Principles of Mechatronic Systems Design

"Striking a Balance is Making Everybody Equally Unhappy"

Paul Pounds

11 March 2014

University of Queensland
But first…

Some house keeping
Housekeeping

- The magical team sort algorithm satisfied all exclusion requests with minor tweaking
  - Things got complex fast

- Friend requests were not included
  - You’ll thank me when you wish to throttle your teammates at 4 am the night before it’s due

- Teams are now posted on Blackboard
METR4810 Directed Graph of Woe

Key:
- Excludes
- Desires
House keeping

• Lab inductions and toolbox handout are Thursday, starting 1 pm in Hawken c404
  – Great time/place to meet up with your team!
  – Must complete the inductions to work in the lab
  – The best/only time to do 3D printer inductions
  – Wear appropriate footwear

• The rules and spec have been updated
  – Make sure you’re on the latest version
House keeping

• Lab sessions:
  – Tutors will be in the labs on Thursdays and Fridays
  – Morning contact sessions are times for holding meetings in, and for seminars and reviews
  – The lab will be accessible throughout the week, so you will have lots of opportunities to work
# Calendar at a glance

<table>
<thead>
<tr>
<th>Week</th>
<th>Dates</th>
<th>Lecture</th>
<th>Reviews</th>
<th>Demos</th>
<th>Assessment submissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3/3 – 7/3</td>
<td>Introduction</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>10/3 – 15/3</td>
<td>Principles of Mechatronic Systems design</td>
<td></td>
<td></td>
<td>Design brief</td>
</tr>
<tr>
<td>3</td>
<td>17/3 – 21/3</td>
<td>Professional Engineering Topics</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>24/3 – 28/3</td>
<td>Your soldering is (probably) terrible</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>31/3 – 4/3</td>
<td>By request</td>
<td>Progress review 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>7/4 – 11/4</td>
<td>By request</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>14/4 – 18/4</td>
<td>By request</td>
<td>Progress seminar</td>
<td>25% demo</td>
<td></td>
</tr>
<tr>
<td>Break</td>
<td>21/4 – 25/4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>28/4 – 3/5</td>
<td>By request</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>5/5 – 9/5</td>
<td>By request</td>
<td>Progress review</td>
<td>50% demo</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>12/5 – 16/5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>19/5 – 23/5</td>
<td></td>
<td></td>
<td>75% demo</td>
<td>Preliminary report</td>
</tr>
<tr>
<td>12</td>
<td>26/5 – 30/5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>2/6 – 6/6</td>
<td>Closing lecture</td>
<td>Final testing</td>
<td></td>
<td>Final report and addendum</td>
</tr>
</tbody>
</table>
Begin the terror

- Design Brief is due next week
- First progress review is two weeks after that

DON’T PANIC

But do get started!
Managing your stress levels

• Some students expect this class will be stressful – this does not have to be the case!
Some points for perspective

• Design brief is only two pages
  – A lot of thinking, but not a lot of “work”
  – I will talk more about it at the end of the class

• Progress reviews are to get you moving early, not to cause you stress
  – Make a reasonable effort and produce some tangible output and you’ll be fine (no ‘tales’)
Design brief design brief

• The objective of the design brief is to convince the client that the student has understood the problem, its scope, and its requirements and has developed insights into how the problem may be addressed. The student must provide a description of the aspects of the project he or she will be working on, a succinct analysis of the key design challenges of these aspects of the problem, the proposed approach to be undertaken in resolving them, and how the student’s proposed solution relates to other subsystems within the project. Students will be assessed on the thoroughness and insight demonstrated in the brief. The design brief is to be no more than two A4 pages.

Assignments are to be submitted through the Faculty of EAIT (Hawken Building 50) assignment chute and require an assignment cover sheet, available from https://student.eait.uq.edu.au/coversheets/
What you need to do

• Convince me you understand:
  – The problem
  – Its scope
  – Its requirements and constraints

• Describe:
  – The aspects you are working on
  – Key challenges of your subtasks (with analysis)
  – Your proposed approach
  – How it relates to the approach of your team
# Marking Rubric

**Design Brief – Individual Mark Sheet.**

<table>
<thead>
<tr>
<th>Grade Band</th>
<th>Task description (20)</th>
<th>ENGG/METR</th>
<th>Team number:</th>
<th>Design Integration (20)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Excellent</strong></td>
<td>Clearly describes the student’s part of the project – its scope and responsibilities. Tasks constitute a valuable, well-motivated and substantive contribution to developing a solution.</td>
<td>20 Problem broken down systematically. The technical challenges are highlighted and it is obvious how the design problems map to tasks. Well-considered specification provided.</td>
<td>30 A well justified, comprehensive breakdown of the approach to be taken. Tasks with resources and duration have been logically ordered and associated with logical milestones. Potential risks are reported where appropriate, with associated mitigation strategies.</td>
<td>20 Captures the most important design decisions and shows how the individual task integrates with the rest of the team. Part of a well-functioning design strategy.</td>
</tr>
<tr>
<td><strong>Very Good</strong></td>
<td>Sets out the work to be done, with some indication of scope or obligations. The student has an assigned task to undertake that will help the group.</td>
<td>18 Key challenges are recognised and described. Dependencies and prerequisites of major technical problems are discussed. A useful attempt at a specification is made.</td>
<td>27</td>
<td>18</td>
</tr>
<tr>
<td><strong>Good</strong></td>
<td>Broadly defined task description that maps to a major area of the project, but it less detailed about what is entailed or why that role was assigned.</td>
<td>16 Broadly defined task description that maps to a major area of the project, but it less detailed about what is entailed or why that role was assigned.</td>
<td>24</td>
<td>16</td>
</tr>
<tr>
<td><strong>Satisfactory</strong></td>
<td>Usable task and responsibility assignment provided. Unclear on boundaries or motivations.</td>
<td>12 Provides statement of what parts are difficult, without delving deeper into why. Little indication of thoughtful analysis. Expected performance requirements are minimal or missing.</td>
<td>21 A somewhat justified list of tasks with resources and duration has been ordered and assigned, with milestones. Risks are considered. Obvious thought has gone into assembling a solution plan.</td>
<td>21 Individual components are shown to work as part of a whole, with an indication of the interfaces between functional areas. Obvious communication in developing the solutions.</td>
</tr>
<tr>
<td><strong>Poor</strong></td>
<td>Some attempt made to provide task assignment and scope.</td>
<td>8 Simple restatement of the project challenges – limited insight into what makes the problem difficult.</td>
<td>18 A weakly justified list of tasks with resources and duration have been ordered and assigned, with illogical milestones. Some sense that thought was given the solution.</td>
<td>18 Functional breakdown across the project is awkward but each subsystem integrated usefully.</td>
</tr>
<tr>
<td><strong>Very Poor</strong></td>
<td>No attempt made at task scope.</td>
<td>4 No attempt made at analysis</td>
<td>12 An unjustified list of tasks with resources and duration have been ordered improperly. Denies apparent analytical consideration.</td>
<td>12 Collective design is haphazard and interfaces are nonexistent or illogical. Left hand has yet to meet right hand.</td>
</tr>
</tbody>
</table>

**Group mark component:** /100

**Comments:**

**Final mark:** /100 **Marker’s Signature:** **Date:**
Frequently Asked Questions

Surprisingly few questions so far
FAQ Roundup

• **When will we get access to c404?**
  – After the induction on Thursday.

• **If the cars are fully autonomous, how do we tell them to pit?**
  – You may have a single button to command the car to enter the pit lane and stop, and to command the car to exit the pit lane.
Back to that design thing…

Mechatronic Systems Design

Lolwut?
What is design, anyway?

• Design:
  – n. A goal or intention
  – v. The process or creating a plan

• In engineering contexts, design is both the process and the end product of formulating a technological solution to a problem
  – Engineering design is the application of scientific knowledge to satisfy a goals
Things that are designed

- Devices/structures
- Materials/chemicals/substances
- Processes/formulae
- Documentation/procedures/policies
- Specifications/guidelines/standards
- etc.

The common thread:

“Things that make stuff work”
Design is purposeful

• All design has some end goal
  eg.
  – Providing power to 100,000 homes
  – Moving 1 Megahuman across a city twice a day
  – Putting a catarrhine into orbit
Design is constrained

• All design is constrained
  – With no constraints – no limits – anything is achievable without need for planning
  – Design is needed when failure is possible

    contraque: no need to ‘design’ trivial things

• Common constraints:
  – Time
  – Budget
  – Labour
  – Materials
  – Energy
  – Logistics
  – Machine access
  – Technology
  – Political capital
A way of thinking about design

Design is the dual of critique:

Analysis

- Specification
- Deconstruction
- Parameterisation

Synthesis

- Implementation
- Constitution
- Optimisation

These are tools and philosophies for thinking about design, not a cookbook or an excuse not to use your brain.
Specifications

• The precise statement of exactly what the system will do
  – Often worked out collaboratively with the engineering team

• Precision is key
  – Reduce uncertainty as much as possible
  – Avoid “feature creep”
  – Clients often don’t know what they want
    (and sometimes change their minds halfway!)
Design brief

- Design briefs communicate the specification of an engineering problem to the engineer

- Describes what must be done
  - Provides precise requirements and constraints
  - Specify metrics to assess success

- Preliminary analysis of the problem
  - Theoretical design implications
  - Possible solutions, their risks, benefits, issues
Specifications are important

• **All of your design effort is geared to meeting the specification**
  – Avoids putting effort into unnecessary areas
  – Clear, complete specs’ lead to better designs
  – Doubles as a performance claim to customers

• In legal disputes, meeting the specification can be a critical defence against breach of contract
And now a word from our sponsor…

A brief detour into systems thinking
So what about systems?

• A system is a set of interrelated elements that interact as a whole
  – eg. transport networks, computers, duct tape, people, vehicles, weather

Systems Engineer maxim: “Everything is a system”
Systems engineering

• Engineering the whole, rather than the parts
  – Structured way of handling complexity
  – Defines the interfaces between major components of the system
  – Abstracts performance and robustness from individual parts towards the gestalt

• Design uses systems to modify the state and behaviour of other systems
The systemic approach

- Systems decompose into networks of subsystems with touching boundaries
  - Information crosses the system interface

![System Diagram]
System interfaces

- All systems have interfaces, designed or otherwise – eg. a box resting on a table
- Good design makes interfaces explicit
  - Software library APIs
  - ATX mounting holes and dimensions
  - IAC 240V/10A plugs and sockets
  - Road rules, air traffic control
- Interface designs are mutual agreements
  “If you function like this, I will work with you”
Systems get complex fast

- Large systems like space shuttles and skyscrapers and Olympic opening ceremonies are fiendishly complex
  - Even small systems (eg. cellphones) can be overwhelming in their entirety

How can we possibly understand them?
Deconstruction

• Specifications provide an end-point for the design process
  – Often work backwards to find a starting point

• Reductionism: break complex things down into understandable pieces
  – Find the essential parts of a system
  – Distil the problem into the core challenges
Deconstruction

• Deconstruction is like systems disassembly:
  Pull things to pieces to find out
  how they work together

• Many ways of doing this
  – Functional decomposition
  – Process flow
  – Causal dependency trees
  – And many more…
Functional* decomposition

- Hierarchical arrangement of processes
  - Interconnected network of “stuff that is done”
  - Not necessarily in order of operation

*Functional as in “Thing that make things work”, not as in `void main(void);`
Causal system dependency

- Systems can be thought of as a cascaded series of enabling functions

![Diagram showing causal system dependency]
Causal system dependency

• For example:
Causal system dependency

• For example:

![Diagram of causal system dependency](image-url)
Causal system dependency

• For example:

Increasing level of detail
Causal system dependency

- For example:

Increasing level of detail
Causal system dependency

• For example:

- Engines
- Launch stack
- Cryogenics

Increasing level of detail

Monkeys in space

Increasing level of detail
Causal system dependency

• For example:

[Diagram of causal system dependency with nodes labeled: Power bus, Launch stack, Cryogenics, Engines, Energetics, Trajectory, Monkey in space. Arrows indicate the flow from Power bus to Launch stack to Cryogenics to Engines to Energetics to Trajectory to Monkey in space.]

Increasing level of detail
Causal system dependency

• For example:

Increasing level of detail
Causal system dependency

- For example:

![Diagram showing causal system dependency with nodes for Power bus, Cryogenics, Launch stack, Engines, Sensors, Flight control, Energetics, Trajectory, and Monkey in space. Arrows indicate direction of dependency.]

Increasing level of detail
Causal system dependency

- For example:

```
Power bus
  v
Cryogenics
  v
Launch stack
  v
Engines
  v
Sensors
  v
Flight control
  v
Energetics
  v
Trajectory
  v
Life support system
  v
Monkey in space
```

Increasing level of detail
Causal system dependency

• For example:

- Power bus
- Launch stack
- Cryogenics
- Engines
- Energetics
- CO$_2$ scrubbers
- Sensors
- Flight control
- Trajectory
- Heaters
- Life support system
- Monkey in space
- O$_2$ supply
Causal system dependency

- For example:

![Diagram of causal system dependency]

- Power bus
  - Launch stack
  - Cryogenics

- Engines
  - Energetics
  - CO₂ scrubbers

- Sensors
  - Flight control
  - Trajectory

- Flight control
  - Trajectory
  - O₂ supply

- Monkey in space
  - Life support system
Aid to understanding failure

• We can also use causal system decomposition to understand failure

• Faults in the system propagate through directed graphs
  – Find the consequences of a failure
  – Work upstream to find the causes of a failure
  – Verify the “causal chain” to prove the system
Aid to understanding failure

• Consider a system defect:

- Engines
- Sensors
- Flight control
- Energetics
- Trajectory
- CO₂ scrubbers
- Life support system
- Heaters
- O₂ supply
- Launch stack
- Cryogenics
- Power bus
- Monkey in space

11 March 2014
Aid to understanding failure

• Consider a system defect:

Power bus ➔ Launch stack ➔ Engines ➔ Energetics ➔ Trajectory ➔ O₂ supply

Power bus ➔ Cryogenics ➔ CO₂ scrubbers ➔ Heaters ➔ Life support system ➔ Monkey in space
Aid to understanding failure

- Consider a system defect:
Aid to understanding failure

• Consider a system defect:
Aid to understanding failure

- Consider a system defect:

```
                Engines
                |          |
                v          v
Energetics        Sensors
                |          |
                v          v
Life support system
```

```
Launch stack

Trajectory

CO₂ scrubbers

Heaters

O₂ supply

Life support system

Monkey in space

Power bus

Cryogenics
```
Aid to understanding failure

• Consider a system defect:

- Engines
- Sensors
- Flight control
- Energetics
- Trajectory
- CO₂ scrubbers
- Life support system
- Heaters
- O₂ supply

Power bus

Launch stack

Monkey will not go to space today
Deconstruction informs design

• System decomposition tells us what functions are required by the design

• Successful design satisfies all prerequisites

• Robust designs provide redundant pathways
  – Identifies “weak links”
Constitutive design

• ‘Constitution’ turns functions into features
  – Tells you how the big picture fits together
  – The “broad strokes” of defining an approach
  – Functional requirements are a guide at best

• Designers have the most flexibility during design constitution – also the highest risk!
  – Bad structural decisions effect everything else
  – Constitution determines ~90% of system costs
Putting the pieces together

- Eg. Monkey capsule must provide oxygen, remove CO$_2$, and regulate temperature.
  - How big do the oxygen tanks need to be?
  - Can we use standard gas tanks or do they have to be custom-built?
  - How much separation is needed between the O$_2$ and the heater elements?
  - How do you get the monkey in there, anyway?
A jigsaw puzzle

• Engineering is just like solving a jigsaw puzzle with many pieces, except each piece costs $1000 and can be one of a dozen shapes or completely custom-made, and if you don’t solve the puzzle right, people die.

• In real life, engineering is often a process of try-and-see iteration
  – Sometimes, there is no “right” way.
Parametric design

• Even when you *do* have a clear high-level structural concept of your solution, there will usually still be many unconstrained variables

• The key dimensions/values/settings that describe a design are called the “design parameters”
Parametric design

- Parameters can be thought of as a series of twist knobs that adjust the design.
Design space

- We can think about a particular configuration of parameters as a point in “design space”
Design space

- Combinations of design parameters map to some realisable performance of the system
The fundamental problem

• Knowing how to set those knobs is difficult
  – Complex interactions between parameters
  – Competing design goals
  – Constraints on parameter space

• You will rarely satisfy all of your goals
  – You will NEVER meet all your ambitions
Design constraints

• University engineering problems are typically tightly constrained – they have only one “right” answer

• In the real world, engineering problems are either under-constrained (many solutions) or over-constrained (no solutions)
Design constraints

Impossible configurations

Outside of specification

“Design Space” (Parameter space)

“Performance Space”
Design metrics

- We can make cost and value functions to encode how “good” a candidate design is
  - eg. Aim to maximise propellant bang for buck, given parameters 1 to $n$, we might use:

  $$Value = \frac{\text{explosive force}}{\text{unit cost}}$$

  $$V = \frac{X(p_1, p_2 \ldots p_n)}{C(p_1, p_2 \ldots p_n)} = f(p_1, p_2 \ldots p_n)$$

  $$V_{\max} = \sup(f(p_1, p_2 \ldots p_n))$$
Metric space*

“Design Space”
(Parameter space)

“Performance Space”

“Metric Space”
(Value space)

*Not quite the same as (but not completely unrelated to) *Sur quelques points du calcul fonctionnel*
Metric space*

*Not quite the same as (but not completely unrelated to) *Sur quelques points du calcul fonctionnel*
Design optimisation

• For many systems, we can explicitly solve for the optimal design point
  – Estimate is only as good as your value function
  – Gradient of the value function is the design parameter “sensitivity”
Finding a value function

• How do we encode the utility of a design?
  – Highly subjective: what does “best” mean?

• Many tools for thinking about utility
  – Multi-criteria decision analysis
  – Pairwise comparison
  – Decision matrix method
  – Management by objectives

There is a whole field of “value engineering”
A quick example

- **Pair-wise decision matrix:**

<table>
<thead>
<tr>
<th></th>
<th>Safe</th>
<th>Low cost</th>
<th>Reusable</th>
<th>Easy to build</th>
<th>Payload capacity</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Safe</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>Low cost</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>Reusable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>Easy to build</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Payload capacity</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
</tr>
</tbody>
</table>

Prioritise by score
“D” for “X”

• How to choose a value function?
  – Design for performance
  – Design for manufacture
  – Design for reliability
  – Design for sustainability
  – Design for cost
  – Design for marketability
  – Design for obsolescence
Of course…

• It is relatively rare that a single value function can fully capture the complex give and take of a real-world design problem
  – Uncertain system constraints/assumptions
  – Uncertain system parameters
  – Uncertain system specifications (!)
  – Mutually exclusive goals
  – Conflicting agendas
  – Conflicting personalities
The most important truth in your degree

Engineering is the art of the trade-off
Methodological approaches

• Ok, that’s great – but how do we do this trade-off thing, exactly?
  – Lots of different ways!
  – Quite likely as many design processes as there are design engineers

• Here are just a few popular design process methodologies…
Alphabet soup

• LAST: Look, Ask, Simplify, Try
• ETC: Express, Test, Cycle
• PDP: “Product Design Process”
  – Define, Model, Implement, Verify, Support
• “Waterfall Model”
  – Requirements, Design, Implement, Verify, Maintain

(And many, many, many more – each more buzzwordy, cliché and feel-good managerial-speak than the last)
Cyclic vs linear models

Cyclic Model

Define → Support → Verify → Implement → Model

Linear Model

Requirements → Design → Implement → Verify → Maintain

Express → Test

Look → Ask → Simplify → Try
So which do you pick?

Every project is different

Slavish adherence to rigid procedure will (probably) just waste your time

Find what process works for you (and your company)
The common threads

1. Work out what to do
   – Specifications; the design brief – be precise
   – Understand the real constraints

2. Find a solution*
   – Iterate until you do

3. Make sure it works
   – Modelling, validation, testing
   – Critically evaluate ideas

*Wasn’t that the problem to begin with??
The synthesis step

- Constitution + Optimisation = Synthesis

The messy, complicated, creative, intuitive, frustrating, marvellous, deep, ineffable, often iterative intellectual challenge that lies at the heart of all brilliant engineering solutions

Art, *not science*: Anyone who claims they can teach you to do this is sadly misguided
<table>
<thead>
<tr>
<th>Step</th>
<th>Instruction</th>
<th>Distance/Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>74.</td>
<td>TAKE THE FERRY ACROSS THE LAKE.</td>
<td>go 2.8 mi</td>
</tr>
<tr>
<td>75.</td>
<td>CLIMB THE HILL TOWARD HANGMAN’S RIDGE, AVOIDING ANY MOUNTAIN LIONS.</td>
<td>up 1,172 ft</td>
</tr>
<tr>
<td>76.</td>
<td>WHEN YOU REACH AN OLD BARN, GO AROUND BACK, KNOCK ON THE SECOND DOOR, AND ASK FOR CHARLIE.</td>
<td>go 52 ft</td>
</tr>
<tr>
<td>77.</td>
<td>TELL CHARLIE THE DANCING STONES ARE RESTLESS. HE WILL GIVE YOU HIS VAN.</td>
<td>CAREFUL</td>
</tr>
<tr>
<td>78.</td>
<td>TAKE CHARLIE’S VAN DOWN OLD MINE ROAD. DO NOT WAKE THE STRAW MAN.</td>
<td>go 1T mi</td>
</tr>
<tr>
<td>79.</td>
<td>TURN LEFT ON COMSTOCK. WHEN YOU FEEL THE BLOOD CHILL IN YOUR VEINS, STOP THE VAN AND GET OUT.</td>
<td>go 3.2 mi</td>
</tr>
<tr>
<td>80.</td>
<td>STAND VERY STILL. EXITS ARE NORTH, SOUTH, AND EAST, BUT ARE BLOCKED BY A SPECTRAL WOLF.</td>
<td>go 0 ft</td>
</tr>
<tr>
<td>81.</td>
<td>THE SPECTRAL WOLF FEARS ONLY FIRE. THE GOOGLE MAPS TEAM CAN NO LONGER HELP YOU, BUT IF YOU MASTER THE WOLF, HE WILL GUIDE YOU. GODSPEED.</td>
<td>go ?? mi</td>
</tr>
</tbody>
</table>
Paul’s philosophy

Here are a collection of words that embody my own personal design philosophy

You may find them helpful.
They are not for everyone.
Philosophy of scope

• Understand the problem (work out what to do)
  – Specification
    • State with precision the problem to be solved
  – Requirements
    • Deconstruct the problem into a set of parameters/metrics the solution must satisfy
  – Background research
    • Identify gaps in your knowledge about the problem
    • Identify gaps in capability needed to solve the problem
    • Experiments where needed to gain information
Philosophy of synthesis

• The Synthesis Step (find a solution)
  – Ideate potential solutions (brainstorming)
  – Critically assess potential solutions (debate)
    • Constructively attack all ideas on merits
    • Promote resilient candidates, cull falsified candidates
    <Special guest appearance by "Design for X" thinking process>
    • Iterate as needed
  – Test candidates (analysis)
    • Demonstrate scientifically that the candidate will successfully solve the problem
    • Math, simulation, small-scale testing as appropriate to gain confidence that the candidate will work
    • Cull falsified candidates
  – Iterate as needed
– Choose fittest candidate solution to implement
Philosophy of ideation

- Vigorously debate and acid-test all ideas
  - No sacred cows: nothing passes untested
  - Be prepared to support your opinions with facts

- Idea Thunderdome (aka “Conceptual Darwinism”)
  - Two ideas enter, one idea leaves

- Don’t be afraid of maths
  - It won’t bite, and is a powerful ally once tamed

Some people are not comfortable voicing opinions or making themselves heard, but sometimes you need to stand up for your ideas! Solid analysis is your sword and your shield.
Philosophy of execution

• Implementing the solution is not actually a design step
  – But thinking about how the solution will be implemented is!

• Validate the solution
  – Testing is a lifestyle
  • Material test
  • Batch test
  • Spot test
  • Subsystem test
  • Integration test
  • All-up test
  – Critically assess real-world performance viz requirements
  – (Optional: Refine solution)
Recommended reading

• “The Existential Pleasures of Engineering”
  – Samuel C. Florman

• “The Design of Everyday Things”
  – Donald A. Norman

• “Materials and Design”
  – M. Ashby and K. Johnson
Questions?
Tune-in next time for...

Professional Engineering Topics

or

“Stuff they should have taught you at university, but didn’t”

Fun fact: Thirty two monkeys have been launched as part of various space programs – most recently by Iran on 31\textsuperscript{st} January 2013.

Nineteen survived.