Principles of Mechatronic System Analysis and Design

or

"Striking a Balance is Making Everybody Equally Unhappy"

Pauline Pounds

3 March 2019
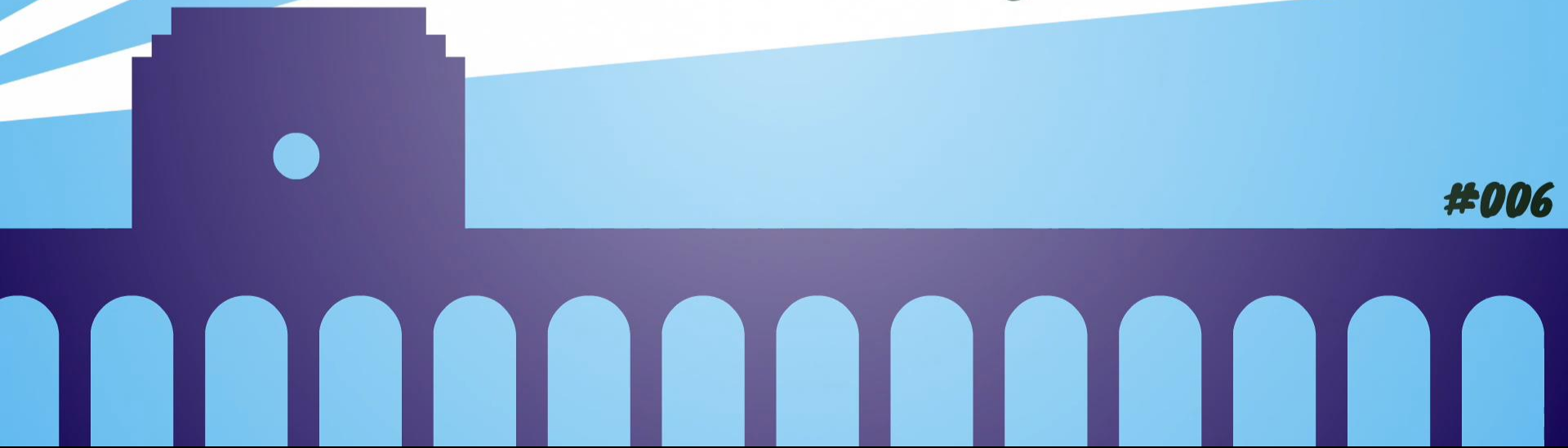
University of Queensland

But first…

A valuable message about safety…

Secondly…

Some house keeping

House keeping

- Problem analysis
  - Assessment, the first - but not the last

- Induction session
  - Do not feed the wild 3D printers

- Team allocations
  - The return of the magical algorithm

Calendar at a glance

| Week	Dates	Lecture	Reviews	Demos	Assessment submissions
1	24/2 – 28/2	Introduction			
2	2/3 – 6/3	Principles of Mechatronic Systems design			Problem analysis
3	9/3 – 13/3	Previous years deconstruction case studies			
4	16/3 – 20/3	Professional Engineering Topics	Progress review 1		
5	23/3 – 27/3	PCB design tips			
6	30/3 – 3/4	Your soldering is (probably) terrible			
7	6/4 – 10/4*	Introduction to firmware design	Progress seminar	25% demo	
Break	13/4* – 17/4				
8	20/4 – 24/4				
9	27/4 – 1/5			50% demo	
10	4/5* – 8/5		Progress review		
11	11/5 – 15/5			75% demo	Preliminary report
12	18/5 – 22/5	No Monday lecture (whew)			
13	25/5 – 29/5	Closing lecture		Final testing	Final report and reflection

6

Begin the terror

- Problem Analysis is due *Friday*

- 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!
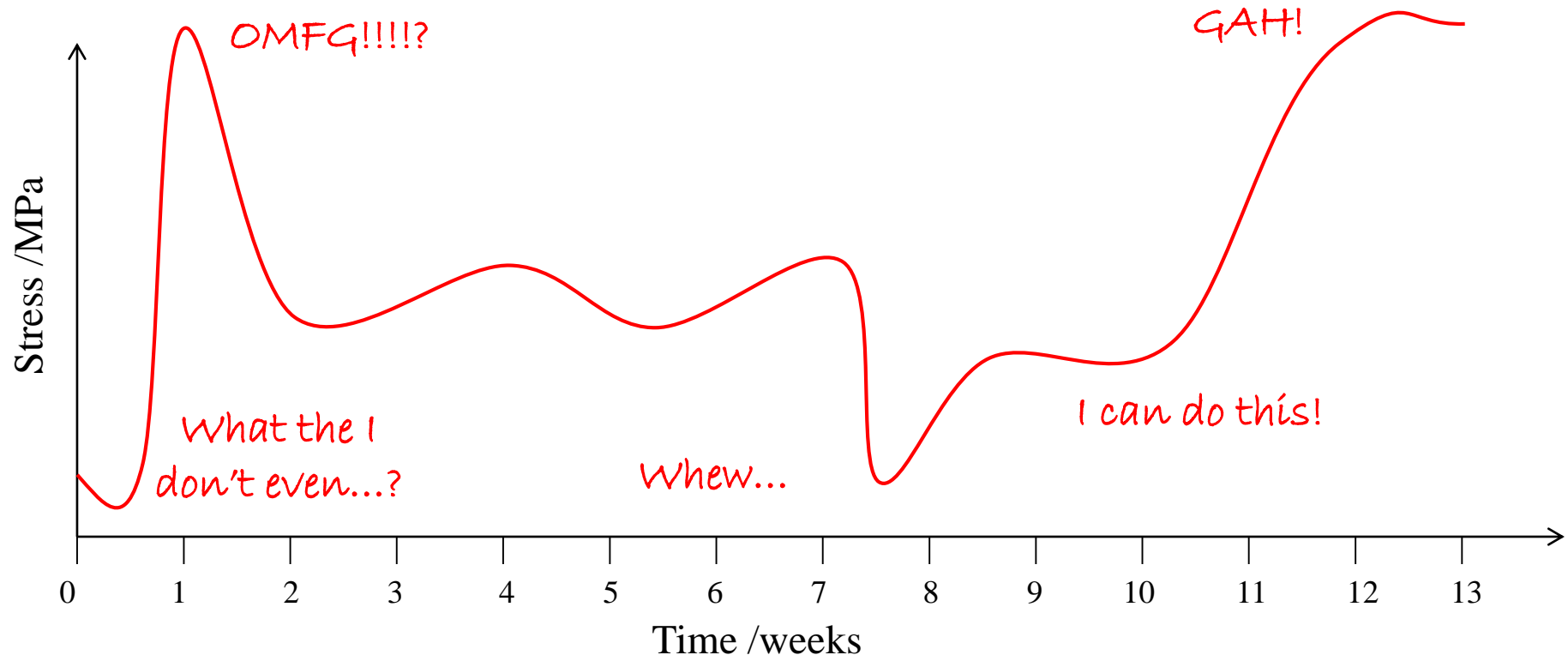
Some points for perspective

- Problem analysis is only two pages
  - A lot of thinking, but not a lot of "work"

- 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')

Problem analysis

Due March 6th – 10%　　　　　(2 pages max)

- Break down the design problem, determine its scope, requirements and constraints.

  – But <u>do not</u> regurgitate the design brief

- Describe the key underlying engineering design challenges to be solved.

- Present a candidate solution, and explain how your approach will overcome them.

  – Analysis is golden.

What are "engineering challenges"?

- ## The hard parts!
  - The things you need maths for!
  - No, just because you *might* be able to hack it, doesn't mean you don't need maths!

- ## Why can't you just buy one of these off the shelf?
  - The things you need an engineer for!
  - Why can't just anyone make one?

Mediations on scope

- "Scope" is the control volume of the problem – what's important, and what isn't
  - Things that must be considered are 'in scope'
  - Things that you don't need to care about are 'out of scope'

Out of scope　　　　　In scope　　　　Out of scope
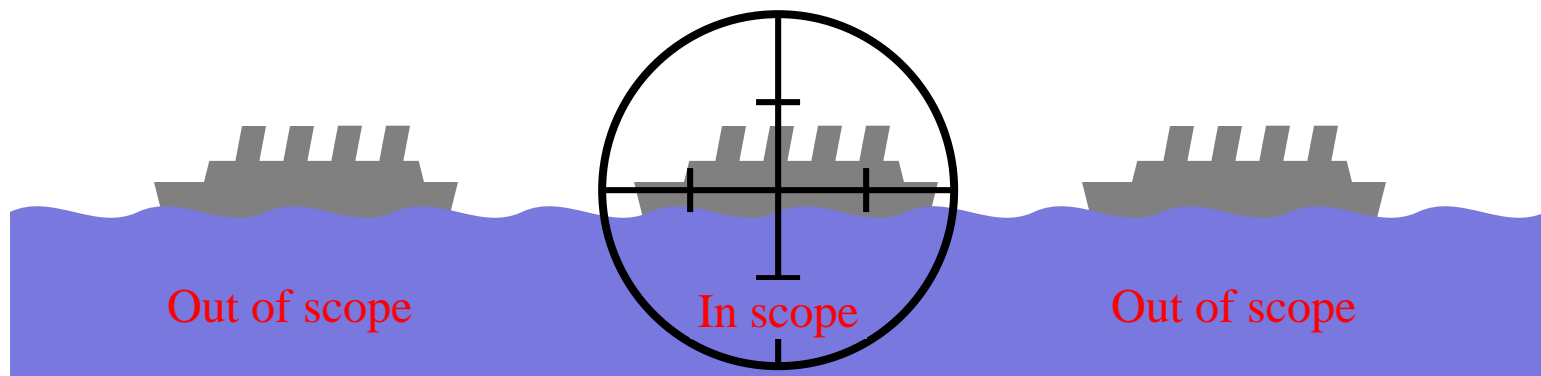
Lab orientation sessions

- *Tomorrow*, Wednesday 4th of March, 2pm in Hawken c404

Sign up here:

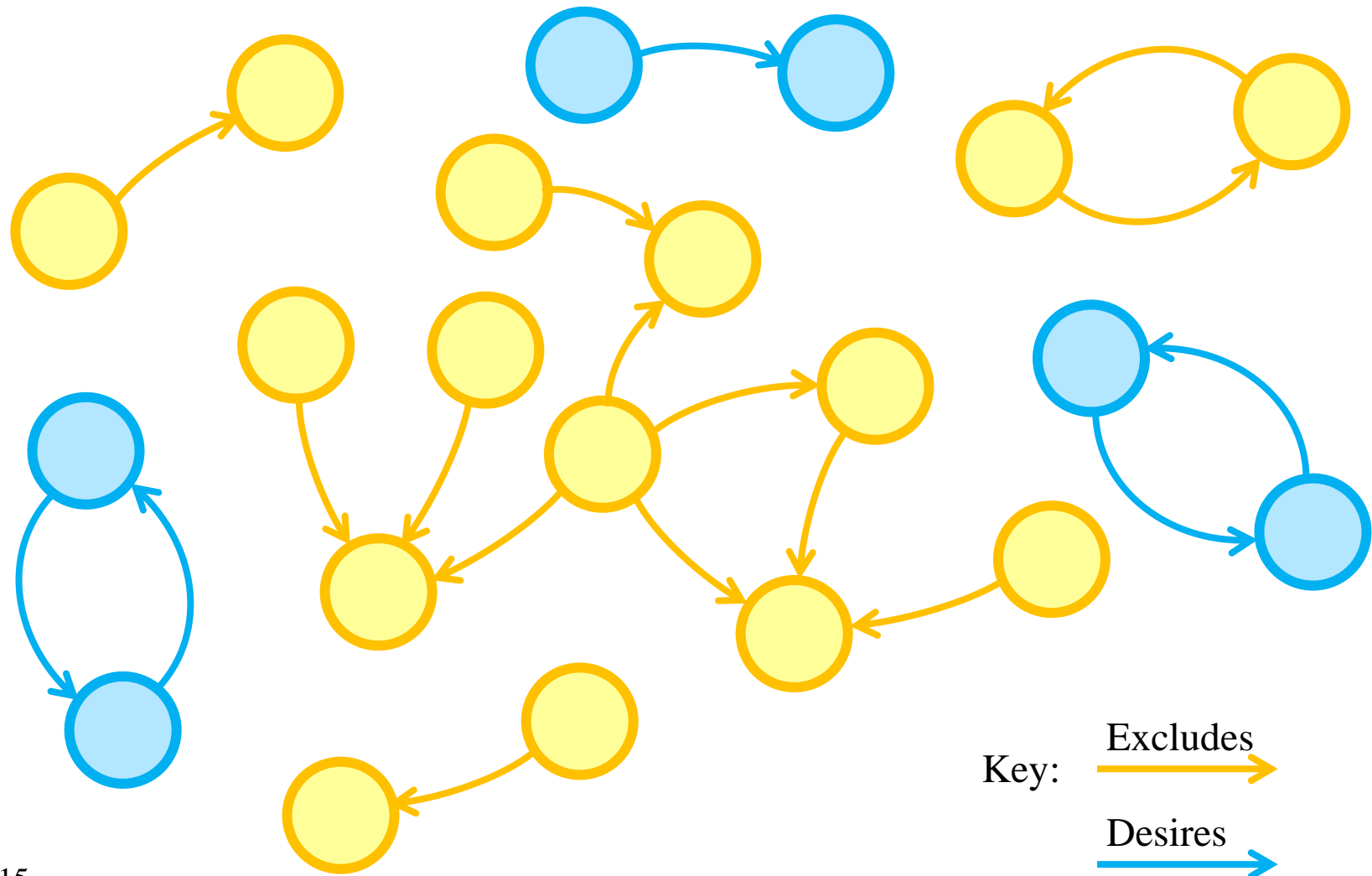https://student.eait.uq.edu.au/urite/?act=show&schedule=264

- It is important you attend:
  - Room safety induction
  - Hand out tool boxes to teams
  - Great time to meet your other group members

On that thought…

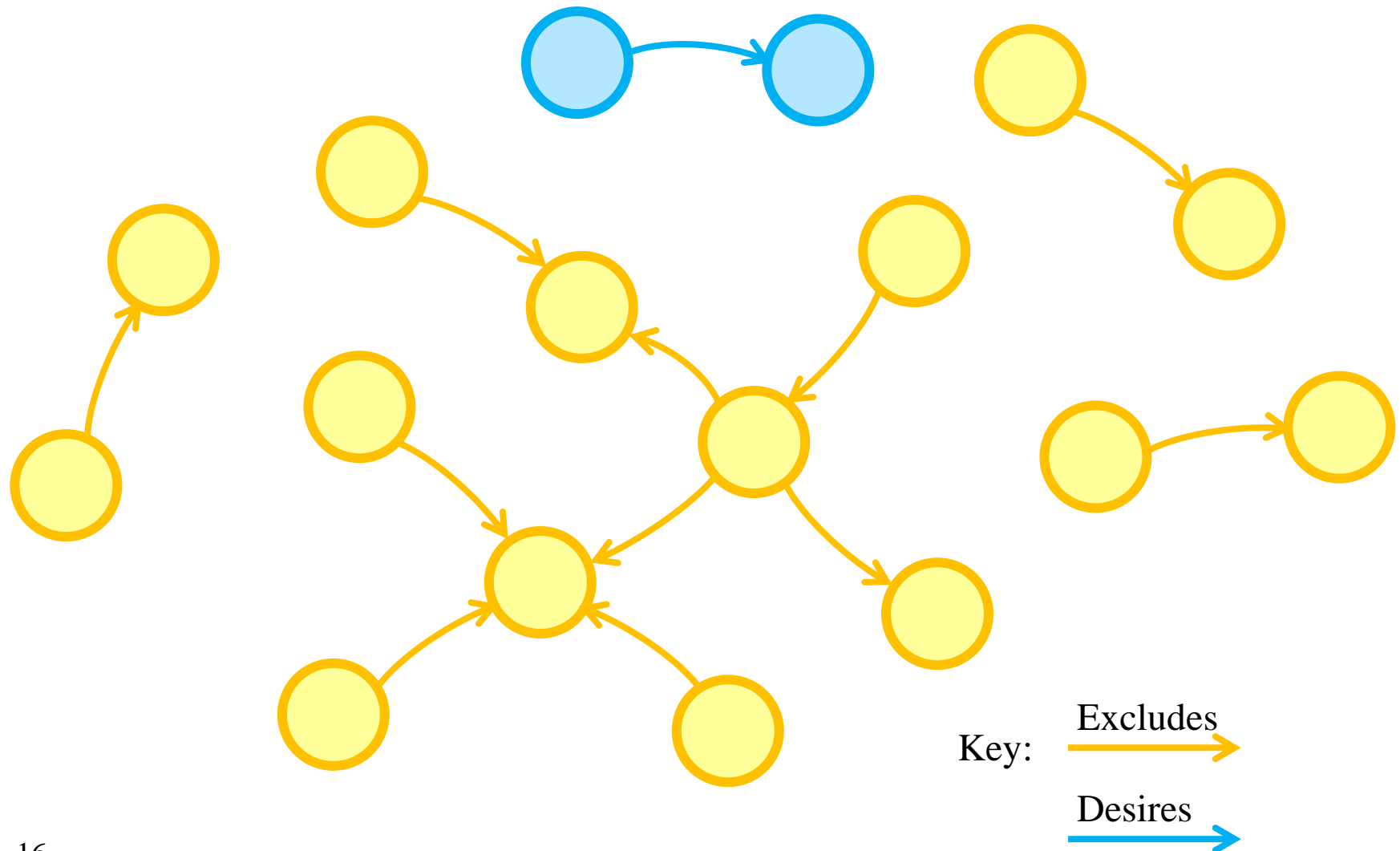- Teams are now posted on Blackboard

- The magical team sort algorithm satisfied all exclusion requests ~~with minor tweaking with an exhausting struggle~~ with hardly any work at all?
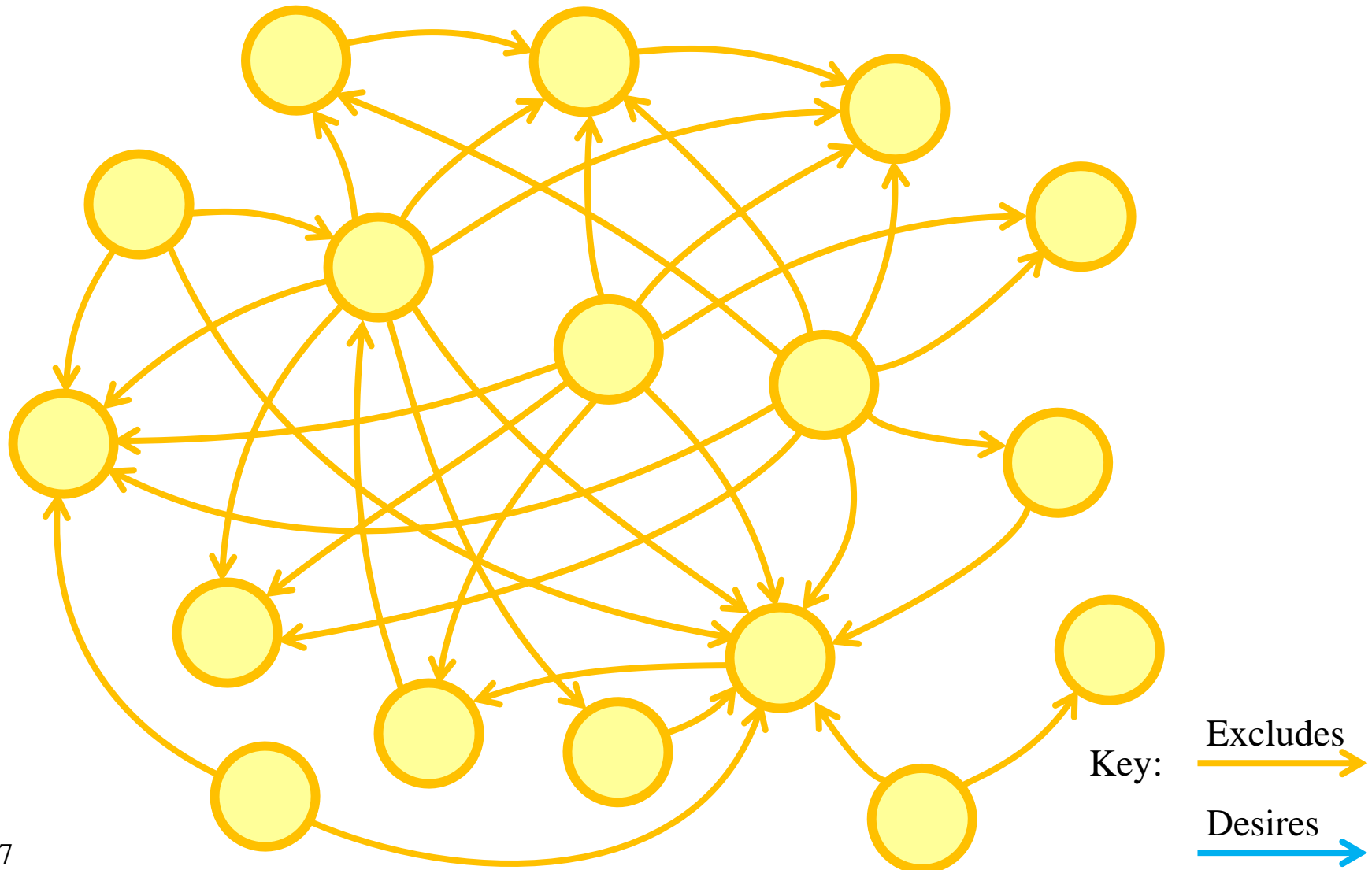 - I'm as surprised as you are
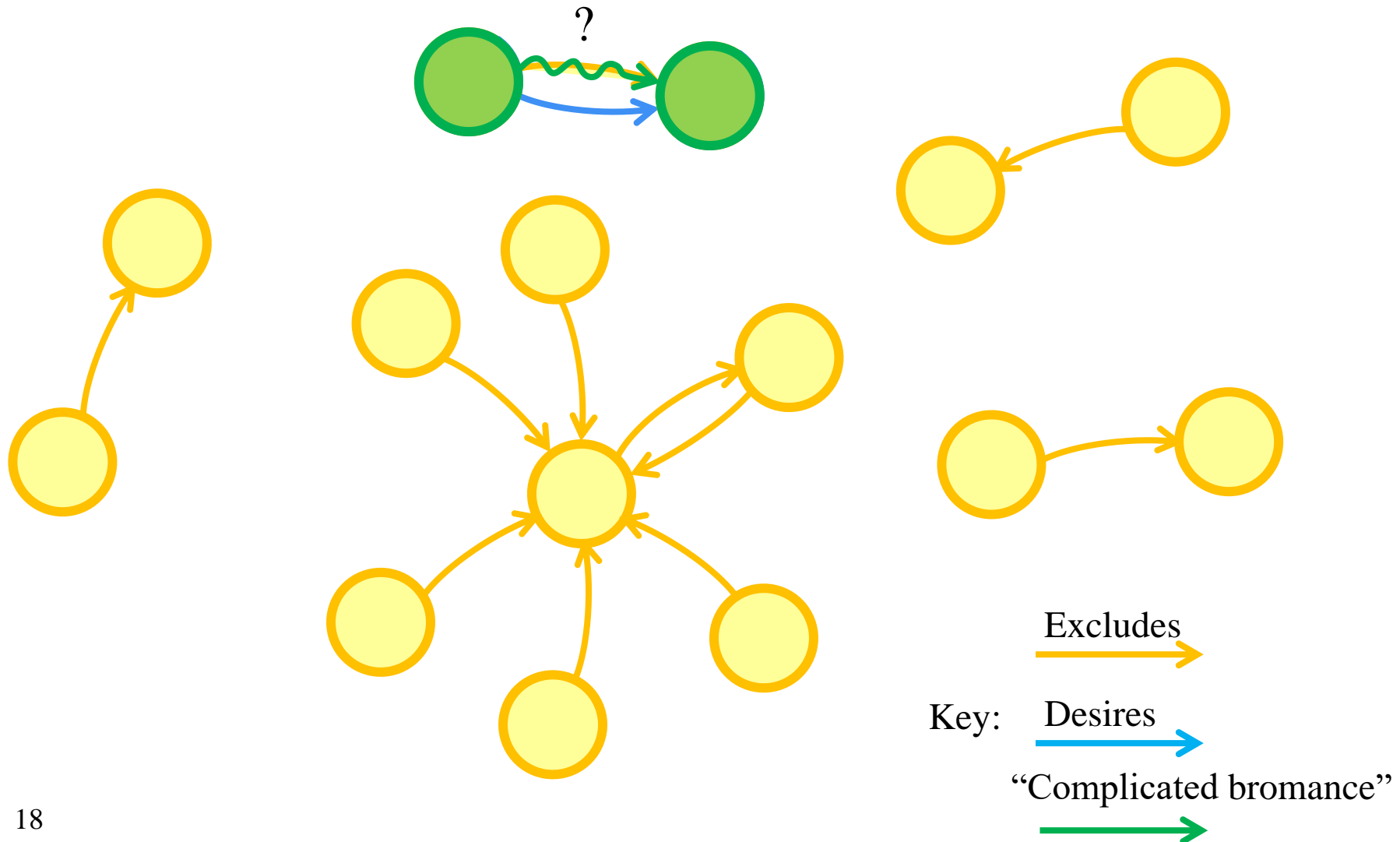
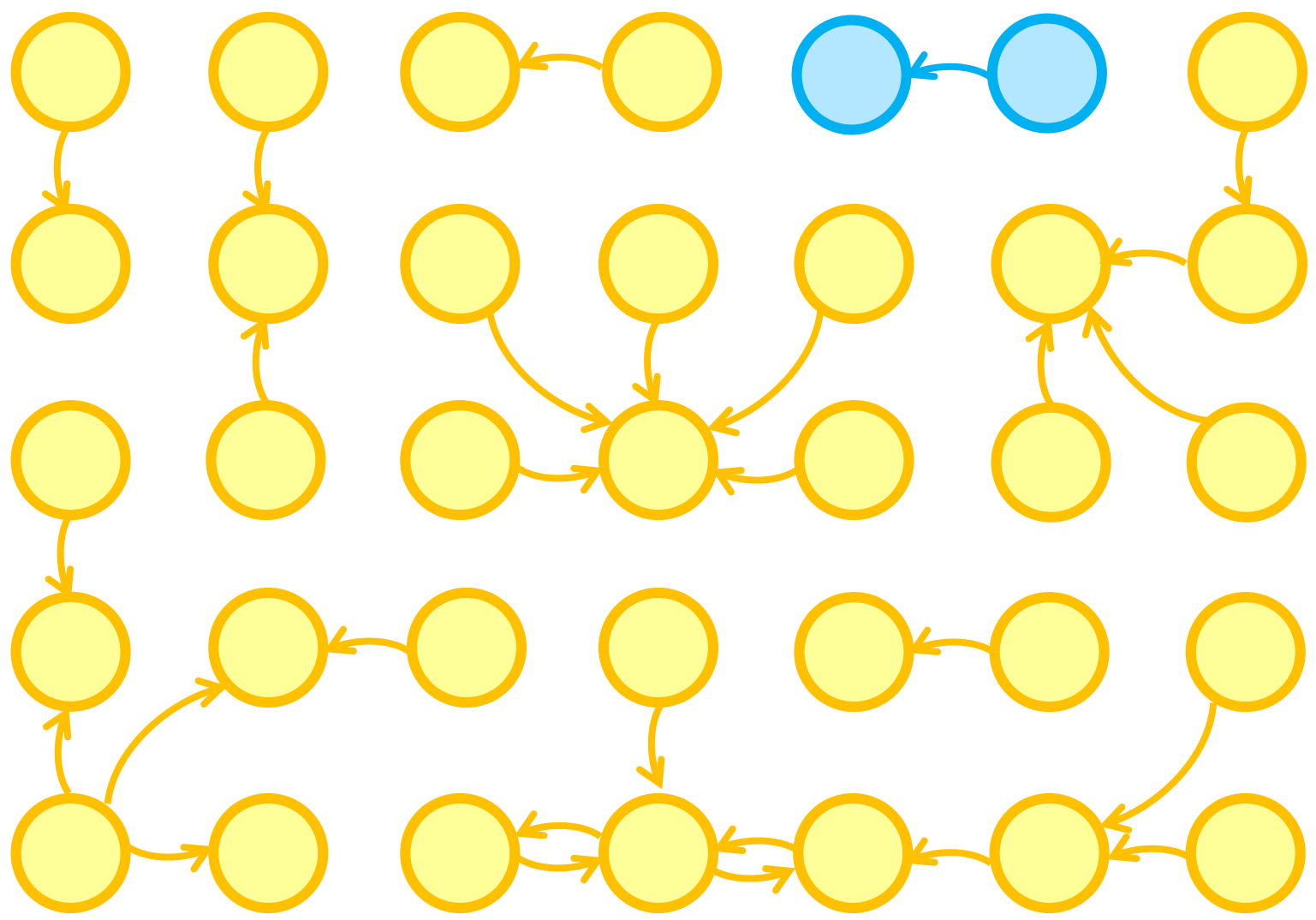 Some context....

2013 Directed Graph of Woe

Key:

Excludes

Desires

2014 Directed Graph of Woe

Key:

Excludes

Desires

2015 Directed Graph of Woe

Key:

Excludes

Desires

2016 Directed Graph of Woe

Key: Excludes

    Desires

    "Complicated bromance"

2017 Directed Graph of Woe

2018 Directed Graph of Woe

Excludes

Key: Desires

2019 Directed Graph of Woe

Excludes

Key: Desires

"Phantom menace"?

2020 Directed Graph of Woe

Key:

Excludes

Desires

Phantom menace
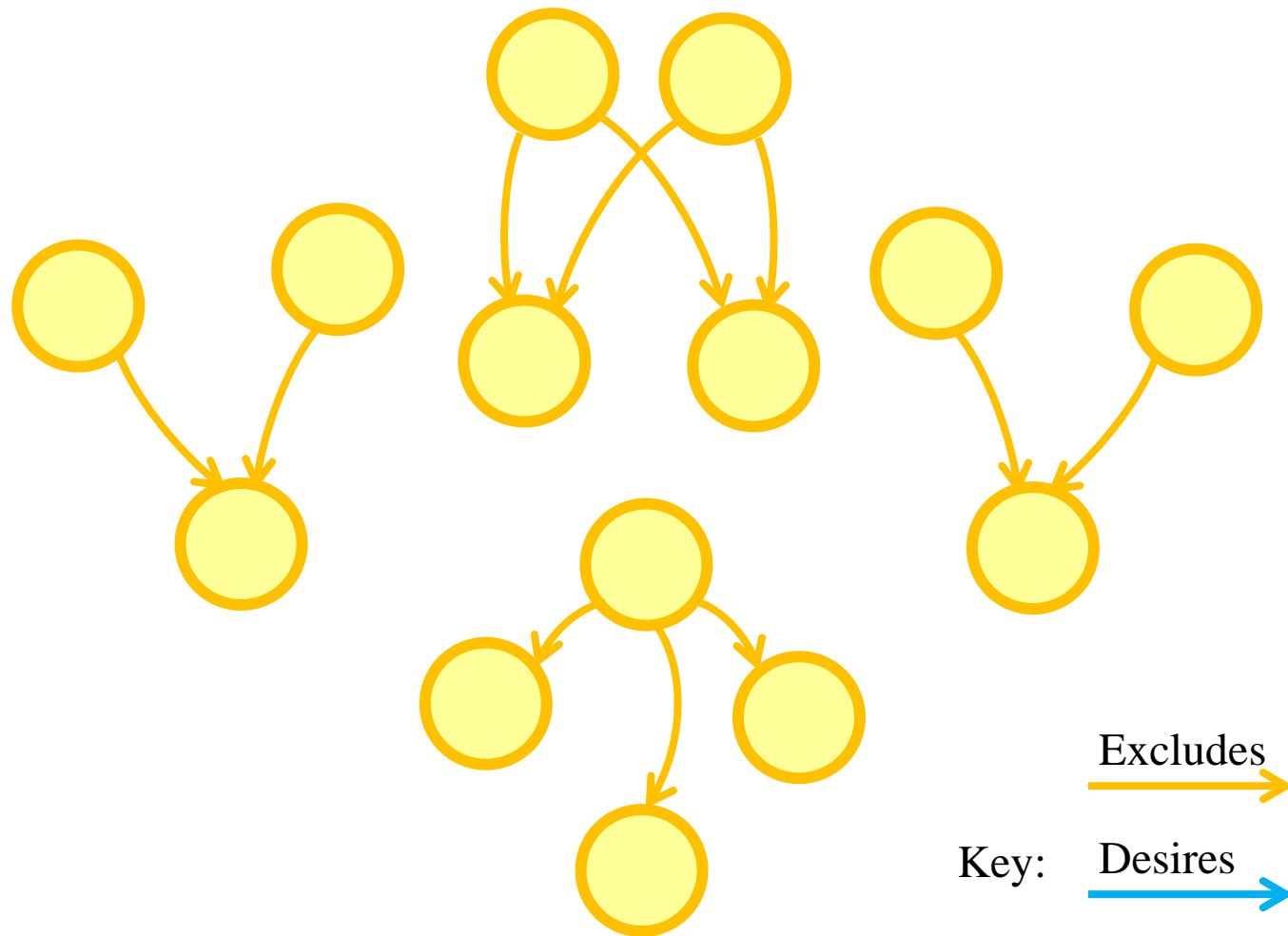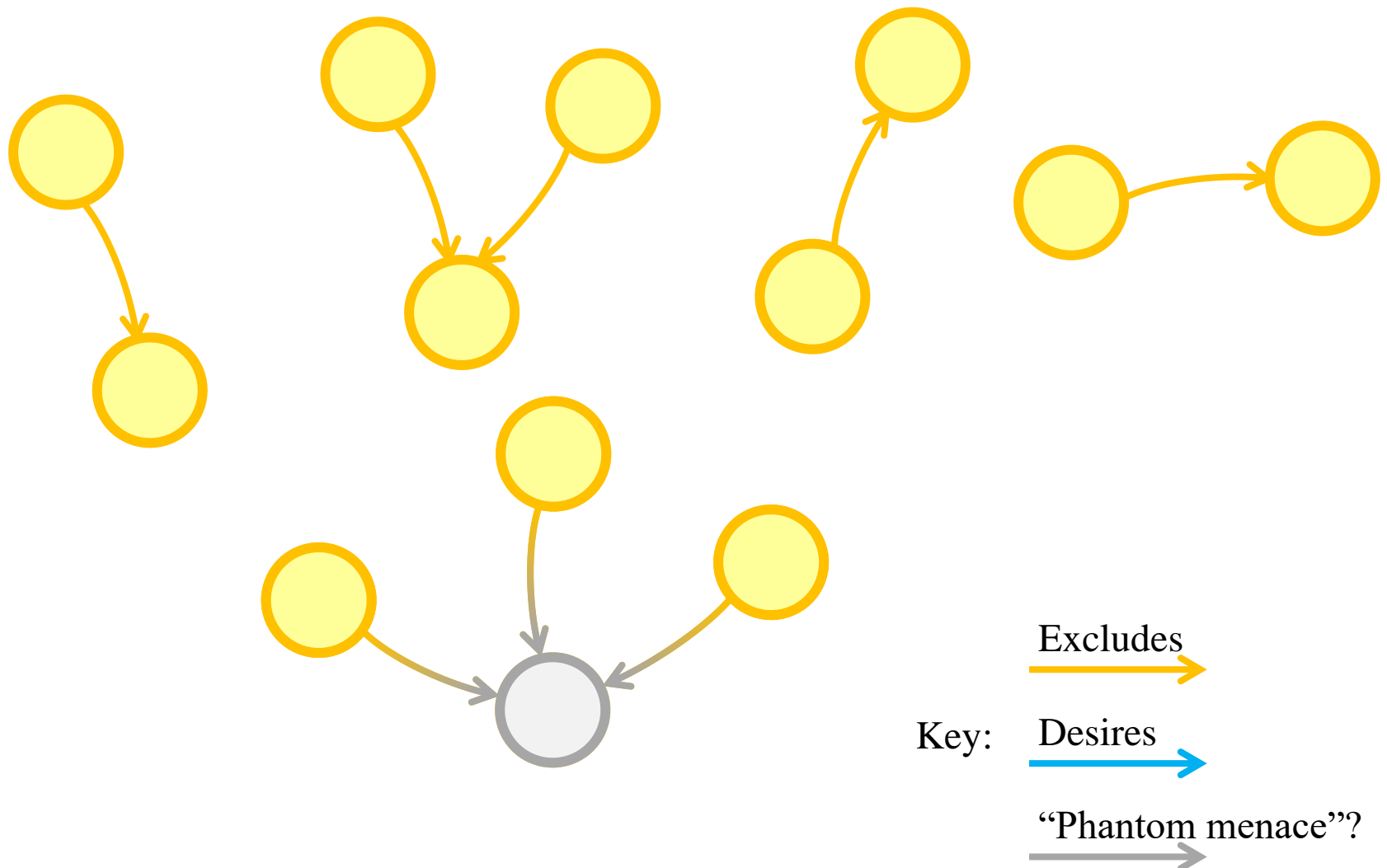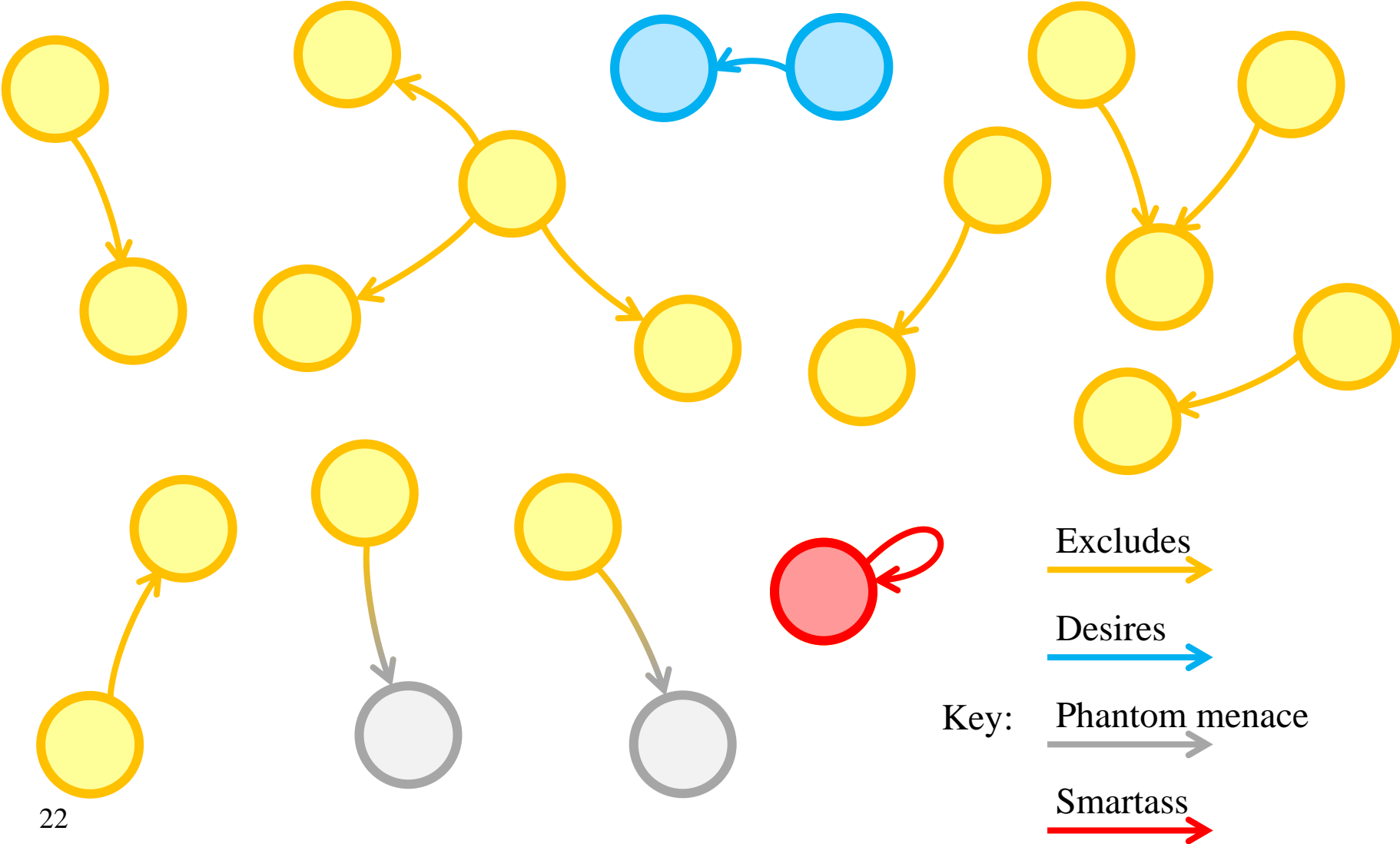
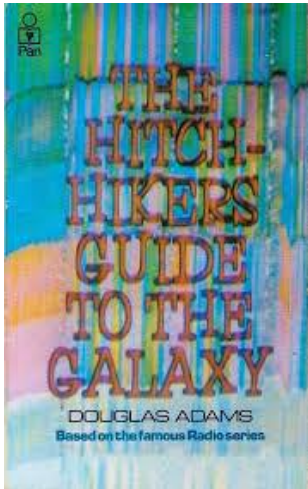Smartass

On being a smartass
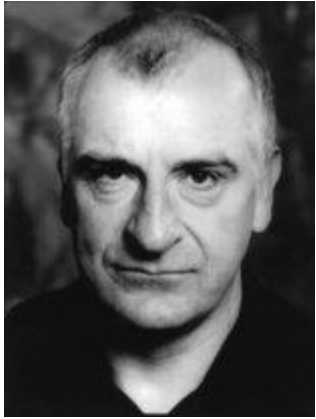
Douglas Adams

"It startled him even more when just after he was awarded the Galactic Institute's Prize for Extreme Cleverness he got lynched by a rampaging mob of respectable physicists who had finally realized that the one thing they really couldn't stand was a *smartass*."

23

FAQ Roundup

- **So, just to be clear.... no practicals or contacts until next week?**
  - Nope, but the induction is tomorrow. Regular practicals start next week. And also, contact sessions aren't really a 'thing' per se.

- **Can we use $XYZ motor/camera/processor/module?**
  - If the project Description, Rules and Procedures document doesn't specifically say you *can't* use it, then presume that you *can* use it. This course is intended to give you freedom: in the sense of "just enough rope".

- **Are you going to stack the containers 15 units high?**
  - Nope – starting with a max of two.

Back to that design thing…

Mechatronic Systems Design

Lolwut?

What is design, anyway?

- Design:
  - n. A goal or intention
  - v. The process of 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 goals

Things that are designed

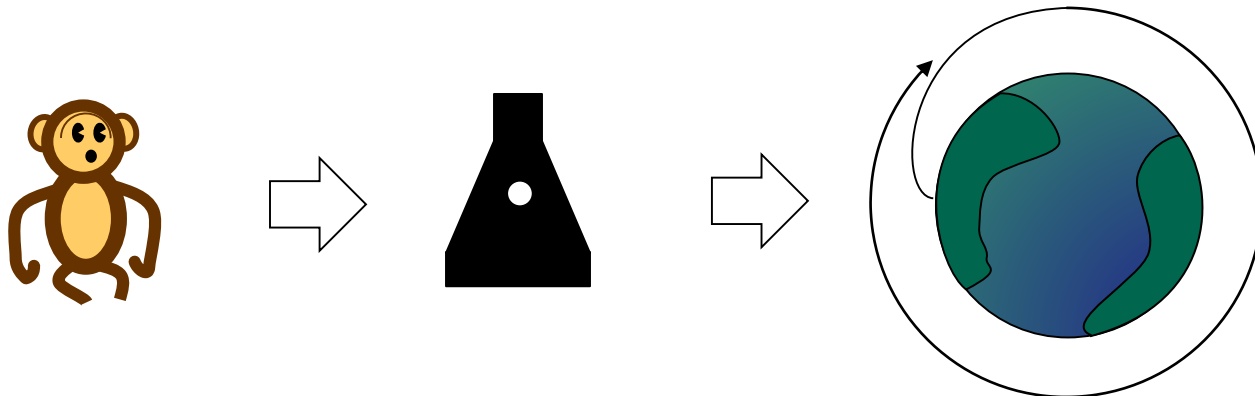- Devices/structures

- 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

    contra: 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:

<div align="center">

Analysis   **Synthesis**

</div>

- Specification ⟷ Implementation

- Deconstruction ⟷ Constitution

- Parameterisation ⟷ Optimisation

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

30

Laying it out

How do we know what
it is we have to design?

An example workflow

```
┌─────────────────────┐
│    Client describes  │
│    the problem       │
└─────────────────────┘
           ↓
┌─────────────────────┐
│    Engineer writes a │
│    design brief      │
└─────────────────────┘
           ↓
┌─────────────────────┐
│  Brief is turned into│
│   a precise design   │
│    specification     │
└─────────────────────┘
           ↓
┌─────────────────────┐
│   Engineer builds a  │
│   solution per the   │
│    specification     │
└─────────────────────┘
```

32
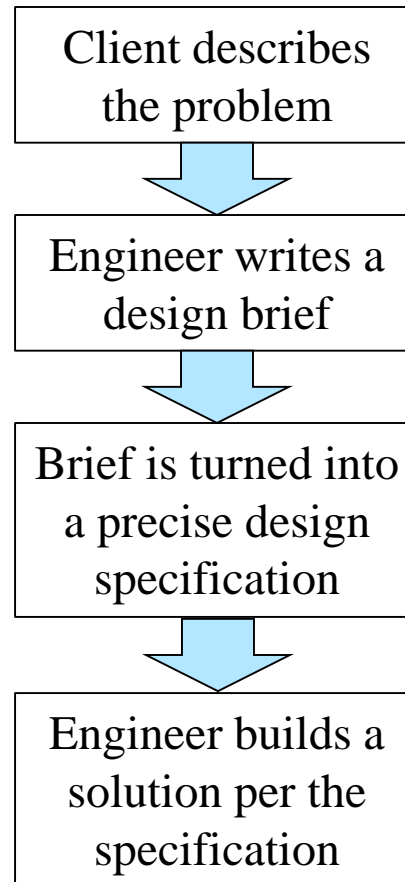
Design brief

- Design briefs communicate the intent of an engineering problem

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

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

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!)

# Specifications are important

- <u>All of your design effort is geared to meeting the specification</u>
  - 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

System boundary

Inputs

System

Outputs

Surroundings
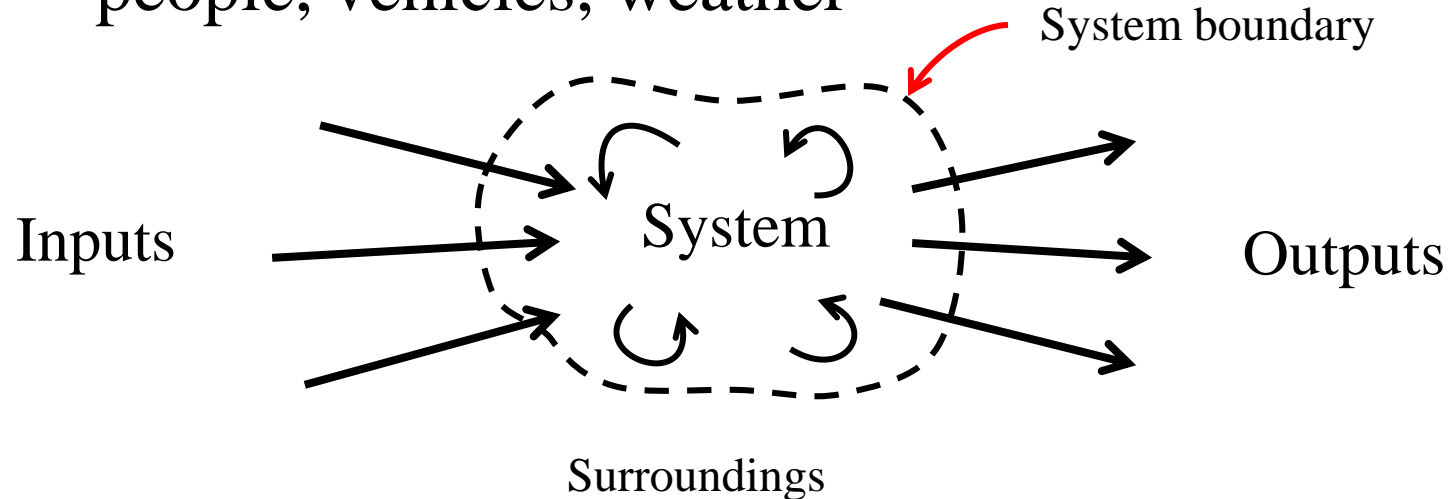
Systems Engineer maxim: "Everything is a system"

37

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 1   System 2
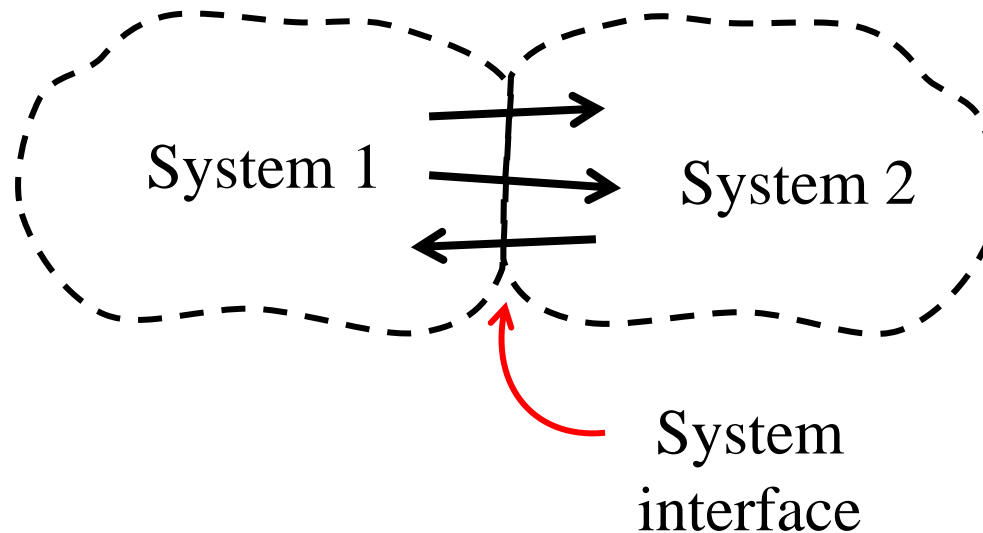
System interface

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"

40

Systems get complex *fast*

- Large systems like space shuttles and skyscrapers and cellphone networks are fiendishly complex
  - Even small systems (eg. smartphones) 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 those pieces 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"
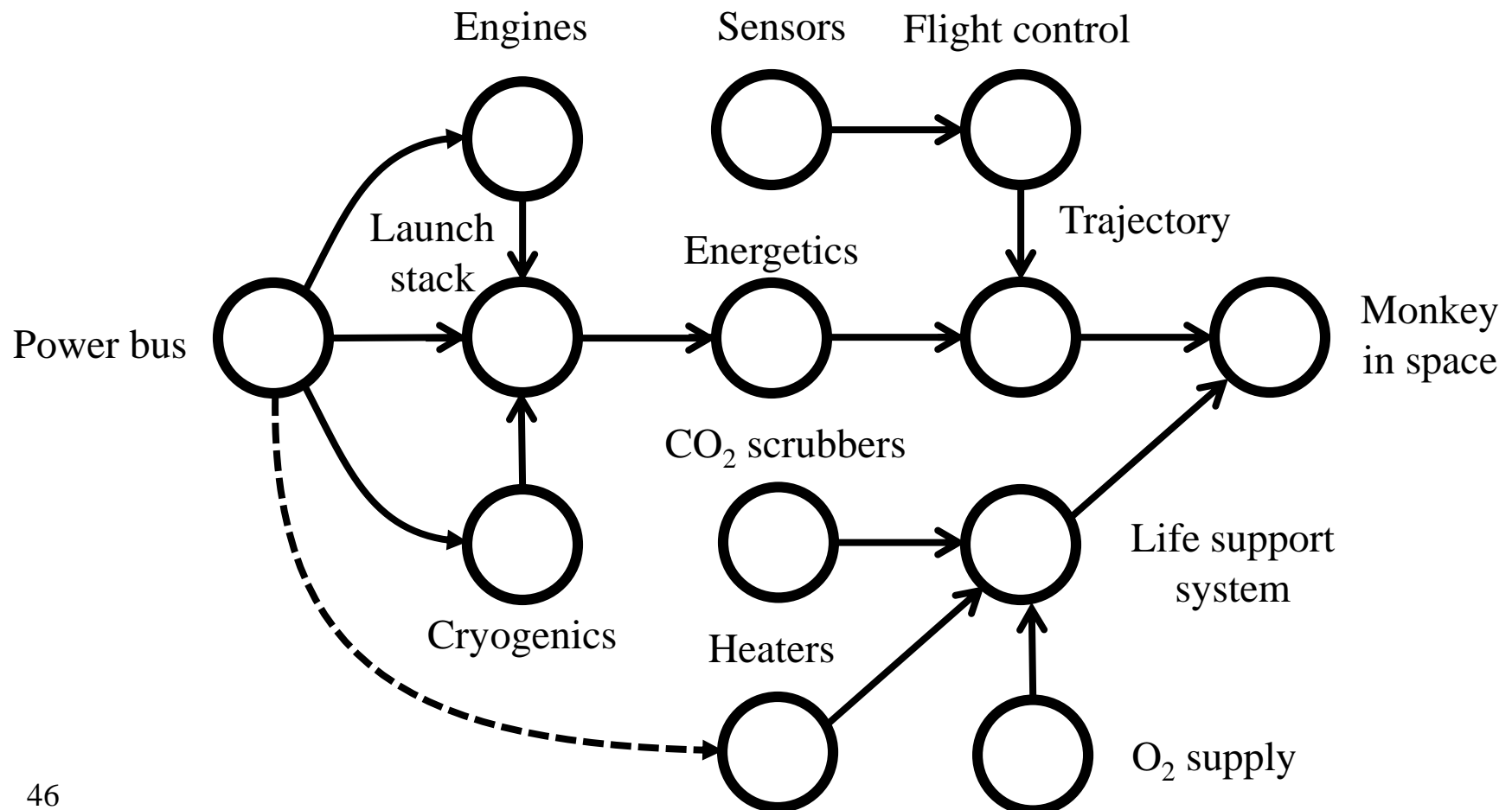 - Not necessarily in order of operation

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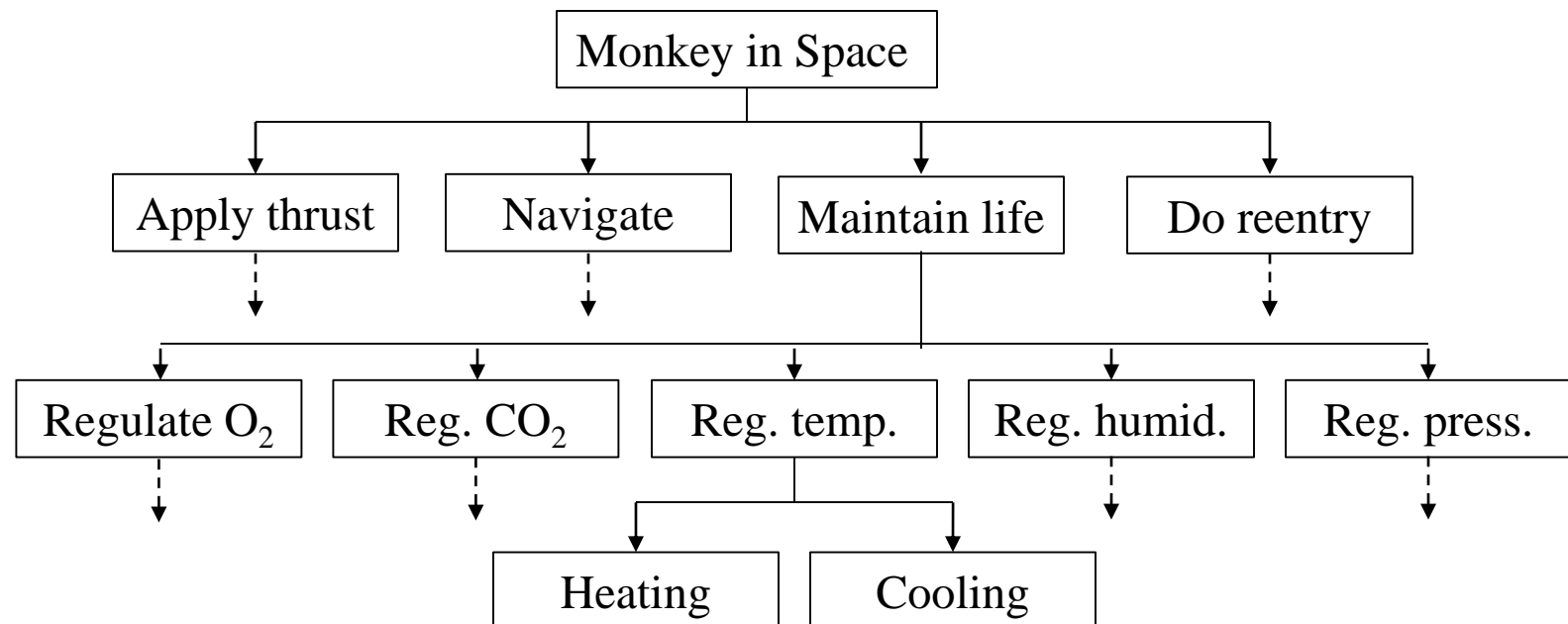
Causal system dependency

- Systems can be thought of as a cascaded series of enabling functions
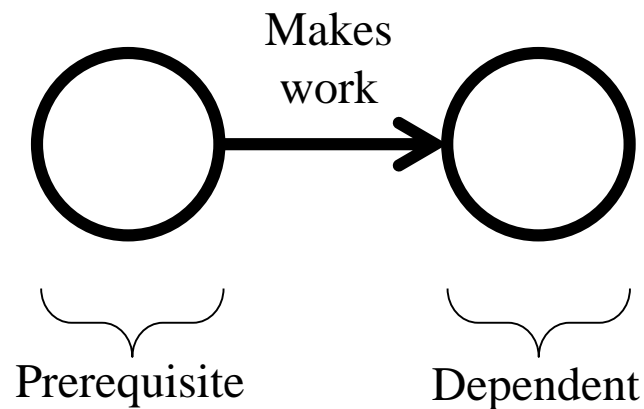
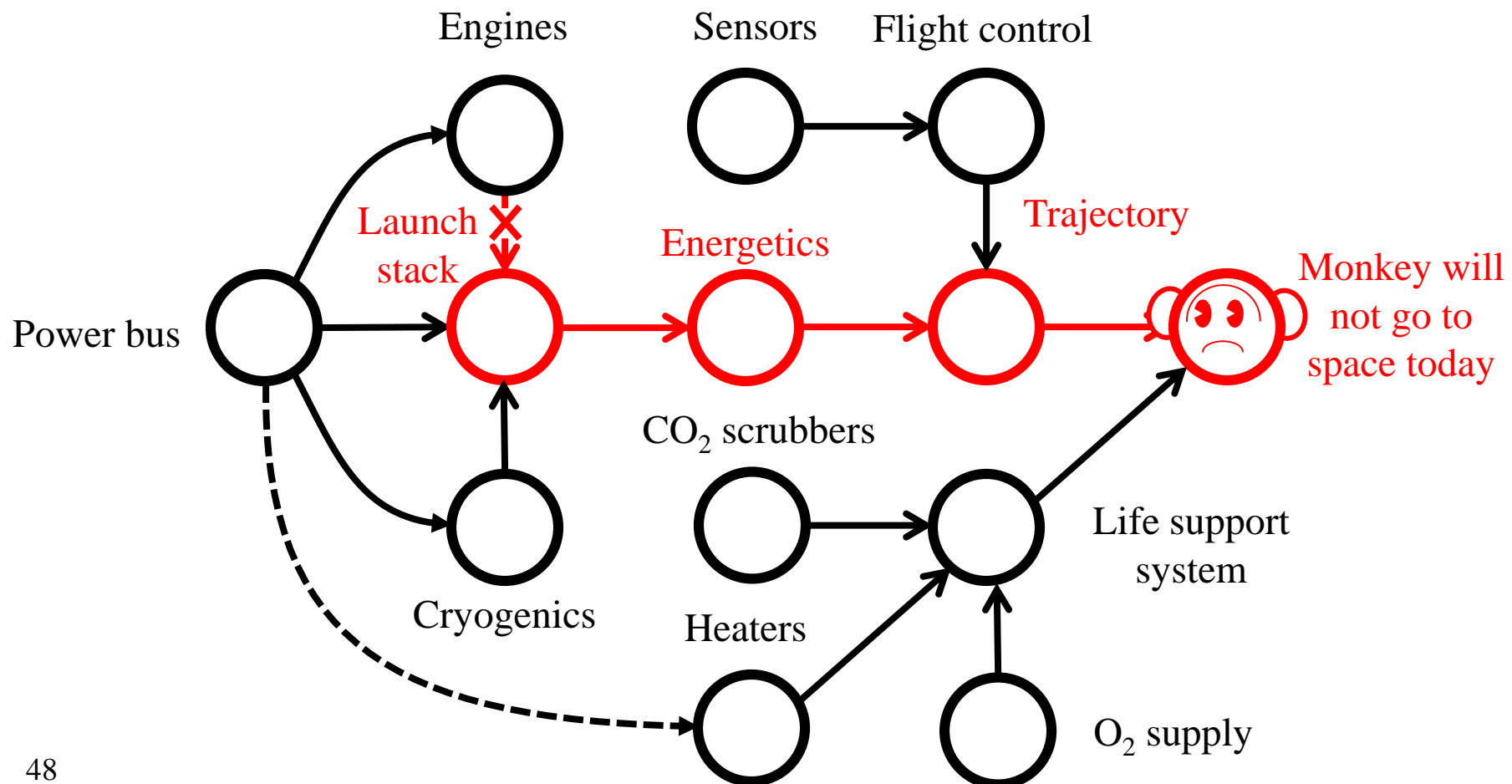# Causal system dependency

- For example:

Failure analysis

- We can also use causal system decomposition to understand failure

- Faults in the system propagate through directed graphs
  - Find the consequences of a fault
  - Work upstream to find the causes of a failure
  - Verify the "causal chain" to prove the system

Failure analysis

- Consider a system defect:

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"

Sneak peek…

Next week we'll go through a few of the previous projects and we'll do this in class!

- Live!

- Interactive!

- Hopefully interesting!

- Certainly educational!

Intermedio.

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 cost

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?

54

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.

55

Parametric design

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

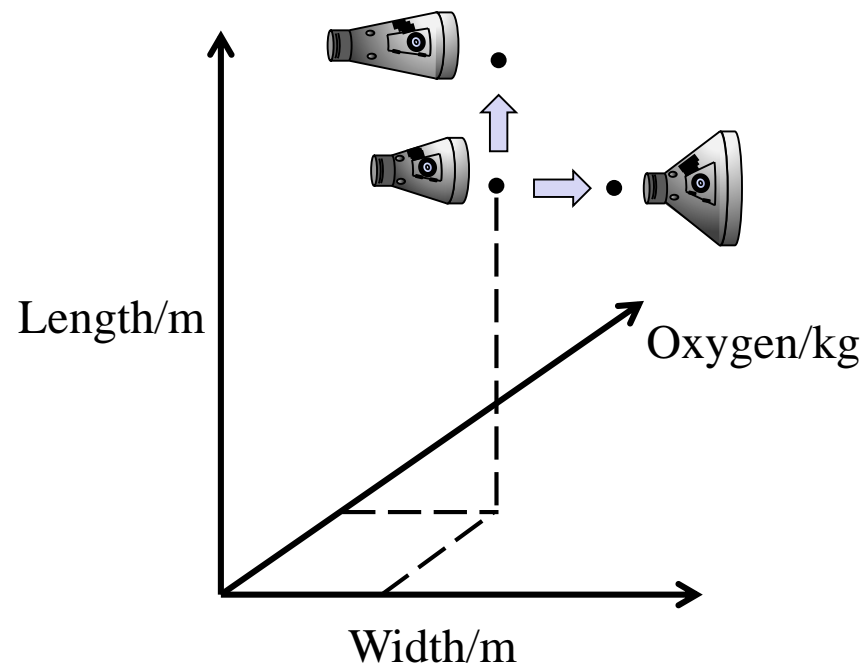- The key dimensions/values/settings that describe a design are called the "design parameters"

Parametric design

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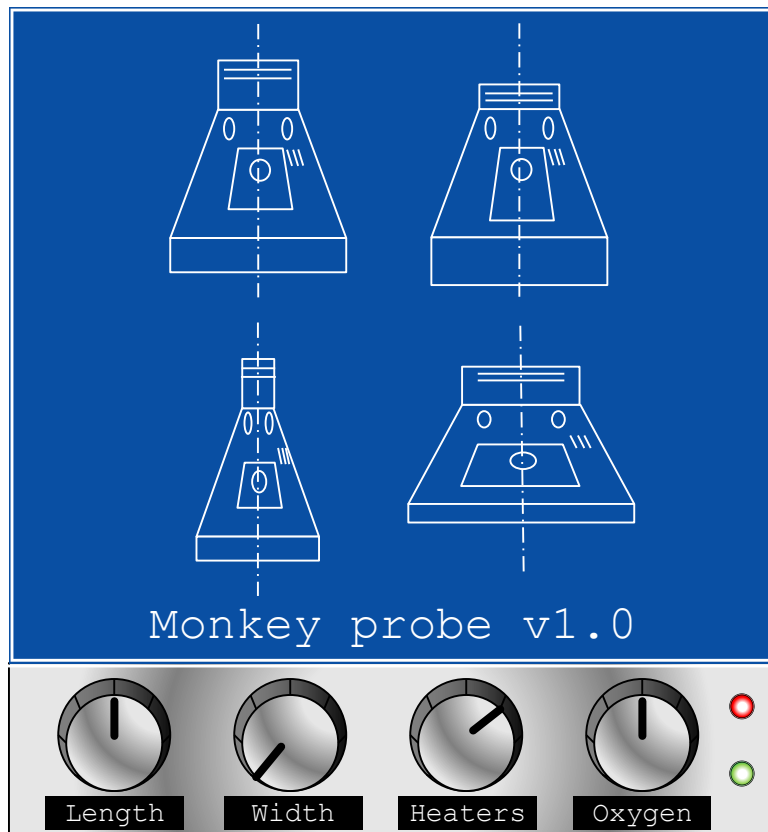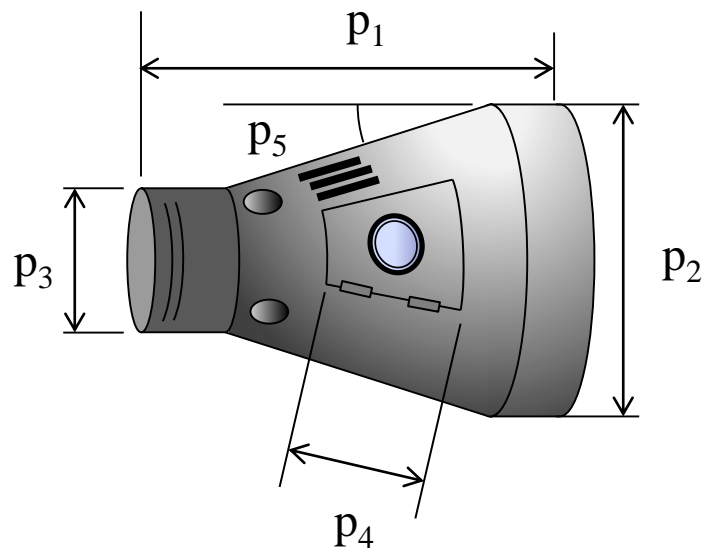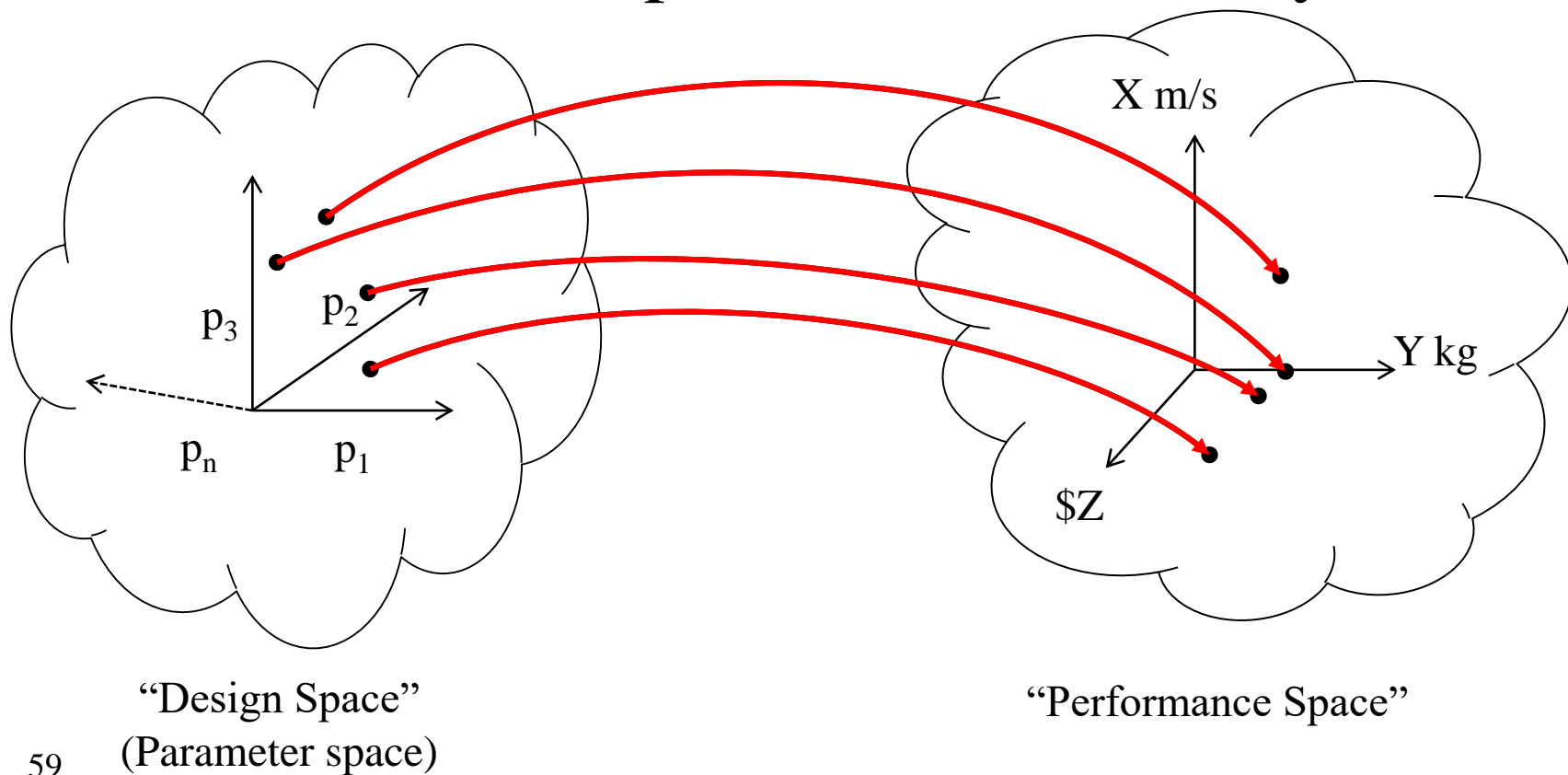
Design space

- 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

"Design Space"
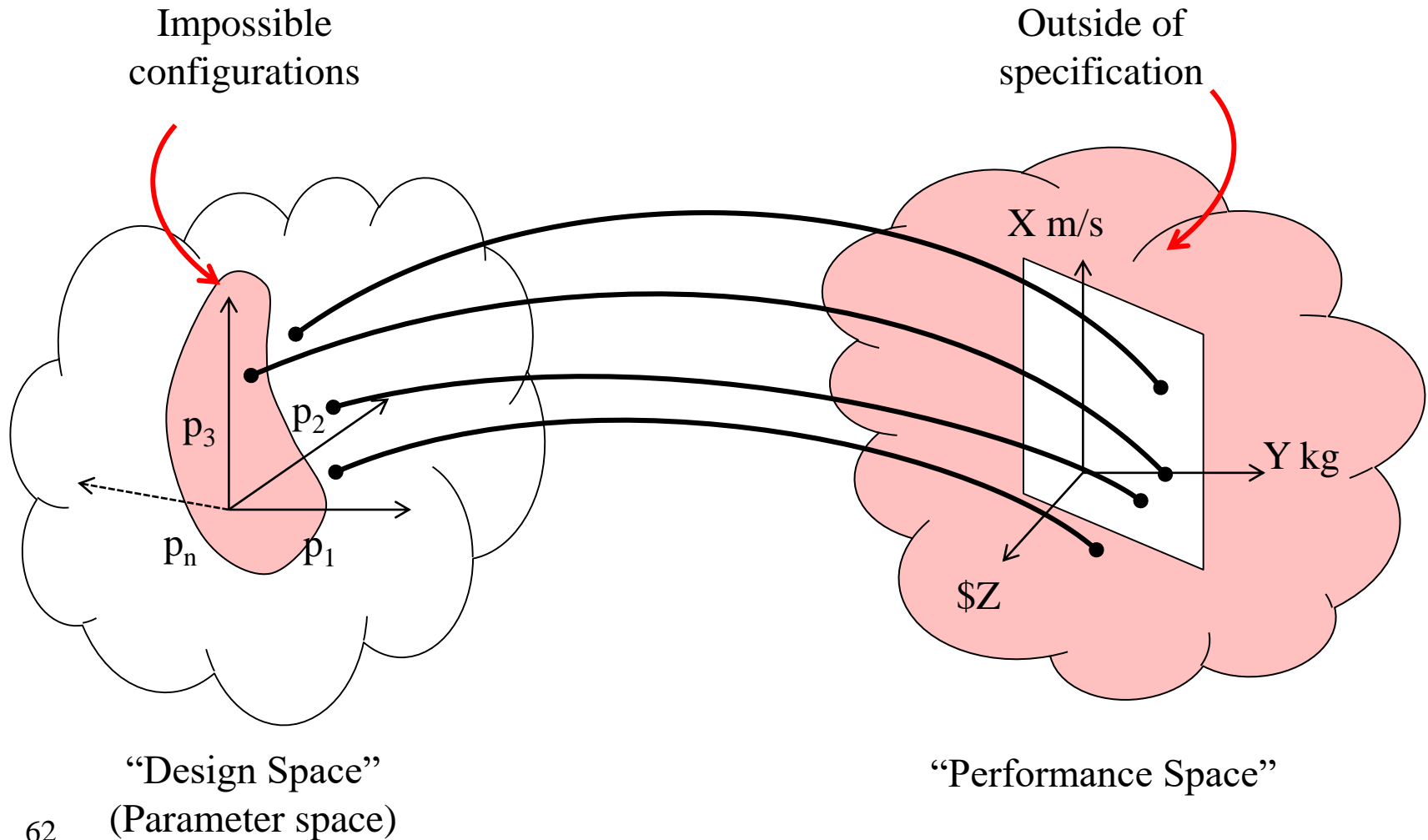(Parameter space)

"Performance Space"

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

X m/s

Y kg

$Z

p_3 p_2

p_n p_1

"Design Space"
(Parameter space)

"Performance Space"
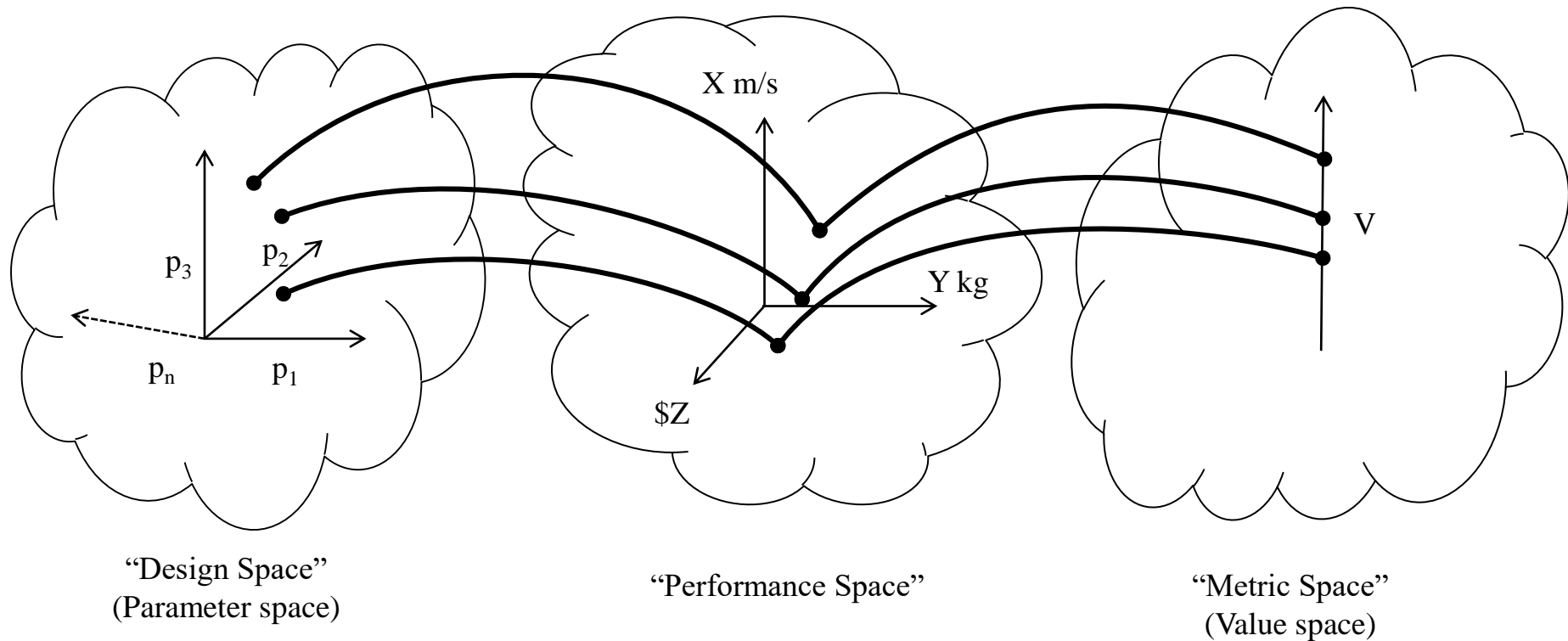
62

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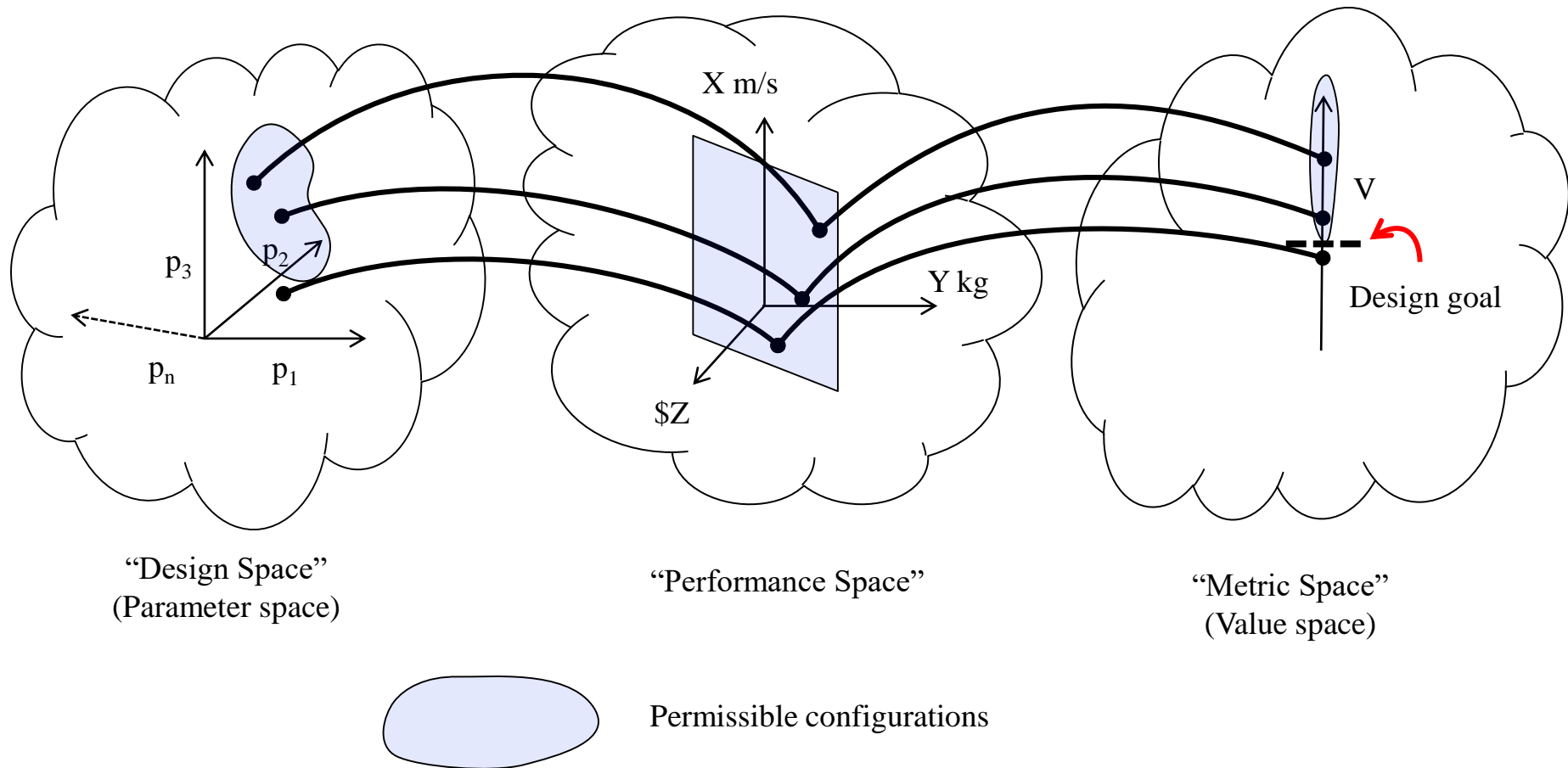
$$Value = \frac{explosive\ force}{unit\ cost}$$

$$V = \frac{X(p_1, p_2 \dots pn)}{C(p_1, p_2 \dots pn)} = f(p_1, p_2 \dots pn)$$

$$V_{max} = \sup(f(p_1, p_2 \dots pn))$$

Metric space*

"Design Space"
(Parameter space)

"Performance Space"

"Metric Space"
(Value space)

64

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

Metric space*

"Design Space"
(Parameter space)

"Performance Space"

"Metric Space"
(Value space)

Permissible configurations

65

*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"
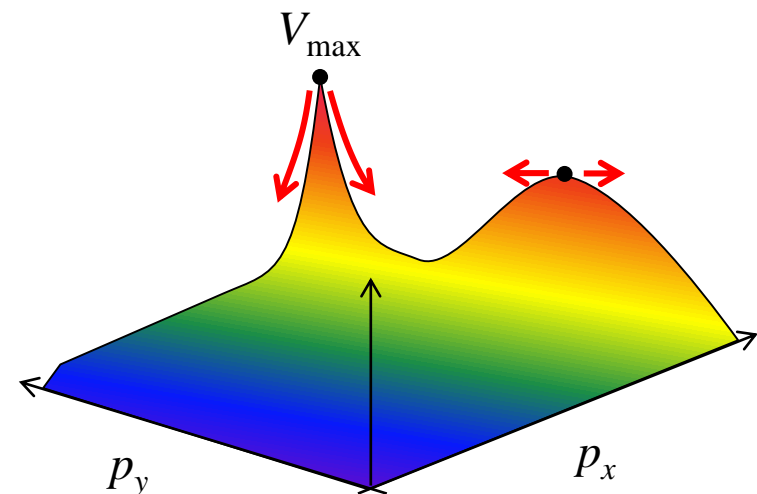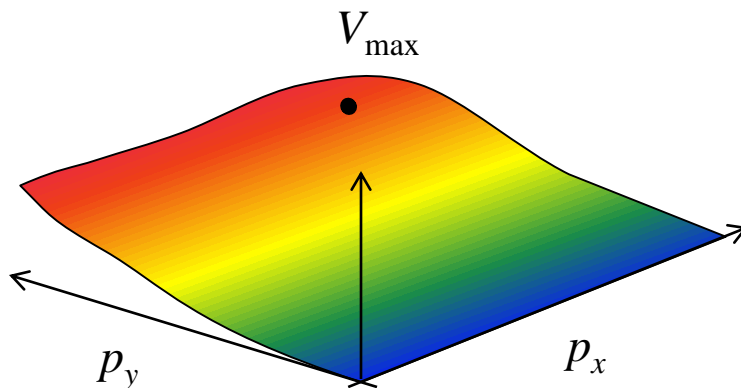
Finding a value function

- 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:

	Safe	Low cost	Reusable	Easy to build	Payload capacity	Score
Safe	■	X	X	X	X	4
Low cost		■	X	X		2
Reusable			■			0
Easy to build			X	■		1
Payload capacity		X	X	X	■	3

Prioritise by score

68

"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

Increasing cynicism

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)

73

Cyclic vs linear models

Define

Model

Implement

Support

Verify

Requirements

Design

Implement

Verify

Maintain

Express

Test

Look

Ask

Simplify

Try

74
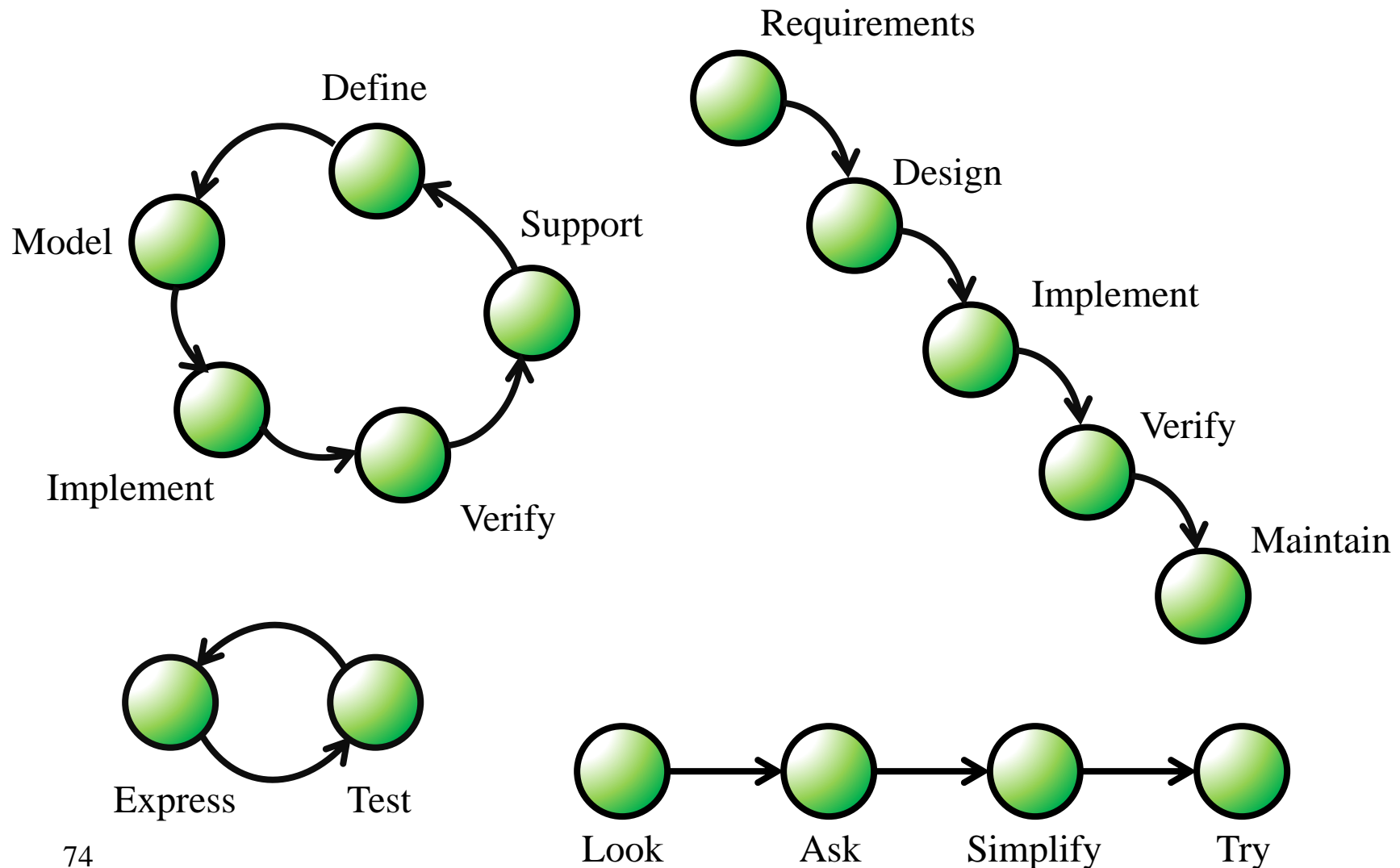
So which do you pick?

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
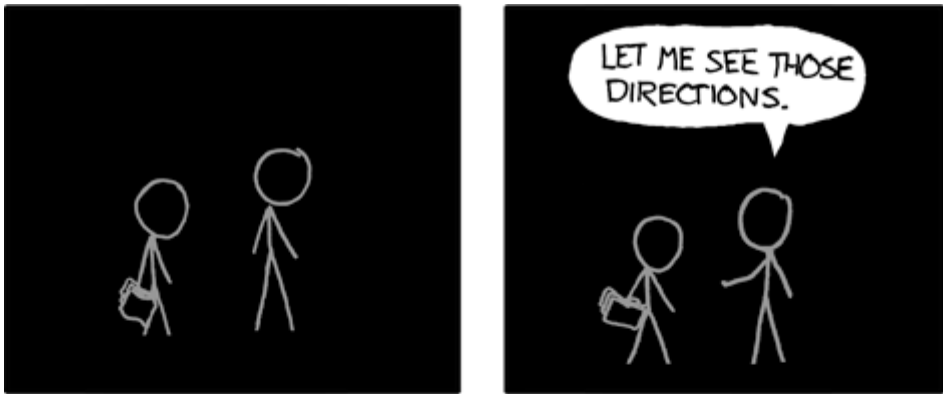   – Modelling, validation, testing
    – Critically evaluate ideas

*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

xkcd: Randall Munroe

Pauline'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

80

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

81

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)

83

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
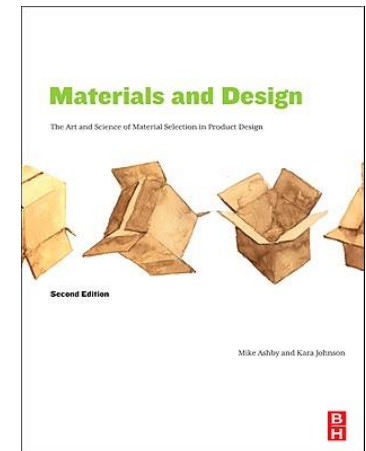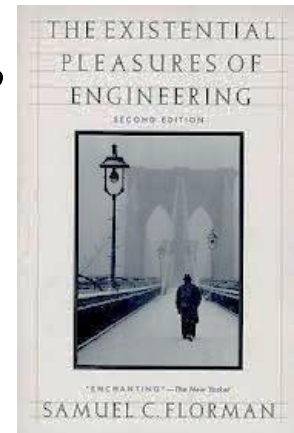
Questions?
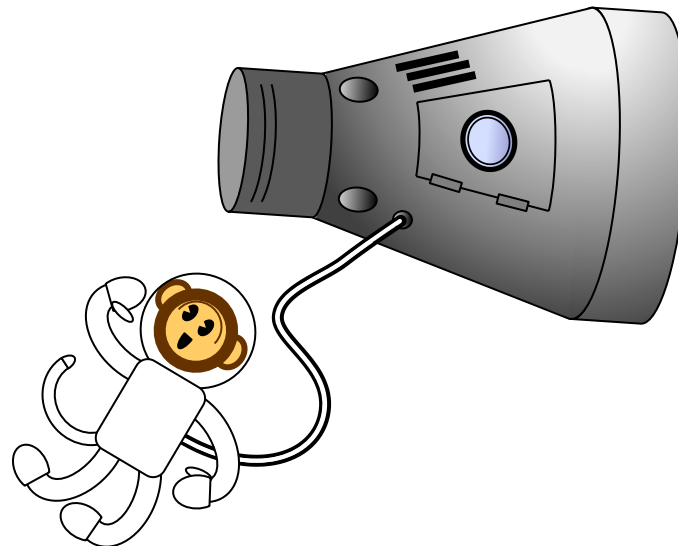
Tune-in next time for…

Previous Years' Design Case Studies

or

"The definition of madness is doing the same thing over
and over again and expecting a different result"

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

Nineteen survived.