UNIT SYLLABUS

UNIT 1

Introduction to Software Engineering: Definition – Size Factor – Quality and Productivity of Factors – Managerial Issues – Planning a software project: Defining the problem – Developing the problem – Developing a solution Strategy – Planning the Development Process – planning an Organization structure – Other Planning Activities.

UNIT 2

Software Cost Estimation: Software cost factors – Software Cost Estimation Staffing level Estimation – Estimating Software Maintenance Costs – The Software Specification – Formal Specification Techniques

UNIT 3

Software design: Fundamentals Design Concepts – Modules and Modularization cost Design Notations – Design Techniques – Detailed Design Considerations –Real-Time and Distributors - System Design – Test Plans – Milestones,walkthrough, and Inspections.

UNIT 4

Implementation issues: Structured Coding Techniques – Coding Style – Standards and Guidelines – Documentation guidelines – Type checking – Scoping Rules –Concurrency Mechanisms.

UNIT 5

Quality Assurance – Walkthroughs and Inspection – Static Analysis – Symbolic Executions Unit Testing and Debugging – System Testing – Formal Verification:

TEXT BOOK

SOFTWARE ENGINEERING CONCEPTS

-Richard Fairley

UNIT-1

INTRODUCTION:

Software product has multiple users, multiple developers and maintainers.
To Develop a software product

1.user needs and constraints must be explicitly stated.
2.Source code must be implemented and tested.
3.Supporting documents must be prepared.(Principles of operation,users manual,installation instruction,training aids,maintenance documents)

Software maintenance task

1.Analysis of change request.
2.Redesign
3.Modification of the source code.
4.Thorough testing of the modified code.
5.updating documents and documentation.
6.Distribution of the modified work to the user.

Software Engineering is based on the foundation of

1.Computer Science
2.Management Science
3.Economics
4.Communication Skill
5.Engineering approach to problem solving

Software Engineerig requires

♉ Technical skill
♉ Managerial control
 Management science provides foundation for software project management.
 Computing System should be developed and maintained on time and within cost estimate.
 Economics provides resource estimation and cost control.
 High Degree o fcommunication is required among customers,managers,hardware
 Engineers,software Engineers and other technologist.
 Engineerig problem solving techniques provides basis for

1.Project planning
2.Project management
3.Systamatic Analysis
4.Design
5.Extensive Validation
6.Ongoing maintenance activities.

Fundamental Priniciple of Software Engineering is to design software products that minimize the intellectual distance between the problem and the solution.

Modules in Software Engineering

♉ Units of decomposition
♉ Software modules have both control interface and data interface.
♉ Relationship among modules is called control interface.
♉ Parameters passed between the modules is called data interface.

Advances in Software Engineering :

1.Analysis technique-Determining the software requirements.
2.Metodological approaches- Software Designing.
3.Implementation techniques-Source code.
4.Software validation techniques-Examine and Quality assurance
5.Formal techniques-Verify

Some Definition:

♉ Software Engineering differs from Computer programming
♉ In Software Engineer Engineering like techniques are used to specify,
♉ Design,implement,validate,maintain software products within the time and budget.
♉ On small project(1 or 2 programmers,duration is 1 or 2 months) primarily technical.
♉ On projects involving more programmers and longer durations, mana
gement control is required to coordinate the technical activities.

Terms:

Programmer:

Denote an individual who is concerned with details of implementing, packaging and modifying algorithms,data structures,returning programming languages.

Software Engineers:

Concern with the issues of

1. Analysis
2. Design
3. Verification &Testing
4. Documentations
5. Software maintenance
6. Project Mangement

Computer Software:

It includes the Source code and all the associated documents and documentation that constitute the software product.

Components of Software products

Documentation:

1. Requirements Documents
2. Design specification
3. Source code
4. Test Plans
5. Principles of Operation
6. Quality assurance procedure
7. Software problem reports
8. maintenance procedures
9. Users manual
10. Installation Instruction
11. Training Aids.

♉ Explains the characteristic of the documents.
♉ Internal Documentation of source code - it describes the characteristics of the code.
♉ External Documentation --- it explains the characteristic of the documentation associated with code.

Developer:

Developer or Software Engineer are used interchangeable.

Customer:

An individual or organization.

Software Reliability:

The ability of a program to perform a required function and their stated conditions for a stated period of time.

Some Size Factors:

Total effort devoted to software:

% of cost
150
100
50
0

H / w S / W c o s t R a t i o

In 1960 the ratio was approximately 80% Hardware cost and 20% Software cost.In 1980 the ratio was reversed. 80 % for software cost and 20% for Hardware cost.

The reason is the transistors,Intetpreter circuits have resulted in dramatic decreases in Hardware cost.

Distribution of Effort:

Life cycle of Software

1. Analyze & Design
2. Implement
3. Test
4. Adapt
5. Enhance
6. Fix

y e a r

The life span of software products is one to 3 years for development 5 to 15 years for maintenance.

Software maintenance involves 3 activities.

1.Enhancing the capability of the product.

2.Adapting the products to new processing environments

3.Correcting bugs.

Distribution of Maintenance:

5
1 0
1 5
2 0
2 5
3 0
3 5
4 0
0
A n a l y z e
I m p l e m e n
t

1.Enhancement—60%
2.Adaptation—20%

3.Correction ---20%

T e s t
A d a p t
E n h a n c e

Distribution of effort for development phase

1.Analysis & Design—40%
2.Implementation,Debugging,unit testing –20%
3.Integration & Acceptance testing—40%

From observations,

Software maintenance activities consumed more resources than software development activities.
A large percentage of total effort is devoted to software enhancement.
Testing requires half the effort during software development .
Activities of system testing,enhancement, adaptation consumed ¾ th of total life cycle effort

PROJECT SIZE CATEGORIES:

Project size is a major factor that determines the level of management control and the types of tools and techniques required on a software project.

F i x

Trivial Project:

No.of programmers :1
Duration :for a few days, few weeks
Product Size :500 source line
Packaged in :10 to 20 subroutines

These Programs are often personal software
Developed exclusively for the use of the programmer
Small amount of formal analysis,elaborate design documentation, extensive test planning or supporting documents are needed.

Small Project:

No.of programmers : 1
Duration : 1 to 6 months
Product Size : 1000 to 2000 source lines
Packaged in 25 to 50 subroutines
Small Programs usually have no interactions with other programs.
Example, Scientific applications written by engineers to solve numerical problems.
Students Projects written in compiler and Operating System.

Small Project requires little interactions between programmers and customers.
Standardized techniques and notations, standardized documents and systematic project reviews should be used in small projects.

Medium Size Projects:

No.of Programmers : 2 to 5
Duration : 1 to 2 years
Product Size : 10,000 to 15,000 source lines
Packaged in 250 to 1000 routines.
Examples
Medium size projects includes assesmblers,Compilers,Small management information system,inventory system and Process control applications.
To develop such programs, interaction among programmers and Customers is required.
A certain degree of formality is required in planning, documents and project reviews.
Most application programs,System programs are developed in 2 years or less by 5.

Larger Size Projects:

○ No.of Programmers : 5 to 20
○ Duration : 2 to 3years
○ Product Size :50,000 to 1 lakhs source lines.
○ Packaged in several sub systems.
○ Large programs has significant interactions with other programs and sub systems.
○ Examples, Compilers,Small time sharing systems, database packages,graphic packages and real time control systems.
○ Communication among programmers,managers,customers are needed.
○ The larger projects requires more than 1 programming team.
○ Example, three teams of 5 persons each.
○ It involves more than 1 level of management.
○ Systamatic process standardized documents and formal reviews are essential.

Very Large Projects:

○ No.of Programmers : 100 to 1000
○ Duration : 4 to 5 years
○ Product Size : 1 million source lines
○ It consists of several major subsystem each of which forms a large system.
○ The subsystem have complex,interactions with one another and with other

separate we developed system.
○ Tele communication and multi tasking.
○ It includes large OS, large database system and military commandor and control system.

Extremely large Projects:

○ No.of Programmers : 2000 to 5000
○ Duration : 5 to 10 years
○ Product Size : 1 million to 10 million source lines
○ It consist of several very large subsystem.
○ It involves real time processing,tele communications,multi tasking and distributed processing.
○ Example,Air traffic control,military commandor control system.

Quality and Productivity Factors:

 Development and maintenance of software product are complex task.
 There is a fundamental difference between writing a small programs for PC and developing or modifying a software product.
 Software Quality and programmer productivity can be improved by improving the process used to develop and maintain software products.

Individual Ability:

 Production and maintenance of software products are labour intensive activities.
 Productivity and Quality are direct functions of individual ability and effort.
 There are two aspects of ability

♉ General competition of the individual
♉ Familiarity of the individual with the particular application area.

♉ Lack of familiarity with the application area can research in low productivity and poor quality.
♉ On very large and extremely large projects no of programmers so large.
♉ The individual differences in programmer productivity will tend to average out.
♉ Modules developed by weaker programmers may show poor quality and may lag in delivery time.
♉ Small and medium size projects (5-fewer programmers)are extremely sensitive to the ability of the individual programmer.
♉ Individual ability is a primary factor in quality and productivity.

Team communication:

- Programming has regarded as an individual and private activity.
- Programmers are rarely preced as public documents and they rarely discuss the exact details of the work in a systematic manner.
- So as a result ,the programmers may mis understand the role of their modules in an evolving system.
- This mades mistake that may not be detected until some time later. Many of the recent innovations in software Engineering such as design,reviews and code reading exercise have the goals of making software more visible and improving communications among programmers.
- Increasing product size results in decreasing programmer productivity due to the increased complexity of interactions among program components.
- Due to this increased communication is required among programmers,managers and customers.

From Brooks observation:

No of communication path among programmers = n(n-1)/2
Where,n=no of programmers.

Increasing the number of team members from 3 to 4 to 5 increases the no of communication path from 3 to 6 to 10.

Brooks law:

"Adding more programmers to a late project may make it later"

Product complexity:

There are 3 levels of product complexity.

1. Application Programs
2. Utility Programs.
3. System level Programs.

Application Program

It includes scientific and data processing routines written in a high level language such as COBOL,FORTRAN,C,C++.

Utility Program

It includes compilers,Assemblers,linkage Editors and loaders. They may be written in high level language or Assembly language.

System Level Programs:

It includes data communication packages real time process control system, OS routines in any kanguages.(i.e)high level or assembly.

- Application programs have the highest productivity and the system programs the lowest productivity.
- Utility programs can be produced at a rate of 5-10 times of system programs.
- Application programs at a rate of 25-100 times of system programs.
- A product that is twice as large or twice as complex as a known product,by whatever measure other than effort may require 10 times or even 100 times the amount of effort required for the known product.

Appropriate Notations:

- In software engineer the representation schemes have fundamental importance, programming languages provides compact notations for the implementation phase of software development.
- But there are no widely accepted notations for stating functional requirements ,design specifications,test plans are performance criteria.
- There are no universely accepted notation in software Engineering.
- Appropriate notations provide vehicles of communication among project personnel.
- It introduces the possibility of using automated software tools to manipulate the notations and verify proper usage.

Systamatic Approaches:

Change Control:

- In every field there are certain accepted procedures and techniques
- A Single approach to software development and maintenance will not be adequate to cover all situations.
- In the evaluation of software engineering it is not clear which of the various approaches to software development should be used in which situation.
- The flexibility of software is a great strength and also a great source of difficulty in software engineering.
- Requirements can also change due to poor understanding of the problem are external economic and political factors beyond the control of the customers or developers.
- Notations and procedures provide the ability to trace and access the impact of proposed changes are necessary to make visible the true cost of apparently small changes to source code.
- Use of appropriate notations and techniques makes control change possible without degrading the quality of work products.
- Planning for software project must include plans for change control.

Level of Technology:

It include factors such as programming language.

Machine Environment

The Programming Practices.

Software tools

Modern Programming languages provide improved facilities for data definition and data usage.

Improve Constructs for specifying control flow,better modularization facilities, user defined exception Handling and facilities for concurrent programming.

The machine environment includes a set of hardware and software facilities for developing, using and maintaining a software product.

Modern programming practices include use of systematic Analysis and design techniques,notations,structure coding,systematic techniques for designing and documenting and testing.

Level of Reliability:

Every software product must possess basic level of reliability.

Extreme reliability is gained only with great care in analysis,design,design implementation,system testing and maintenance of software product.

Both human and machine resources are required to obtained increased reliability.

Problem Understanding:

♉ Failures to understand the true nature of the problem to be solved is a common and difficult issue.

Available Time:

♉ Often the customer does not truly understand nature of the problem.

♉ Often the software engineering does not understand the application area and has trouble communicating with the customer because of differences in educational backgrounds view the points and technology.

♉ Careful planning customer interviews,task observations and prototyping, a preliminary version of the user's manual and precise product specification can increase both customer and developer understanding of the problem to be solved.

A software project requiring 6 programmer-months of effort can be completed by 1 programmer in 6 months or by 6 programmers in 1 month.

Software projects are sensitive not only to total effort but also to ellapse time and the no.of people involved.

Utilising 6 programmers for 1 month will be less effective than using 1 programmer for 6 months.

This is because the learning curve for 6 programmers on an 1 month schedule will occupy a large percentage of the elapsed time and because the effort required for co-ordination and communication among 6 programmers.

Programmer Productivity is also sensitive to the calendar time available for project completion.

Determining optimum staff in levels and proper elapsed times for various activities in software product development is an important and difficult aspects of cost and resource estimation.

Required Skill:

○ Software Engineering requires a vast range of skills.
○ Good Communications
○ Knowledge of application area
○ Requirement definition and design
○ Problem solving skills
○ Implementation of software (i.e)Good programming knowledge,no syntax error
○ Debugging and test plans
○ Inter personnel communication skill.

Facilities and Resources:

 Work related factors such as
 Good machine access and quiet place to work are more important.
 Software project managers must be effective in dealing with the

Adequacy of Training:

factors that motivate the programmers to maintain high product quality,high programmer productivity and high job satisfaction.
 Express oneself clearly in English
 Develop and Validate software requirements and design specifications.
 Work within application area
 Perform software maintenance
 Perform economic analysis.
 Work with project management techniques
 Work in groups

Management Skills:

○ Many of the problems in software project management are unique.
○ Managers experienced in management of computer hardware projects find software project management to be difficult.
○ This is due to the differences in design methods,notations and development tools.
○ Many Organisations offer project management training to software engineers to prepare them for project management task.

Appropriate Goals:

Primary Goal of software engineering is to development of software products for their intended use.

Every software product must provide optimal level of

1. Generality
2. Reliability
3. Efficiency

Raising Expectations:

There are two interrelated aspects of raising expectations

1.How much functionality,reliability and performance can be provided by a given amount of development effort.

2.Issues of fundamental limitations of software technology.

Managerial Issues:

 Success of Software project involves

Technical Activities
Managerial Activities

 Managers Control

1.Resources
2.Environment

 Important managers responsibility

Software product delivered on time
Software working according to customer’s wish
Software within cost estimates

Other managerial responsibility:

Business Plans
Recruiting customers
Developing Marketing Strategies
Recruiting and training employees

Important Problems:

 Planning is poor
 Selection for project managers are poor (i.e) Procedures and Techniques
 Description of project is poor
 Estimation of resources for software project is poor
 Success criteria is inappropriate
 Decisions rules are poor(for selecting the proper organizational structure,correct management techniques)
 Procedures ,methods and techniques are not readily available.

Methods for solving these peoblems:

 Educate and Train
Top management
Project
Software developers

 Analyze the data from previous software project to find effective methods
 Define objectives,quality
 Establish success priority criteria
 Develop accurate cost and schedult that are accepted by management and customer
 Selection of project managers
 Specific work assignments to software developers

Planning a software product:

 Goals can be formulated using concise statement,constraints.
 Goal apply to both development process and work product.
 It can be either qualitative or quantitative.
 Every development process
 Should provide product on time.
 Within cost estimates .
 Opportunities for project personnel to learn new skill.

Requirements includes:

 Functional requirements

 Performance
 Requirements for hardware,software and firmware.

Qualified requirements

 Response to external interrupts shall we 25 second maximum
 50KB of primary memory.
 Full operation 95% of each 24 hour period.

Quanlitative requirements

 Accuracy
 Efficient use of primary memory.
 95% relaiable

planning the development process:

1. software life cycle activities
2. define
3. develop
4. test
5. deliver
6. operate
7. maintain.

 lifecycle activities are given above. These activities are change
 no single life cycle models is used.
 Different models are used for various software product.
 A life cycle model that is understood and accepted by all concerned

parties improves project communications and project manageability , resource allocations, cost control and product quality.

The phased lifecycle model:

 Series of successive activities.
 Requires well defined input, process and results in well defined output.
 Resources is required to complete each phase.
 Application of explicit methods, tools and techniques.

Analysis consist of two sub phases

 Planning
 Requirements definition.

This phase includes

 Understanding the customer problem.
 Performing a feasibility study.
 Developing solution stragedy
 Acceptance criteria
 Planning the development process.

The products of planning are

 System definitions.
 Project plan.

System definitions:

 Expressed in English or some other language.

 It includes charts, figures, graphs, tables, and equations.

Project plan:

 Contains lifecycle model to be used.
 Organitational structure.
 Basic development schedule, resource estimate, staffing requirements, tools and techniques to be used.
 Time and cost are basically calculated because it is not possible to estimate exactly without doing basic design.

Requirements definitions:

 It includes basic functions of software components in hardware, software, and people subsystem.

The product of requirements definition:

 The product of requirements definition is a specification that describes
 The processing environment
 The required software functions.
 Performance constraints on the software.
 Exception handling
 Acceptance criteria.

Design phase:

 In the phased model, software design follows analysis
 Design phase identified software components
1. Functions.
2. Data streams
3. Data stores
 It specifies relationship among components.
 It specifies software structures.
 Maintaines a record of design decision.
 Blueprint for the implementation phase.
 Design phase consist of
1. Architectural design
2. Detailed design

Architectural design:

It involves identifying the software components dividing them into software modules and conceptual data structures, specifying interconnection among components.

Detailed design

It is concerned with the details of “how to”

- Package the processing modules.
- Implement the processing, algorithm, data structures and interconnection among modules.

Implemention phase:

It involves translation of design specification into source code and debugging, documentation and unit testing of source code.

Errors:

Implementation phase may include errors in routines, functions, logical errors, and algorithm, errors in data structure layout.

System testing:

It involves 2 kinds of activities

1. Integration testing
2. Acceptance testing

Integration testing:

Developing a stratedy for intergrating the software components into a function requires careful planning so that modules are available for integration when needed.

Acceptance testing:

- It involves planning an execution of various type of test that software system satisfied requirements documents.
- After getting the acceptance from the customer software system of released for production work and mainteance phase.

Mainteance Phase:

It Includes

- the enchancement of capabilities.
- Adaptation of software to new processing environment.
- correction of software bugs.

Milestones, documents and reviews:

 Another view of the software lifecycle g softwareive importance to the milestones, documents and reviews.
 Ask the software products evolves through the development phase it is difficult for themanager and team members to determine resources extended to predict schedule delays extra.
 Establishing milestones, reviews points, documents and management sign offs can improve project visibility.
 The development process becomes more public activity and tangible.
 This result is improved
 Product quality
 Increased programmer productivity.
 Better moralae among team members.

A system definition and project plan:

Product fesability review(PFR)

 PFR is held to determine the feasibility to project continuation.
 The outcome of review may be
 Termination of the project.
 Redirection of the proect.
 Or continuation of the project as planned.

A primilarly version of the user's manual is prepared:

 It involves a vehicle of communication between customer and developer.

 It is prepared using information from the system definition and result of prototype studies and mock ups of user displays and reports.

A software requirements specification is prepared:

 It defines each essential requirements for software Product.
 External interface to software ,hardware, firm ware, people subsystem.
 Each requirements should be define show that it can be verified by a methods such as
 Inspection
 Demonstration
 Analysis or testing

A primilarly version of the software verification of the plan is prepared

 It states the methods to be used
 Results to be obtained.

A software requirements reviews(SRR)

Is held to make sure the adequacy of
1. system definition
2. project plan
3. software requirements specification
4. software verification plan
5. preliminary user’s manual.

Software design specification:

The design team creates this specification in two stages
1. Architectural design document is created.
2. Following that the preliminary design review is held then the detailed design specification is generated.

A preliminary design review(PDR)

- Is held to evaluate of adequacy of the architectural design insatisfying the SPS(software Product specification)
- Another reviews may be required to resolve problems under format sign offs is required of the project manager.

Critical design review:

Is held
- CDR is used determine the acceptablility of the software design specification.
- A format sign offs is required.

During the design phase , the software verification plan is expanned to include method:

- To verify that the design is complete and consistent with respect to the requirements.
- To verify that the source code is complete and consistent with respect to the requirements and design specification.

A software verification review is held to evaluate the adequacy and completeness   of the verification plan:

- To review the primilinary acceptance test plan(ATP)
- ATP includes
- Actual test cases
- Expected result.
- Capabilities to be demonstrated by each test.
- The acceptance plan is initiated during the design phase and completed during the implementation.

During the implementation phased:

- Source code is written.
- Debug.
- Unit tested.
- Standard parcties in the following area
- Logical structure
- Coading style
- Data layout
- Comments
- Debugging
- Unit testing

Source code reviews are held during implementation:

- This is to ensure that all the code has been reviewed by atleast one person other than programmer.

- Inspection are conducted during product evaluation to verify the completeness, consistency and suitability of the work products.

- The users manual the installations and training plans and the software maintenance plans are completed during the implementation phase.
- A final acceptance review is performed prior to product delivery'.
- Software verification summary is prepared.
- It describes the results of all the reviews, audits,inspection and test throughout the development cycle.

A project legancy is written:

- The legancy summarises the project and provides a record of what went well and what went wrong during the project.

The cost model:

- This model is used specify the cost of performing various activities in a Software project.
- The cost of conducting a Software project is the sum of the cost involved in conducting each phase of the project.

The cost involved eac phase include:

- The cost of performing the process
- Preparing the products of the phase.
- Plus the cost of verifying the product of the present phase are complete and

consistent with the previous phase.

 Cost of producing system definition and project plan =performing planning functions and preparing documents+ cost of verifying the system definition and project plan.
 Cost of SRS= Cost of requirements definition and document + Cost of modifying system definition and project plan + Cost of verifying SRS is complete and consistence.

 Cost of design= Cost of preparing design specification and test plan+ Cost of modifying and correcting the system definition, project, SRS(Software requirement specification)+cost of verifying design

 Cost of product implementation= Cost of implementing documenting, debugging and unit tesing of source code+ Cost of users manual, verification plan, maintenance procedure, instalization and tranning instructions+ Cost of modifying and correcting system definition, project plan,SRS, design specification, verification plan+the Cost of verifying the implementation is complete and consistent.

 Cost of system test= Cost of planning and conducting the test+ Cost of modifying and correcting the source code+ Cost of verifying the test.

 Cost of maintenance Software= Cost of performing product enhancement +making adaptation to new processing requirements and fixing bugs.

The prototype lifecycle model:

 Importance to the sources of product request , go/no go decisions points and the use of the prototypes.
 Prototype is a mock up or model of the Software product.
 A prototype incorporates components of the actual model.
 There are several reasons for developing a prototype.

Important reason:

 It illustrates input data formats, messages, reports and interactive dialogues for the customer.
 To explore technical problems in the proposed system.
 In situations where phased model of analysis, design, implementation is not appropriate.

Successive version:

 Product development by the mothod of successive versions is an   extension of

prototyping.
 In which an initial products skeleton is refined in to increasing the level of capabilities
 It illustrates the analysis phase followed by interactive design, implementation and assessment of successive version.
 The dashed line indicates that the assessment of version I may indicate the need for the further analysis before designing version I+1.
 Version I is the prototype version of the software product.
 Versions one through N of the product or designed prior to any implementation activities.
 The dashed line indicates that implementation of the Ith version may reveal the need for further analysis snd design before proceeding with implementation of version I+1

Planning an organizational structure:

 Contains various task
 The task include
1. Planning
2. Product development
3. Services
4. Publications
5. Quality assurance
6. Support and maintenance

Planning task identifiers:

♉ External cutomers
♉ Internal product needes
♉ Conducts feasibility study.

Development Task Identifiers:

♉ design
♉ implements
♉ debuggs
♉
test and integrate the product

service task provides:

- automated tools and computer resources for all other task.
- Performs configuration.
- Product distribution

Publication task develops:

- Users manual
- Installization instruction
- Principles of operation
- Supporting documents

Quality assurance task provides:

- Independent evaluvation of source code.
- Publications prior to releasing them to customer.

Support task:

- Promotes the product.
- Trainers user.
- Installs the product.

Maintenance task provides:

- Error connection
- Enhancement

Methods for organizing these task include:

1. Project format
2. Functional format
3. Matrix format

Project structures

Project format

- It involes assuming a team of programmers.
- Project team members do

1. Product definition
2. Design the product
3. Implement it

4. Test it
5. Conducts Project review
6. Preparing supporting document.

Functional format:

 In this approach a different team of programmers perform each phase of the Project
 The work products pass from team to team as they evolved
 Functional format involves 3 teams
1. An analysis team.
2. A design team and implementation team.
3. test formatting and maintenance team.

Matrix format

 In this format eac of the functions has its own management team.
 This format involves a group of specialist personnel concerned only with that function.
 Each development project has a project manager concerned only with that Project
 The Project manager generates and reviews documents.
 Each functional group participate in each Project
 Ex: software development team members belongs to the development function similarly testing belong the testing function.

Programming team structure:

 Every programming team must have an internal structure.
 Team structure depends on the nature of the Project and the product
 Basic team structure includes
 Demacratic team
 All team members participate in all decisions.

The chief programmer team:

 chief programmer is assited and supported by other team members.
 Ex: doctors, surgeon

Hierarchical team:

 In combines the aspects of the democratic team and chief programmer team.
 Each team should be limited to not more than 5 or 17 members for effective coordination and communication.

Democratic team

- this teams was first described as egoless team.
- Group leadership rotates from member to member based on the task to be performed and the differing abilities of the team members.
- A Democratic team differs from an egoless team is that one team members is designsted as team leader and occupies the position of first among equals.
- This is because a team fuctions best when one individual is responsible for coordinationg team activities and for making final decision.

Advantages:

- Opptunities for each team members to contribute to decision.
- To learn from one another
- Increased job satisfaction
- Non threatening work environment.

Disadvantages:

- Weeknening of individual and authority.

chief programmer teams:

- this teams are highly structured.
- the chief programmer

- design the product.
- Implement critical parts of the product
- Makes all the major technical decision.
- Work is allocated to the individual programmer by the chief programmers.
- A program librarian maintains program listing, design documents, test plans etc in a central location.
- The chief programmer is assited by an administrative program manager.

Advantages:

- Centralized decision making.
- Reduced communication paths.

Hierarchical team structure:

 This structure occupies a middle position between the extremes of Democratic teams and chief programmer teams.
 The Project needed assigns, task, attends, reviews,detects problem areas, balances the word load the participate in technical activities.
 This structure limits the number of communication paths in the Project

Disadvantages:

 The most technical competetant programmer tend to be promoted in to management positions.
 Promotion of the best programmer have the two negative effects.
 Losing a good programmer.
 Creating a poor manager.

Other planning activities:

 Planning for configuration management and quality assurance.

Configuration management:

 Modeof arrangement
 Concerned witj controlling changes in the work products.
 Accounting for the status of the work products
 Mainteaning the program support library

Quality assurance:

 Develops and monitors the Project standars.
 Performs audits.
 Develop and perfoms acceptance test.

During planning phase:

 The two activities are specified.
 Tools are identified an acquired.

During design phase:

 Requriments and design specification are performed.
 Adherence to project standard is monitor.

During implementation phase:

 Requirements, design specification and source code are perfomed .

During testing phase:

 Acceptance and preparation of test results are performed.

Planning for independent verification and validation:

 An independent organization may provide verification of work products for some critical software Project
 Veification makes sure that various work products are complete and consistence.
 An external organization may verify that the design specification are complete and cosistance.
 Source code is complete.
 Validation involves.
 Planning and execution of text cases.
 Independent verification and validation results in high quality software product.

Planning phase-dependent tools and technique:

 Automated tools,specialized notation and modern techniques are used to develop software requriments specification, architectural and detailed design and the source code.
 Management tools such as structures, charts, are used to track and control progress.

Other planning activities:

 It includes:
1. primilinary cost estimate.
2. primilinary development schedule
3. primilinary staffing levels.
4. primilinary estimates of the computing resources and personnel require to operate and maintain the system.

UNIT II

SOFTWARE COST ESTIMATION

INTRODUCTION

MAJOR FACTORS

- most difficult task in software engineering
- difficult to make estimate during planning phase
- series of cost estimation
- preliminary estimate is prepared during planning
- an improved estimate is presented at the software requirements review
- final estimate is prepares at the preliminary design view
- program ability
- Product complexity
- Product size
- Available time
- Required reliability
- Level of technology

PROGRAM ABILITY

- The goal was to determine relative influence of batch and time shared access on programmer's productivity
- Example: 12 experienced programmer's were each given two programming problems to solve some use batch facilities and some using time sharing

- Resulting differences in individual performance among the programmers were much greaterthan could be atributed to the relatively small effect of batch or time shared machine access

 On very large projects te differences in individual programmers ability will tend to average out

 But on projects involving 5 or fewer programmers, individual difference in ability can be significant

PRODUCT COMPLEXITY

There are three categories of software products

Application programs-include data processing and scientific programs

Utility programs-it include compilers,assemblers

System programs-it include operating system,dbms,real time system

Application programs are defveloped in environment provided by the language compilers such as fortran,pascal

Utility programs are written to provide user procesing environment

System programs interact directly with the hardware

Brook’s states that utility programs are three times as difficult to write as application programs

System programs are three times as difficult to write as utility programs

 Product complexity are 1-3-9 for application ,utility,system programs

 Boehm uses three levels of product complexity equations of total programmer month of effort pm is

provided in terms of the number of thousands of
delievered source instruction ,KDSI

HOUSE KEEPING CODE

 programmer cost for the software project=the effort in programmer mnth*cost per programmer month

in this terminology the three levels of product complexity are organic ,semidetached,embedded

organic-application,entity-semidetached,embeddedsystem

application program:pm=2.4*(KDSI)**1.05
utility programs:pm=3.2*(KDSI)**1.12
system programs:pm=3.6*(KDSI)**1.20

 example:for a development of a60,000 line application programs,utility programs and system programs the ratio of pm:1 to 1.7 to 2.8

 the development time for a program

    application program TDEV=2.5*(pm)**0.38
    utility programs TDEV=2.5*(pm)**0.35
    system programs TDEV=2.5*(pm)**0.32

 given the total programmer months for a project and the development time the average staffing level is obtained by

application
program:176.6pm/17.85mo=9.9programmers
utility program:294pm/18.3mo=16programmers
system programm:489.6pm/18.1mo=27programmers

 failures in estimating the numberof source instructions in a software product is to under estimate the amount of house keeping code require

 posuiton of the source code that handles input,output,interactive user communication,error checking and error handling

PRODUCT SIZE

 A large software product is more epensive to develop than a small one

- Boehm equation indicate that the rate of increase in required effort grows with number of source instruction at an exponential
- Using exponents of 0.91 and 1.83 results in estimatesof 1.88 and 3.5 more effort for a product that is twice as large ,and factors of 8.1 and 67.6 for products that are 10 times as large as known product

- These estimates differ by factors of 1.86(3.5/1.88)for products that are twice as large and 8.3(76.6/8.1) for products that are 10 times as large

Effort equation	Schedule equation	Reference
PM=5.2(KDSI)**0.91	TDEV=2.47(MM)**0.35	(WAL77)
PM=4.9(KDSI)**0.98	TDEV=3.04(MM)**0.36	(NEL78)
PM=1.5(KDSI)**1.02	TDEV=4.38(MM)**0.25	(FRE79)
PM=2.4(KDSI)**1.05	TDEV=2.50(MM)**0.38	(BOE81)
PM=3.0(KDSI)**1.12	TDEV=2.50(MM)**0.35	(BOE81)
PM=3.6(KDSI)**1.40	TDEV=2.50(MM)**0.32	(BOE81)
PM=1.0(KDSI)**1.50	-	(JON77)
PM=0.7(KDSI)**1.50	-	(HAL77)

- Depending on the exponent used we can easily be off by a factor of 2 in estimating effort for a product twice the size of a known product and by a factor of 10 for a product 10 times the size of known product,even if all other factors tat influence cost remain constant

AVAILABLE TIME

- Total project effort is sensitive to the calander time

available for project competitiom

- Software projects require more total effort,if development time is compressed or expanded from the optional time

- According to putnam, project effort is inversely proportional to the fourth power of the development time E=k/(TD**4)

□
This formula predicts zero effort for infinite development time

□
Putnam also states that the development schedule cannot be compressed below about 86%of the nominal schedule regardless of the number of people or resources utilized

□
Boehm states that “there is a limit beyond which a software project cannot reduce its schedule by buying more personnel and equipment “

REQUIRED LEVEL OF RELIABILITY

□ The ability of a program to perform a required function under stated conditions for a stated period of time
♉ Accuracy
♉ Robustness
♉ Completeness
♉ Consistency
□ These characteristics can be built in to a software product
□ There is a cost associated with different hases to ensure high reliability
□ Product failure may cause slightly inconvienience to the user
□ While failure of other products may incur high financial loss or risk to human life

Development effort multiliers for software reliability

Category	Effect of failure	Effort multiplier
Very low	Slight inconvinience	0.75
Low	Losses easily recovered	0.88
Nominal	Moderately difficult to recover loses	1.00
High	High financial loss	1.15
Very high	Resk to human life	1.40

LEVEL OF TECHNOLOGY

 In a software development required project is reflected by
1. programming language
2. abstract machine
3. programming practises
4. software tools used

 modern programming languages provides additional features to improve programmer productivity and software reliability

 these features include
1. strong type checking
2. data abstraction
3. separate computation
4. exception handling

 productivity will suffer if programmers must learn a new machine environment as part of the development process

 modern programming practises include the use of
a) systematic analysis and design technique
b) structure designed notations
c) inspection
d) structured coding
e) systematic testing
f) program development library

 software tools range from elementary tools such as assemblers compilers,interactive text editors and DBMs

SOFTWARE COST ESTIMATION TECHNIQUES

 software cost estimates are based on past perfomance

 cost estimates can be made either (i)top down (ii)bottom up

Top-down:focus on system level cost such as computer resources.personnel level required to develop the system

Bottom up:the cost to develop each module are subsystem. Then combined to arrive at an overall estimate

1)EXPERT JUDGEMENT

- Most widelly used cost estimation techniques(top down)
- Expert judgement relies on the experience background and business sense of one or more key people in an organisation mode. Eg: an expert might arrive at a cost estimate in a following manner.

i) To develop a process control system
ii) It is similar to one that was developed last yr in 10 months at a cost of one million dollar

iii) It was not a respectible profit
iv) The new system has same control functions as the previous but 25%more control activities
v) So the time and cost is increased by 25%
vi) The previous sstem developed was the same
vii) Same computer and controlling devices and many of the same people are available to develop the new sysem therefore 20% of the estimate is reduced

viii) Resume of the low level code from the previous reduces the time and cost estimates by 25%
ix) This results in estimation of eight lakhs $ and eight months.
x) Small margin of safety so eight lakhs 50,000$ and nine months development time
xi) Advantage: experience

2)DELPHI COST TECHNIQUES(ESTIMATION)

- This technique was developed at the rand corporation in 1948

□
This technique can be adapted to software estimation in the following manner

1. A co-ordinator was developed at the rand corporation in 1948

2.
estimators study the document and complete their estimates

3.
they ask questions to the co-ordinator but they wont discuss with one another

4. the co-ordinator prepares and distributes a summary of the estimators response

5. the estimators complete another estimate from the previous estimator

6.
the process is iterated for as many as required

Input
system
Read
module

3)WORK DOWN BREAK STRUCTURE

□ A bottom-up estimation tool

□
WBS is a hierarchical chart that accounts for the

PROCESS HIERARCHY:

product
Transform
subsystem
parser
Data
validator
Output
subsystem
mmm
Results
computer
indiviual parts of the system

- WBS chart can indicate either product hierarchy or process hierarchy
- It identifies the product components and indicates the manner in which the components are interconnected
- It identifies the work activity and relationship among those activities
- Using WBS cost are estimated by assigning cost to each individual component in the chart and summing the cost
- WBS are the most widely used cost estimation techniques

4)ALGORITHMIC COST MODEL

- Constructive cost model(COCOMO)
- Algorithmic cost estimators compute the estimated cost of software system as the some of the cos of the module this model is bottom up estimates
- The constructive cost model (COCOMO)is an algorithmic cost model described by boehm
- In COCOMO model the equation calculates the programmmar month and deelopment schedule are used for the program unit based on the number of deliver source instruction(DSI)

□ Effort multipliers are used to adjacent the estimate for product attribute,computer,customer,and project attribute

□ The effort multipliers examines the daa from 63 project and by using delphi technic

□ The COCOMO equation incorporates a number assumption .for eg. The organic mode application program equation applied in the following type of situation

(i) small to medium size project
(ii)familiar application area
(iii)stable
(iv)in house development effort
(v)effort multipliers are used to modify these assumption

□ It includes cost of dovumentation and reviews

□ It includes cost of program managers and program librarian

□ Software project estimated by COCOMO model include the following:
(i) careful definition and validation
and requirements is performend
by a small number of people
(ii) the requirements remains the
same throughput the project
(iii) definition and validation
techniques of n architecture
design is performed by a small
number of capable people
(iv) detailed design, coding and unit
testing are performed in similar
by a group of programmers
working ion a teams
(v) interface errors are mostly found
by unit testing and by inspection
(vi) documentation is performed as a
path of development process

Multiplier Range of values
Product attributes:
  Required reliability
  Database size
  Product complexity
Computer attributes:

Execution time constraint
Main storage constraint
Virtual machine volatility
Computer turnaround time
Personnel attributes:
Analyst capability
Programmer capability
Applications experience
Virtual machine experience
Programming language experience
Project attributes
Use of modern programming practises
Use of software tools
Required development schedule

0.75to1.40
0.94to1.16
0.70to1.65

1.00to1.66
1.00to1.56
0.87to1.30
0.87to1.15

1.46to0.71
1.42to0.70
1.29to0.82
1.21to0.90
1.14to0.95

1.24to0.82
1.24to0.83
1.23to1.10

STAFFING LEVEL ESTIMATION

- the number of personel required throughput a software

development project is not constant
- planning an analysis are performed by a small group of people
- architectural design by a larger or smaller group
- detailed design by a larger number of people
- implementation and system testing requires the largest number of people

□ in 1958 norden observed that research and development project follows a cycle of planning,design,prototype development and use wqit corresponding personnel utilization

RAYLEIGH EQUATION:

□ Any particular point on the rayleigh curve represents the number of fulltime equivalent personnel required at the instant in time
□ Rayleigh curve is specified by two parameters
(i)td-the time at which the curve reaches its maximujm value
(ii)k-the toal area under the curve(ie) the total effort required for the project
□ In 1976 putnam studied 50 army software life cycle using rayleign curve
□ From his observation ,rayleigh curve reaches its maximum value td,during system testing and product release for many software products
□ From Boehm observation:Rayleigh curve is an accurate estimator of personal requirements for the development cycle from architectural design through implementation and system testing
□ FSP=PM(0.15TDEV+0.7t)
-(0.15TDEV+0.7t)^2
(0.25(TDEV)^2 0.15(TDEV)^2

ESTIMATING SOFTWARE MAINTENANCE COST

□ Software maintenance cost requires 40 to 60%of the total life

cycle devoted to software product

□

In some cases it may be 90%

□ Maintenance activities include

1. enhancement to the product
2. adapting the product to new processing enviroinment
3.

correcting problems

4. distribution and maintenance activities includes

enhancement-60%, adaptation-20%,error correction20%

□ during planning phase of the software project the major concened about the maintenance are
(i) estimating the number of maintenance programmmers that will be needed

(ii) specifying the facilities required for
maintenance
 A widely used estimators of maintenance personnel is the number of source lines that can be maintained by an individual programmers

LIENTZ AND SWANION OBSERVATION

 Maintenance programmers in a business data processing installatons maintains 32K

 Full itme software personnel needed for software maintenance can be determined by dividing the estimated number of source instructions to be maintained by the estimated number of instruction that can be maintained by a maintenance programmer
 For example if a maintenance programmer can maintain 32KDSI 2 maintenance programmer are required to maintain 64KDSI ,FSPM=(64KDSI)/(32KDSI/FSP)=2FSPM
 Boehm suggest that maintainence effort can be estimated by use of an acivity ratio,which is the number of source instructions to be added and modified in any given time period divided by the total number of instructions
 Step:1
ACT=(DSI added+DSI modified)/DSI total)
 The activity ratio is then multiplied by the number of programmer months required for development in a given time period to determine the number of programmer months required for maintenance in the corresponding time period
 Step :2
PM=ACT*MMdev
 The enhancement is provided by an effort sdjustment factor to differentiate effort multipliers for maintenance
 It is different from the effort multipliers used for ,multipliers
 Step:3
PMm=ACT*EAF*Mmdev
 Heavy importance on reliability and the use of modem programming practises during development may reduce the amount of effort required for maintenance
 If less importance on reliability and programming practises during development will increase the difficulty of maintenance

Maintenance effort distribution (from LIE80)

Activity %effort
Enhancement

Improved efficiency
Improved docmentation
User enhancement
Adaptation
Input data,files
Hardware,opeating system
Correctin
Emergency fixes
Scheduled fixes
other

51.3
4.0
5.5
41.8
23.6
17.4
6.2
21.7
12.4
9.3
3.4

UNIT III
UNIT-3

SOFTWARE REQUIREMENTS DEFINITION

Introduction

The analysis phase of s/w development involves project planning and s/w requirement definitions.

The s/w requirement specification records the outcome of the s/w requirements definition activity.

SRS is a technical specification of requirements for the s/w products.

The SRS is based on the system definition.

The requirements specification will state that what of the s/w product without implying how.

4.1The S/W REQUIREMENTS SPECIFICATION

Format of an s/w requirements specification

Section1: Product overview and summary
Section2: Development, operating, and maintenance environments
Section3: External interfaces and data flow
Section4: Functional requirements
Section5: Performance requirements
Section6: Exception handling
Section7: Early subsets and implementation priorities
Section8: Foreseeable modifications and enhancements
Section9: Acceptance criteria
Section10: design Hints and Guidelines
Section11:cross-reference index
Section12: Glossary of terms

Section 1 and 2 present an overview of product features and the processing environments.

Section 3 includes

1. User displays
2. Report formats
3. Dataflow diagrams
4. Data dictionary

Dataflow diagram specify

1. Data sources
2. Data stores
3. Transformations to be performed on the data.
4. Flow of data

5. Data stores

Data store is a conceptual data structure that is logical characteristics of data or given importance on a dataflow diagrams.

Data stores and data sinks are depicted by shaded rectangles.
Transformations by ordinary rectangles.

Data stores by open ended rectangles
The arcs specify the dataflow.

Dataflow diagrams are not concerned with decision structure or algorithmic details like flowchart.

A data dictionary

Entries include the name of the data item and its attributes.

Section 4 of a SRS specifies the functional requirements for an s/w product.

It specifies the relationship among inputs, actions and outputs.

Section 5 specifies the performance requirements such as

1. Response time for various activities.
2. Processing time for various processes.
3. Memory constraints.

Section 6 specifies the exception handling.

It includes actions to be taken and the messages to be display in response to undesired situations or events.

Possible exceptions include

1. Temporary resource failure.
2. Permanent resource failure
3. Incorrect
4. Inconsistent
5. out of range input data and parameters.

Section 7 specifies early subset an implementation priorities for the system under development.

Section 8 specifies the modification and enhancement.

Section 9 specifies the acceptance criteria (various test).

Section 10 contains design hints and guide lines.

Section 11 specifies cross reference index.

A cross reference directory should be provided to index to find the specific paragraph numbers in SRS.

Section 12 provides definition of terms that are unfamiliar to the customers and the product developers.

Desirable properties

A requirement document should be

1. Correct
2. Complete
3. Consistent
4. Unambiguous
5. Functional
6. Verifiable
7. Easily changed.

An incorrect, an incomplete set of requirements can result in an s/w product that satisfies its requirements but doesn’t satisfy customer needs.

S/w requirements should be functional in nature that is they should describe what is required without implying how the system will meet its requirements.

Changes will occur and project success often depends on the ability to incorporate change without starting over.

4.2 FORMAL SPECIFICATION TECNIQUQES

Functional characteristics of an s/w product are one of the most important activities to be then during the requirement analysis.

The advantage of formal notation is concise and ambiguous.

They provide the basis for verification of the resulting s/w product.

Two nations are used to specify the functional characteristics.

1. Relational
2. State oriented

Relational notations

It is based on the concept of entities and attributes.

Entities are named elements in a system.

The names are chosen to denote the nature of the elements.
Eg: stack, queue

Attributes are specified by applying functions and relations to the named entities.

Attributes specify permitted operations on entities, relationships among entities and data flow between entities.

Relational notations include,

1. Implicit equations
2. Recurence relations
3. Algebric axioms
4. Regular expressions

State oriented notations include

1. Decision tables.
2. Even tables.
3. Transition tables.
4. Finite state mechanism.-

5. Petrinets.

4.2.1 RELATIONAL NOTATIONS

Implicit equations

It states the properties of a solution without stating a solution method.
E.g. Matrix inversion $M*M^1=I+-E$

Matrix inversion is specified such that matrix product of the original matrix M and the inverse of M, M yields the identity matrix I+_ the error matrix e (computation error).

Complete specification of matrix inversion must include items such as matrix size, type of data elements etc.…

Given a complete functional specification for matrix inversion, design involves specifying a data structure, an algorithm for computing the inverse.

Recurrence relations

It consists of on initial part called the basis and one or more recursive parts.

The recursive part describe the decide value of a function in terms of other value of the function

Eg.fibonaccies number,F1(0)=0,F1(1)=1,F1(n)=F1(N-1) belongs to F1(N*2) for all n>1.

Recurrence relation is easily transformed into recursive programs.

4.2.2 STATE ORIENTED NOTATIONS

Decision tables

It provides a mechanism for recording complex decision logic.

Decision table is segmented into four quadrants

1. Condition stub
It contains all of the conditions being examined.

2. Condition entry
It is used to combine conditions into decision rules.

3. Action stub
It describes the actions to be taken in response to a decision rules.

4. Action entry
It relates decision rules to actions.

Orders are approved if the credit limit is not exceeded or if the credit limit is exceeded but past experience is good or if a special arrangement has made otherwise the order is rejected.

The(y, n,-) entries in each column of the condition entry quadrant form a decision rule.

Ambiguous pairs of decision rules that specify identical actions are said to be redundant.

Those specifying different actions are contradictory.

Table 4.5

R3 and R4 are redundant rules.

R2 and R3, R2 and R4 are contradictory.

A decision table is complete if every possible set of conditions has a corresponding actions prescribed.

Table 4.6

EVENT TABLES

Its specify actions to be taken when event occur under different sets of conditions,

A two dimensional event table relates actions to two variables.

F (M, E) =A

M denotes the current set of operating conditions.
E is the event.

A is the actions to be taken.

E.g. if a   system is in startup mode (EU) and event E13 occur action A16 is to be taken f (SU, E13) =A16.

E.g. actions separated by semicolon .e.g. (A14, A32) it denotes A14 followed sequentially by A32.

TRANSITION TABLE

It is used to specify changes in the state of a system.
Given the correct state and the current condition the next state results.

F (si,sj)= sk

SK=next state

Cj=current condition

Sj=current state

Given current state S1 and current input b the system will go to state S0.
F (S1, b) =S0

In transition diagrams state becomes nodes. In a directed craft.
Transitions are represented as arcs between nodes.

FINITE STATE MECHANISM

Data flow diagrams, regular expressions, transition tables are combined in finite state mechanism.

The data flow diagram for a s/w system consisting of a set of process interconnected by data streams.

Data streams are specified using regular expressions.

Process can be described using transition table.

Processes split states in initial state so and wait for input D3.

In state S2 split writes zero or more D12 msgs to F7, then on receipt of the end of data marker De. Closes F7 and returns to state so to wait for next transmit ion.

Limitations

Finite state mechanism is not possible for complex system. Because it involves large no of states and many combination of input data.

PETRINETS

Petrinets were invented in the 1960s by Carl Petri at the university of Bonn, West Germany.

They provide a graphical representation technique and systematic and systematic methods.

It was invented to overcome the limitations finite state mechanism.
(e.g.) 1. Concurrent systems are designed to permit simultaneous execution of the s/w

Components called task or process on multiple processors.

Concurrent task must be synchronized to permit communication among task that operates at different execution rates, to prevent simultaneous updating of shared data and to prevent deadlock.

Deadlock occurs when all the task in a system are waiting for data or other resources that can also waiting on other task.

Fundamental problems of concurrency are synchronization, mutual exclusion and deadlocks.

A pertinent is represented as a misdirected graph

Two types of nodes are used in a petrinets called places and transition.

Places are marked by tokens.

Petrinets are characterized by an initial marking of places and a firing rule.

A firing rule has two aspects.

A transition is enabling if every input place has at least one token.

An enable a transition can fire.
When a transition fires each input place of that transition looses one token and each output place of that transition gains one token.

A marked petrinets is commonly defined as a quadruple -4 values in a system.

Consisting of a set of places as set of transition t, a set of arcs a and a marking m .C= (p, t, a, m).

4.3 LANGUAGES AND PROCESSORS FOR REQUIREMENTS SPECIFICATION

A no of special purpose language and processor have been developed.

It permits concise stmt and automated analysis of requirement specification for s/w.

Some specification languages are graphical in nature.

Some are textual imager.

Problem stmt language/problem stmt analyzer (PSL/PSA)

It was developed by proff.Daniel Teichrow at the University of Michigan.

The problem stmt analyzer is the PSL processor.

This model describes a system as a set of objects and each object have properties and each property have their own values.

Objects may be interconnected.

These connections are called relationship.

The objective of PSL

Is to permit expression of much of the information's that commonly appears in SRS.

In PSL system discretions can be divided into 8 major categories.

1. System input output flow
It deals with the interaction b/w the system and its environment.

2. System structure
It is concerned with the higher achy among objects in a system.

3. Data structure
It includes all the relationships that exist among the data used.
4. Data derivation
It specifies which data objects are involved in particular process in the system.

5. System size and volume
It is concerned with the size of the system and those factors that influence the volume of processing required.

6. System dynamics
It presents the manner in which the system behaves overtime.

7. System properties
It specifies the properties if the system object.

8. Project management
It specifies the project related information as well as product related information.

Problem statement analyzer

The problem statement analyzer is an automated analyzer for processing requirements stated in PSL.

PSA operates on a database of information collected from a PSL description.

PSA system provide reports in four categories

1. Database modification reports.
2. Reference reports.
3. Summary reports.
4. Analysis report.

1. Database modification reports

It list changes mode in the last report with warning msgs.

These reports provide a record of changes for error correction and recovery.

2. Reference reports

It includes name list reports.

It lists all the objects in the database with types and data lost change.

1. Formatted problem statement reports.
It shows properties and relationships for a particular object.

2. Dictionary report
It provides a data dictionary.

3. Summary report

It presents information's collected from several relationships.
1. Database summary report

It provides project management information's by listing the total no of objects of various types.

4. Structure report

It shows complete and partial higherarchiy.

1. External picture report
It describes data flow in graphical form.

5. Analysis report

1. Content comparison report
It compares the similarity of input and output.

2. Data processing interaction report
Used to detect gapes in information flow and unused data objects.

3. Processing chain report
It shows the dynamic behavior of the system.

PSL/PAS is a useful tool for documenting s/w requirements.

It changes the ways in which s/w is developed in the organization.

Some times the changes may be better and some times for the worst.

Tools don’t solve problems.

It is used to improve s/w quality and programmer productivity.

RSL (REQUIREMENT STATEMENT LANGUAGE)

It was developed by the TRW defense and space systems group.

It is used to permit concise and unambiguous specification of requirement for real time s/w systems.

RSL and REVS are components of s/w requirements engineering methodology.

RSL has for primitive concepts

1. Elements-which name objects.

2. Attributes
It describes the characteristics of elements.

3. Relationships
It describes the relation b/w elements.

4. Structure
Composed of notes and processing steps.
(e.g.) RSL element
“Data”
“Initial value” is an attribute of the element data.
“Input” specifies the relationship b/w a data item and a processing step.

Specifying requirements in this approach makes explicit the sequences of the processing steps required.

The processing step may be done by several different s/w components and an s/w component may incorporate several processing step.

Follows are specified in the RSL as requirement network

R-NETS have both graphical and textual representation.

The requirements engineering validation system operates on RSL stmts.

It consists of three major components
1. A translator for RSL
2. a centralized database, the abstract system semantic model.
3. A set of automated tools for processing information in ASSM.

(e.g.)Air defense system.

STRUCTURED ANALYSIS AND DESIGN TECHNIQUES

SADT implements a graphical language. And a set of methods and management guidelines for using the language.

The SADT language is called a language of structured analysis.

Each diagram is drawn on a single page.

On an actigram the nodes denote activities and the arcs specify data flow b/w activities.

Datagram’s re important for at least two reasons: to indicate all activities affected by a given data object, and to check the completeness and consistency of an SADT model by constructing data diagrams from a set of actigrams.

Fig a illustrates the formats of actigrams and datagram nodes. It is important to note that four distinct types of arcs can be attached to each node.

Arcs coming into the left side of a node carry inputs and arcs leaving the right side of a node convey outputs.

Arcs entering the top of a node convey control and arcs entering bottom specify mechanism the concept of input, output, control and mechanism bound the context of each node in an SA diagram.

Outputs provide input and control for other nodes.

In a datagram the input is the activity that creates the data object and the output is the activity that used the data object.

STRUCTURED SYSTEM ANALYSIS

Two similar several of SSA was described by Gane and Sarson and by Demarco.

SSA is used to traditional data processing environment.

SSA uses a graphical language to build models of systems.

SSA incorporates databases concept.

There are four basic features of SSA
1. Data flow diagrams
2. Data dictionaries
3. Procedure logic representation
4. Data store structuring techniques.

1.
Data flow diagrams

Open ended rectangle indicates resources
Labels on the arcs denote data items.

Shaded rectangles depicts source and since for data.
Remaining rectangles indicates processing steps.

2.
 Data dictionary
It is used defined and record data elements

3.
 Procedure logic representation
Decision tables ad structured English are used to specify algorithmic processing details.

Important features of SSA

Relational model is used to specify data flow and data stores.

Relations are composed from the fields of data records.

These fields are called a domain of the relation.

If the record has n fields then the relation is called n tuple.