Estimating With Objects - Part IX

Contents

This column is the ninth in a series about estimating. After this there is only one more column on estimating methods. There is then one further column in this series on the results engineers have obtained by using the PROBE method in practice. After that, you will know how object-oriented methods can help you make good estimates and plans for your software work.

If you are new to this series of columns on Estimating with Objects, the first was in the July 1996 Object Currents issue. The prior columns in this series gave an overview of estimating and defined the steps needed to make size and resource estimates. If you have not read these earlier columns, you should look at them first to understand the context for this discussion and to see how these various estimating topics relate. To repeat what I have said in previous columns, the estimating method described here is called PROBE. If you want to quickly learn more about PROBE, you should read my book A Discipline for Software Engineering, from Addison Wesley. This book introduces the Personal Software Process (PSP)^{SM}, which is an orderly and defined way for software engineers to do their work.

This month's column continues the discussion of how object-oriented techniques can help you to estimate and plan your work. To make a project plan, you need a resource estimate and, to estimate resources, you need to estimate the size of the product you plan to build. Also, to make good estimates, you need historical data on the sizes and development times for the programs you have previously written. The previous columns described how to gather these data and how to use them to make size and resource estimates. In this column, we describe how to produce a project schedule from these data. While the steps are not complex, they involve some subtle considerations. This column reviews the issues involved in making schedules, discusses how to address these issues, and outlines a helpful schedule estimating procedure.

Product and period planning

The reason schedules are hard to produce is that they marry two different kinds of planning. Product planning concerns building products. Product planning issues are largely driven by such product-related activities as defining requirements, producing the design, and testing. The way you do these tasks is determined by your background, the processes, tools, and methods you use, and the technical needs of the project. The other kind of planning is period planning. Here you plan activities for a calendar period. An example would be your daily meeting calendar or your weekly schedule of workdays, weekends, holidays, etc.

When you produce a schedule, you bridge these two planning worlds. While doing this is simple in concept, it is not necessarily easy. The reason is that you work on products, but you live in periods. Thus, the project schedule must address product needs but conform to the realities of your calendar.

Figure 1 shows how product and period planning are related in a business. Engineering develops products for release to manufacturing, marketing, and the customer. Engineering also sends product plans to the finance department which produces price and forecast information for manufacturing and marketing. Finance also generates cost and revenue projections for corporate. Corporate management then decides what dividends to pay the investors and how much they can spend on marketing, manufacturing, and new product development. They then provide the needed period funding.

Period planning is important to you. You get paid in periods and management needs period cash flow to get the money to pay you. If your plans or your department's plans are not accurate, finance's projections will also likely be wrong, and corporate may not have the cash when they need it. This could mean that there won't be enough money to give you a raise, or maybe not even enough money to pay you at all. You could then be laid off. Without sound engineering plans, businesses cannot make sound financial projections. And without sound financial projections, there may not be enough money to run the business. This is often why businesses fail. So accurate product and period planning are important to your business, and they are important to you.

Producing a project schedule

A key part of making good plans is making good schedules. To make a schedule, start with a list of the tasks to be done and the hours each task will likely take. From the previous eight columns, you know how to estimate the time required to do each of the project's tasks. To be most useful for scheduling purposes, however, you need these times at a fairly detailed level. For most jobs, I suggest you estimate down to tasks of about 10 hours. More detail generally means more accuracy. As we will see next month, you will also need this detail to accurately track project status.

Another important piece of information is the hours available for doing the work. When they first consider this issue, engineers often dismiss it as trivial. If their organization works a standard 40 hour week, they assume they will have 40 hours available to work on the project. This turns out to be a serious mistake.

Available working hours

If you gather data on how you spend time, you will probably find that you only spend about half of the official working week actually doing project tasks. The rest of the time you probably answer mail, attend meetings, solve support system problems, get supplies, or consult with other engineers. One engineer told me she had 5 hours of interruptions while doing a task that was supposed to take only a little over 2 hours.

While the numbers will vary by organization, and they will also differ enormously by individual, my experience is that few engineers consistently spend much more than 50% of their time on direct project work. It is true that when engineers get into a schedule crunch, they often work late at night, stop going to meetings, and even skip answering the mail or chatting with friends. While this is a common way to work in software organizations, nobody should plan to work this way all the time. Software work has enough crises without building them into our plans.

Also, don't assume that a new project will be staffed instantly at the start. And don't even take management literally when they tell you they will assign all the needed engineers on day one. It never happens. Ask which engineers will be assigned, check their availability, and find out the hours they each expect to have available each week. Then you can make a reasonable plan. Remember that understaffing is the single biggest reason projects are late. When they start with an unrealistic staffing plan, they are immediately behind schedule. Unless they later overstaff, such projects rarely catch up.

Now that you know the time each task will take and the available weekly hours by engineer, you only need one additional piece of information to lay out the schedule. This is the order in which the tasks will be done. While this information need not be precisely correct, you need a reasonable projection. We will talk next month about the problems of inaccurate task order plans.

The planning templates

With all this data, you can now lay out the schedule. The mechanics for doing this, however, can be a little tricky. The first step is to lay out a task list. For this we use a Task Planning Template, as shown in Table 1. (For more information on these templates, see Chapter 6 in my book, A Discipline for Software Engineering.)

Table 1. An Example Task Planning Template

Task	**Task Name**	**Task Hours**		**Planned**
Number		**Per Unit**	**Cumulative**	**Week**
1	Requirements	140	140	
2	Requirements Inspection	22	162	
3	Overall Design	170	332	
4	Design C1	16	348	

5	Design C2	12	360	
6	Design C3	22	382	
7	C1 Design Inspection	4	386	
8	C2 Design Inspection	4	390	
9	C3 Design Inspection	6	396	
10	Implement C1	90	486	
11	Implement C2	76	562	
12	Implement C3	110	672	
13	C1 Code Inspection	8	680	
14	C2 Code Inspection	10	690	
15	C3 Code Inspection	16	706	
16	Test C1	36	742	
17	Test C2	66	808	
18	Test C3	68	876	
19	System Test	310	1186	
	Total	1186		

On the Task Planning Template, list each task in the order you expect to do it. Also list the hours each task is expected to take and calculate the cumulative hours for each of the tasks. Table 1 shows an example of these data for a hypothetical project. Here, for example, 140 hours were estimated for defining the requirements and another 22 hours for inspecting and fixing any requirements problems. Overall design was planned for 170 hours followed by developing the three product components (C1, C2, and C3). The cumulative hours column is the cumulative sum of the hours for the tasks so far. For example, the cumulative time for Overall Design (task 3) is the sum of the times for tasks 1, 2, and 3, or 140+22+170=332 hours.

After calculating the cumulative hours, stop work on the Task Planning Template and move to the Schedule Planning Template shown in Table 2. Here, lay out the available hours for each week, and the cumulative total hours available for all the weeks the project is likely to take.

Table 2 shows the planned weekly hours for the 3 engineers on this hypothetical project. Note that instead of the same numbers every week, there will often be planned trips, holidays, courses, meetings, or other activities that you should consider in estimating weekly hours.

Table 2. An Example Schedule Planning Template

Week	**Hours Planned per Engineer**			**Total Hours**	
Number	**A**	**B**	**C**	**Weekly**	**Cumulative**
1	20	25	18	63	63
2	20	25	18	63	126
3	20	25	18	63	189
4	20	25	18	63	252
5	20	25	18	63	315
6	20	25	18	63	378
7	20	25	18	63	441
8	20	25	18	63	504
9	20	25	18	63	567
10	20	25	18	63	630
11	20	25	18	63	693
12	20	25	18	63	756
13	20	25	18	63	819
14	20	25	18	63	882
15	20	25	18	63	945
16	20	25	18	63	1008
17	20	25	18	63	1071
18	20	25	18	63	1134
19	20	25	18	63	1197
20	20	25	18	63	1260
Total	400	500	360	1260	

Now, with the total and cumulative weekly hours go back to the Task Planning Template. Note for each task when the cumulative task hours are met or exceeded by the cumulative hours on the Schedule Planing Template. The week with these cumulative hours is the week you can expect to finish that task. Enter the week number this occurs on the Task Planning Template.

Table 3 shows the completed Task Planning Template with the week numbers for each task. Here, for example, the date for completing the requirements would be the week when the total cumulative hours exceed 140 hours. From the Schedule Planning Template, you can see that 189 hours in week 3 is the first time the cumulative weekly scheduled hours exceed 140. You thus expect to finish the requirements during week 3. Similarly, Design C3 (task 6) should be completed when the cumulative hours on the Schedule Planning Template exceed the cumulative 382 hours for task 3. This is week 7 on the Schedule Planning Template.

Table 3. Completed Task Planning Template

Task	**Task Name**	**Task Hours**		**Planned**
Number		**Per Unit**	**Cumulative**	**Week**
1	Requirements	140	140	3
2	Requirements Inspection	22	162	3
3	Overall Design	170	332	6
4	Design C1	16	348	6
5	Design C2	12	360	6
6	Design C3	22	382	7
7	Design Inspection C1	4	386	7
8	Design Inspection C2	4	390	7
9	Design Inspection C3	6	396	7
10	Implement C1	90	486	8
11	Implement C2	76	562	9
12	Implement C3	110	672	11
13	Code Inspect C1	8	680	11
14	Code Inspect C2	10	690	11
15	Code Inspect C3	16	706	12
16	Test C1	36	742	12
17	Test C2	66	808	13
18	Test C3	68	876	14
19	System Test	310	1186	19

	Total	1186		

Note that by the end of week 7, the team has actually spent 441 hours, so they will probably finish Design C3 well before the end of the week. If you wanted to know the precise day, you could lay out the schedule planning template by day. Assuming you also had the cumulative hours for every day, you could see precisely which day this task was supposed to be completed.

Once you have a date for each task, you have the schedule. You now have all the information needed to produce the project schedule in any desired format.

Some complications

While this scheduling method is simple in concept, there are always complications in the real world. Examples are the following:

1. An hour isn't always an hour. When you try to make up for lost time by working overtime, you will probably be tired, less efficient, and more likely to make costly and time-consuming mistakes.
2. When engineers start on a new project, they often must phase in gradually while they wrap up prior work. This is generally no problem unless the prior project gets into trouble, but then it will take priority, costing your new project a lot of time.
3. When adding new people to a project, it usually takes time before they are fully productive.
4. Tasks have interdependencies that must be considered. Often, in fact, unanticipated interdependencies can cause serious delays.
5. There are also often external dependencies to systems, hardware, or other software groups that may cause problems.
6. Until you have data on the times tasks actually take, most of your estimates are likely to be low.

While these complications can be troublesome, the PROBE estimating and scheduling method provides an orderly framework for addressing them.

Next month's topics

Next month we will talk about how to track progress against the schedule. We will also describe how to determine project status. This is done with something called earned value tracking. Earned value provides a way to track project progress against a schedule, even when the task order changes from the plan. It also provides a convenient and surprisingly accurate way to forecast when you will likely finish the job. After next month, there is only one column left in this series on Estimating with Objects. That final column will discuss the experiences of engineers who have used the PROBE method.

Watts S. Humphrey
watts@sei.cmu.edu