AN INTEGRATED SET OF TOOLS TO ASSIST IN THE DEVELOPMENT AND MAINTENANCE OF PROJECT LIFE CYCLES

By

SRIDHAR CHANDRASHEKAR

Bachelor of Engineering

P.E.S. College of Engineering

University of Mysore

Mysore, India

1986

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
July, 1991

AN INTEGRATED SET OF TOOLS TO ASSIST

IN THE DEVELOPMENT AND MAINTENANCE

OF PROJECT LIFE CYCLES

Thesis Approved:

Blayne E. Mayfield

Thesis Adviser

J. P. Chandler

William Daniel Miller

Norman N. Durham

Dean of the Graduate College

ACKNOWLEDGEMENTS

I profoundly thank Dr. B. Mayfield for his unstinted help and guidance His constructive criticism helped me in gaining confidence during my graduate program. My sincere thanks to Drs D Miller and J Chandler for serving on my graduate committee Their suggestions and support were very helpful throughout the study

I would not be doing justice to myself if I do not thank Dr M Samadzadeh, without whose help my knowledge in Software Engineering would have remained limited He not only helped me in my thesis, but also in my course work

I would like to express my gratitude to Learmonth & Burchett Management Systems, Inc., for sponsoring this project in part In particular, I offer my thanks to Mr John Bantleman, Mr Rick Plezcko, and Mr David Hsieh for their help I would also like to thank Dr Hedrick for helping me to go to Houston (LBMS Inc.) to complete this project

My special thanks are extended to Mr Gopal Kulkarni for helping me in everything, from designing to debugging Last but not least, I express my gratefulness to Mr Manohar Rao, Mr Ravi Kumar, Mr. Sarvesh Jagannivas, Mr. Suresh Subramanian, and Mr Sangram Bhosale for helping me get my project report ready

TABLE OF CONTENTS

Chapter Page

Chapter Page

LIST OF FIGURES

NOMENCLATURE

CASE. Computer-Aided Software Engineering.

CAPE. Computer-Aided Project Engineering.

Decision 123. An AI rule-based tool developed by The Proctor and Gamble AI team Decision 123 helps the user build the knowledgebase through tables, decision networks and/or rules The output code can run in either M1 (an AI package) or KnowledgePro (a Windows based product combining hypermedia, expert systems and object-oriented programming technologies which also provides a software development environment)

Direct Manipulation. The use of a pointing device to perform actions on objects Examples are mouse-click and mouse-drag [IBM SAA/CUA Guide, 1990]

Double-Click To press and release a mouse button within a user-defined time limit without moving the mouse pointer off the choice [IBM SAA/CUA Guide, 1990].

Function Point A measure of complexity of what is to be delivered in a project

Hot Link A direct link between two applications

Icon A pictorial representation of an object or a selection choice. Icons can represent objects that users want to work on or actions that users want to perform. A unique icon also represents the application when it is minimized

KnowledgePro. A development tool for Windows applications

Menu. A component of a dialog design consisting of a screen which can display options and receive control input.

Multiple Document Interface. An interface style that allows users to view many objects at the same time or the same object many times within one primary window [IBM SAA/CUA Guide, 1990]

PERT Program Evaluation and Review Technique, a method to indicate top-management presentations on major deliverables, that is, the milestones or events [Zells, P , 1990]

Software Development Kit (SDK) A development kit supplied by Microsoft for software development under Windows This works with the Microsoft C optimizing compiler

Software Life Cycle. Various stages of software development.

The possible stages are

- Project Initiation
- Requirements and Analysis study
- Project Specification and Logical Design
- Physical Design and High level design
- Coding, Debugging, and Testing
- Maintenance

OPRR Object, Property, Role and Relationship A term used in conjunction with a repository schema developed at LBMS Inc, The repository design incorporates an object-oriented design and use of the concept of object-property-role and relationship or OPRR, keeping in mind the future trends in software automation

Windows (3.0) A user interface developed by Microsoft Corporation for IBM PCs and compatibles

Window. An area of the screen with visible boundaries through which information is displayed. A window can be smaller than or equal in size to the screen. Windows can overlap on the screen and give the appearance of one window being on top of another.

Work Breakdown Structure The process of dividing a whole project into several small pieces which can be easily manageable

CHAPTER I

INTRODUCTION

Software Engineering encompasses a variety of technical methods, a set of management procedures, and a suite of automated tools (often called CASE - Computer-Aided Software Engineering) that enhance our ability to build effective computer-based systems [Pressman, 1988] A project goes through several stages, from project initiation to implementation and maintenance; this is often referred to as the software development life cycle Specific project development life cycles constitute the foundation of software systems Proper management of these life cycles lead to better project planning and management. Fig. 1.1 illustrates a standard life cycle for software engineering. The different stages in the life cycle can generally be listed as follows·

- Project Initiation (Planning)
- Requirements Analysis
- Project Specification and Logical Design
- Physical Design (High Level Design)
- Coding and Debugging
- Testing

. Implementation (either in part or full as required)
. Maintenance

Every stage includes a loop back to the previous stage and/or to some of the previous stages. This concept of loop back allows for system checks against delays versus scheduled task time lengths so that stage time estimates can be revised to depict task time changes.

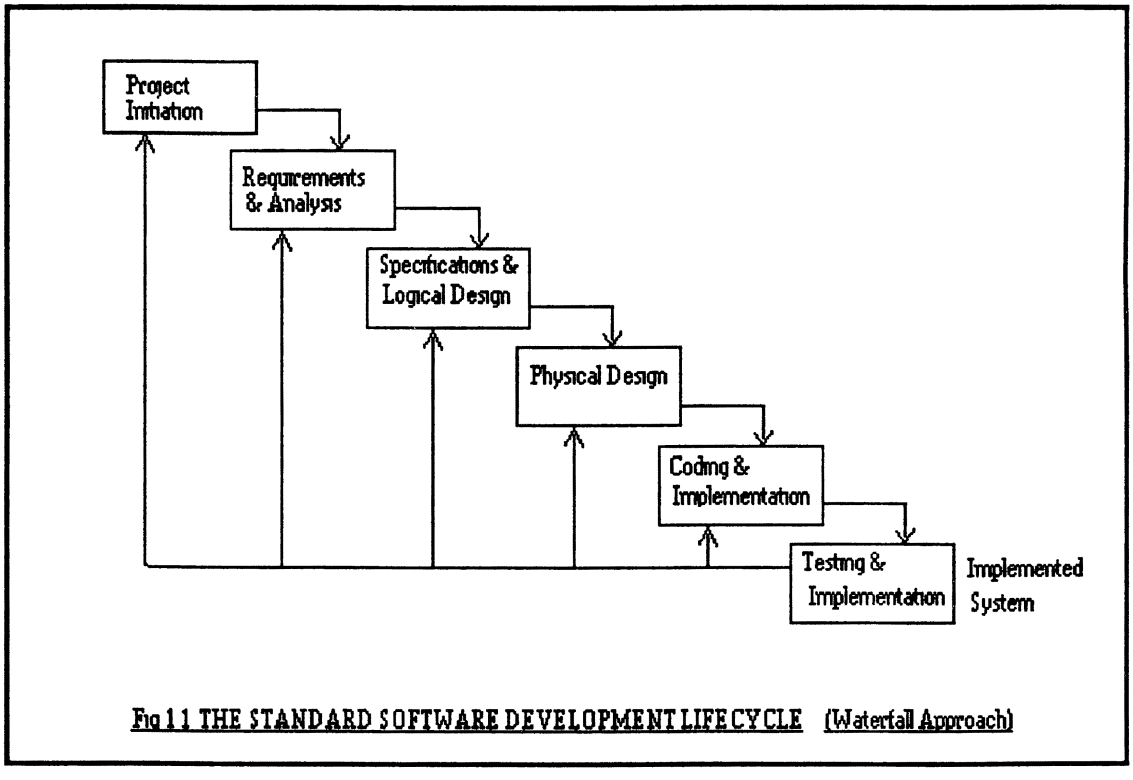


Fig 1.1 THE STANDARD SOFTWARE DEVELOPMENT LIFE CYCLE (Waterfall Approach)

Figure 1.1 The Standard Software Development Life Cycle. (The Waterfall approach)

Within the context of software engineering, a method is a procedure or technique for performing some significant portion of the software life cycle [Orr et al , 1989] A methodology, in software engineering terms, is a collection of methods based on a common philosophy that fit together in a framework called the systems development life cycle A software life cycle may be implemented through any of several methodologies Some of the well-known methodologies are given below

- The Warnier/Orr approach [Orr et al , 1989]
- The Yourdon approach [Orr et al , 1989]
- The Gane/Sarson approach [Orr et al , 1989]
- The Entity-Relationship approach
- HIPO (Hierarchical Input Process Output) approach
- The Michael Jackson approach [Cameron, 1986]

CASE is a combination of software tools and structured development methodologies [McClure, 1989] Whereas tools attempt to automate the software process, methodologies define the process to be automated

Project Management

A project may be defined as a group of interrelated tasks taken one at a time to achieve a specific goal Project management is the art of managing such tasks, usually performed by a project leader A group of people will be assigned to a project with a leader to manage the

project Every project has to be planned to be properly carrıed out Thus a systematıc approach ıs called for Whıle plannıng ıs essentıal, ıt does not, by ıtself, produce technıcal delıverables A tool may be helpful. Several project management tools exist ın the marketplace, but most of these tools only perform the schedulıng and estımation actıvıtıes and help ın drawıng charts

Project Management Tools (PMT)

The ıdea of a paperless offıce has gıven rıse to several ımportant applıcatıons to be maıntaıned on the computer Among many areas of software development are the software packages dealıng wıth the technıcal aspects of software engıneerıng Tools whıch assıst the project manager ın maıntaınıng software projects are called Project Management Tools (PMTs) PMTs range from sımple schedulıng tools to complex estımatıon and analysıs tools These tools provıde sımple user ınterfaces, provıde graphıc dısplays and help ın maıntaınıng the dynamıcally changıng project phases Examples of such exıstıng tools are Harvard Project Manager, SuperProject Expert, Tıme Lıne, and Mıcrosoft Project It should be noted that any of these software tools, by themselves, do not aıd the project manager ın all the actıvıtıes of project management Thus, there are

specialized tools to maintain separate parts of the project, viz scheduling tools and estimation tools, to name a few

Scheduling Tools

Projects have to be scheduled as to when they would start, how many days will be required to complete each phase of the project, etc Likewise, appropriate resources have to be assigned Scheduling tools help in maintaining these numbers It should be understood that the numbers assigned at the inception of a project are tentative numbers and are subject to change due to many reasons For example, an unexpected delay in acquiring resources, in the form of hardware/software required for the project, or, the unavailability of some key technical person(s), may lead to delays which may be carried over to the successive phase(s) Actual time and other resources taken for past projects can be used to compute the schedule of the present project

Estimating Tools

A tool to estimate (calculate) the effort required to do a particular task or a number of tasks in the project is termed as an estimation tool There are two general ways to approach estimation function point-based estimation and object-based estimation

Function points are used to measure system size as a component of productivity measurement [Zells, 1990] Users count the number of inputs and outputs, number of internal files, number of interfaces, etc. Starting the process at a point, when a comfortable amount of analysis and design has been completed, the estimators classify and count raw function points This, along with other important data, aid in estimating projects

Object-based estimation pertains to measuring the effort involved in completing a particular task in a project life cycle The effort depends upon several factors, for example, the person doing a particular task in a project is one of such factors Object-based estimation is the most common method used in software tools

Advantages and Limitations

The scheduling tools help the project manager in keeping track of time expended and time required for different phases of a project Similarly, the estimation tools aid in planning and maintaining the effort required to complete the project Other management tools portray the state of the project using charts, and pictorially represent the same

The tools mentioned above support the project manager in several different aspects of project management Figure

1 2 depicts the performance of different tools with respect to different phases of a project It may also be observed that a tool to support a software project at all phases is desired

Computer-Aided Project Engineering (CAPE)

There are two aspects to a software project techniques and planning [Hsieh, 90] While the techniques helps in implementing the project, planning aid, in designing the project to be properly conceived and executed As mentioned

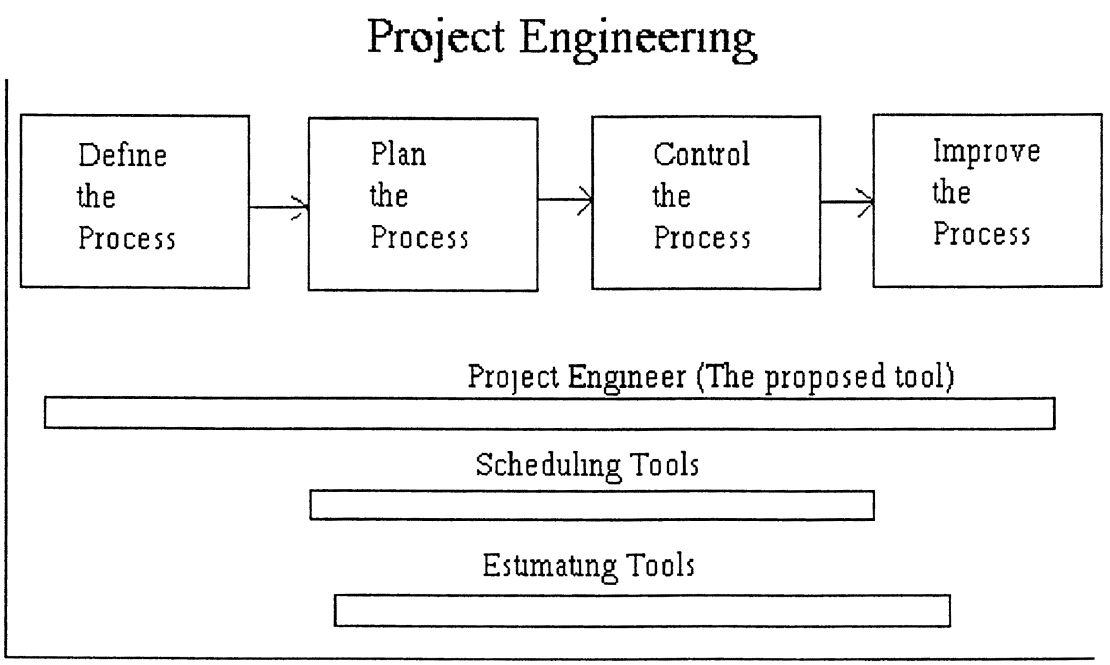


Fig 1 2 The limitations of scheduling tools

in the previous section, there are several tools to automate the techniques part (in particular the CASE tools that support the Programmer/Analyst/Designer). However, the planning and management aspects along with the automation of software methodologies usually are not supported by existing tools satisfactorily. While CASE tools automate the production of technical deliverables, CAPE tools automate the production of planning deliverables. Delivery of planning deliverables usually is accomplished by project managers manually, which makes manipulation and maintenance of the same cumbersome. A tool to perform these management aspects could increase the productivity of the project manager, while providing a proper foundation for the project from the start. Currently, support for the Project Manager is limited to such single-purpose tools

Statement of the Problem

Software projects range from simple to complex systems. Each project has a unique Work Breakdown Structure (WBS) constituting the project life cycle According to Zells,

> When managers try to plan a project without being able to reference an internal corporate project history file, a methodology checklist, or even a

> book, they may simply draw a blank and be stymied about where to start [Zells, 1990]

Lack of pre-existing templates of project plans may thus hinder proper planning. In other words, an existing template helps in this first and most important step (planning).

Software methods are used to plan projects. In the past, one life cycle was adapted to fit all projects, this approach was subsequently modified later to adopt one of several life cycles to fit a particular project. Multiple life cycles soon become a problem as they proliferate and are a problem to maintain A solution to this problem can be found in modular methods, which can be maintained easily while being used with proper changes for specific projects

Objectives of the Study

The objective of this study is to design, implement, and test a Computer-Aided Project Engineering (CAPE) tool for automating and maintaining the software development life cycle for (software) projects from a Project Management perspective. This tool allows the user to load and manipulate project life cycles and templates, view the project from various perspectives, and provides on-line

method help (hypermedia based), as well as an export link to scheduler packages Emphasis is given to the design of the user-interface component of this tool following the IBM SAA/CUA standard [IBM SAA/CUA standards, 1990] The proposed tool is called Project Engineer Project Engineer uses Multiple Document Interface (MDI) child windows to display information to the user from different perspectives A pictorial representation of the Work Breakdown Structure is also be provided for the user to view the project hierarchy (the other representation being the standard Activity Outline window or the List window) At this stage, the user is allowed only to view the pictorial representation

In short, Project Engineer is designed to support a wide range of project management activities in a modular and integrated fashion

CHAPTER II

CONCEPTS USED IN LIFE CYCLE BUILDER

Introduction

The purpose of this project is to provide a better way to enhance the delivery and accessibility of (software) Methods, to automate the work of the Project Manager, and to provide support for continuous process improvement while maintaining project document consistency This tool also implements the best available industry/organization practices with respect to user interface and modular software methods

Work Breakdown Structures

A project may be split into stages, steps and tasks, tasks being the smallest deliverable unit A group of associated tasks comprise a step and a group of associated steps comprise a stage This partitioning into several small modules is referred to as a Work Breakdown Structure

(WBS) Given below is an example of the WBS for the Project Initiation stage (Only one Step is expanded).

PI - Project Initiation (Stage)
PI.PI1 - Determine Scope (Step)
PI PI1.10 - Review Related Studies
PI.PI1.20 - Establish Scope
PI.PI1 30 - Establish Major Objectives
PI.PI1.40 - Establish Constraints
PI PI1.50 - Identify Outline Solution
PI PI2 - Establish Project Plan and Budget
etc

The Project Database

CASE tools handle data that are related in complex ways They need data integrity and non-redundancy in data representation. A data repository is best suited for such an application. A repository is a mechanism for storing and organizing all information concerning a software system, including planning, analysis, design, implementation and project management information [McClure, 1989] It is also referred to as a design dictionary, database, object-oriented database, knowledgebase, or encyclopedia. The purpose of a repository is to store system information at a central place, keep the data uniform, and be accessible to all users The repository must be robust enough to cater

to the needs of large software projects and must be scalable [McClure, 1989]

One of the most essential steps in software development is to identify and define the different types of data involved This is realized by data modeling Two basic concepts in data modeling are 'entity' and 'relationship' Matthews and McGee define an entity as any identifiable thing or event that can be characterized in terms of a set of attributes and their associated values and a relationship as an association of two or more entities which may have attributes of its own [Mattews and McGee, 1990]

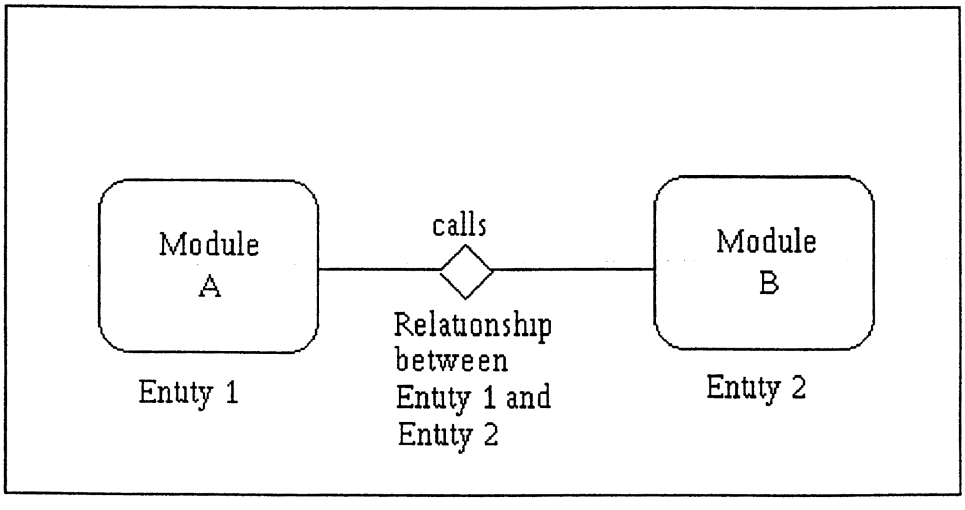


Fig 2 1 Entity-Relationship example

For example, as in Figure 2 1, let A and B be two modules and let A call B, then A and B are two entities and 'calling' is a relationship between A and B There are

several data models employing the concepts of entities and relationships. Some of these models are delineated below [Welke, 1989]

The Binary Model

This is a simple model characterized by the involvement of exactly two entities and a single relationship between them. There are two types of binary models: binary-1 and binary-2 The binary-1 form allows multiple instances of only one object type usually, denoted as 1.M (one to many) Binary-2 form allows multiple instances of both entity types (M·M, many-to-many) Because of their simplicity, binary models have very limited applicability to complex applications such as CASE tools.

The Entity-Attribute-Relationship-Attribute (EARA)Model

To overcome the limitations of the binary model and to express multi-part relationships, the EARA model is used. In this model, properties or attributes are associated with each entity participating in a relationship To illustrate this, let us consider the previous example of module A calling module B as depicted in Figure 2 2. Assume that module A calls module B iteratively

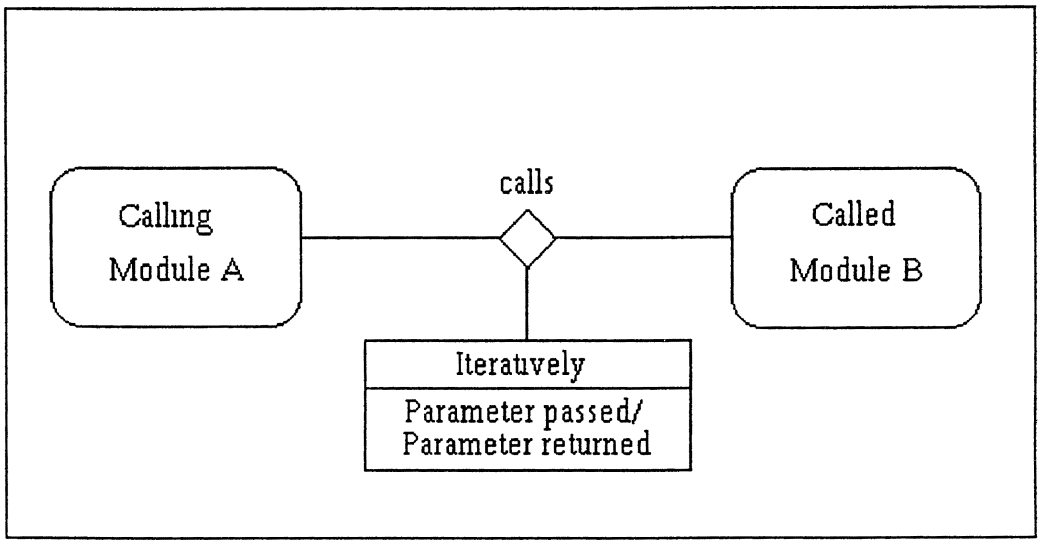


Fig 2 2 The EARA model example

This can be expressed by associating the attribute 'iteratively' to relationship 'calls' Now consider an even more complex situation in which the parameter is either 'passed' or 'returned' This can be expressed by associating the attribute 'parameter type' (passed or returned) to the 'calls' relationship Even though the EARA model allows sophisticated multi-part relationship's, it does not express clearly the complexity involved in the relationship It is not possible to clearly express the roles played by the modules and the parameters, as is evident in the above example Thus, there is a need for a higher meta model

The Object, Property, Role, and Relationship (OPRR) Model

To overcome the limitations of the EARA model, the OPRR model is used. In the OPRR model, roles are associated with entities that modify the way an entity takes part in a relationship This adds another degree of freedom and hence enhances the expressiveness of the model. A meta model is a powerful modeling technique to clearly and unambiguously capture the meaning of design notations. In other words, a meta model is a "database schema" for a data model. For example in the previous section, the role of each object (module) with respect to the 'calls' relationship can be made explicit The role attribute indicates the direction, i.e, the type of parameter (passed or returned). Thus there is no data redundancy and the semantics are expressed clearly ensuring data integrity.

In the OPRR database, data modeling can be done at a meta level. Hence many-to-many binary relationships with roles attached to each participating entity and each entity having multiple properties can be expressed very easily. Two or more objects can take on different roles and participate in one or more relationships with multiple instances. This gives additional degrees of freedom in data representation compared to other data modeling techniques explained in the previous sections. The representation is complete and can be modified easily without affecting other relationships and entities. Welke makes a comparative study

of various modeling techniques and proves the superiority of the OPRR technique [Welke, 1988]

The OPRR data engine actually uses two kinds of databases in its operation: the Meta database and the Instance database A meta database contains the meta schema definition that defines what kind of objects and relationships are stored in the database i e , Activities, Description of tasks, Status and Schedule information, etc An Instance database contains the actual data about a project This closely follows the object-oriented paradigm (the meta database is like an object class and the instance database is an instance of that object class) For example, a WBS object is defined as an entity in the meta database and all the tasks in a project are instances of this object Simultaneously the tasks inherit all the properties associated with the WBS object

The OPRR database can be accessed using an interface called the Logical Device Interface (LDI) The LDI allows only one instance database to be the "current" instance at any given time

Thus the Meta database approach imparts flexibility by supporting multiple methodologies and lets the user customize his/her own methodology [Welke, 1989] Also it fully supports future evolution of the project and lets the user add more analysis and reporting functions The OPRR database has an object-oriented design and is easy to maintain

The User Interface

Recent studies have shown that the user interface forms a significant part of any application [Myers, 1988] It is also arguably the most difficult part to develop, since it is necessary for the designer to understand the problem technically, while considering the human factors involved The user interface of any software package is that part which accepts input, interacts with the user, and should presents him/her with a friendly environment It should be designed in such a way that it makes the interaction between the user and the system easy and intelligent, with good response time and efficient use of available resources The interaction technique consists of the way of using physical devices such as a mouse, the keyboard, and/or a light pen to input a certain value There are different styles (interaction styles) in which this interaction can be made, viz , menu selection, form fill-in, command language, natural language, and direct manipulation [Shneiderman, 1987]. Of these, direct manipulation is the most popular The reason is that direct manipulation involves selecting an object of interest and performing the required action Elaborate graphics, mode-free interface, multiple ways for the same command, ease of use and semantic feedback, all make direct manipulation the most used interaction style in recent software packages

Different Approaches

Initially, two approaches were tried to display the project tasks in a window. One was to use list boxes stacked side by side, giving an appearance of tabular data display fashion. The other method was to use edit boxes for the purpose. The latter gives the flexibility to edit fields as they are, although it is problematic to select multiple items across rows and columns The former method is too awkward to code and maintain. Both methods are not standard for such purposes, but it was a good exercise. The Life Cycle Builder (Chapter 4) gives more details about the third (and final) approach selected.

Final User Interface Design

The design of the user-interface for this project is generally inspired by the IBM SAA/CUA standards The design adopts all the relevant features listed in the SAA Advanced Interface Design Guide [IBM SAA/CUA Guide, 1990]. Direct manipulation interface design along with form fill-in approach will be used for certain parts . The platform selected to design and implement this package is Microsoft Windows version 3 0. It features a mouse-integrated, graphical user interface in addition to using commands and

conventions considered standard for 'Windows-style' programs

Project Templates

Different methodologies use different strategies in breaking up the project into several parts A project can be divided into several stages depending upon its complexity Thus a small three-month software project may have just three stages and another two-year project may have eight stages

The need for a template application occurs when certain groups of jobs must be used over and over again within the same project or across several projects [Zells, L , 1990] Templates bring with them valuable information that is needed to give the user a head start Since templates are proven software life cycles themselves, they bring experience and expertise with them This can save the user considerable time and effort

Project Views and Task Hiding

Project Views is a feature that will allow the user to restrict the view of the project database to those parts concerning a single project team member This will be the basis for a report that can be handed to a project team

member as his/her personal project plan, including action items, estimates, resources, etc. The ViewPoints facility will give precisely the information required by hiding redundant data For example, to view the details regarding the estimation of a particular project, the estimation ViewPoints displays Estimated effort, Revised estimation, Effort expended so far, and the Remaining Effort. This helps the user to understand relatively easily and analyze estimation requirements

CHAPTER III

DESIGN AND IMPLEMENTATION

Introduction

Project Engineer is coded in C, using the Software Development Kit supplied by Microsoft Corporation to develop applications which run in Windows version 3 0 The application requires Windows 3 0 as a base and runs on PC platforms

Implementation of the Concepts

In the following subsections a brief description of the implementations of concepts discussed in the previous chapter is provided

Implementation of the Database

Project Engineer uses an OPRR meta database model discussed in the previous chapter Two types of files exist

in Project Engineer. Life Cycle Templates and Project Databases. There is no internal difference between the two types (except in the DOS file extension) and both types are based on the same OPRR Meta database (both are instance databases). Different Life cycles can be retrieved just by changing the instance database. In the Life Cycle Builder module's meta database, the WBS object is defined as an entity with properties associated with it. Simple binary relationships between these objects have been defined and used.

Project Templates

A life cycle template will always be used as the basis for a Project Database. It will be modified only when changes that affect all future uses of the template are made The user also will be allowed to make changes to a Template/Project and save it as a template This facility is useful if the users handle projects which follow a pattern but differ slightly. In order to create a new project, a template will be loaded, modified, and then saved as a project.

Hence, several templates will be provided in Project Engineer for the user to cater to the different profiles of software life cycles These templates present a starting

point for the user to tune the appropriate template to build the required life cycle

The User Interface Implementation

Project Engineer uses Multiple Document Interface (MDI) windows to display information from different perspectives. MDI Child windows are windows which are controlled by and appear within a parent or 'main' window [Petzold, 1990] These child windows function exactly like main windows, including minimize/maximize buttons, resizability, etc , but they are limited to the boundaries of the parent window. Each MDI Child window performs a function and can be 'iconized' and brought back to full size whenever needed This interface is used by many other Windows products (like Microsoft Excel and Word) and should seem familiar to experienced Windows users

Project Views and Task Deletion

Different views of the selected task details can be seen in Project Engineer by changing views This is provided to help the project manager to look at the project in different perspectives

Objects in Project Engineer are never physically deleted Instead they are marked as deleted and 'hidden'

from view There are two reasons for this The primary reason is that certain modules like the Life cycle Advisor and Validator may need to access objects are 'missing' as well as those that are present Further, the user can easily 'undelete' a task at any point in time, without having to recreate all of the associated information A Hide/Unhide function will allow the user to toggle between views that include or exclude 'deleted' objects

The Software Architecture

The architecture employed for this project is shown in Fig 3 1 As usual the user interface forms the front-end and uses Windows version 3 0 Other tools that coexist with the Life Cycle Builder are shown as internal tools The front-end is connected to the database (back-end) via the LDI Database Interface The connection to CASE tools is planned but not yet implemented Similarly the interface to popular schedulers will be implemented when the licensing of the same is completed

The In-Memory Data Structure and N Levels

A copy of the tasks in the list window is stored in memory for faster access to display them in the list window whenever the window is repainted This feature also saves

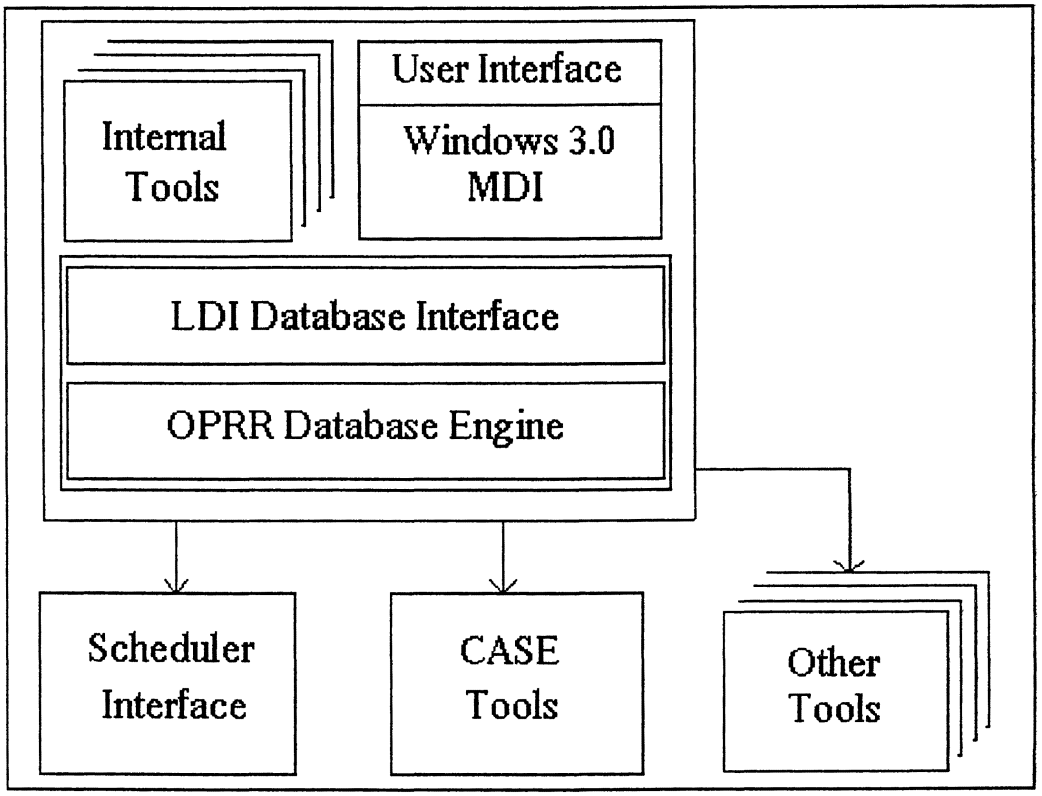
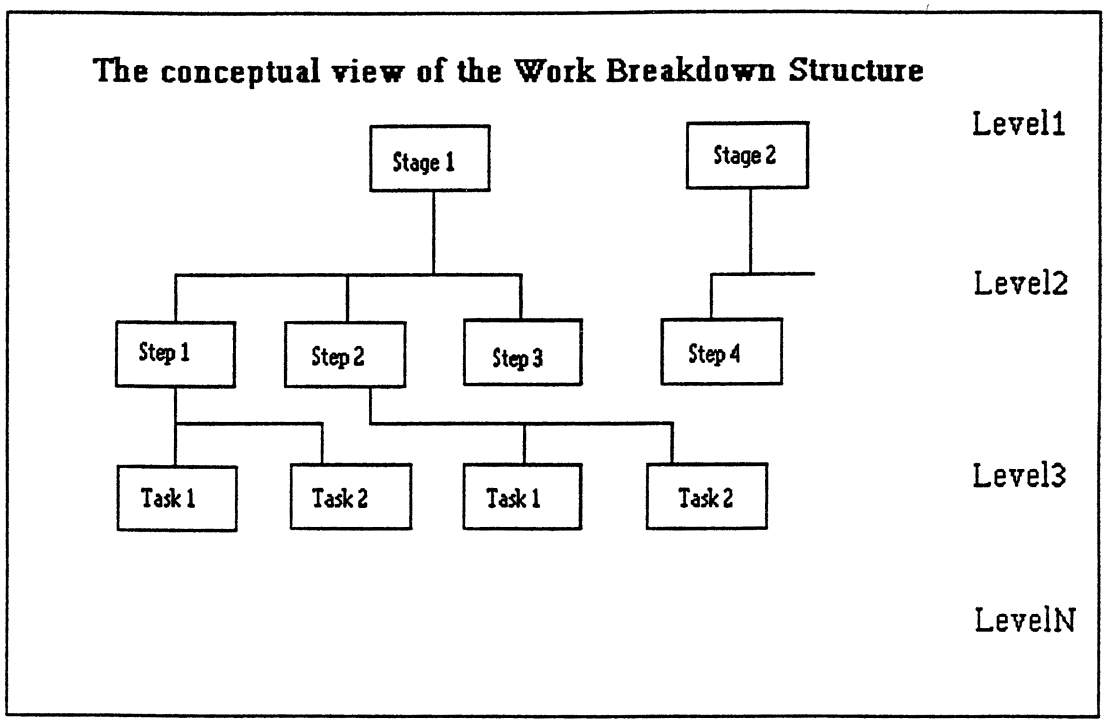


Figure 3 1 The Software Architecture

unnecessary disk accesses Since a project life cycle has a hierarchy, it can be stored as a left-linked right-sibling tree Since the maximum number of children in any level is not limited, the link structure is provided Figure 3 2 depicts the data structure as it is stored in memory

Object-Oriented Design

Project Engineer employs an object-oriented approach in the design of its modules The isolation of the interactive



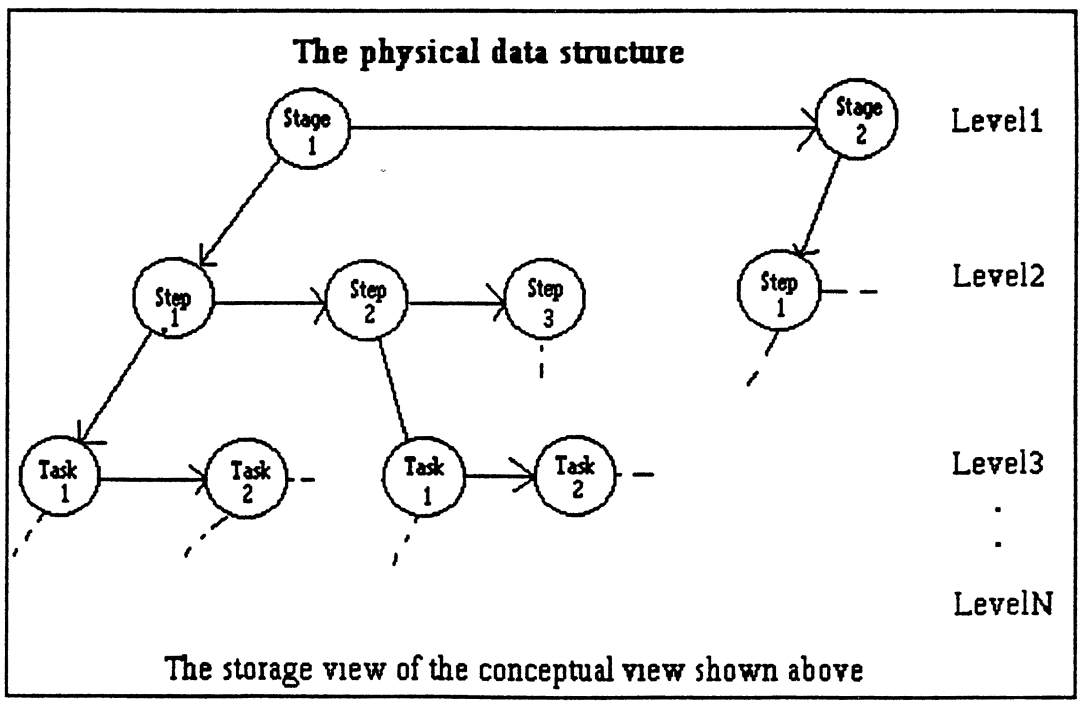


Fig 3.2 The left-linked right-sibling tree data structure

part from the application part in any software design has several advantages [Linton et al, 1989] This isolation can be achieved easily using object-oriented design providing abstraction and encapsulation Project Engineer is modular in design and incorporates an integrated set of tools that can be mixed and matched Each additional module will be a separate executable segment and uses Microsoft Windows version 3 0 message passing facilities to coordinate activities between modules and synchronize repository activity There is a subtle separation between the front end, which is what the user sees and uses and the back end, or the database A set of database interface routines are used to provide this separation Thus either the front or the back end can be changed without affecting the other significantly This helps in maintaining the software easily while adapting to changing environments with little difficulty If in the future the database engine needs to be changed, it can be done without extensive alteration of the code, only the interface routines will have to be modified This can be called database encapsulation

The concept of reusability is emphasized in this design Encapsulation allows us to build entities that can be depended upon to behave in certain ways, and to contain certain information [Wirfs-Borck, et al , 1990] Such entities can be reused in every application that can make use of this behavior and knowledge Microsoft Windows allows the creation of Windows Classes, which, once coded,

can be used for several different purposes The Grid Window Class described below is developed as a Windows Class It can even be made into a Dynamic Link Library (DLL) in future versions, which makes it easily usable with other modules whenever required Another window, the Object Editor, is also developed as a Windows Class Each instance of the Object Editor has its own in-memory storage area That is how the user can invoke several instances of the Object Editor and modify all of them simultaneously

The appendix gives an overview of all the modules that make up Project Engineer Project Engineer uses the concept of Multiple Document Interface (MDI) standards to permit several tools to run under the same application This has several advantages over simple individual windows design, multiple tools can be invoked at the same time and the interaction between tools can be established while providing the ability to display several windows (tools) simultaneously Two of those modules are selected for the purpose of demonstration They are the Risk Analysis module and the Life Cycle Builder module

The Risk Analysis module is coded using an expert system package (KnowledgePro) and later will be integrated into Project Engineer as another MDI child window

The Life Cycle Builder (LCB) module is an MDI child window in Project Engineer It uses the Grid Window class described below to display project information in a tabular fashion It helps the user bring up the Object Editor,

another MDI child window to help the user in editing any selected object Interaction between the Object Editor window and LCB window is provided by the MDI

The Grid Window Class

One of the most frustrating aspects of GUI applications is that the control mechanisms of the user interface are not always consistent While window, menu, and dialog box controls are fairly standardized, other user interface controls are not Designers must always balance the unique user interface requirements of their application against the benefits of adopting a familiar control mechanism Several control designs were evaluated/prototyped including a list box based control and an edit-control based control Finally the decision was made to develop a Grid Window Class because it is more user friendly and has been accepted by the users (in Microsoft Excel and Wingz spreadsheets) Even though it takes more time to develop this control, it can be used by other modules later

The Grid Window Class is designed to provide an interface for displaying and managing tabular data Since the data associated with projects has to be displayed from various perspectives with each row of objects/tasks representing a Work Breakdown Structure object, a grid structure would be appropriate Columns indicate the Name,

Status, Estimation information, etc Columns can be customized by the user to view what the user wants at any point in time. The columns are resizable with the use of a mouse It is essential to be able to accommodate the variety of data that can be displayed in each column For example, the Name column for any WBS object can be up to sixty characters in size whereas the WBS code will have a maximum width of fifteen characters If the same column is used for both these situations, the user can reduce/enlarge the column width The resizing of columns will be achieved by direct manipulation Scroll bars will be provided to scroll the grid vertically or horizontally

Rows and columns can be 'highlighted' as in Microsoft Excel A highlighted rectangle appears whenever the user selects a particular 'cell', this action will remove the highlighted rectangle from the previously selected area Simultaneously, the contents of the selected cell will be displayed in an 'edit' window where it can be modified

Since this grid structure design allows the display of information in a matrix fashion and since this is designed and implemented as a Windows Class, it can be reused in other modules by just creating another instance of this class

The LCB Object Editor

The purpose of this MDI child window is to display all the information for one Activity, and allow the user to edit any of the information fields After a modification, the user either can accept or cancel the changes made Modifications to an Activity description will require an explicit commit action (via an 'OK' button) before any other functions can be accessed This requirement ensures a synchronization among all MDI child windows

The object editor 'pops' up when the user 'double-clicks' the mouse button on any selected Activity in the List window of the Life Cycle Builder module This editor also validates the modified data for any error in the formatting For example, if the user enters an alphabetical string for a date field, an Error MessageBox is displayed requesting the user to input the data in proper format

The object editor is filled with the data associated with the selected Activity in the List window and the user can modify any field The same object editor 'pops' up when the user tries to insert a 'new' activity in the List window, in this case the fields will be empty except for the default Activity type Each insertion is checked/validated to make sure that the Activity level hierarchy is maintained

Fig 3 3 shown on the next page gives an idea of how the Object Editor is displayed when the user double-clicks

an object in the LCB window. It may be observed that PI.PI1.10 is selected, since that row is highlighted in inverse. There also are two other Object Editors which have been iconized.

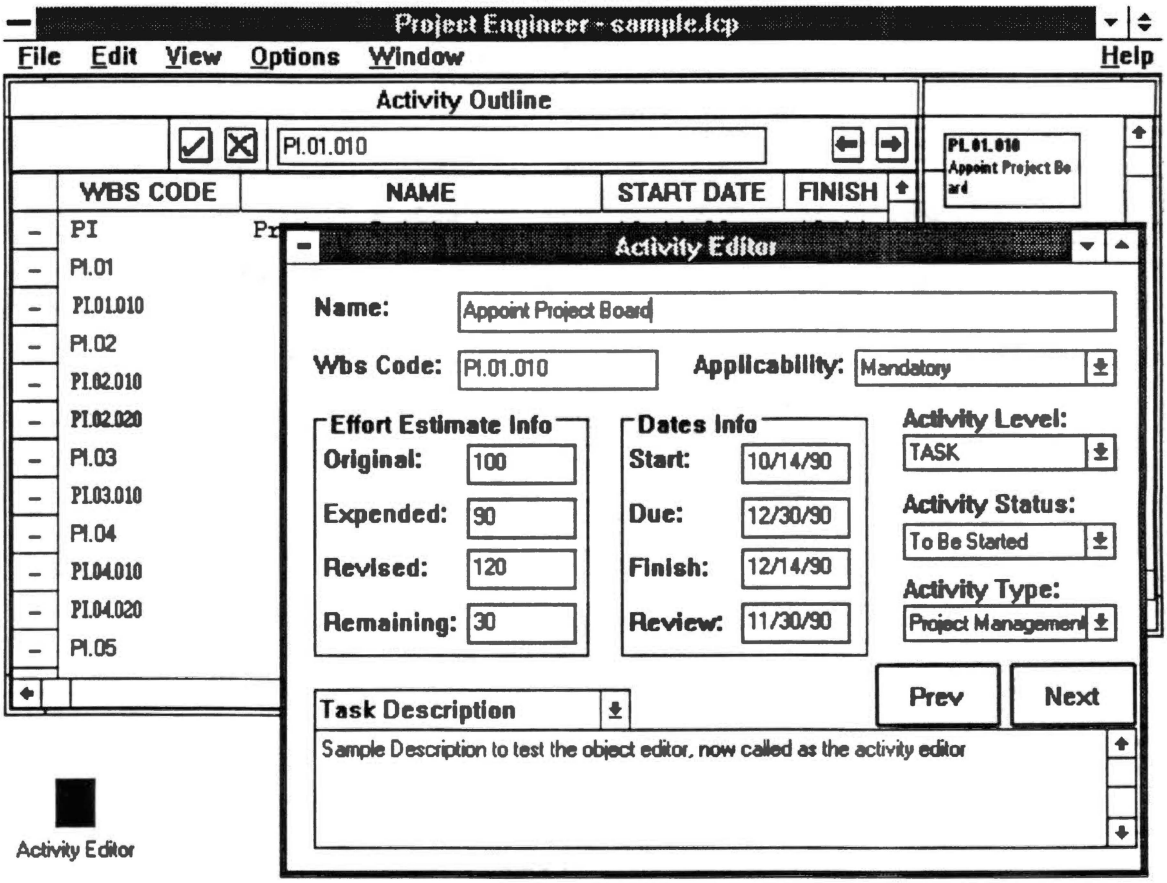


Fig. 3.3 The Object Editor.

Project Modules

Project Engineer consists of several major modules, each of which is an independent tool. The modules are (the names indicate their purpose) :

- Life Cycle Builder (described below).
- Estimator
- Scheduler Interface (since there are several scheduler packages existing, only an interface to common ones are provided)
- Resource Manager
- Metrics Tracker
- Life Cycle Validator
- Life Cycle Advisor
- Hypertext Method
- Risk Analysis

Only the Life Cycle Builder module is selected for demonstration of this concept A prototype of Risk Analysis is also developed. To develop the whole package with all tools is outside the scope of this thesis

Life Cycle Builder

This module provides core services for Project Engineer It is the framework for all Project Engineer activities and provides underlying services for other modules (OPRR database engine and Traffic Controller)

Risk Analysis

This module supports Project Engineer's other modules by analyzing the risks associated with a software

development project and recommends the actions to be taken to minimize those risks

CHAPTER IV

FEATURES OF LIFE CYCLE BUILDER (LCB)

Introduction

The Life Cycle Builder module is the framework for all Project Engineering activities and provides underlying services for other modules Its purpose is to allow users to load and manipulate project life cycles and templates The following are some of the features of the LCB

- Load a project database or a Life Cycle template
- Save/Save_As projects and templates
- Display various views of the Life Cycle
- Print the Life Cycle plan at various levels of detail
- Insert, Delete, Modify, Copy, Paste Activities
- Promote and Demote Activities
- Collapse and Expansion of Activities
- Explode an Activity to a detailed description
- Hide/Unhide 'deleted' tasks
- Export to other scheduler packages
- Allow the user to customize the appearance of a Life Cycle with fonts, etc
- Provide On-line Method (hyper-media based help) integration and synchronization [Garg, 1990]

How to Use LCB

The LCB is designed to be used by the project managers to create and maintain project life cycles. Project life cycles can be created from existing templates or afresh (by selecting an empty template). To start with, the user can open a new project life cycle by selecting an appropriate template that might closely match the new project. Then appropriate names are given to tasks along with their tentative starting dates and ending dates, the estimated duration of that task etc. All unwanted tasks can be deleted and new ones added as required by either copying other tasks and changing the parameters or by inserting a new task wherever needed and filling in the details At least one parameter, the Work Breakdown Structure code should be filled, as it is the unique factor that differentiates one task from an other. The edited project can be saved, either as a project or a template, and used as.

The details regarding a task can be edited by selecting the object editor, which is brought up by double-clicking the required task Another way to edit the fields in the list window is to select the field by clicking the mouse. The selected item appears in an edit window (as in Microsoft Excel) and can be edited there. The changes automatically appear in the selected field too The changes may either be

accepted or deleted by selecting the appropriate button next to edit window.

The LCB List Window

This child window will display the Life Cycle in list form, in a scrollable window. Entries (Work Breakdown Structure objects) in the window can be assigned different fonts and sizes based on styles, where styles are assigned by level in the Work Breakdown Structure.

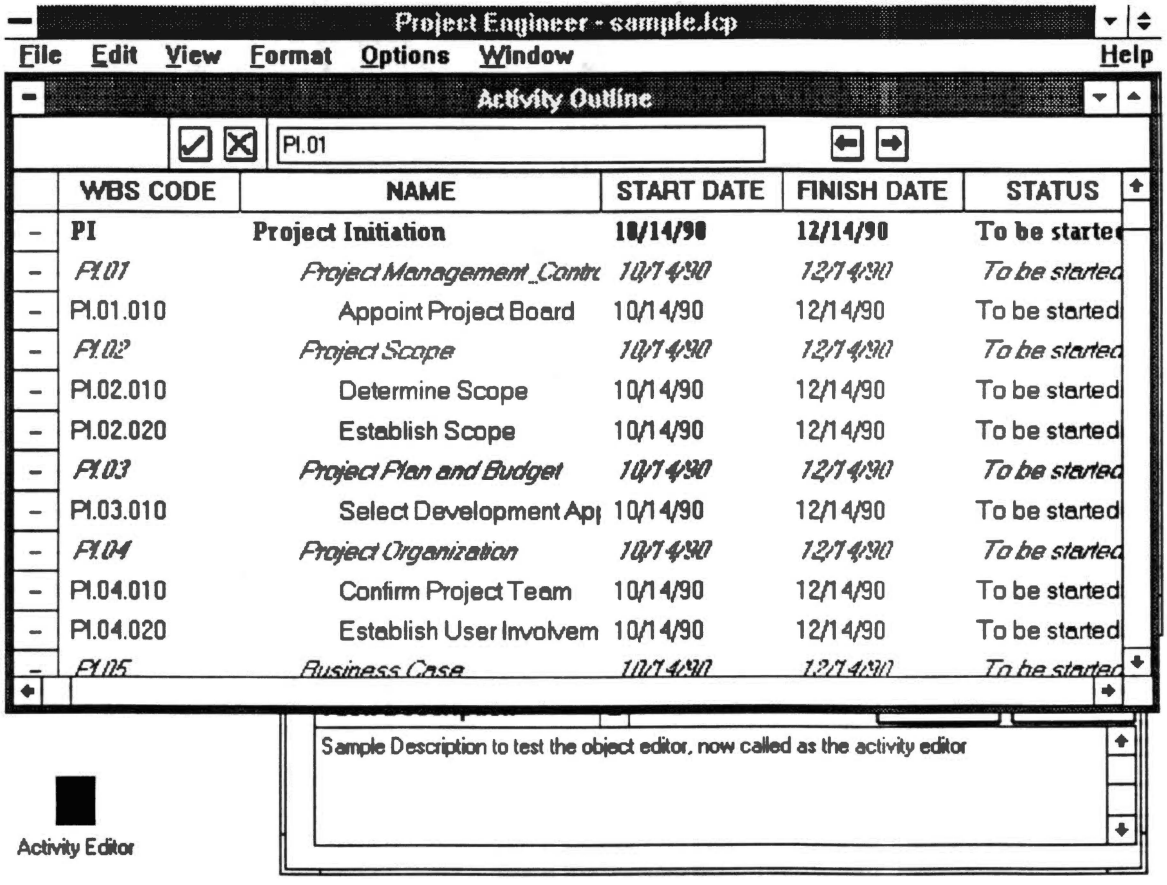


Fig. 4.1 Project Engineer LCB window.

The interface style for the list window will closely emulate the grid class feature found in Microsoft Excel. Figure 4.1 is an actual picture of Project Engineer LCB window with actual objects filled in. The list window will also provide column configurability. Column configurability will allow choices of column formats (i.e., different combinations of columns customized for a particular purpose). The list window will support character attributes determined by the activity level. Each activity level (Stage, Step, Task, etc.) can have its own font, style, and size. The List Window also supports an outline format (with activities indented for appropriate levels) and a straight list format. Users can highlight rows/lines, columns, cells or rectangular areas, and perform editing of the selected items Column widths can also be resized dynamically, and the list window will automatically clip text outside a boundary.

Addition and Deletion of Objects

Project tasks can be added or deleted as and when needed. Addition of the tasks is done by either inserting a new task wherever required and filling in the items or by copying another task and pasting it at the requisite place. The user interface for this operation follows Microsoft Excel's format Similarly deletion of objects can be done

by selecting the objects and deleting them. For these operations the edit menu item provides Copy, Paste, Delete etc. as a standard practice. Objects are not actually deleted, they are only hidden. The reason for this is that the Life Cycle Validator module check upon the consistency that exists in the project life cycle and provides suggestions. The deleted tasks may be for producing a deliverable that is needed in other tasks. In such cases it (Life Cycle Validator) points to these errors.

Collapse and Expansion of Objects

Since the project life cycle's work breakdown structure follows a hierarchy, collapse and expansion of levels may be needed This abets the user to see only the top hierarchy or the details regarding a particular phase, etc. This feature is very useful because the user can see only a certain number of tasks at any time on the screen, or he/she has to scroll the window to see more tasks, which may be clumsy at times.

This feature is provided under the View menu item and also as a button in the first column of the list window, which, when double-clicked, will toggle between collapse and expansion of tasks

For example assume that the tasks given below are shown in the window. By collapsing 'PI.PI1', we get the situation rendered below.

PI - Project Initiation (Stage)
PI.PI1 - Determine Scope (Step)
PI.PI1.10 - Review Related Studies (Task)
PI PI1.20 - Establish Scope
PI PI1.30 - Establish Major Objectives
PI PI1 40 - Establish Constraints
PI PI1.50 - Identify Outline Solution
PI.PI2 - Establish Project Plan and Budget

.

After collapsing 'PI.PI11', we get

PI - Project Initiation (Stage)
PI.PI1 - Determine Scope (Step)
PI PI2 - Establish Project Plan and Budget

Similarly, this will be restored to the previous (expanded) state when the above mentioned button is double-clicked again

Promotion and Demotion of Objects

The Work Breakdown Structure divides the tasks into several level heirarchies For example, in the LBMS methodology [LBMS Systems Engineering Methods Handbook,

1990], there are three levels and these levels can vary. The proposed tool, Project Engineer has 'n' levels, even though ten levels seems to be sufficient for most project life cycles. An object of Level1 will have several Level2 objects as its 'children' and so on. A task of Level1 can be demoted to Level2 or a task of Level3 can be promoted to a Level2 This is needed because a task might become too important for it to be in that level and should be promoted to a level above and children be provided for it Likewise a task at a higher level may become so trivial it should not remain at that level and be demoted A task at any level can only be promoted/demoted to its previous/next level There are certain restrictions though To exemplify this, consider the following work breakdown structure

```
Level 1   PI - Project Initiation (stage)
Level 2        PI PI1 - Determine Scope (step)
Level 3             PI PI1 10 - Review Related Studies(task)
                    PI PI1.20 - Establish Scope
                    PI PI1 30 - Establish Major Objectives
                    PI PI1 40 - Establish Constraints
                    PI PI1 50 - Identify Outline Solution
Level 2        PI PI2 - Establish Project Plan/Budget (step)
Level 2        PI PI3 - A new step (step)
```

The following promotions are valid :

1). PI.PI1.30 (task) at level three can be promoted to level two, in which case PI.PI1 (step) will now have only two children (PI.PI1.10 and PI.PI1.20) and PI.PI1.40 and PI.PI1 50 become children of the promoted task PI.PI1 30.

2). PI.PI2 (step) at level 2 can be promoted to level one. Now PI has only one child at level two, viz, PI.PI1.

An example of an invalid promotion is trying to promote PI.PI1 at level 2 to level 1, because now level 1 will have children which are at level 3 which is not correct.

Promotion and demotion of objects between levels are possible only when the work breakdown structure is logically valid

These two features are also provided under the View menu item.

The WBS Diagram Window

The WBS Diagram Window is another MDI child window, and can be activated from the tools palette This provides a graphic display of the WBS structure. Each Activity will be displayed as an icon containing to indicate the Activity code. A graphic library supplied by LBMS Inc., London, were

used to develop this window. At this stage of development, users are not able to edit the structure from this window, but they are able to navigate through the structure by clicking and double-clicking icons to display the Object Editor.

All operations within the diagram (highlighting, collapse/expansion, etc) will synchronize with the list mode window (if visible) and the Object Editor (if visible).

This window is used for viewing the project life cycle in a graphical form. Objects can be selected and the object editor brought up by double-clicking on the object Collapsing and expanding can be performed by clicking on the oval that joins two levels. A '+' appears in the oval when the level is collapsed.

CHAPTER V

RISK ANALYSIS

Introduction

This is one of the several modules of the Project Engineer tool A prototype of this module was developed for study purposes Risks that are involved in a (software) project can be reduced if proper action is taken The Risk Analysis module leads the project manager through a consultancy session using a Risk Analysis Questionnaire, to produce a log of questions asked and the replies given, and provides the option of producing a hardcopy of this log The report produced details the areas of risk associated with the project and makes recommendations of actions required to minimize risks This report should be viewable in a scrollable window and may optionally be printed on a hardcopy device

This module was only developed for study purposes and is not part of the thesis

Project Risks

A software project depends upon various things such as hardware, software, and other subsystems to be completed successfully These are called project risks Any of these items can go wrong An expert system to compute these risks is being written using KnowledgePro, an expert system that runs on Windows version 3 0 on a PC platform These risks can be divided into several areas

Risk Areas

The level of risk is assessed in five different areas

External dependencies of a project
Organizational dependencies
Planning risks
Business case risks
Technical and Implementation risks

The risk level calculated is based upon the project manager's responses to a pre-defined set of questions and supporting rules regarding these questions A sample of two questions with answer choices is given below

Q Is the project dependent on scarce resource/skills?
A Yes, No, Skip and Previous.

Q The number of major subsystems in the project is?

A. 1. 2. 3. etc . and Skip and Previous.

The project manager may or may not answer all the questions for reasons such as the following·

a) The project manager may not have the information to answer all the questions at that time
b) The question can be answered only partially

In either case the project manager can either skip the question entirely or answer the question partially For example, the project manager may answer a particular question in this way

> "There is a 20% chance that a particular function will fail, rather than a definite YES or NO answer regarding the possibility of that function failing "

In other words a fuzzy logic approach may be used [Leber, 1990] Currently, the prototype is not following fuzzy logic, but there are plans to implement it in a future version

Possible Solutions

Once the questions have been answered and the corresponding rules applied, a report is produced The risk analyzer produces this report recommending those actions

required to reduce or minimize the level of risk in specific areas The report also mentions those questions that were not answered and that the level of risk might increase depending upon the answers to those questions The report may be viewed on a screen, copied to a file or printed

The Risk Analysis option is provided under the Tools options of the Project Engineer as an icon which when double-clicked will activate the icon The prototype is developed using Decision 123, an AI development package, and runs on KnowledgePro, another rule-based development tool under Windows 3 0 This module has to be integrated with Life Cycle Builder

The computed value of risk is then converted into an estimating factor which may be used in the Estimator module

CHAPTER VI

CONCLUSIONS AND FUTURE WORK

Life Cycle Builder is the first module, designed and built in the integrated set of tools detailed in the Appendix A. Since a project life cycle has to be built in any project first, the project life cycle is the first module to be designed and developed The scheduler interface to couple Project Engineer to other packages is developed next. Parallel to this, the next most important modules in Project Engineer, the estimator and a client server for the database, have to be completed.

The user interface is on par with the existing Windows version 3 0 based tools. Minimum documentation will be needed for the user to start and use this tool, and since it is coded in C, the speed is quite satisfactory The OPRR database library consumes about 125K of memory and the LCB module uses about 100K of memory The graphics library used in the WBS diagram editor is 450K in size and when compiled with the driver takes up about 200K of memory It is suggested that Project Engineer be run in 386 enhanced mode in Windows version 3.0 rather than in the real mode.

The application may require other MDI child windows to be added for clarity and better user interface It is also possible that some of the windows described above may change in their functionality as the development continues in future

Multiple font support can be afforded for displaying tasks in different fonts for various purposes A hot link (please refer to the glossary for details) to other scheduler packages can be provided instead of an interface

Most of the items described above will be accomplished in future Of course, there is a chance that some of the minor points may be missed In fact, several of the items mentioned above have been completed and tested satisfactorily already

SELECTED BIBLIOGRAPHY

Beck, L. B., and Perkins, T. E , "A Survey of Software Engineering Practice Tools, Methods, and Results", IEEE Transactions on Software Engineering, pp 541-561, Vol SE-9, No 5, September 1983

Boehm, B W , "A Spiral Model of Software Development and Enhancement", Computer, pp 61-722, May 1988

Cameron, J R , "An Overview of JSD", IEEE Transactions on Software Engineering, Vol SE-12, No 2, pp 222-240, February 1986

Coad, P , and Yourdon, E , Object Oriented Analysis, Yourdon Press, New Jersey, 1990.

Constantine, L L , "Objects, Functions, and Program Extensibility", Computer Language, pp 34-54, January 1990

Cornish, M , "Four Principles of User Interface Design", Computer Language, pp. 67-75, March 1990

Cox, B J, Object Oriented Programming: An Evolutionary Approach, Addison-Wesley Publishing Company, Massachusetts, 1987

Daly, E D , "Management of Software Development", IEEE Transactions on Software Engineering, pp 229-242, May 1977

Decision 123 AI package with Reference Manual, Proctor & Gamble Artificial Intelligence Team, 1990.

Dodani, M. H , Hughes, C. E., and Moshell, J. M , "Separation of Powers", BYTE, pp 255-262, 1989

Garg, P. K , and Scacchi, W , "A Hypertext System to Manage Software Life-Cycle Documents", IEEE Software, pp. 90-98, May 1990

Hsieh, D , Personal communications with David Hsieh, 1990

IBM Systems Application Architecture - Common User Access Advanced Interface Design Guide, International Business Machines Corp., 1989

Kemmerer, R A , "Integrating Formal Methods into Development Process", IEEE Software, pp 37-50, September 1990

Kernighan, B W , and Ritchie, D M , The C Programming Language, Prentice Hall, Englewood Cliffs, New Jersey, 1978

KnowledgePro (Windows) rule-based development system with its Manuals, Knowledge Garden, Inc , 1990

LBMS Systems Engineering Methods Handbook, LBMS Inc , Houston, 1990

Leber, J B , "A Fuzzy Approach to Data Repair", Database Programming & Design, January 1990

Lee, E , "User-Interface Development Tools", IEEE Software, pp 31-36, May 1990

Linton, M. A., Vlissides, J. M., and Calder, P R , "Composing User-Interfaces with Interviews", Computer, pp 8-22, February 1989

Luqi , "Software Evolution Through Rapid Prototyping", Computer, pp 13-25, May 1989

Matthews, R W , and McGee, W. C, "Data modelling for software development", IBM Systems Journal, pp 228-235, Vol. 29, No 2, 1990

McClure, C , CASE is Software Automation, Prentice Hall, New Jersey, 1989

McClure, C , "The CASE Experience", BYTE, pp 235-246, April 1989

Microsoft Windows Guide to Programming for Windows 3.0, Microsoft Corporation, 1990

Microsoft Windows Programmers Reference for Windows 3 0, Vol 1 and Vol 2, Microsoft Corporation, 1990

Microsoft Windows Programming Tools for Windows 3 0, Microsoft Corporation, 1990

Muller, M J , "Multifunctional Cursor for Direct Manipulation User Interfaces", SIGCHI, 1988

Myers, B J , "Creating User Interfaces by Demonstration", Academic Press, Inc , California, 1988

Orr, K , Gane, C , Yourdon, E , Chen, P P , and Constantine, L, L , "Methodology The Experts Speak", BYTE, pp 221-233, April 1989

Paterson, T , and Flenniken, S., "Managing Multiple Data Segments Under Microsoft Windows", Dr.Dobbs Journal, February 1990

Petzold, C., Programming in Windows, Microsoft Press, 1988.

Petzold, C , "A New Multiple Document Interface API Simplifies MDI Application Development", Microsoft Systems Journal, pp 53-63, July 1990

Pressman, R. S., Making Software Engineering Happen, Prentice Hall, New Jersey, 1988.

Reisman, S , "Management and Integrated Tools", IEEE Software, pp 71-78, May 1990

Schildt, H , C The Complete Reference Second Edition, Osborne McGraw Hill, California, 1990

Shneiderman, B , Designing the User Interface Strategies for Effective Human-Computer Interaction, Addison-Wesley Publishing Company, 1987

Towner, L E , CASE Concepts and Implementation, McGraw-Hill Book Company, 1989

Ward, P T , "The Transformation Schema· An Extension of the Data Flow Diagram to Represent Control and Timing", IEEE Transactions on Software Engineering, Vol SE-12, No 2, pp. 198-210, February 1986

Welke, R J., "Meta Systems on Meta Models", CASE Outlook, pp 35-43, Vol 4, 1989

Wirfs-Brock, R., Wilkerson, B., and Wiener, L., Designing Object-Oriented Software, Prentice Hall, New Jersey, 1990

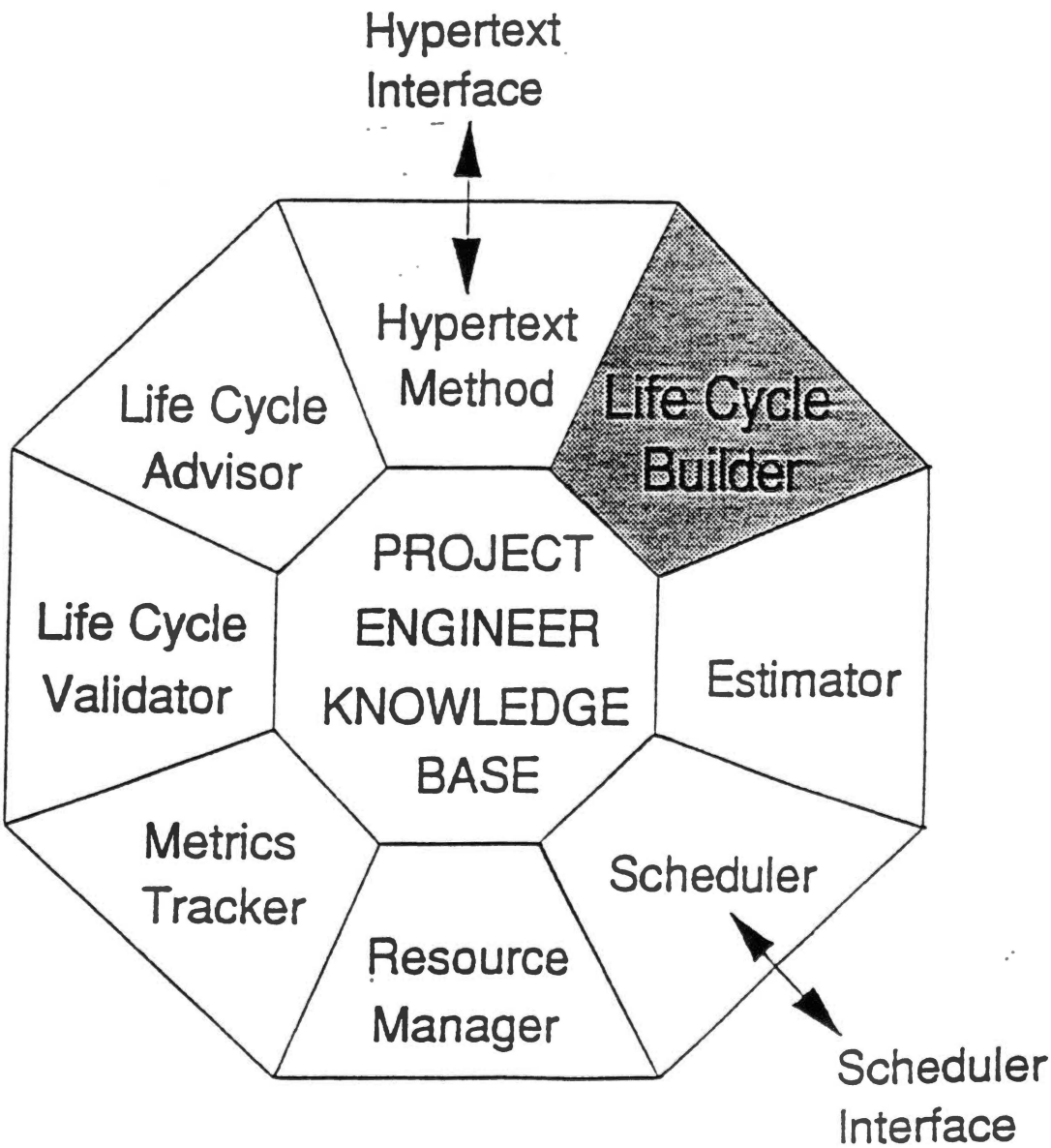
Zells, L., Managing Software Projects, QED Information Sciences, Inc., Massachusetts, 1990

APPENDIXES

APPENDIX A

PROJECT ENGINEER MODULES

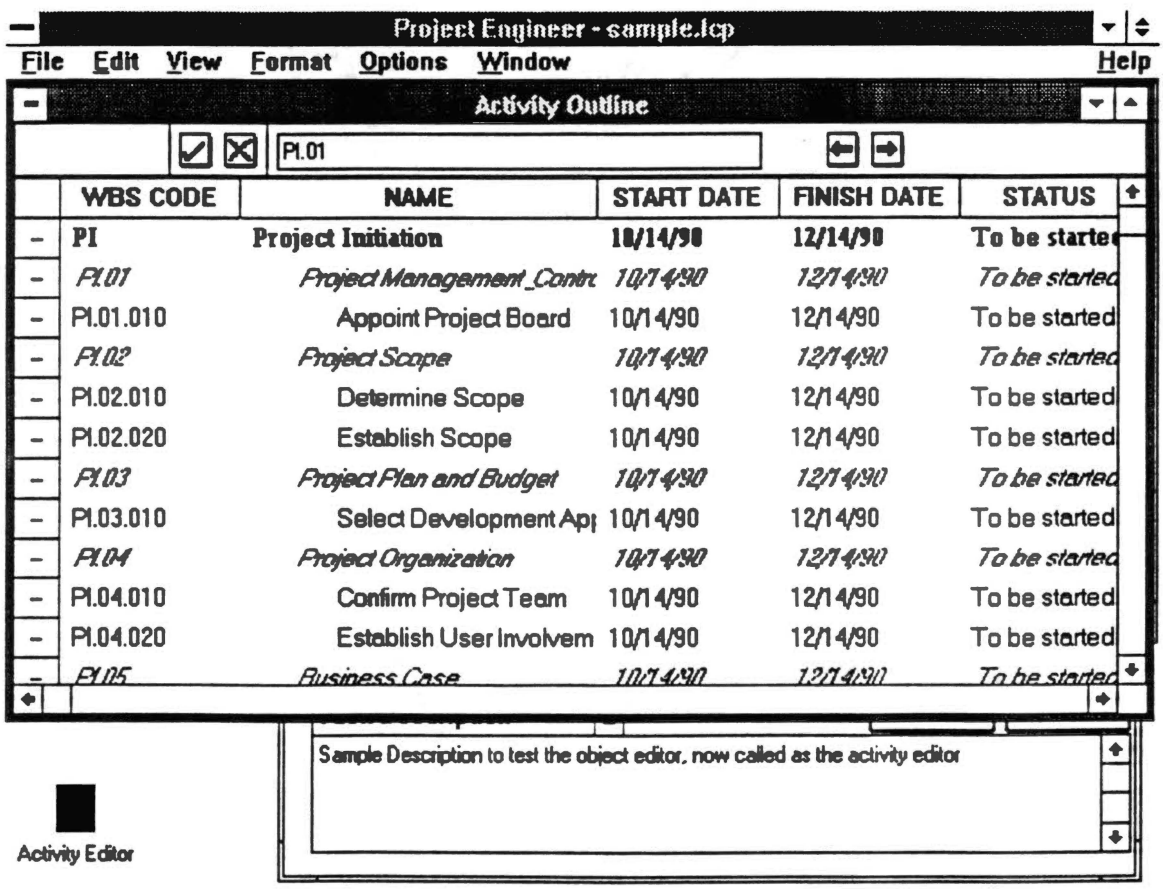
The figure presented below gives an idea of all the modules present in Project Engineer.



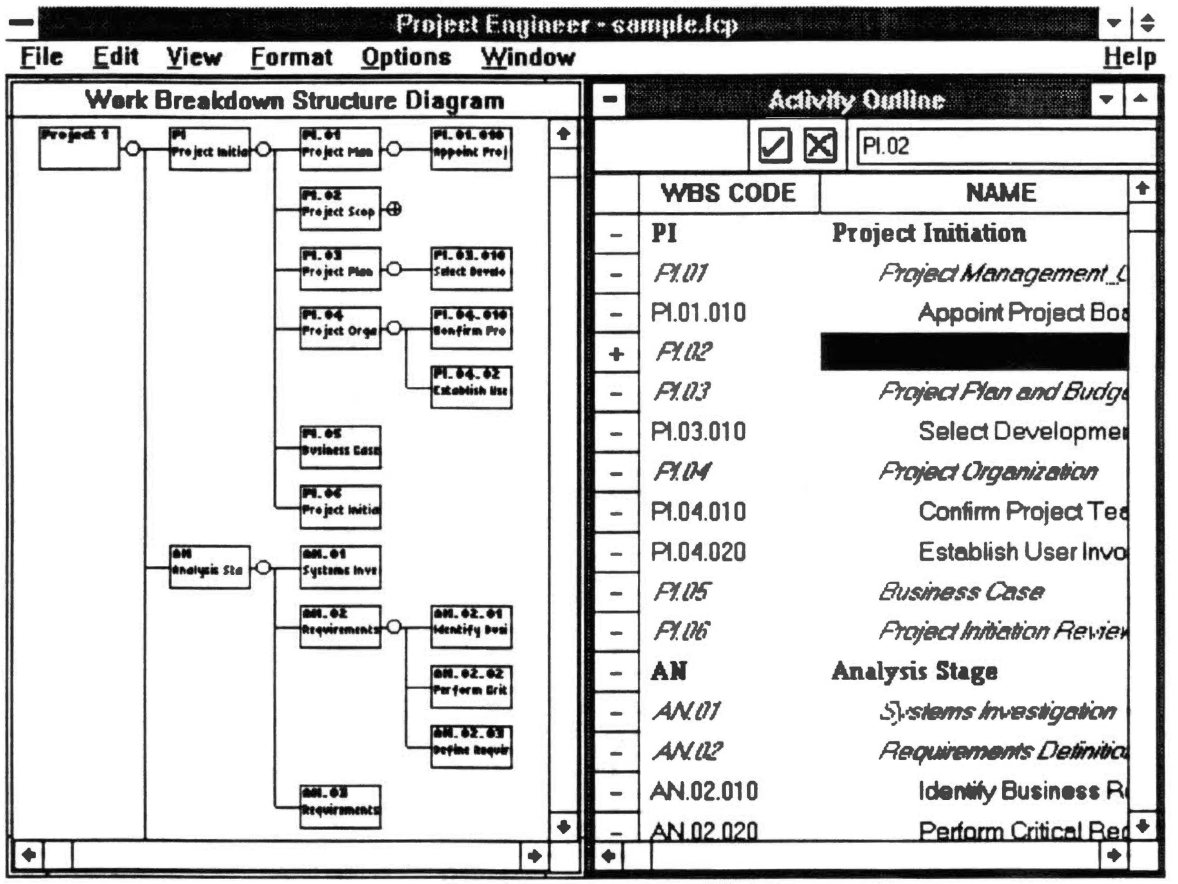
APPENDIX B

PROJECT ENGINEER SNAPSHOTS

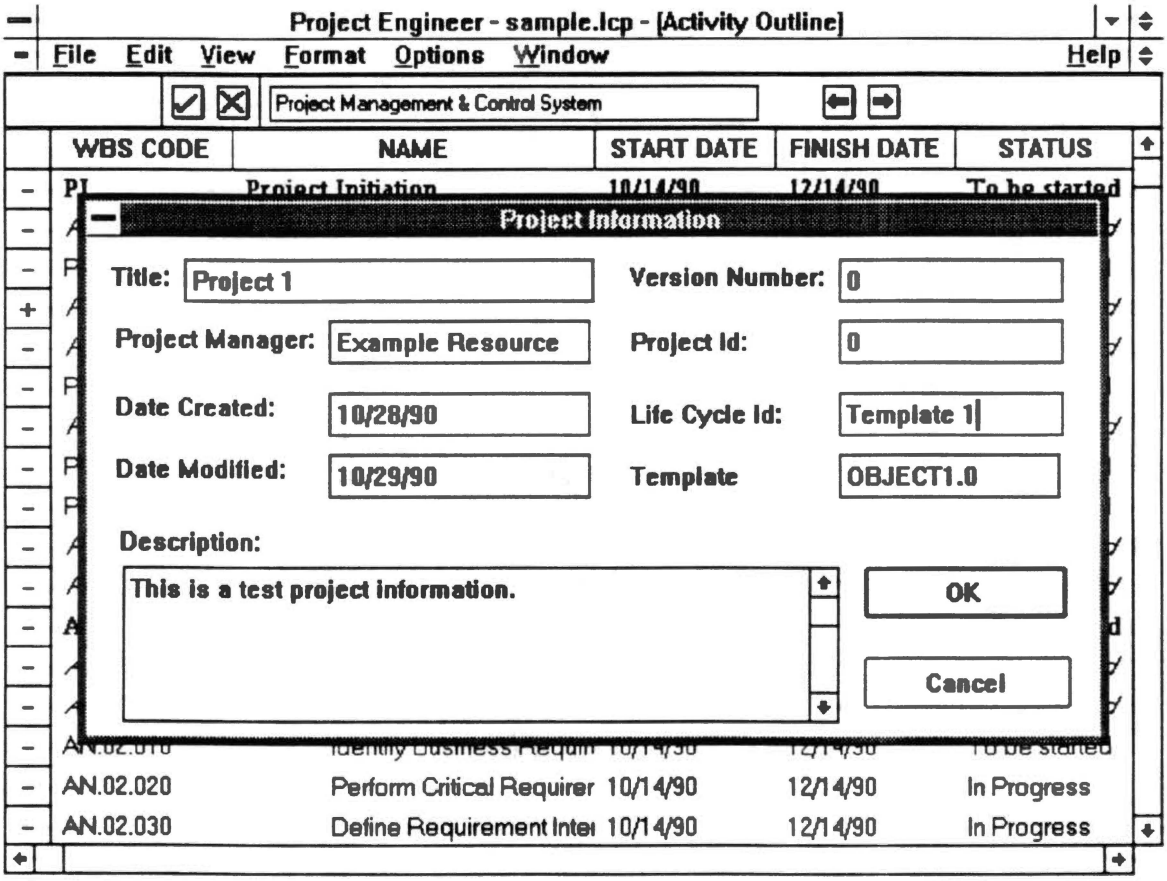
The figure presented below gives an idea of how the Life Cycle Builder module looks like.



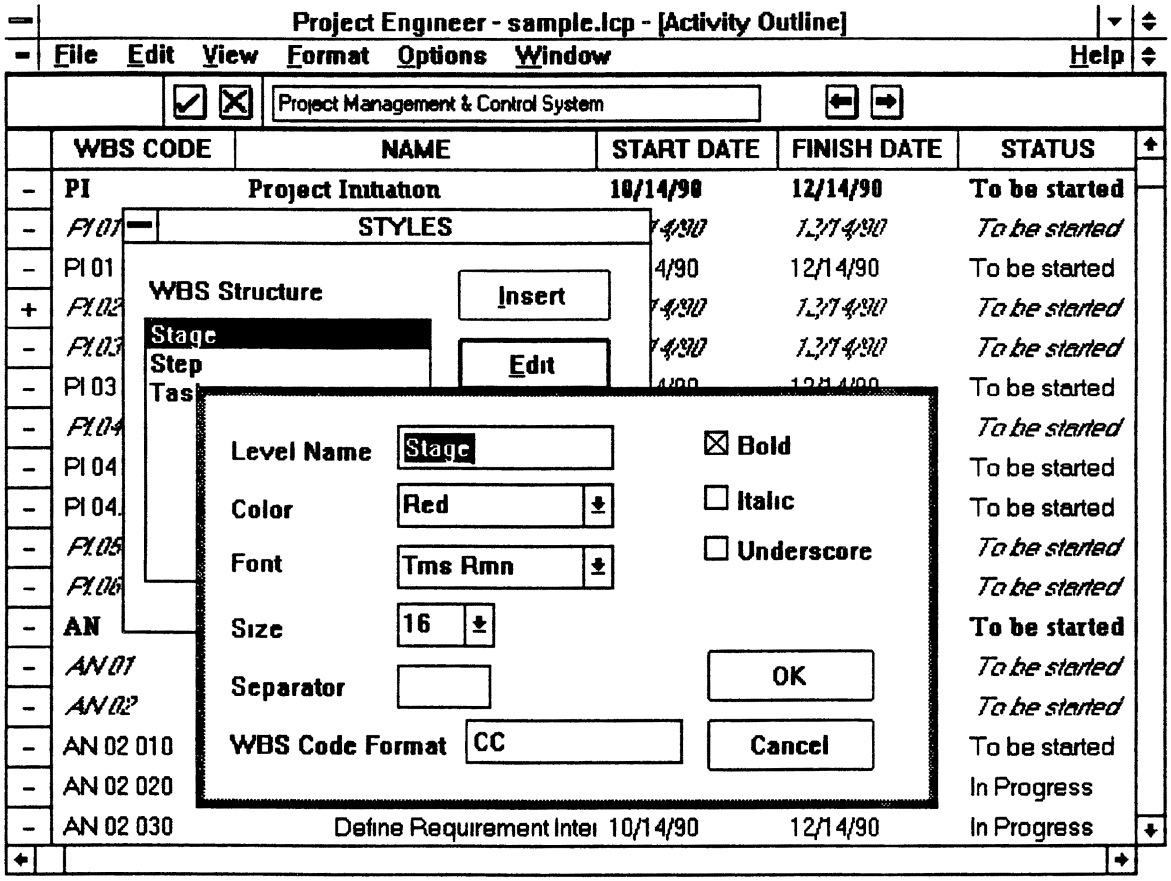
The figure presented below shows WBS Diagram Editor and Life Cycle Builder modules 'tiled' next to each other. Also observe that the activity 'PI.02' is 'collapsed' and that can be seed in both the modules. A '+' sign indicates that the task is collapsed.



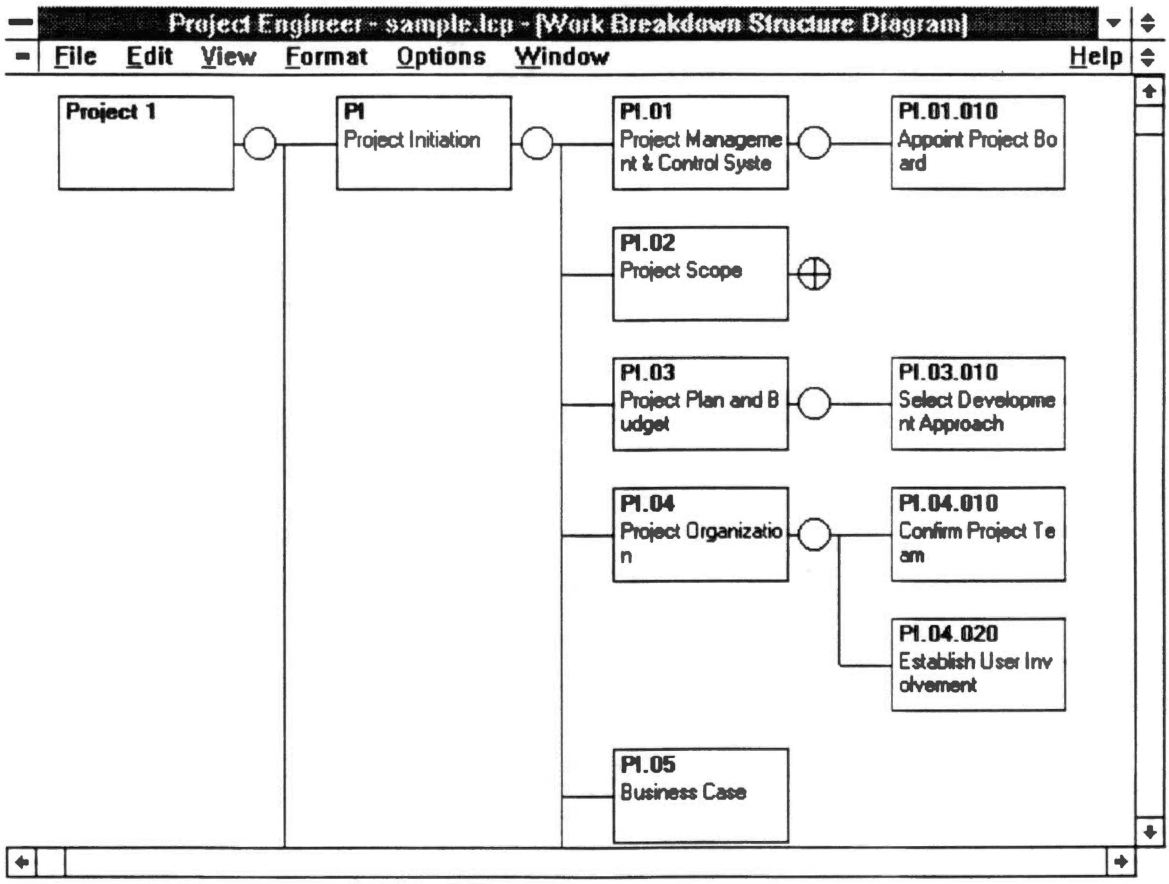
The figure presented below shows project information dialog box used to display the information about any project. It can also be modified as the project progresses.



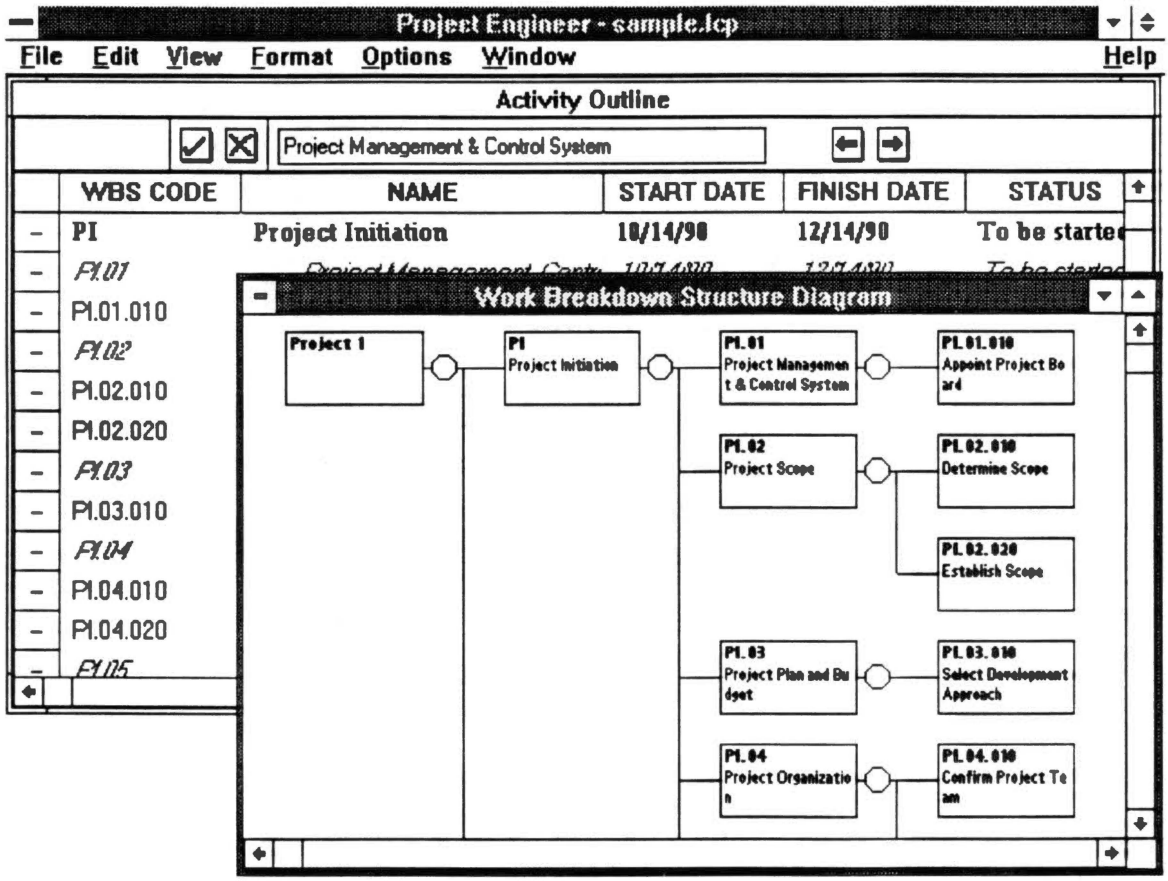
The figure presented below shows Styles dialog box. This is used to change the font, size, color, style (bold, italic or underline) of any activity level (stage, step or task) in the project. It is also used to change the WBS code format.



The figure presented below shows the WBS diagram. This presents the graphical view of the project work breakdown structure



The figure presented below shows the WBS Diagram Editor on top of the Life Cycle Builder. This shows that multiple windows can co-exist in the same tool and help the user in viewing the project in different ways. This is achieved using the Microsoft Windows 3.0's multiple document interface (MDI) feature.



VITA

Srıdhar Chandrashekar

Candıdate for the Degree of

Master of Scıence

Thesıs: AN INTEGRATED SET OF TOOLS TO ASSIST IN THE DEVELOPMENT AND MAINTENANCE OF PROJECT LIFE CYCLES

Major Fıeld: Computer Scıence

Bıographıcal:

Personal Data: Born ın Bangalore, Indıa, June 20, 1963, the son of K.S. Chandrashekar Iyer and R. Sarojamma.

Educatıon: Receıved Bachelor of Engıneerıng Degree ın Electronıcs and Communıcatıons Engıneerıng from Unıversıty of Mysore, Mysore, Indıa ın January, 1986; completed requırements for the Master of Scıence degree at Oklahoma State Unıversıty ın July, 1991.

Professıonal Experıence:
Research Assıstant, Department of Computer Scıence, Oklahoma State Unıversıty, September, 1990, to December, 1990.
Software Engıneer, Learmonth & Burchett Management Systems, 1800, W. Loop. S, Suıte 1800, Houston, Texas, May, 1990, to August, 1990.
Research Assıstant, Department of Busıness Admınıstratıon, Oklahoma State Unıversıty, September, 1989, to Aprıl, 1990.
Scıentıst, Defense Research and Development Organısatıon, Bangalore, Indıa, October, 1987, to August, 1989.
Customer Support Engıneer, Hındustan Computers Lımıted, Bangalore, Indıa, Aprıl, 1987, to September, 1987.