Lecture 35 — Library Card

Jeff Zarnett
jzarnett@uwaterloo.ca

Department of Electrical and Computer Engineering
University of Waterloo

November 14, 2015

Library Card

Some advanced engineering design concepts.

Book one: *The Mythical Man-Month*.

Book two: *The Logic of Failure*.

Book three: *The Human Factor*.

Part I

The Mythical Man-Month

The Mythical Man-Month

Sometimes during development, you might discover that your project is behind or buggy.

Management might attempt to solve this by adding more resources (workers) to the project.

The Mythical Man-Month

Adding manpower to a late software project makes it later.

The Mythical Man-Month

Estimates are difficult to make and usually optimistic.

Our estimating techniques confuse effort with progress.
We assume programmers and monte are interchangeable.

When a schedule slip is recognized, the natural tendency is to add more programmers.

It makes the problem worse!

The Mythical Man-Month

A man-month is defined as the amount of work one programmer can accomplish in one calendar month.

Cost varies by programmers and months, but progress doesn't.

It is therefore dangerous and wrong to try to measure a programming task using this unit.

The Mythical Man-Month

Workers and time can be traded off against one another only if there is no communication required between workers.

(Does this sound familiar?)

When more programmers are added, we have to consider training and intercommunication.

The Mythical Man-Month: Training

Training scales linearly (n) with the number of people.

Training consists of bringing a new programmer up to speed on the project and its related technology.

This is a one-time cost when someone joins a project.

A new programmer, no matter how intelligent, is not fully productive immediately.

The Mythical Man-Month: Intercommunication

Intercommunication scales at $n(n-1)/2$.
if each part of the task is separately co-ordinated.

3 workers require 3x as much intercommunication as 2.

4 require 6 times as much as 2.

The Mythical Man-Month: Intercommunication

Adding a programmer increases the communication costs.

This cost can outweigh the benefit the programmer brings.

Thus, the project gets later.

The Mythical Man-Month: Example

Suppose a task is estimated at 12 man-months.

It is divided up such that three are three programmers who will work for four months.

Milestones A, B, C, and D, 1 at the end of each month.

The Mythical Man-Month: Example

A delay occurs and milestone A isn't completed in 1 month.

They assume that only milestone A is incorrect and 9 man-months of work remain, and two months remain.

So 4.5 programmers will be needed.

Hire two programmers.

The Mythical Man-Month: Example

Assume it takes 1 month to train these 2 new programmers.

Developer time is consumed by training the new people.

Tasks must be re-partitioned and re-assigned.

Integration testing increases.

The Mythical Man-Month: Example

At the end of the third month, we would still have more than 7 man-months of work to do and five trained programmers.

The project is still late!

The temptation is to then add yet more people, but this makes things worse, not better.

No Silver Bullet

There is no single development, in either technology or in management technique, that by itself promises even one order-of-magnitude improvement in productivity, in reliability, in simplicity.

No Silver Bullet

The term “silver bullet” comes from the werewolf folklore.

Werewolves can be stopped by being shot with a silver bullet.

A metaphor meaning a straightforward, simple solution to what is otherwise a complex problem.

No Silver Bullet

Example: diet books.

Promise something simple as the solution to weight loss.

“Eat Goji Berries!”

It is, unfortunately, not that simple.

No Silver Bullet

It is unlikely that we will ever find a silver bullet.

There are probably no inventions that will do for software development what electronics did for computer hardware.

The hard part of building software is not the accidental complexity, but the essential complexity.

The hard part is the specification and design of testing the concepts, not the work of implementing it.

No Silver Bullet: Complexity

Software is very complex, and this is an essential property.

In physics we have simplified models for complex phenomena.
(e.g., Newton's equations).

Complexities (like relativity) can be ignored.

In software, the details, are the essential parts and cannot be abstracted away.

No Silver Bullet: Complexity

Much of the complexity is arbitrary.

It must conform to other software or hardware, designed by imperfect humans.

Your software has to run on Windows, and has to conform to all the details and oddities of Windows...

No Silver Bullet: Changeability

Software is much more changeable than physical objects.

Cars and buildings get upgrades infrequently, if ever.

If Ford comes up with a better version of a car, that goes into next year's model.

Software is changeable because it is not a physical artifact.

No Silver Bullet: Changeability

The fact that the cost of changing a building is high discourages arbitrary changes.

Successful software is changed because users want to:

- apply it to new problems; or
- adapt it to new hardware or operating systems.

No Silver Bullet: Tools

Our tools have improved to make attacking some of this complexity better, but none alone is the magic solution:

- high-level languages
- integrated development environments
- object-oriented programming
- verification

No Silver Bullet: Tools

breakthroughs have been promised for a long time, but never seem to come through:

- artificial intelligence
- automatic programming
- graphical programming

Maybe these will arrive at some point.

No Silver Bullet: Solution

The most powerful tool we have is to have great developers.

(Which is why you are in this class.)

Part II

The Logic of Failure

The Logic of Failure

Tanaland is a fictional region in West Africa.

Used for a simulation.

Participants in the study get dictatorial power.

Goal: improve the inhabitants' lives over 10 years.

The Logic of Failure: Tanaland

Fields are not very productive, partly because of the rodents.

The obvious solution is to exterminate the pests.

At first, this results in an increase in the fields' productivity.

The Logic of Failure: Tanaland

Then, insect populations increase (they feed on the fields).

Larger predators begin attacking the cattle.
There are no longer the mice and rats to feed on.

Eliminating the rodents in the end has no effect
or it may have actually made the situation worse.

The Logic of Failure: Tanaland

The critical insight from the simulations:
We cannot do any one thing in isolation.

Any change has multiple effects.

Some of them unpredictable.

The Logic of Failure: Definitions

Positive Feedback: an increase in a variable produces a further increase in that variable.

A decrease produces a further decrease.

Example: Rabbit population (in isolation).

The Logic of Failure: Definitions

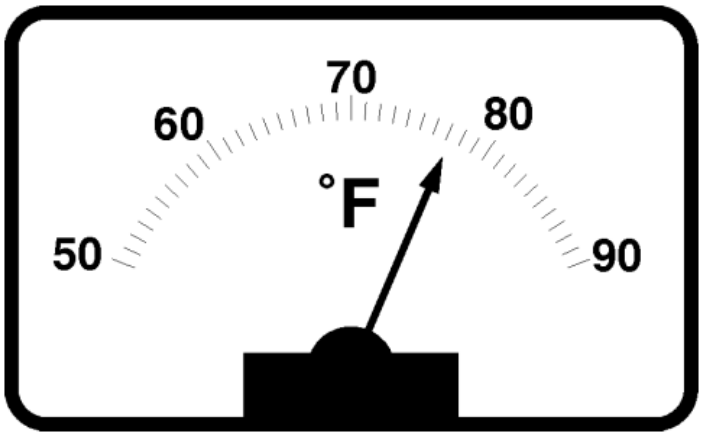
Negative Feedback: an increase in one variable produces a decrease in another and vice versa.

This maintains equilibrium.

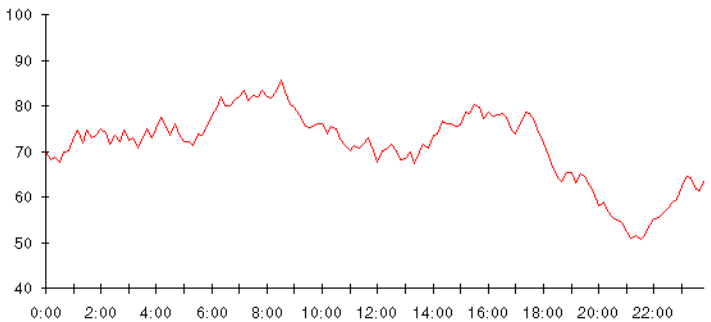
Example: Predator-Prey System.

The Logic of Failure: Feedback

Suppose we have a thermometer:

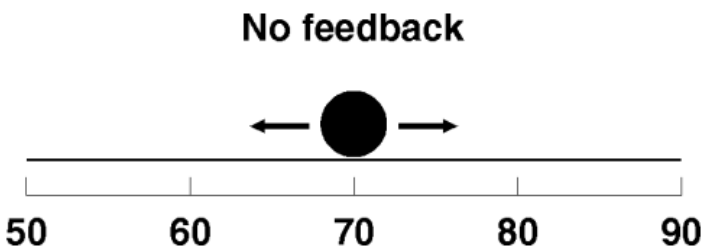


And the temperature varies as follows:



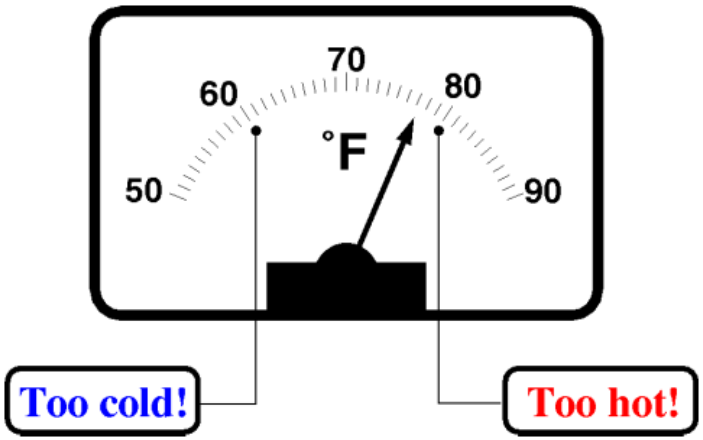
The Logic of Failure: Feedback

Our situation looks something like this:

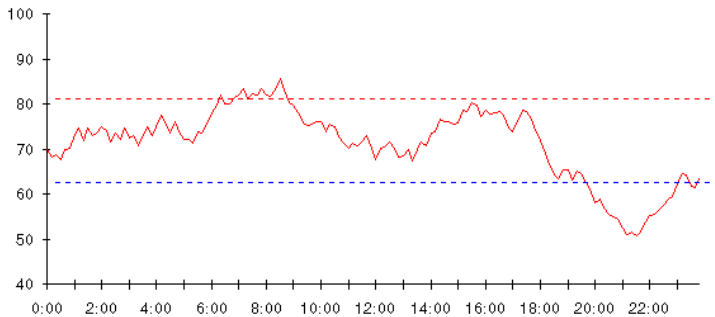


The Logic of Failure: Feedback

Suppose we have a concept of too hot and too cold:

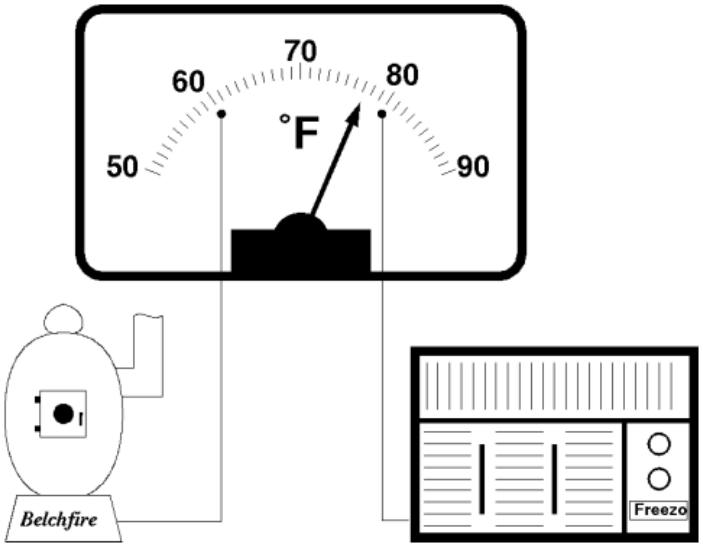


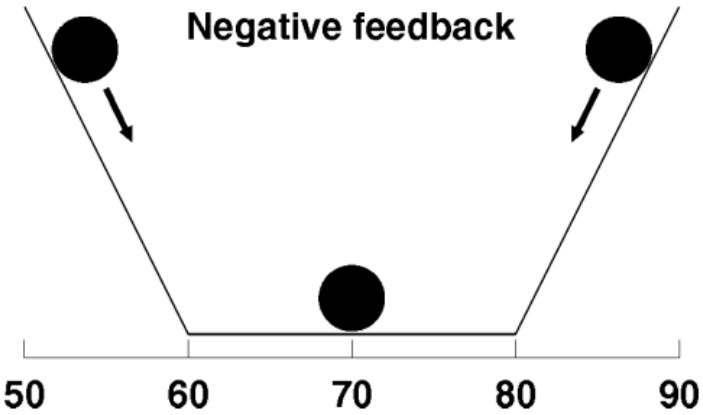
Representing these limits on the temperature chart, we see:



The Logic of Failure: Feedback

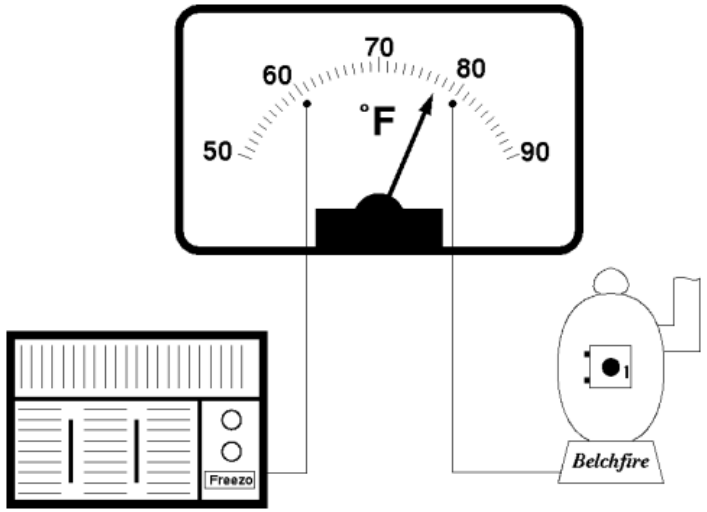
Let's set up a negative feedback system:

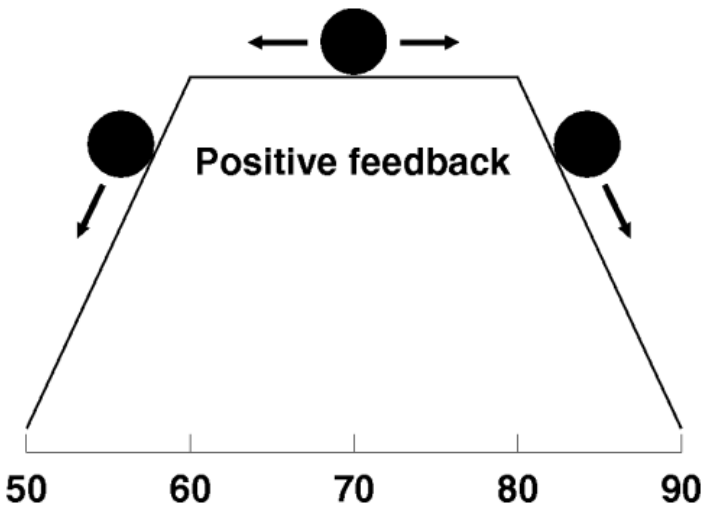




The Logic of Failure: Feedback

What if we hooked it up backwards?





The Logic of Failure: Definitions

Well-Buffered System: has many variables regulated by negative feedback.

It can absorb many disturbances without becoming unstable.

However, buffers are limited and may eventually be exhausted.

Predators can hunt the prey to extinction.

The Logic of Failure: Definitions

Critical Variables: interact with many other variables.

Altering one has a major impact on the state of the system.

The Logic of Failure: Definitions

Indicator Variables: depend on many other variables.

Exert very little influence themselves.

Help understanding of the current status of the system.

The Logic of Failure: Conflicting Goals

We can rarely focus on one goal at a time.

Analysis shows that shopping options and transportation links in a suburb are inadequate.

We can choose to improve one of these.

Choosing one will have an impact on the other.

The Logic of Failure: Conflicting Goals

Improve transportation, and people might shop in the city.
    Depriving local shops of revenue.

Or people from the city will come shop in the suburbs.

If we improve local shopping opportunities, people will shop locally and things will improve for local businesses.

The Logic of Failure: Conflicting Goals

By solving problem *X* we create problem *Y*.

If the appearance of *Y* is not immediate, it might not be obvious that solving *X* was the cause.

Solving *Y* could very well result in a recurrence of *X*.

The cycle begins again.

Example: headache and stomachache.

The Logic of Failure: Back to Tanaland

Let's try to help some inhabitants of Tanaland, the Moros.

Normal strategy: solve one problem at a time.

Most people start with combating the Tsetse fly.

The next most likely: drill wells to alleviate a water shortage.

The Logic of Failure: Back to Tanaland

Cattle live longer and the water shortage is eased.
They can have bigger fields of grass and more cattle.

They can sell some grain and cattle.

The next step most people take is to address medical care.
Infant mortality declines and life expectancy increases.

This results in an increase in population.

This can go three ways, but all end in the same (grim) way.

The Logic of Failure: Tanaland Scenario 1

Number of cattle exceeds the carrying capacity of the fields.

The cattle are not only eating the grass, but tearing it up.

The vegetated area is overgrazed and shrinks.

The smaller the amount of grass, the more desperate the hunger of the cattle, and the more they damage the grass.

The Logic of Failure: Tanaland Scenario 1

There is a solution – slaughter or sell the majority of the cows.

Seems like a “radical” step and people are reluctant to do it.

Once the grass is gone: the cattle will die, the Moros will be short of food, and they will starve.

The Logic of Failure: Tanaland Scenario 1

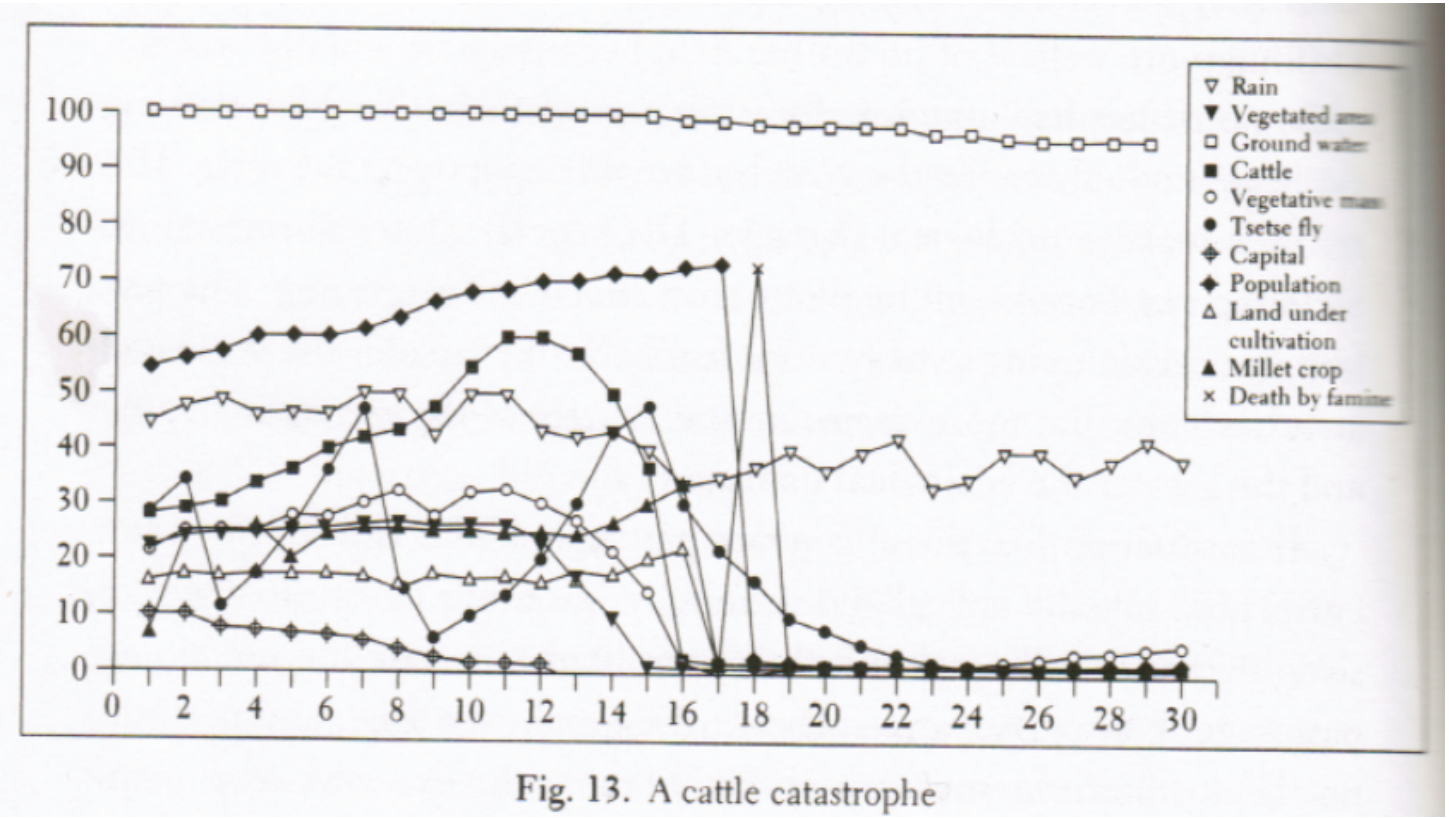


Fig. 13. A cattle catastrophe

The Logic of Failure: Tanaland Scenario 2

If cattle and human populations are kept in check, the system might seem stable for some time.

At some point, however, the wells start yielding less water.

The obvious solution is to drill more wells.

Might write it off as a temporary problem.

The Logic of Failure: Tanaland Scenario 2

Groundwater supply is being depleted.

Drilling wells to solve depleted wells is positive feedback.

More wells = faster depletion.

The Logic of Failure: Tanaland Scenario 2

Shortage of water means no irrigation of the grass.

Cattle start overgrazing, then die.

Then Moros are short of food and starve.

Just like scenario 1.

The Logic of Failure: Tanaland Scenario 3

Another participant just focused on healthcare.

Infant mortality decreased, life expectancy increased.

The population grows: more requirement for food and water.

If nothing is done, people will starve.

Attempts to fix it can turn into Scenario 1 or 2.

The Logic of Failure: Tanaland Conclusions

Participants dealt with things as independent mini-systems.
Not a single, complex, interrelated system.

They focus on the problem at hand.
Not what the consequences are or could be.

Future problems don't seem important.

Cause-effect relationships are unclear.
At best, revealed by experimentation.

The Logic of Failure: Lessons Learned

Good participants asked why-questions, not what-questions.

More interested in how the events were interrelated.

People who did poorly tended to take events at face value.

The Logic of Failure: Tips

Some tips for dealing with complex systems:

- State goals clearly.
- Recognize that it's not always possible to accomplish all our goals at once (goals may conflict).
- Establish priorities and change them when circumstances require.

The Logic of Failure: Tips

Some tips for dealing with complex systems:

- Learn a model of the system and use it to predict likely outcomes.
- Gather information at an appropriate level: neither too much nor too little.
- Know when to gather more information and when enough.
- Avoid excessive abstraction.

The Logic of Failure: Tips

Some tips for dealing with complex systems:

- Don’t hastily blame all events on one central cause.
- Avoid “knee-jerk” reactions.
- Analyze errors and learn from them.

Part III

The Human Factor

The Human Factor: Mercedes E320

In the Mercedes-Benz E320, drivers can check the oil electronically from the driver's seat.

Old, manual procedure:

1. Turn the car off.
2. Wait for the oil to settle.
3. Open the hood.
4. Find a rag to wipe the dipstick.
5. Identify the dipstick.
6. Wipe it.
7. Reinsert it.
8. Extract it.
9. Read it.

The Human Factor: Mercedes E320

New procedure in the E320.

1. Turn the car off.
2. Wait for the oil to settle.
3. Turn the ignition two notches to the right.
4. Wait five seconds.
5. Press the odometer reset button twice.

The Human Factor: Mercedes E320

Will users remember these set of steps? Probably not.

Most people will just perform the manual procedure.

Designers didn't deliberately build this system to be complicated and unintuitive.

The Human Factor: Designers

Designers are responsible for this.

Tech people like playing with gadgets and figuring out the cool features of something.

Most people just want to use the thing.

Most users are not like designers.

The Human Factor: BMW iDrive

The 2003 BMW 7 series had an “iDrive” system.

iDrive has between 700 and 800 features.

The Human Factor: BMW iDrive

It reminds me of software designers who become so familiar with the workings of their products that they forget actual customers at some point will have to learn how to use them.

Road & Track Magazine

The Human Factor: Designers

Unrealistic assumptions about physical processes?

We blame the designer.

Unrealistic assumptions about users?

We blame the users.

The Human Factor: Human Error

Physicians are expected to function without error, an expectation that physicians translate into the need to be infallible. One result is that physicians… come to view an error as a failure of character – you weren't careful enough, you didn't try hard enough… the perfectibility model: if physicians could be properly trained and motivated, then they would make no mistakes.

- Lucian Leape

The Human Factor: Human Error

Patient safety research shows that in most cases where patients are injured or killed are the result of honest mistakes.

Expecting them to be perfect is not going to happen.

Human nature is to find a culprit and blame that person.

The remedy is in design: create better systems to make it:

- Harder to make a mistake.
- Easier to do the right thing.

The Human Factor: Airplane Example

Pilots of several kinds of US aircraft retracted the wheels rather than the flaps.

Pulling up the wheels when the plane is on the ground is bad.

The problem is not just limited to new, inexperienced pilots.

The military considered this “pilot error”.

The Human Factor: Airplane Example

The levers controlling the landing gear and the flaps were next to each other and very similar in appearance.

Moving the controls is unrealistic in already-built planes.

As the short-term fix, add:

- A small wedge-shaped end to the flap controls.
- A small rubber disk to the wheel control.

The Human Factor: Airplane Example

Now a clear relationship between the controls & their function:

Wheels are round and made of rubber.

Flaps are metallic and wedge-shaped.

If a pilot touched the wrong one, it was immediately obvious.

Pilots using the new design stopped retracting their wheels while on the ground.

The Human Factor: Designing for Humans

We should respect what human bodies are like.

Don’t ask users to read a dial located 4m off the ground.

We also can’t ask them to lift 1000 kg unaided.

The Human Factor: Designing for Humans

Example: metal lathe.

Ideal user would be:
- 130 cm tall.
- Shoulders 60 cm wide.
- Arm span of 243 cm.

In other words, a very strange looking alien.

Somehow, machinists cope with using this lathe...
- Probably by moving around and contorting themselves a lot.

The Human Factor: Designing for Humans

Even if a design is physically a good fit, consider psychology.

e.g., the length of human short- and long-term memories.

Turning the steering wheel left should make the car go left.

Computer problems are often psychological.
Users can use the mouse, but remembering the command...

The Human Factor: Four Burners

A typical stove top has four burners on it.

Then we have four knobs to control each of those burners.

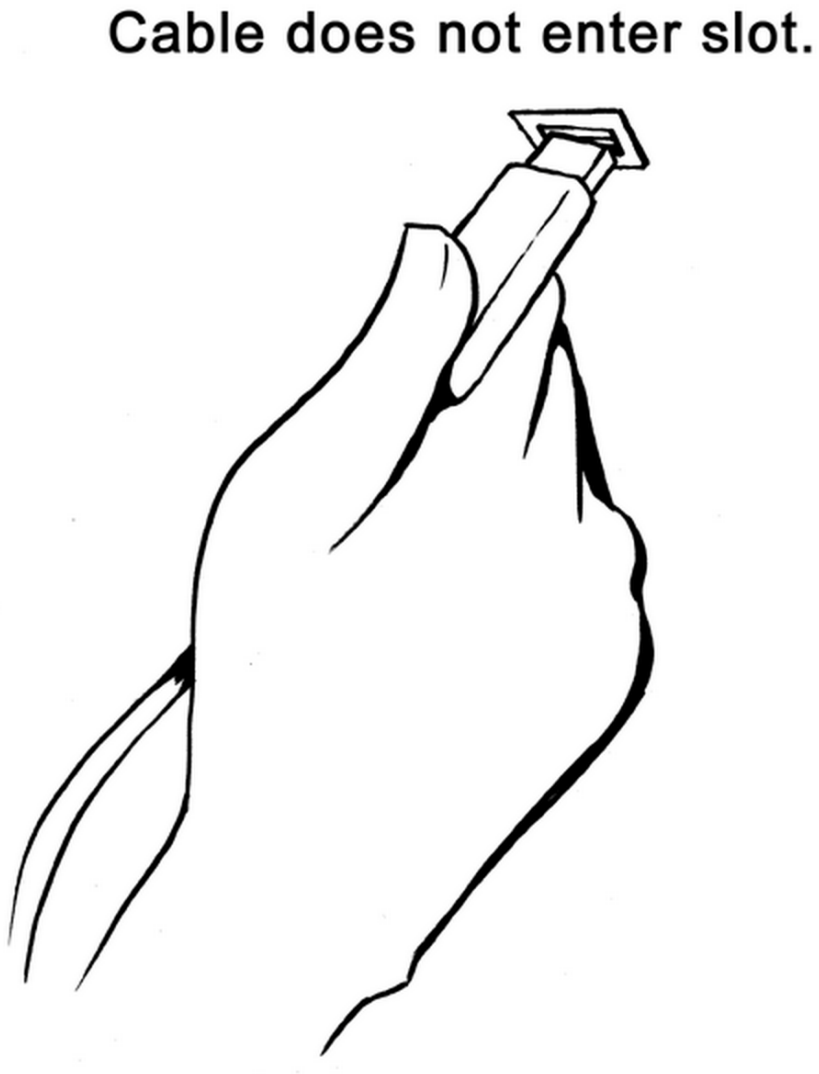
The burners are arranged in a square formation, one in each quadrant of the stove top.

The controls, however, are in a linear sequence.

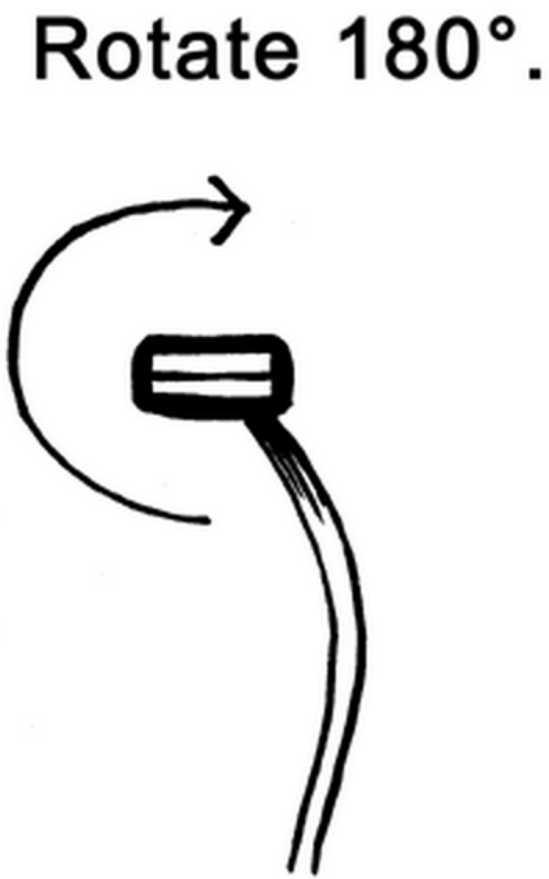
Which knobs control which burners?

The Human Factor: Four Burners

The Human Factor: USB Cables



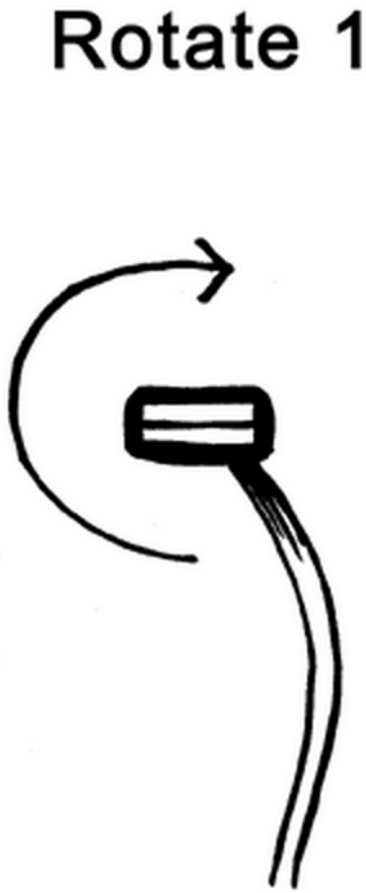
The Human Factor: USB Cables



The Human Factor: USB Cables

Cable does not enter slot.

The Human Factor: USB Cables



The Human Factor: USB Cables

Cable enters slot.

The Human Factor: Make Errors Impossible

[USB pictures from
`http://www.smbc-comics.com/?id=2388`.]

USB group has finally figured it out:
Type C connector is reversible. No “wrong” way.

Or laptop batteries: asymmetrical, obvious how it fits in.

The Human Factor: Feedback

Humans respond well to feedback.

Imagine a campfire.

It's obvious what happens if you step towards/away from fire.

Warning signs: discomfort or pain to warn of the danger of getting too close (getting burned).

The Human Factor: Feedback

Lack of feedback results in a lot of errors.

Not sure if button press worked? Press again.

Did that send it twice?

Example: TV with a slow turn-on.

The Human Factor: Feedback and Safety Rules

Breaking safety rules is usually reinforced, which is to say, it pays off. Its immediate consequence is only that the violator is rid of the encumbrance the rules impose and can act more freely. Safety rules are usually devised in such a way that a violator will not immediately be blown sky high, injured, or harmed in any other way, but will instead find that his life is made easier... The positive consequences of of violating safety rules reinforce our tendency to violate them, so the likelihood of a disaster increases.

Dietrich Dörner

The Human Factor: Conclusion

We require UW Eng students to take a class in economics.

It doesn't make anyone an expert in economics;
It exposes students to the relevance of economic factors.

Engineering projects will eventually involve interaction between technologies and humans.

Brief introduction to the human factor is the least we can do.