COSC345 Week 18 Notes

This follows the previous pair of lectures because inspection is one of the tools we use here too, but for different reasons.

It's important to realise that "quality", like "intelligence", is not a single dimension. A program might have a wonderful user interface but get its sums hopelessly wrong; another program might do its calculations perfectly but require a month's special training to use it. Another program might be just right both of those ways, but be unmaintainable, so that the next release of Windows kills it. For example, Think Pascal worked and many people liked its interface, but Apple's switch from M68k to PPC hardware killed it.

It may actually be rational and ethical to offer a low quality product: if you are starving, food that is past its use-by date but is at a price you can actually afford may save your life. (I have seen a shop specialising in just such goods in a major Western city.) Second-hand cars and computers are, by definition, of lower quality than new ones, but as long as the goods live up to the claims made by the vendor, there is no harm in that. This year, David Eyers and I are buying a couple of second-hand multicore servers we couldn't have otherwise afforded. Products that do not at all do what they are supposed to, or expose the user to unnecessary danger, are clearly wrong, but slow or old-fashioned need not be a problem. What you have to do is decide what your market needs and what your market can afford and what you can afford to do for that price. You have to *decide* what level of which aspects of quality to aim for, and you have to *measure* to see that you have achieved your aims.

Can you find other examples of mangled names? (Apparently one US political candidate in the 1940's used to refer to the inventor of Relativity as "Professor Weisenstein".) You are fortunate indeed if you have never had your own name mangled by others.

Mass production manufacturing systems run standardised processes to produce large numbers of similar physical objects. In the case of lightbulbs the objects may be as close to identical as we can make them. In the case of cars, there may be all sorts of options that may be selected by the initial purchaser, but it's a selection from a small range of choices and there will be very many objects with the same choice of some feature. The ISO 9000 quality approach has its origins in World War II bomb factories where it was important not too have too many bombs go off too early or not at all. Such manufacturing systems can to some extent work by inspecting samples of the final product and sending entire batches back for rework or recycling if the defect rate is too high. This isn't really an option for programming. Our defects are not in the "manufacturing" process (writing tapes in the old days, burning CD-ROMS or putting things up on the Web these days) but in the "design" process. By the time we have something corresponding to the classical notion of "product" doing anything about serious defects is going to take a lot of expensive work right when we were expecting to be paid.

This doesn't mean that we can't fix things after they have been fixed. The very notion of a "service pack" (Microsoft terminology for a patch) tells us otherwise. The Macintosh I am writing this on periodically nags me about new versions of this program or that, and the departmental iPad I use tells me about updates *every day*. Customers don't mind a stream of new releases as long as (a) they don't have to pay a lot to buy or install them and (b) they get more than just bug fixes; lots of bug fix releases makes you look bad. What it does mean is that late fixes to major aspects of systems are scarily expensive.

The iPad example reminds me that updates which *remove* features or make gratuitous changes to an interface people have got used to will drive customers to frothing rage. One of my colleagues, a big fan of Windows, has now switched to Mac OS X because of Windows 8.

In short, we need to know when we start to drift off course, not when we end up on the rocks.

The slides still refer to Sommerville's discussion of quality management because it was one of the best things in his book and I haven't yet seen anything I like better.

The "ilities" are so-called because most of them have names ending with "ility". It's basically a joke. You will find other lists, some longer, some shorter. The key points are

1. there is more than one aspect to quality

2. different aspects matter to different people (users and maintainers in this table, but there are others; it is a useful exercise to try to identity some other groups and what matters to them)
3. we have to find measures that approximate these qualities
4. we may need to measure or estimate quality aspects in incomplete software, ideally at the architecture level
5. we may not be able to put numbers on something, but if we can keep a history of recognisable patterns that led to quality problems in the past we can watch for those patterns turning up again

The good aspect of ISO 9000 is writing down what you meant to do, what you did do, and what happened, so that you do not forget the experience we would hope you can learn from.

I want to repeat what I said about formal inspections. Quality reviews are *not* looking for someone to blame. They are concerned with finding out how things are now, whether we have a problem or not, and they are concerned with why we have the problems we do, but a smart organisation is looking for system level issues rather than individuals to blame. An organisation that goes around looking for people to blame is going to force workers to put more effort into protecting themselves than into advancing the purposes of the organisation. At various times in these lectures I mention the idea of "social capital" and "high trust" *vs* "low trust" societies and organisations. High trust outfits are efficient because people can concentrate on the work that is before them instead of watching for enemies behind their backs. They are also efficient because people are happier that way.

[NEW ZEALAND EDITION] There is a science fiction/fantasy trilogy called "Illuminatus!" It's firmly rooted in 60's ideas, and I found it morally and religiously offensive, but a good read none the less. One of the slogans of the Discordian Society is "Communication is only possible between equals". That has stuck with me as one of the few things in the book I can really agree with. One of the noblest ideas in the history of politics is *parrhesia*, freedom of political speech, the idea that the meanest citizen in the land is not just allowed to tell the political leaders the truth as a favour, but is right to do so and it is wrong to punish anyone even slightly for telling the truth. One of my heroes is Thersites, who in the Iliad tells Agamemnon a few home truths, and is beaten severely by Odysseus for his trouble. But Thersites was *right*. An organisation in which workers dare not tell the truth to the bosses is an insane organisation, quite literally; one in which the bosses will be managing according to a fantasy of what people think they want to hear rather than according to the way things actually are.

[/NEW ZEALAND VERSION]

[OMAN VERSION]

There is a slide about truth, with some quotations from the Quran. Please replace them with more appropriate ones. The point is that truth is both a religious obligation (if I have understood Islam correctly) and a practical necessity, and this means *both* that workers should tell their superiors the truth about the state of the project *and* that their superiors have a moral obligation and a practical need to create and maintain an environment in which workers are not afraid to do so. If bosses only listen to what they want to hear, they will be managing according to stories told them by the fearful, and will make decisions that are out of touch with reality. That is practically bad. It is morally bad to deceive people who have a right to the truth from you, even if they wish to be deceived, and it is worse to be the one who pushes his inferiors into such sin.

There are several computer societies around the world. Most of them have a code of ethics, and the codes that I am familiar with include an obligation of honesty towards one's clients.

If you find yourself in a situation where you must either lie, suffer, or leave, choose to leave. Such a project is doomed anyway.

[/OMAN VERSION]

I give an example from my own experience. I once worked at a startup in California where, after a couple of years, a manager decided to introduce time reporting. I dutifully did my best to fill out my sheets. The trouble was, the number of hours worked added up to nearly 80 hours a week. That's because that's how much I was working. I was called in by my manager and told "these time sheets are no good; they have to add up to 40 hours". At that point I lost all respect for him as a manager and decided that if he didn't want to know the truth, I would just have to make up numbers he would believe. (Remember, he wouldn't listen

to the truth.) From then on until I left, every time sheet was a complete fantasy. What point is there in managing with time sheets that are complete fantasies? And how can they not be complete fantasies if you are unwilling to accept truthful ones?

Internal audits may be problematic. Kantner says that "because internal auditors may not be people with any responsibility for the areas they are auditing, the audits tend to be objective". I've never worked for a large business (other than a University, which Dijkstra rightly said "should be as unlike a business as possible"). The largest company I worked for had about 40 employees. When it came to auditing the quality of our software, there was nobody at the company who was able to do so who wasn't already involved in producing it. Small companies just have to rely on everyone knowing each other, and on people knowing that other people will be able to see the same things, so there is no point in not being objective.

One noteworthy point about inspection is that if three people study the same piece of code, they may all turn up to the meeting with 8 issues in it, but they won't be the *same* 8. If you can possibly spend some lab time on a real inspection, do so. People will be very surprised by the things other people saw that they didn't.

We provide a directory with some metrics tools (metrics.d). For C this is largely superseded by Spinellis's CScout program. I have a great deal of scepticism about fancy metrics. Pretty much everything correlates with SLOC. However, you can develop your own situation-specific metrics. (Such as memory budget, or number of untested lines, or spelling mistakes in comments, or anything that you have some intention of acting on.)

The "let's look at a metric" slide examines a question from the paper evaluation questionnaires at this University. You will probably have your own evaluation forms, and they will probably be, let's put this kindly, questionable too. The point here is to show that there can be metrics for things other than source code, and that metrics can themselves be rationally appraised. You should find your own example. Regrettably, it won't be hard.

Process models: you'll find them described in software engineering journals. These days the Business Process Modelling Notation is a popular graphical notation for workflows (programming people like machines). I really want to stress the point that any organisation in which the rules stop you doing the right thing is at best crazy and at worst evil. There have to be meta-rules allowing the use of human discretion simply because humans are not the All-Knowing All-Wise Maker of the Universe.

"Process metrics" (long, resources) are largely about the kind of thing that accountants need to worry about. We need to ensure that they have good information because a company that goes broke is a company that can't pay us any more.

I repeat the endorsement of anything by Weinberg. After reading his book about metrics you will want a total rewrite of this pair of lectures.

Here is the part of the Iliad dealing with Thersites. The pro-aristocratic Homer cannot vilify him enough, but the truth shines through. In particular, in this case, he is *not* trying to make the Achaeans laugh. Remember, given the rest of the Iliad, every word that Thersites says is absolutely true. As the Wikipedia article on the Trojan War puts it, "Few of the Achaeans returned safely to their homes". Thersites' advice was *good* advice.

"The rest now took their seats and kept to their own several places, but Thersites still went on wagging his unbridled tongue—a man of many words, and those unseemly; a monger of sedition, a railer against all who were in authority, who cared not what he said, so that he might set the Achaeans in a laugh. He was the ugliest man of all those that came before Troy—bandy-legged, lame of one foot, with his two shoulders rounded and hunched over his chest. His head ran up to a point, but there was little hair on the top of it. Achilles and Ulysses hated him worst of all, for it was with them that he was most wont to wrangle; now, however, with a shrill squeaky voice he began heaping his abuse on Agamemnon. The Achaeans were angry and disgusted, yet none the less he kept on brawling and bawling at the son of Atreus.

'Agamemnon,' he cried, 'what ails you now, and what more do you want? Your tents are filled with bronze and with fair women, for whenever we take a town we give you the pick of them. Would you have yet more gold,

which some Trojan is to give you as a ransom for his son, when I or another Achaean has taken him prisoner? or is it some young girl to hide and lie with? It is not well that you, the ruler of the Achaeans, should bring them into such misery. Weakling cowards, women rather than men, let us sail home, and leave this fellow here at Troy to stew in his own meeds of honour, and discover whether we were of any service to him or no. Achilles is a much better man than he is, and see how he has treated him—robbing him of his prize and keeping it himself. Achilles takes it meekly and shows no fight; if he did, son of Atreus, you would never again insult him.'

Thus railed Thersites, but Ulysses at once went up to him and rebuked him sternly. 'Check your glib tongue, Thersites,' said be, 'and babble not a word further. Chide not with princes when you have none to back you. There is no viler creature come before Troy with the sons of Atreus. Drop this chatter about kings, and neither revile them nor keep harping about going home. We do not yet know how things are going to be, nor whether the Achaeans are to return with good success or evil. How dare you gibe at Agamemnon because the Danaans have awarded him so many prizes? I tell you, therefore—and it shall surely be—that if I again catch you talking such nonsense, I will either forfeit my own head and be no more called father of Telemachus, or I will take you, strip you stark naked, and whip you out of the assembly till you go blubbering back to the ships.'

On this he beat him with his staff about the back and shoulders till he dropped and fell a-weeping. The golden sceptre raised a bloody weal on his back, so he sat down frightened and in pain, looking foolish as he wiped the tears from his eyes. The people were sorry for him, yet they laughed heartily, and one would turn to his neighbour saying, 'Ulysses has done many a good thing ere now in fight and council, but he never did the Argives a better turn than when he stopped this fellow's mouth from prating further. He will give the kings no more of his insolence.'" — The Iliad, by Homer, translated by Samuel Butler, book II.

Some quality issues that hit me yesterday:

1. The "live" release of a certain major operating system (a) reported that some services timed out and others were KILLed during startup, (b) required you to log in (in order to install the real thing) but nowhere told you who to log in *as*.
2. A debugging tool I have been known to recommend appeared to freeze when asked to navigate into /home and I had to place a symbolic link to the directory I really wanted in /Scratch.
3. The latest release of a well known C compiler works fine when I do "cc foobar.c" but when I do "cc -O foobar.c" it complains that the run-time linked cannot find a certain undocumented function.
4. The 'Mail' program on my desktop Mac has been refusing to work for 10 days, claiming that the mailbox is in use by another program, which it is not.
5. In a set of programs written to persuade researchers in a certain field that a certain programming language was good, the program accesses matrices by columns instead of by rows, which makes it run more than 2 times slower than it should. It also has work inside a loop that should be outside, adding 3%. In a claim about performance, this is shoddy.
6. Two security researchers have announced that UCB is fundamentally insecure. Other people have explained why they must be at the very least overgeneralising (many USB devices *cannot* be reprogrammed, and USB firmware runs on the USB device, not in the host). I don't know how, but *somebody* is full of it.