Load Factor

The latest book which KentBeck authors wins over an earlier book which he wrote. People change their minds about concepts as they gain experience. It is important to embrace such change and go with it when it serves well. Example: pg 61 and 62 of PlanningExtremeProgramming ISBN 0201710919 says: "We used to use load factor a lot in planning, but since then we have learned that it's easier to just use velocity."

PaulChisholm asks about Consistent Estimation Factor in ProjectManagement, and AlistairCockburn asks about its arithmetic in LoadFactorArithmetic. See also MeasuringProjectVelocity.

WardCunningham taught RonJeffries the core of this technique.

Here's what we do on C3. We ask each engineer to estimate each task in PerfectEngineering Days. These are those days you have, few and far between, when no one interrupts you and you are really flowing along. How many perfect engineering days will it take to do that?

Here is where I have problems with the approach. It is based on an a project that did not finish. Perhaps it did not finish because it did not consider the tough stuff during design (what design in XP?). For example, let's say that we wish a to design an abstract class for a Business Object. I think that we should consider ahead of time a lot of things like: strong type checking, data binding, events, collections of Business Objects, persistence, validation, rules, etc. etc. If you don't, you need to start new many times; this I think is more inefficient. These comments about LoadFactor are very good, thanks. (Fritz Schenk).

Once a week, maybe twice, the Tracker (TrackerRole) goes around and for each person, for each task, asks how many perfect engineering days he got in on the task, and how many there are to go.

[How can the trackee report "perfect" days expended on a task? He knows for certain only the "calendar" days it took him. Is he supposed to adjust his answer by his estimated "inefficiency" (for lack of a better term), or the current LoadFactor?]

If not many days have gotten in, Tracker might ask what happened. The answer might be "I was sick", "Marie interrupted me a lot", "I had to help Dave", or something. Tracker will know right away if it makes sense.

If the task was originally 5 days, and there are 3 days in and 5 to go, Tracker knows there's a problem. Usually Tracker will bring it to my attention and I'll dig in and see what is going on. Sometimes it's an attempt to work from inadequate stories ... more often the developer / team have gone off track and should DoTheSimplestThingThatCouldPossiblyWork.

We use a LoadFactor to estimate the number of real days in an engineering day. Currently on C3 there are 2.5. We have been prepared to go to an individual LoadFactor, but it hasn't been necessary. The only person whose LoadFactor seems to vary from the standard is the Coach. My factor is somewhere around 10, that is, I can get a half-day of engineering done in a week.

When we sign up for tasks for a three-week iteration, there are typically 15 real work days. Divide by 2.5, there are 6 perfect engineering days per developer. Sign up for six days of work.

TimMackinnon - Is this really 6 days per developer or 6 days per pair? Or do you just mentally compensate for pairs? There is a paragraph in ExtremePlanning that talks about calculating a total by dividing by the number of developers and multiplying by the LoadFactor. I find it very confusing.

Sometimes upper managers get upset at the factor. We can usually say what makes the LoadFactor what it is. Some of it is estimation error, some is support, some is useless meetings called by the upper management. We usually don't brook much interference with the factor, because it is measured. It's a result, not a control variable.

We say "We are producing software at this rate. There are this many more cards to do, requiring this many more engineering days. Therefore the delivery date is MMDDYYYY. It's hard to argue with the numbers. (Sometimes they try, but that's another story. See also NegotiateEstimates.)

"Tracker" (TrackerRole) has been mentioned a few times. Can someone say some more about this role in XP?

That's it above, what else would you like to know? Tracker goes around a time or two a week, asks each developer how she's doing, listens to the answer, takes action if things seem to be going off track. -- RonJeffries

Is Tracker a manager, consultant, or a role that developers assume periodically? I suppose that my question is really about ExtremeRoles. From what I've read so far, it appears that everyone works together as peers, PairProgramming. the design work is shared, communication is maximized, but there is some unilateral decision making authority in place: "this is the way we do it." Are there roles that have been assumed? If so, defacto or dejure? Or, is it the case that with business driving the UserStories, teams jell to the point that the right decision in a context of values is obvious, and you reach consensus rapidly?

While all of that above sounds rather hypothetical, I can bring it to the experiential level by asking whether in C3 the developers who worked on the original aborted Smalltalk payroll attempt are still there. If so, what is the quality of their relationship with the newer hires? If you were to look right now would you see a distinction which would map to any notion of development staff roles in XP?

-- MichaelFeathers

Tracker at C3 happens to be the nominal manager of the group. (He doesn't manage the team but he does go to the management meetings, and he signs my invoices.) I would do it myself, but I seem to be constitutionally incapable of it. For a while one of the other contractors did it. It really doesn't matter who does it, except that they have to be non-threatening. The point is just to have a point of checking-up and noticing that something is going off track. It takes a few hours a week.

The C3 developers from Chrysler who were on the original payroll are there. They basically self-selected to continue. Some of the original contractors stayed on for a while. All are gone except one that hired in to Chrysler. The relationship with newer hires is wonderful. (All new people are contractors, as it turns out.) The only distinction you will see is that when it comes time to be in production support, we have one of the first-year folks paired with one of the second year folks. That is, we always require one long-term Chrysler person on support.

The contractors are generally more experienced Smalltalkers than the Chrysler people, with specialized knowledge of Unix and/or GemStone. The Chrysler people know the system (and payroll) better. They're an amazingly well-behaved team with essentially no politicing or other maladaptive behavior.

It's a very jelled team with good skills at reaching consensus. That said, there are of course people whose opinions are respected more than others ... but there's no sense of seniority or hierarchy. -- RonJeffries

Worth noting that Microsoft's MSF uses a LoadFactor in its estimates too. They call it "buffer percentage" - see

http://www.microsoft.com/solutionsframework/AppDev/DevSkedChecklist.doc (BrokenLink). They use a factor of 1.3 but allow lots of fudge space for schedule changes, developer dependencies, meeting time, and other imponderables. In short, they don't keep as good a track of the real ratio as XP does, and it seems likely to me that their true LoadFactor would be about 5 if you factor in the time they waste producing non-deliverables. -- PeterMerel

I find the 1.3 factor interesting. It is about 1/2 of 2.5, and the 2.5 contains the pairing partner's time. So they match, from my perspective. They also match what one manager told me - to only expect any person to be able to work productively about 60-70% of their time. So, interesting, interesting. Also interesting, because I myself, and others I meet, use multipliers of 1.3, 3 or 9, depending on something, we haven't figured out what, yet, but having to do with the number of people involved and the granularity of the task list being estimated. -- AlistairCockburn

---

I once heard about 'the rule of Pi', that stated you should take initial estimated and multiply then my Pi (3.14159 and then some). Generally discarded by the people I had to work with as too drastic. I suppose this gets pretty close to the ideal number of 2.5, with a little padding. -- StijnSanders (that would be RuleOfE then?)

It's a version of the 'multiply by three rule' that's fancied up with a beloved constant.

Pi and e are both good. Irrational numbers, for an irrational task. And by giving yourself a few more digits of approximation, you get yourself time for that "one last task".

See: ProjectVelocity or VelocityVsLoadFactor

- LoadFactor is a deprecated XpProcess (it seems that the PointOfViewing has changed when we substitute "Velocity" for a multiplyingConstant

---

CategoryExtremeProgramming

Last edit July 28, 2009, See github about remodeling.