How to improve the quality of your system(s)

Perdita Stevens

School of Informatics
University of Edinburgh

Prior question

Why improve a system (let alone all your systems?)

Prior question

Why improve a system (let alone all your systems?)

To manage (reduce, predict and plan for) risk that bad things will happen to you.

Risk management

- Analyse risks during the (pre-)planning stage of a project, so you can
- Manage them when they happen during the project

Roughly, risks relate to the project, the product, or the business.

Examples of typical risks: staff loss, management change, missing equipment, changing requirements, delays in requirements analysis, mis-estimation of cost, competitor gets to market first.

Planning for risks

Identify the risks early, in various possible categories.

Analyse each identified risk. Is it minor, major, serious, fatal? What is the chance of it happening?

Plan how to cope with each risk. Can it be avoided by reducing the probability of occurrence? Can you plan to minimise the effect if it does happen? What is your contingency plan if it does happen?

Exercise: Plan for:

- The company makes a loss and has to cut R&D.
- The lead designer on the human interface team leaves.
- The customer changes the requirements to adapt to a new company acquisition.

Software quality

What is it?

Software quality

What is it?

Ultimately, anything that the customer cares about.

Approaches to improving quality

May focus on

1. the software product itself
2. the process by which the software is produced

Verification, validation, testing, code/design reviews, inspections, walkthroughs are product-focused approaches to quality improvement.

Product focus

We have already discussed VV&T.

One slide on code/design reviews/inspections/walkthroughs (on which one could have a whole course...)

Reviews, key points

A review is a meeting of a few people,

which reviews one specific artefact (e.g., design document, or defined body of code)

for which specific entry criteria have been passed (e.g., the code compiles).

Participants study the artefact before the meeting.

Someone, usually the main author of the artefact, presents it and answers questions.

The meeting does not try to fix problems, just identify them.

The meeting has a fixed time limit.

Process focus

Advantages:

- potential to improve *all* products
- possibility of certifying a whole organisation
- some important things, especially planning (and thus time-to-market, cost etc.), are hard to approach in any other way.

Disadvantages:

- done badly, can easily prove very costly with low benefits: easy to spend time and money without improving *any* product

Centres of process-focused QA

Possibly directions of influence, each with own philosophy:

Organisation $\longleftrightarrow$ Project $\longleftrightarrow$ Individual

1. Organisation's management decrees, influencing projects whose managers direct individuals into desired behaviour. (Underlying philosphy: people try to get out of work and need to be controlled.)
2. Individuals introduce improvements which are rolled up and out to the rest of the project and the rest of the organisation. (Underlying philosophy: people want to do good work and need their efforts to be noticed and enabled.)

Areas and terminology

- Quality planning – how will you ensure that this project delivers a high quality product?
- Quality metrics – what measurements must you make in order to tell whether what you're doing is making the difference you intend?
- Quality improvement – what can you learn from this project to help you plan and run the next one better?
- Quality control – how can you ensure and prove that your quality plan was followed?
- Quality assurance – an umbrella term for the whole field.

Standard QA models

We'll look briefly at two examples with different aims:

- CMMI, the Capability Maturity Model Integration (from Carnegie Mellon's Software Engineering Institute), a development of the very popular Capability Maturity Model (CMM).
  Crucial idea here is that maturity increases: quality planning, control and *improvement* framework.
- ISO9000
  Less emphasis on improvement: quality *control* framework.

Standards

There are many types of standard. Coding standards tell you how to format code. Documentation standards tell you how to organize documentation. Quality standards tell you how to . . . organize quality control.

ISO 9001 (alias BS5750) is an international standard for quality assurance. It specifies how to specify documents and procedures that a company should follow in its quality control. It *does not* specify or require any level of product quality.

CMMI

The Integrated Capability Maturity Model is a successor to the influential CMM. It provides a (generic, specializable to software) description of process areas, goals associated with each area, and practices that may achieve goals. Organizations are assessed at a maturity level according to *how* they achieve goals and follow practices. Levels are:

1. Initial: goals may be met, but by heroics
2. Managed: documented plans, resource management, monitoring, etc.
3. Defined: organization defines process framework. Measurements collected for use in improvement.
4. Quantitatively managed: use measurement and statistical etc. methods during process.
5. Optimizing: process improvement happens, driven by quantitative measures.

Total Quality Management

Plan, Do, Check, Act

TQM embodies the idea that improving quality is everyone's job – not just that of the QA department.

Stands in relation to ISO9001 and CMM(I) rather as XP does to UP – an attempt to emphasise basics and reduce bureaucracy.

Don't be seduced by Methodology

There is a big difference between *methodology* and *Methodology*. A *methodology* is a basic approach one takes to get the job done, consisting of:

- a tailored plan (specific to the work at hand)
- a body of skills necessary to effect the plan

A *Methodology* is an attempt to centralize thinking. All meaningful decisions are taken by the *Methodology* builders, not by the staff assigned to do the work. The overt case for the *Methodology* includes standardization, documentary uniformity, managerial control and state-of-the-art techniques. The covert case is simpler and cruder: the idea that *project people aren't smart enough to do the thinking*.

(adapted from *Peopleware*, Tom DeMarco & Timothy Lister, p115)

Management

Two types of management are relevant to software:

- people management
- project management

Both types are best seen as enabling activities: a good manager doesn't do the actual work of building software but aims to maintain an environment in which it's possible for people to get on and do so!

Of course many would argue that the bottom line is always financial.

A manager is not the same as a leader...

One view: management as enabling activity

No project can succeed without management support. The best sort of management support is the kind in which management doesn't find out about the project until it's a market success. If management notices a project too soon it'll support it in the following ways:

- *Demand frequent status reports to explain why the team doesn't have enough time to meet deadlines.*
- *Demand explanations of how the project differs from all the projects that have similar acronyms.*
- *Ask the team what it could do if it had only half as much funding.*
- *Appoint an Oversight Committee whose members are always on trips.*

To put it another way, managers understand that their role is to remove obstacles from the project team. They probably could do that, with the help of Dr Kevorkian, but most managers are not such good sports. Therefore, coincidentally, the biggest obstacle to the success of any project is management itself.

(from The Dilbert Principle, p233)

Another view: management as financial control

Here's an extreme expression of a financial view of management (from Drucker, P.F. The Practice of Management):

Management must always, in every decision and action, put economic performance first. It can only justify its existence and its authority by the economic results it produces. There may be great non-economic results: the happiness of the members of the enterprise, the contribution to the welfare or culture of the community, etc. Yet management has failed if it fails to produce economic results. It has failed if it does not improve or at least maintain the wealth producing capacity of the economic resources entrusted to it... The first definition of management is therefore that it is an economic organ of an industrial society. Every act, every decision, every deliberation of management has as its first dimension an economic dimension.

Software project management

A project manager arranges for the following functions to be fulfilled:

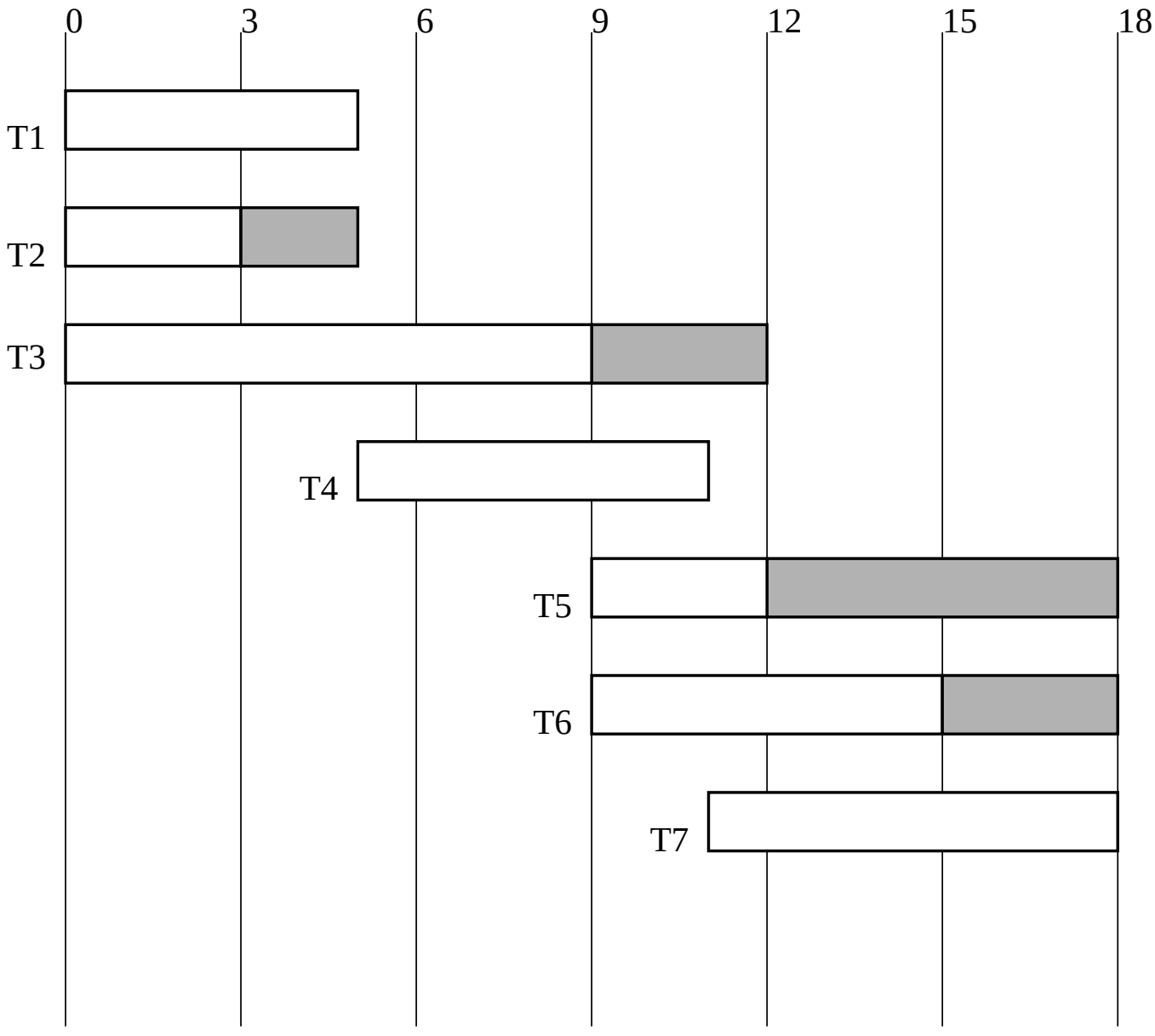
- Planning
- Organizing
- Staffing
- Monitoring
- Controlling
- Innovating
- Representing

Gantt charts

An example of a project planning tool, to help with scheduling.

Divide project into tasks, with milestones at the end. Analyse (e.g. in graphical network) dependencies between tasks. Now lay these out as bars running across time, respecting dependencies. This reveals the critical path of tasks for the project. Optionally, show permissible slippage with shaded bars.

- Task 1 takes 5 weeks.
- Task 2 takes 3 weeks.
- Task 3 takes 9 weeks.
- Task 4 takes 6 weeks, and depends on tasks 1 and 2.
- Task 5 takes 3 weeks, and depends on task 3.
- Task 6 takes 6 weeks, and depends on tasks 1 and 3.
- Task 7 takes 7 weeks, and depends on task 4.


0
3
6
9
12
15
18
T1
T2
T3
T4
T5
T6
T7

Project tracking

The project manager needs to decide how and what to track. E.g.:

- how much time each person spent on each task, and when?

- just total effort expended?

- something in between?

Aim is to find a happy medium between having too little information to tell whether things are OK, and so much that it's very time-consuming to manage the information. Ideally want meaningful info!

Tools are available and helpful, esp. for big projects, but they don't create the data or decide what to do as a result...

Revising the project plan

As the project goes on, estimates have to be revised in the light of progress, unforeseen problems etc. Typically a large project will replan once a week.

Depending on the circumstances slippages may mean

- reallocating resource (but see *The Mythical Man Month*)
- cutting functionality
- asking the customer for more money
- losing profit

Tell them the most important metric is what proportion of the project staff's time is spent explaining to the customer why the project is late Anonymous software engineer.

Reading

Required: GSWEBOK Ch11

Suggested: Sommerville §5.4, Ch 27,28 and/or Stevens Ch19,20.

Suggested: Sommerville Chs 4,5,17 and/or Stevens Ch 4.

Quotes of the day

> *Adding manpower to a late software project makes it later.*

("Brooks' Law") Fred Brooks in *The Mythical Man-Month* (1975), chapter 2

> *The ultimate management sin is wasting people's time.*

Tom DeMarco and Timothy Lister in *Peopleware*

> *Any process that tries to reduce software development to a "no brainer" will eventually produce just that: a product developed by people without brains.*

Any Hunt and Dave Thomas in *Cook until done*