ECE155: Engineering Design with Embedded Systems Winter 2013

Lecture 29 — March 22, 2013

Patrick Lam *version 1*

Objectives

After this lecture, you will be able to:

- distinguish between planning, scheduling and estimation;
- identify the parts of a vision and scope document and a project plan;
- participate in estimation processes and understand their components.

Planning and Estimation

Planning is notoriously difficult to get right for software projects, because estimates are always wrong. We are going to look at techniques which may help you better estimate, and hence plan, your software projects.

Estimation. An *estimate* is an informed guess, based on facts, about the amount of resources (primarily effort/time) that a task will require to complete.

Planning. When given a project to accomplish, the first thing to figure out is *how* to do that project. We'll call this *planning*—dividing the project into a set of small, related tasks. An important part of planning is identifying the dependencies between the tasks, so that we can properly order them and avoid unnecessary dependency-related delays.

What's an example of a plan?

Scheduling. This is a more mechanical process. There can be many valid ways to carry out a plan. When *scheduling*, we propose a particular order in which to carry out the plan. The inputs of scheduling are a set of tasks, their dependencies, and estimates of the amount of effort required. The output is a set of start and end times for each subtask.

Recap. Planning: figuring out *how* to do a task, in terms of breaking it down into subtasks. Scheduling: figuring out *when* to do the subtasks.

Planning

A first, pre-planning, step in preparing for a project is to figure out what the project entails. We will talk about the vision and scope document, and then the different parts of the project plan.

Vision and Scope Document. Writing this document helps you make sure that all of the stakeholders, or people who need to care about the project, are on board with the proposed project. You can think of this as preliminary requirements gathering. We'll talk about requirements in more detail later. The goal is to communicate with the stakeholders and, at a high level, get them to sign off on what the project will and won't include.

A *vision and scope document* summarizes the design problem, the stakeholders, the users, the risks, the assumptions, and the desired features of a solution. In particular, it:

- identifies the features that will be included;
- discusses features that stakeholders might expect but aren't slated for inclusion (in this iteration);
- summarizes agreed-upon expectations about the project's scope.

Project proposals are similar to vision and scope documents.

Here is a potential outline and example points for a vision and scope document:

- Problem Statement: The project will develop a research prototype for a better mousetrap.
- Project Background: [Some paragraphs briefly discussing the current state of the art in mousetrap design.]
- Stakeholders: The ACME Corporation eventually aims to sell these mousetraps to end users.
- Users: End users will deploy the mousetraps in mouse-infested locations.
- Risks: Mousetrap technology is a mature area, and many inventors have already attempted to develop better mousetraps, so this project may not successfully invent a better mousetrap.
- Assumptions: The engineers have access to modern CAD software, a machine shop, and commonplace electronic components, for building their better mousetrap. The engineers' toy mice accurately represent the behaviour of actual real-world mice.
- Vision of the Solution: The better mousetrap will use modern embedded systems technology, including sensors and actuators, to effectively detect and trap mice without harming them.
- List of features: The mousetrap will be able to detect and capture mice. Users will be able to calibrate the mousetrap for their intended deployment area. The mousetrap will include a provision for including mousebait.
- Scope of phased release: The first version of the mousetrap will detect mice, but will not capture the mice. The second version will detect and capture mice.
- Features not to be developed: It remains the responsibility of the end-user to relocate the mice that this mousetrap catches.

Project Plan. OK, so you have your vision and scope document. Now you can start figuring out what you're going to need to do to carry out the vision in the vision and scope document.

The *project plan* enumerates the work to be done on the project, as well as who is to do the work. It has the following components:

- Statement of Work: describes all work projects to be produced, and identifies the people who will do work.
- Resource List: contains a list of all resources needed for the project and summarizes resource availability.
- Work Breakdown Structure (WBS): describes all project activities.
- Project Schedule: estimates start and end times for project activities.
- Risk Plan: describes anticipated risks, likelihood of their occurence, and how to mitigate them.

Statement of Work

A *statement of work* (SOW) describes all work products to be produced and identifies who will do the work.

This statement includes:

- a list of all project features that will be developed;
- a description of each deliverable that will be developed (at this stage, one paragraph per deliverable); and
- the estimated effort required for each deliverable, if known.

Resource List

A *resource list* enumerates all resources required for the project and summarizes the availability of the resources.

A *resource* is a person, a hardware component, a software licence, a room, or anything else that is necessary for the project but limited in availability.

The resource list is often maintained in a spreadsheet or a database, and contains records consisting of:

- resource name;
- a brief (one-line) description of the resource;
- a summary of resource availability (start dates, end dates, etc.); and
- the cost of the resource (if applicable).

Work Breakdown Structure (WBS)

A *work breakdown structure* (WBS) describes all project activities. That is, it comprehensively lists all project activities required to complete the project, including intermediate deliverables.

The WBS needs to evolve as the project becomes better-defined and more details about required project activities become known.

Revision control (like Subversion) is also appropriate for non-software products, like WBSes, which are expected to evolve over time. Future project planning may benefit from studying the revision history of similar projects which happened in the past.

The *Software Project Management* textbook describes the "Wideband Delphi Process" for estimating the time required for each activity in a WBS.

Project Schedule

A *project schedule* estimates start and end times of all project activities. These estimates should be developed using a repeatable, defensible process.

One way of getting estimates is by running meetings where participants defend their estimates.

1. Each participant writes down an initial estimate, along with assumptions, for each project activity.
2. A moderator collects all estimates and summarizes them.
3. The participants discuss the estimates.
4. Repeat as needed.

Documenting assumptions helps refine estimates and reduce variance in estimates. Although the participants may not reach consensus, the resulting estimates should at least not differ too radically.

Risk Plan

Recall that a risk is a potential threat to the successful completion of your project. A *risk plan* therefore describes anticipated risks, the likelihood of their occurrence, and how they might be handled. Good risk plans are comprehensive and include potential ways to mitigate significant risks.

The risk plan is usually developed in a risk assessment and planning session:

- Team members brainstorm to create a list of potential risks that threaten the project.
- For each risk, team members estimate the likelihood of the risk and its potential impact.
- Taking likelihood and impact into account, team members develop a plan for mitigating each risk.

Risk Plan Example. See page 29 of *Applied Software Project Management* for a sample risk plan.

Estimates

Project plans include project schedules, which estimate how long everything takes. Estimates are notorious for being wrong (and not in the good way). However, this may help you come up with accurate time estimates:

- An accurate work breakdown structure for the project.
- An effort estimate for each task listed in the work breakdown structure.
- A list of the assumptions made during the establishment of the effort estimates.
- Consensus among the members of the project team that effort estimates are reasonable.

On Assumptions. Do assumptions make estimates less accurate? Not necessarily! Unstated assumptions, however, are troublesome.

If an assumption is well-documented and properly communicated, it can improve the quality of an estimate:

- it may reveal an inobvious problem or detail;
- it may reveal a simplification that some, but not all, members of the estimation team were relying on; and,
- finding out that an assumption is untrue is a good time to revisit an estimate.

Agreeing on a common set of assumptions allows the estimation team to move beyond the assumptions and come up with good estimates.

Padding Estimates. ("Scotty Time[1]"). Padding an estimate seems like all win: underpromise and overdeliver, etc. This is, however, not necessarily a good idea. Let's talk about the benefits and costs of this approach.

Benefits:

- You look like a hero by doing better than you said you would.
- Working based on padded time estimates helps you alleviate time pressures.

Problems:

- You can only look like a hero a couple of times before you look like a zero and managers adapt to your padded estimates. (Managers aren't dumb.)
- Managers don't trust engineers who habitually pad time estimates.
- Clients or managers may decide to not proceed based on the too-expensive padded estimate.

[1] `http://tvtropes.org/pmwiki/pmwiki.php/Main/ScottyTime`

Estimation Techniques

We are going to talk about a number of estimation techniques, including the Wideband Delphi Process, PROBE, COCOMO II, and The Planning Game.

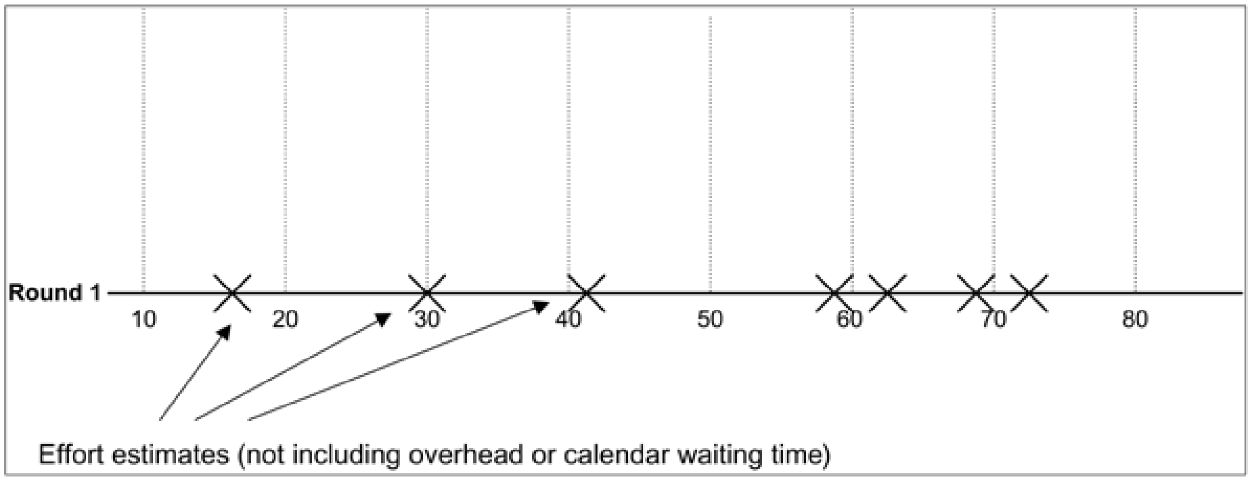
Wideband Delphi Process

We mentioned this process last time. This repeatable process consists of a set of straightforward steps:

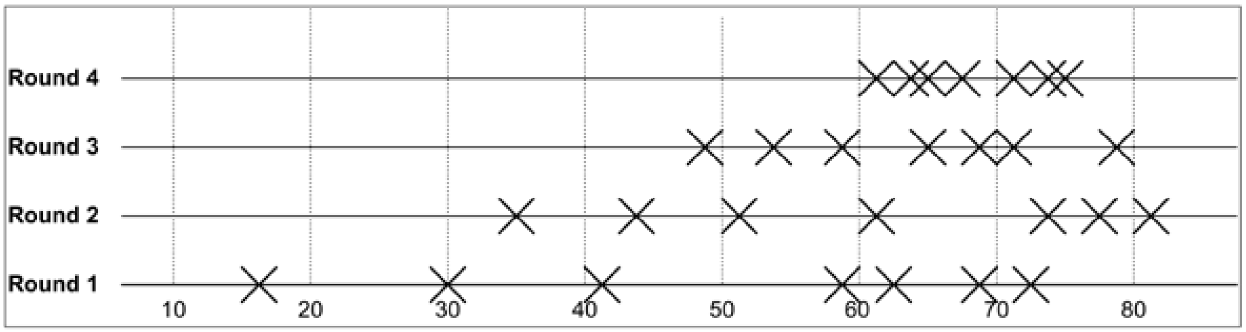
- Choose an estimation team and moderator.
- Hold a kickoff meeting to develop a work breakdown structure, a preliminary list of assumptions, and a unit of estimation.
- Allow estimation team members to independently prepare estimates for each task and prepare a list of additional assumptions.
- Hold an estimation session, where the moderator requests estimates from team members, and attempts to achieve consensus on estimates by having members explain their estimates.
- Assemble the final list of tasks and estimates.
- Review the results with the project manager and the estimation team.

Note that estimation sessions are key to the Wideband Delphi Process.

Starting an Estimation Session. First, team members share effort estimates for each task in the work breakdown structure. The moderator records these estimates and plots them on a graph or records them in a table. One might expect the range of these estimates to be large, because team members may make different assumptions. See below:



Achieving consensus. The point of working in a team is to come up with a consensus which is, ideally, better than the results that individual members would have come up with. The Wideband Delphi Process uses a number of rounds of discussion in the estimation session. In each round, team members discuss assumptions, clarify project details, and revise estimates. Here's an example of convergence in estimates:



Other Estimation Techniques

There are a number of estimation techniques besides Wideband Delphi. Here are some well-known estimation techniques for software.

PROBE. It's sensible to believe that if you're doing something similar to something that you've done in the past, it should take about the same amount of time. So, the idea here is to obsessively track how long it took to do things in the past, and then use linear regression to estimate how long it'll take in the future.

What are some potential problems with this technique?

COCOMO II. The idea here is to feed in a number of guesses about your project's scope, apply a formula full of fudge factors (developed based on empirical data), and get an estimate of size and effort. Examples of guesses include memory constraints, analyst capability, and product complexity. COCOMO stands for *COnstructive COst MOdel.*

The Planning Game

This isn't one of those fun games like Mass Effect. Instead, it's an estimation technique developed by Kent Beck (inventor of extreme programming) while working at Chrysler in the 1990's.

The *Planning Game* is a planning process that 1) identifies the scope of the project as well as 2) the tasks required to complete the project. It also 3) estimates the effort required for these tasks. This game consists of two phases: release planning and iteration planning. Release planning plans the scope of the project, while iteration planning plans the activities and tasks of the developers. The Planning game requires a team consisting of customers and developers.

Release Planning. Each week, the team meets to plan the immediate future of the project. The team writes user stories describing project requirements on index cards, and assigns effort estimates for the stories (e.g., 1, 2, or 3 weeks). The team prioritizes the requirements.

Iteration Planning. The team divides requirements into sets of tasks to be completed. It then assigns these tasks to developers, and estimates effort for these tasks. Developers complete these tasks and match them back up with the user stories.