# SS ZG622: Software Project Management (Lecture #6)

BITS Pilani
Pilani|Dubai|Goa|Hyderabad

T V Rao, BITS-Pilani Off-campus Centre, Hyderabad

# Text Books


**T1:** Bob Hughes, Mike Cotterel, and Rajib Mall, Software Project Management, 5th Edition, McGraw Hill, 2011

**T2:**Pressman, R.S. Software Engineering : A Practitioner's Approach, 7th Edition, McGraw Hill, 2010

**R1**: Sommerville, I., Software Engineering, Pearson Education, 9th Ed., 2010

**R2**: Capers Jones., Software Engineering Best Practices, TMH ©2010

**R3**: Software Project Effort Estimation: Foundations and Best Practice Guidelines for Success by Adam Trendowicz and Ross Jeffery Springer © 2014

**R4**: George Stepanek, Software Project Secrets : Why Software Projects Fail, Apress ©2012

**R5**: A Guide to the Project Management Body of Knowledge (PMBOK® Guide), Fifth Edition by Project Management Institute Project Management Institute © 2013

**R6**: Robert K. Wysocki, Effective Software Project Management, John Wiley & Sons © 2006

innovate achieve lead

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

# L6: Sotware Project Estimation –

## Decomposition, Empirical, Agile approaches

**Source Courtesy**: Some of the contents of this PPT are sourced from materials provided by publishers of prescribed books

# What is a Project? (review)

- Project Defined
  - A complex, nonroutine, one-time effort limited by time, budget, resources, and performance specifications designed to meet customer needs.
- Major Characteristics of a Project
  - Has an established objective.
  - Has a defined life span with a beginning and an end.
  - Requires across-the-organizational participation.
  - Involves doing something never been done before.
  - Has specific time, cost, and performance requirements.

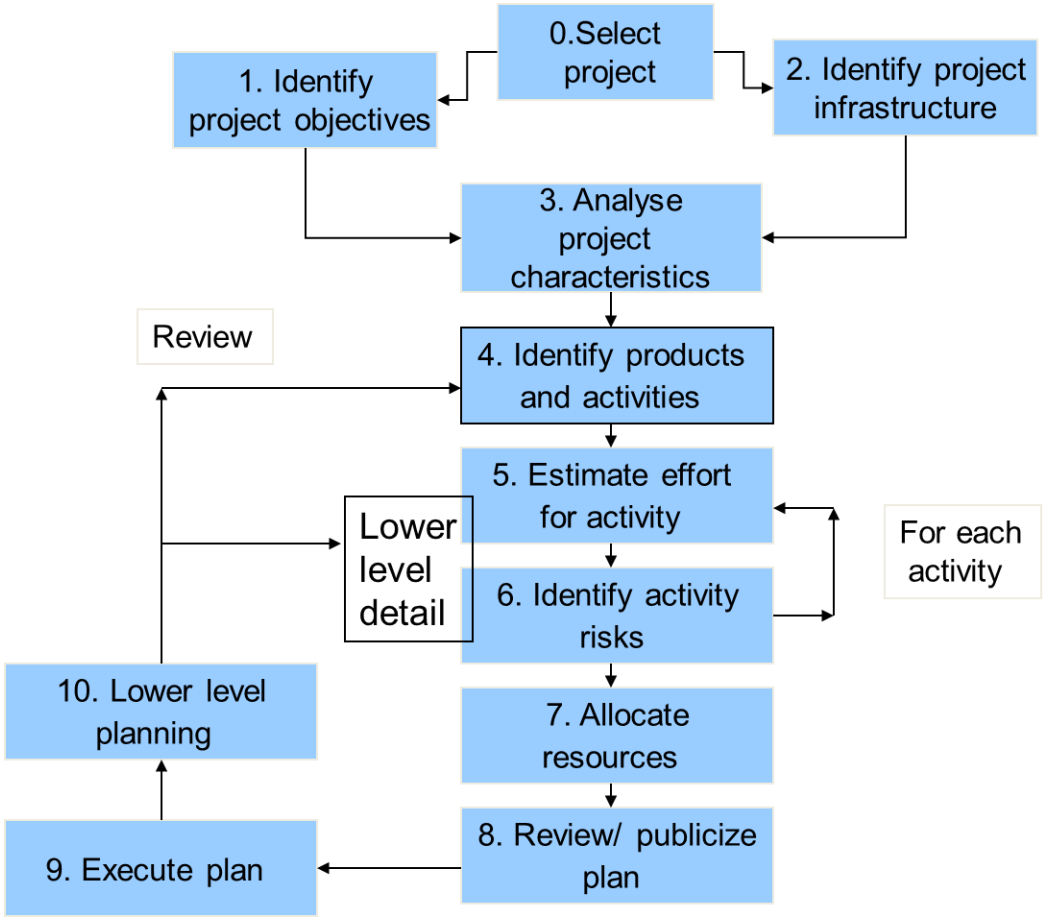
# Project Characteristics (review)

- Non-routine tasks
- Need for planning
- Objectives to be met
- Beneficiary
- Multi-disciplinary
- Ad-hoc team
- Phases
- Constraints
- Complexity

# Management Activities (review)

- **Planning** – deciding what is to be done
- Organizing – making arrangements
- Staffing – selecting the right people for the job
- Directing – giving instructions
- **Monitoring** – checking on progress
- **Controlling** – taking action to remedy hold-ups
- Innovating – coming up with solutions when problems emerge
- Representing – liaising with clients, users, developers and other stakeholders

# Step Wise approach to Planning Software Projects(review)



# Choice of process models(review)

- 'waterfall' also known as 'one-shot', 'once-through'
- incremental delivery
- evolutionary development
- 'agile methods' e.g. extreme programming

etc.

# Why do we measure ? (review)

- To characterize in order to
  - Gain an understanding of processes, products, resources, and environments
  - Establish baselines for comparisons with future assessments
- To evaluate in order to
  - Determine status with respect to plans
- To predict in order to
  - Gain understanding of relationships among processes and products
  - Build models of these relationships
- To improve in order to
  - Identify roadblocks, root causes, inefficiencies, and other opportunities for improving product quality and process performance

  – Park, Goethert, and Florac [Guidebook on Software Measurement]

- Can be applied to the software process with the intent of improving it on a continuous basis
- Can be used throughout a software project to assist in estimation, quality control, productivity assessment, and project control
- Can be used to help assess the quality of software work products and to assist in tactical decision making as a project proceeds

# Software Project Estimation

- Planning requires technical managers and the software team to make an estimation that determines how much money, effort, resources, and time it will take to build a specific system or product
- Process and project metrics can provide a historical perspective and valuable input for generation of quantitative estimates
- Past experience can aid greatly
- Estimation carries inherent risk, and this risk leads to uncertainty. Risk increases with
  - Project Complexity
  - Project Size
  - The degree of structural uncertainty
- The availability of dependable historical information has a strong influence on estimation risk
- A project manager should not become obsessive about estimation
  - Plans should be iterative and allow adjustments as time passes and more is made certain

"It is the mark of an instructed mind to rest satisfied with the degree of precision that the nature of the subject admits, and not to seek exactness when only an approximation of the truth is possible." ARISTOTLE

# Software Project Estimation (continued)

- Project scope must be understood
  - *Software scope* describes
    - the functions and features that are to be delivered to end-users
    - the data that are input and output
    - the "content" that is presented to users as a consequence of using the software
    - the performance, constraints, interfaces, and reliability that *bound* the system
- Elaboration (decomposition) is necessary
- At least two different techniques should be used
- Uncertainty is inherent in the process

# Estimation Techniques

- Bottom-up - activity based
- Parametric or algorithmic models e.g. function points
- Expert opinion - just guessing?
- Analogy - case-based, comparative
- Parkinson – available effort (estimates scope rather than effort)
- 'price to win'
  - Hughes et al.
- Past (similar) project experience
- Conventional estimation techniques
  - task breakdown and effort estimates
  - size (e.g., FP) estimates
- Empirical models
- Automated tools
  - Pressman

# Decomposition Techniques

## Task break-down, Bottom-up

# Introduction

- Before an estimate can be made and decomposition techniques applied, the planner must
  - Understand the scope of the software to be built
  - Generate an estimate of the software's size
- Then one of two approaches are used
  - Problem-based estimation
    - Based on either source lines of code or function point estimates
  - Process-based estimation
    - Based on the effort required to accomplish each task

# Approaches to Software Sizing

- Function point sizing
  - Develop estimates of the information domain characteristics
- Standard component sizing
  - Estimate the number of occurrences of each standard component
  - Use historical project data to determine the delivered LOC size per standard component
- Change sizing
  - Used when changes are being made to existing software
  - Estimate the number and type of modifications that must be accomplished
  - Types of modifications include reuse, adding code, changing code, and deleting code
  - An effort ratio is then used to estimate each type of change and the size of the change
- Fuzzy Logic sizing
  - Requires the planner to derive size from understanding of qualitative aspects
  - Putnam & Myers

# Problem-Based Estimation

1) Start with a bounded statement of scope

2) Decompose the software into problem functions that can each be estimated individually

3) Compute an LOC or FP value for each function

4) Derive cost or effort estimates by applying the LOC or FP values to your baseline productivity metrics (e.g., LOC/person-month or FP/person-month)

5) Combine function estimates to produce an overall estimate for the entire project

(More on next slide)

# Problem-Based Estimation (continued)

- In general, the LOC/pm and FP/pm metrics should be computed by project domain
  - Important factors are team size, application area, and complexity
- LOC and FP estimation differ in the level of detail required for decomposition with each value
  - For LOC, decomposition of functions is essential and should go into considerable detail (the more detail, the more accurate the estimate)
  - For FP, decomposition occurs for the five information domain characteristics and the 14 adjustment factors
    - External inputs, external outputs, external inquiries, internal logical files, external interface files

pm = person month

# Problem-Based Estimation (continued)

- For both approaches, the planner uses lessons learned to estimate an optimistic, most likely, and pessimistic size value for each function or count (for each information domain value)
- Then the expected size value S is computed as follows:

$$S = (S_{opt} + 4S_m + S_{pess})/6$$

- Historical LOC or FP data is then compared to S in order to cross-check it

# An Example

A software package to be developed for a computer-aided design application for mechanical components. The software is to execute on an engineering workstation and must interface with various computer graphics peripherals including a mouse, digitizer, high-resolution color display, and laserprinter

# Example: LOC Approach

| Function | Estimated LOC |
|---|---|
| user interface and control facilities (UICF) | 2,300 |
| two-dimensional geometric analysis (2DGA) | 5,300 |
| three-dimensional geometric analysis (3DGA) | 6,800 |
| database management (DBM) | 3,350 |
| computer graphics display facilities (CGDF) | 4,950 |
| peripheral control (PC) | 2,100 |
| design analysis modules (DAM) | 8,400 |
| *estimated lines of code* | 33,200 |

Average productivity for systems of this type = 620 LOC/pm.

Burdened labor rate =$8000 per month, the cost per line of code is approximately $13.

Based on the LOC estimate and the historical productivity data, the total estimated project cost is **$431,000** and the estimated effort is **54 person-months.**

# Example: FP Approach

| Information Domain Value | opt. | likely | pess. | est. count | weight | FP-count |
|---|---|---|---|---|---|---|
| number of inputs | 20 | 24 | 30 | 24 | 4 | 97 |
| number of outputs | 12 | 15 | 22 | 16 | 5 | 78 |
| number of inquiries | 16 | 22 | 28 | 22 | 5 | 88 |
| number of files | 4 | 4 | 5 | 4 | 10 | 42 |
| number of external interfaces | 2 | 2 | 3 | 2 | 7 | 15 |
| count-total | | | | | | 321 |

The estimated number of FP is derived:

$FP_{estimated}$ = count-total * [0.65 + 0.01 * Sum ($F_i$)]

Assuming Sum(Fi) for the System is 52, $FP_{estimated}$ = 375

organizational average productivity = 6.5 FP/pm.

burdened labor rate = $8000 per month, approximately $1230/FP.

Based on the FP estimate and the historical productivity data, total estimated project cost is **$461,000** and estimated effort is **58 person-months**.

Does the system require reliable backup and recovery? - 4
Are specialized data communications required to transfer information to or from the application? - 2
Are there distributed processing functions? - 0
Is performance critical? - 4
Will the system run in an existing, heavily utilized operational environment? - 3
Does the system require on-line data entry? - 4
Does the on-line data entry require the input transaction to be built over multiple screens or operations? - 5
Are the internal logical files updated on-line? - 3
Are the inputs, outputs, files, or inquiries complex? - 5
Is the internal processing complex? - 5
Is the code designed to be reusable?- 4
Are conversion and installation included in the design? - 3
Is the system designed for multiple installations in different organizations? - 5
Is the application designed to facilitate change and for ease of use by the user? - 5

# Process-Based Estimation

1) Identify the set of functions that the software needs to perform as obtained from the project scope

2) Identify the series of framework activities that need to be performed for each function

3) Estimate the effort (in person months) that will be required to accomplish each software process activity for each function

(More on next slide)

# Process-Based Estimation (continued)

4) Apply average labor rates (i.e., cost/unit effort) to the effort estimated for each process activity

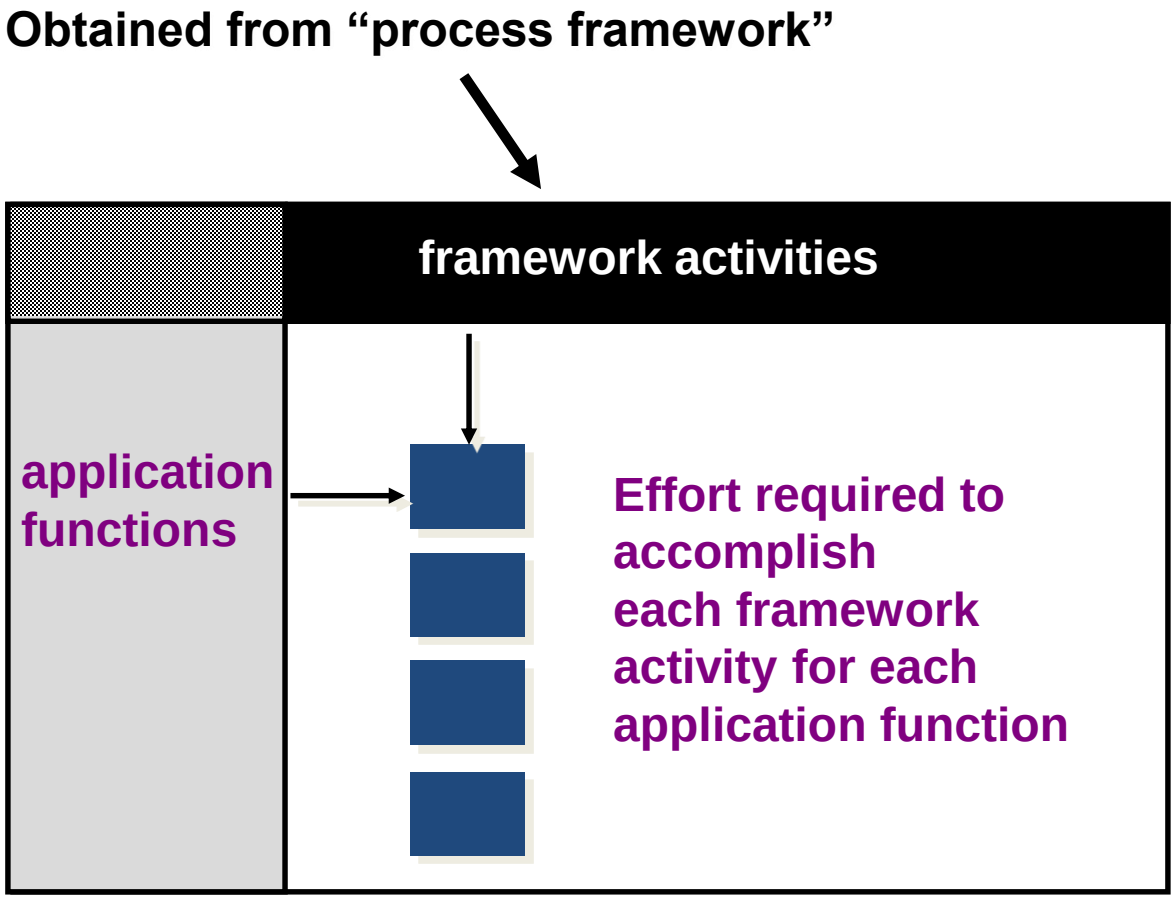
5) Compute the total cost and effort for each function and each framework activity

6) Compare the resulting values to those obtained by way of the LOC and FP estimates

- If both sets of estimates agree, then your numbers are highly reliable
- Otherwise, conduct further investigation and analysis concerning the function and activity breakdown

This is the most commonly used of the two estimation techniques (problem and process)

# Process-Based Estimation



# Process-Based Estimation Example

| Activity → | CC | Planning | Risk Analysis | Engineering | | Construction Release | | CE | Totals |
|---|---|---|---|---|---|---|---|---|---|
| Task → | | | | analysis | design | code | test | | |
| | | | | | | | | | |
| Function | | | | | | | | | |
| ▼ | | | | | | | | | |
| UICF | | | | 0.50 | 2.50 | 0.40 | 5.00 | n/a | 8.40 |
| 2DGA | | | | 0.75 | 4.00 | 0.60 | 2.00 | n/a | 7.35 |
| 3DGA | | | | 0.50 | 4.00 | 1.00 | 3.00 | n/a | 8.50 |
| CGDF | | | | 0.50 | 3.00 | 1.00 | 1.50 | n/a | 6.00 |
| DSM | | | | 0.50 | 3.00 | 0.75 | 1.50 | n/a | 5.75 |
| PCF | | | | 0.25 | 2.00 | 0.50 | 1.50 | n/a | 4.25 |
| DAM | | | | 0.50 | 2.00 | 0.50 | 2.00 | n/a | 5.00 |
| | | | | | | | | | |
| | | | | | | | | | |
| **Totals** | 0.25 | 0.25 | 0.25 | 3.50 | 20.50 | 4.50 | 16.50 | | 46.00 |
| | | | | | | | | | |
| **% effort** | 1% | 1% | 1% | 8% | 45% | 10% | 36% | | |

CC = customer communication CE = customer evaluation

Based on an average burdened labor rate of $8,000 per month, the total estimated project cost is $368,000 and the estimated effort is 46 person-months.

# Estimation with Use Cases

Use Cases are widely used. But estimation with use cases is problematic

- Many different formats and Styles
- Many levels of abstraction
- Do not specify complexity of functions

Smith(Rational Corp) suggests using “structural hierarchy” of use cases, scenarios for types of subsystems (e.g. real-time, business, scientific etc.)

# Estimation with Use Cases (contd)

An empirical formula for estimation can be

LOC estimate =

$$N * LOC_{avg} + [(S_a/S_h - 1) + (P_a/P_h - 1)] * LOC_{adjust}$$

Where

N = actual no of use cases

$LOC_{avg}$ = historical LOC based on use case of this type

$S_a$ and $S_h$ = actual and historical scenarios per use case

$P_a$ and $P_h$ = actual and historical pages per use case

$LOC_{adjust}$ = adjustment factor (a fraction of $LOC_{avg}$)

# Estimation with Use-Cases (example)

| | | | | Historical Information | | | |
|---|---|---|---|---|---|---|---|
| | Use cases | Scenarios | Pages | Scenarios | Pages | LOC | LOC estimate |
| Subsystem1 | 6 | 10 | 6 | 12 | 5 | 560 | 3366 |
| Subsystem2 | 10 | 20 | 8 | 16 | 8 | 3100 | 31233 |
| Subsystem3 | 5 | 6 | 5 | 10 | 6 | 1650 | 7970 |
| Total LOC estimate | | | | | | | 42568 |

For Subsystem1, LOC estimate without adjustment = 6*560 = 3360

Adjustment count with 0.3 multiplier = (10/12 – 1 + 6/5 – 1) * 560 * 0.3 ~= 6

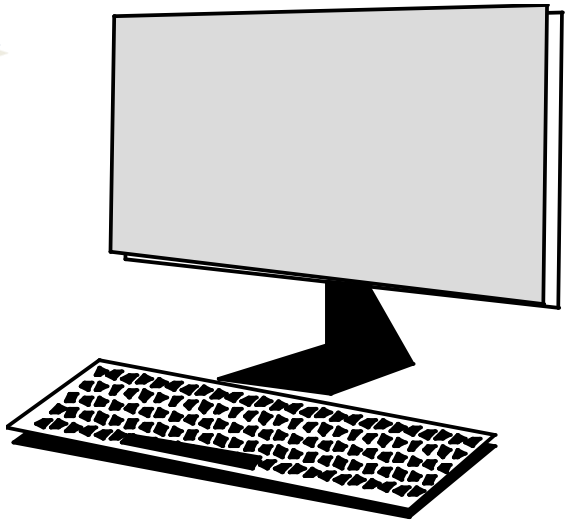
Using 620 LOC/pm as the average productivity for systems of this type and a burdened labor rate of $8000 per month, the cost per line of code is approximately $13. Based on the use-case estimate and the historical productivity data, **the total estimated project cost is $552,000 and the estimated effort is 68 person-months.**

# Tool-Based Estimation

**project characteristics** →

**calibration factors** →

**LOC/FP data** →



# Empirical Models

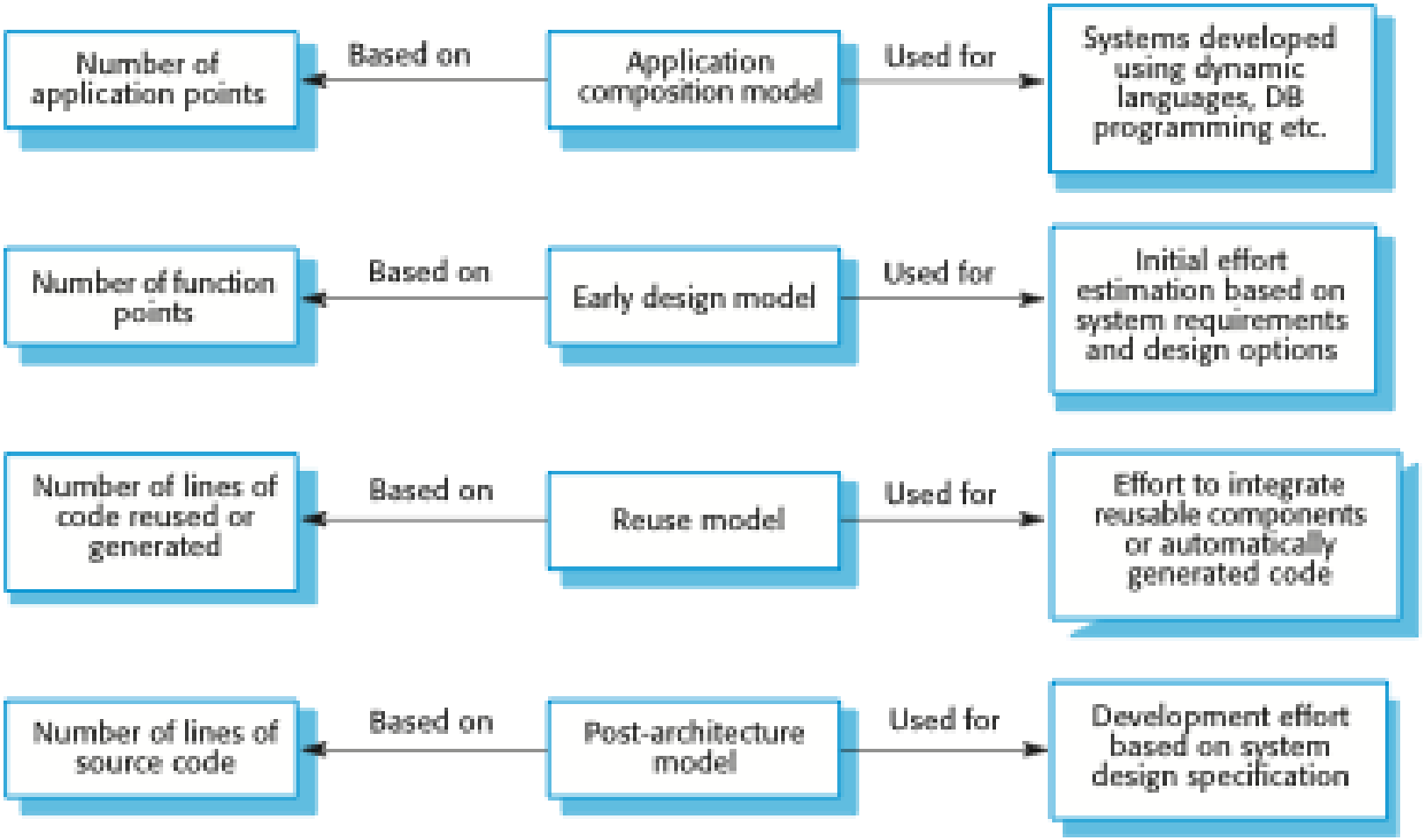
# The COCOMO 2 model

- An empirical model based on project experience.
- Well-documented, 'independent' model which is not tied to a specific software vendor.
- Long history from initial version published in 1981 (COCOMO-81) through various instantiations to COCOMO 2.
- COCOMO 2 takes into account different approaches to software development, reuse, etc.
- Brings out diseconomies of scale in Software Development

# COCOMO 2 models

- COCOMO 2 incorporates a range of sub-models that produce increasingly detailed software estimates.
- The sub-models in COCOMO 2 are:
  - Application composition model. Used when software is composed from existing parts.
  - Early design model. Used when requirements are available but design has not yet started.
  - Reuse model. Used to compute the effort of integrating reusable components.
  - Post-architecture model. Used once the system architecture has been designed and more information about the system is available.

# COCOMO estimation models



# Application composition model

- Supports prototyping projects and projects where there is extensive reuse.
- Based on standard estimates of developer productivity in application (object) points/month.
- Takes CASE tool use into account.
- Formula is
  - PM = ( NAP × (1 - %reuse/100 ) ) / PROD
  - PM is the effort in person-months, NAP is the number of application points and PROD is the productivity.

# Application-points[Boehm98]

## Complexity Weights

| Object Type | Simple | Medium | Complex |
|---|---|---|---|
| Screen | 1 | 2 | 3 |
| Report | 2 | 5 | 8 |
| 3GL component | | | 10 |

## Productivity Rates

| Developer's experience and capability | Very low | Low | Nominal | High | Very high |
|---|---|---|---|---|---|
| ICASE maturity and capability | Very low | Low | Nominal | High | Very high |
| PROD (NAP/month) | 4 | 7 | 13 | 25 | 50 |

# Early design model

- Estimates can be made after the requirements have been agreed.
- Based on a standard formula for algorithmic models
  - $PM = A \times Size^{B} \times M$ where
  - $M = PERS \times RCPX \times RUSE \times PDIF \times PREX \times FCIL \times SCED;$
  - A = 2.94 in initial calibration, Size in KLOC, B varies from 1.1 to 1.24 depending on novelty of the project, development flexibility, risk management approaches and the process maturity.

# Multipliers

- Multipliers reflect the capability of the developers, the non-functional requirements, the familiarity with the development platform, etc.
  - RCPX - product reliability and complexity;
  - RUSE - the reuse required;
  - PDIF - platform difficulty;
  - PREX - personnel experience;
  - PERS - personnel capability;
  - SCED - required schedule;
  - FCIL - the team support facilities.

# The reuse model

- Takes into account black-box code that is reused without change and code that has to be adapted to integrate it with new code.
- There are two elements:
  - Black-box reuse where code is not modified. An effort estimate (PM) is computed.
  - White-box reuse where code is modified. A size estimate equivalent to the number of lines of new source code is computed. This then adjusts the size estimate for new code.

# Reuse model estimates 1

- For generated code:
  - PM = (ASLOC * AT/100)/ATPROD
  - ASLOC is the number of lines of generated code
  - AT is the percentage of code automatically generated.
  - ATPROD is the productivity of engineers in integrating this code.

# Reuse model estimates 2

- When code has to be understood and integrated:
  - ESLOC = ASLOC * (1-AT/100) * AAM.
  - ASLOC and AT as before.
  - AAM is the adaptation adjustment multiplier computed from the costs of changing the reused code, the costs of understanding how to integrate the code and the costs of reuse decision making.

# Post-architecture level

- Uses the same formula as the early design model but with 17 rather than 7 associated multipliers.
- The code size is estimated as:
  - Number of lines of new code to be developed;
  - Estimate of equivalent number of lines of new code computed using the reuse model;
  - An estimate of the number of lines of code that have to be modified according to requirements changes.

# The exponent term

- This depends on 5 scale factors (see next slide). Their sum/100 is added to 1.01
- Example case :
  - A company takes on a project in a new domain. The client has not defined the process to be used and has not allowed time for risk analysis. The company has a CMM level 2 rating.
    - Precedenteness - new project (4)
    - Development flexibility - no client involvement - Very high (1)
    - Architecture/risk resolution - No risk analysis - V. Low .(5)
    - Team cohesion - new team - nominal (3)
    - Process maturity - some control - nominal (3)
  - Scale factor is therefore 1.17.

# Scale factors used in the exponent computation in the post-architecture model

| Scale factor | Explanation |
|---|---|
| Precedentedness | Reflects the previous experience of the organization with this type of project. Very low means no previous experience; extra-high means that the organization is completely familiar with this application domain. |
| Development flexibility | Reflects the degree of flexibility in the development process. Very low means a prescribed process is used; extra-high means that the client sets only general goals. |
| Architecture/risk resolution | Reflects the extent of risk analysis carried out. Very low means little analysis; extra-high means a complete and thorough risk analysis. |
| Team cohesion | Reflects how well the development team knows each other and work together. Very low means very difficult interactions; extra-high means an integrated and effective team with no communication problems. |
| Process maturity | Reflects the process maturity of the organization. The computation of this value depends on the CMM Maturity Questionnaire, but an estimate can be achieved by subtracting the CMM process maturity level from 5. |

# Multipliers

- Product attributes : Concerned with required characteristics of the software product being developed.
  - Required software reliability, Size of application database, Complexity of the product
- Computer attributes : Constraints imposed on the software by the hardware platform.
  - Run-time performance constraints, Memory constraints, Virtual machine environment volatility, Required turnaround time
- Personnel attributes : Multipliers that take the experience and capabilities of the people working on the project into account.
  - Analyst capability, Software engineer capability, Applications experience, Virtual machine experience, Programming language experience
- Project attributes : Concerned with the particular characteristics of the software development project.
  - Use of software tools, Application of SwEng methods , Required development schedule

# The effect of cost drivers on effort estimates

| Exponent value | 1.17 |
|---|---|
| System size (including factors for reuse and requirements volatility) | 128,000 DSI |
| **Initial COCOMO estimate without cost drivers** | **730 person-months** |
| Reliability | Very high, multiplier = 1.39 |
| Complexity | Very high, multiplier = 1.3 |
| Memory constraint | High, multiplier = 1.21 |
| Tool use | Low, multiplier = 1.12 |
| Schedule | Accelerated, multiplier = 1.29 |
| **Adjusted COCOMO estimate** | **2,306 person-months** |

# The effect of cost drivers on effort estimates

| Exponent value | 1.17 |
|---|---|
| Reliability | Very low, multiplier = 0.75 |
| Complexity | Very low, multiplier = 0.75 |
| Memory constraint | None, multiplier = 1 |
| Tool use | Very high, multiplier = 0.72 |
| Schedule | Normal, multiplier = 1 |
| **Adjusted COCOMO estimate** | **295 person-months** |

# Project duration and staffing

- As well as effort estimation, managers must estimate the calendar time required to complete a project and when staff will be required.
- Calendar time can be estimated using a COCOMO 2 formula
  - $TDEV = 3 \times (PM)^{(0.33+0.2*(B-1.01))}$
  - PM is the effort computation and B is the exponent computed as discussed above (B is 1 for the early prototyping model). This computation predicts the nominal schedule for the project.
- The time required is independent of the number of people working on the project.

# Staffing requirements

- Staff required can't be computed by diving the development time by the required schedule.
- The number of people working on a project varies depending on the phase of the project.
- The more people who work on the project, the more total effort is usually required.
- A very rapid build-up of people often correlates with schedule slippage.

# The Software Equation

*A dynamic multivariable model*

$$E = [LOC \times B^{0.333}/P]^3 \times (1/t^4)$$

where

E = effort in person-months or person-years

t = project duration in months or years

B = "special skills factor"

P = "productivity parameter"

- Putnam et al.

# Estimation for OO Projects-I

-Lorenz & Kidd

- Develop estimates using effort decomposition, FP analysis, and any other method that is applicable for conventional applications.
- Using object-oriented requirements modeling, develop use-cases and determine a count.
- From the analysis model, determine the number of key classes.
- Categorize the type of interface for the application and develop a multiplier for support classes:

| **Interface type** | **Multiplier** |
|---|---|
| No GUI | 2.0 |
| Text-based user interface | 2.25 |
| GUI | 2.5 |
| Complex GUI | 3.0 |

# Estimation for OO Projects-II

- Multiply the number of key classes (step 3) by the multiplier to obtain an estimate for the number of support classes.
- Multiply the total number of classes (key + support) by the average number of work-units per class. Lorenz and Kidd suggest 15 to 20 person-days per class.
- Cross check the class-based estimate by multiplying the average number of work-units per use-case

# Agile planning

- Agile methods of software development are iterative approaches where the software is developed and delivered to customers in increments.
- Unlike plan-driven approaches, the functionality of these increments is not planned in advance but is decided during the development.
  - The decision on what to include in an increment depends on progress and on the customer's priorities.
- The customer's priorities and requirements change so it makes sense to have a flexible plan that can accommodate these changes.

# Agile planning stages

- Release planning, which looks ahead for several months and decides on the features that should be included in a release of a system.
- Iteration planning, which has a shorter term outlook, and focuses on planning the next increment of a system. This is typically 2-4 weeks of work for the team.

# Estimation for Agile Projects

- Each user scenario (a mini-use-case) is considered separately for estimation purposes.
- The scenario is decomposed into the set of software engineering tasks that will be required to develop it.
- Each task is estimated separately. Note: estimation can be based on historical data, an empirical model, or "experience."
  - Alternatively, the 'volume' of the scenario can be estimated in LOC, FP or some other volume-oriented measure (e.g., use-case count).
- Estimates for each task are summed to create an estimate for the scenario.
  - Alternatively, the volume estimate for the scenario is translated into effort using historical data.
- The effort estimates for all scenarios that are to be implemented for a given software increment are summed to develop the effort estimate for the increment.

# Agile Techniques

- Similarity to other tasks
- Delphi technique
- Planning Poker (Wideband Delphi)
- Cost set by the stakeholders

# Agile Techniques

- Similarity to Other Tasks

Some of the tasks may be similar to tasks completed. Recollections of those tasks and their duration can be used to estimate the present task's duration.

Triangulation : The process of estimating the size of a user story relative to two other user stories with the purpose of increasing the reliability of the estimate. An example logic of triangulation may be as follows: *I am estimating the size of user story US-2 being equal to 2 story points because it looks to me somewhat larger than user story US-1, which I have already estimated at 1 story point, and somewhat smaller than user story US-3, which I already estimated at 3 story points.*

## Cost set by the stakeholders

Some projects have the cost set for them by the stakeholders. This may be an actual upper limit on what you can spend. It may be the initial amount your stakeholders are willing to spend, but they're prepared to spend more if you're providing good return on investment (ROI).

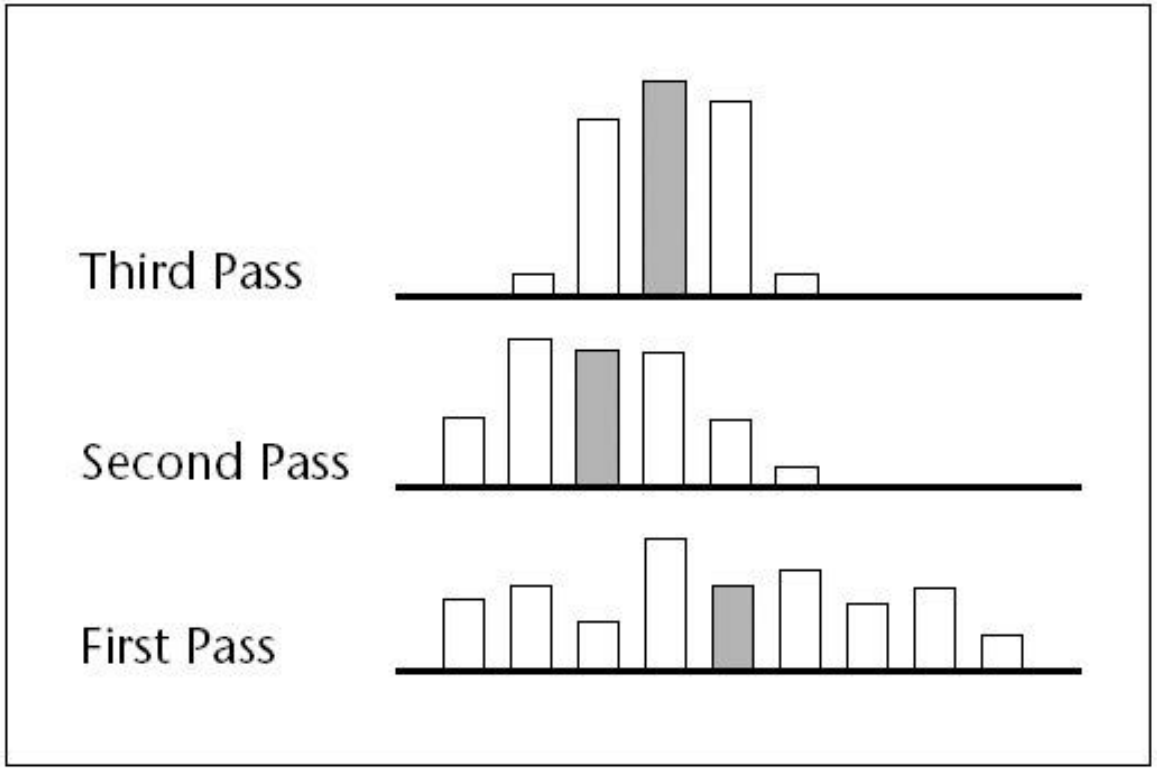
# Delphi Technique

This is a group technique that extracts and summarizes the knowledge of the group to arrive at an estimate.

Each individual in the group is asked to make his or her best guess of the task duration.

Those participants whose estimates fall in the outer quartiles are asked to share the reason for their guess.

After listening to the arguments, each group member is asked to guess again.



# Planning Poker

- Each estimator is given a deck of cards, each card has a valid estimate written on it.
  - The estimate values correspond roughly to the Fibonacci sequence, such as 0, 1, 2, 3, 5, 8, 13, with higher numbers added such as 20 and 40
- Customer/Product owner reads a story and it's discussed briefly
- Each estimator selects a card that's his or her estimate
- Cards are turned over so all can see them
- Discuss differences (especially outliers)
- Re-estimate until estimates converge

– Agile alliance member Mike Cohn

# Reconciling Estimates

- The results gathered from the various estimation techniques must be reconciled to produce a single estimate of effort, project duration, and cost
- If widely divergent estimates occur, investigate the following causes
  - The scope of the project is not adequately understood or has been misinterpreted by the planner
  - Productivity data used for problem-based estimation techniques is inappropriate for the application, obsolete (i.e., outdated for the current organization), or has been misapplied
- The planner must determine the cause of divergence and then reconcile the estimates

# State of the Art for estimating Large Systems

- Formal sizing approaches based on FP
- Secondary sizing approach based on LOC
- Tertiary sizing approaches using information such as screens, reports etc.
- Inclusion of reusable materials in the estimates
- Inclusion of supply chains if multiple companies are involved
- Inclusion of travel costs for international distributed teams
- Comparison with historical benchmark data

Capers Jones

# Rank Order of Large System Software Cost Elements

1. Defect Removal (inspections, static analysis, testing, finding, and fixing bugs)
2. Producing paper documents (plans, architecture, specifications, user manuals)
3. Meetings and communication (clients, team members, managers)
4. Programming
5. Project Management

Capers Jones

# Rank Order of Agile Software Cost Elements

1. Programming
2. Meetings and communication (clients, team members, managers)
3. Defect Removal (inspections, static analysis, testing, finding, and fixing bugs)
4. Project Management
5. Producing paper documents (plans, architecture, specifications, user manuals)

Capers Jones

# Thank You

## Any Questions?