SOFTWARE ENGINEERING CS304 NOTES

09 September 2021 16:08

- Subhajit Ghimire 1912160

* SYLLABUS:

1. Software Development Life Cycle
2. Requirement Analysis → SRS
3. Software Project Management → COCOMO
4. Software Design → Coupling, Cohesam, UML, DFD, Class Diagram
5. Coding and Testing
6. Maintenance
7. Quality Management, Reuse

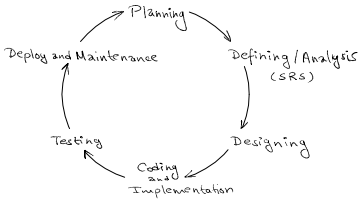
* Software Engineering Definition and Evolution:

- It is systematic, disciplined, cost-effective techniques for software development.
- Engineering Approach to develop a software.

Evolution:

- 1945-65 : Origin
- 1965-85 : Crisis
- 1990-2000 : Internet
- 2000-10 : Lightweight
- 2010-Now : AI, ML, DL

* SDLC : (Software Development Life Cycle)



* Classical Waterfall Model:

- Feasibility Model
  - Requirement Analysis and Specifications
    - Design
      - Coding and Unit Testing
        - System Testing and Integration
          - Maintenance

Pros:
- Base Model to other models
- Simple and Easy
- Small Projects

Cons:
- No feedback
- No Experiment
- No Parallelism
- High Risk
- 60% Effort in Maintenance

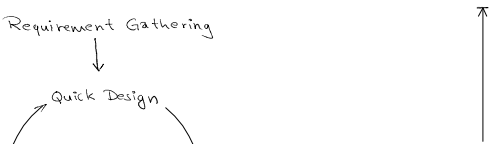
* Iterative Waterfall Model:

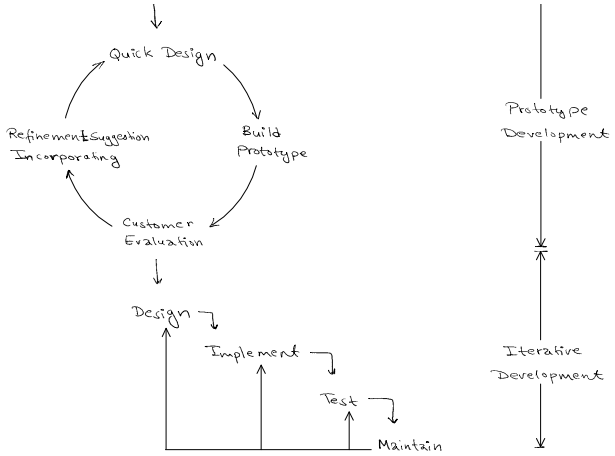
- Feasibility Study
  - Requirement Analysis and Specifications
    - Design
      - Coding and Unit Testing
        - System Testing and Integration
          - Maintenance

Pros:
- Base Model
- Simple and Easy
- Small Projects
- Feedbacks

Cons:
- No phase and overlapping
- No intermediate delivery
- Rigid (No changes)
- Less customer interaction

* Prototyping Model:

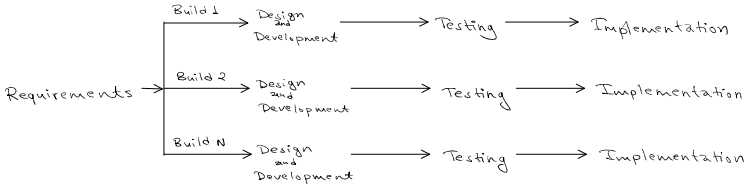




Pros
→ Good for technical and requirement risk.
→ If customer not clear with the idea, they'll be shown dummy models as help and feedback changes.
→ Throwaway model, hence saves effort if customer disapproves of the model.

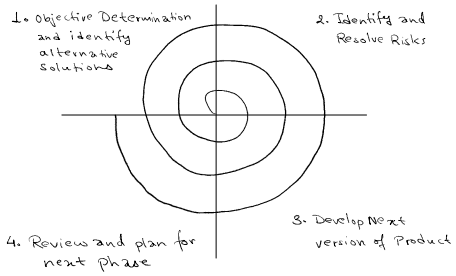
Cons:
→ Increase in Cost of Development

✳ Incremental Model :



PROs:
→ Module by Module Working
→ Maximum Customer Interaction
→ Feasible for Large Projects
→ Early Release Product Demand
→ Flexible to changes.

✳ Spiral Model :



→ Risk Handling
→ Radius of Spiral = Cost
→ Angular Dimension = Progress
→ Metamodel

PROs:
→ Risk Handling
→ Large Projects
→ Flexible
→ Customer Satisfaction

CONS:
→ Complex
→ Expensive
→ Too much Risk Analysis
→ Time

✳ Comparision of Various SDLC Models :

1. Classical Waterfall : Adv. → Basic
   Disadv. → Rigid, Inflexible, Not for Real Project.
2. Iterative Waterfall : Adv. → Basic, Problem is well understood, feedback
3. Prototype Model : Adv. → Reusability, User Requirement not clear, No Early Lock on Requirements, High User Involvement (Flexible)
   Disadv. → Costly.
4. Incremental Model : Adv. → Module by module delivery, Easy to test and debug.
5. Evolutionary Model : Adv. → Large Projects
6. RAD Model : Adv. → Reusability, User at all levels, Time and Cost Constraint.
   (Rapid Application Development)
7. Spiral Model : Adv. → Risk Analysis, No early lock on requirement, Less experience can work
   Disadv. → Not for small projects.

8. Agile Model : Adv → Flexible, Advanced, Parallelism, Process divided into sprints

* Project Estimation Techniques:
  - Estimation of various projects parameters is an important project planning activity.
  - Parameters of project that need to be estimated:
    1. Project size
    2. Effort required to complete the project.
    3. Project Duration
    4. Cost

    Accurate estimation of these parameters is important for resource planning and scheduling.
  - Classification of Estimation Techniques:
    1. Empirical Estimation Techniques
    2. Heuristic Estimation Techniques
    3. Analytical Estimation Techniques

- Empirical Estimation Technique:
  - Based on making an educated guess of the project parameters and commonsence
  - Based on prior experience of development of similar products and projects.
  - Educated guess based on past experience.
  - Two popular Empirical Estimation Techniques:
    1. Expert Judgement Technique :
       - → An expert makes an educated guess of the problem size after analysing the problem thoroughly
       - → The expert estimates the cost of the different components of the system like GUI, database module etc.
       - → Combines them to arrive at the overall estimate.
    2. Delphi Cost Estimation :
       - → Carried out by a team comprising of a group of experts and a coordinator.
       - → Coordinator provides each estimator with a copy of the SRS document and a form for recording his cost estimate.
       - → Estimators complete their individual estimates anonymously and submit to the coordinator.

- Heuristic Estimation Technique:
  - Relationship that exists among different project parameters can be modeled using suitable mathematical expressions
  - Once the independent parameters are known, the dependent parameters can be easily determined by substituting the values of the independent parameters in the corresponding mathematical expressions
  - Assume that the characteristics to be estimated can be expressed in terms of some mathematical expression.
  - Can be classified as:

    1. Single Variable Model :

    $$\text{Estimated Parameter} = c_1^{e^{d_1}}$$

    where,
    e is characteristics of the software which has already been estimated (independent variable)
    Estimated Parameter is the dependent parameter to be estimated like effort, project duration, staff size etc.
    c_1 and d_1 are constants and are usually determined using data collected from historical data.

Basic COCOMO model → The basic COCOMO model is an example of Single Variable Cost Estimation Model.

    2. Multivariable Cost Estimation Model :

    $$\text{Estimated Resource} = c_1 * e_1^{d_1} + c_2 * e_2^{d_2} + \ldots$$

    where,
    $e_1, e_2 \ldots$ are basic (independent) characteristics of the software already estimated.
    $c_1, c_2 \ldots, d_1, d_2 \ldots$ are constants.

Intermediate COCOMO model → The intermediate COCOMO model can be considered to be an example of Multivariable Estimation Model.

- Analytical Estimation Techniques :
  - → Derive the required results starting with certain basic assumptions regarding a project.
  - → These techniques have certain scientific basis.
  - → For example: Halstead's Software Science

  - Halstead's Software Science
    - → Used to derive results starting with a few simple assumptions.
    - → Analytical technique to measure size, development effort and development cost of software products.
    - → Useful for estimating software maintenance efforts.
    - → Used few primitive program parameters to develop expressions for overall program length, potential minimum value, actual volume, effort etc.

* COCOMO (Constructive Cost Model):
  - → Is a Heuristic Estimation Technique.
  - → This implies that size is primary factor for cost, other factors have lesser effect.
  - → Prescribes 3-stage process for project estimation.
    - → In first stage, initial estimate is obtained.
    - → Over the next two stages, initial estimate is refined to arrive at more accurate estimate.
  - → Attributes:
    - → Ranging in size from 2000 to 100000 lines of code (LOC).
    - → Programming languages (PLs) ranging from assembly to PLII.
    - → Based on waterfall model of software development.

- Bohem classified Software Development Project into three categories.

  1. Organic:
     - → Deals with developing a well understood application program.

↳ Size of development is reasonably small and experienced.
↳ Team members are experienced in developing similar kind of projects.

2. Semidetached:
↳ Dev team consists of both experienced and inexperienced staff.
↳ Team members have limited experience about some aspect but are totally unfamiliar with some aspects of the system being developed.
↳ Mixed Experience.

3. Embedded:
↳ Software being developed is strongly coupled to complex hardware.
↳ Software projects that must be developed within a set of tight software, hardware and operational constraints.

Mode	Project size	Nature of Project	Innovation	Deadline of the project	Development Environment
Organic	Typically 2-50 KLOC	Small size project, experienced developers in the familiar environment. For example, pay roll, inventory projects etc.	Little	Not tight	Familiar & in house
Semi detached	Typically 50-300 KLOC	Medium size project, Medium size team, Average previous experience on similar project. For example: Utility systems like compilers, database systems, editors etc.	Medium	Medium	Medium
Embedded	Typically over 300 KLOC	Large project, Real time systems, Complex interfaces, Very little previous experience. For example: ATMs, Air Traffic Control etc.	Significant	Tight	Complex Hardware/ customer interfaces required

- Acc. to Boehm (Dr. Barry Boehm proposed COCOMO in 1981), software cost estimation should be done through three stages:
  1. Basic COCOMO
  2. Intermediate COCOMO
  3. Complete COCOMO.

1. Basic COCOMO Model:

↳ Computes software development effort, time and cost as a function of program size.
↳ Program size is expressed in estimated thousands of Source Lines of Code (KLOC or SLOC)

$$\text{EFFORT} = a_1 * (\text{KLOC})^{a_2} \text{ PM}$$
$$T_{dev} = b_1 * (\text{EFFORT})^{b_2} \text{ Months}$$

where,
KLOC is estimated thousands of source lines of code.
a_1, a_2, b_1, b_2 are constants for each category of software products.
T_{dev} is estimated time to develop the software, in months.
PM (Person Months) is unit used to express total efforts required to develop the software product.

PROS
→ Good for quick, rough and early estimate of software cost.

CONS
→ Does not account for differences in hardware constraints, personnel quality and experience, use of modern tools and techniques, and so on.
→ Accuracy is limited because it does not consider certain factors for cost estimation of software.

2. Intermediate COCOMO Model:

↳ Computes software development effort as function of program size and a set of "cost drivers" that include subjective assessment of product, hardware, personnel and project attributes.
↳ Uses set of 15 cost drivers

Product Attributes:
→ Required Software Reliability
→ Size of Application Database
→ Complexity of the product.

Hardware Attributes
→ Run-time performance constraints.
→ Memory constraints.
→ Required turnabout time.
→ Volatility of Virtual Machine Env.

Personnel Attributes:
→ Analyst Capability
→ Software Engineering Capability
→ Applications experience
→ Programming Lang. Experience

Project Attributes:
→ Use of software tools
→ App. of S/W Engg methods
→ Required development schedule.

$$\text{EFFORT} = a_1 (\text{KLOC})^{b_1} * \text{EAF}$$

where,
EFFORT is in terms of person months (PM)
KLOC is Source Lines of Code in thousand.
EAF is effort adjustment factor
a_1 and b_1 are constants for various class of software projects.

PROS
→ Can be applied to entire software product for easy and rough cost estimation during early stage.
→ Can be applied at the software product component level for obtaining more accurate cost estimation.

CONS
→ Effort multipliers are not dependent on phases.
→ Product with many components is difficult to estimate.

3. Complete COCOMO Model:

↳ Proposed to overcome shortcomings and limitations of Basic and Intermediate COCOMO:

Limitations of Basic and Intermediate COCOMO
(i) Consider a software product as single homogeneous entity.
(ii) Most large systems are made up of several smaller sub-systems.
(iii) Some sub-systems considered as organic type, some as embedded etc.
(iv) for some sub-systems, reliability requirements may be high and so on.

↳ Incorporates all characteristics of intermediate version with an assessment of the cost driver's impact on each step (analysis, design etc.) of the software engineering process.
↳ Considers the differences in characteristics of all the subsystems and estimates the effort and development time as sum of the estimates for the individual sub systems.

- Considers the differences in characteristics of all the subsystems and estimates the effort and development time as sum of the estimates for the individual sub systems.
- Uses different effort multipliers for each cost driver attribute.
- Effort is calculated as function of program size and a set of cost drivers given according to each phase of software life cycle.
- Cost of each subsystem is estimated separately
- Costs of the sub-systems are added to obtain total cost.
- Reduces margin of error in the final estimate.

EXAMPLE OF COMPLETE COCOMO:

- **Example of complete COCOMO Model:**
- A Management Information System (MIS) for an organization having offices at several places across the country:
  - Database part (semi-detached)
  - Graphical User Interface (GUI) part (organic)
  - Communication part (embedded)
- Costs of the components are estimated separately:
  - summed up to give the overall cost of the system.