Manajemen Proyek SI

Oleh :

Ir. I Gede Made Karma, MT

Software Project Estimation[*)]

- Effective software project estimation is one of the most challenging and important activities in software development.
- Proper project planning and control is not possible without a sound and reliable estimate.

*) *Kathleen Peters*, Software Productivity Centre Inc. (SPC) in Vancouver, British Columbia, Canada & Simon Fraser University.

The four basic steps in software project estimation are:

1. Estimate the size of the development product. This generally ends up in either Lines of Code (LOC) or Function Points (FP).
2. Estimate the effort in person-months or person-hours.
3. Estimate the schedule in calendar months.
4. Estimate the project cost in dollars (or local currency)

Estimating size

- An accurate estimate of the size of the software to be built is the first step to an effective estimate.
- Your source(s) of information regarding the scope of the project should, wherever possible, start with formal descriptions of the requirements - for example, a customer's requirements specification or request for proposal, a system specification, a software requirements specification.
- If you are [re-]estimating a project in later phases of the project's lifecycle, design documents can be used to provide additional detail.
- In any case, you must communicate the level of risk and uncertainty in an estimate to all concerned and you must re-estimate the project as soon as more scope information is determined.

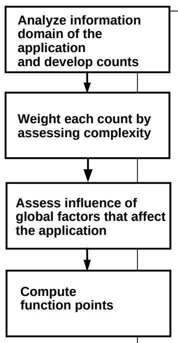
Estimating size (2)

1. By analogy.
   - Having done a similar project in the past and knowing its size, you estimate each major piece of the new project as a percentage of the size of a similar piece of the previous project.
   - Estimate the total size of the new project by adding up the estimated sizes of each of the pieces.
   - An experienced estimator can produce reasonably good size estimates by analogy if accurate size values are available for the previous project and if the new project is sufficiently similar to the previous one.

Estimating size (3)

2. By counting product features and using an algorithmic approach such as Function Points to convert the count into an estimate of size.
   - Macro-level "product features" may include the number of subsystems, classes/modules, methods/functions.
   - More detailed "product features" may include the number of screens, dialogs, files, database tables, reports, messages, and so on.

Computing Function Points


Analyze information domain of the application and develop counts
Weight each count by assessing complexity
Assess influence of global factors that affect the application
Compute function points

Establish *count* for input domain and system interfaces

Assign level of complexity or *weight* to each count

Grade significance of external factors, F_i such as reuse, concurrency, OS, ...

function points = $\sum$(count x weight) x C

where:

complexity multiplier: C = (0.65 + 0.01 x N)

degree of influence: N = $\sum F_i$

Analyzing the Information Domain

measurement parameter	count		weighting factor simple	avg.	complex		
number of user inputs	☐	X	3	4	6	=	☐
number of user outputs	☐	X	4	5	7	=	☐
number of user inquiries	☐	X	3	4	6	=	☐
number of files	☐	X	7	10	15	=	☐
number of ext.interfaces	☐	X	5	7	10	=	☐
count-total $\longrightarrow$							☐
complexity multiplier							☐
function points $\longrightarrow$							☐

Taking Complexity into Account

Factors are rated on a scale of 0 (not important) to 5 (very important):

data communications	on-line update
distributed functions	complex processing
heavily used configuration	installation ease
transaction rate	operational ease
on-line data entry	multiple sites
end user efficiency	facilitate change

Rough Estimates of LOC/FP

Programming Language	LOC/FP (Rata-rata)
Assembly	320
C	128
COBOL	106
FORTRAN	106
Pascal	90
C++	64
Ada95	53
Visual Basic	32
Smalltalk	22
Powerbuilder (code generator)	16
SQL	12

Estimating effort

- Once you have an estimate of the size of your product, you can derive the effort estimate.
- This conversion from software size to total project effort can only be done if you have a defined software development lifecycle and development process that you follow to specify, design, develop, and test the software.
- A software development project involves far more than simply coding the software – in fact, coding is often the smallest part of the overall effort.
- Writing and reviewing documentation, implementing prototypes, designing the deliverables, and reviewing and testing the code take up the larger portion of overall project effort.
- The project effort estimate requires you to identify and estimate, and then sum up all the activities you must perform to build a product of the estimated size.

Estimating effort (2)

1. The best way is to use your organization’s own historical data to determine how much effort previous projects of the estimated size have taken. This, of course, assumes
   a) your organization has been documenting actual results from previous projects,
   b) that you have at least one past project of similar size (it is even better if you have several projects of similar size as this reinforces that you consistently need a certain level of effort to develop projects of a given size), and
   c) that you will follow a similar development lifecycle, use a similar development methodology, use similar tools, and use a team with similar skills and experience for the new project.

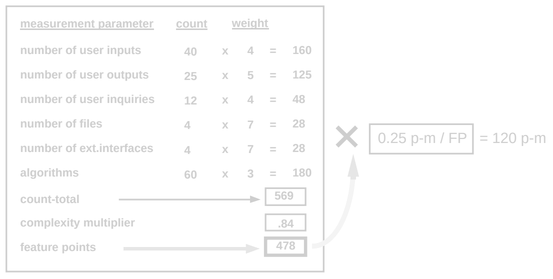
Estimating effort (3)

2. If you don’t have historical data from your own organization because you haven’t started collecting it yet or because your new project is very different in one or more key aspects,
   - you can use a mature and generally accepted algorithmic approach such as Barry Boehm’s COCOMO model or the Putnam Methodology to convert a size estimate into an effort estimate.
   - These models have been derived by studying a significant number of completed projects from various organizations to see how their project sizes mapped into total project effort.
   - These “industry data” models may not be as accurate as your own historical data, but they can give you useful ballpark effort estimates.

Example: LOC Approach

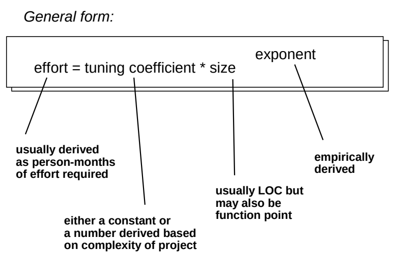
Functions	estimated LOC	LOC/pm	$/LOC	Cost	Effort (months)
UICF	2340	315	14	32,000	7.4
2DGA	5380	220	20	107,000	24.4
3DGA	6800	220	20	136,000	30.9
DSM	3350	240	18	60,000	13.9
CGDF	4950	200	22	109,000	24.7
PCF	2140	140	28	60,000	15.2
DAM	8400	300	18	151,000	28.0
Totals	33,360			655,000	145.0

Example: FP Approach



measurement parameter	count		weight		
number of user inputs	40	x	4	=	160
number of user outputs	25	x	5	=	125
number of user inquiries	12	x	4	=	48
number of files	4	x	7	=	28
number of ext.interfaces	4	x	7	=	28
algorithms	60	x	3	=	180
count-total					569
complexity multiplier					.84
feature points					478

Empirical Estimation Models


General form:
effort = tuning coefficient * size
exponent
usually derived as person-months of effort required
either a constant or a number derived based on complexity of project
usually LOC but may also be function point
empirically derived

Estimating schedule

- This generally involves estimating the number of people who will work on the project, what they will work on (the Work Breakdown Structure), when they will start working on the project and when they will finish (this is the “staffing profile”).
- Once you have this information, you need to lay it out into a calendar schedule.
- Again, historical data from your organization’s past projects or industry data models can be used to predict the number of people you will need for a project of a given size and how work can be broken down into a schedule.

Estimating schedule (2)

- If you have nothing else, a schedule estimation rule of thumb [McConnell 1996] can be used to get a rough idea of the total calendar time required:

  Schedule in months = 3.0 * (effort-months) $^{1/3}$
- Opinions vary as to whether 2.0 or 2.5 or even 4.0 should be used in place of the 3.0 value – only by trying it out will you see what works for you.

Estimating Cost

- There are many factors to consider when estimating the total cost of a project.
- These include labor, hardware and software purchases or rentals, travel for meeting or testing purposes, telecommunications (e.g., longdistance phone calls, video-conferences, dedicated lines for testing, etc.), training courses, office space, and so on.

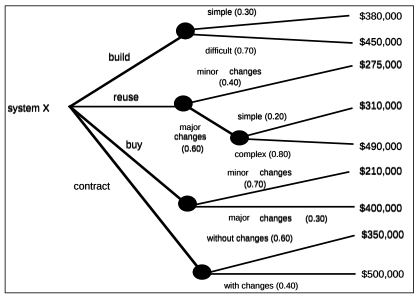
Estimating Cost (2)

- Exactly how you estimate total project cost will depend on how your organization allocates costs.
- Some costs may not be allocated to individual projects and may be taken care of by adding an overhead value to labor rates ($ per hour).
- Often, a software development project manager will only estimate the labor cost and identify any additional project costs not considered “overhead” by the organization.

Estimating Cost (3)

- The simplest labor cost can be obtained by multiplying the project’s effort estimate (in hours) by a general labor rate ($ per hour).
- A more accurate labor cost would result from using a specific labor rate for each staff position (e.g., Technical, QA, Project Management, Documentation, Support, etc.).
- You would have to determine what percentage of total project effort should be allocated to each position.
- Again, historical data or industry data models can help.

The Make-Buy Decision



Computing Expected Cost

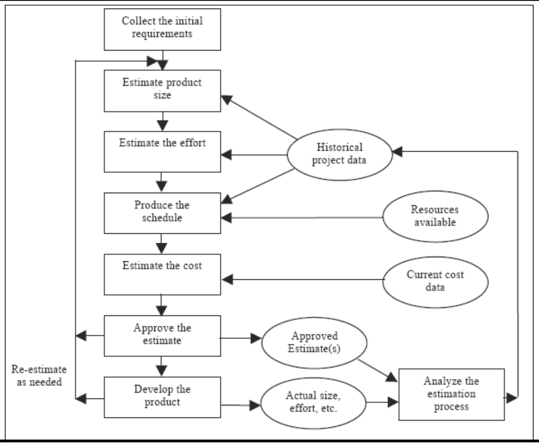
expected cost =
$\sum$(path probability)$_i$ x (estimated path cost)$_i$

For example, the expected cost to build is:

expected cost$_{build}$ = 0.30($380K)+0.70($450K)
= $429 K

similarly,

expected cost $_{reuse}$ = $382K
expected cost $_{buy}$ = $267K
expected cost $_{contr}$ = $410K



Working Backwards from Available Time

- Projects often have a delivery date specified for them that isn't negotiable –
  - "The new release has to be out in 6 months";
  - "The customer's new telephone switches go on-line in 12 months and our software has to be ready then".
- If you already know how much time you have, the only thing you can do is negotiate the set of functionality you can implement in the time available.
- Since there is always more to do than time available, functionality has to be prioritized and selected so that a cohesive package of software can be delivered on time.

Working Backwards … (2)

- Working backwards doesn't mean you skip any steps in the basic estimation process outlined above.
- You still need to size the product, although here you really do have to break it down into a number of pieces you can either select or remove from the deliverable, and you still need to estimate effort, schedule, and cost.
- This is where estimation tools can be really useful.
- Trying to fit a set of functionality into a fixed timeframe requires a number of "what if" scenarios to be generated.
- To do this manually would take too much time and effort. Some tools allow you to play with various options easily and quickly.

Understanding an Estimate's Accuracy

- Whenever an estimate is generated, everyone wants to know how close the numbers are to reality.
- The bottom line is that you won't know exactly until you finish the project – and you will have to live with some uncertainty.
- Naturally, you will want every estimate to be as accurate as possible given the data you have at the time you generate it.
- And of course you don't want to present an estimate in a way that inspires a false sense of confidence in the numbers.

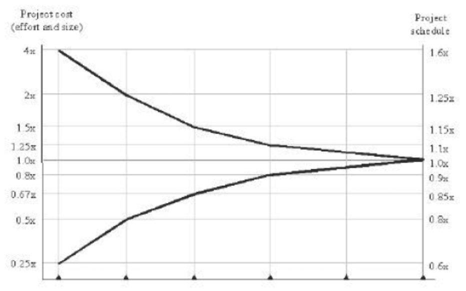
Understanding an … (2)

- What do we mean by an "accurate" estimate? Accuracy is an indication of how close something is to reality. Precision is an indication of how finely something is measured.
- For example, a size estimate of 70 to 80 KLOC might be both the most accurate and the most precise estimate you can make at the end of the requirements specification phase of a project.
- If you simplify your size estimate to 75000 LOC it looks more precise, but in reality it's less accurate.
- If you offer the size estimate as 75281 LOC, it is precise to one LOC but it can only be measured that accurately once the coding phase of the project is completed and an actual LOC count is done.

Understanding an … (3)

- If your accurate size estimate is a range, rather than a single value, then all values calculated from it (e.g., effort, schedule, cost) should be represented as a range as well.
- If, over the lifetime of a project, you make several estimates as you specify the product in more detail, the range should narrow and your estimate should approach what will eventually be the actual cost values for the product or system you are developing.

Understanding an … (4)



Understanding an … (5)

- Of course, you must also keep in mind other important factors that affect the accuracy of your estimates, such as:
  - the accuracy of all the estimate's input data (the old adage, "Garbage in, Garbage out", holds true)
  - the accuracy of any estimate calculations (e.g., converting between Function Points and LOC has a certain margin of error)
  - how closely the historical data or industry data used to calibrate the model matches the project you are estimating
  - the predictability of your organization's software development process, and
  - whether or not the actual project was carefully planned, monitored and controlled, and no major surprises occurred that caused unexpected delays.

Understanding the Tradeoffs

- Once you've generated a project estimate, the real work begins – finding some combination of functionality, schedule, cost and staff size that can be accepted by management and customers!
- This is where a solid understanding of the relationships between these variables is so important, and where being armed with different project estimates illustrating the tradeoffs is very useful for establishing what the limits are.

Understanding the Tradeoffs (2)

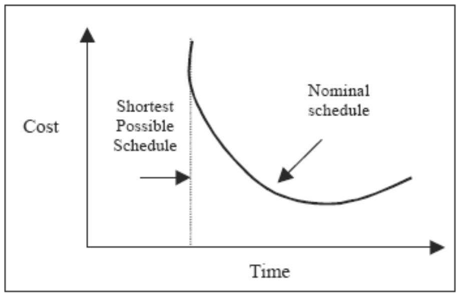
Here are a few facts of life you need to remember during the estimate "adjustment" phase:

1. If you lengthen the schedule, you can generally reduce the overall cost and use fewer people.
   - Sometimes you only have to lengthen the schedule by a few weeks to get a benefit.
   - Usually management and customers don't want a long delay, but see how much "extra" might be acceptable.
   - Many people don't consider generating an estimate option that lengthens the schedule to see what effect it has unless they are driven to it in an attempt to reduce cost or staff size.

Understanding the Tradeoffs (3)

2. You can only shorten a schedule three ways.
   - You can reduce the functionality (reducing the effort by doing less),
   - increase the number of concurrent staff (but only if there are tasks you could now do in parallel to take advantage of this!), or
   - keep the number of staff constant but get them to work overtime.

Understanding the Tradeoffs (4)



Understanding the Tradeoffs (5)

3. There is a shortest possible schedule for any project and you have to know what it is.
   - You can only shrink
     - a schedule so far given the functionality you are required to implement,
     - the minimum process you have to follow to develop and test it, and
     - the minimum level of quality you want in the output.
   - Don't even think of trying to beat that limit!

Understanding the Tradeoffs (6)

4. The shortest possible schedule may not be achievable by you.
   - To achieve the shortest possible
     - schedule your project team had better all be highly skilled and experienced,
     - your development process had better be well defined and mature, and
     - the project itself has to go perfectly.
   - There are not many organizations that can hope to make the shortest possible schedule, so it's better not to aim for this.
   - Instead, you need to determine what your shortest achievable schedule is(also known as the "nominal" schedule).
   - Data from past projects is your best source of information here.

Understanding the Tradeoffs (7)

5. Always keep in mind the accuracy of the estimate you are attempting to adjust.
   - If your schedule estimate is currently "5 to 7 months" then a small change, for example 2 weeks, either way doesn't mean much yet.
   - You can only adjust the schedule in increments that have some significance given the accuracy of the estimate.

The Trouble with Estimates

- Estimating size is the most difficult (but not impossible) step intellectually, and is often skipped in favor of going directly to estimating a schedule.
- Customers and software developers often don't really recognize that software development is a process of gradual refinement and that estimates made early in a project lifecycle are "fuzzy".
- Organizations often don't collect and analyze historical data on their performance on development projects.
- It is often difficult to get a realistic schedule accepted by management and customers.

Maintenance & Enhancement Projects vs. New Development

- When sizing new development for a maintenance project you have to keep in mind that inserting this new functionality will only be feasible if the product's existing architecture can accommodate it. If it cannot, the maintenance effort must be increased to rework the architecture.
- It's tricky to attempt to size adaptation work in the same manner as new work. An experienced individual estimating maintenance effort by analogy is a more common approach than attempting to size adaptation work in LOC or Function Points and then converting size to effort .

Maintenance & … (2)

- Estimation models that are calibrated to produce effort and schedule estimates for new development projects assume everything is created from scratch. This isn't the case for maintenance projects where you are modifying a certain amount of existing documentation, code, test cases, etc. Using these models may tend to over-estimate maintenance projects.
- Often, maintenance work has fixed delivery dates (e.g., a maintenance release every 6 months or once a year) and is done by a fixed number of people (i.e., an allocated maintenance team) so estimates have to deal with fitting work into a fixed timeframe with a constant staffing level.

Estimating Small Projects

- Many people work on small projects, which are generally defined as a staff size of one or two people and a schedule of less than six months.
  - Existing industry-data project estimation models are not calibrated from small projects and so are of little or no use here unless they can be adequately adjusted using an organization's small project historical data.
- Estimates for small projects are highly dependent on the capabilities of the individual(s) performing the work and so are best estimated directly by those assigned to do the work.
  - An approach such as Watts Humphrey's Personal Software Process (PSP) [Humphrey 1995] is much more applicable for small project estimation.

Estimating a "New Domain" Project

- How do you estimate a project in a new application domain where no one in your organization has any previous experience?
  - If it's a leading-edge (or "bleeding-edge"!) project, no one else has any previous experience either.
  - The first time you do something you are dealing with much more uncertainty and there is no way out of it except to proceed with caution and manage the project carefully.
  - These projects are always high risk, and are generally under-estimated regardless of the estimation process used [Vigder 1994].
  - Knowing these two facts, you must
    a. make the risks very clear to management and customers,
    b. avoid making major commitments to fixed deadlines, and
    c. re-estimate as you become more familiar with the domain and as you specify the product in more detail.

Estimating a … (2)

- Selecting a project lifecycle which best accommodates the uncertainty of new-domain projects is often a key step that is missing from the development process.
  - An iterative life cycle such as the Incremental Release Model where delivery is done in pieces, or the Spiral Model where revisiting estimates and risk assessment is done before proceeding into each new step, are often better approaches than the more traditional Waterfall Model.

Some Estimating Tips

- Allow enough time to do a proper project estimate – rushed estimates are inaccurate, high-risk estimates! For large development projects, the estimation step should really be regarded as a miniproject.
- Where possible, use documented data from your organization's own similar past projects. It will result in the most accurate estimate. If your organization has not kept historical data, now is a good time to start collecting it.
- Use developer-based estimates. Estimates prepared by people other than those who will do the work will be less accurate.

Some Estimating Tips (2)

- Use at least one software estimation tool. Estimation tools implement complex models that would take significant time to learn to apply manually. They also make sure you don't forget anything, and allow you to tune an estimate quickly and relatively painlessly.
- Use several different people to estimate and use several different estimation techniques (using an estimation tool should be considered as one of the techniques), and compare the results.

Some Estimating Tips (3)

- Re-estimate the project several times throughout its lifecycle. As you specify the product in more detail, your estimates should begin to approach what you will actually use to complete the project.
- Create a standardized estimation procedure that all involved can and do buy into. That way you can't argue about the outputs, only the inputs, and your effort is spent productively understanding the scope and cost drivers for the project.
- Focus some effort on improving your organization's software project estimation process. As each project is completed, compare actuals to estimates – how well did you do in predicting the project's effort and schedule? What did you miss? Where could you improve?