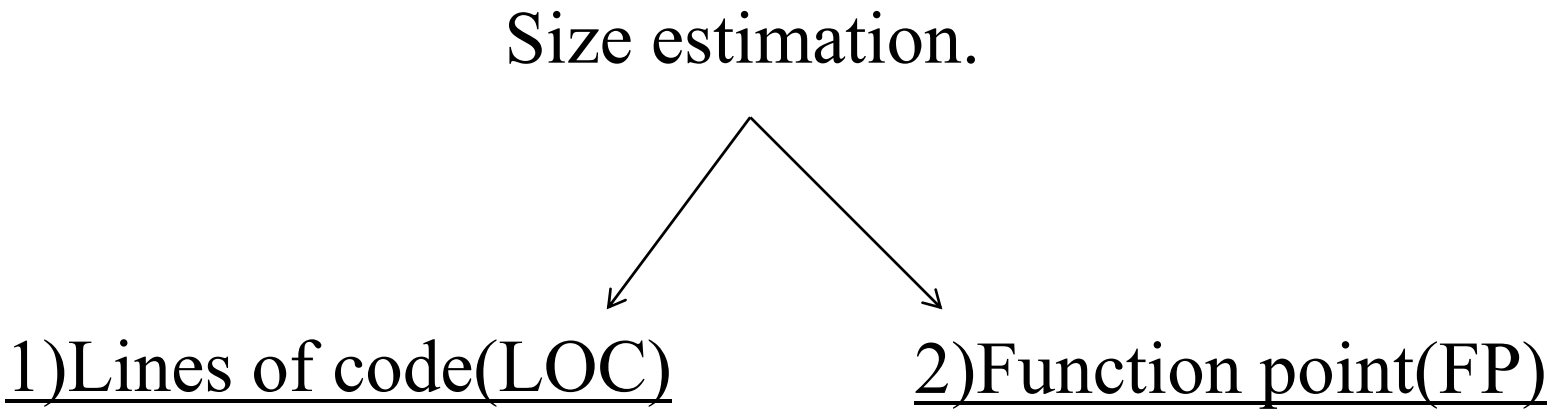
SOFTWARE ENGINEERING

Riya Roy
BCA

Metrics for project size estimation

- Project Size:-The size of executable code.



1) Lines of code(LOC)

- LOC :- Simplest among all metrics available.

  Most popular.
- Counting no.of source instruction in the developed program.(obviously comment lines, and header lines are ignored).
- LOC count at beginning of project is difficult. (Guess work by PM).
- LOC at end of project is simple.

Drawback of LOC :-

1. LOC measure coding activity alone.

   (Total effort needed to develop a prjt is proportional to coding effort.)

2. LOC depend on specific instruction.

   (Same problem diff LOC).

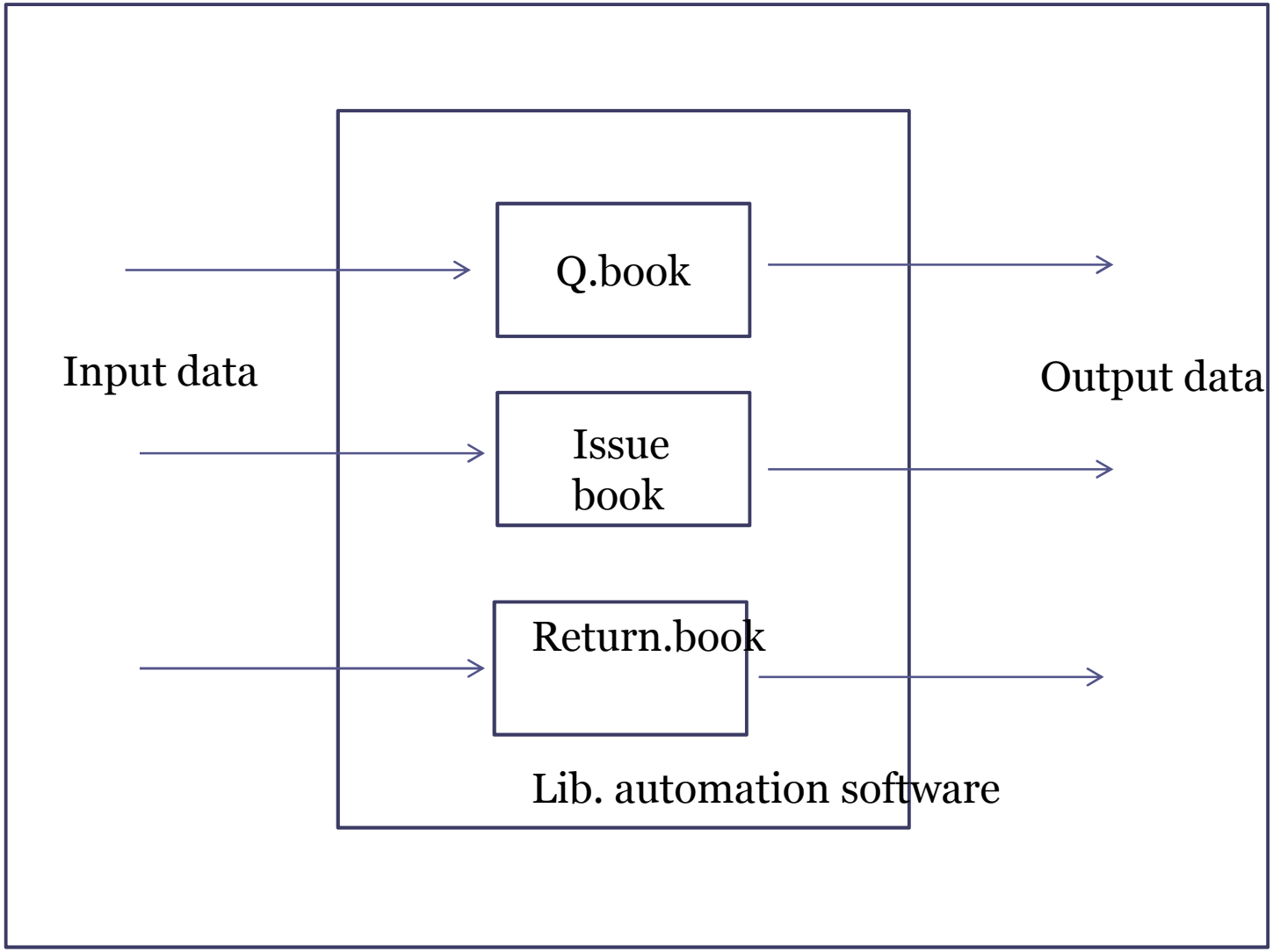
3. LOC measure correlates poorly with quality and efficiency of code.

4. LOC use higher level programming languages and code.
5. LOC measures lexical complexity and does not address logical and structural complexities.
6. Difficult estimate LOC of the final program.

2) Function point(FP)

- Overcome drawback of LOC, Most popularity.
- It can easily computed from prbm specification itself.
- Idea:- size of software product is directly dependent on the no.of diff. higher level ftn.

- Diff. features may take diff. amount of efforts to develop.(ex:- banking sftwr).
- Counting no.of i/t & o/t data items and no.of files accessed by the ftn.(fig:3.2 lib. automation stwr).



Mapping of i/t and o/t.

FP metric computation:-

Step 1: UFP (unadjusted ftn points)computation.

Step 2: Refine parameters.

Step 3: Refine UFP based on complexity of over the all project.

Step 1: UFP computation.

- UFP computed as weighted sum of 5 characteristics.
- UFP =(no.of i/t)*4 +(no.of o/t)*5 +(no.of inquiries)*4 +(no.of files)*10 +(no.of interfaces)*10

Step 2: Refine parameters.

- UFP need to be refined.
- UFP refined by taking into account the complexities of the parameters of UFP computation.
- The complexity of each parameters graded into 3 broad categories.(table 3.1)

Step 3: Refine UFP based on complexity of over all prjt.

- Several factors impact – over all prjt size.
- TCF(technical complexity factor) computed.
- FP = UFP * TCF.
- TCF expresses overall impact of the crspnd paramtr on development effort.

(ex:3.1)

PROJECT ESTIMATION TECHNIQUES

1. Empirical estimation techniques.
2. Heuristic techniques.
3. Analytical estimation techniques.

1. Empirical estimation techniques.

- Guess the project parameters.
- Based on common sense and subjective decision.
- Two popular techniques :-

  1. Expert judgement.
  2. Delphi cost estimation.

Expert judgement.

- Widely used estimation techniques.
- An expert make guess abut prbm size, after analysing prbm.
- Expert estimate cost of diff. components.
- Combines estimate for individual modules.

Drawback:-

- Human error.
- Expert may not have relevant experience.
- Lack of familiarity with particular subjects.

Delphi cost estimation

- Over come drawback of expert judgement.
- Team – group of experts and co-ordinator.
- Estimators submit the estimation to co-ordinator.
- Co-ordinator prepares summary of responses of all estimator
- Summary infmtn is distributed to estimators.
- Based on summary estimators re estimate.
- This process iterated for several rounds.

2. Heuristic techniques.

- Assume the relationships that exist among diff. project.
- It can be modelled using suitable mathematical expressions.
- Once the parameters known, other parameters easily determined.
- Substituting values of parameters in crpnd mathemathical expression.

3. Analytical estimation techniques.

- Required basic assumptions regarding project.
- Starting with assumptions.
- Useful for estimating software maintenance efforts.
- EX:- Halsteads software

COCOMO –heuristic estimation techniqu

- COCOMO:- COnstructive COst estimation MOdel.
- Proposed by boehm[1981].
- Three stage process for project estimation.
- 1ST STAGE- Initial estimate
- 2nd and 3rd STAGE- Refined - to more accurate estimation.

Three stages of COCOMO estimation techniques:-

1. Basic COCOMO.
2. Intermediate COCOMO.
3. Complete COCOMO.

Basic COCOMO.

Development complexity :-

1. Organic:-

- Project deal with developing **well understood application** program.
- **Size** of **development** team **small**.
- Team member **experienced**.

2. Semidetached:-

- Development team consists of **mixture of experienced and inexperienced staff**.
- Team members have **limited experience**.
- **Unfamiliar** with some of system being developed.

3 Embedded:-

- Team members **have limited experienced**.
- **Unfamiliar with some aspects** of system being developed

Person month(PM)

- Popular unite for **effort measurement**.
- **Developer** are typically assigned a project for certain **no.of months**.
- Personnel working on project usually increase or decrease by integral no.
- Sharp edges in plot.(fig 3.3)

COCOMO Expression

- Effort = a1 *$(KLOC)^{a2}$ PM
- Tdev = b1 * $(Effort)^{b2}$ months
- KLOC - (killo lines of code) - estimated size of software product.

- a1, a2, b1,b2 –constants.
- Tdev – estimated time to develop the software.
- Effort – total effort required to develop software product.
- Value of a1, a2, b1,b2 historical data, collected from large no.of actual projects.

Effort- size plot(fig 3.4)

- **Effort requi**red to develop a product **increase with project size**.
- **COCOMO assumes** project designed and developed by using **SE principles**.
- **Cost estimation** obtained by multiplying **estimated effort** by **manpower cost** per month.
- Overhead cost :- cost due to h/w, s/w for project, office space and electricity etc…

Intermediate COCOMO

- **Effort** to develop a product vary **depending** the **development environment**.
- This model **recognise the fact** and **refine the estimates.**
- Scaling up or down based the evaluation of set of software.
- Uses set of **15 cost drivers** to determined the **attribute**s of software development.

- Cost drivers – attributes of following items.
- Product:- complexity of product, requirements etc..
- Computer:- speed required, storage space etc..
- Personnel:- experience level, programming capability, analysis capability etc..
- Development environment:- development facilities available to developers..

Complete COCOMO

- Overcome drawback of other cocomo.
- Consider software product as **single homogenous entity**.
- The large system made up of – smaller subsystem – diff characteristics.
- **Estimate effort** and development as **sum of** estimates of the **individual sub system**.

COCOMO 2

- Application composition model.
- Early design model.
- Post- architecture model.

Application composition model.

- Based on counting no of screen, reports, and modules.
- Each **components** consider as **objects.**
- Following steps:-
- **Estimate** no of screen, reports, and modules from SRS.
- Determine **complexity level**.

- Use the **weight values**.(weight for crspnd effort)
- **Add all** assigned **complexity values** for the objects.
- **Estimates percentage** of reuse estimated.
- Determine **productivity.**
- Finally **estimated effort in PM** calculate.

Early design model.

- Effort calculated using:-
- Effort – KSLOC * IIi Cost driveri

Post- architecture model

- Effort calculated using:-
- Effort – a * KSLOC * IIi Cost driveri

Staffing level estimation

- Nordan's work
- Putnam's work
- Jensen's model

Norden's work

- Norden's studied staffing patterns of several **research and development (R&D)** projects.
- Conclusion – R&D project is vary diff from **manufacturing or sales.**
- **Change over time**.(fig 3.6)
- **Initial** stage man power is **low**, as **project progress** manpower requirement **increases**.(eq)

Putnam’swork

- Only **small no of developers** needed at **beginning** of project.
- As project progress, more detailed work is performed, no.of developers increase and reach peak during testing
- Putnam derived expression .(eq 3.2 & 3.3)

Jensen's model

- Similar to putnam model
- Jensen proposed eq:-

scheduling

- Scheduling is important project planning activity.

1. Work break down structure.
2. Activity networks
3. Critical path model
4. PERT chart
5. Gantt chart

Activity networks

- Activity on Node(AON)
- Activity on edge(fig 3.8)(t 3.7)

Critical path model

- Minimum time(MT)
- Earliest tome(ES)
- Earliest finish time(EF)
- Least start time(LST)
- Latest finish(LF)
- Slack time(ST)

PERT chart

- Project evaluation and review technique(PERT)
- Optimistic(O)
- Most likely estimate(M)
- Worst case(W)

Organisation and team structure

- Organisation handles several project at any time.
- Organization structure
- Team structure

Organization structure

- Functional format
- Project format
- Matrix format

Team structure

- Chief programmer team
- Democratic team
- Mixed control organisation

Staffing

- Good technical knowledge
- Good programming abilities.
- High motivation.
- Intelligence
- Discipline
- Good communication

Risk management

- Risk identification
- Risk assessment
- Risk mitigation

Risk identification

- Project risk
- Technical risk
- Business risk

Risk mitigation

- Avoid risk.
- Transfer the risk


Thank You !