		char sBuf2[10];
		char sBuf3[10];
		char sBuf4[10];

		PeopleEditField->GetLine(sBuf1, 10, 1);
		MonthEditField->GetLine(sBuf2, 10, 1);
		DayEditField->GetLine(sBuf3, 10, 1);
		YearEditField->GetLine(sBuf4, 10, 1);

		switch(cDonePhase)
		{
				case 1:
						cData->Phase1Done=1;
						cData->Phase1Staff=atoi(sBuf1);
						cData->Phase1DoneMonth=atoi(sBuf2);
						cData->Phase1DoneDay=atoi(sBuf3);
						cData->Phase1DoneYear=atoi(sBuf4);
						break;
				case 2:
						cData->Phase2Done=1;
						cData->Phase2Staff=atoi(sBuf1);
						cData->Phase2DoneMonth=atoi(sBuf2);
						cData->Phase2DoneDay=atoi(sBuf3);
						cData->Phase2DoneYear=atoi(sBuf4);
						break;
```

```
                        case 3:
                                cData->Phase3Done=1;
                                cData->Phase3Staff=atoi(sBuf1);
                                cData->Phase3DoneMonth=atoi(sBuf2);
                                cData->Phase3DoneDay=atoi(sBuf3);
                                cData->Phase3DoneYear=atoi(sBuf4);
                                break;
                        case 4:
                                cData->Phase4Done=1;
                                cData->Phase4Staff=atoi(sBuf1);
                                cData->Phase4DoneMonth=atoi(sBuf2);
                                cData->Phase4DoneDay=atoi(sBuf3);
                                cData->Phase4DoneYear=atoi(sBuf4);
                                break;
                }

  CmOk();
}
```

```
#if !defined(__tcmptrpd_h)              // Sentry, use file only if it's not already included.
#define __tcmptrpd_h

/*  Project fellows
    Texas A&M University
    Copyright © 1994. All Rights Reserved.

    SUBSYSTEM:    fellows.apx Application
    FILE:         tcmptrpd.h
    AUTHOR:       Jason Thompson


    OVERVIEW
    ========
    Class definition for TComputerPredictionsDlg (TDialog).
*/

#include <owl\owlpch.h>
#pragma hdrstop


#include "fllwsapp.rh"          // Definition of all resources.


//{{TDialog = TComputerPredictionsDlg}}
class TComputerPredictionsDlg : public TDialog {
private:
  ProjectData *cData;
        TEdit                   *CurrentPhaseEditField;
        TEdit                   *Phase1EditField;
        TEdit                   *Phase2EditField;
        TEdit                   *Phase3EditField;
        TEdit                   *Phase4EditField;
        TEdit                   *MonthsEditField;

public:
    TComputerPredictionsDlg (ProjectData *data, TWindow* parent, TResId resId = IDD_ESTIMATES, TModule* module = 0);
    virtual ~TComputerPredictionsDlg ();

//{{TComputerPredictionsDlgVIRTUAL_BEGIN}}
public:
    virtual void SetupWindow ();
//{{TComputerPredictionsDlgVIRTUAL_END}}
};    //{{TComputerPredictionsDlg}}


#endif                          // __tcmptrpd_h sentry.
```

```
/* Project fellows
            Texas A&M University
            Copyright © 1994. All Rights Reserved.

            SUBSYSTEM:   fellows.apx Application
            FILE:        tcmptrpd.cpp
            AUTHOR:      Jason Thompson


            OVERVIEW
            ========
            Source file for implementation of TComputerPredictionsDlg (TDialog).
*/

#include <owl\owlpch.h>
#pragma hdrstop

#include "tcmptrpd.h"
#include "dbparam.h"
#include <math.h>
#include <time.h>


//{{TComputerPredictionsDlg Implementation}}


TComputerPredictionsDlg::TComputerPredictionsDlg (ProjectData *data, TWindow* parent, TResId resId, TModule* module):
            TDialog(parent, resId, module)
{
        // INSERT>> Your constructor code here.
        cData = data;
        CurrentPhaseEditField = new TEdit(this, IDC_PREDICT_ECURRENT);
        Phase1EditField = new TEdit(this, IDC_PREDICT_EPHASE1);
        Phase2EditField = new TEdit(this, IDC_PREDICT_EPHASE2);
        Phase3EditField = new TEdit(this, IDC_PREDICT_EPHASE3);
        Phase4EditField = new TEdit(this, IDC_PREDICT_EPHASE4);
        MonthsEditField = new TEdit(this, IDC_PREDICT_EMONTH);
}


TComputerPredictionsDlg::~TComputerPredictionsDlg ()
{
        Destroy();

        // INSERT>> Your destructor code here.
        delete CurrentPhaseEditField;
        delete Phase1EditField;
        delete Phase2EditField;
        delete Phase3EditField;
        delete Phase4EditField;
        delete MonthsEditField;
}


void TComputerPredictionsDlg::SetupWindow ()
{
        TDialog::SetupWindow();

        // INSERT>> Your code here.
        DBParam             *pdb;
        double      cocanswer, phasetime, timediff;
        char                        sBuf[100];
        struct tm   *tp1, *tp2;
        time_t              endtime, starttime;
        double              totaltime, phase1time, phase2time, phase3time, phase4time;
        int                         percent;
```

```
pdb = new DBParam();
switch(cData->ProjectType)
{
        case EMBEDDED:
                pdb->seek("EMBEDDED");
                break;
        case ORGANIC:
                pdb->seek("ORGANIC");
                break;
        case SEMIDETACHED:
                pdb->seek("SEMIDETACH");
                break;
}

cocanswer=pow(cData->EstimatedSize, atof(pdb->CocomoB()));
cocanswer*=atof(pdb->CocomoA());

switch (cData->IntParam1)
{
        case 1: cocanswer*=2; break;
        case 2: cocanswer*=1.5; break;
        case 4: cocanswer*=.75; break;
        case 5: cocanswer*=.5; break;
}
switch (cData->IntParam2)
{
        case 1: cocanswer*=2; break;
        case 2: cocanswer*=1.5; break;
        case 4: cocanswer*=.75; break;
        case 5: cocanswer*=.5; break;
}
switch (cData->IntParam3)
{
        case 1: cocanswer*=2; break;
        case 2: cocanswer*=1.5; break;
        case 4: cocanswer*=.75; break;
        case 5: cocanswer*=.5; break;
}
switch (cData->IntParam4)
{
        case 1: cocanswer*=2; break;
        case 2: cocanswer*=1.5; break;
        case 4: cocanswer*=.75; break;
        case 5: cocanswer*=.5; break;
}
switch (cData->IntParam5)
{
        case 1: cocanswer*=2; break;
        case 2: cocanswer*=1.5; break;
        case 4: cocanswer*=.75; break;
        case 5: cocanswer*=.5; break;
}
switch (cData->IntParam6)
{
        case 1: cocanswer*=2; break;
        case 2: cocanswer*=1.5; break;
        case 4: cocanswer*=.75; break;
        case 5: cocanswer*=.5; break;
}
switch (cData->IntParam7)
{
        case 1: cocanswer*=2; break;
        case 2: cocanswer*=1.5; break;
        case 4: cocanswer*=.75; break;
        case 5: cocanswer*=.5; break;
}
switch (cData->IntParam8)
{
        case 1: cocanswer*=2; break;
        case 2: cocanswer*=1.5; break;
```

```
		case 4: cocanswer*=.75; break;
		case 5: cocanswer*=.5; break;
	}
	switch (cData->IntParam9)
	{
		case 1: cocanswer*=2; break;
		case 2: cocanswer*=1.5; break;
		case 4: cocanswer*=.75; break;
		case 5: cocanswer*=.5; break;
	}
	switch (cData->IntParam10)
	{
		case 1: cocanswer*=2; break;
		case 2: cocanswer*=1.5; break;
		case 4: cocanswer*=.75; break;
		case 5: cocanswer*=.5; break;
	}
	switch (cData->IntParam11)
	{
		case 1: cocanswer*=2; break;
		case 2: cocanswer*=1.5; break;
		case 4: cocanswer*=.75; break;
		case 5: cocanswer*=.5; break;
	}
	switch (cData->IntParam12)
	{
		case 1: cocanswer*=2; break;
		case 2: cocanswer*=1.5; break;
		case 4: cocanswer*=.75; break;
		case 5: cocanswer*=.5; break;
	}
	switch (cData->IntParam13)
	{
		case 1: cocanswer*=2; break;
		case 2: cocanswer*=1.5; break;
		case 4: cocanswer*=.75; break;
		case 5: cocanswer*=.5; break;
	}
	switch (cData->IntParam14)
	{
		case 1: cocanswer*=2; break;
		case 2: cocanswer*=1.5; break;
		case 4: cocanswer*=.75; break;
		case 5: cocanswer*=.5; break;
	}
	switch (cData->IntParam15)
	{
		case 1: cocanswer*=2; break;
		case 2: cocanswer*=1.5; break;
		case 4: cocanswer*=.75; break;
		case 5: cocanswer*=.5; break;
	}

	sprintf(sBuf, "%6.2f", cocanswer);
	MonthsEditField->SetText(sBuf);
	cocanswer*=152;
	totaltime=0;

	if(cData->Phase1Done == 1)
	{
		tp1 = new struct tm;
		tp1->tm_sec=0;
		tp1->tm_min=0;
		tp1->tm_hour=0;
		tp1->tm_mon=cData->Phase1DoneMonth-1;
		tp1->tm_mday=cData->Phase1DoneDay;
		tp1->tm_year=cData->Phase1DoneYear;
		endtime = mktime(tp1);
		delete tp1;

		tp2 = new struct tm;
		tp2->tm_sec=0;
		tp2->tm_min=0;
		tp2->tm_hour=0;
		tp2->tm_mon=cData->StartDateMonth-1;
		tp2->tm_mday=cData->StartDateDay;
		tp2->tm_year=cData->StartDateYear;
		starttime = mktime(tp2);
		delete tp2;

		timediff = difftime(endtime, starttime);
		timediff/=360;
		phase1time=timediff;
		phasetime=timediff;
		phasetime/=cData->Phase1Staff;
		phasetime/=152;
		sprintf(sBuf, "%6.2f", phasetime);
		Phase1EditField->SetText(sBuf);
}
else
{
		CurrentPhaseEditField->SetText("Requirements Analysis and Definition");
		phasetime=cocanswer*pdb->PercentPhase1()/100;
		phasetime/=cData->Phase1Staff;
		phasetime/=152;
		sprintf(sBuf, "%6.2f", phasetime);
		Phase1EditField->SetText(sBuf);
}
if(cData->Phase2Done == 1)
{
		tp1 = new struct tm;
		tp1->tm_sec=0;
		tp1->tm_min=0;
		tp1->tm_hour=0;
		tp1->tm_mon=cData->Phase2DoneMonth-1;
		tp1->tm_mday=cData->Phase2DoneDay;
		tp1->tm_year=cData->Phase2DoneYear;
		endtime = mktime(tp1);
		delete tp1;

		tp2 = new struct tm;
		tp2->tm_sec=0;
		tp2->tm_min=0;
		tp2->tm_hour=0;
		tp2->tm_mon=cData->Phase1DoneMonth-1;
		tp2->tm_mday=cData->Phase1DoneDay;
		tp2->tm_year=cData->Phase1DoneYear;
		starttime = mktime(tp2);
		delete tp2;

		timediff = difftime(endtime, starttime);
		timediff/=360;
		phase2time=timediff;
		phasetime=timediff;
		phasetime/=cData->Phase2Staff;
		phasetime/=152;
		sprintf(sBuf, "%6.2f", phasetime);
		Phase2EditField->SetText(sBuf);
}
else
{
		if(cData->Phase1Done == 1)
				CurrentPhaseEditField->SetText("System and Software Design");
		phasetime=cocanswer*pdb->PercentPhase2()/100;
		phasetime/=cData->Phase2Staff;
		phasetime/=152;
		sprintf(sBuf, "%6.2f", phasetime);
		Phase2EditField->SetText(sBuf);
}

if(cData->Phase3Done == 1)
{
		tp1 = new struct tm;
		tp1->tm_sec=0;
		tp1->tm_min=0;
		tp1->tm_hour=0;
		tp1->tm_mon=cData->Phase3DoneMonth-1;
		tp1->tm_mday=cData->Phase3DoneDay;
		tp1->tm_year=cData->Phase3DoneYear;
		endtime = mktime(tp1);
		delete tp1;

		tp2 = new struct tm;
		tp2->tm_sec=0;
		tp2->tm_min=0;
		tp2->tm_hour=0;
		tp2->tm_mon=cData->Phase2DoneMonth-1;
		tp2->tm_mday=cData->Phase2DoneDay;
		tp2->tm_year=cData->Phase2DoneYear;
		starttime = mktime(tp2);
		delete tp2;

		timediff = difftime(endtime, starttime);
		timediff/=360;
		phase3time=timediff;
		phasetime=timediff;
		phasetime/=cData->Phase3Staff;
		phasetime/=152;
		sprintf(sBuf, "%6.2f", phasetime);
		Phase3EditField->SetText(sBuf);
	}
	else
	{
		if(cData->Phase2Done == 1)
			CurrentPhaseEditField->SetText("Implementation and Unit Testing");
		phasetime=cocanswer*pdb->PercentPhase3()/100;
		phasetime/=cData->Phase3Staff;
		phasetime/=152;
		sprintf(sBuf, "%6.2f", phasetime);
		Phase3EditField->SetText(sBuf);
	}

	if(cData->Phase4Done == 1)
	{
		tp1 = new struct tm;
		tp1->tm_sec=0;
		tp1->tm_min=0;
		tp1->tm_hour=0;
		tp1->tm_mon=cData->Phase4DoneMonth-1;
		tp1->tm_mday=cData->Phase4DoneDay;
		tp1->tm_year=cData->Phase4DoneYear;
		endtime = mktime(tp1);
		delete tp1;

		tp2 = new struct tm;
		tp2->tm_sec=0;
		tp2->tm_min=0;
		tp2->tm_hour=0;
		tp2->tm_mon=cData->Phase3DoneMonth-1;
		tp2->tm_mday=cData->Phase3DoneDay;
		tp2->tm_year=cData->Phase3DoneYear;
		starttime = mktime(tp2);
		delete tp2;

		timediff = difftime(endtime, starttime);
		timediff/=360;
		phase4time=timediff;
		phasetime=timediff;
		phasetime/=cData->Phase4Staff;
		phasetime/=152;
		sprintf(sBuf, "%6.2f", phasetime);
		Phase4EditField->SetText(sBuf);
```

```
				//update percentages
				totaltime=phase1time+phase2time+phase3time+phase4time;
				percent=100;
				sprintf(sBuf, "%2.0f", 100*phase1time/totaltime);
				percent-=100*phase1time/totaltime;
				pdb->PercentPhase1(sBuf);
				sprintf(sBuf, "%2.0f", 100*phase2time/totaltime);
				percent-=100*phase2time/totaltime;
				pdb->PercentPhase2(sBuf);
				sprintf(sBuf, "%2.0f", 100*phase3time/totaltime);
				percent-=100*phase3time/totaltime;
				pdb->PercentPhase3(sBuf);
				sprintf(sBuf, "%2d", percent);
				pdb->PercentPhase4(sBuf);
		}
		else
		{
				if(cData->Phase3Done == 1)
					CurrentPhaseEditField->SetText("Integration and System Testing");
				phasetime=cocanswer*pdb->PercentPhase4()/100;
				phasetime/=cData->Phase4Staff;
				phasetime/=152;
				sprintf(sBuf, "%6.2f", phasetime);
				Phase4EditField->SetText(sBuf);
		}

		delete pdb;
}




#if !defined(__testmtsd_h)		// Sentry, use file only if it's not already included.
#define __testmtsd_h

/*  Project fellows
   Texas A&M University
   Copyright © 1994. All Rights Reserved.

   SUBSYSTEM:    fellows.apx Application
   FILE:         testmtsd.h
		AUTHOR:       Jason Thompson


   OVERVIEW
   ========
   Class definition for TEstimatedStaffingDlg (TDialog).
*/

#include <owl\owlpch.h>
#pragma hdrstop


#include "fllwsapp.rh"			// Definition of all resources.


//{{TDialog = TEstimatedStaffingDlg}}
class TEstimatedStaffingDlg : public TDialog {
private:
  ProjectData			*cData;
		TEdit					*Phase1EditField;
		TEdit					*Phase2EditField;
		TEdit					*Phase3EditField;
		TEdit					*Phase4EditField;

public:
   TEstimatedStaffingDlg (ProjectData *data, TWindow* parent, TResId resId = IDD_ESTIMATED_STAFFING, TModule* module =
0);
		virtual ~TEstimatedStaffingDlg ();
```

```
//{{TEstimatedStaffingDlgRSP_TBL_BEGIN}}
protected:
   void CmOkButtonHandler ();
//{{TEstimatedStaffingDlgRSP_TBL_END}}
DECLARE_RESPONSE_TABLE(TEstimatedStaffingDlg);
};   //{{TEstimatedStaffingDlg}}


#endif                              // __testmtsd_h sentry.




/*  Project fellows
   Texas A&M University
   Copyright © 1994. All Rights Reserved.

   SUBSYSTEM:    fellows.apx Application
   FILE:         testmtsd.cpp
   AUTHOR:       Jason Thompson


   OVERVIEW
            ========
   Source file for implementation of TEstimatedStaffingDlg (TDialog).
*/

#include <owl\owlpch.h>
#pragma hdrstop

#include "testmtsd.h"


//
// Build a response table for all messages/commands handled
// by the application.
//
DEFINE_RESPONSE_TABLE1(TEstimatedStaffingDlg, TDialog)
//{{TEstimatedStaffingDlgRSP_TBL_BEGIN}}
   EV_BN_CLICKED(IDOK, CmOkButtonHandler),
//{{TEstimatedStaffingDlgRSP_TBL_END}}
END_RESPONSE_TABLE;


//{{TEstimatedStaffingDlg Implementation}}


TEstimatedStaffingDlg::TEstimatedStaffingDlg (ProjectData *data, TWindow* parent, TResId resId, TModule* module):
            TDialog(parent, resId, module)
{
            // INSERT>> Your constructor code here.
            cData = data;

            Phase1EditField = new TEdit(this, IDC_STAFF_EPHASE1);
            Phase2EditField = new TEdit(this, IDC_STAFF_EPHASE2);
            Phase3EditField = new TEdit(this, IDC_STAFF_EPHASE3);
            Phase4EditField = new TEdit(this, IDC_STAFF_EPHASE4);
}


TEstimatedStaffingDlg::~TEstimatedStaffingDlg ()
{
            Destroy();

            // INSERT>> Your destructor code here.
            delete Phase1EditField;
            delete Phase2EditField;
            delete Phase3EditField;
            delete Phase4EditField;
}
```

```
void TEstimatedStaffingDlg::CmOkButtonHandler ()
{
            // INSERT>> Your code here.
            char sBuf[10];

            Phase1EditField->GetLine(sBuf, 10, 1);
            cData->Phase1Staff=atoi(sBuf);
            cData->Phase1Done=0;
            Phase2EditField->GetLine(sBuf, 10, 1);
            cData->Phase2Staff=atoi(sBuf);
            cData->Phase2Done=0;
            Phase3EditField->GetLine(sBuf, 10, 1);
            cData->Phase3Staff=atoi(sBuf);
            cData->Phase3Done=0;
            Phase4EditField->GetLine(sBuf, 10, 1);
            cData->Phase4Staff=atoi(sBuf);
            cData->Phase4Done=0;

  CmOk();
}
```

```
#if !defined(__tdcmntsd_h)              // Sentry, use file only if it's not already included.
#define __tdcmntsd_h

/*  Project fellows
            Texas A&M University
   Copyright © 1994. All Rights Reserved.

   SUBSYSTEM:    fellows.apx Application
   FILE:         tdcmntsd.h
   AUTHOR:       Jason Thompson


   OVERVIEW
   ========
   Class definition for TDocumentsDlg (TDialog).
*/

#include <owl\owlpch.h>
#pragma hdrstop


#include "fllwsapp.rh"          // Definition of all resources.


//{{TDialog = TDocumentsDlg}}
class TDocumentsDlg : public TDialog {
private:
  ProjectData            *cData;

            TRadioButton            *Doc1NotRadioButton;
            TRadioButton            *Doc1StartRadioButton;
            TRadioButton            *Doc1DoneRadioButton;

            TRadioButton            *Doc2NotRadioButton;
            TRadioButton            *Doc2StartRadioButton;
            TRadioButton            *Doc2DoneRadioButton;

            TRadioButton            *Doc3NotRadioButton;
            TRadioButton            *Doc3StartRadioButton;
            TRadioButton            *Doc3DoneRadioButton;

            TRadioButton            *Doc4NotRadioButton;
            TRadioButton            *Doc4StartRadioButton;
            TRadioButton            *Doc4DoneRadioButton;
            TRadioButton        *Doc5NotRadioButton;
            TRadioButton        *Doc5StartRadioButton;
            TRadioButton        *Doc5DoneRadioButton;

            TRadioButton        *Doc6NotRadioButton;
            TRadioButton        *Doc6StartRadioButton;
            TRadioButton        *Doc6DoneRadioButton;

            TRadioButton        *Doc7NotRadioButton;
            TRadioButton        *Doc7StartRadioButton;
            TRadioButton        *Doc7DoneRadioButton;

            TRadioButton        *Doc8NotRadioButton;
            TRadioButton        *Doc8StartRadioButton;
            TRadioButton        *Doc8DoneRadioButton;

            TRadioButton        *Doc9NotRadioButton;
            TRadioButton        *Doc9StartRadioButton;
            TRadioButton        *Doc9DoneRadioButton;

            TRadioButton        *Doc10NotRadioButton;
            TRadioButton        *Doc10StartRadioButton;
            TRadioButton        *Doc10DoneRadioButton;

            TRadioButton        *Doc11NotRadioButton;
            TRadioButton        *Doc11StartRadioButton;
            TRadioButton        *Doc11DoneRadioButton;

            TRadioButton        *Doc12NotRadioButton;
            TRadioButton        *Doc12StartRadioButton;
            TRadioButton        *Doc12DoneRadioButton;

            TRadioButton        *Doc13NotRadioButton;
            TRadioButton        *Doc13StartRadioButton;
            TRadioButton        *Doc13DoneRadioButton;

public:
            TDocumentsDlg (ProjectData *data, TWindow* parent, TResId resId = IDD_KNOWNS_DOCUMENTS, TModule* module
= 0);
   virtual ~TDocumentsDlg ();

//{{TDocumentsDlgRSP_TBL_BEGIN}}
protected:
   void CmOkButtonHandler ();
//{{TDocumentsDlgRSP_TBL_END}}
DECLARE_RESPONSE_TABLE(TDocumentsDlg);
};   //{{TDocumentsDlg}}


#endif                          // __tdcmntsd_h sentry.




/*  Project fellows
   Texas A&M University
   Copyright © 1994. All Rights Reserved.

   SUBSYSTEM:    fellows.apx Application
   FILE:         tdcmntsd.cpp
   AUTHOR:       Jason Thompson


            OVERVIEW
   ========
   Source file for implementation of TDocumentsDlg (TDialog).
*/

#include <owl\owlpch.h>
```

```
#pragma hdrstop

#include "tdcmntsd.h"


//
// Build a response table for all messages/commands handled
// by the application.
//
DEFINE_RESPONSE_TABLE1(TDocumentsDlg, TDialog)
//{{TDocumentsDlgRSP_TBL_BEGIN}}
   EV_BN_CLICKED(IDOK, CmOkButtonHandler),
//{{TDocumentsDlgRSP_TBL_END}}
END_RESPONSE_TABLE;


//{{TDocumentsDlg Implementation}}


TDocumentsDlg::TDocumentsDlg (ProjectData *data, TWindow* parent, TResId resId, TModule* module):
                TDialog(parent, resId, module)
{
            // INSERT>> Your constructor code here.
            cData = data;

            Doc1NotRadioButton = new TRadioButton(this, IDC_DOC1_NOT);
            Doc1StartRadioButton = new TRadioButton(this, IDC_DOC1_START);
            Doc1DoneRadioButton = new TRadioButton(this, IDC_DOC1_DONE);

            Doc2NotRadioButton = new TRadioButton(this, IDC_DOC2_NOT);
            Doc2StartRadioButton = new TRadioButton(this, IDC_DOC2_START);
            Doc2DoneRadioButton = new TRadioButton(this, IDC_DOC2_DONE);

            Doc3NotRadioButton = new TRadioButton(this, IDC_DOC3_NOT);
            Doc3StartRadioButton = new TRadioButton(this, IDC_DOC3_START);
            Doc3DoneRadioButton = new TRadioButton(this, IDC_DOC3_DONE);

            Doc4NotRadioButton = new TRadioButton(this, IDC_DOC4_NOT);
            Doc4StartRadioButton = new TRadioButton(this, IDC_DOC4_START);
            Doc4DoneRadioButton = new TRadioButton(this, IDC_DOC4_DONE);

            Doc5NotRadioButton = new TRadioButton(this, IDC_DOC5_NOT);
            Doc5StartRadioButton = new TRadioButton(this, IDC_DOC5_START);
            Doc5DoneRadioButton = new TRadioButton(this, IDC_DOC5_DONE);

            Doc6NotRadioButton = new TRadioButton(this, IDC_DOC6_NOT);
            Doc6StartRadioButton = new TRadioButton(this, IDC_DOC6_START);
            Doc6DoneRadioButton = new TRadioButton(this, IDC_DOC6_DONE);

            Doc7NotRadioButton = new TRadioButton(this, IDC_DOC7_NOT);
            Doc7StartRadioButton = new TRadioButton(this, IDC_DOC7_START);
            Doc7DoneRadioButton = new TRadioButton(this, IDC_DOC7_DONE);

            Doc8NotRadioButton = new TRadioButton(this, IDC_DOC8_NOT);
            Doc8StartRadioButton = new TRadioButton(this, IDC_DOC8_START);
            Doc8DoneRadioButton = new TRadioButton(this, IDC_DOC8_DONE);

            Doc9NotRadioButton = new TRadioButton(this, IDC_DOC9_NOT);
            Doc9StartRadioButton = new TRadioButton(this, IDC_DOC9_START);
            Doc9DoneRadioButton = new TRadioButton(this, IDC_DOC9_DONE);

            Doc10NotRadioButton = new TRadioButton(this, IDC_DOC10_NOT);
            Doc10StartRadioButton = new TRadioButton(this, IDC_DOC10_START);
            Doc10DoneRadioButton = new TRadioButton(this, IDC_DOC10_DONE);

            Doc11NotRadioButton = new TRadioButton(this, IDC_DOC11_NOT);
            Doc11StartRadioButton = new TRadioButton(this, IDC_DOC11_START);
            Doc11DoneRadioButton = new TRadioButton(this, IDC_DOC11_DONE);

            Doc12NotRadioButton = new TRadioButton(this, IDC_DOC12_NOT);
            Doc12StartRadioButton = new TRadioButton(this, IDC_DOC12_START);
            Doc12DoneRadioButton = new TRadioButton(this, IDC_DOC12_DONE);

            Doc13NotRadioButton = new TRadioButton(this, IDC_DOC13_NOT);
            Doc13StartRadioButton = new TRadioButton(this, IDC_DOC13_START);
            Doc13DoneRadioButton = new TRadioButton(this, IDC_DOC13_DONE);
}


TDocumentsDlg::~TDocumentsDlg ()
{
            Destroy();

            // INSERT>> Your destructor code here.

}


void TDocumentsDlg::CmOkButtonHandler ()
{
            // INSERT>> Your code here.
            if (Doc1NotRadioButton->GetCheck()) cData->Document1=0;
            else if (Doc1StartRadioButton->GetCheck()) cData->Document1=1;
            else if (Doc1DoneRadioButton->GetCheck()) cData->Document1=2;

            if (Doc2NotRadioButton->GetCheck()) cData->Document2=0;
            else if (Doc2StartRadioButton->GetCheck()) cData->Document2=1;
            else if (Doc2DoneRadioButton->GetCheck()) cData->Document2=2;

            if (Doc3NotRadioButton->GetCheck()) cData->Document3=0;
            else if (Doc3StartRadioButton->GetCheck()) cData->Document3=1;
            else if (Doc3DoneRadioButton->GetCheck()) cData->Document3=2;

            if (Doc4NotRadioButton->GetCheck()) cData->Document4=0;
            else if (Doc4StartRadioButton->GetCheck()) cData->Document4=1;
            else if (Doc4DoneRadioButton->GetCheck()) cData->Document4=2;

            if (Doc5NotRadioButton->GetCheck()) cData->Document5=0;
            else if (Doc5StartRadioButton->GetCheck()) cData->Document5=1;
            else if (Doc5DoneRadioButton->GetCheck()) cData->Document5=2;

            if (Doc6NotRadioButton->GetCheck()) cData->Document6=0;
            else if (Doc6StartRadioButton->GetCheck()) cData->Document6=1;
            else if (Doc6DoneRadioButton->GetCheck()) cData->Document6=2;

            if (Doc7NotRadioButton->GetCheck()) cData->Document7=0;
            else if (Doc7StartRadioButton->GetCheck()) cData->Document7=1;
            else if (Doc7DoneRadioButton->GetCheck()) cData->Document7=2;

            if (Doc8NotRadioButton->GetCheck()) cData->Document8=0;
            else if (Doc8StartRadioButton->GetCheck()) cData->Document8=1;
            else if (Doc8DoneRadioButton->GetCheck()) cData->Document8=2;

            if (Doc9NotRadioButton->GetCheck()) cData->Document9=0;
            else if (Doc9StartRadioButton->GetCheck()) cData->Document9=1;
            else if (Doc9DoneRadioButton->GetCheck()) cData->Document9=2;

            if (Doc10NotRadioButton->GetCheck()) cData->Document10=0;
            else if (Doc10StartRadioButton->GetCheck()) cData->Document10=1;
            else if (Doc10DoneRadioButton->GetCheck()) cData->Document10=2;

            if (Doc11NotRadioButton->GetCheck()) cData->Document11=0;
            else if (Doc11StartRadioButton->GetCheck()) cData->Document11=1;
            else if (Doc11DoneRadioButton->GetCheck()) cData->Document11=2;

            if (Doc12NotRadioButton->GetCheck()) cData->Document12=0;
            else if (Doc12StartRadioButton->GetCheck()) cData->Document12=1;
            else if (Doc12DoneRadioButton->GetCheck()) cData->Document12=2;

            if (Doc13NotRadioButton->GetCheck()) cData->Document13=0;
            else if (Doc13StartRadioButton->GetCheck()) cData->Document13=1;
            else if (Doc13DoneRadioButton->GetCheck()) cData->Document13=2;
```

```
  CmOk();
}




#if !defined(__tstrtdtd_h)            // Sentry, use file only if it's not already included.
#define __tstrtdtd_h

/*  Project fellows
    Texas A&M University
    Copyright © 1994. All Rights Reserved.

    SUBSYSTEM:    fellows.apx Application
    FILE:         tstrtdtd.h
    AUTHOR:       Jason Thompson


    OVERVIEW
    ========
    Class definition for TStartDateDlg (TDialog).
*/

#include <owl\owlpch.h>
#pragma hdrstop


#include "fllwsapp.rh"          // Definition of all resources.


//{{TDialog = TStartDateDlg}}
class TStartDateDlg : public TDialog {
private:
  ProjectData           *cData;
        TEdit                           *DayEditField;
        TEdit                           *MonthEditField;
        TEdit                           *YearEditField;

public:
    TStartDateDlg (ProjectData *data, TWindow* parent, TResId resId = IDD_KNOWNS_STARTDATE, TModule* module = 0);
    virtual ~TStartDateDlg ();

//{{TStartDateDlgVIRTUAL_BEGIN}}
public:
    virtual void SetupWindow ();
//{{TStartDateDlgVIRTUAL_END}}

//{{TStartDateDlgRSP_TBL_BEGIN}}
protected:
    void CmDecreaseDateButtonHandler ();
    void CmIncreaseDateButtonHandler ();
    void CmOkButtonHandler ();
//{{TStartDateDlgRSP_TBL_END}}
DECLARE_RESPONSE_TABLE(TStartDateDlg);
};    //{{TStartDateDlg}}


#endif                          // __tstrtdtd_h sentry.




/*  Project fellows
        Texas A&M University
        Copyright © 1994. All Rights Reserved.

        SUBSYSTEM:    fellows.apx Application
        FILE:         tstrtdtd.cpp
```

```
            AUTHOR:     Jason Thompson


            OVERVIEW
            ========
            Source file for implementation of TStartDateDlg (TDialog).
*/

#include <owl\owlpch.h>
#pragma hdrstop

#include "tstrtdtd.h"
#include <time.h>


//
// Build a response table for all messages/commands handled
// by the application.
//
DEFINE_RESPONSE_TABLE1(TStartDateDlg, TDialog)
//{{TStartDateDlgRSP_TBL_BEGIN}}
  EV_BN_CLICKED(IDC_KNOWNS_STARTDATE_BDOWN, CmDecreaseDateButtonHandler),
  EV_BN_CLICKED(IDC_KNOWNS_STARTDATE_BUP, CmIncreaseDateButtonHandler),
  EV_BN_CLICKED(IDOK, CmOkButtonHandler),
//{{TStartDateDlgRSP_TBL_END}}
END_RESPONSE_TABLE;


//{{TStartDateDlg Implementation}}


TStartDateDlg::TStartDateDlg (ProjectData *data, TWindow* parent, TResId resId, TModule* module):
            TDialog(parent, resId, module)
{
            // INSERT>> Your constructor code here.
            cData = data;
            DayEditField = new TEdit(this, IDC_KNOWNS_STARTDATE_EDAY);
            MonthEditField = new TEdit(this, IDC_KNOWNS_STARTDATE_EMONTH);
            YearEditField = new TEdit(this, IDC_KNOWNS_STARTDATE_EYEAR);
}


TStartDateDlg::~TStartDateDlg ()
{
            Destroy();

            // INSERT>> Your destructor code here.
            delete DayEditField;
            delete MonthEditField;
  delete YearEditField;

}


void TStartDateDlg::SetupWindow ()
{
            TDialog::SetupWindow();

            // INSERT>> Your code here.
            time_t      now;
            char                    sBuf[10];

            now = time(NULL);
            strftime(sBuf, 3, "%m", localtime(&now));
            MonthEditField->SetText(sBuf);
            strftime(sBuf, 3, "%d", localtime(&now));
            DayEditField->SetText(sBuf);
            strftime(sBuf, 3, "%y", localtime(&now));
            YearEditField->SetText(sBuf);
}
```

```
void TStartDateDlg::CmDecreaseDateButtonHandler ()
{
        // INSERT>> Your code here.
        struct tm    *tp;
        time_t                  newtime;
        char                            sBuf[10];

        tp = new struct tm;
        tp->tm_sec=0;
        tp->tm_min=0;
        tp->tm_hour=0;
        MonthEditField->GetLine(sBuf, 9, 1);
        tp->tm_mon=atoi(sBuf)-1;
        DayEditField->GetLine(sBuf, 9, 1);
        tp->tm_mday=atoi(sBuf);
        YearEditField->GetLine(sBuf, 9, 1);
        tp->tm_year=atoi(sBuf);

        tp->tm_mday-=1;
        newtime = mktime(tp);

        strftime(sBuf, 3, "%m", localtime(&newtime));
        MonthEditField->SetText(sBuf);
        strftime(sBuf, 3, "%d", localtime(&newtime));
        DayEditField->SetText(sBuf);
        strftime(sBuf, 3, "%y", localtime(&newtime));
        YearEditField->SetText(sBuf);
        delete tp;
}

void TStartDateDlg::CmIncreaseDateButtonHandler ()
{
        // INSERT>> Your code here.
        struct tm    *tp;
        time_t                  newtime;
        char                            sBuf[10];

        tp = new struct tm;
        tp->tm_sec=0;
        tp->tm_min=0;
        tp->tm_hour=0;
        MonthEditField->GetLine(sBuf, 9, 1);
        tp->tm_mon=atoi(sBuf)-1;
        DayEditField->GetLine(sBuf, 9, 1);
        tp->tm_mday=atoi(sBuf);
        YearEditField->GetLine(sBuf, 9, 1);
        tp->tm_year=atoi(sBuf);

        tp->tm_mday+=1;
        newtime = mktime(tp);

        strftime(sBuf, 3, "%m", localtime(&newtime));
        MonthEditField->SetText(sBuf);
        strftime(sBuf, 3, "%d", localtime(&newtime));
        DayEditField->SetText(sBuf);
        strftime(sBuf, 3, "%y", localtime(&newtime));
        YearEditField->SetText(sBuf);
        delete tp;
}

void TStartDateDlg::CmOkButtonHandler ()
{
        // INSERT>> Your code here.
        char        sBuf[10];

        MonthEditField->GetLine(sBuf, 9, 1);
        cData->StartDateMonth=atoi(sBuf);
        DayEditField->GetLine(sBuf, 9, 1);
```

```
            cData->StartDateDay=atoi(sBuf);
            YearEditField->GetLine(sBuf, 9, 1);
            cData->StartDateYear=atoi(sBuf);

  CmOk();
}




#if !defined(__tintrcpd_h)           // Sentry, use file only if it's not already included.
#define __tintrcpd_h

/*  Project fellows
   Texas A&M University
            Copyright © 1994. All Rights Reserved.

   SUBSYSTEM:    fellows.apx Application
   FILE:         tintrcpd.h
   AUTHOR:       Jason Thompson


   OVERVIEW
   ========
   Class definition for TIntermediateCocomoParameterDlg (TDialog).
*/

#include <owl\owlpch.h>
#pragma hdrstop


#include "fllwsapp.rh"          // Definition of all resources.


//{{TDialog = TIntermediateCocomoParameterDlg}}
class TIntermediateCocomoParameterDlg : public TDialog {
private:
  ProjectData             *cData;
            TRadioButton          *Param1PoorRadioButton;
            TRadioButton          *Param1BadRadioButton;
            TRadioButton          *Param1NormalRadioButton;
            TRadioButton          *Param1GoodRadioButton;
            TRadioButton          *Param1GreatRadioButton;

            TRadioButton          *Param2PoorRadioButton;
            TRadioButton          *Param2BadRadioButton;
            TRadioButton          *Param2NormalRadioButton;
            TRadioButton          *Param2GoodRadioButton;
            TRadioButton          *Param2GreatRadioButton;

            TRadioButton          *Param3PoorRadioButton;
            TRadioButton          *Param3BadRadioButton;
            TRadioButton          *Param3NormalRadioButton;
            TRadioButton          *Param3GoodRadioButton;
            TRadioButton          *Param3GreatRadioButton;

            TRadioButton          *Param4PoorRadioButton;
            TRadioButton          *Param4BadRadioButton;
            TRadioButton          *Param4NormalRadioButton;
            TRadioButton          *Param4GoodRadioButton;
            TRadioButton          *Param4GreatRadioButton;

            TRadioButton          *Param5PoorRadioButton;
            TRadioButton          *Param5BadRadioButton;
            TRadioButton          *Param5NormalRadioButton;
            TRadioButton          *Param5GoodRadioButton;
            TRadioButton          *Param5GreatRadioButton;

            TRadioButton          *Param6PoorRadioButton;
            TRadioButton          *Param6BadRadioButton;
```

```
        TRadioButton        *Param6NormalRadioButton;
        TRadioButton        *Param6GoodRadioButton;
        TRadioButton        *Param6GreatRadioButton;

        TRadioButton        *Param7PoorRadioButton;
        TRadioButton        *Param7BadRadioButton;
        TRadioButton        *Param7NormalRadioButton;
        TRadioButton        *Param7GoodRadioButton;
        TRadioButton        *Param7GreatRadioButton;

        TRadioButton        *Param8PoorRadioButton;
        TRadioButton        *Param8BadRadioButton;
        TRadioButton        *Param8NormalRadioButton;
        TRadioButton        *Param8GoodRadioButton;
        TRadioButton        *Param8GreatRadioButton;

        TRadioButton        *Param9PoorRadioButton;
        TRadioButton        *Param9BadRadioButton;
        TRadioButton        *Param9NormalRadioButton;
        TRadioButton        *Param9GoodRadioButton;
        TRadioButton        *Param9GreatRadioButton;

        TRadioButton        *Param10PoorRadioButton;
        TRadioButton        *Param10BadRadioButton;
        TRadioButton        *Param10NormalRadioButton;
        TRadioButton        *Param10GoodRadioButton;
        TRadioButton        *Param10GreatRadioButton;

        TRadioButton        *Param11PoorRadioButton;
        TRadioButton        *Param11BadRadioButton;
        TRadioButton        *Param11NormalRadioButton;
        TRadioButton        *Param11GoodRadioButton;
        TRadioButton        *Param11GreatRadioButton;

        TRadioButton        *Param12PoorRadioButton;
        TRadioButton        *Param12BadRadioButton;
        TRadioButton        *Param12NormalRadioButton;
        TRadioButton        *Param12GoodRadioButton;
        TRadioButton        *Param12GreatRadioButton;

        TRadioButton        *Param13PoorRadioButton;
        TRadioButton        *Param13BadRadioButton;
        TRadioButton        *Param13NormalRadioButton;
        TRadioButton        *Param13GoodRadioButton;
        TRadioButton        *Param13GreatRadioButton;

        TRadioButton        *Param14PoorRadioButton;
        TRadioButton        *Param14BadRadioButton;
        TRadioButton        *Param14NormalRadioButton;
        TRadioButton        *Param14GoodRadioButton;
        TRadioButton        *Param14GreatRadioButton;

        TRadioButton        *Param15PoorRadioButton;
        TRadioButton        *Param15BadRadioButton;
        TRadioButton        *Param15NormalRadioButton;
        TRadioButton        *Param15GoodRadioButton;
        TRadioButton        *Param15GreatRadioButton;

public:
    TIntermediateCocomoParameterDlg (ProjectData *data, TWindow* parent, TResId resId = IDD_INTERMEDIATE_COCOMO,
TModule* module = 0);
    virtual ~TIntermediateCocomoParameterDlg ();

//{{TIntermediateCocomoParameterDlgRSP_TBL_BEGIN}}
protected:
    void CmOkButtonHandler ();
//{{TIntermediateCocomoParameterDlgRSP_TBL_END}}
DECLARE_RESPONSE_TABLE(TIntermediateCocomoParameterDlg);
};    //{{TIntermediateCocomoParameterDlg}}
```

```
#endif                          // __tintrcpd_h sentry.




/*  Project fellows
    Texas A&M University
    Copyright © 1994. All Rights Reserved.

    SUBSYSTEM:    fellows.apx Application
    FILE:         tintrcpd.cpp
    AUTHOR:       Jason Thompson


    OVERVIEW
    ========
            Source file for implementation of TIntermediateCocomoParameterDlg (TDialog).
*/

#include <owl\owlpch.h>
#pragma hdrstop

#include "tintrcpd.h"


//
// Build a response table for all messages/commands handled
// by the application.
//
DEFINE_RESPONSE_TABLE1(TIntermediateCocomoParameterDlg, TDialog)
//{{TIntermediateCocomoParameterDlgRSP_TBL_BEGIN}}
    EV_BN_CLICKED(IDOK, CmOkButtonHandler),
//{{TIntermediateCocomoParameterDlgRSP_TBL_END}}
END_RESPONSE_TABLE;


//{{TIntermediateCocomoParameterDlg Implementation}}


TIntermediateCocomoParameterDlg::TIntermediateCocomoParameterDlg (ProjectData *data, TWindow* parent, TResId resId,
TModule* module):
            TDialog(parent, resId, module)
{
            // INSERT>> Your constructor code here.
            Param1PoorRadioButton = new TRadioButton(this, IDC_PARAM1_POOR);
            Param1BadRadioButton = new TRadioButton(this, IDC_PARAM1_BAD);
            Param1NormalRadioButton = new TRadioButton(this, IDC_PARAM1_NORMAL);
            Param1GoodRadioButton = new TRadioButton(this, IDC_PARAM1_GOOD);
            Param1GreatRadioButton = new TRadioButton(this, IDC_PARAM1_GREAT);

            Param2PoorRadioButton = new TRadioButton(this, IDC_PARAM2_POOR);
            Param2BadRadioButton = new TRadioButton(this, IDC_PARAM2_BAD);
            Param2NormalRadioButton = new TRadioButton(this, IDC_PARAM2_NORMAL);
            Param2GoodRadioButton = new TRadioButton(this, IDC_PARAM2_GOOD);
            Param2GreatRadioButton = new TRadioButton(this, IDC_PARAM2_GREAT);

            Param3PoorRadioButton = new TRadioButton(this, IDC_PARAM3_POOR);
            Param3BadRadioButton = new TRadioButton(this, IDC_PARAM3_BAD);
            Param3NormalRadioButton = new TRadioButton(this, IDC_PARAM3_NORMAL);
            Param3GoodRadioButton = new TRadioButton(this, IDC_PARAM3_GOOD);
            Param3GreatRadioButton = new TRadioButton(this, IDC_PARAM3_GREAT);

            Param4PoorRadioButton = new TRadioButton(this, IDC_PARAM4_POOR);
            Param4BadRadioButton = new TRadioButton(this, IDC_PARAM4_BAD);
            Param4NormalRadioButton = new TRadioButton(this, IDC_PARAM4_NORMAL);
            Param4GoodRadioButton = new TRadioButton(this, IDC_PARAM4_GOOD);
            Param4GreatRadioButton = new TRadioButton(this, IDC_PARAM4_GREAT);

            Param5PoorRadioButton = new TRadioButton(this, IDC_PARAM5_POOR);
```

```
    Destroy();

    // INSERT>> Your destructor code here.

}


void TIntermediateCocomoParameterDlg::CmOkButtonHandler ()
{
        // INSERT>> Your code here.
        if(Param1PoorRadioButton->GetCheck()) cData->IntParam1=1;
        else if(Param1BadRadioButton->GetCheck()) cData->IntParam1=2;
        else if(Param1NormalRadioButton->GetCheck()) cData->IntParam1=3;
        else if(Param1GoodRadioButton->GetCheck()) cData->IntParam1=4;
        else if(Param1GreatRadioButton->GetCheck()) cData->IntParam1=5;

        if(Param2PoorRadioButton->GetCheck()) cData->IntParam2=1;
        else if(Param2BadRadioButton->GetCheck()) cData->IntParam2=2;
        else if(Param2NormalRadioButton->GetCheck()) cData->IntParam2=3;
        else if(Param2GoodRadioButton->GetCheck()) cData->IntParam2=4;
        else if(Param2GreatRadioButton->GetCheck()) cData->IntParam2=5;

        if(Param3PoorRadioButton->GetCheck()) cData->IntParam3=1;
        else if(Param3BadRadioButton->GetCheck()) cData->IntParam3=2;
        else if(Param3NormalRadioButton->GetCheck()) cData->IntParam3=3;
        else if(Param3GoodRadioButton->GetCheck()) cData->IntParam3=4;
        else if(Param3GreatRadioButton->GetCheck()) cData->IntParam3=5;

        if(Param4PoorRadioButton->GetCheck()) cData->IntParam4=1;
        else if(Param4BadRadioButton->GetCheck()) cData->IntParam4=2;
        else if(Param4NormalRadioButton->GetCheck()) cData->IntParam4=3;
        else if(Param4GoodRadioButton->GetCheck()) cData->IntParam4=4;
        else if(Param4GreatRadioButton->GetCheck()) cData->IntParam4=5;

        if(Param5PoorRadioButton->GetCheck()) cData->IntParam5=1;
        else if(Param5BadRadioButton->GetCheck()) cData->IntParam5=2;
        else if(Param5NormalRadioButton->GetCheck()) cData->IntParam5=3;
        else if(Param5GoodRadioButton->GetCheck()) cData->IntParam5=4;
        else if(Param5GreatRadioButton->GetCheck()) cData->IntParam5=5;

        if(Param6PoorRadioButton->GetCheck()) cData->IntParam6=1;
        else if(Param6BadRadioButton->GetCheck()) cData->IntParam6=2;
        else if(Param6NormalRadioButton->GetCheck()) cData->IntParam6=3;
        else if(Param6GoodRadioButton->GetCheck()) cData->IntParam6=4;
        else if(Param6GreatRadioButton->GetCheck()) cData->IntParam6=5;

        if(Param7PoorRadioButton->GetCheck()) cData->IntParam7=1;
        else if(Param7BadRadioButton->GetCheck()) cData->IntParam7=2;
        else if(Param7NormalRadioButton->GetCheck()) cData->IntParam7=3;
        else if(Param7GoodRadioButton->GetCheck()) cData->IntParam7=4;
        else if(Param7GreatRadioButton->GetCheck()) cData->IntParam7=5;

        if(Param8PoorRadioButton->GetCheck()) cData->IntParam8=1;
        else if(Param8BadRadioButton->GetCheck()) cData->IntParam8=2;
        else if(Param8NormalRadioButton->GetCheck()) cData->IntParam8=3;
        else if(Param8GoodRadioButton->GetCheck()) cData->IntParam8=4;
        else if(Param8GreatRadioButton->GetCheck()) cData->IntParam8=5;

        if(Param9PoorRadioButton->GetCheck()) cData->IntParam9=1;
        else if(Param9BadRadioButton->GetCheck()) cData->IntParam9=2;
        else if(Param9NormalRadioButton->GetCheck()) cData->IntParam9=3;
        else if(Param9GoodRadioButton->GetCheck()) cData->IntParam9=4;
        else if(Param9GreatRadioButton->GetCheck()) cData->IntParam9=5;

        if(Param10PoorRadioButton->GetCheck()) cData->IntParam10=1;
        else if(Param10BadRadioButton->GetCheck()) cData->IntParam10=2;
        else if(Param10NormalRadioButton->GetCheck()) cData->IntParam10=3;
        else if(Param10GoodRadioButton->GetCheck()) cData->IntParam10=4;
        else if(Param10GreatRadioButton->GetCheck()) cData->IntParam10=5;

        if(Param11PoorRadioButton->GetCheck()) cData->IntParam11=1;
```

```
            else if(Param11BadRadioButton->GetCheck()) cData->IntParam11=2;
            else if(Param11NormalRadioButton->GetCheck()) cData->IntParam11=3;
            else if(Param11GoodRadioButton->GetCheck()) cData->IntParam11=4;
            else if(Param11GreatRadioButton->GetCheck()) cData->IntParam11=5;

            if(Param12PoorRadioButton->GetCheck()) cData->IntParam12=1;
            else if(Param12BadRadioButton->GetCheck()) cData->IntParam12=2;
            else if(Param12NormalRadioButton->GetCheck()) cData->IntParam12=3;
            else if(Param12GoodRadioButton->GetCheck()) cData->IntParam12=4;
            else if(Param12GreatRadioButton->GetCheck()) cData->IntParam12=5;

            if(Param13PoorRadioButton->GetCheck()) cData->IntParam13=1;
            else if(Param13BadRadioButton->GetCheck()) cData->IntParam13=2;
            else if(Param13NormalRadioButton->GetCheck()) cData->IntParam13=3;
            else if(Param13GoodRadioButton->GetCheck()) cData->IntParam13=4;
            else if(Param13GreatRadioButton->GetCheck()) cData->IntParam13=5;

            if(Param14PoorRadioButton->GetCheck()) cData->IntParam14=1;
            else if(Param14BadRadioButton->GetCheck()) cData->IntParam14=2;
            else if(Param14NormalRadioButton->GetCheck()) cData->IntParam14=3;
            else if(Param14GoodRadioButton->GetCheck()) cData->IntParam14=4;
            else if(Param14GreatRadioButton->GetCheck()) cData->IntParam14=5;

            if(Param15PoorRadioButton->GetCheck()) cData->IntParam15=1;
            else if(Param15BadRadioButton->GetCheck()) cData->IntParam15=2;
            else if(Param15NormalRadioButton->GetCheck()) cData->IntParam15=3;
            else if(Param15GoodRadioButton->GetCheck()) cData->IntParam15=4;
            else if(Param15GreatRadioButton->GetCheck()) cData->IntParam15=5;

  CmOk();
}




#if !defined(__taddnwpd_h)          // Sentry, use file only if it's not already included.
#define __taddnwpd_h

/*  Project fellows
            Texas A&M University
   Copyright © 1994. All Rights Reserved.

   SUBSYSTEM:    fellows.apx Application
   FILE:         taddnwpd.h
   AUTHOR:       Jason Thompson


   OVERVIEW
   ========
   Class definition for TAddNewProjectDlg (TDialog).
*/

#include <owl\owlpch.h>
#pragma hdrstop


#include "fllwsapp.rh"          // Definition of all resources.


//{{TDialog = TAddNewProjectDlg}}
class TAddNewProjectDlg : public TDialog {
public:
   TAddNewProjectDlg (ProjectData *data, TWindow* parent, TResId resId = IDD_NEW_PROJECT, TModule* module = 0);
   virtual ~TAddNewProjectDlg ();

private:
  TEdit          *ProjectNameEditField;
  ProjectData *cData;

//{{TAddNewProjectDlgRSP_TBL_BEGIN}}
```

```
protected:
   void CmOkButtonHandler ();
//{{TAddNewProjectDlgRSP_TBL_END}}
DECLARE_RESPONSE_TABLE(TAddNewProjectDlg);
};   //{{TAddNewProjectDlg}}


#endif                              // __taddnwpd_h sentry.





/*  Project fellows
   Texas A&M University
   Copyright © 1994. All Rights Reserved.

   SUBSYSTEM:    fellows.apx Application
   FILE:         taddnwpd.cpp
   AUTHOR:       Jason Thompson


   OVERVIEW
   ========
   Source file for implementation of TAddNewProjectDlg (TDialog).
*/

#include <owl\owlpch.h>
#pragma hdrstop

#include "taddnwpd.h"
#include "dbproj.h"


//
// Build a response table for all messages/commands handled
// by the application.
//
DEFINE_RESPONSE_TABLE1(TAddNewProjectDlg, TDialog)
//{{TAddNewProjectDlgRSP_TBL_BEGIN}}
   EV_BN_CLICKED(IDOK, CmOkButtonHandler),
//{{TAddNewProjectDlgRSP_TBL_END}}
END_RESPONSE_TABLE;


//{{TAddNewProjectDlg Implementation}}


TAddNewProjectDlg::TAddNewProjectDlg (ProjectData *data, TWindow* parent, TResId resId, TModule* module):
   TDialog(parent, resId, module)
{
		// INSERT>> Your constructor code here.

		ProjectNameEditField = new TEdit(this, IDC_NEW_PROJECT_ENAME);
		cData = data;
}


TAddNewProjectDlg::~TAddNewProjectDlg ()
{
		Destroy();

   // INSERT>> Your destructor code here.

}


void TAddNewProjectDlg::CmOkButtonHandler ()
{
		// INSERT>> Your code here.
		// Check to see if the field has been entered.
		char sBuf[21];

		ProjectNameEditField->GetLine(sBuf, 21, 1);
		if(sBuf[0]==NULL)
		{
				MessageBox("You must enter a project name");
		}
		else
		{
				DBProj *pdb;
				int returnval,i;

				for(i=strlen(sBuf);i<20;i++)
						sBuf[i]=' ';
				sBuf[20]=NULL;

				pdb=new DBProj;
				returnval=pdb->seek(sBuf);
				if (returnval == 0) //entry already exists
				{
						MessageBox("A project already exits by that name");

						delete pdb;
				}
				else
				{
						pdb->bottom();
						pdb->append();
						pdb->ProjectName(sBuf);
						pdb->save();

						delete pdb;
	strcpy(cData->ProjectName, sBuf);
						CmOk();
				}
		}
}



#if !defined(__topnprjd_h)		// Sentry, use file only if it's not already included.
#define __topnprjd_h

/*  Project fellows
   Texas A&M University
   Copyright © 1994. All Rights Reserved.

   SUBSYSTEM:    fellows.apx Application
   FILE:         topnprjd.h
   AUTHOR:       Jason Thompson


		OVERVIEW
   ========
   Class definition for TOpenProjectDlg (TDialog).
*/

#include <owl\owlpch.h>
#pragma hdrstop


#include "fllwsapp.rh"		// Definition of all resources.
#include "dbproj.h"


//{{TDialog = TOpenProjectDlg}}
class TOpenProjectDlg : public TDialog {
public:
   TOpenProjectDlg (ProjectData *data, TWindow* parent, TResId resId = IDD_OPEN_PROJECT, TModule* module = 0);
```

```
    virtual ~TOpenProjectDlg ();

private:
   TComboBox            *ProjectComboBox;
   ProjectData          *cData;

//{{TOpenProjectDlgVIRTUAL_BEGIN}}
public:
    virtual void SetupWindow ();
//{{TOpenProjectDlgVIRTUAL_END}}

//{{TOpenProjectDlgRSP_TBL_BEGIN}}
protected:
    void CmOkButtonHandler ();
//{{TOpenProjectDlgRSP_TBL_END}}
DECLARE_RESPONSE_TABLE(TOpenProjectDlg);
};    //{{TOpenProjectDlg}}


#endif                                  // __topnprjd_h sentry.




/*  Project fellows
                Texas A&M University
                Copyright © 1994. All Rights Reserved.

                SUBSYSTEM:    fellows.apx Application
                FILE:         topnprjd.cpp
                AUTHOR:       Jason Thompson


                OVERVIEW
                ========
                Source file for implementation of TOpenProjectDlg (TDialog).
*/

#include <owl\owlpch.h>
#pragma hdrstop

#include "topnprjd.h"
#include "dbproj.h"


//
// Build a response table for all messages/commands handled
// by the application.
//
DEFINE_RESPONSE_TABLE1(TOpenProjectDlg, TDialog)
//{{TOpenProjectDlgRSP_TBL_BEGIN}}
    EV_BN_CLICKED(IDOK, CmOkButtonHandler),
//{{TOpenProjectDlgRSP_TBL_END}}
END_RESPONSE_TABLE;


//{{TOpenProjectDlg Implementation}}


TOpenProjectDlg::TOpenProjectDlg (ProjectData *data, TWindow* parent, TResId resId, TModule* module):
                TDialog(parent, resId, module)
{
                // INSERT>> Your constructor code here.

                ProjectComboBox = new TComboBox(this, IDC_OPEN_PROJECT_CNAME);
                cData=data;
}


TOpenProjectDlg::~TOpenProjectDlg ()
```

```
{
            Destroy();

            // INSERT>> Your destructor code here.

            delete ProjectComboBox;
}


void TOpenProjectDlg::SetupWindow ()
{
            TDialog::SetupWindow();

            // INSERT>> Your code here.
            DBProj      *pdb;

            pdb = new DBProj();
            ProjectComboBox->ClearList();
            for(pdb->top();!pdb->eof();pdb->skip())
                        ProjectComboBox->AddString(pdb->ProjectName());
}


void TOpenProjectDlg::CmOkButtonHandler ()
{
            // INSERT>> Your code here.
            char                        sBuf[21];

            ProjectComboBox->GetText(sBuf, 20);
            if(sBuf[0]==NULL)
            {
                        MessageBox("You must choose a project from the list box");
            }
            else
            {
                        strcpy(cData->ProjectName, sBuf);
   CmOk();
            }
}




#if !defined(__tbscccpd_h)           // Sentry, use file only if it's not already included.
#define __tbscccpd_h

/*  Project fellows
   Texas A&M University
   Copyright © 1994. All Rights Reserved.

   SUBSYSTEM:    fellows.apx Application
   FILE:         tbscccpd.h
   AUTHOR:       Jason Thompson


   OVERVIEW
   ========
   Class definition for TBasicCocomoParameterDlg (TDialog).
*/

#include <owl\owlpch.h>
#pragma hdrstop


#include "fllwsapp.rh"          // Definition of all resources.


//{{TDialog = TBasicCocomoParameterDlg}}
class TBasicCocomoParameterDlg : public TDialog {
private:
  ProjectData          *cData;
  TEdit                                        *SlocEditField;
            TRadioButton         *EmbeddedRadioButton;
            TRadioButton         *OrganicRadioButton;
            TRadioButton         *SemiDetachedRadioButton;
public:
   TBasicCocomoParameterDlg (ProjectData *data, TWindow* parent, TResId resId = IDD_BASIC_COCOMO, TModule* module =
0);
   virtual ~TBasicCocomoParameterDlg ();

//{{TBasicCocomoParameterDlgRSP_TBL_BEGIN}}
protected:
   void CmOkButtonHandler ();
//{{TBasicCocomoParameterDlgRSP_TBL_END}}
DECLARE_RESPONSE_TABLE(TBasicCocomoParameterDlg);
};   //{{TBasicCocomoParameterDlg}}


#endif                          // __tbscccpd_h sentry.




/*  Project fellows
   Texas A&M University
   Copyright © 1994. All Rights Reserved.

   SUBSYSTEM:    fellows.apx Application
   FILE:         tbscccpd.cpp
   AUTHOR:       Jason Thompson


   OVERVIEW
   ========
   Source file for implementation of TBasicCocomoParameterDlg (TDialog).
*/

#include <owl\owlpch.h>
#pragma hdrstop

#include "tbscccpd.h"


//
// Build a response table for all messages/commands handled
// by the application.
//
DEFINE_RESPONSE_TABLE1(TBasicCocomoParameterDlg, TDialog)
//{{TBasicCocomoParameterDlgRSP_TBL_BEGIN}}
            EV_BN_CLICKED(IDOK, CmOkButtonHandler),
//{{TBasicCocomoParameterDlgRSP_TBL_END}}
END_RESPONSE_TABLE;


//{{TBasicCocomoParameterDlg Implementation}}


TBasicCocomoParameterDlg::TBasicCocomoParameterDlg (ProjectData *data, TWindow* parent, TResId resId, TModule* module):
            TDialog(parent, resId, module)
{
  // INSERT>> Your constructor code here.
            cData=data;
            SlocEditField = new TEdit(this, IDC_BASIC_COCOMO_ESLOC);
            EmbeddedRadioButton = new TRadioButton(this, IDC_BASIC_COCOMO_RTYPE_EMBEDDED);
            OrganicRadioButton = new TRadioButton(this, IDC_BASIC_COCOMO_RTYPE_ORGANIC);
            SemiDetachedRadioButton = new TRadioButton(this, IDC_BASIC_COCOMO_RTYPE_SEMIDETACHED);
}


TBasicCocomoParameterDlg::~TBasicCocomoParameterDlg ()
```

```
{
            Destroy();

            // INSERT>> Your destructor code here.

}


void TBasicCocomoParameterDlg::CmOkButtonHandler ()
{
            // INSERT>> Your code here.
            char sBuf[100];

            SlocEditField->GetLine(sBuf, 7, 1);
            if (sBuf[0]==NULL)
            {
                        MessageBox("You must enter the number of source lines of code");
            }
            else if (!EmbeddedRadioButton->GetCheck() &&
                                                !OrganicRadioButton->GetCheck() &&
                                                !SemiDetachedRadioButton->GetCheck())
            {
                        MessageBox("You must choose a project type");
            }
            else
            {
                        if(EmbeddedRadioButton->GetCheck())
                                    cData->ProjectType=EMBEDDED;
                        else if(OrganicRadioButton->GetCheck())
                                    cData->ProjectType=ORGANIC;
                        else if(SemiDetachedRadioButton->GetCheck())
                                    cData->ProjectType=SEMIDETACHED;

                        cData->EstimatedSize=atol(sBuf);

   CmOk();
            }
}
```

Bibliography

Abdel-Hamid, Tarek, and Stuart E. Madnick. "Lessons Learned from Modeling the Dynamics of Software Development." *Communications of the ACM.* December, 1989. p.1426-1438

Belova, L. A. and V. V. Lipaev. "Cost Estimation of Complex Software Development For Management." *Automation and Remote Control.* July 1988. p.949-955

Boehm, Barry W. "A Spiral Model of Software Development and Enhancement." *Computer.* May 1988. p.61-72.

Henry, Sallie, and Calvin Selig. "Predicting Source-Code Complexity at the Design Stage." *IEEE Software.* March, 1990. p.26-43.

Khoshgoftaar, Taghi M., John C. Munson, Bibhuti B. Bhattacharya, and Gary D. Richardson. "Predictive Modeling Techniques of Software Quality from Software Measures." *IEEE Transactions on Software Engineering.* November, 1992. p. 979-986.

Lehder, Wilfred E., D. Paul Smith, and Weider D. Yu. "Software Estimation Technology." *AT&T Technical Journal.* July/August, 1988. p.10-18.

Lutz, Martin. "Better models, better estimates." *IEEE Software.* July, 1988. p.110-111.

Putnam, Lawrence H. "Trends in Measurement, Estimation, and Control." *IEEE Software.* March, 1991. p.105-107.

Sommerville, Ian. *Software Engineering.* England: Addison-Wesley Publishing Company, 1992.