More research on estimation

(c) Allan Kelly, 2013

http://www.allankelly.net, allan@allankelly.net

A little over two years ago I set out to better understand the art of time estimation, specifically time estimation for software development work. This was initially motivated by a request from Jon Jagger to support two assertions I claimed were "proven by research." I read and reviewed a number of academic studies on estimation and posted the results online - see http://www.allankelly.net/writing/misc.html.

Earlier this year my interest in the subject was sparked again and I decided to look for more research in the field. Once again I have read a few hundred pages of academic research studies.

The notes in this document are my own notes on these studies. The notes are primarily intended for myself but I am more than happy to share them. I have also attempted to draw some conclusions from these studies.

Effects of Deadlines - "Procrastination, deadlines, and performance: self-control by pre-commitment"

Ariely & Wertenbroch - (Ariely and Wertenbroch 2002)

This is a study concerned about the role of deadlines in combatting individual procastinations. As such it is a little different to the other studies I have reviewed because it is not concerned with estimation, it is concerned with deadlines. I have reviewed and included the study here because deadlines have a bearing on the iteration and estimation debate.

Before looking at what the study says there need to be two caveats on this review. Firstly the study as primarily concerned with deadlines and whether people could combat procrastination through self-imposed deadlines. Procrastination is sometimes and issue for software development teams but is not an issue which is discussed much, and it is not an issue I have set out to examine in these reviews. However, I believe the findings of this study are interesting in the debate all the same.

Secondly this study looked at individual not team behaviour. The authors have nothing to say about the role of deadlines in team working. If the findings of this study can be extended to teams then they are interesting. I am going to make the rather general assumption that they can.

Perhaps I should say, I stumbled across this study by chance, following references from other studies here, so I have not gone looking for contradictory studies or other studies in this field.

This was a laboratory style study in which the researches divided subjects into three groups and applied slightly different conditions in order to test three questions:

a) Do people self-impose costly deadlines to overcome procrastination?

b) Are self-imposed deadlines effective in improving task performance?

c) Do people set self-imposed deadlines optimally?

And the answers are.... Yes, Yes, No.

Self-imposed deadline are a little like iterations, we set our teams reoccuring deadlines, every two weeks. That teams self-impose dealdine is obvious to students of Agile, this study merely says: it is a strategy that individuals employ too.

If we regard iterations (i.e. sprints) as a form of self-imposed deadline then we can expect a team that works in iterations to be more effective than team which doesn't work with deadlines. Now while that might seem reasonable it rests on two assumptions: that we can extend research on individuals to teams and that iterations are self-imposed deadlines. Which lead nicely to the next question.

The researchers find that individuals do not impose their own deadline optimally. Now optimally might be a strong word, it is true the research doesn't consider every permutation but they do contrast self-imposed deadlines with two other types: maximally extended deadlines and externally imposed evenly spaced deadlines.

Maximally imposed deadlines correspond to traditional project management: we set a date, if it is not acceptable to "the business" then it is reduced.

However we we can extend it some more then we probably do - if not in the first instance then by adding “contingency” or when projects start to “slip” asking for more time.

But, according to the researchers, participants performance is highest when they work in externally imposed regular deadlines. Regular deadlines correspond to fixed-length iteration in the software development world.

So if this research is valid for teams then working in fixed length iterations produces higher performance than either traditional maximum deadlines or Kanban-esque freeflow.

This leaves the point of self-imposed v. external. The researches did not consider self-imposed regular deadlines so we have no data here. However I’m not convinced this is undermines the research from a software development point of view. For an Agile team it is debatable whether an iteration deadline is self-imposed or not. The team might be given the schedule in the first instance but teams have power to change it - probably via a retrospective but possibly via other mechanisms.

Secondly, since the development team have a say in what work is scheduled into the iteration the deadline is in part externally imposed and in part self-imposed.

Caveats and debates aside I think this research supports that working in fixed length iteration can be an effective approach for teams.

How Long Will It Take? Power Biases Time Predictions

Mario Weick and Ana Guinote - (Weick and Guinote 2010)

This paper looks at the effects that power can have on forward looking time estimates. The authors hypothesis is that “power increases the tendency to underestimate task completion time”. To investigate this hypotheses they conducted several experiments, some in a lab. As a result of which they claim to have proved their hypotheses.

Thus the paper claims: people in positions of power and authority tend to provide shorter time estimates than those in lesser positions.

“Power” is defined as social-power, " the ability to influence and control others’ outcomes and resources."

The transmission mechanism is first the goal: those in power focus more on the goal. This has two consequences. First they focus their attention more narrowly on the goal thus consider other sources of information (e.g. how long something similar took last time) less.

Secondly because they are focused on the goal "powerful individuals are more oriented towards achieving gains and rewards, and less apprehensive about avoiding threats or constraints as compared to powerless individuals". This does mean that these people are more action oriented and being goal focused may benefit them during the work but in planning accurately it is a hinderance.

The authors also point out several other transmission mechanisms which might be at work:

- Powerful individuals may well be more optimistic than others and therefore dismiss impediments and threats
- Powerful individuals may have more confidence in their abilities
- Power itself may make the individual more optimistic.

The study authors focus on the goal explanation and claim that it is the goal and consequent changes in attention, rather than optimism, that leads to lower estimates. ("We did not find evidence that power leads to greater bias in time predictions because powerful people are more optimistic; ... Likewise, the effects of power could not be attributed to differences in people's beliefs in their abilities (i.e., self-efficacy), or to differences in mood.")

As a result of their studies the authors say: "powerful participants, more than [the] control [group] or powerless participants, consistently underestimated the time it would take them to complete future tasks. ... power has a fundamental impact on planning behaviour, increasing biases in time estimates."

It needs noting that the study was concerned with *power*, not *powerlessness*. There are no claims, or data, that reducing an individuals power will increase the accuracy of their estimates.

One interesting aside the authors mention is the role of abstraction. Abstraction, like power, leads individuals to focus more on the outcome, drawing

attention away from the tasks. Thus, estimating a very abstract piece of work, say a piece of software, might be more prone to underestimates.

Conversely the authors also note that abstraction can lead people to consider past event more, thus correcting the bias. Either way this adds an interesting twist to the software prediction debate.

Although the individuals in this study had "power" they did not exercise power. This might be a small flaw in the study from some points of view. However the authors note that the more power someone has the greater their bias (i.e. underestimate) is likely to be.

Finally, another corollary is noted: "the same mechanisms that make people prone to underestimate time also increase the likelihood that individuals take unwarranted risks. The present findings thus complement past research that has shown that power promotes more optimistic risk-assessment."

Vierordt's Law

Vierordt's law concerns the perception of perceived duration of a time period in retrospect. The law actually dates back to 1868 and is named after Karl von Vierordt, a German physician, who first observed it. According to Wikipedia Vierordt was the first person to carry out experiments in time perception.

The law states that people overestimate short periods of time and under estimate long period of time. For example, suppose you are waiting for a bus without any means of telling time. If you wait for two minutes and someone asks "How long have you been waiting?" you may well answer "For ever, at least five minutes, possibly 10". But, if someone else asks you 18 minutes later you may well say "Ages, about 10 minutes."

While I just invented those numbers those numbers several sources suggest the cut-off point is between 2 and 10 minutes, i.e. anything less than two minutes is a short interval and anything more than 10 long, in between is a grey area.

> Not only are short periods remembered as being longer and long periods remembered as being shorter but the longer the period the greater the underestimate of time. Wait another 20 minutes for that bus and when someone asks you may well say "Too long, 20 minutes" when in fact you have waited 40 minutes in total.
>
> While a couple of studies mention Vierordt's Law many do not. One might even say it is notable by its absence. I'm not sure why this is so, possibly the research is too old to be considered relevant, possibly the various researchers are also aware of the law or possibly, compared to modern techniques, the Vierordt's research is not valid.

Effect of task length on remembered and predicted duration

** Michael M. Roy and Nicholas J.S. Christenfeld ** - [@Roy2008]

This study examined Vierordt's Law and found the law held. Actually, this paper, and this law answers one of the original question I had when I started this research. I wanted to know whether retrospective time estimation was any more accurate than forward time estimation. Vierordt's law goes a long way to answering that question.

Actually this paper goes a long way to answer my general question. The authors examined whether Vierordt's law holds looking forward: certainly this would fit with most of the other research I have reviewed but this is the first paper that a) discussed Vierordt's law and b) looks at this specific question.

The authors conducted several laboratory experiments with over 200 participants given simple tasks to perform. The found that Vierordt's law held: participants given the short task over estimated how long the tasks took (by 5.9%) while those given longer tasks underestimated how long the tasks too (by 36.2% for the longest task).

And perhaps more interestingly the researchers found the same pattern for predicted time. That is, participants given the shorter tasks over estimated

how long the tasks would take and underestimated how long the longer tasks would take.

Although the switch-over point from longer to shorter was slightly later it still fell within the 2 to 10 minute range. To be exact, tasks taking less than 1 minute 32 seconds were over estimated in advance while tasks taking more than 1 minute 56 second were underestimated in retrospect.

It is worth quoting from the papers findings:

- "Task duration had a significant effect on magnitude of bias: Participants were inclined to overestimate when the task was short and to underestimate when the task was longer, with the amount of underestimation escalating as task duration increased."
- "The results indicate that Vierordt's law applies not just to recollections of past tasks, but also to estimates of future task duration. We found no difference between past and future estimation, with both affected in the same manner by duration of the task."

Although two minutes seems a very short frame of reference the authors do suggest that the cut-off point changes as the scale (second, minutes, hours, days, weeks) changes.

For software development

These findings have are quite significant for software development. Given that few development tasks are less than two minutes in duration, and those that are are unlikely to be estimated or managed, Vierordt's law is dominant. Even if switching to a different scale moves the cut-off point one might assume it is still towards the low end of the interval.

Two immediate thoughts.

First any attempt at traditional time tracking - filling in time recording sheets - is pretty much doomed. One of the other studies I looked at before (Zackay and Block 2004) suggested that if you wanted to accurately track time spent on a task you had to devote energy to it which detracted from actually undertaking the task.

Second, estimating larger things leads to a greater underestimate: whether at a task or project level the bigger the thing you are estimating the greater the undershoot. (This would also feed into the diseconomies of scale argument.)

Finishing on time: When do predictions influence completion times?

Buehler, Peetz, Griffin 2009-2010 (Buehler, Griffin, and Peetz 2010)

This study looked at whether the predicted time to undertake a task influenced the actual time it took to do the task. To investigate this question the authors conducted four laboratory style experiments. The result is not a simple yes or no, to understand the results we need to introduce the concept of "Open tasks" and "Closed tasks"

- A closed task is one which can be carried out in a single session, e.g. proof-reading an article
- An open task is one which needs multiple sessions; perhaps it requires reference to other people, or some research or look-up which is outside the main task, e.g. filling a tax return

For both types of task, in all studies, participants underestimated how long it would take to do the task. However, for closed tasks those with optimistic estimates finished the task sooner than those with pessimistic estimates.

To illustrate this imagine an optimistic programmer asked to undertake a fairly short code change. The optimist estimates the work will take fours hours may actually take six hours to undertake the task. However a pessimistic programmer who estimates the task will take six hours might actually take eight.

Note we are not talking about coercion here. While these researchers used an anchoring procedure to induce optimistic or pessimistic estimates there was no equivalent of a manager standing over our imaginary programmer saying "Can you reduce that estimate?". There is no data in this paper on what effect that would have.

(Although in their review the researchers do note that offering financial reward and publicly sharing estimates causes individuals to suggest lower estimates.)

Now the the open tasks. those which could not be performed in one sitting. Here the researchers found no difference in how long it took to undertake the task. In both cases optimistic and pessimistic estimates were short.

Lets return to our imaginary programmers. If the task involves undertaking some work, passing it to someone else for test, getting the results back and then finishing the work both optimistic and pessimistic will take the same time.

While we can speculate on why this might be, and the researchers do, it occurs to me that perhaps what we should be considering is: *how can open tasks be made more closed?*

Lets continue with our example: if our programmers are practising TDD, or if a tester is working directly with the programmer - perhaps pairing - then the task will be more closed than otherwise. Regardless of the benefits of these practices in their own right they may also improve estimation. I hasten to add I have no data here so there is an element of speculation here.

Returning to the open tasks, although the optimism did not make the task happen any faster - unlike closed tasks - it did cause the task to be started earlier. But this did not translate into an earlier finish. Once started those who started earlier (those with optimistic estimates) worked at the task for longer than those who started later (those with pessimistic estimates).

Both groups finished the task by the deadlines and there was no evidence that the quality of the work differed. Simply: starting the open task sooner meant a longer elapsed time.

For example, given a programming task an optimistic programmer may estimate it will have six hours as opposed to a neutral programmer who estimates it will take eight. The optimistic programmer starts work immediately and end up taking ten hours. The neutral programmer starts work two hours later and takes nine hours.

One more finding from the paper is worth noting:

> "Notably, in our studies tasks that were quite similar in duration showed markedly different completion times depending on how prone they were to interruptions and delays, suggesting that task completion times are determined to a great extent by factors other than the duration of a task itself"

This fits with the popular notion of "flow" and the desire of many programmers to free themselves of disruptions and interruptions.

This paper is interesting for two more reasons.

First the paper introduces the *mere-mention effect*: "Predicting, per se, may also have self-fulfilling effects on behavior; simply asking people to predict whether they will perform desirable actions (e.g., voting, donating to the Cancer Society) increases the probability that they will perform these acts)."

Second it also demonstrates anchoring in action and proves the effect (again). If a time period is suggested - explicitly or implicitly - before the estimate is given the estimate will be influenced by the suggestion.

For software

Moving slightly away from what the paper says and putting these findings into a software context. What emerges is a rational for optimistic estimates: either by reducing the elapsed time or simply by bringing both start and end dates forward work should benefit.

This in itself might explain why the planning fallacy is so prevalent: it makes sense to underestimate tasks be optimism has positive side effects. Unfortunately this detects from accuracy of plans, if we could adjust for this optimism and produce accurate plans the optimistic predictions would look less optimistic, even pessimistic. This in turn would increase the duration of closed tasks and delay the start of open tasks.

In effect this logic proves Hofstadter's Law: "It always takes longer than you expect, even when you take into account Hofstadter's Law." (Hofstadter 1980).

The finding that open tasks take longer when started earlier supports the "fit the work to the deadline" approach rather than the "set a deadline to match the estimates" approach, i.e. the time-boxed Agile way of working over the traditional estimate and plan approach.

This turn supports the view that long-range planning in development is of limited value. Consider an analyst tasked with understanding requirements for work that might happen in six months time. On the face of it she has plenty of time to understand what is needed. However this approach might

make for six months of on-off-on again working, say three months of effort. While starting the work two months before the requirements are needed may cause all necessary work to be compressed into two months.

The above notes suggest an interesting line of thinking which brings together several different elements discussed here, and my own observation, so you have to decide for yourself if you agree or not. Ideally someone would do research on this.

As the authors noted there are several ways of influencing people to produce a shorter estimate, e.g. anchoring (as per the experiment) and offering financial incentives.

Lets assume a programmer is offered a financial incentive, a bonus, to complete a piece of work. Based on the research we would expect them tender a shorter estimate - the person asking for the work is probably already happy at this point. But what happens now?

If the work is a closed task then we might expect - as per the research - that the task will indeed be finished earlier, the programmer will get his bonus and everyone will be happy.

However if the work is of an open nature then the estimate will be shorter than without the bonus but the completion date will not change. The programmer will be unhappy because they will not get their bonus and the person wanting the work will be unhappy because the work is actually later than they would have expected - and the work may actually cost more because it took longer.

Now an observation of my own: financial incentives are seldom - if ever - offered for closed tasks in IT. In my experiences bonuses are offered for significant tasks, tasks which take more than on sitting, tasks which often involve co-ordination with others - typically things we call "projects."

Thus, offering a bonus may bring about exactly the opposite effect of that which is desired.

Correcting Memory Improves Accuracy of Predicted Task Duration

Roy, Mitten and Christenfeld, 2008 - [@Roy2008]

This paper sets out to examine whether predictive estimation can be made more accurate. To do this the researchers conducted several experiments which involved providing participants with feedback about how long previous tasks took.

The results: estimates did get more accurate with feedback.

Looking at all the research literature there are two reoccurring explanations of why people underestimate. One is the goal focus - when people focus on the end result they think less about the work needed, the desirability of the end result causes them to become optimistic in estimates.

The second explanation is that people neglect to consider past experience in undertaking tasks and therefore are over optimistic. This explanation may further be split: people neglect past experience because that are resistant to doing so, or that the memory itself is inaccurate. This paper considers the second of these explanations, that memory is inaccurate.

These two theories (goal focus and memory neglecting) are not entirely exclusive, my guess would be both are at play in creating underestimates.

The key point in this research seems to be the feedback loop. Although some of the research I've read suggests people neglect past experience (e.g. Buehler, Griffin, and Ross 1994) these researchers suggest it has a role to play.

One problem I see in a work environment, specifically software development, is that quantifiable past experience is hard to come by. We could ask someone "How long did it take the last time?" but this falls foul of Vierordt's Law (and the supporting research) which says we retrospectively underestimate. We could use traditional time reporting systems but a) Vierordt's Law is at work again and b) Capers Jones suggests such systems are inaccurate to the point of risk (Jones 2008).

Short of having someone actually observe and time how long individuals take on their work (as the researchers do for this study) it is difficult to see how we could gather this data. (Although source code control systems might provide an interesting data set to analyse.)

Complicating matters is that software development, unlike many of the experiments in these research tasks, is a complex field that involves innovation and novel tasks. The fact that we tend to automate much routine work more or less guarantees that any task we undertake is indeed novel.

These researchers also mention that "Underestimation also occurs more frequently for familiar tasks, whereas novel tasks are often overestimated" and cite supporting research. I've not had a chance to follow this reference up but this would tend to contradict what we see in software development.

The actual research consisted of several paper counting tasks (closed tasks to use the earlier jargon). In the first experiment participants performed the task and were timed. Some of the participants were then asked to estimate how long it took (which they underestimated) and then asked to estimate a repeat of the exercise (which they also underestimated.)

Other participants were told how long it took and then asked to estimate how long a repeat would take. This group over estimated how long the task would take to repeat, although another way of looking at this is to say they underestimated the improvement they made.

Unfortunately the paper doesn't say how long it took each group to complete the second (repeat) exercise. Consequently it is not possible to tell which group completed the task more quickly. In the light of the previous paper it would have been interesting to know whether the lower estimates were related to better improvement overall.

This highlights several points that need to be stated clearly:

- Estimation is not an end in its own right; the estimation can be a step along the way to achieving a finished task. I would suggest most people are more concerned about how long the task takes than they are about the estimates.
- Estimation accuracy isn't just a measure of how close to "actual" the estimate is. There is the direction of the inaccuracy (under or over estimate) and when more than one estimate is made there is a need to consider variance, i.e. it is possible to have some estimate which are individually closer to actual while others are further away.

After experiment 1 the authors state:

> "Overall, it appears that supplying feedback about how long the task took previously improved participants' ability to predict how long it would take them to perform the task again in comparison to participants relying on memory of previous task duration."

I'm not sure about this, I wonder if what the authors doing is actually anchoring participants. Are they really correcting memory? And does it matter?

Another twist to the retrospective estimation is added by this paper: when the retrospective estimate is made. In one experiment there is a very small overestimate when the retrospective estimate is made one week later but a small underestimate when it is made immediately.

Overall the second experiment has similar findings to the first (it also attempts to investigate whether giving feedback at different times makes a difference.) This time those who estimate how long they took on the first exercise (of experiment 2) overestimate how long it will take to do the second exercise, although they do not overestimate by as much as those who are given timed feedback rather than asked to estimate retrospectively. So again, people underestimate their own improvement.

(Notice there is no way to win here, underestimation seems to occur in some form each time!)

I must admit at about this point my ability to understand this paper starts to break down. I'm trying, and I will keep trying, but jumping between tables, numbers, statements and log statistics makes it hard to follow!

Some of the complications in this paper come from the researchers desire to see how giving feedback at different times (immediately after the exercise or a week later) influences their estimates. While this is an worthy exercise it is not what I'm looking for. If you are interested you are best reading the paper yourself.

The outcome of this research is best summarised towards the end of the paper: "Supplying participants with feedback on previous task duration, whether their own or that of people in general, led to increased accuracy for several different tasks". While this might seem obvious the researchers point out that some previous studies do suggested that feedback did not effect estimates.

However the paper also states: "we found no benefit from asking participants to remember previous task duration before making a prediction".

The research suggest that people are willing to consider information on past tasks but that their recall of past tasks is faulty. Therefore to supply the information requires a specific procedure.

For me this paper raises almost as many questions as it answers. In terms of software development this is one of the more difficult papers to draw lessons from. While all the research papers I have reviewed to date look at tasks far simpler than coding (e.g. counting paper) there are usually lessons that can be drawn. In this case it is hard to see the lessons because software development work is so hard to measure in the first place. Therefore obtaining good feedback to inform judgements is fraught exercise itself.

This research is the first paper I've read where I actually find myself taking issue with some of the research. Perhaps I'm getting too much practice at this but:

- All tasks considered are "closed"
- All tasks are around 15 minutes long so we would expect Vierordt's Law to operate and participants to underestimate
- There is no consideration of whether the estimation process changes the time it takes to perform the tasks

Ironically right at the end of paper the authors say: "there are cases in which data on task duration are available but underutilized. For example, in the area of software design, companies routinely keep detailed records of how long previous projects have taken in the form of billable-hours records."

These seems lax thinking on behalf of the researches. The data they propose using - at least at the individual and task level - is itself the result of retrospective estimation with all the inherent problems in recalling duration as they have been discussing.

This data is also subject to a number of other flaws, e.g. failure to pay or record overtime. Capers Jones suggests traditional time tracking systems loose 20% or more of the actual time spent (Jones 2008). He also states that determining the start and end dates for projects is extremely difficult.

Bias in memory predicts bias in estimation of future task duration

Roy & Christenfeld, 2007, (Roy and Christenfeld 2007)

Another paper from Roy and Christenfeld (as the previous paper, although this one is a year old) that sets out to explain the failure to estimate future durations accurately due to the failure to accurately remember past events.

These two authors seem to be the main proponents of the "failure to remember accurately leads to failure to predict accurately" argument. This argument is not wholly at odds with the "optimism and goal focus leads to under estimation" argument but it is different. Personally I can accept that both play a part but I am sure there are some who would argue that the two are exclusive.

As this paper pre-dates the previous paper and the authors largely cover similar ground I don't intend to examine it as closely as the previous one. Although they don't mention Vierordt's Law the authors do present an impressive list of studies demonstrating that people underestimate past durations.

The authors find that:

> "(1) the tendency to underestimate future duration disappears when the task is novel,
>
> (2) there is similar bias in estimation of both past and future durations, and
>
> (3) variables that affect memory of duration, such as level of experience with the task and duration of delay before estimation, affect prediction of duration in the same way.
>
> It appears that, at least in part, people underestimate future event duration because they underestimate past event duration."

Of these #1 and #3 are the most interesting. Finding #2 supports the argument I have made before that the thing the software industry likes to call "actuals" are more accurately called "retrospective estimates."

The authors conducted a paper folding experiment to test the hypothesis that memory bias wold not exist for novel tasks - the paper shape the participants were asked to fold was new to them. (These were closed tasks, tasks that could be completed in one sitting, to use the terminology introduced before.)

Those who estimated the task duration before folding the shape over estimated how long it would take while those who estimated after the fact under estimated.

This does seem to support the authors hypothesis. However if we are to consider "Accuracy" this doesn't get us very far. The under-estimators (those estimating after the exercise) estimated the task took 10 minutes when it took 13.5 minutes (both averages). So an average displacement of 3.5 minutes.

But the over-estimators thought it would take 20 minutes (on average) while it only took 11.5 minutes. A displacement of 8.5 minutes. (The fact that participants were rounding 10 and 20 minutes is noted towards the end of the paper but the authors say little more, this could be worth exploring further.)

For me these timings are just too close to Vierordt's cut-off at 2 to 10 minutes. Yes, the task took more than 10 minutes but Vierordt's cut-off isn't fixed.

Although the authors say that the actual difference in time to complete the exercise (13.5 v. 11.5 minutes on average) isn't statistically significant it is interesting to note. This could support the case that estimating before hand helps lead to shorter completion times.

I'm not sure how relevant this experiment is to the software world. As previously noted we seldom estimate such small tasks. However it does raise the question of what is novel in the software world?

On the one hand every piece of code which is written is novel because each piece of code does something unique. On the other hand coding is coding, coding one piece of Java isn't a million miles from coding another piece. Perhaps it suggests that initial estimates - say when working in a new language or on a new system - will be more prone to over estimation (although that doesn't ring true with personal experience.)

From my point of view the most significant thing about this research is that it demonstrates how the whole field of estimation is still not well understood and sometimes throws up seemingly contradictory evidence.

The second experiment was another paper folding exercise and addressed point #2 and #3 above. This throws up a few interesting findings:

- "At the very least, it seems clear that properly utilizing memory will not increase the accuracy of prediction." This finding would seem to undermine the authors main argument.

- "Underestimation was greater after a 10-min delay." When the authors introduced a 10 minute delay in asking for the retrospective time estimate the participants increased their underestimate. For anyone trying to have software developers record "actual times" from memory this should be worrying - and again suggests that if an organization wants accurate "actuals" they need to put some effort into the activity.

- "Estimation of duration moved from overestimation to underestimation as experience with the task grew" This finding is very interesting, put together with the "authority leads to lower estimate" this would support the case for not leaving estimated to experienced staff. In fact the authors note, with specific reference to the software development field, that the practice of leaving estimating to "an expert" may result in the estimate being done by the most biased person.

What is clear is that a large number of variables effect estimation accuracy. Not least level of experience but also, at least for retrospective estimates, the delay between undertaking as task and making an estimate.

As the authors point out "participants continued to be biased in their estimates even though these factors were controlled." If accurate estimates (forward or retrospective) cannot be obtained in safe laboratory conditions then one wonders if accurate, estimates can ever be obtained.

When I started to read this paper I didn't expect to get much from it. But having read (most of) it I was proved wrong. That said there are a few questions I think the paper leaves unanswered, potentially further research:

- Is rounding prevalent?
- Do the same findings hold true with longer tasks?
- How novel is novel? What if the novel tasks were more involved.

The Planning Fallacy: Cognitive, Motivational, and Social Origins

Buehler, Griffin and Peetz - (Buehler, Peetz, and Griffin 2010)

Although some of the other studies reviewed here are big, this is the biggest at 62 pages. However it contains no new experiments, it is a review of other work - including much of the work mentioned here - and an attempt to create a cognitive theory of the field.

That is all by way of saying: I haven't read every word of this study, I might have missed something. However, I would like to quote a few statements that drew my attention:

- "The signature of the planning fallacy, then, is not that planners are optimistic but that they maintain their optimism about the current project in the face of historical evidence to the contrary." This might be a little academic on my part but I think it is useful to be reminded of this element of the original planning fallacy.
- "We know from the literature on software development projects that overestimation is typically found when measured either way: as programmer hours or by the time to the completion of the project. Nonetheless, larger overruns are generally found in estimates of completion time"
- In one experiment the researches asked participants for time estimates "if everything went as poorly as it possibly could." The subsequent estimates still proved to be short for 70% of participants. However the estimated times were correlated with actual times, thus providing evidence that people can perform relative estimation.

  Yet when asked to recall the time participants under reported how long tasks took. Again supporting the retrospective estimation is little different than forward estimation in accuracy.

- The reviewed studies again show that for tasks measured from a few seconds to a few minutes people over estimate how long a task will take. As the tasks get longer - hours to days - under estimation is the norm. Vierordt's Law again.

There are some interesting insights into how estimates can become be biased, or more biased towards optimisim:

- "Despite the predictive value of past experiences, our findings suggest that people neglect them while forming predictions."
- Motivated people tend to estimate shorter completion times but those completion times do not match actual times. For example, in one study Canadian tax payers expecting to receive a tax refund estimated they would complete their tax forms 10 days earlier than those with less motivation, but they actually only completed their forms 3 days earlier. The authors conclude "motivation enhances the planning fallacy through a heightened focus on future plans."
- Directly offering monetary incentives seems to create the same effect: estimates are lower but actual task completion time was little changed.
- Group estimation also leads to a great bias towards low estimates: "participants in all conditions underestimated how long they would take to complete the tasks, but predictions based on group discussion were significantly more optimistic (and hence more biased) than predictions generated by individual group members"
- Verbal estimates are more prone to underestimation bias than written estimates.

Obviously if you want unbiased estimates a first step would be to stop: remove financial incentives, avoid group estimates and write estimates rather than speak them. Beyond there are several strategies for de-biasing estimates:

- "Reference Class Forecasting" asks people consider previous, similar, tasks and present them with the data. Of course you need a) similar previous tasks to reference and b) accurate data which given peoples retrospective underestimation needs to be objectively measured somehow. (This is something I should read more about.)
- Observers estimates can be better than participants; I would suggest including observers, or asking estimates "to estimate for another team member" but I would shy away from having all estimates done by people outside the team. Partly because this removes some incentive effects and seems unfair.

- Manipulating people so they focus on obstacles rather than goals can produce later estimates but again (to my mind at least) could act to demotivate.
- "Unpacking tasks" into component parts may also help, indeed this might help people think about obstacles too (although one other study questioned whether unpacking would help.)

But improving estimation should not just be able improving the estimation process. Changing or influencing behaviour after the estimation has occurred can also help people finish earlier and therefore closer to their expected date.

- One technique was to have participants mentally practice doing tasks every day before undertaking the tasks.
- Another technique was specify where and when tasks would be done.

In both cases participants are being asked to mentally rehearse for their tasks. These might be akin to morning stand-up meetings.

More studies....

As is often the case when you start reading research studies each study leads you to more studies. There are some more studies I have not had a chance to read - in most cases because the studies are locked behind paywalls and the researches have not made provision for non-academics to access these papers.

What follows is a short list of studies I have not been able to read beyond the abstract but which I want to include for completeness.

If you don't want to be late, enumerate: Unpacking reduces the planning fallacy

(Kruger and Evens 2004)

Another set of experiments which show that if participants are asked to "unpack" a task, i.e. break it down into the steps they will perform, then estimation bias is reduced. This fits nicely with one of the papers I reviewed

previously (Francis-Smythe and Robertson 1999) which suggests that those with more control of their own work produce more accurate estimates. That paper also suggested that some of the effect may be due to better time management from setting ones own deadline.

The (Ariely and Wertenbroch 2002) study showed that self-imposed deadlines improved performance. It could be that by engaging with the task - via unpacking - individuals are setting a form of deadline in their estimate.

Also, the (Buehler, Griffin, and Peetz 2010) study showed that for closed tasks (and it seems several of the Kruger tasks might be considered closed, although not all) optimistic deadlines could improve performance. That too could lead to more accurate estimates.

However the (Buehler, Griffin, and Ross 1994) study suggested that observers, not actors, produced more accurate estimates.

Underestimating the Duration of Future Events: Memory Incorrectly Used or Memory Bias

(Roy, Christenfeld, and McKenzie 2005)

From the abstract it doesn't seem like this paper adds much to the research already reviewed from these authors. The abstract does contain a nice summary of these authors central argument:

> "People base predictions of future duration on their memories of how long past events have taken, but these memories are systematic underestimates of past duration. People appear to underestimate future event duration because they underestimate past event duration."

Biases and Fallacies, Memories and Predictions: Comment on Roy, Christenfeld, and McKenzie (2005)

(Griffin and Buehler 2005)

This is a comment piece not research, I include it here because it demonstrates that there is no consensus on why people underestimate. Roy et al argue one explanation while Beuhler et al argue another.

For the record I find myself more attracted by Beuhlers explanations and arguments.

The Role of Motivated Reasoning in Optimistic Time Predictions

(Buehler, Griffin, and MacDonald 1997)

Unfortunately I haven't managed to get a copy of this study but I have read the abstract and it is references by several of the other studies mentioned here. This study shows - perhaps unsurprisingly - that when individuals are motivated to complete a task estimates will exhibit a greater bias.

To quote from the abstract: "Monetary incentives for early completion led to optimistic predictions, increased attention to detailed future plans, and reduced attention to relevant past experiences."

Thanks to the researchers

I'd like to give thanks to all the academics - many of them listed above - who have arrange to be able to list their studies on their own websites. In a few cases this has required cunning, they provide draft or working papers while the final versions on locked up.

And to any academic who bemoans the disconnect between their research and industry I say: take steps to make your research more available

- Make versions of papers available where they are generally available, e.g. your website and not behind a Springer paywall
- Write in language which is accessible: reading these studies has required some serious effort, I understand why many give up

Further research

I would like to make a few suggestions for anyone inclinded to conduct further research in this field. Without answers to these questions (and others) there are still significant gaps. These questions may also be taken as point of caution over the work reviewed.

- Do all the mechanisms for inducing lower estimates result in similar changes for open and close tasks? Specifically: if a financial incentive is offered and a lower estimate produced do closed tasks still get done quicker and open tasks as before?
- What can we learn from source code control systems? For example, if every task (open or closed) is assigned its own branch, the creation of this branch considered the start time and the closing (or merger to truck) considered the end can we synthesis an accurate time tracking system?
- To what degree are software development tasks open and closed? Can we find ways of making open tasks closed, and if so can we improve estimation?
- What are the effect of changing the parameters of the task while performing the task?
- Does power also reduce retrospective estimates or is this phenomenon only apparent in forward looking estimates?

In reviewing all this research it occurred to me that there should be a body of estimation literature concerning software development, and even Agile/XP style velocity and Planning Poker. Indeed I have already identified several studies on estimation in software and, assuming I continue this endeavour, they will be my next stop.

It also seems to me that estimates by programmers could be a fertile area of study for both psychologists and computer scientists. I'd like to propose one experiment:

> Take a group of programmers, divide them into three groups and give them a set of programming tasks to be completed during the day. One group estimates in time (hours), estimate in abstract units (points), and on group doesn't estimate (no estimates).
>
> They are instructed to attempt all tasks in a predefined order (for consistency) and use a source code control system to check in their work as they go. They may even be given a test suit to program against, when they pass the tests they are done.

The experiment ends when the day is done or the programmers complete their tasks.

From the source code control system it should be possible to tell how much time is spent on each task (look at the checkin timestamps.) Now we can ask: which group completed the most work? Which group was fastest? Which was the best indicator of time spent the hours or points group?

As a further twist the experiment could be repeated with participants being asked this time to break the tasks down into smaller pieces before starting to program.

Conclusions?

The more I read of the estimation research the more potential twists and turns it seems to take. A couple of things do seem to be reasonably consistent:

- The Planning Fallacy holds: although the original paper was short on quantitive studies the subsequent research has supported the hypothesis
- Vierordt's Law holds: probably the oldest piece of research on the subject of estimation - or more correctly time perception - it has stood the test of time. Indeed one might argue that the Planning Fallacy is simply a consequence of this law.
- Vierordt's Law and Planning Fallacy hold prospectively (looking forward) and retrospectively (looking backwards)
- Anchoring is real: the suggestion of a time can influence ones estimates
- The things the industry calls "actuals" would be better called "retrospective estimates"

Beyond this I thing get a little more complicated. Drawing on the papers here and from my last review I am inclined to believe:

- Deadlines are more significant than estimates in determining when a task will be completed. Regular externally imposed deadlines are best of all.
- Multiple independent estimates (e.g. wide band delphi and planning poker) can be effective
- A number of practices (commonly followed in software development) can result in lower estimates, e.g. group estimation, the offer of financial reward and publicising estimates. Some, like financial rewards, are easy to eliminate, others like group estimates are probably necessary.
- Feedback can help improve the process but it might simply be providing an anchoring mechanism
- Closed tasks can be estimated more accurately than Open tasks, although some of the reason for this might be less to do with estimation than with motivation
- Open tasks (typical in software work) are more difficult to estimate (i.e. more variance) and early starts can increase the total amount of work done
- Authority, power, seniority and expert status can all lead to more optimistic, and therefore less accurate, estimates
- Starting early makes more work: pre-work, preparation, should be limited in the name of efficiency
- We should seek to make more tasks closed

One way in which all these experiments differ from real-life software development is that they deal with fixed tasks. The typical software tester and programmer faces a moving target. At any point in their work they may find the task has been changed, or that their assumptions of what is involved are wrong.

If we cannot estimate fixed, known, tasks with any degree of certainty what chance have we of estimating moving targets?

Personally

At the end - or possible breakpoint - of this literature review I have yet to find anything which substantially changes my opinion. I may be exhibiting confirmation bias here. Or it may be that the personal experience coupled with (largely second hand) research I have been exposed to was largely right.

I continue to believe the estimation process I have described elsewhere (Guide to iteration planning meetings - http://www.softwarestrategy.co.uk/static/dlgs/GuideToPlanningMee a combination of work breakdown, planning poker, velocity measurement and limited time horizons can work effectively.

I continue to believe estimation can be a useful tool in software developers but perhaps not in the way if is usually thought of. I believe estimation is a useful tool in the short run for both loading an iteration, eliciting requirements/specification detail and prompting design conversations.

I also believe that, due to the changing nature of the tasks we work on, estimation is not just about estimation, it is about bounding a task and limiting the work that is done.

References

Ariely, D., and K. Wertenbroch. 2002. "Procrastination, deadlines, and performance: self-control by precommitment." *Psychological Science* 13 (3).

Buehler, R., D. Griffin, and H. MacDonald. 1997. "The Role of Motivated Reasoning in Optimistic Time Predictions." *Personality and Social Psychology Bulletin* 23 (3): 238–247.

Buehler, R., D. Griffin, and J. Peetz. 2010. "The Planning Fallacy: Cognitive, Motivational, and Social Origins." *Advances in Experimental Social Psychology* 43: 1–62.

Buehler, R., D. Griffin, and M. Ross. 1994. "Exploring the 'Planning Fallacy:' Why People Underestimate Their Task Completion Times." *Journal of Personalty and Social Psychology* 67 (3): 366–381.

Buehler, R., J. Peetz, and D. Griffin. 2010. "Finishing on time: When do predictions influence completion times?" *Organizational Behavior and Human Decision Processes* (111).

Francis-Smythe, J. A., and I. T. Robertson. 1999. "On the relationship between time management and time estimation." *British Journal of Psychology* 90 (3): 333–347. http://www.ingentaconnect.com/content/bpsoc/bjp/1999/00000090/00000003/art00002.

Griffin, D., and R. Buehler. 2005. "Biases and Fallacies, Memories and Predictions: Comment on Roy, Christenfeld, and McKenzie (2005)." *Psychological Bulletin* 131 (5): 757–760.

Hofstadter, Douglas R. 1980. *Godel Escher Bach: An eternal golden braid*. Harmondsworth: Penguin Books.

Jones, C. 2008. *Applied Software Measurement*. McGraw Hill.

Kruger, J., and M. Evens. 2004. "If you don't want to be late, enumerate: Unpacking reduces the planning fallacy." *Journal of Experimental Social Psychology* 40 (5): 586–598.

Roy, M. M., J. S. Christenfeld, and C. R. M. McKenzie. 2005. "Underestimating the Duration of Future Events: Memory Incorrectly Used or Memory Bias." *Psychological Bulletin* 131 (5): 738–756.

Roy, M. M., and J. S. Christenfeld. 2007. "Bias in memory predicts bias in estimation of future task duration." *Memory & Cognition* 35 (2): 557–564.

Weick, M., and A. Guinote. 2010. "How long will it take? Power biases time predictions." *Journal of Experimental Social Psychology* 46.

Zackay, D., and R. A. Block. 2004. "Prospective and retrospective duration judgments: an executive-control perspective." *Acta Neurobiol Ex* (64): 319–328.