Software Engineering Project Management & Planning, Part 3

Erik Fredericks // frederer@gvsu.edu

Adapted from materials provided by Byron DeVries, Jagadeesh Nandigam

Outline

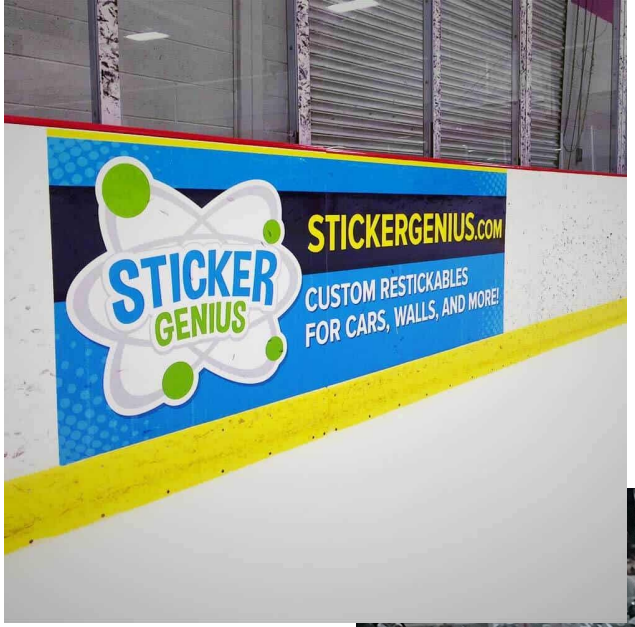
Project Cost Estimation

Project Tracking with Earned Value Analysis (EVA)

BUT FIRST, a diversion

Let's pretend we are running a hockey arena

- What are some ways we can make money?
- Food/service?
- Advertising revenue, yes
  - Let's make some digital ad space for the TV viewers on the hockey rink boards!
  - Instead of requirements, let's make a few user stories...



https://v.redd.it/aadfajggnht91

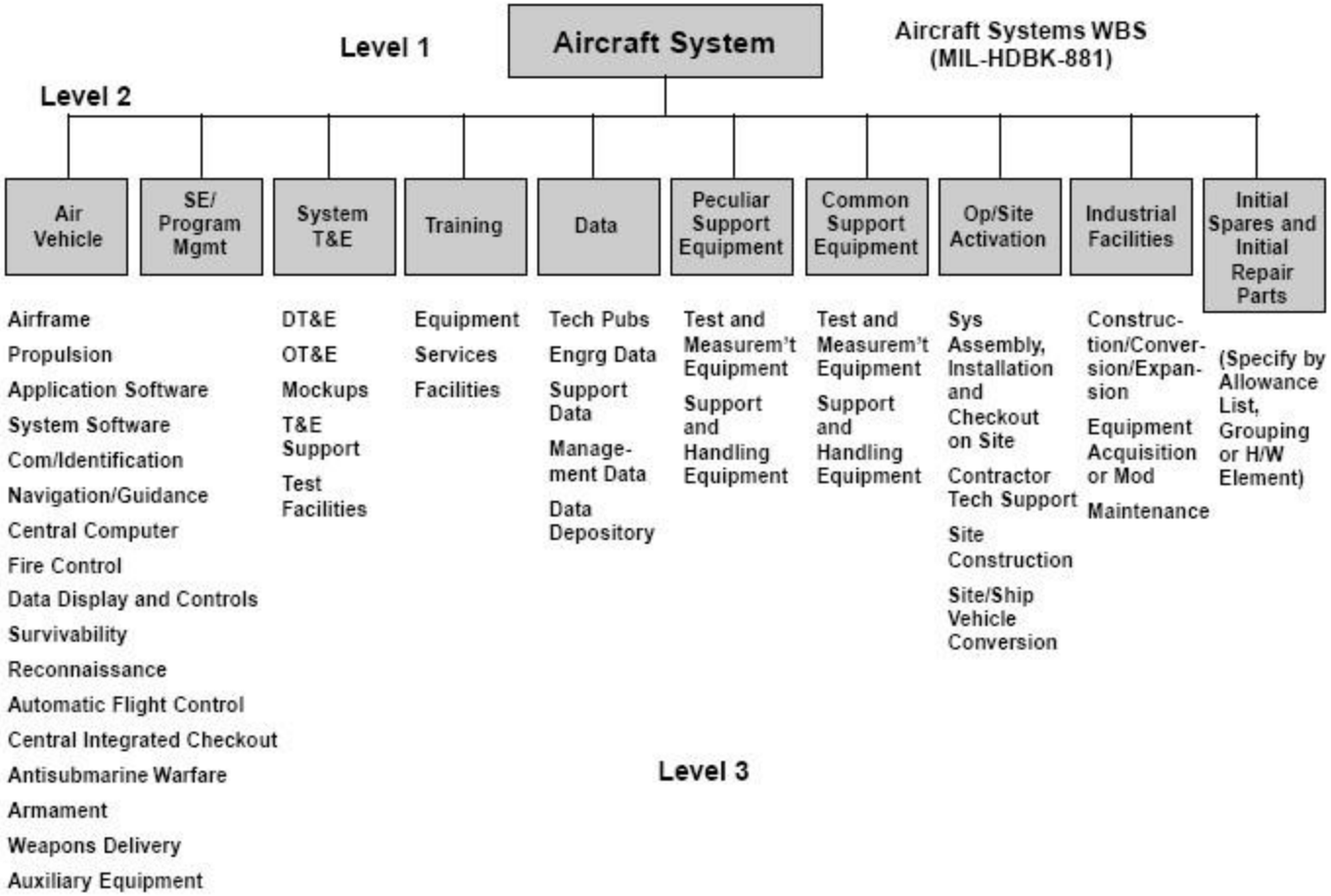
Cost Estimation Techniques

Expert Judgement

- Solicit estimates from multiple experts in software and application domain
- Arrive at an agreed estimate

Several experts on the proposed software development techniques and the application domain are consulted. They each estimate the project cost. These estimates are compared and discussed. The estimation process iterates until an agreed estimate is reached

Often use a **work breakdown structure** to divide the work into appropriate domains and pieces that can be estimated.

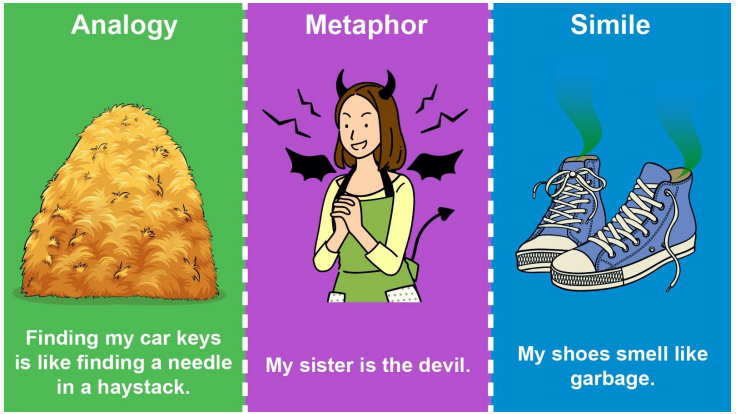

Level 1
Aircraft System
Aircraft Systems WBS
(MIL-HDBK-881)
Level 2
Air Vehicle
SE/ Program Mgmt
System T&E
Training
Data
Peculiar Support Equipment
Common Support Equipment
Op/Site Activation
Industrial Facilities
Initial Spares and Initial Repair Parts
Airframe
Propulsion
Application Software
System Software
Com/Identification
Navigation/Guidance
Central Computer
Fire Control
Data Display and Controls
Survivability
Reconnaissance
Automatic Flight Control
Central Integrated Checkout
Antisubmarine Warfare
Armament
Weapons Delivery
Auxiliary Equipment
DT&E
OT&E
Mockups
T&E Support
Test Facilities
Equipment
Services
Facilities
Tech Pubs
Engrg Data
Support Data
Manage-ment Data
Data Depository
Test and Measurem't Equipment
Support and Handling Equipment
Test and Measurem't Equipment
Support and Handling Equipment
Sys Assembly, Installation and Checkout on Site
Contractor Tech Support
Site Construction
Site/Ship Vehicle Conversion
Construc-tion/Conver-sion/Expan-sion
Equipment Acquisition or Mod
Maintenance
(Specify by Allowance List, Grouping or H/W Element)
Level 3

Cost Estimation Techniques

Estimation by Analogy

- Cost is estimated using costs of similar completed projects

This technique is applicable when other projects in the same application domain have been completed. The cost of a new project is estimated by analogy with these completed projects.



Cost Estimation Techniques

Parkinson's Law

- Work expands so as to fill the time available for its completion
- Cost is determined by available resources

Parkinson's Law states that work expands to fill the time available. The cost is determined by available resources rather than by objective assessment. If the software has to be delivered in **12 months** and **5 people** are available, the effort required is estimated to be **60 person- months**

Cost Estimation Techniques

Pricing to Win

- Cost is determined by customer's available budget

The software cost is estimated to be whatever the customer has available to spend on the project. The estimated effort depends on the customer's budget and not on the software functionality

Cost Estimation Techniques

Schedule-driven budget estimation

- Based on resources allocated to tasks in the baseline schedule

The software is estimated to be the cost of the supplied resources over the course of the schedule.

How does this differ from Parkinson's Law?

Schedule is estimated, rather than pre-defined.

Cost Estimation Techniques

Algorithmic Cost Estimation

- Estimate the size of the problem
- Use an empirically obtained algorithm/model that relates size to effort/cost

A model based on historical cost information that relates some software metric (usually its size) to the project cost is used. An estimate is made of that metric and the model predicts the effort required

Algorithmic Cost Estimation with COCOMO

Best known algorithmic model

COnstructive **CO**st **MO**del developed by Dr. Barry Boehm

Empirical model derived by collecting data from a large number of completed software projects

See any issues so far?

COCOMO models

Two primary COCOMO Models:

COCOMO I (aka COCOMO 81)
- Original model first published in 1981
- Identified with waterfall process

COCOMO II (aka COCOMO 2000)
- Updated model, first published in 2000
- Accounts for recent changes in software methodologies

COCOMO I model

Estimates effort (cost) as a function of project size

Includes different measures of size, including:

- Lines of Code
  - Non-comment source lines
  - KLOC (thousands of lines of code)
- Function-Points
  - External inputs and outputs
  - User interactions
  - External Interfaces
  - Data files used by the system
- Object/Application Points
  - Number of user interface screens
  - Number of reports produced
  - Number of software components

COCOMO I Model

COCOMO I model

Three modes (or levels) in COCOMO I:

Basic: computes effort as a function of program size (applied early in the project)

Intermediate: computes effort as a function of program size and a set of cost drivers that include subjective assessments of product, computer, personnel and project attributes (applied after requirements are specified)

Advanced: computes effort as a function of program size and a set of cost drivers weighted according to each phase of the software lifecycle (applied after design is complete)

COCOMO I model

Three development (or project) modes in COCOMO I:

Organic: relatively small, simple software projects in which small teams with good application experience work to a set of less than rigid requirements

Semi-Detached: an intermediate (in size and complexity) software project in which teams with mixed experience levels must meet a mix of rigid and less than rigid requirements

Embedded: a software project that must be developed within a set of tight hardware, software and operational constraints

General formula for cost/effort estimation:

$$Effort = C * Size^{k} * M$$

- The constants *C* and *K* are established empirically and given by the model.
- The multiplier *M* is based on product, project, computer, and personnel attributes.
- Measure of effort:
  - **Person-months**, where one person month is the amount of work/effort by one person in one month.

https://www.geeksforgeeks.org/software-engineering-cocomo-model/?ref=rp

Basic Model:

- **Organic**: $Effort = 2.4 * (KLOC)^{1.05}$
- **Semi-Detached**: $Effort = 3.0 * (KLOC)^{1.12}$
- **Embedded**: $Effort = 3.6 * (KLOC)^{1.20}$

Intermediate Model:

- **Organic**: $Effort = 3.6 * (KLOC)^{1.05} * M$
- **Semi-Detached**: $Effort = 3.0 * (KLOC)^{1.12} * M$
- **Embedded**: $Effort = 2.8 * (KLOC)^{1.20} * M$

COCOMO I model

COCOMO I intermediate and advanced models use 15 cost drivers to calculate *M*

- Cost drivers are grouped into four categories:
  - Product, Computer, Personnel, and Project

- Each cost driver is rated on a six-point ordinal scale:
  - very low, low, nominal, high, very high, exceptionally high

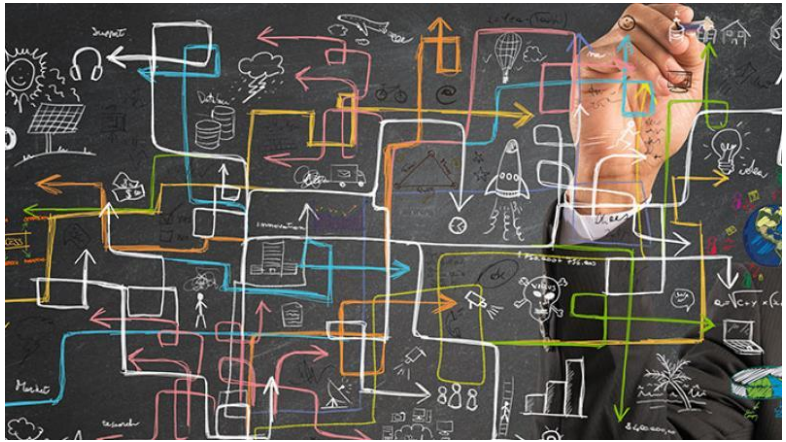
COCOMO I model

Product cost drivers:

- RELY: Required Software Reliability
- DATA: Database Size
- CPLZ: Product Complexity

Project Cost Drivers:

- MODP: Modern Programming Practices
- TOOL: Software Tools
- SCED: Development Schedule



COCOMO I model

Computer Cost Drivers:

- TIME: Execution Time Constraint
- STOR: Main Storage Constraint
- VIRT: Virtual Machine Volatility
- TURN: Computer Turnaround Time

Personnel Cost Drivers:

- ACAP: Analyst Capability
- AEXP: Applications Experience
- PCAP: Programmer Capability
- VEXP: Virtual Machine Experience
- LEXP: Language Experience

COCOMO I model

Total development time (in months) required for the three project modes:

Organic: $TDEV = 2.5 * (Effort)^{0.38}$

Semi-Detached: $TDEV = 2.5 * (Effort)^{0.35}$

Embedded: $TDEV = 2.5 * (Effort)^{0.32}$

SOFTWARE PROJECTS	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

COCOMO I model ➜ Example

Using COCOMO I basic model, estimate effort and duration for an embedded system estimated at 10,000 delivered source instructions:

Effort $= 3.6 * (10)^{1.20}$
➜ *Effort = 58 person-months*

TDEV $= 2.5 * (58)^{0.32}$
➜ *TDEV = 9 months*

COCOMO I model

How likely is it that the COCOMO I Model will estimate the correct effort and schedule?

CAVEATS

The **time required** to complete a project is a function of the total effort required for the project.

It does not depend on the number of software engineers working on the project

CAVEATS

Dividing the effort (in person-months) required on a project by the development schedule (in months) does not give a useful indication of the number of people required for the project team.

It does not depend on the number of software engineers working on the project

What do you think, is this true?

Project tracking (or Earned Value Analysis (EVA))

Earned Value Analysis (EVA) is a measure of progress in a project.

- Assess the “percent of completeness” of a project using quantitative analysis rather than relying on qualitative approaches or gut feelings.

- EVA can only be conducted if resources and their rates/costs have been assigned to tasks

- EVA uses baseline estimate of the schedule and the actual progress to date to determine the completeness status.

EVA technique

To determine the earned value, first compute the following:

- Budgeted cost of work scheduled (BCWS):
  - Sum of the cost budgeted for each work task scheduled for completion by a point in time

- Budget at completion (BAC):
  - Sum of BCWS values for all work tasks in a project

- Budgeted cost of work performed (BCWP):
  - Sum of the BCWS values for all work tasks that have actually been completed by a point in time on the project schedule

- Actual cost of work performed (ACWP):
  - Sum of the effort actually expended on work tasks that have been completed by a point in time on the project schedule

EVA technique

Given BCWS, BAC, BCWP, and ACWP values for a project, the following progress indicators can be computed:

- Schedule Performance Index (SPI) = BCWP / BCWS
- Schedule Variance (SV) = BCWP – BCWS
- Percent Scheduled for Completion (PSFC) = BCWS / BAC
- Percent Complete (PC) = BCWP / BAC
- Cost Performance Index (CPI) = BCWP / ACWP
- Cost Variance (CV) = BCWP – ACWP

EVA technique

Project Schedule and Budget can be assessed as:

- SPI > 1.0: ahead of schedule
- SPI < 1.0: behind schedule

- CPI > 1.0: under budget
- CPI < 1.0: over budget

- SV > 0: ahead of schedule
- SV < 0: behind schedule

- CV > 0: under budget
- CV < 0: over budget

EVA technique - Example

Assume that you are a software project manager and that you've been asked to compute earned value statistics for a small software project. The project has **56 planned work tasks** that are estimated to require **582 person-days to complete**. At the time that you've been asked to do the earned value analysis, **12 tasks have been completed**. However the project schedule indicates that **15 tasks should have been completed**.

Schedule data (in person-days) is on next slide

- Determine BAC, BCWS, BCWP, and ACWP from the data
- Compute the SPI, SV, PSFC, PC, CPI, and CV for the project

EVA Technique - Example

Task	Planned Effort	Actual Effort
1	12.0	12.5
2	15.0	11.0
3	13.0	17.0
4	8.0	9.5
5	9.5	9.0
6	18.0	19.0
7	10.0	10.0
8	4.0	4.5
9	12.0	10.0
10	6.0	6.5
11	5.0	4.0
12	14.0	14.5
13	16.0	-
14	6.0	-
15	8.0	-

EVA technique - Example

BAC = 582 person-days (from problem statement)

BCWS = Sum of all numbers in the “Planned Effort” column
BCWS = 156.5 person-days

BCWP = Sum of numbers in the “Planned Effort” column for the first 12 tasks that have been completed
BCWP = 126.5 person-days

ACWP = Sum of all numbers in the “Actual Effort” column
ACWP = 127.5 person-days

EVA technique - Example

SPI = BCWP / BCWS = 126.5 / 156.5 = **0.81**

SV = BCWP – BCWS = 126.5 – 156.5 = **-30 person-days**

PSFC = BCWS / BAC = 156.5 / 582 = 0.2689 = **26.9%**

PC = BCWP / BAC = 126.5 / 582 = 0.2173 = **21.73%**

CPI = BCWP / ACWP = 126.5 / 127.5 = **0.99**

CV = BCWP – ACWP = 126.5 – 127.5 = **-1 person-days**

Gantt Chart Example

Your next assignment will be to create a list of tasks for your term project

1) Enumerate the tasks you have completed **thus far**
    a) And associated time

2) Enumerate the tasks you have **remaining** to get a **minimum viable product**
    a) And associated time

3) Create either a Gantt chart or burn-down or burn-up chart
    a) The Agile methods don't strictly apply in our current format, however if you are tracking user stories over tasks then it may be more beneficial to you

I don't have MS Project

Nor do I want to sign my life away to LucidCharts, no matter how cute their snek vids are

Gantt in Google Sheets:
https://www.howtogeek.com/447783/how-to-create-a-gantt-chart-in-google-sheets/

Gantt in Draw.io:
(There is a template, though it just pre-populates a bunch of rectangles)

Burn-down / Burn-Up in Excel:
https://www.extendoffice.com/documents/excel/2446-excel-burndown-chart-or-burn-up-chart.html

Open-Source Tools!

Here's a list, I'm going to use GanttProject: https://www.goodfirms.co/blog/best-free-open-source-gantt-chart-software-solutions

There are some nifty web-based solutions out there, but no need to register for anything for this

What tasks have I done?

Artifacts:

- Proposal
- Use cases

Implementation:

- PyGame interface (1 week)
- Tilemap (1 day)
- Spritesheet (2 days)
- Initial Perlin noise map (1 week)
- Entity-Component-System framework (2 weeks)
- Tracery demos (2 weeks)

What tasks do I have remaining?

Artifacts:

- Whatever the professor doles out as homework (3 weeks)
- Software requirements spec (4 weeks)

Implementation

- Define Tracery rules (2 weeks)
  - Hook Tracery into world generation (1 week)
  - Hook Tracery into NPC conversations (1 week)
- Fix scrolling map problems (3 days)
- Implement player/entity interactions (2 weeks)
- Add randomization to noise function (1 week)
- Add towns (1 week)
- Add quests (2 weeks)