The Ganssle Group

Perfecting the Art of Building Embedded Systems

The Ganssle Group
PO Box 38346, Baltimore, MD 21231
Email info@ganssle.com

© 2000,2001, 2002 The Ganssle Group

We schedule dysfunctionally.

Published in ESP, February 2001

Schedule Madness

An inhabited planet shows on the forward sensor array, the first in this sector. Monitoring very odd teleradio chatter – mostly entities being unbearably unkind to each other on something called "Jerry Springer" and paranoid ballistic missile radar – we learn this place is called "Earth". The scanners zoom in, seeing continents resolve into cities, rivers into estuaries, R&D parks to individual buildings. Roofs are transparent to the tensfrap beam; we continued to home in closer and closer till our point of view is over the shoulder of a worker-entity, hunched over a keyboard and screen. Closer, closer and we see a single open window on the CRT, something labeled "Microsoft Project".

A mouse dances furiously in the worker's hand, moving triangles around in a blaze of motion even our quad eye-stalks can barely follow. Oddly, the last triangle on the screen – something labeled "ship date" – stays put.

Ah – then we understand. This is a computer game of some sort. The object seems to be to make lots of complicated interrelated events happen in a fixed time. A variant of chess, perhaps, an idea confirmed as we listed to the entity's mutterings. It seems the enemy is a dark force named "management". Listening more intently we learn that the goal is to convince the opposing generals that the game can be completed in a timeframe they've fixed. Things get spicier when the enemy gets their turn; then they'll sometimes arbitrarily decrease the allotted project time.

What an odd game! We turn to each other, shrug, neither of us understanding how either side could be having fun. Perhaps all of these entities are as cruel to each other as we see on that Jerry Springer broadcast.

This seems more an exercise in frustration than gaming thrills. Humans – they're so hard to understand. We decide to abandon Earth and return to Epsilon Theta 4 where the Bzorkians long ago evolved beyond such petty conflict.

Dysfunctional Scheduling

The proliferation of personal computers in the early 80s caused a frenetic search for the post-spreadsheet "killer app". For a while it seemed that project management (PM) software might fill that niche. Those dreams faded, though these software packages did form a basic part of most developers' toolkits.

In the pre-PM days creating and maintaining schedules was a nightmare. It was like writing code in hex instead of assembly; every little change meant manually moving all timelines and task dates around. So PM software was clearly a boon to managers.

Or, was it? I can't fault the PM apps themselves, but almost without exception it seems people use these tools in entirely dysfunctional ways. The vast majority of engineers scheduling a project start with a capriciously-imposed deadline (the end date) whose location on the timeline is fixed for all time. That triangle is somehow anchored in place by forces that go far beyond reason and reality.

With a locked-in end-date the rest of the scheduling process seems aimed at making a plan that's believable, rather than one that's right. "Yeah, sure this looks good; better give that task a week which leaves only three days for integration; I guess that'll work."

No, actually, scheduling a project this way will not work. Ever.

When properly used, PM software lets us decide how tasks interrelate, so we can identify and optimize critical paths before starting the project. It shows just how long each event line will take, ultimately giving us a reasonable prediction about development time. Without sheer luck in the form of a generous though capricious end date, working backwards from a fixed time is doomed to failure.

So, in my opinion, we waste our time creating a schedule meant to hit a mandated ship date. We're engaging in the art of creative lying, trying to contort the facts into an impossible deceit aimed at making the boss happy now, deferring the inevitable consequences till later. It's like the kid who hides the report card till Monday, so the parents won't ruin his weekend. There will be hell to pay, but we do our best to delay the consequences till a later date.

Napoleon said: "Any commander-in-chief who undertakes to carry out a plan which he considers defective is at fault; he must put forth his reasons, insist on the plan being changed, and finally tender his resignation rather than be the instrument of his army's downfall." If we, or our bosses, lived to that standard of honor than perhaps all schedules would be accurate. Or the unemployment office would have to open a special line for embedded systems developers.

Inadequate Designs

Imposed ship dates are only part of dysfunctional scheduling.

I'm a great believer in using a defined software design process. In using the best possible software engineering practices. For decades gurus have taught us the right ways to build programs.

Unfortunately, too often these sages are so shielded by the forgiving arms of academia that what they preach is simply impossible for those of us in the trenches to use. Not that they are wrong; just impractical.

For example, the philosophy behind traditional estimation schemes is that size is an accurate predictor of the resources and time needed to develop a product. At the risk of gross simplification, most scheduling methods convert software size (measured in some variant of lines-of-code or function points) to person months. For example, Boehm's COCOMO model predicts time by an equation of the form:

$$\text{Time} = \text{constant} * (\text{thousands of source lines})^{\text{exponent}}$$

COCOMO and its derivatives go to great lengths to define the constant and exponent, which are each functions of many other characteristics of the project. Often glossed over is the critical variable: the size of the program in lines of code (or function points, or any of a dozen other measures). How many of us know to any degree of certainty this value when doing a schedule?

It seems to me there are two influences that make estimating a project's size problematic. The first is obvious: we do it badly. Under the best of circumstances it's very hard to come up with a correct, or even close, size estimate. In my experience it seems most folks under-guess by about a factor of 2. That's a tremendous error intolerable in any other business endeavor. For instance: a healthy company might run a 10% profit margin. An error costing just 5% of the business's gross revenues slashes profits in half. 10% errors yield losses and layoffs. Only .coms can afford (or, at least tolerate in some mysterious manner) the 100% error rates we routinely experience.

The second problem is that only rarely do the developers have a chance to create an accurate estimate. Let's face it: when the boss wants a schedule he wants it *today*. Or tomorrow. End of the week at the latest. Yet to create an honest size approximation requires us to do a fairly complete system design, which might take weeks or even months.

The boss wants to know a ship date, generally with a margin of error approaching zero, and gives us an impossibly short deadline to come up with the date. Is there any surprise 80% of embedded systems are delivered late? Of *course* estimates are always way off!

Yet the boss might be responding to pressure from above. To determine if a product idea is profitable it's important to know when it will be released and the cost of the engineering. It's hard for him to justify spending months and lots of money creating a system design when he's only interested in a cost number for trading off business issues.

One approach is a best-guess estimate created in the time allowed, but mediated by broad error bands. "Give us 6 months, minus 2 plus 8." Unhappily, as illustrated in several Dilbert cartoons, the boss usually hears "probably 4 months, no longer than 6".

A final problem is poor product specifications. As a young engineer my boss once told me: "let's do a new xxx product. Just like the last one, but smaller and faster. More accurate, too. How long will it take?" The sound of my jaw crashing to the floor barely fazed him. My estimate was exactly as precise as his product definition. It's hard to create a comprehensive spec, which often takes much more time than the boss cares to invest just for the sake of scheduling.

So we have all the elements of disaster:

- The specification we start with might be inadequate or practically non-existent.
- We have too little time to do the real design needed to create a meaningful completion date
- Future viability of the product and maybe the company stems us getting both the schedule and the product right
- The boss does not want to hear the truth

Again, of course we're always late!

The quixotic schedule dance between developers and managers takes an even more dysfunctional dive when the capricious end date is set to meet a big show or conference. The show. Aren't you sick of hearing about "the show" and how the product simply *must* be ready for demos by then? The boss seems to think that missing that date means utter doom; the competition will introduce their product, ultimately prevailing in the industry! Or this is the only chance we have to get the press needed to make that big splash the marketing folks demand.

There's always a show looming, one whose importance to the company gets trumpeted with ever-increasing frenzy as its date nears.

Now, I'm not discounting the importance of shows and of marketing. What bothers me is that we ignore the truth behind show demonstrations. We then create an impossible deadline, and deliver a horribly flawed product as a result.

Go to the show. Watch the attendees strolling along the floor. Stop at a few booths and see how the salespeople actually put the product through its paces. That demo might last 10 seconds! 15, tops.

How much really has to work?

Even more enlightening, get an exhibitor's badge and visit the show floor the day before the exhibits open, during booth set-up. There's nothing more fascinating than watching the salespeople removing bubble wrap from the latest new product. Their eyes are full of terror, terror induced by too much experience showing off products that are just not ready. So many defects, so many system crashes; the best sales folks are those with hides so thick that no bug-induced failure creates the least embarrassment.

It seems to me that the show needs a product that demos reliably, and that passes the 10 to 15 second test. You don't need 50 functioning features. It's far better to extract the few really important parts of the product, and design and test those so they're perfect.

Much more realistic scheduling puts the show as simply one deliverable of many; one with a few "wow" features that are utterly reliable.

Alternatives

One substitute to the conventional design-and-schedule-it-all-up-front approach is to develop incrementally. Start building the system based on your current understanding of the requirements and design tradeoffs, but continually refine both the design and the schedule. Issue intermediate releases, which all asymptotically approach the final desired product and give stakeholders a chance to see how the product works.

As we proceed from requirements to design to coding to test, at each stage the schedule estimate will grow more precise with narrower error bands. A 100% uncertainly at the outset will shrink once the specs are understood, the risks identified, and things start working.

Incremental development might be the most honest of all ways to build a product, as it acknowledges that the spec will be faulty, the design flawed, and that only through the process of actually making the thing will we figure out what we're doing. These truths may be ugly but are inescapable.

But how many managers are willing to tolerate the uncertainty of such a process? "Whatdoya mean the schedule's gonna keep changing? I need an accurate number *before* we start!" I do think that if we combine incremental development with an initial fuzzy estimate bounded by error bands we'll be as truthful and as accurate as we're likely to be. But few bosses seem willing to proceed without a hard and fast delivery date. Even when we all know that date is set in quicksand.

Instead we participate instead in a mutual delusion dialog, settling for an agreeable but clearly outrageous plan.

Destiny

A wise man once told me everything I need to know about my destiny. Someday I'll die. Between now and then everything else is up to me.

So too with bosses. Some day the boss will die, or be promoted, or quit, or move to sales (rather like dieing, I suppose). At some point *you'll* find yourselves in the leader's role. *You'll* be the one tempted to impose an arbitrary ship date, or give inadequate design time.

What action will you take at that point? Can you resist the pressure from the CEO, stockholders and customers to do things right? Or, will you cave to the overwhelming forces that drive people to management by fiat?

It's up to you.