



monday, february 5, 2007

Side effects (Here Be Dragons) and the Hubble

In the olden days before satellite imagery overlaid with Google Maps, cartographers had a rough job. They were supposed to draw an accurate reference from sketchy, incomplete information. While they did the best they could, there were always unexplored areas they couldn't know about. Since blank areas on maps don't inspire confidence (or sales), they filled them in with lovely renderings of horrific sea creatures. "*Here be dragons.*" Of course we haven't explored here, they said. *We know* it's dangerous.

This didn't particularly encourage future explorers to venture there, either. If the mapmakers said there were dragons, well - there were dragons. You weren't supposed to go outside the known boundaries; you'd fall off the edges of the world.

However, when you fall off the edges of your world, you usually fall into a new (and larger) one. Take that metaphor and spin it around into systems thinking today. Instead of *here be dragons*, we have the much subtler *factors beyond our control* or even *unexplained side effects*. Can't do anything about dragons; they're just there, and if something flies into their realm, best let it go. We've roamed outside the known boundaries of our field.

Much has been said about the need to de-silo the working world. The words "interdisciplinary collaboration" are usually followed by the buzzword "innovation" somewhere. Executive books and management tomes (even engineering management tomes) are sprinkled with vague aphorisms like "listen to customer feedback" and "have a clear chain of accountability." The case study of the Hubble telescope, while impressive and clearly written, has one major flaw. **It has no stories.** As Boris said, "I agree with everything [the Hubble case study] said, but it doesn't help me."

It's like saying *here be dragons*, followed by instructions to *be cautious around dragons* and *avoid getting your ship sunk by dragons*. What we need are more specific stories: "...and there was the one time we found this beast of a dragon, purple wings and scaly tales, who flew in from the sky to attack our boat... except Jorge discovered they were vulnerable along the underside of their wings..." If a purple dragon is ramming at your ship, you want to know how people have gotten rid of them in the past, not that you "should avoid purple dragons."

Then there's dragon practice. As Andy said, "the more you can build complex projects that don't matter, the better. Maybe when you run a million-dollar project, then you won't mess up." (He was referring primarily to POE class, which is notorious as a learning experience in systems project failure - individual parts will work, but the whole will not.)

Me? I think the Hubble did a reasonable job of recognizing their dragons post-mortem. I'm impressed the Hubble worked in the first place. Precision parts and pre-internet distributed design? No matter how you slice it, that's a tall order, and they managed to pull it off.

posted by mel at 1:46 am 0 comments

blog archive

▼ 2007 (6)

▼ February (1)

Side effects (Here Be Dragons) and the Hubble

► January (5)

about me



Mel

Mischief-maker since 1986.

[View my complete profile](#)

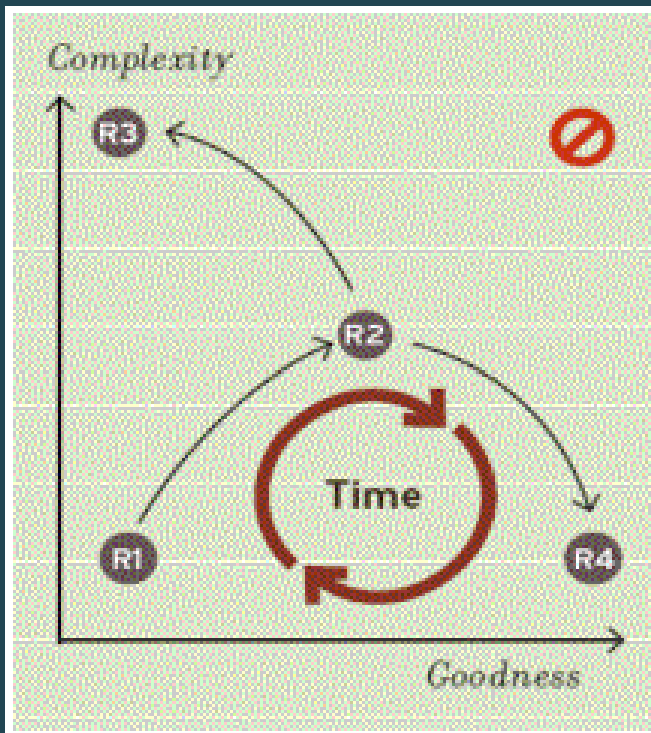
monday, January 29, 2007

What's the system of systems?

Part of the discipline of systems engineering is quantifying the inherently unquantifiable. By abstracting the fuzzy messiness, you turn it into non-fuzzy clean-ness (or so you hope), but you also drop a lot of information in the process. That's why Brian says that all models are false. They are. I tried making a model of the information flow between profs, NINJAs, and students for the ECS class. It was terrible. I thought I could describe the basic interactions, but I ended up going "but I can't put that in if I put *that* in!" until I conceded that Brian was right.

So I asked him what the system of systems was, since systems engineering is itself an engineering system, fulfilling the standards (by one route of defining it) by being complex, large-scale, interdisciplinary, and an open system. The recursive meta-ness elicited groans from Chris. We tossed around ideas for a while, landing on George's definition of systems engineering as an inherently dynamic discipline because it grouped previously complex technologies into simple systems.

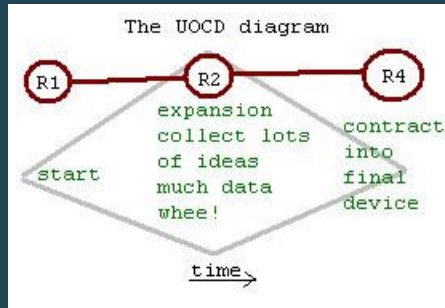
For instance, the making of steel... becomes part of the system of a ball bearing... which can be used on the shaft of a car... that gets sensors bolted on to run the DARPA grand challenge. Each advance in technology depends on the previous generation's technology becoming "common" enough to be used as a component. It's almost fractal-like complexity with boxes within boxes within boxes, only spiraling outwards and forward in time. At this, Chandra got to the board and drew this diagram, which USAF Maj. Dan Ward had presented to the seniors last week. It's called the Simplicity Cycle, and it describes product development.



The oversimplified version is that projects start simplistic (R1) with very little information, balloon up into complexity (R2), and then either blow up into complicatedness (R3) or find ways to turn that complexity into a new, richer kind of simpleness. Systems engineering, Chandra said, does exactly that; turn complex things into simple ones. And they're aided by the flow

of time, which tends to push things from simple (R4) to simplistically obvious (R1), allowing engineers to use them as subcomponents in new and more complicated (but eventually simple) things.

At this point, someone (Boris, I think) said "It's like UOCD!" And indeed it is. If you flatten out the path that goes from R1 to R2 to R4, you get this, which is pretty funky.



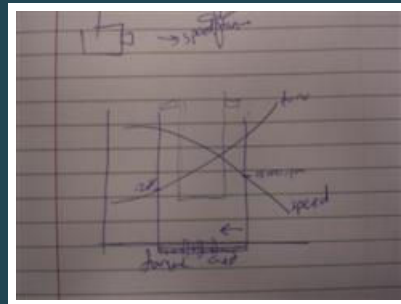
posted by mel at 3:27 pm 0 comments

RC planes vs Air traffic control

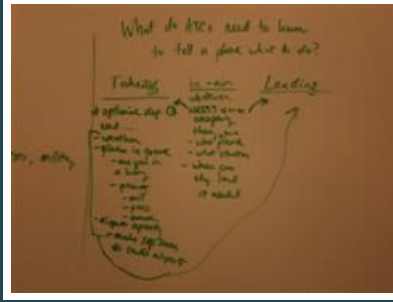
Brian had us do two quickie systems exercises. First we had to find a motor for an RC plane, then we had to design an air traffic control (ATC) system, and we got 5 minutes for each round. "You won't finish," he said. We didn't. But we learned a few things.

First, there are a range of systems engineering problems. The RC plane involved a lot of physical constraints (due to the laws of thermodynamics, electromagnetism, and that sort of stuff). It was easy to quantitize and optimize even if we had to make decisions about tradeoffs between the electrical, mechanical, and thermal systems. The ATC wasn't was fuzzier, "designer-y", and harder to defend our answers because there wasn't an universal understanding of what was "good." In fact, one of the main characteristics of systems that I'm finding is that the "right answer" is very dependent on your definition of the problem, moreso than other fields I've experienced.

Secondly, it was amusing to compare different approaches to the same problem (we worked in pairs). For instance, Chris and Andy drew diagrams for the RC plane problem that reminded me of the minimax theorem and linear optimization in game theory (...er, among other things). Very pictorial, more holistic and simultaneously mathematical.



In contrast, Marco and I went to the board and started listing out criteria for the ATC. We would have gone on to do a flowchart of decisions if we hadn't run out of time. More sequential and linear, broken into subcomponents, in written format, and... almost programmatic.



The RC plane, Brian said, was more characteristic of traditional engineering education. Mm, math, optimize. The ATC was more Olin-ish, although we still do a lot of RC plane stuff. "And the ATC was more exciting," Boris pointed out, "because it has the potential to save lots of lives, whereas a toy plane..." However, even as the world moves towards systems problems, engineering education still remains behind to work on traditional engineering ones.

That's okay. After all, you need to learn addition before you get into ring theory or Gauss-Jacobi sums. We're always going to need the RC plane optimizers, but now we need the ATC ones too.

posted by mel at 3:04 pm 0 comments

thursday, january 25, 2007

Last semester's responses: What is systems engineering?

I read the other papers (just because) and tried to summarize them as best I could. If any of the paper authors are reading this post, I apologize for the butchery... it's hard to summarize an emerging field in a 3-page-paper, and even worse to squeeze it into 1-2 sentences.

Zach Brock: As a new discipline, it's hard to tell what systems engineering is; is there a "calculus" that systems engineers can be taught, or is it a collection of skills you're inherently good or bad at?

Luis Diego Cabezas: Systems engineers are the ones working on the project at the highest level of abstraction. We've traditionally taken "systems engineers" from other engineering disciplines, but we need to start training systems engineers in their own discipline from the start.

Cathy Murphie: Systems engineering is a black box; you don't know what goes on inside it, but what comes out of it is a solution that balances processes and people to satisfy both external (design requirements, environmental factors) and internal (inexperienced project team members, too-small budgets, short schedules) constraints. The toolbox of a systems engineer is lifelong learning.

Mark Penner: Systems engineering uses a top-down perspective to focus on the interfaces between components in order to make the whole more efficient.

Mike Siripong: The "universal" toolbox of systems is organizational management; gantt charts, system diagrams, etc because specific technical tools vary wildly from system to system. Systems engineers have a big picture perspective that lets them lead projects, especially in interdisciplinary debugging.

Matt Tesch: Systems engineers allow their teammates to specialize in their fields by taking over the "big picture" problems and facilitating communication and interfacing between specialty groups when appropriate.

Dan Rice: Systems engineering is the coordination of all the components of a system, and its "calculus" is high-level thinking; feedback loops, mathematical models, and other tools that will mature along with the field. Since systems engineers need to see the big picture, it makes sense to let them roam outside the traditional hierarchy so they have the freedom to do that, and for them to have "people skills" so they can work with many different groups at once.

Lee Edwards: Systems engineering is the interdisciplinary study of interactions of two or more technical components in order to solve human problems.

posted by mel at 11:22 am 0 comments

George Jemmott: The engineering of modern technical systems

We were asked to read at least one of the papers on "What is systems engineering?" written by the students in last semester's Systems class and give a brief summary and commentary. This is my take on George Jemmott's, which I mostly agree with.

Systems engineering is not fuzzy, nor is it all "people skills," nor is it management. It's *engineering*.

Since systems are complex arrangements of subsystems, all technologically assisted actions constitute systems. Since the subsystems will change as technologies and the cultures they exist within progress, the exact technical definition of a system can't be pinned down to specific disciplines, tools, or even mental models to some extent.

Systems engineering involves understanding, creating, maintaining, and otherwise working with these evolving complex *technical* systems. You can understand a system when you understand its subcomponents, either through prior direct experience or rapid learning facilitated by indirect experience, so systems engineers typically have plenty of years logged in a variety of different things.

posted by mel at 11:03 am 0 comments

Systems Engineering Start

This is a blog for my notes on the MetaOlin independent study that six of us are doing in Fall 2006. We will be looking at Olin from a bunch of different perspectives, our first module being Systems Engineering (for the "10k feet up" view, as Brian Bingham said). Brian gave us some readings to kick things off.

Robert W. Lucky, Bozos on the Bus (IEEE Spectrum, 1996)

One sentence summary: We're blindly wandering in the dark and have no idea of what's coming ahead... but neither does anyone else, and this levels the playing field.

Question: Is the advantage in the new world system (whatever the heck that means) and the creation of new systems (products, etc.) skewed towards the young? Since young people tend to be much more used to "being bozos" on account of not knowing enough to be much of anything else, we're apt to adapt better to a world where everyone's thrown into bozo-hood, much like being blind during a nighttime power outage. I have a hard time believing this; experience and the wisdom of years is usually transferrable to different situations.

Daniel Hastings, The Future of Engineering Systems: Development of Engineering Leaders

Summary: Systems are all around us - education, healthcare, government, etc. These systems are becoming increasingly technologically

enabled (telecommunications, the internet...), but most engineers haven't learned how to work with systems of this magnitude that also interact with nontechnical, more sociological things. Since there are many possible views of the same complex system, Systems engineers have to be able to synthesize many different perspectives at once.

Systems engineering typically focuses on things with the following properties

- Technologically enabled - there are components that require solid engineering backgrounds.
- Large scale & complex - wide-reaching, far-ranging, and plenty of parts.
- Dynamic uncertainty - things should be changing in this system, and it should not be strictly predictable (fairly easy to get this when the system is large scale and complex).
- Interaction with nontechnical factors - there are people in the world, and we need to account for them.
- Emergent behavior - the whole is greater than the sum of its parts; some things will come out that you never predicted.

I was particularly fascinated by the paper's insistence that we need more *rigor* and *mathematics* and *quantitative modeling!!!* in systems education. It's almost as if they're trying to make sure that systems engineering is a "real" discipline, not a "fuzzy, soft thing" that "real engineers" will dismiss as hand-wavy.

This reminds me of a conversation I had with MIT anthropology professor Dr. Susan Silbey about "engineer's arrogance," which is what UOCD at Olin tries to cure. We assume that since we know about technology, and we're people, that we can make technology for people without learning about them. We don't need humanities! We just *know* what to make.

Anyhow, as much as I love math, I hope that systems doesn't get sucked into the "if it isn't quantitative, it isn't *real engineering*" trap. There are many ways to solve a problem, and numbers are only one.

The other part that I was struck by was the repeated echoes of "this discipline is not mature." That's exciting. As Lucky's paper put it, we're all bozos on this bus. Bill Strachan from IBM told me that systems engineering right now is around the same place that computer science was a generation ago - it's drawing on all these other disciplines, but it's more than the sum of *their* parts... in other words, **systems engineering is in itself a system**. It fits all the above criteria, after all.

posted by mel at 10:05 am 0 comments

Subscribe to: [Posts \(Atom\)](#)