Estimation

SE423: Software Project Management

Outline

- Estimation
- Estimation Methodologies
- Activity Resource Estimating
- Activity Duration Estimating
- Function Points
- COCOMO
- Final Notes

Estimation

Estimation

- Why it's difficult: evolving requirements; technical unknowns; team skill and availability; external dependencies.
- Created, used or refined during
  - Strategic planning: Aligns project goals with organizational capacity and timelines
  - Feasibility study and/or SOW(Statement of Work: document that defines the scope, deliverables, timelines and responsibilities): Determines whether the project is viable and worth pursuing
  - Proposals: Helps justify cost, scope, and timeline to stakeholders or clients
  - Vendor and sub-contractor evaluation: Compares bids, assesses realism, and negotiates contracts
  - Project planning (iteratively): Refines estimates as more information becomes available
- Basic process
  1) Estimate the **size** of the product
  2) Estimate the **effort** (person-months)
  3) Estimate the **schedule**
- NOTE: Not all of these steps are always explicitly performed

Estimation

- Remember, an “exact estimate” is not possible
- Estimate how long will it take you to get home from class today
  - On what basis did you do that?
  - Experience is useful
  - Likely as an “average” (based on historical data), probable estimation
  - For most software projects there is no such ‘average’ Most software estimations are off by 25-100%

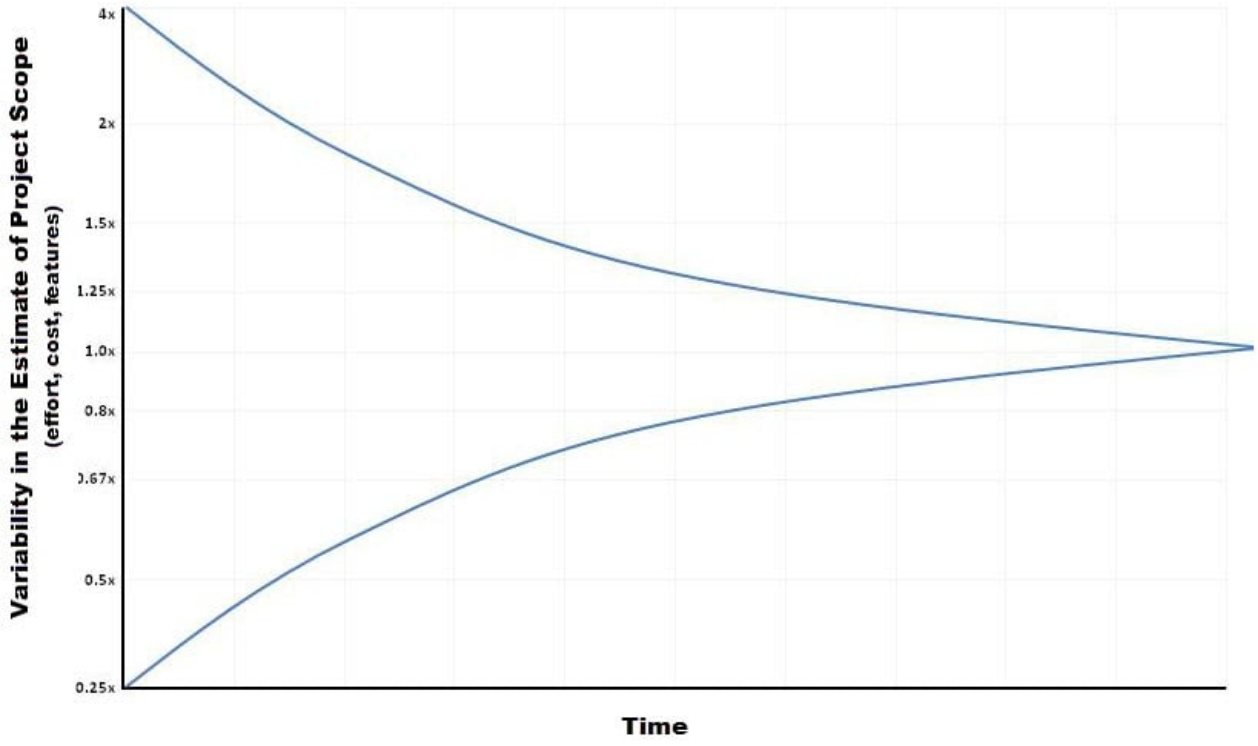
Estimation: definitions

- Target vs. Committed Dates
  - Target: Proposed by business or marketing
    - Do not commit to this too soon! (Learn to say no and how to say it)
  - Committed: Team agrees to this (based on actual planning)
    - After you've developed a schedule
- As project size increases, so does uncertainty:
  - Small projects (10-99 FPs), variance of 7% from post-requirements estimates
  - Medium (100-999 FPs), 22% variance
  - Large (1000-9999 FPs) 38% variance
  - Very large (> 10K FPs) 51% variance

-> push back on target dates, understand that **variance is normal**

Cone of Uncertainty

- The **upper curve** represents the *maximum possible overestimation*.
- The **lower curve** represents the *maximum possible underestimation*.
- Over time, these curves **converge toward 1x**, meaning your estimate becomes more accurate and reliable.
- Early estimates (e.g., during feasibility or proposal stages) can be wildly inaccurate — anywhere from 0.25x to 4x the actual scope.
- As the project moves through requirements gathering, design, and development, uncertainty decreases



Estimation Methodologies

Estimation Methodologies

- Expert Judgment: Relies on the experience and intuition of seasoned professionals.
- Top-down: Starts with an overall project estimate, then breaks it down into components. Useful in early planning stages when only high-level scope is known.
- Bottom-up: Builds estimates from detailed task-level analysis, then aggregates them. Requires a well-defined scope and work breakdown structure (WBS).
- Analogy: Compares the current project to past similar projects.
- Priced to Win (request for quote – RFQ): Estimation driven by competitive pricing or client budget constraints.
- Parametric or Algorithmic Method: Uses mathematical models and historical data to generate estimates (COCOMO, Function Point analysis, etc.)

Expert Judgment

- Use somebody who has recent experience on a similar project
- You get a “guesstimate”
- Accuracy depends on their ‘real’ expertise
- Comparable application(s) must be accurately chosen

Top-down Estimation

- Based on overall characteristics of project
  - Some of the others can be “types” of top-down (Analogy, Expert Judgment, and Algorithmic methods)
- Advantages
  - Easy to calculate, quick and strategic
  - Effective early on (like initial cost estimates)
- Disadvantages
  - Less accurate because may overlook complexity in individual tasks.

Bottom-up Estimation

- Create WBS Work Breakdown Structure, identify individual tasks to be done.
- Add from the bottom-up
- Advantages
  - More accurate and granular.
  - Works well if activities well understood
- Disadvantages
  - Specific activities not always known
  - More time consuming and depend on detailed information

Estimation by Analogy

- Use past project
  - Must be sufficiently similar (technology, type, organization)
  - Find comparable attributes (ex: # of inputs/outputs)
- Advantages
  - Based on actual historical data
- Disadvantages
  - Difficulty ‘matching’ project types
  - Prior data may have been mis-measured
  - How to measure differences – no two exactly same

Priced to Win

- The quote may not reflect actual costs or effort required.
- Needs information on other estimates (or prices): To price competitively, vendors need access to market intelligence: previous bids, or client budget expectations (competitor pricing is not public).
- Purshaser must closely watch trade-offs: Is the low price realistic? Will quality or scope suffer?
- Priced to lose? some vendors may intentionally underbid to win the contract, knowing they'll cut corners, request scope change later, absorb losses for strategic reasons (e.g., market entry)
- Strength: Aligns with market expectations.
- Risk: May undercut realistic effort, leading to overruns.

Algorithmic Measures

- Lines of Code (LOC)
- Function points
  - Feature points or object points
- Other possible
  - Number of bubbles on a DFD
  - Number of ERD entities
  - Number of processes on a structure chart
- LOC and function points most common (of the algorithmic approaches)
- Majority of projects use none of the above

Wideband Delphi

- Group consensus approach
- Rand Corp. used original Delphi approach in the 1940's to estimate future projects
- Present experts with a problem and response form: They fill out initial estimates independently.
- Conduct group discussion, collect anonymous opinions, then share feedback: This helps surface hidden factors and clarify misunderstandings
- Conduct another discussion & iterate until consensus
- Advantages
  - Easy, inexpensive, utilizes expertise of several people
  - Does not require historical data
- Disadvantages
  - Difficult to repeat
  - May fail to reach consensus, reach wrong one, or all may have same bias

Code-based Estimates

- LOC Advantages
  - Commonly understood metric: intuitive and familiar to most developers and managers
  - Permits specific comparison
  - Actuals easily measured: Once code is written, LOC is straightforward to count
- LOC Disadvantages
  - Difficult to estimate early in cycle (Before requirements and design are clear)
  - Counts vary by language and availability of reusable components and programmer style (avg. programmer productivity: 3,000 LOC/yr)
  - Many costs not considered (ex: requirements, UI design, etc)
  - Programmers may be rewarded based on this: If incentives are tied to LOC, developers may write more code than necessary, reducing efficiency and maintainability.
  - Code generators produce repeated code: Auto-generated code inflates LOC without reflecting actual developer effort, skewing estimates and metrics.

Function Points

- Software size measured by number & complexity of functions it performs
- More methodical than LOC counts
- Four basic steps (next slide)

Function Point Process

1. Count # of business functions per category
   - Categories: outputs, inputs, db inquiries, files or data structures, and interfaces
2. Establish Complexity Factor for each and apply
   - Simple, Average, Complex
   - Set a weighting multiplier for each (0 –>15)
   - This results in the “unadjusted function-point total”
3. Compute an “influence multiplier” and apply
   - Based on 14 factors rated from 0 to 5 (such as performance requirements, data communications, transactions rates, reusability, etc)
4. Results in “function point total”
   - This can be used in comparative estimates

Effort Estimation

Man Power Basic calculation.

- Now that you know the “size” (function points, LOC, etc.), determine the “effort” needed to build it (in person-months or staff-months)
- Various models:
  - Empirical: Based on historical data and past project performance
  - Mathematical: Uses formulas and algorithms, example: Effort = a × (Size)^b
  - Subjective: Relies on expert judgment and team intuition (Wideband Delphi)
- Depends on Efficiency/productivity of people
  - Motivation
  - Ability of team
  - Application experience
  - Range of personnel

Duration

- What is the duration?
  The duration of a project is the elapsed time in business working days, not including weekends, holidays, or other nonworking days.
- Duration is different from work effort.
  - **Duration** = Work Effort ÷ Number of People -> more useful in scheduling
  - Work Effort = **Duration** × Number of People -> more useful in cost estimation
  - Work effort is labor required to complete an activity.
  - E.g. five people reviewing a document; the duration may be 10 hours but the work effort is 50 staff-hours

Duration

- When we talk about estimates and duration, there are two types of time that are not the same:
  - Labor Time: actual effort required to complete a task (in staff-hours)
  - Clock time: elapsed calendar time needed to complete the task.
- For example, if we estimate that a task would require 40 hours of labor to complete, then in a normal business setting we need at least 50 business hours, ***Why***?
  - Productivity is usually 50-75% of a business day
  - a developer might only get 5–6 productive hours per day of 8 office hours (meetings, emails and communication, breaks and fatigue, tasks switching and setup, etc.)

Causes of Variation

- Since we cannot know what factors will be effective when work is underway on an activity, we cannot know exactly how long it will take.
- Causes of Variation in the Actual Activity Duration
  - Varying skill levels: Our estimate of the activity duration is based on average skilled engineer. We may get a higher or lower skilled engineer assigned to the activity, causing the actual duration to vary from the planned duration.
  - Unexpected events: Random natural events, vendor delays, power failures, etc.
  - Efficiency of work time: Every time a worker is interrupted it takes more time to get up to the level of productivity prior to the time of the interruption. These interruptions may have little or substantial impact on the worker's productivity
  - Mistakes and Misunderstandings: Even in organizations that have a documented process, rework may have its impact on the actual activity duration.

Estimation Issues

- Quality estimations needed early but information is limited
- Precise estimation data available at end but not needed anymore
  - Or is it? What about the next project?
  - At the end of the project you should have determined just how accurate your original estimate was.
- Best estimates are based on past experience
- Politics of estimation:
  - You may anticipate a “cut” by upper management (so you add some contingency duration)
- For many software projects there is little or no data for estimation
  - Technologies change
  - Historical data unavailable
  - Wide variance in project experiences/types
  - Subjective nature of software estimation

Over and Under Estimation

- Over estimation issues
  - The project will not be funded: Conservative estimates guaranteeing 100% success may mean funding probability closer to zero.
  - Parkinson's Law: Work expands to take the time allowed
  - Danger of feature and scope creep
  - Be aware of “double-padding”: project member + business manager
- Under estimation issues
  - Quality issues (shorten key phases like testing)
  - Inability to meet deadlines
  - Morale and other team motivation issues

Introduction: Activity Resource Estimation

- Resource estimation is concerned with *who/what*, *how much*, and *when*
- *Who/what*
  - Resource requirements of people or equipment
  - Matching the right resources to activities
  - May be generic (*n* Java programmers) or resource-specific (*Salah* as senior architect)
- *How much:* Amount of resource(s) needed for activity
- *When:* When the resource will be needed for activities

Controlling factors in resource estimation

- Controlling factors in resource estimation
  - **Enterprise environment:** May be defined by constraints and assumptions identified in Project Scope Statement
  - **Organization policy:** How the organization allocates and manages internal and external resources.
  - **Resource availability:** When the resource will be available for activities and whether dedicated or part-time

Estimating tools

- ***Expert judgment*** One of the most effective tools for estimating. May be used in conjunction with other tools
- ***Alternative analysis*** Examines different approaches to staff and deliver work, such as **in-house development *vs.* contracting *vs*. COTS** Commercial of The Shelf (oracle for example). Within your organization.
- ***Published estimating data*** Provides empirical baselines for productivity, typical staffing levels, and effort per unit of size (LOC, FP, features). May be available in-house in form of historical project data or may be available from outside sources (*published survey research papers, technical reports*)

Activity Resource Estimation Output

- *Activity resource requirements*
  - Identify what and how much of resource(s) are needed for each activity in a work package (3 developer, 1 tester…)
  - Activity resource requirements are summed to provide estimate for entire work package
- *Resource calendar*
  - Documents working and non-working days
  - Resource-specific holidays may be identified
- *Requested changes*
  - Activity resource estimating process may generate requests for changes to activity list

Introduction: Activity Duration Estimation

- Focuses on estimating the number of work periods needed to complete individual schedule activities
- Estimates originate from person or groups most familiar with the type of work needed in the activity
- Estimate is progressively elaborated: duration estimates are improved as project progresses
- Activity duration output
  - Quantitative activity duration estimates, including range of possible values

Controlling factors in duration estimation

- Controlling factors in duration estimation
  - ***Historical data*** If available, enterprise- or industry-specific historical data can be used as starting point for estimates
  - ***Activity resource requirements*** The type and number of resources applied to an activity must account for
    - Resource qualifications
    - Quantity / productivity trade-offs: putting more people on a task may reduce productivity
  - ***Resource calendars*** Must account for full/part time commitments

A useful rule of thumb

- Keep all times in the same units
- Don't mix increments
  - *Example:* Schedule everything in hours, then convert to days and fractions as a last step
  - *Note:* Most PM software tools enforce this rule by using a uniform time unit and increment

Estimating tools

To improve the accuracy and resilience of project schedules:

- ***Reserve analysis:*** **C**ontingency or time reserves can be added to schedule to allow for schedule risks (uncertainty)
  - *Contingency reserves* may be used or adjusted as more precise project information becomes available
  - *Known as buffer time:* Do not confuse with slack time.
- ***Three-point estimation techniques*** use a mathematical combination of three different estimates: *Most likely, Optimistic, and Pessimistic:*
  - *Basic:*
    - *Estimate = { M + O + P } / 3 : get an average of the three estimations*

Three Point Estimating

- Used whenever there is a potential variance in an estimate. Assumes a Gaussian distribution of estimates.
  - Because of uncertainties in: Duration, Effort, Size

- ***Three-point estimating techniques*** use a mathematical combination of three different estimates:
  - best-case, worst-case, and expected scenarios
  - Get estimates from experts: Most likely, Optimistic, and Pessimistic
  - *Compute an average: (Gaussian distribution assumption)*
    (Optimistic + 4(Likely)+ Pessimistic)/6

Example: Three Point Estimation of LOC

CAD program to represent mechanical parts

- Estimated LOC = (Optimistic + 4(Likely)+ Pessimistic)/6

Major Software Functions	ID	Optimistic	Most Likely	Pessimistic	Estimated LOC
User interface and control facilities	UICF	1,500	2,300	3,100	2,300
Two-dimensional geometric analysis	2DGA	3,800	5,200	7,200	5,300
Three-dimensional geometric analysis	3DGA	4,600	6,900	8,600	6,800
Database management	DBM	1,600	3,500	4,500	3,350
Computer graphics display features	CGDF	3,700	5,000	6,000	4,950
Peripheral control	PC	1,400	2,200	2,400	2,100
Design analysis modules	DAM	7,200	8,300	10,000	8,400
Estimated lines of code		23,800	33,400	41,800	33,200

Example: Three Point Estimation using Function Points

- Each function is divided into one of five categories, then multiplied by the appropriate number below

Function type	Low	Average	High
External inputs	× 3	× 4	× 6
External outputs	× 4	× 5	× 7
External queries	× 3	× 4	× 6
Internal logical files	× 7	× 10	× 15
External interface files	× 5	× 7	× 10

COCOMO

COnstructive COst Model (COCOMO)

- COnstructive COst Model: An algorithmic software cost estimation model
- Uses a basic regression formula with parameters that are derived from historical project data and current project characteristics: $E = a(KLOC)^b$
- Helps estimate **person-months** (effort) based on the **size of the software:** Input – LOC, Output - Person Months
- Intermediate COCOMO Adjusts its estimates based on the type of application, size, and “Cost Drivers”
- Cost drivers using High/Med/Low levels, include:
  - Motivation, Ability of team, Application experience, etc.
- Biggest weakness?
  - Requires input of a product size estimate in LOC (needs to be reliable estimate of LOC)

Basic COCOMO – More Constants

In COCOMO, projects are categorized to one of three types:

- **Organic**: Small , simple projects with experienced teams and flexible requirements.
- **Semi-Detached**: Medium complexity, mixed experience, some constraints.
- **Embedded**: Complex, creativity needed, tightly coupled systems with strict requirements (e.g., real-time or safety-critical systems).

Basic COCOMO:

$$E = a(KLOC)^b$$

- E is the Effort in staff months
- a and b are coefficients to be determined
- KLOC is thousands of lines of code
- a and b used for (calculate Effort from KLOC)
- c and d used for (calculate Time from Effort)

Basic COCOMO – The Modes

- Organic: 2 to 50 KLOC, small, stable, little innovation
- Semi-detached: 50 to 300 KLOC, medium-sized, average abilities, medium time constraints
- Embedded: > 300 KLOC, large project team, complex, innovative, severe constraints

Mode	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semi-Detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Basic COCOMO – Example

- Suppose size is 200 KLOC
- Organic
  - $2.4(200)^{1.05} = 626$ staff-months
- Semi-Detached
  - $3.0(200)^{1.12} = 1{,}133$ staff-months
- Embedded
  - $3.6(200)^{1.20} = 2{,}077$ staff-months

$$E = a(KLOC)^b$$

- E is the Effort in staff

Mode	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semi-Detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Basic COCOMO – Development Time

$$TDEV = c(E)^d$$

- TDEV is time for development
- c and d are constants to be determined
- E is the effort

Mode	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semi-Detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Basic COCOMO – Example

- Suppose size is 200 KLOC
- Organic
  - E = 626 staff months
  - TDEV = $2.5(626)^{0.38}$= 29 months
- Semi-Detached
  - E = 1,133
  - TDEV = $2.5(1133)^{0.35}$= 29 months
- Embedded
  - E = 2077
  - TDEV = $2.5(2077)^{0.32}$= 29 months

$$TDEV = c(E)^{d}$$

- TDEV is time for development

Mode	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semi-Detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Basic COCOMO – Average Staff Size

COCOMO estimates three primary things:

- Effort (E): Measured in person-months, it's the total amount of work required.
- Time to Develop (TDEV): Measured in months, it's the schedule duration.
- Staff Size (SS): The average number of people working on the project.

$$SS = \frac{E}{TDEV}$$

$$SS = \frac{Effort\ Applied}{Development\ Time}$$

$$SS = \frac{staff\ \cancel{months}}{\cancel{months}} \rightarrow staff/month$$

$$SS = staff$$

Basic COCOMO – Productivity

- COCOMO estimates **Effort (E)** based on project attributes, and **Size** is often the input (e.g. estimated KLOC).

- Once you have both, you can compute **Productivity (P)** to evaluate how efficiently your team is working (or should be working)

Size / Effort = Productivity.

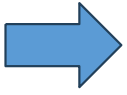
$$P = \frac{Size}{E}$$

$$P = \frac{[KLOC]}{[staff\ months]}$$

COCOMO Complete Example - Organic

- Suppose an organic project has 7.5 KLOC,
  - Effort $2.4(7.5)^{1.05}$= 20 staff–months
  - Development time $2.5(20)^{0.38}$= 8 months
  - Average staff per month needed: 20 / 8 = 2.5 staff per month
  - Productivity 7,500 LOC / 20 staff-months = 375 LOC / staff-month

Item	Organic
Effort (staff-months)	20
Development Time	8
Average Staff	2.5
Productivity	375

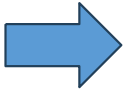


Conclusion: Your target productivity for the given estimation is of 375 lines of code monthly on average per staff member

COCOMO Complete Example - Embedded

- Suppose an embedded project has 50 KLOC,
  - Effort $3.6(50)^{1.20}$= 394 staff–months
  - Development time $2.5(394)^{0.32}$= 17 months
  - Average staff per month needed: 394 / 17 = 23 staff per month
  - Productivity 50,000 LOC / 394 staff-months = 127 LOC / staff-month

Item	Embedded
Effort (staff-months)	394
Development Time	17
Average Staff	23
Productivity	127



Conclusion: ?

Intermediate COCOMO – Cost Drivers

- The basic COCOMO model assumes that the effort is only a function of the number of lines of code and some constants evaluated according to the different software systems. However, in reality, no system's effort and schedule can be solely calculated based on LOC.
- The Intermediate COCOMO Model utilizes 15 drivers for cost estimation: **Cost Drivers (multipliers)**

	Ratings					
Cost Drivers	Very Low	Low	Nominal	High	Very High	Extra High
Product attributes						
Required software reliability	0.75	0.88	1.00	1.15	1.40	
Size of application database		0.94	1.00	1.08	1.16	
Complexity of the product	0.70	0.85	1.00	1.15	1.30	1.65
Hardware attributes						
Run-time performance constraints			1.00	1.11	1.30	1.66
Memory constraints			1.00	1.06	1.21	1.56
Volatility of the virtual machine environment		0.87	1.00	1.15	1.30	
Required turnabout time		0.87	1.00	1.07	1.15	
Personnel attributes						
Analyst capability	1.46	1.19	1.00	0.86	0.71	
Applications experience	1.29	1.13	1.00	0.91	0.82	
Software engineer capability	1.42	1.17	1.00	0.86	0.70	
Virtual machine experience	1.21	1.10	1.00	0.90		
Programming language experience	1.14	1.07	1.00	0.95		
Project attributes						
Application of software engineering methods	1.24	1.10	1.00	0.91	0.82	
Use of software tools	1.24	1.10	1.00	0.91	0.83	
Required development schedule	1.23	1.08	1.00	1.04	1.10	

Intermediate COCOMO – Cost Drivers

Intermediate COCOMO introduces Cost Drivers

- They are used because
  - they are statistically significant to the cost of the project; and
  - they are not correlated to the project size (KLOC)
- To obtain multiplier
  - Determine each number using the grid
  - **Multiply them – the product is C**

$$E = a(KLOC)^b \times C$$

- E is the Effort
- a and b are constants as before
- KLOC is thousands of lines of code
- C is the effort adjusted factor: **Product of all Factors**

COCOMO Advantages

- Based on history
- Repeatable
- Unique adjustment factors
- Has different modes
- Works well on similar projects
- Highly calibrated
- Well-documented
- Easy to use

COCOMO Limitations

- Ignores requirements volatility
- Ignores documentation
- Ignores customer's "skill"
- Oversimplifies security
- Ignores software safety
- Ignores personnel turnover
- Ignores many hardware issues
- Personnel experience may be obsolete
- Must know the cost drivers
- Must be able to predict project size

Final Notes

Code Reuse and Estimation

- Code reuse does not come for free (understanding the code, adapting, testing, integrating, etc.): consume time and effort.
- Code types: New, Modified, Reused
- If code is more than 50% modified, it's "new": estimated as if totally "new"
- Reuse factors have wide range
  - Reused code takes 30% effort of new
  - Modified is 60% of new
- Integration effort with reused code almost as expensive as with new code

Estimation for Agile

- Agile estimation starts with **user stories or scenarios:** Each user scenario is considered separately (as a mini-project)
- The scenario is decomposed into a set of engineering tasks (UI, backend..)
- Each task is estimated separately
  - May use historical data, empirical model, or experience
  - Scenario volume can be estimated (LOC, FP, use-case count, etc.)
- Total scenario estimate computed
  - Sum estimates for each task
  - Translate volume estimate to effort using historical data or COCOMO
- The effort estimates for all scenarios in the increment are summed to get an increment estimate

Know Your Deadlines

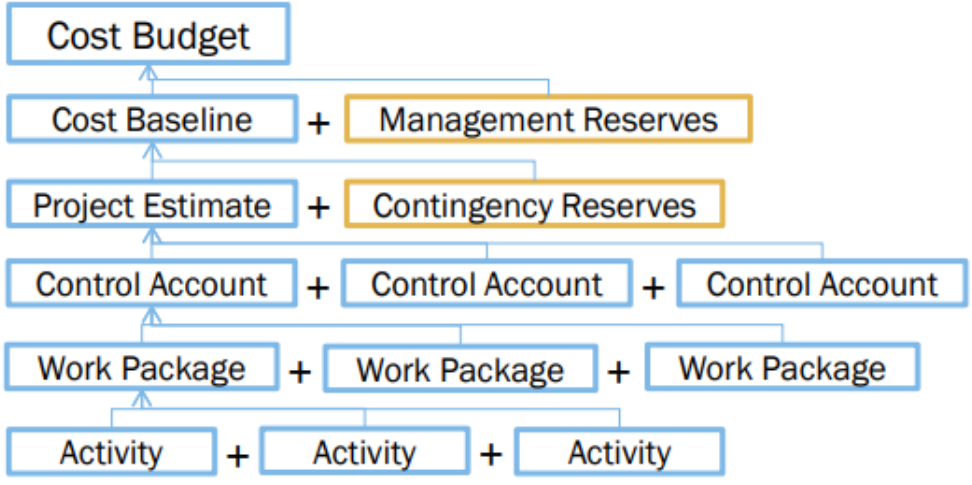
- Are they ‘Real Deadlines’?
  - Tied to an external event
  - Have to be met for project to be a success
  - Ex: end of financial year, contractual deadline, Y2K
- Or ‘Artificial Deadlines’?
  - Set by arbitrary authority
  - May have some flexibility (if pushed)

Estimation Presentation (Communication)

- How you present the estimation can have huge impact
- Techniques
  - Plus-or-minus qualifiers
  - 6 months +/-1 month -> Communicates a **central estimate** with a **buffer**
  - Ranges
    - 6-8 months
  - Risk Quantification
    - +/- with added information (more info about uncertainty)
    - Example: +1 month (if new tool not working as expected)
    - -2 weeks (if hiring new developers happens sooner)
  - Cases: offer scenario based estimates (if things go well … otherwise ….)
    - Best / Planned / Current / Worst cases
  - Coarse Dates : useful when finer estimates are not feasible
    - Q3 of 2026
  - Confidence Factors
    - April 1 with 10% probability, July 1 with 50% probability, etc.

Final Estimates

- For Time or Cost Estimates:
  - Aggregation into larger units (Work Packages, Control Accounts, etc.)
  - Perform Risk Analysis to calculate Contingency Reserves (Controlled by PM)
  - Add Management Reserves: Set aside to cover unforeseen risks or changes (Total company funds available – requires Change Control activities to access)



Estimation Guidelines

- Estimate iteratively!
  - Process of gradual refinement
  - Make your best estimates at each planning stage
  - Refine estimates and adjust plans iteratively
  - Plans and decisions can be refined in response
  - Balance: too many revisions vs. too few

Other Estimation Factors

- Account for resource experience or skill
  - Up to a point
  - Often needed more on the “low” end, such as for a new or junior person
- Allow for “non-project” time & common tasks
  - Meetings, phone calls, web surfing, sick days
- There are commercial ‘estimation tools’ available
  - They typically require configuration based on past data

Other Estimation Notes

- Remember: “manage expectations”
- Parkinson’s Law
  - “Work expands to fill the time available”
- The Student Syndrome
  - Procrastination until the last minute (cram)