

Modern Software Engineering

Building Better Software Faster

Dave Farley



<https://www.davefarley.net>



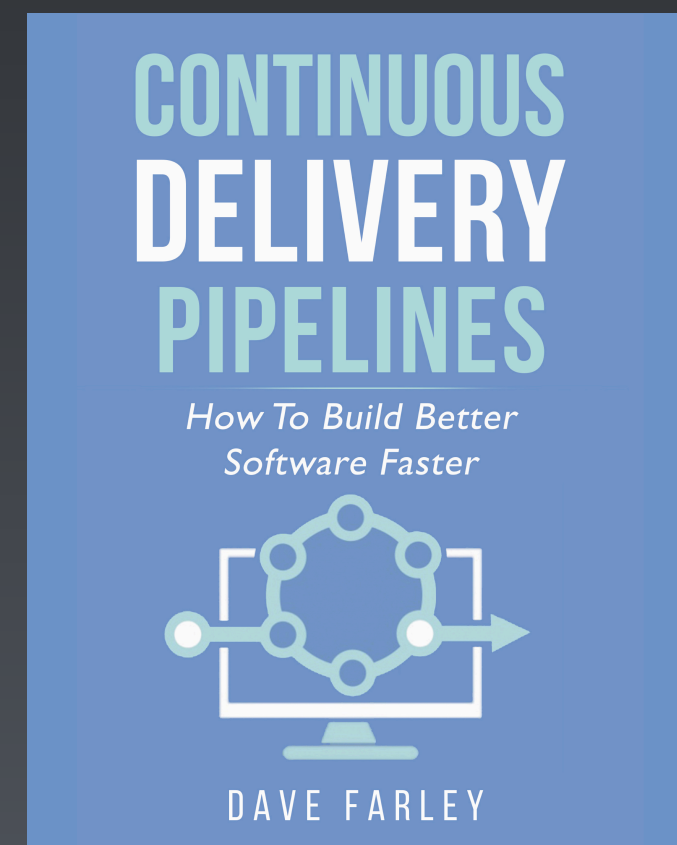
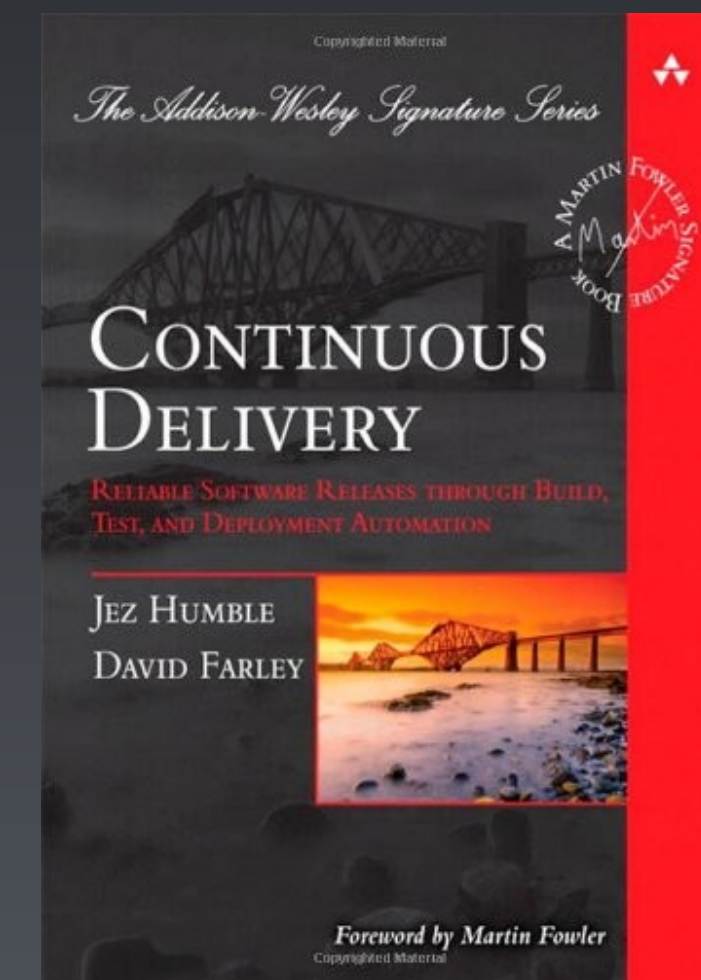
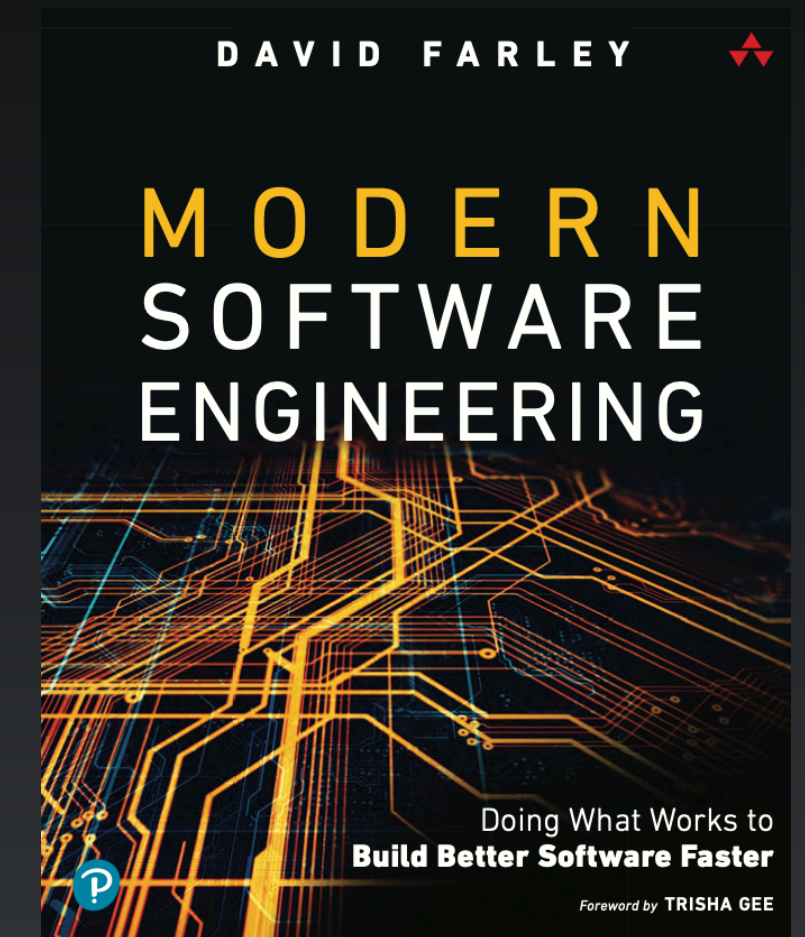
@davefarley77



<https://bit.ly/CDonYT>



<http://www.continuous-delivery.co.uk>



- ***What is Engineering***

- ***What is Engineering***
- ***Common Foundational Principles***

- ***What is Engineering***
- ***Common Foundational Principles***
- ***10 Guiding Principles for Software Engineering***

- ***What is Engineering***
- ***Common Foundational Principles***
- ***10 Guiding Principles for Software Engineering***
- ***Applying the Guidelines to Code***

What is Engineering

- *Engineering in Other Disciplines is:*
"The Stuff That Works"

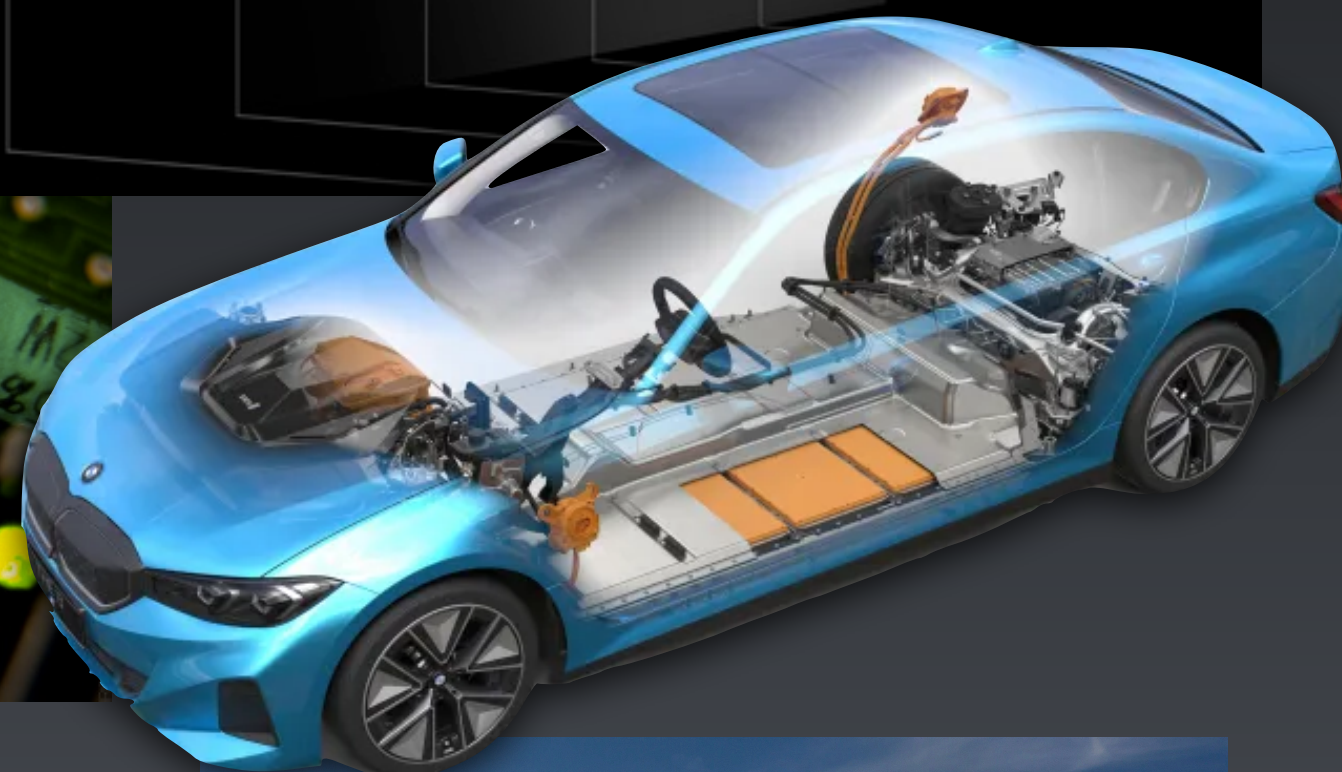
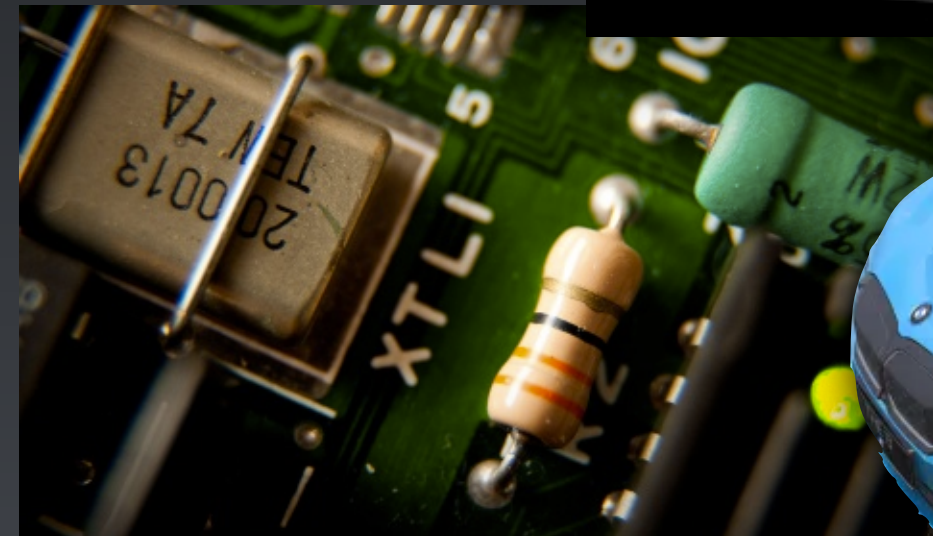
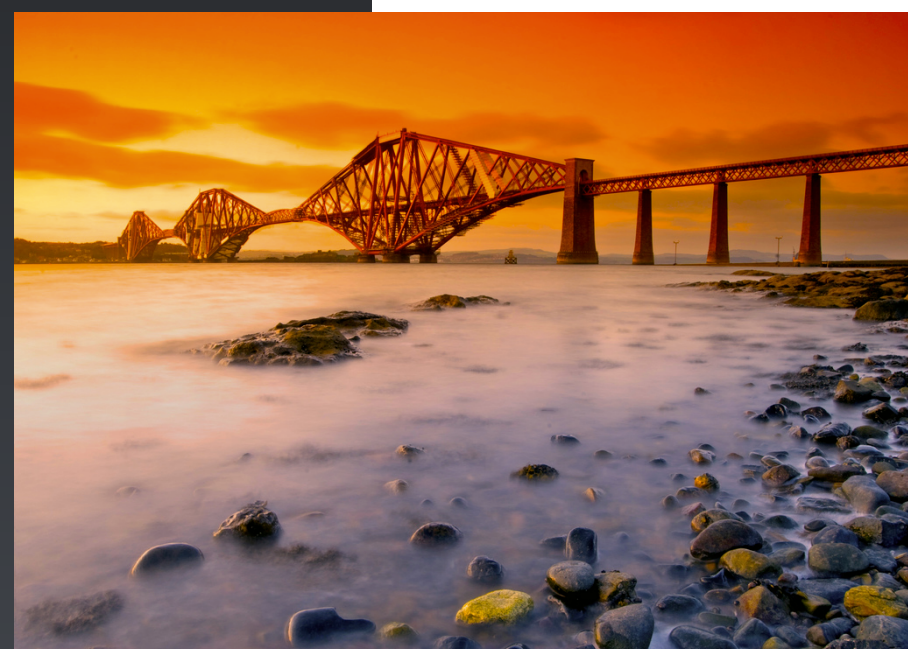
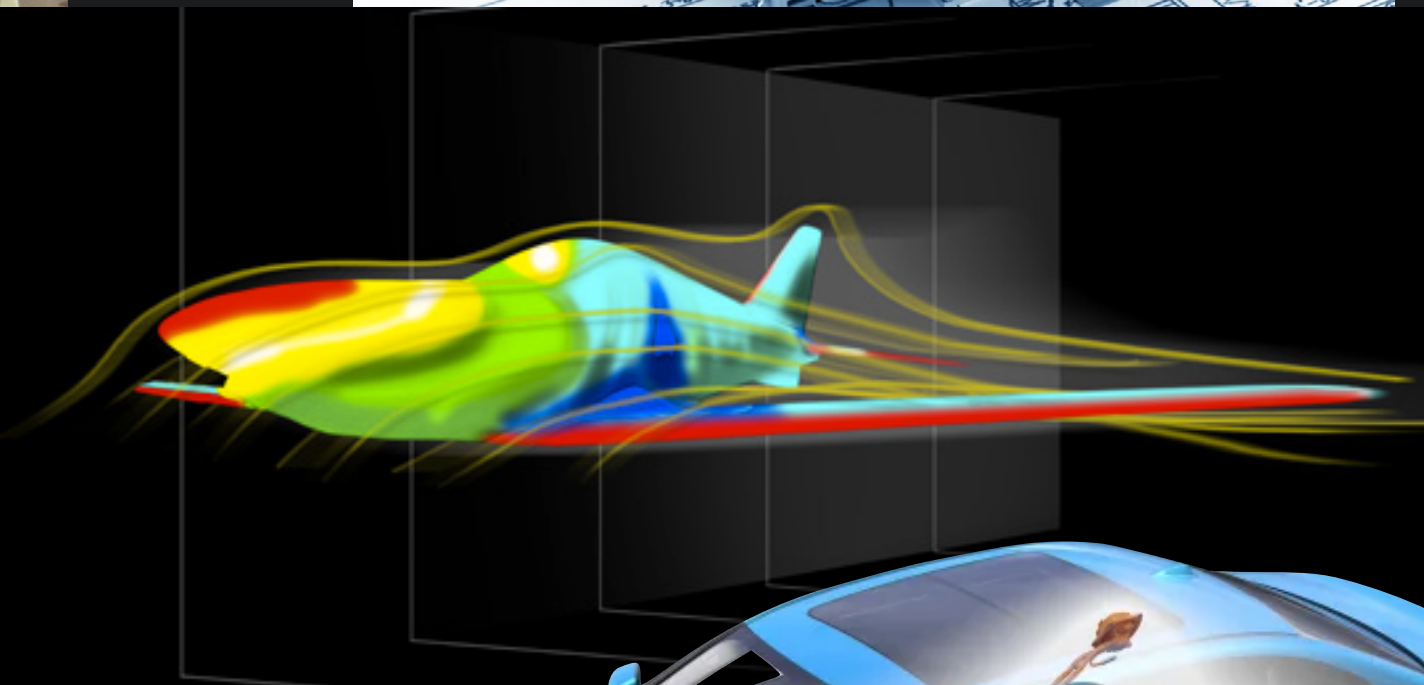
- *Engineering in Other Disciplines is:*
"The Stuff That Works"
- *Software is Difficult!*
*So What **"Works"** for Us?*

All Engineering is not the same!

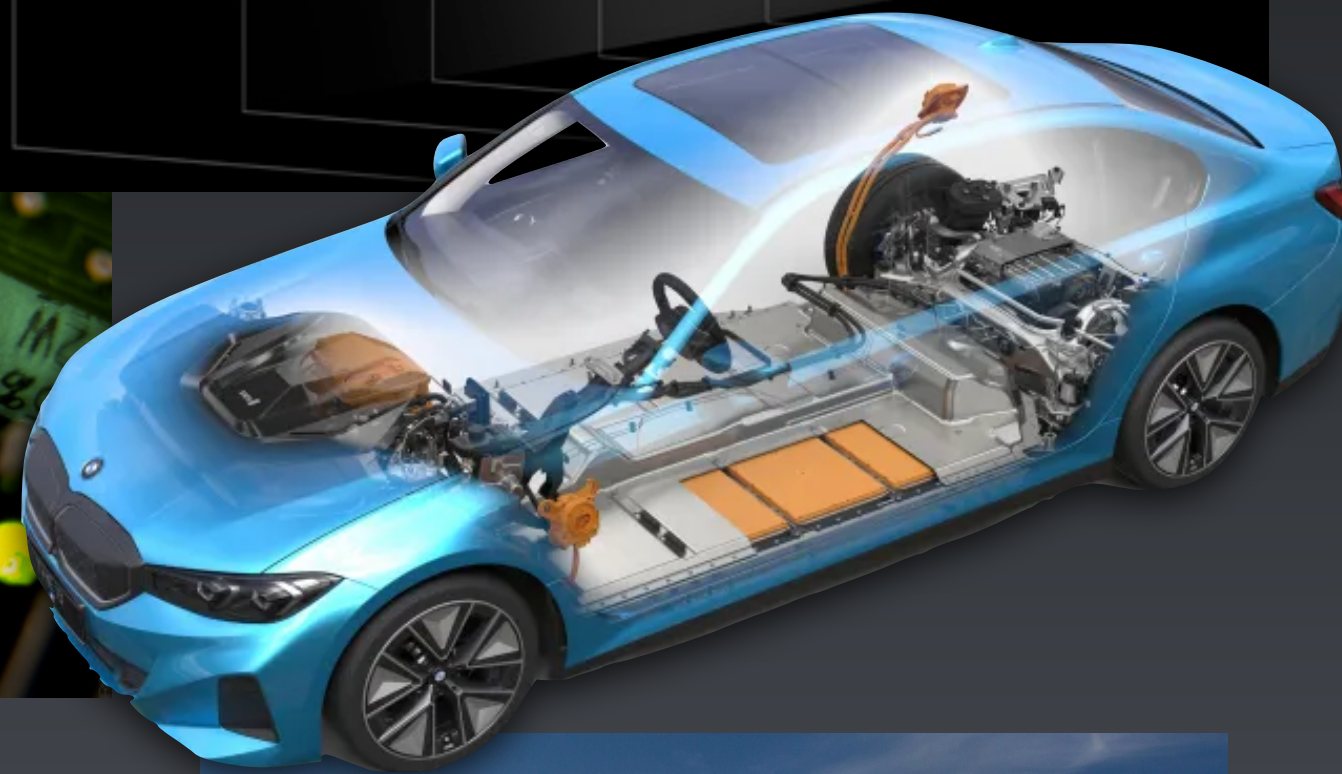
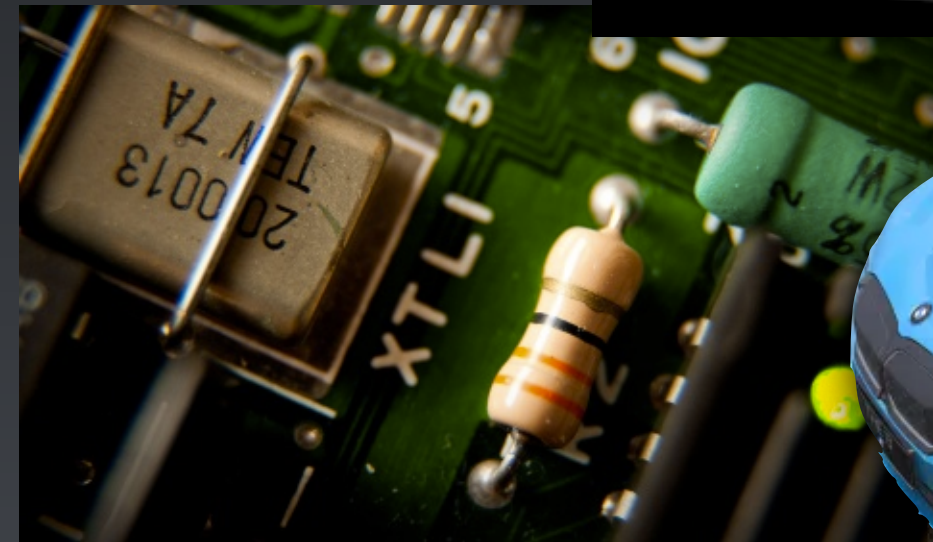
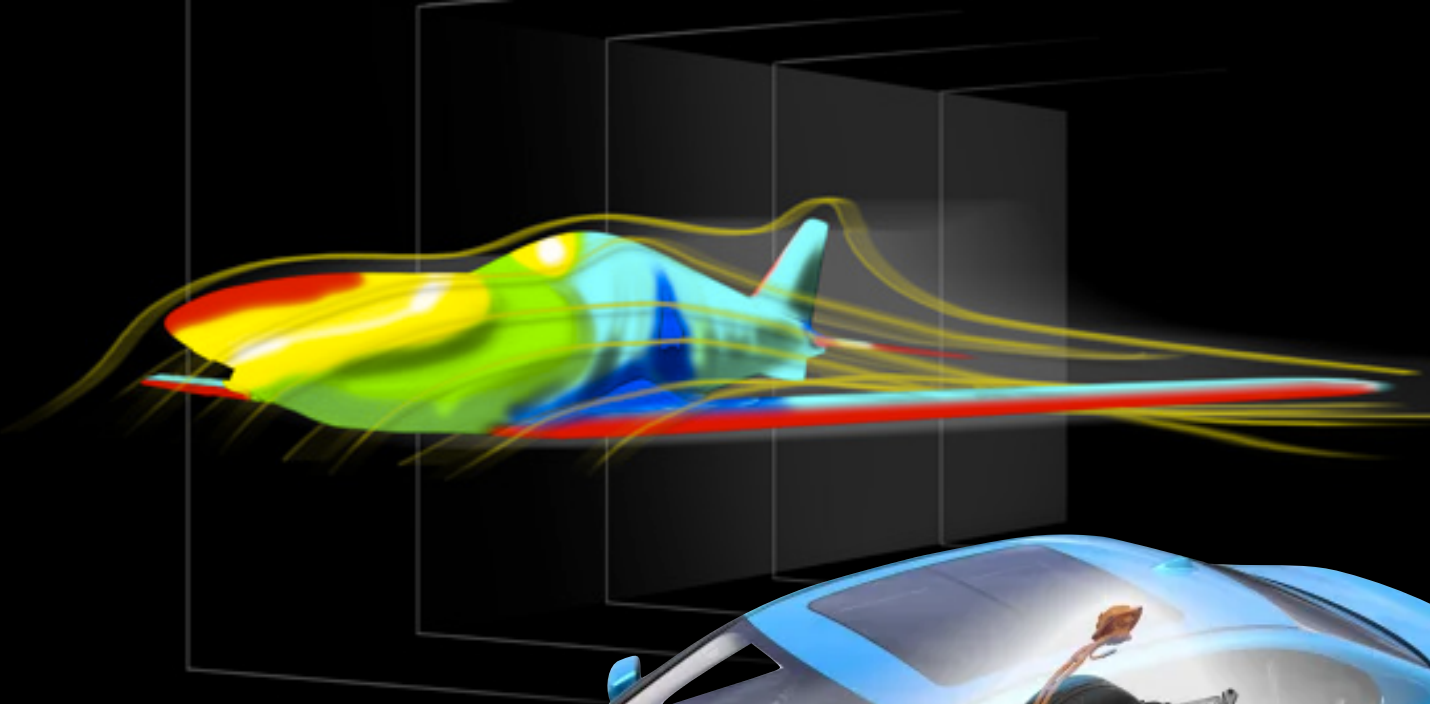




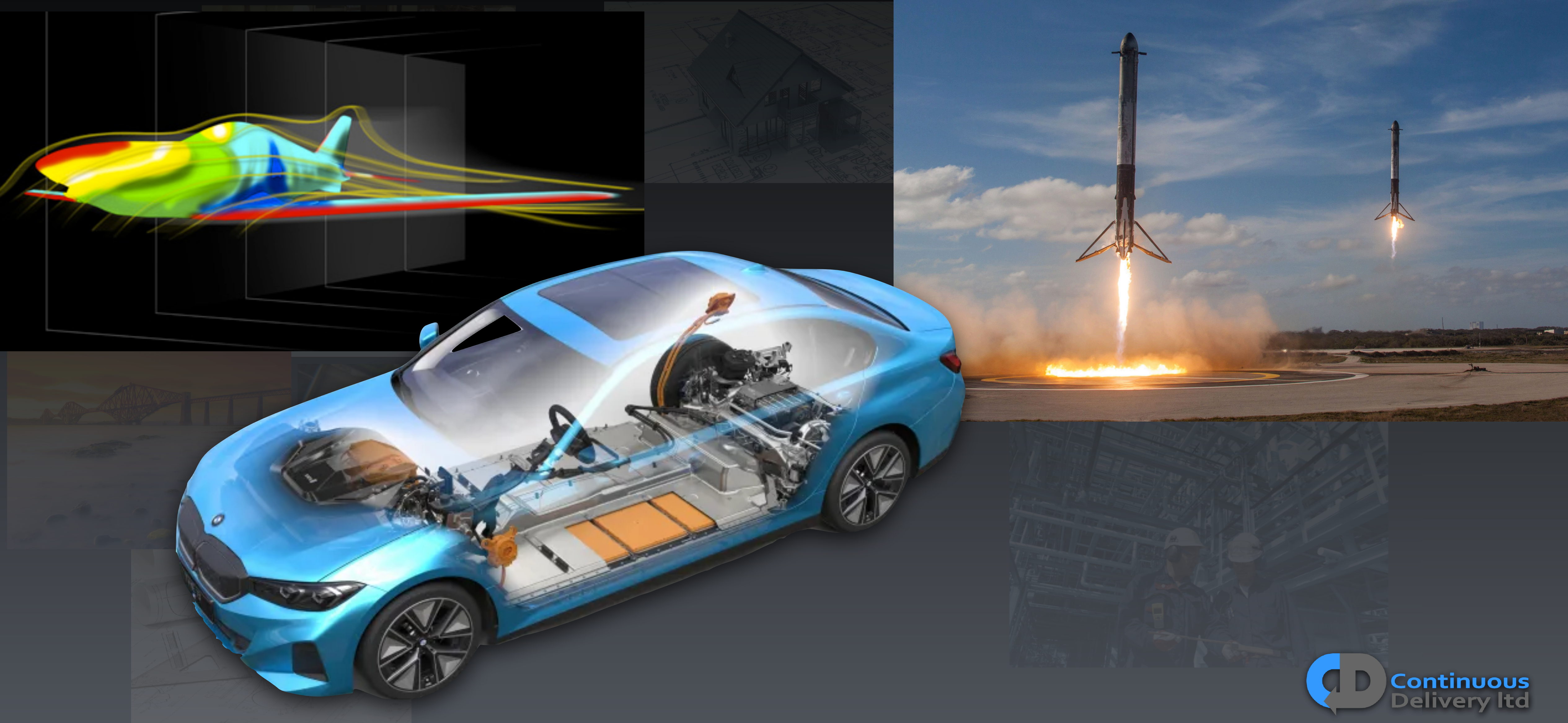
All Engineering is not the same!



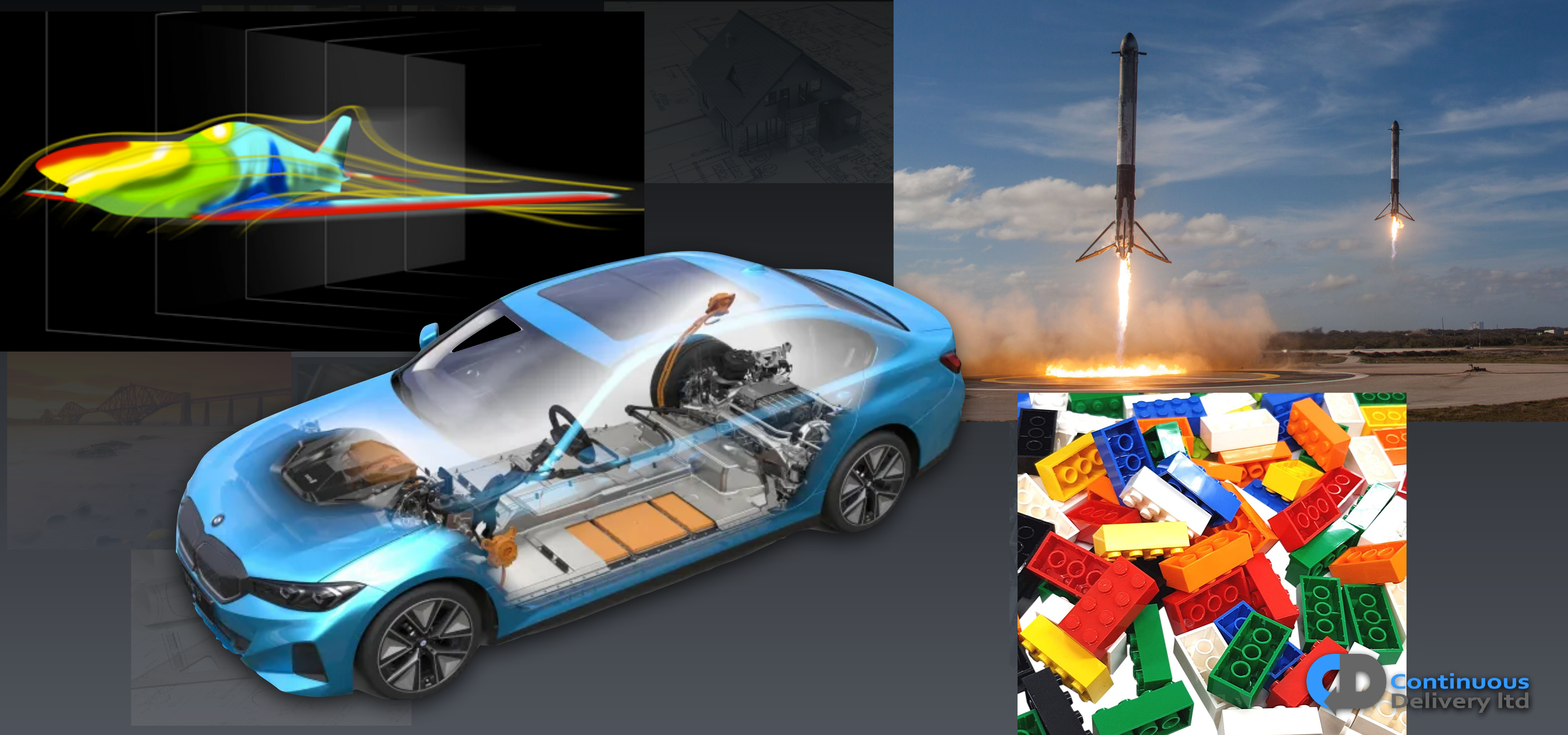
Learning From Other Engineering Disciplines



Learning From Other Engineering Disciplines



Learning From Other Engineering Disciplines



Learning From Other Engineering Disciplines

**What Does This
Mean for Us?**

What is 'Engineering'?

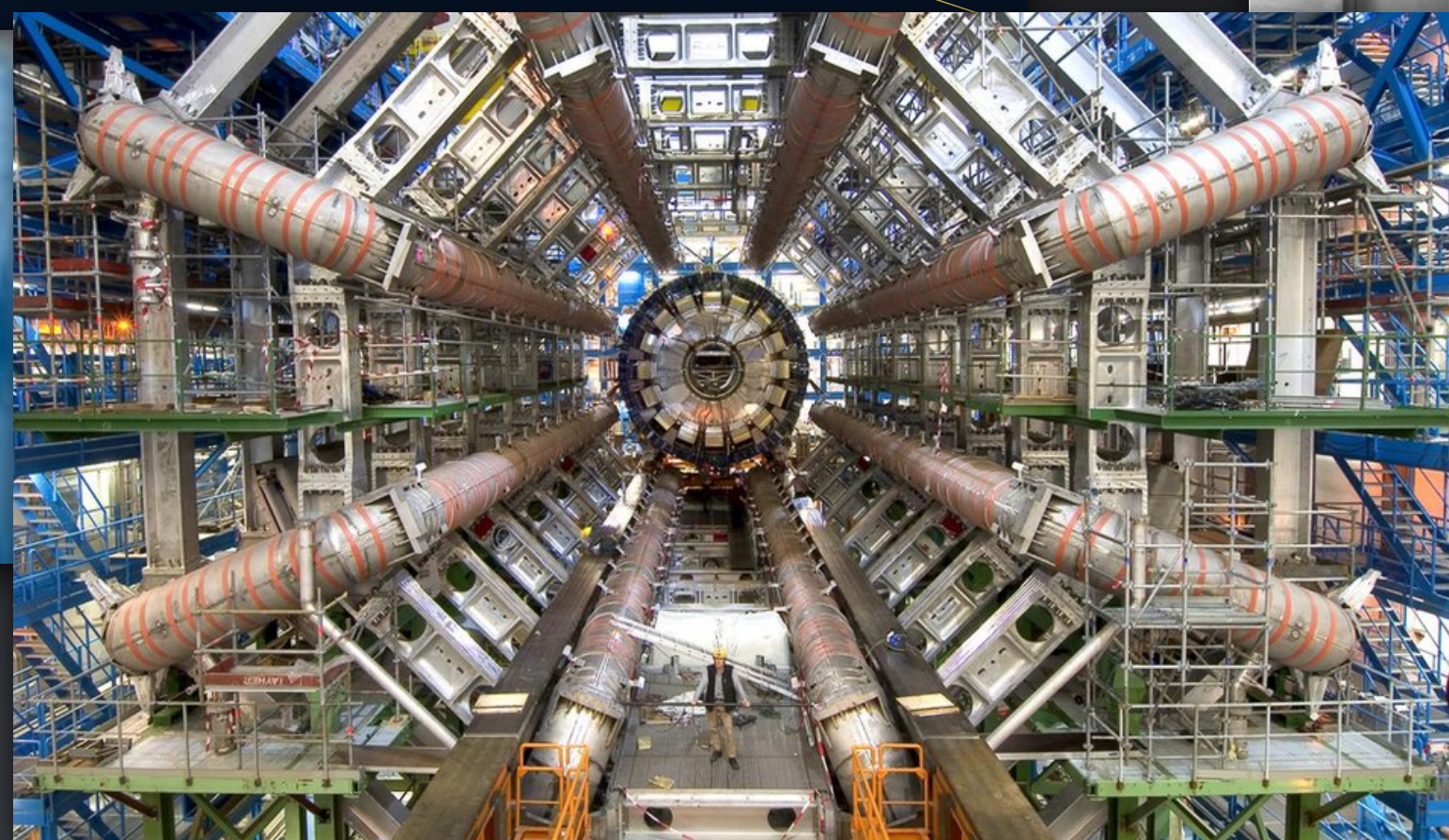
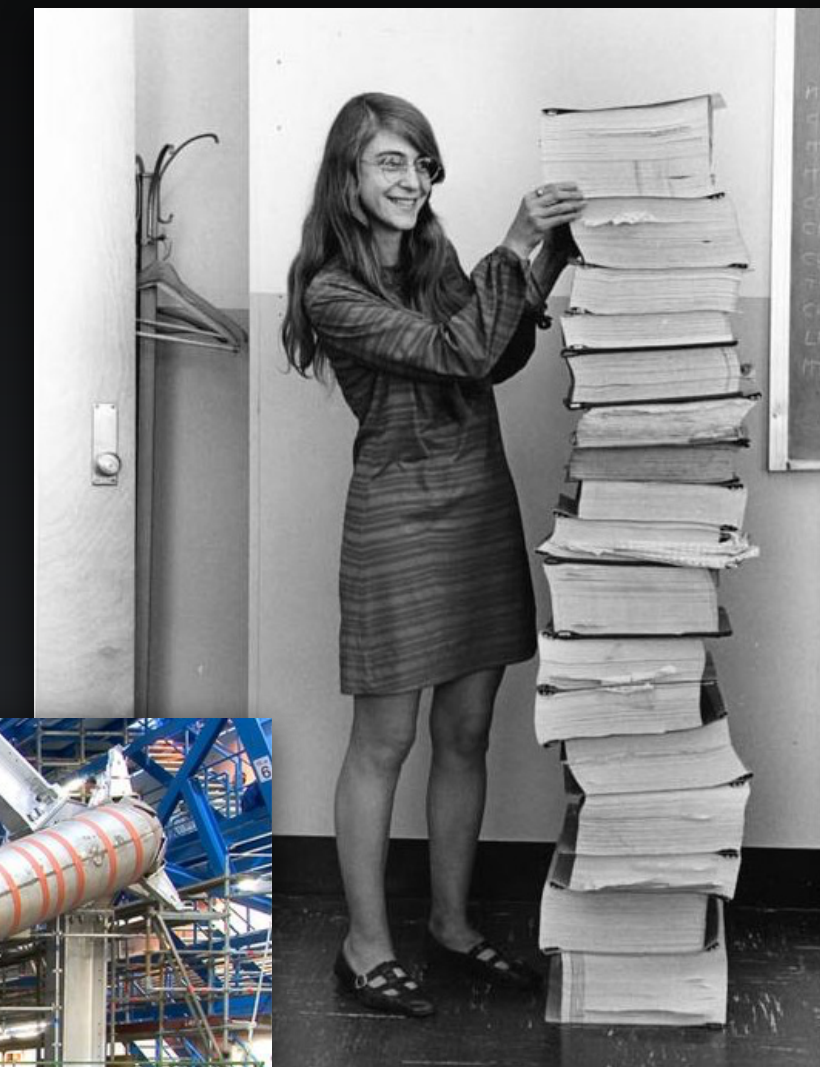
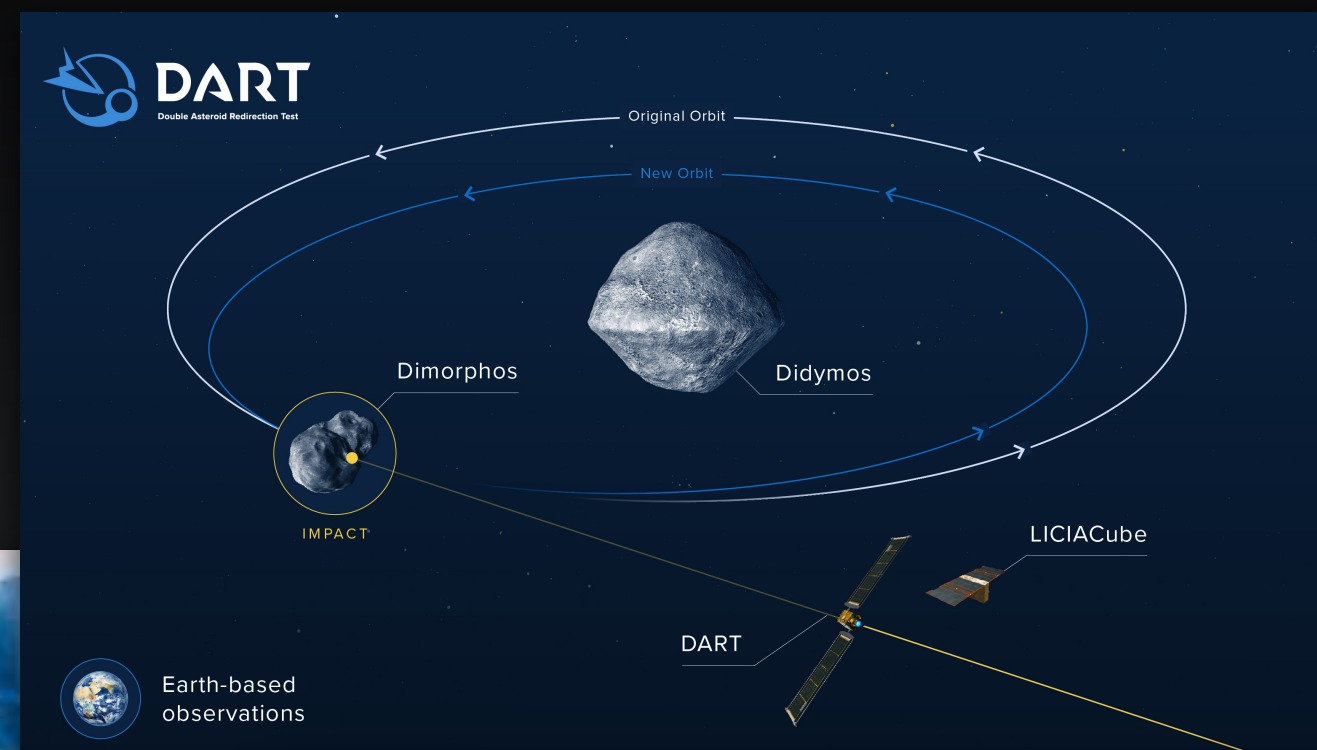
What is 'Engineering'?

Engineering is the application of an empirical, scientific approach to finding efficient solutions to practical problems.

(Dave Farley - Just Now!)

To Put It More Simply...

To Put It More Simply...

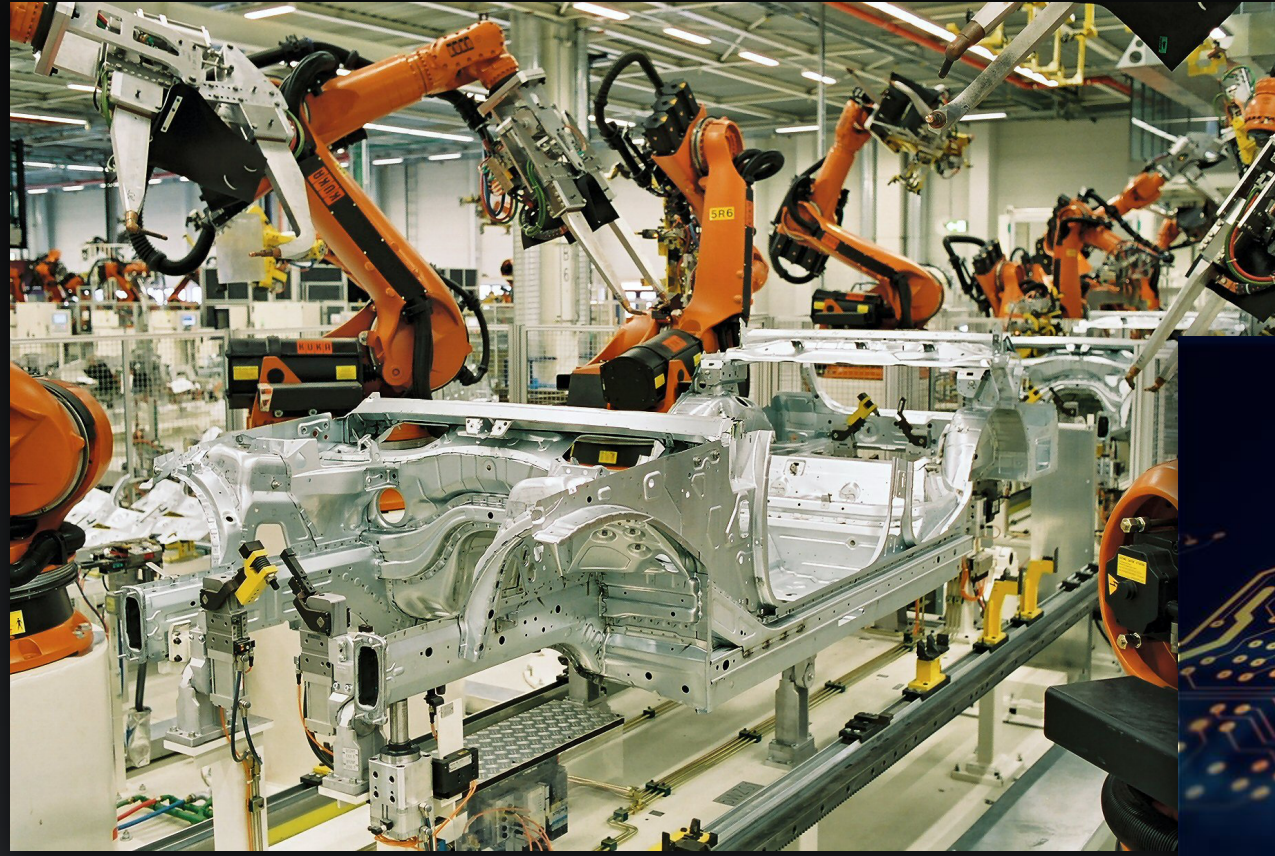


Engineering

is How We Solve the Hard Problems!

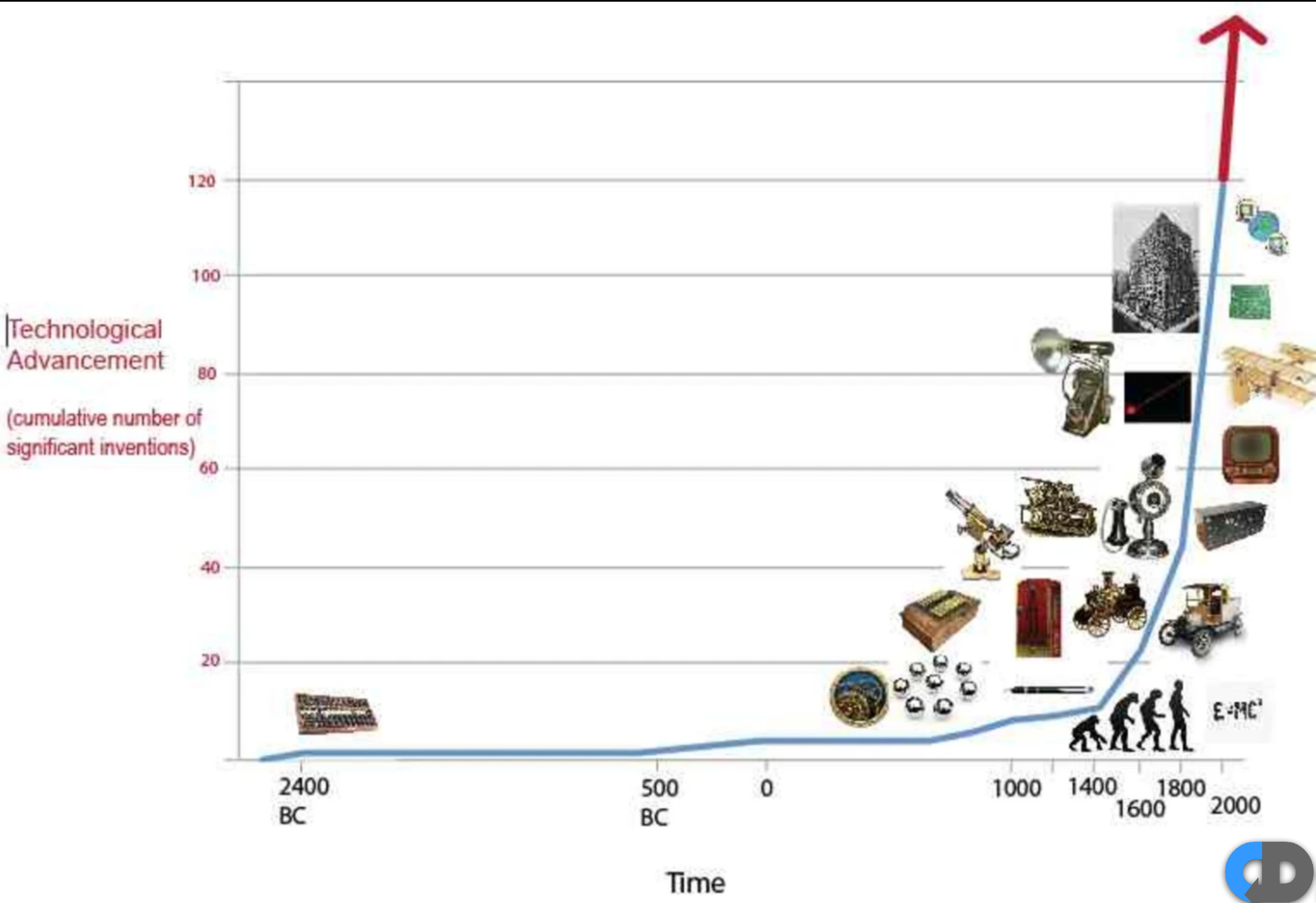
To Put It More Simply...

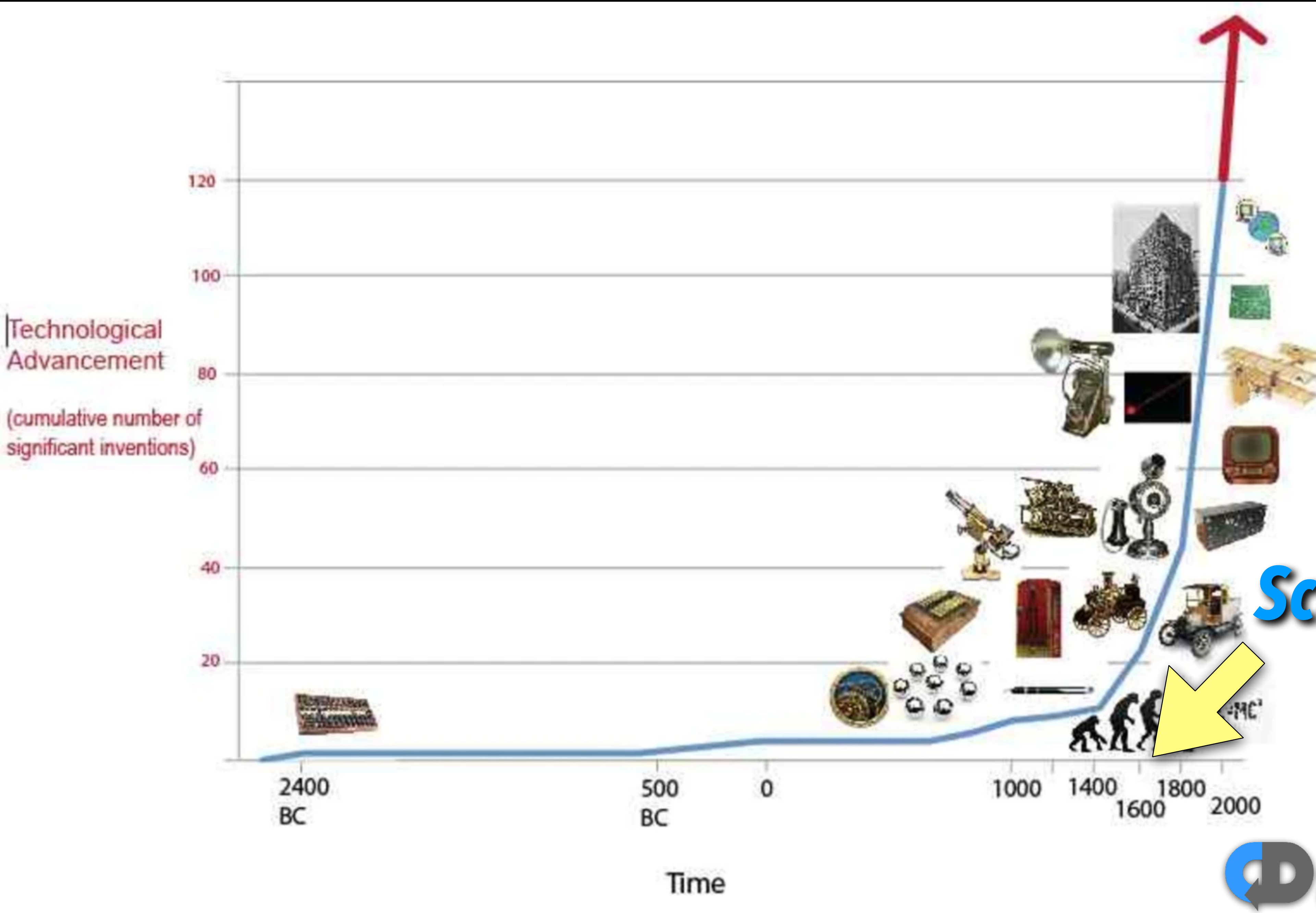
To Put It More Simply...



Engineering

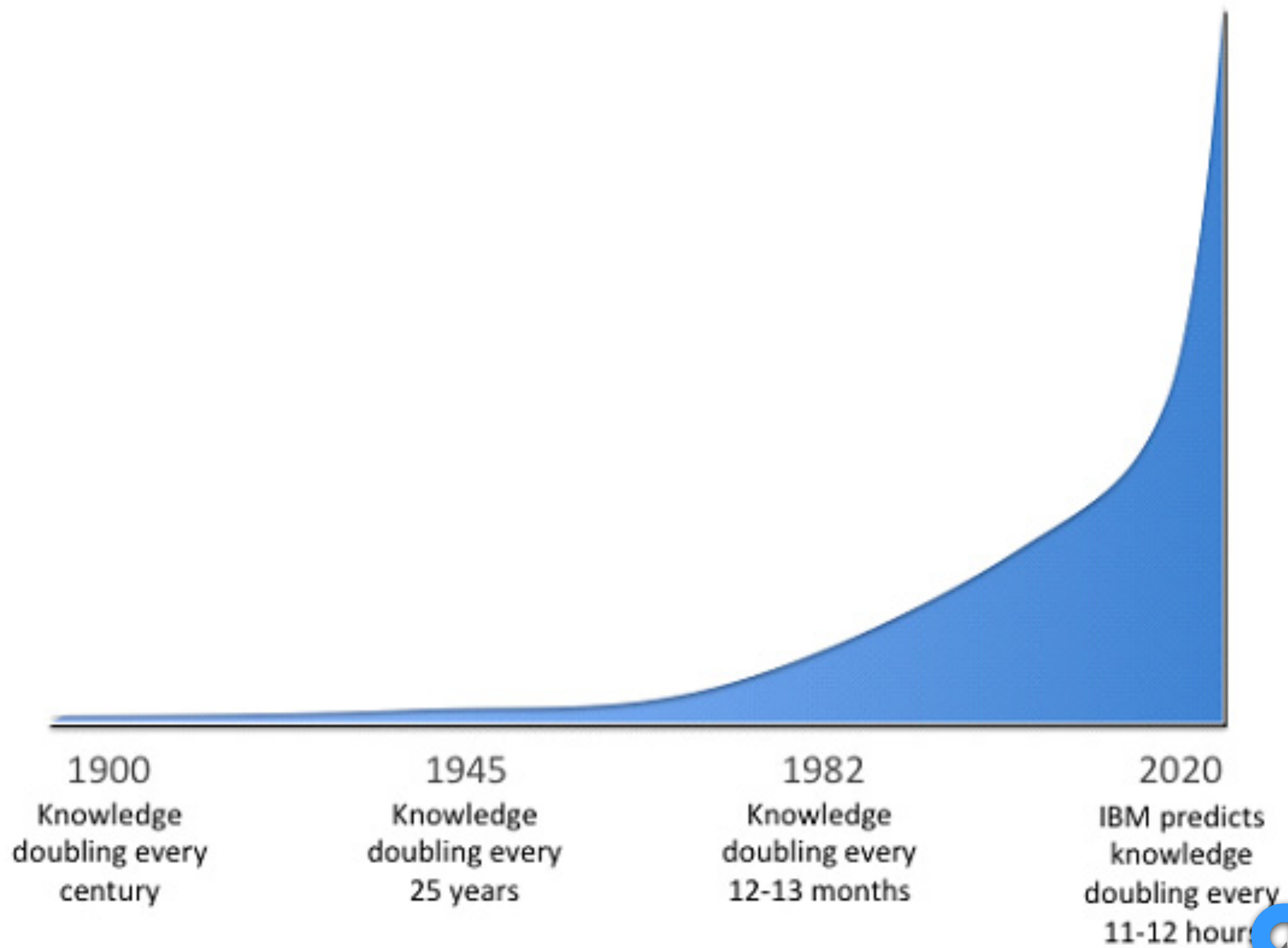
*The Foundation on Which Our
High-Tech Culture is Built!*





Science







1900

Knowledge doubling every century

1945

Knowledge doubling every 25 years

1982

Knowledge doubling every 12-13 months

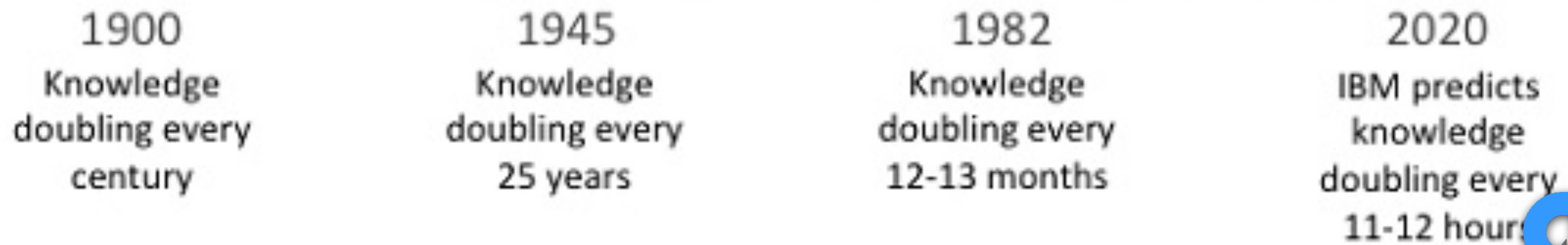
2020

IBM predicts knowledge doubling every 11-12 hours

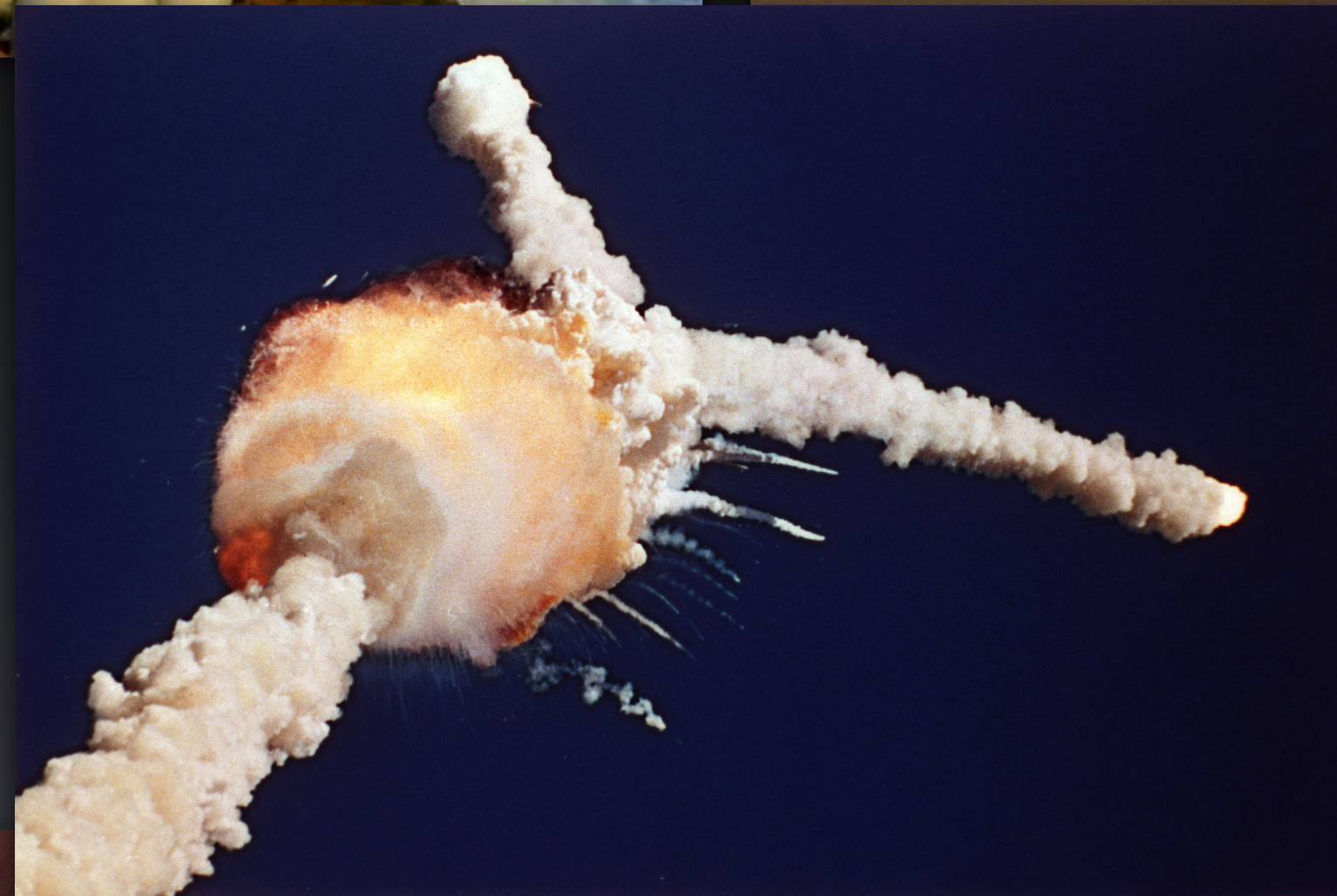
Science

Computers & Software

Science



What is Engineering Really For?



**It's Not to Define
Solutions!**



It Rules Out Bad Ideas!

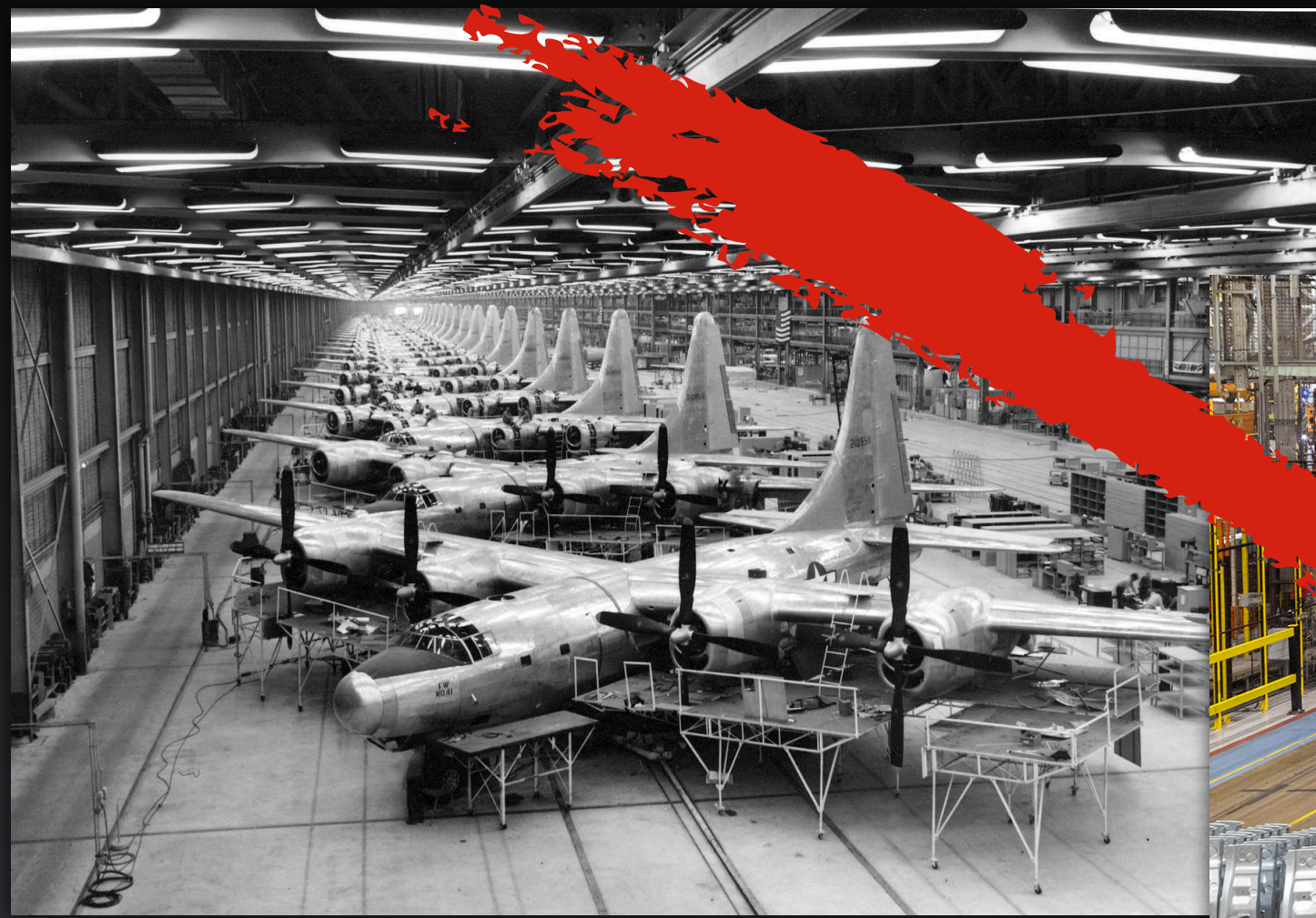


“The Things We Can’t Afford to Get Wrong”

“The Things We Can’t Afford to Get Wrong”



~~"The things We Can't Afford to Get Wrong"~~



~~This is NOT our Problem!~~


```
acceptance-testing - BookShoppingDsl.java [cd-acceptance-testing_test]
acceptance-testing src test java com cd acceptance dsl BookShoppingDsl checkout
BookShoppingDsl.java BookShoppingDsl.java BookShopDriver.java BookShopDrivers.java MyLocalBookStoreBookShopDriver.java Card.java
Project
  acceptance-testing ~/dev/acceptanc
    .gradle
    .idea
    build
    gradle
    src
      main
      test test sources root
        acceptance
          com
            cd
              acceptance
                examples
                  ExampleAmaz
                  ExampleBook
                  ExampleMyLo
                utils
          java
            com
              cd
                acceptance
                  dsl
                    drivers
                      BookShopping
                      Card
                      Channel
                      Channels
                      CucumberDsl
                      Dsl
                      Params
                      TestAnnotatio
                    feature
                    resources
          acceptance-testing.iml
          build.gradle
          gradlew
          gradlew.bat
          run-selenium.sh
          settings.gradle
        External Libraries
        Scratches and Consoles
  Favorites
  Structure
  Z:
    TODO
    Build
    Git
    Services
    Terminal
    SonarLint
    Event Log
  IntelliJ IDEA 2020.2.1 available: // Update... (yesterday 19:58)
  48:82 LF UTF-8 4 spaces master
```

```
1 package com.cd.acceptance.dsl;
2
3 import com.cd.acceptance.dsl.drivers.BookShopDrivers;
4
5 public class BookShoppingDsl
6 {
7     private final BookShopDrivers driver;
8
9     public BookShoppingDsl(BookShopDrivers drivers)
10    {
11        this.driver = drivers;
12    }
13
14    public void searchForBook(String... args)
15    {
16        Params params = new Params(args);
17        String title = params.Optional( name: "title", defaultValue: "Continuous Delivery");
18
19        driver.searchForBook(title);
20    }
21
22    public void selectBook(String... args)
23    {
24        Params params = new Params(args);
25        String author = params.Optional( name: "author", defaultValue: "David Farley");
26
27        driver.selectBook(author);
28    }
29
30    public void addSelectedItemToShoppingBasket() { driver.addSelectedItemToShoppingBasket(); }
31
32    public void assertItemListedInShoppingBasket(String... args)
33    {
34        Params params = new Params(args);
35        String item = params.Optional( name: "item", defaultValue: "Continuous Delivery");
36
37        driver.assertListedInShoppingBasket(item);
38    }
39
40    public void checkout(String... args)
41    {
42        Params params = new Params(args);
43        String item = params.Optional( name: "item", defaultValue: "Continuous Delivery");
44        String price = params.Optional( name: "price", defaultValue: "£10.00");
45        Card card = parseCard(params.Optional( name: "card", defaultValue: "1234 5678 9101 0001 12/23 007"));
46
47        driver.checkOut(item, price, card);
48    }
49
50    public void assertItemPurchased(String... args)
51    {
52        Params params = new Params(args);
53    }
54
```

```
acceptance-testing - BookShoppingDsl.java [cd-acceptance-testing_test]
acceptance-testing | src | test | java | com | cd | acceptance | dsl | BookShoppingDsl | checkout
BookShoppingDsl.java | BookShoppingDsl.java | BookShopDriver.java | BookShopDrivers.java | MyLocalBookStoreBookShopDriver.java | Card.java
Project | .gradle | .idea | build | gradle | src | main | test | test sources root | acceptance | com | cd | acceptance | examples | ExampleAmaz | ExampleBook | ExampleMyLo | utils | java | com | cd | acceptance | drivers | BookShopping | Card | Channel | Channels | CucumberDsl | Dsl | Params | TestAnnotatio | feature | resources | acceptance-testing.iml | build.gradle | gradlew | gradlew.bat | run-selenium.sh | settings.gradle | External Libraries | Scratches and Consoles
Z: Structure | Z: Favorites
IntelliJ IDEA 2020.2.1 available: // Update... (yesterday 19:58) | 48-82 | LF | UTF-8 | 4 spaces | master | Event Log

package com.cd.acceptance.dsl;

import com.cd.acceptance.dsl.drivers.BookShopDrivers;

public class BookShoppingDsl
{
    private final BookShopDrivers driver;

    public BookShoppingDsl(BookShopDrivers drivers)
    {
        this.driver = drivers;
    }

    public void searchForBook(String... args)
    {
        Params params = new Params(args);
        String title = params.Optional( name: "title", defaultValue: "Continuous Delivery");

        driver.searchForBook(title);
    }

    public void selectBook(String... args)
    {
        Params params = new Params(args);
        String author = params.Optional( name: "author", defaultValue: "David Farley");

        driver.selectBook(author);
    }

    public void addSelectedItemToShoppingBasket() { driver.addSelectedItemToShoppingBasket(); }

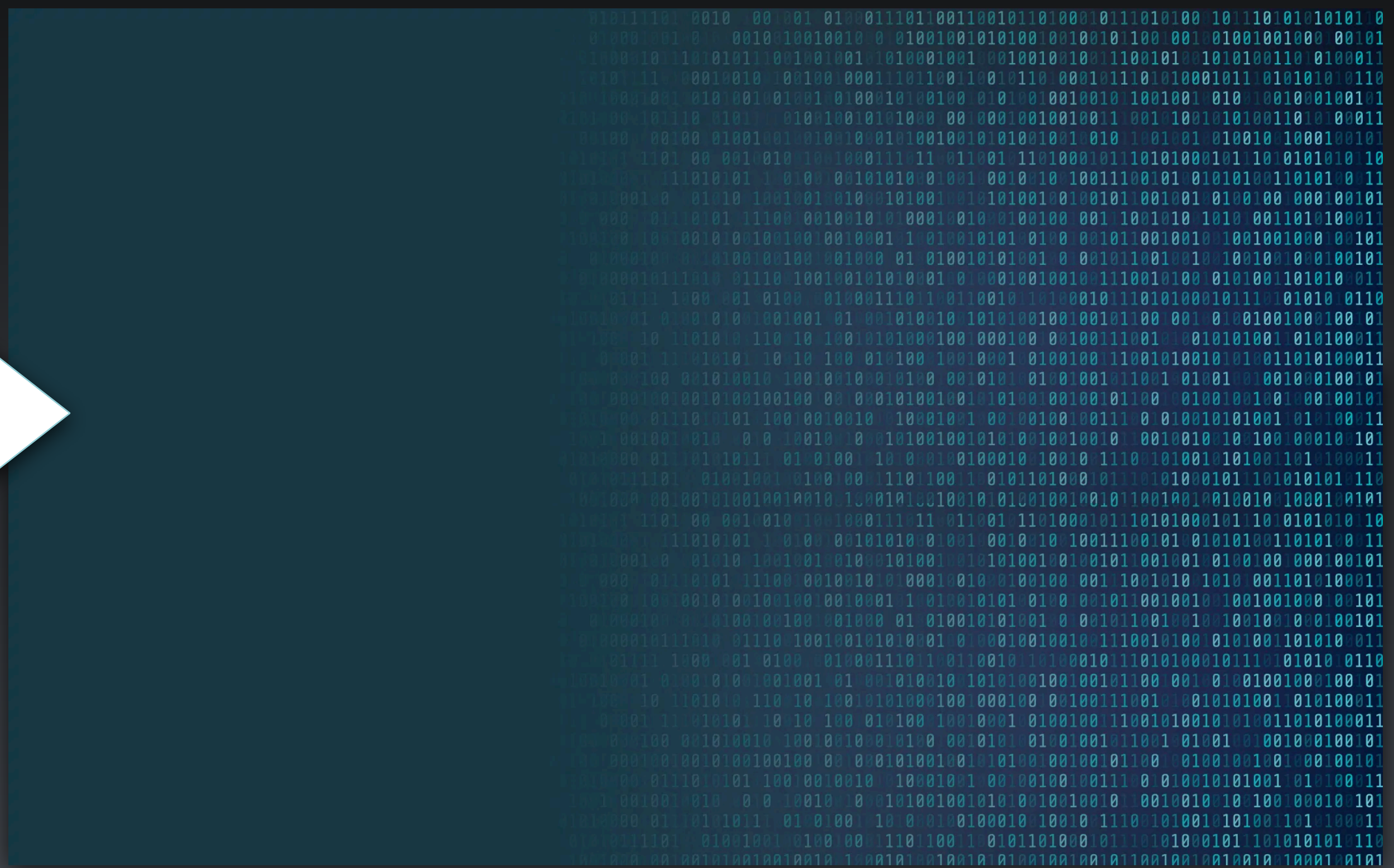
    public void assertItemListedInShoppingBasket(String... args)
    {
        Params params = new Params(args);
        String item = params.Optional( name: "item", defaultValue: "Continuous Delivery");

        driver.assertListedInShoppingBasket(item);
    }

    public void checkout(String... args)
    {
        Params params = new Params(args);
        String item = params.Optional( name: "item", defaultValue: "Continuous Delivery");
        String price = params.Optional( name: "price", defaultValue: "£10.00");
        Card card = parseCard(params.Optional( name: "card", defaultValue: "1234 5678 9101 0001 12/23 007"));

        driver.checkout(item, price, card);
    }

    public void assertItemPurchased(String... args)
    {
        Params params = new Params(args);
    }
}
```



```

1 package com.cd.acceptance.dsl;
2
3 import com.cd.acceptance.dsl.drivers.BookShopDrivers;
4
5 public class BookShoppingDsl
6 {
7     private final BookShopDrivers driver;
8
9     public BookShoppingDsl(BookShopDrivers drivers)
10    {
11        this.driver = drivers;
12    }
13
14    public void searchForBook(String... args)
15    {
16        Params params = new Params(args);
17        String title = params.Optional( name: "title", defaultValue: "Continuous Delivery");
18
19        driver.searchForBook(title);
20    }
21
22    public void selectBook(String... args)
23    {
24        Params params = new Params(args);
25        String author = params.Optional( name: "author", defaultValue: "David Farley");
26
27        driver.selectBook(author);
28    }
29
30    public void addSelectedItemToShoppingBasket() { driver.addSelectedItemToShoppingBasket(); }
31
32    public void assertItemListedInShoppingBasket(String... args)
33    {
34        Params params = new Params(args);
35        String item = params.Optional( name: "item", defaultValue: "Continuous Delivery");
36
37        driver.assertListedInShoppingBasket(item);
38    }
39
40    public void checkOut(String... args)
41    {
42        Params params = new Params(args);
43        String item = params.Optional( name: "item", defaultValue: "Continuous Delivery");
44        String price = params.Optional( name: "price", defaultValue: "£10.00");
45        Card card = parseCard(params.Optional( name: "card", defaultValue: "1234 5678 9101 0001 12/23 007"));
46
47        driver.checkOut(item, price, card);
48    }
49
50    public void assertItemPurchased(String... args)
51    {
52        Params params = new Params(args);
53    }
54 }

```







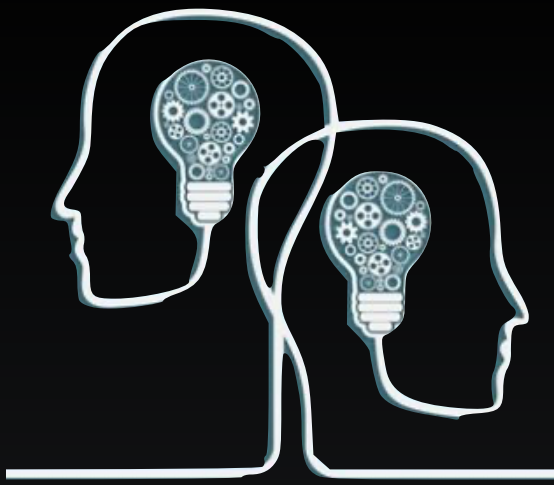
01011101 0010 001001 0100011101100110010110100010111010100101110101010101 0
010001010 0010010010010 0 010010010101001001001011001001001001000 00101
010001011101011001001001 1010001001 00100100100111001010010100110 0100011
010011100010010 00100 000111011001100110 0001011101010001011 0101010 0110
0101001001 0101001001001 0100010100100 010100100100101001001 01001001000100101
01000010110 010100 010010010101000 00100010010010011 00101001010011010 00011
00100 00100 0100100100100 0010001001010010010010 1001001 010010 1000100101
010101101 00 001010 1001000111011 011001 110100010111010100010111010101010
1101001 11010101 1 0100 00101010001001 0010 10 10011100101 01010100110101 0011
010010010 01010 100100100100100100101001001001011001001 0100100 000100101
1000001010101 11100 0010010 00010010 0100100 001 1001010 1010 0011010100011
0001001001001001001001001001001 10010010101 0100 00101100100100100100010101
0 01000100 0 1010010100 001000 01 010010101001 0 00101100100100100100100101
0100001011010 01110 100100101010001 0100010010010 1110010100 0101001101010 011
10001111 1000 001 0100 01000111011 0110010 1010001011101010001011101010 0110
0001001 0100 01001001001 01 001010010 1010100100100101100 001 01001001000100 01
0100010 1101010 110 10 10010101000100 000100 001001110010 0010101001 010100011
0000001 1101010 10 10 100 010100 10010001 0100100 11001010010101011010100011
010000100 001010010 100 0010 00100 0010101 01001001011001 0100100 001000100101
010000100100100100100100 00 00101001001010100100100101100 0100100100100100101
0100000111010101 1000010010 10001001 00100100100111 0 010010101001 010100011
1001 001001010 0 0 10010 10 00101001001010100100100101 0010010 100100010 101
0100000 011011011 01 01001 10 00100100010 10010 1110 1010010101001101100011
0100111010 0100100100100 1110110011 01011010001011101010001011101010101 110
01001000 001001010010010 10001010010010101001001011001001001001001000100101
010011101 00 0010010 1001000111011 011001 110100010111010100010111010101010
01001001 0 01010 1001001001001001001001010100100100101 001001 0100100 000100101
0000001010101 11100 0010010 00010010 0100100 001 1001010 1010 0011010100011
01010 10010010101001001001001 10010010101 0100 00101100100100100100100010101
0 01000100 0 10100100100 001000 01 010010101001 0 00101100100100100100100100101
0100001011010 01110 100100101010001 0100010010010 1110010100 0101001101010 011
10001111 1000 001 0100 01000111011 0110010 101000101110101000101110101010 0110
0001001 0100 01001001001 01 001010010 1010100100100101100 001 01001001000100 01
0100010 1101010 110 10 10010101000100 000100 001001110010 0010101001 010100011
0000001 1101010 10 10 100 010100 10010001 0100100 11001010010101011010100011
010000100 001010010 100 0010 00100 0010101 01001001011001 0100100 001000100101
010000100100100100100100 00 00101001001010100100100101100 0100100100100100101
0100000111010101 1000010010 10001001 00100100100111 0 010010101001 010100011
1001 001001010 0 0 10010 10 00101001001010100100100101 0010010 100100010 101
0100000 011011011 01 01001 10 00100100010 10010 1110 1010010101001101100011
01000111010 0100100100100 1110110011 01011010001011101010001011101010101 110
0100000 001001010010010 10001010010010101001001011001001001001001000100101
010011101 00 0010010 1001000111011 011001 110100010111010100010111010101010
01001001 0 01010 1001001001001001001001010100100100101 001001 0100100 000100101
0000001010101 11100 0010010 00010010 0100100 001 1001010 1010 0011010100011
01010 10010010101001001001001 10010010101 0100 00101100100100100100100010101
0 01000100 0 10100100100 001000 01 010010101001 0 00101100100100100100100100101
0100001011010 01110 100100101010001 0100010010010 1110010100 0101001101010 011
10001111 1000 001 0100 01000111011 0110010 101000101110101000101110101010 0110
0001001 0100 01001001001 01 001010010 1010100100100101100 001 01001001000100 01
0100010 1101010 110 10 10010101000100 000100 001001110010 0010101001 010100011
0000001 1101010 10 10 100 010100 10010001 0100100 11001010010101011010100011
010000100 001010010 100 0010 00100 0010101 01001001011001 0100100 001000100101
010000100100100100100100 00 00101001001010100100100101100 0100100100100100101
0100000111010101 1000010010 10001001 00100100100111 0 010010101001 010100011
1001 001001010 0 0 10010 10 00101001001010100100100101 0010010 100100010 101
0100000 011011011 01 01001 10 00100100010 10010 1110 1010010101001101100011
01000111010 0100100100100 1110110011 01011010001011101010001011101010101 110
0100000 00100010010010010 10001010010010100100100101100100100100100100010101

01011101 0010 001001 010011101100110010110100010111010100101110101010101 0
01000101 0 001010010010 0 010010010101001001001011001001001001000 00101
01000101110101110010010011010001001 0010010010011100101001010100110 0100011
0010111 00010010 00100 00011101100110 010110 0001011101010001011 0101010 0110
0101001001 0101001001001 0100010100100 010100100100101001001 01001001000100101
01000101110 01010 010010010101000 00100010010010011 0010100101010011010 00011
00100 00100 010010010010 0010001001010010010010 1001001 010010 1000100101
010101101 00 001010 100100011111 011001 110100010111010100010111010101010
1101001 11010101 1 0100100101010001001 001010 10011100101 01010100110101 0011
01001001 0 01010 10010011010001010010010101001001011001001 0100100 000100101
10000010110101 11100 0010010 00010010 0100100 0011001010 1010 0011010100011
10010 10010010101001001001001001 10010010101 0100 00101100100100100100010101
0 01000100 0 1010010100 001000 01 010010101001 0 00101100100100100100100101
01000010111010 01110 100100101010001 0100010010010 1110010100 0101001101010 011
10001111 1000 001 0100 01000111011 0110010 101000101110101000101110101010 0110
0001001 0100 01001001001 01 001010010 1010100100100101100 001 01001001000100 01
010010 1101010 110 10 10010101000100 000100 001001110010 0010101001 010100011
000001 1101010 10 10 100 010100 10010001 0100100 11001010010101011010100011
01000100 001010010 100 0010 00100 0010101 01001001011001 0100100 001000100101
010000100100100100100 00 00101001001010100100100101100 0100100100100100101
0100000111010101 10010010010 10001001 00100100100111 0 010010101001 010100011
1001 001001010 0 0 10010 10 00101001001010100100100101 0010010 100100010 101
0100000 0110110111 01 01001 10 00100100010 10010 1110 1010010101001101100011
010011101 010010010100 00 1110110011 01011010001011101010001011101010101 110
0001000 001001010010010 10001010010010 10100100100101100100100100 1000100101
010011101 00 0010010 100100011111 011001 110100010111010100010111010101010
01001001 0 01010 1001001101000101001001010100100100101 001001 0100100 000100101
00000010110101 11100 0010010 00010010 0100100 0011001010 1010 0011010100011
10010 10010010101001001001001 10010010101 0100 00101100100100100100100010101
0 01000100 0 10100100100 001000 01 010010101001 0 001011001001001001000100101
01000010111010 01110 100100101010001 0100010010010 1110010100 0101001101010 011
10001111 1000 001 0100 01000111011 0110010 101000101110101000101110101010 0110
0001001 0100 01001001001 01 001010010 1010100100100101100 001 01001001000100 01
010010 1101010 110 10 10010101000100 000100 001001110010 0010101001 010100011
000001 1101010 10 10 100 010100 10010001 0100100 11001010010101011010100011
01000100 001010010 100 0010 00100 0010101 01001001011001 0100100 001000100101
010000100100100100100 00 00101001001010100100100101100 0100100100100100101
0100000111010101 10010010010 10001001 00100100100111 0 010010101001 010100011
1001 001001010 0 0 10010 10 00101001001010100100100101 0010010 100100010 101
0100000 0110110111 01 01001 10 001001000010 10010 1110 1010010101001101100011
0100011101 010010010100 00 1110110011 01011010001011101010001011101010101 110
0001000 001001010010010 10001010010010 10100100100101100100100100 1000100101
010011101 00 0010010 100100011111 011001 110100010111010100010111010101010
01001001 0 01010 1001001101000101001001010100100100101 001001 0100100 000100101
00000010110101 11100 0010010 00010010 0100100 0011001010 1010 0011010100011
10010 10010010101001001001001 10010010101 0100 00101100100100100100100010101
0 01000100 0 10100100100 001000 01 010010101001 0 001011001001001001000100101
01000010111010 01110 100100101010001 0100010010010 1110010100 0101001101010 011
10001111 1000 001 0100 01000111011 0110010 101000101110101000101110101010 0110
0001001 0100 01001001001 01 001010010 1010100100100101100 001 01001001000100 01
010010 1101010 110 10 10010101000100 000100 001001110010 0010101001 010100011
000001 1101010 10 10 100 010100 10010001 0100100 11001010010101011010100011
01000100 001010010 100 0010 00100 0010101 01001001011001 0100100 001000100101
010000100100100100100 00 00101001001010100100100101100 0100100100100100101
0100000111010101 10010010010 10001001 00100100100111 0 010010101001 010100011
1001 001001010 0 0 10010 10 00101001001010100100100101 0010010 100100010 101
0100000 0110110111 01 01001 10 001001000010 10010 1110 1010010101001101100011
0100011101 010010010100 00 1110110011 01011010001011101010001011101010101 110
0001000 001001010010010 10001010010010 10100100100101100100100100 1000100101

DEPLOY

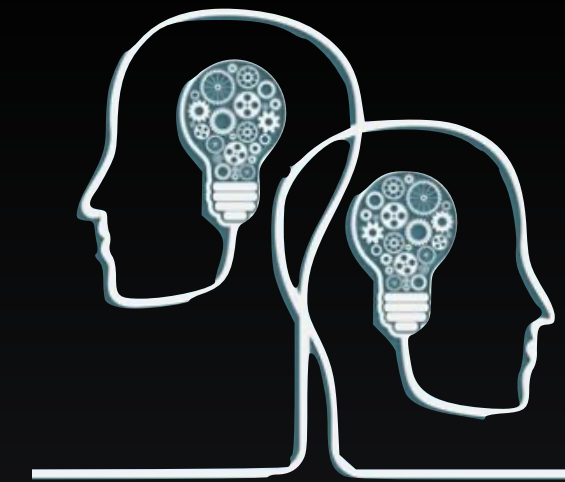
We Can Recreate Entire
Systems for Free!

DEPLOY



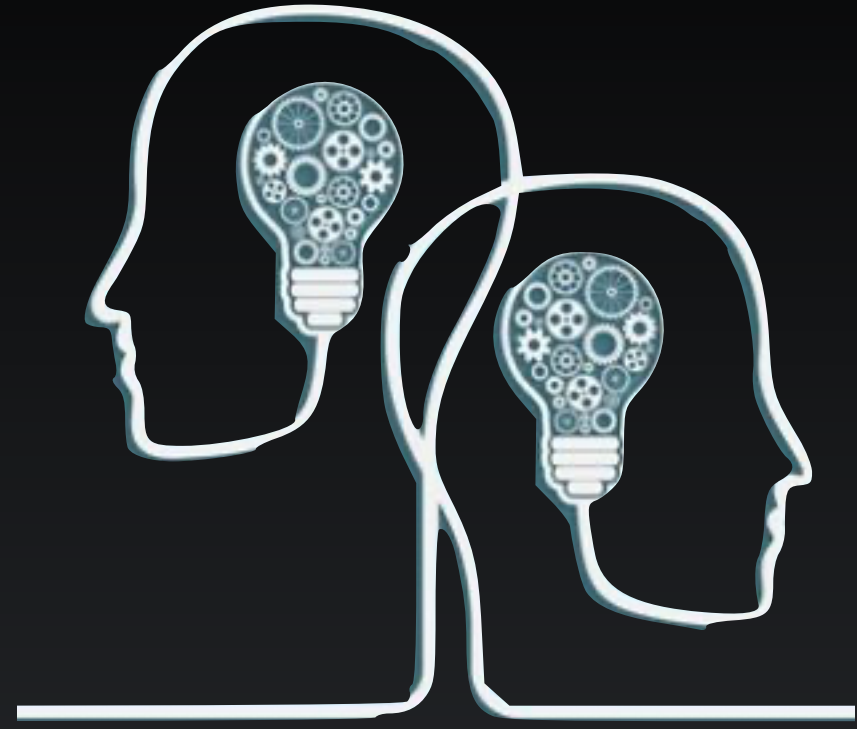
Engineering Exploration & Discovery ***IS*** About

Software Development Exploration ***IS*** About & Discovery



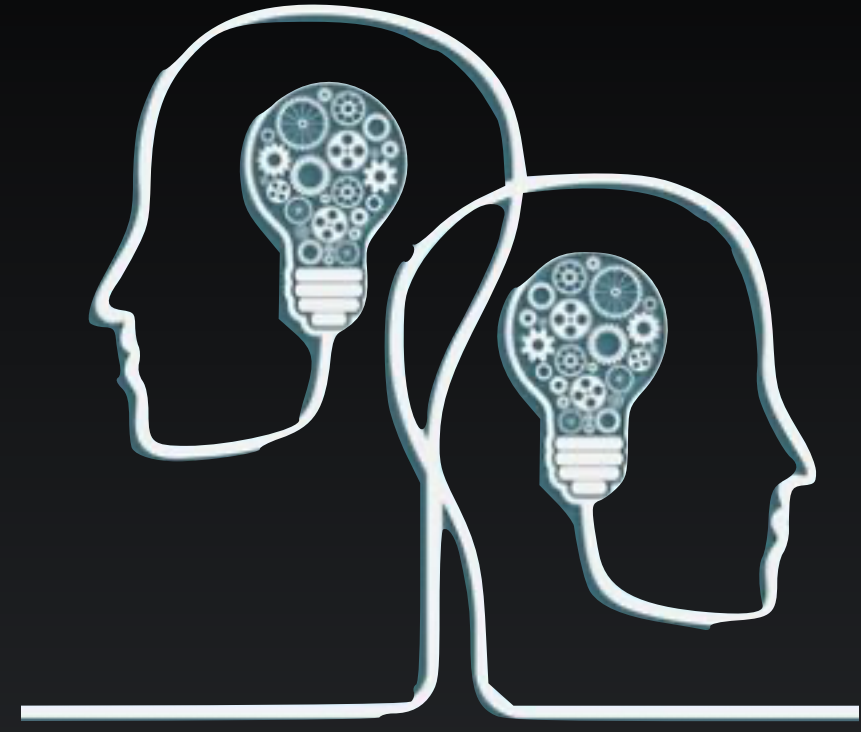
The Stuff That Works

The Stuff That Works



Optimise for Learning

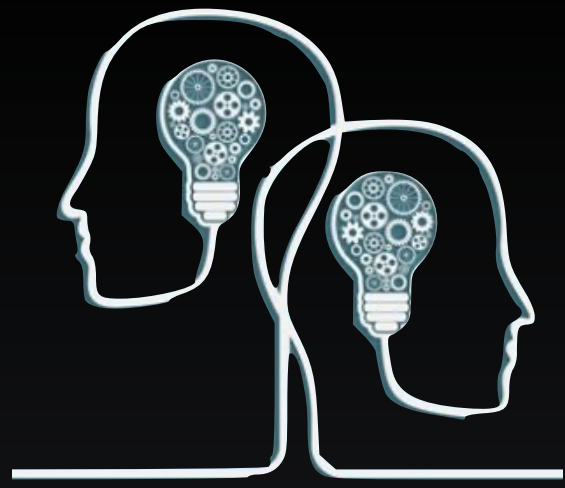
The Stuff That Works



Optimise for Learning

Optimise for Managing Complexity

Optimise for Learning

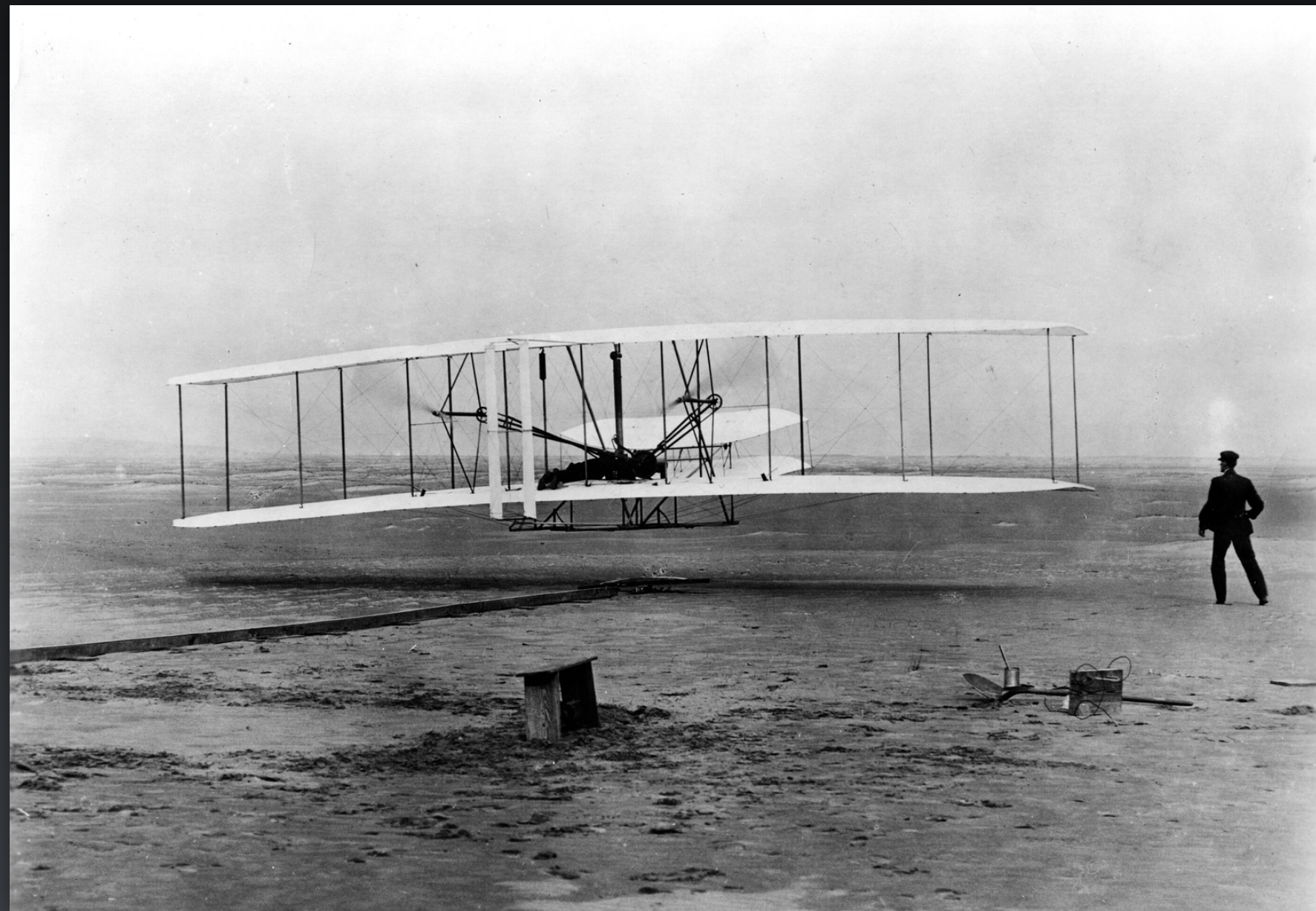
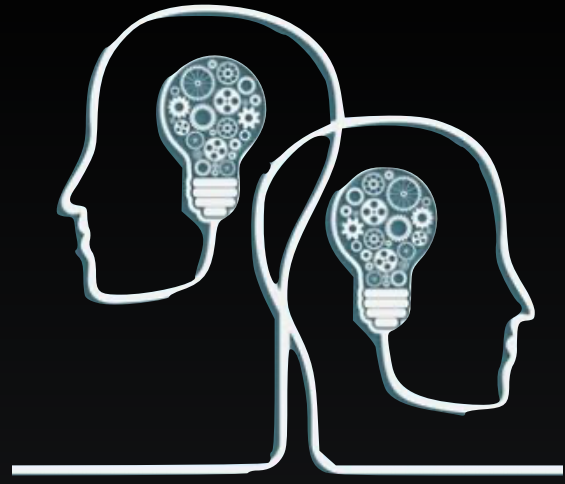


Iterative Experimental

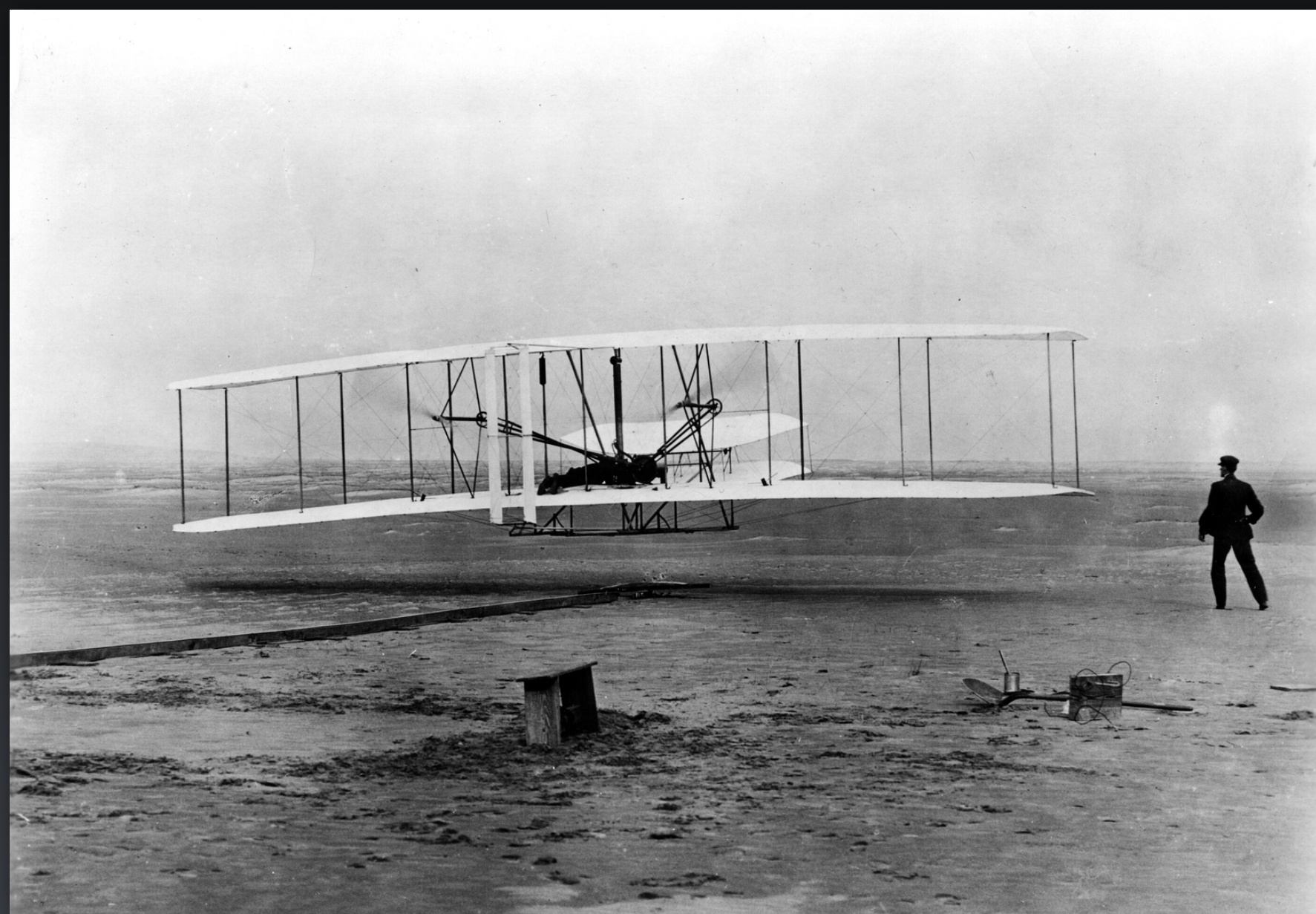
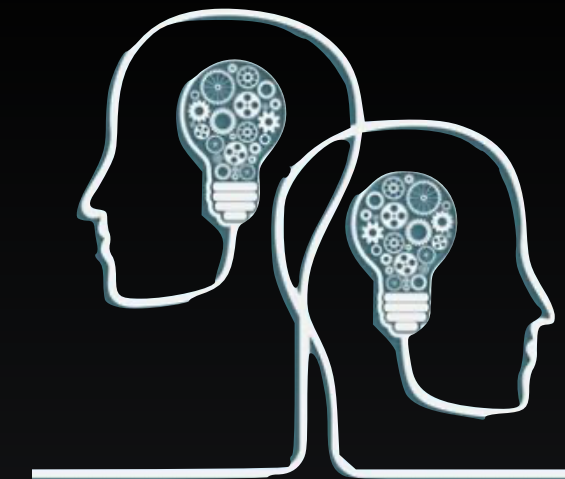
Feedback Driven

Incremental Empirical

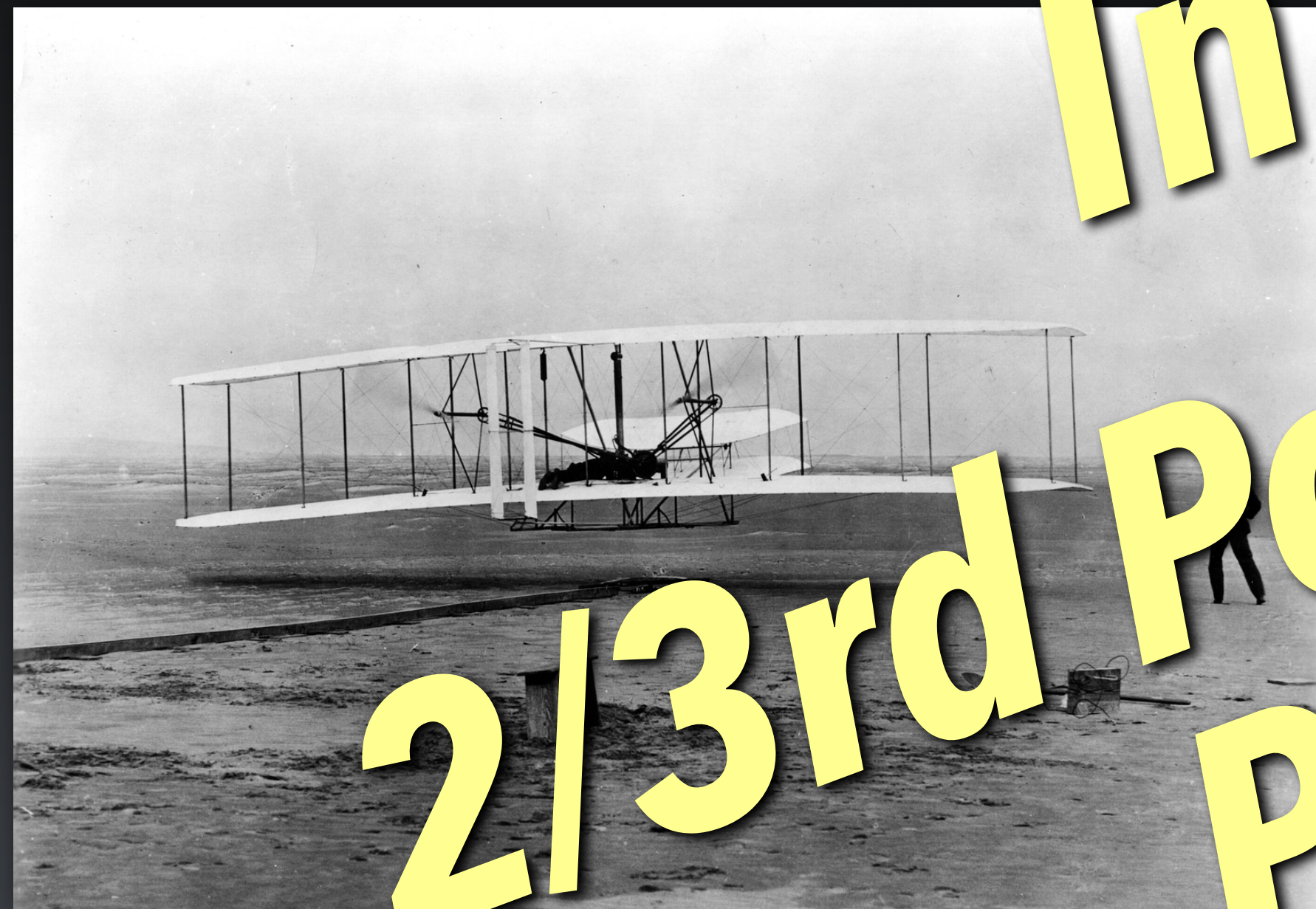
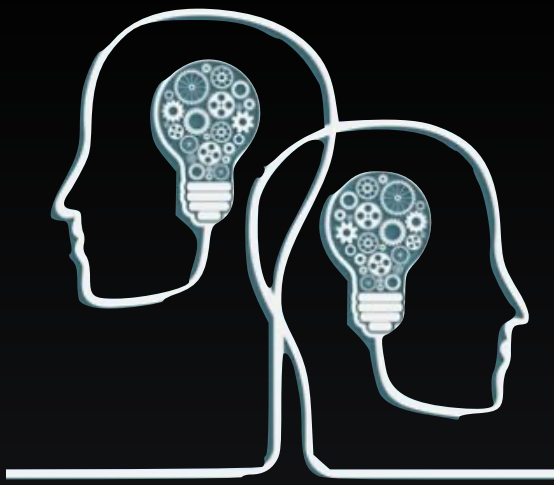
Iteration



Iteration



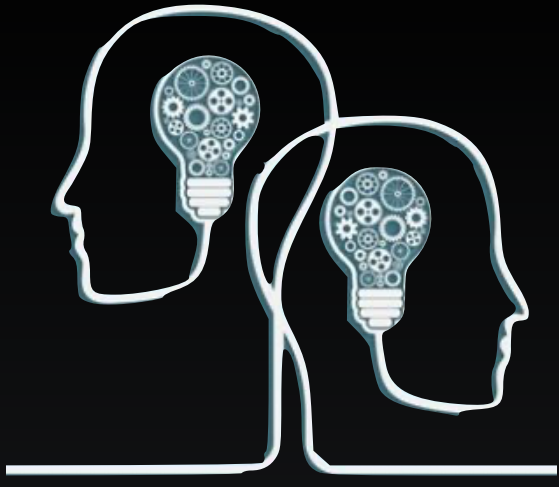
Iteration



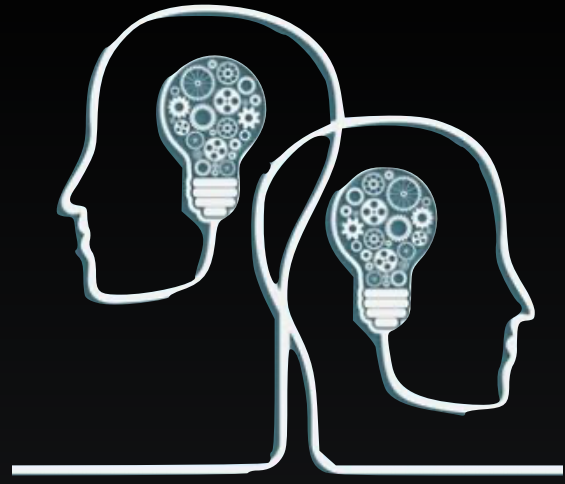
In 2017
2/3rd Population of the
Planet Flew
With no Fatalities!



Iteration

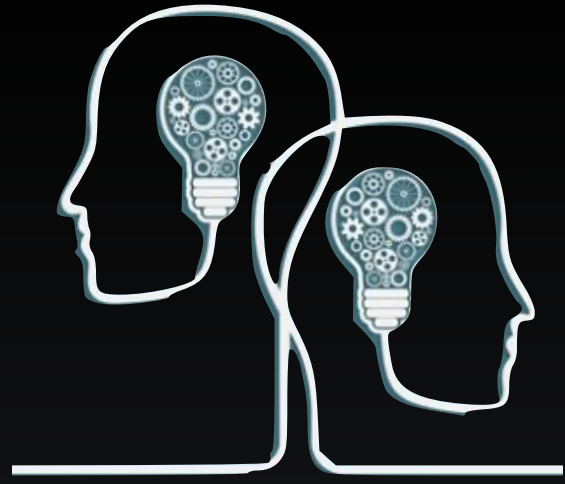


Iteration



The Goal

Iteration

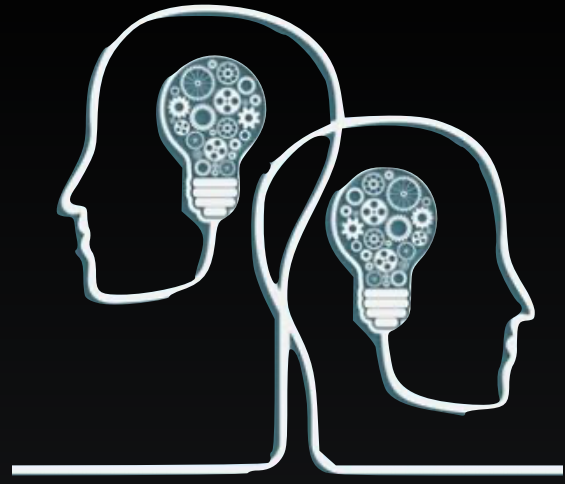


The Goal



Start

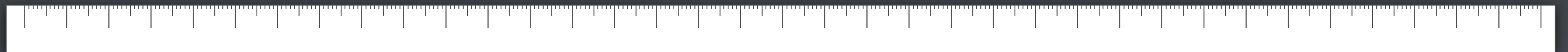
Iteration



The Goal

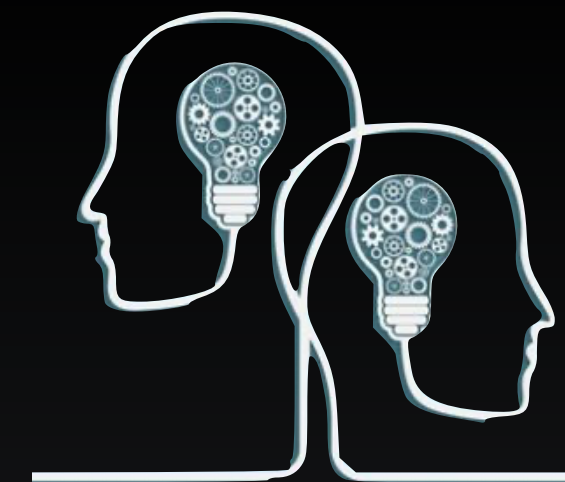


Start

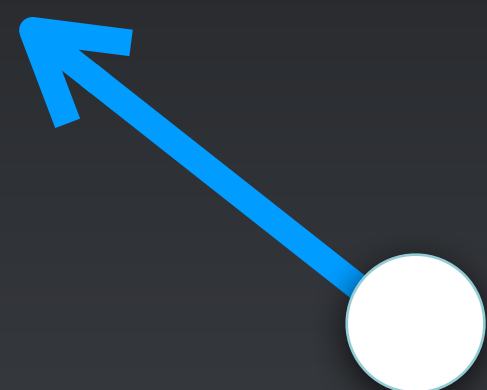


Fitness Function

Iteration



The Goal

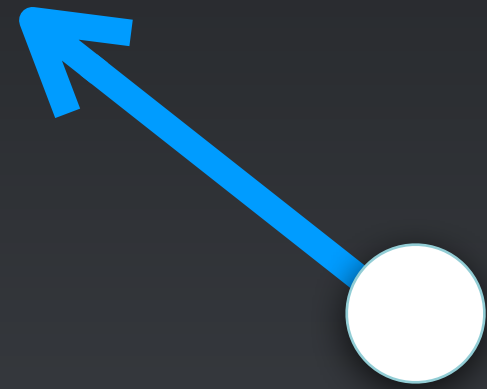
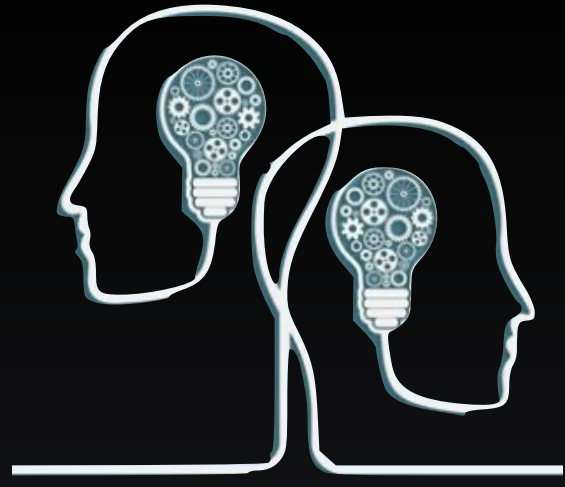


Start

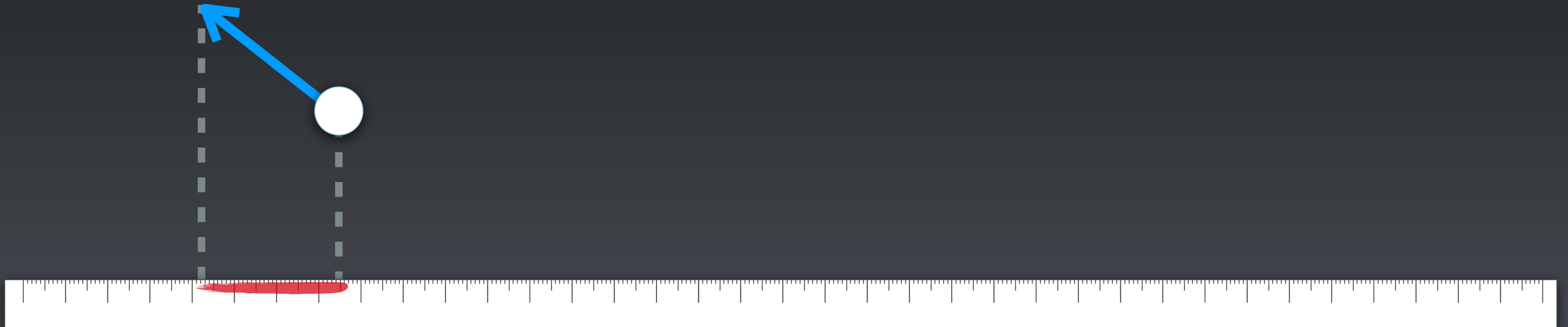
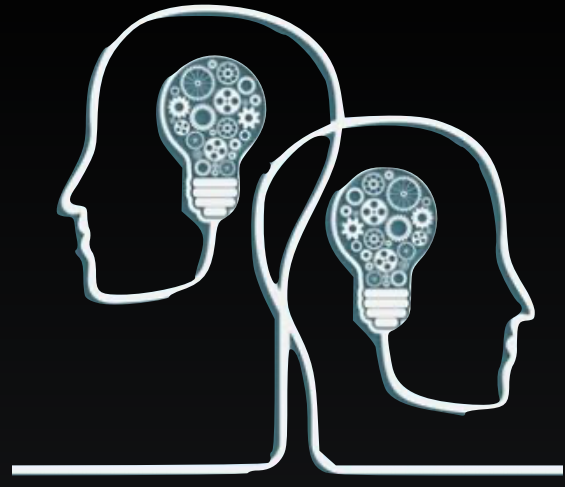


Fitness Function

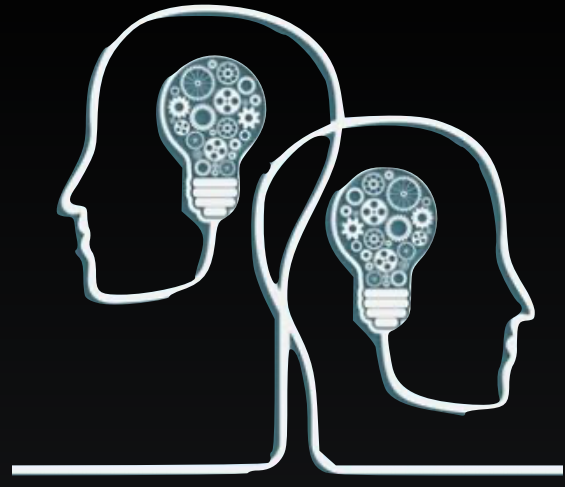
Iteration



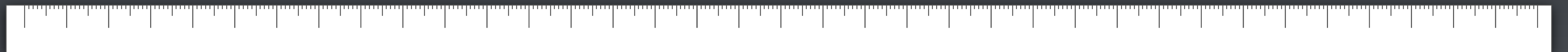
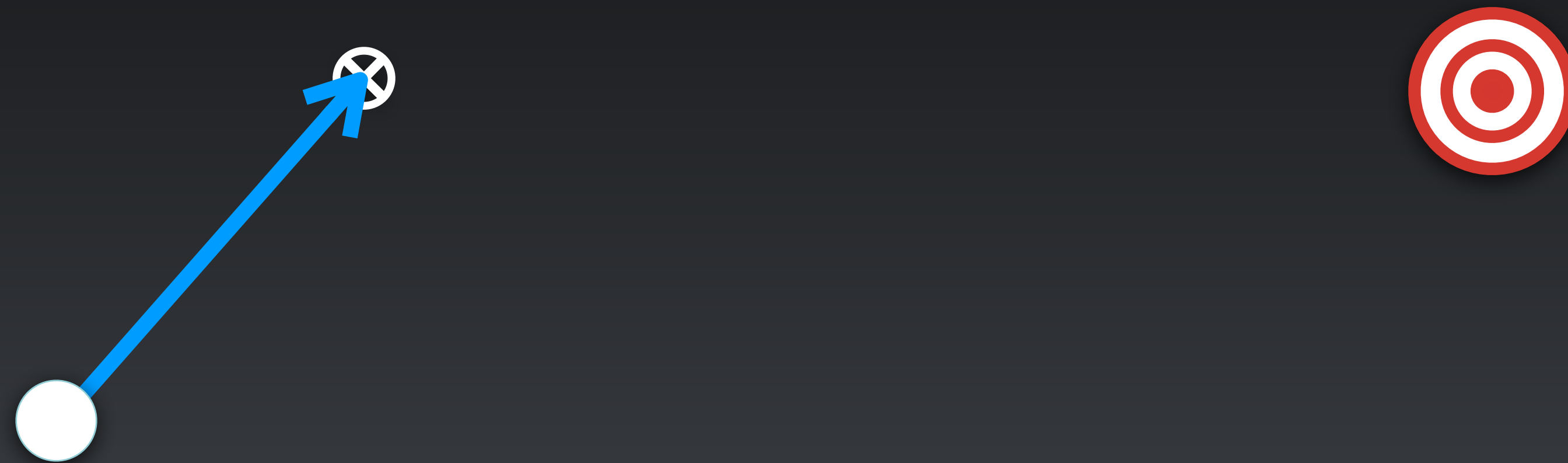
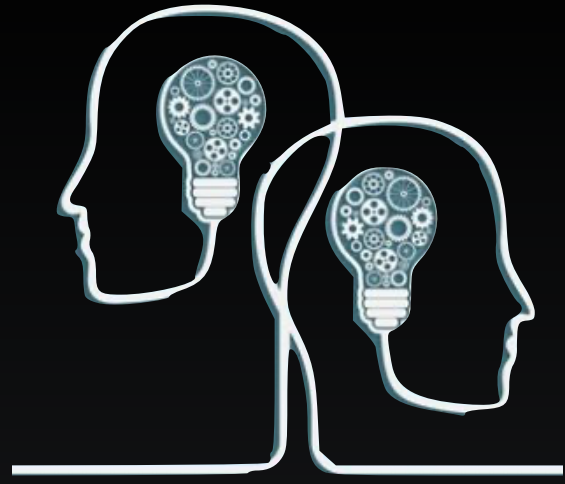
Iteration



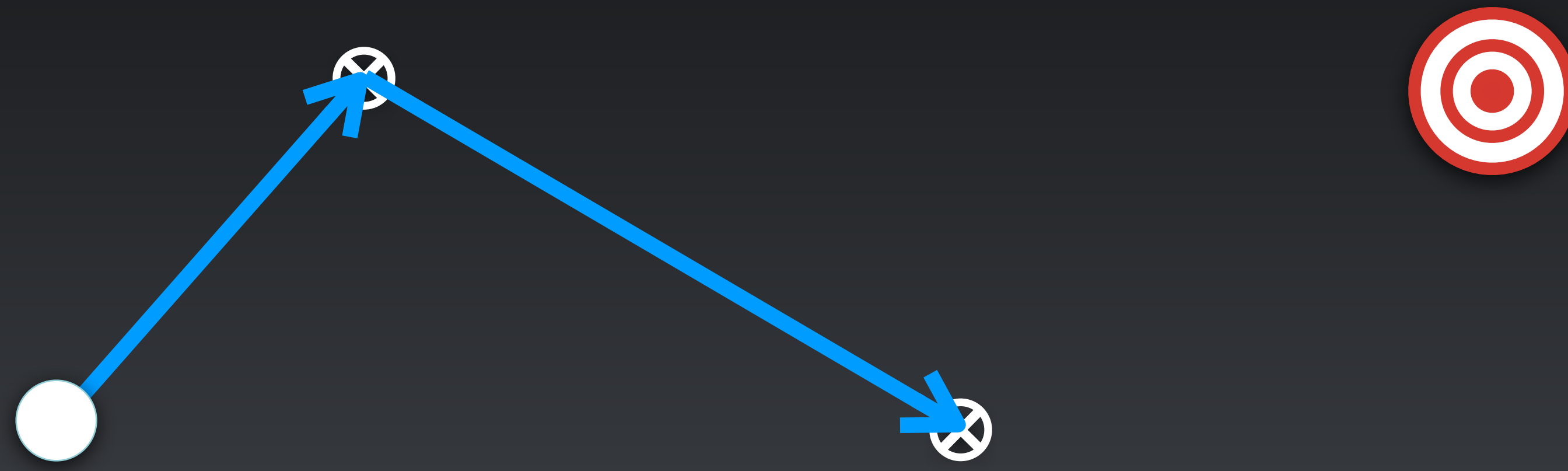
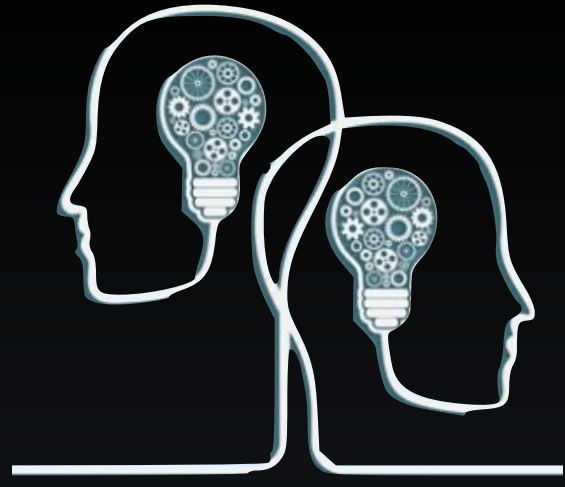
Iteration



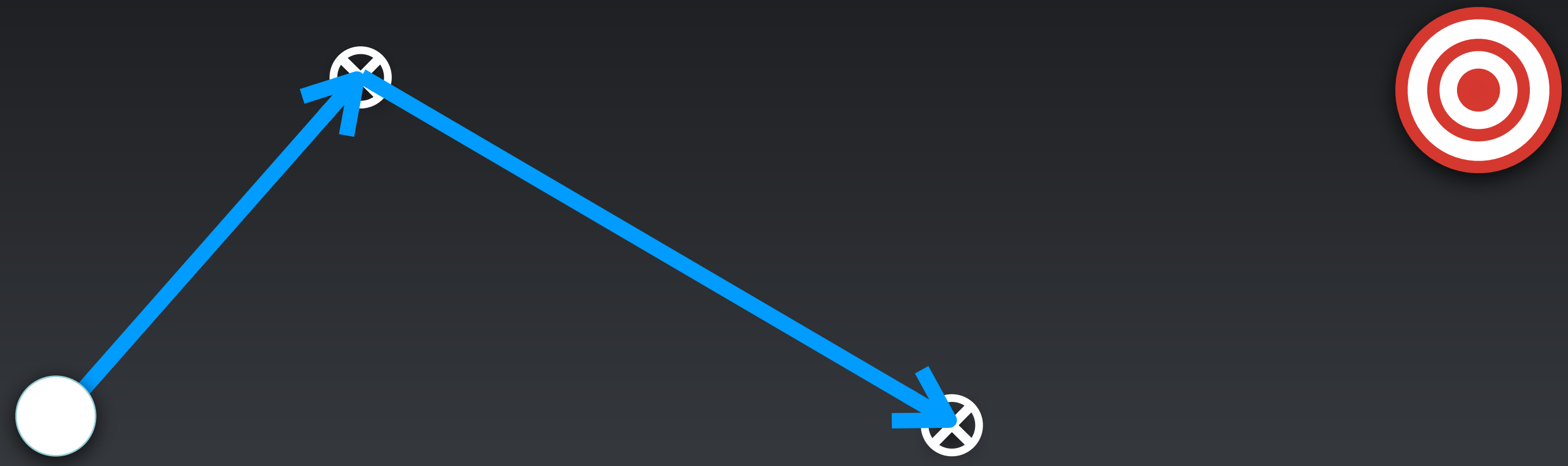
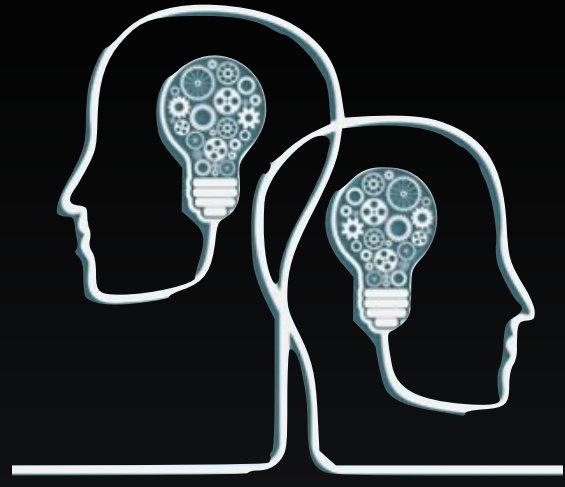
Iteration



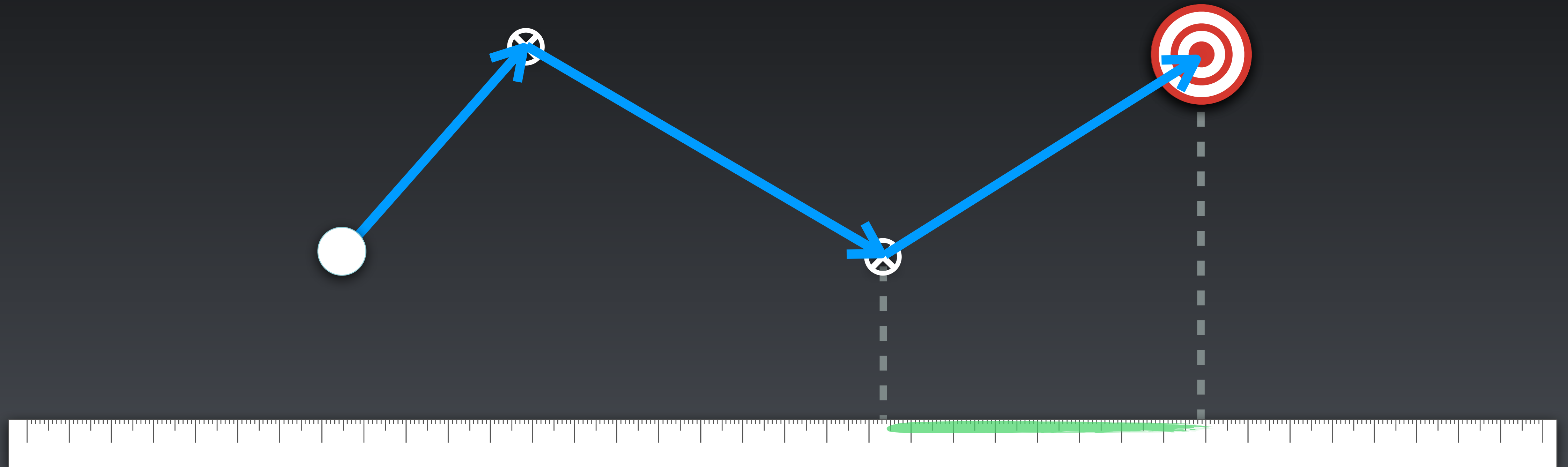
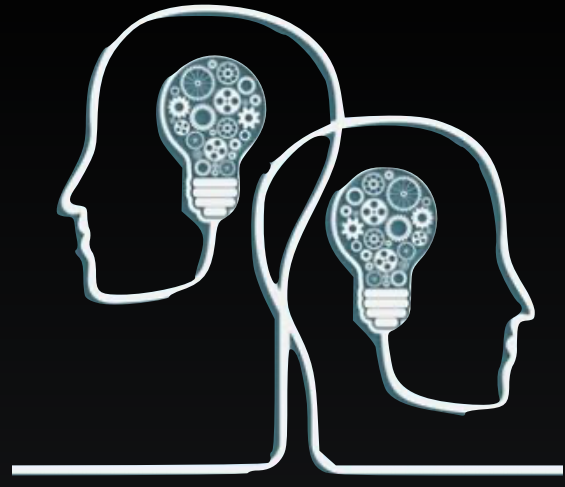
Iteration



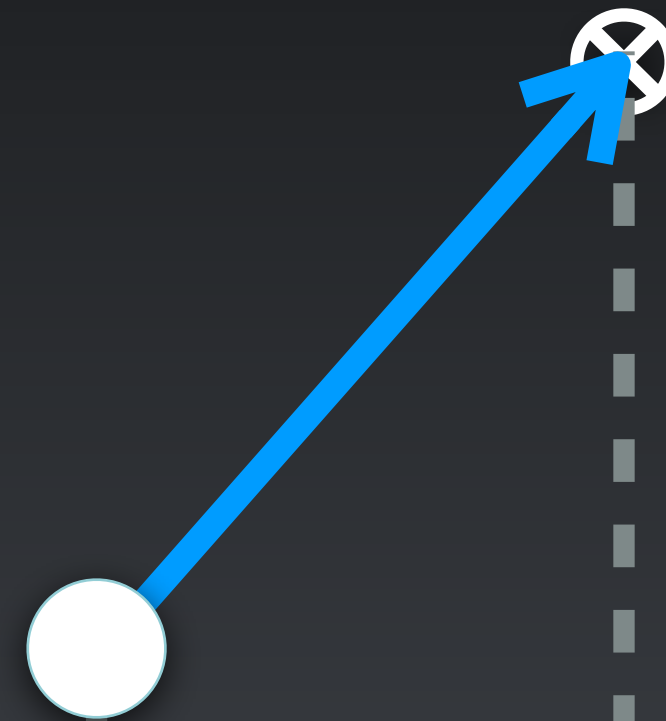
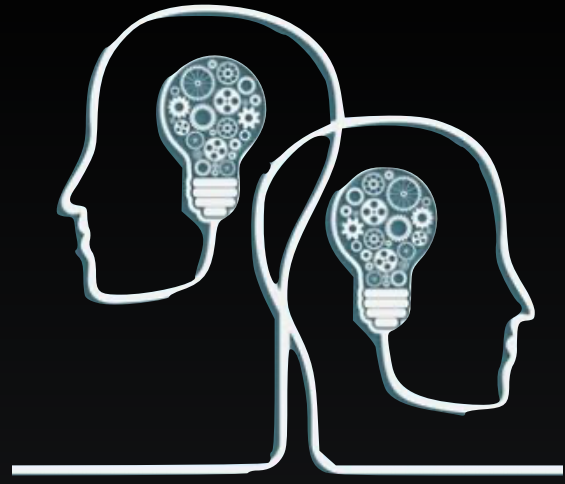
Iteration



Iteration

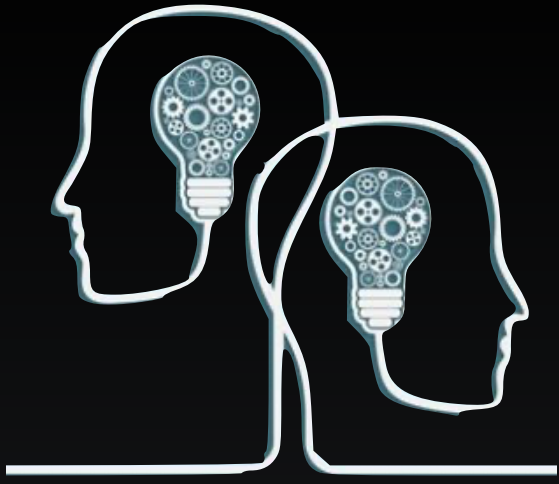


Feedback

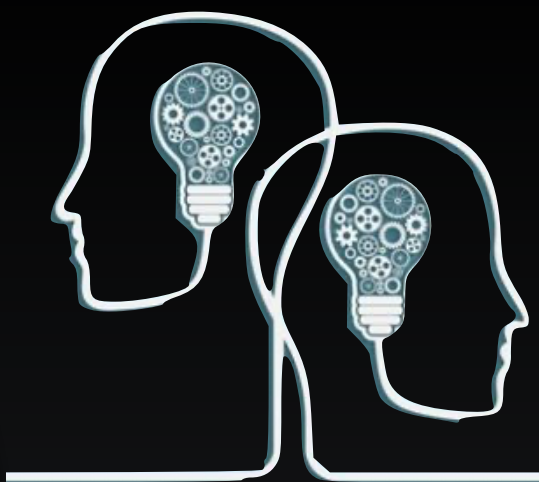


Feedback!

Feedback



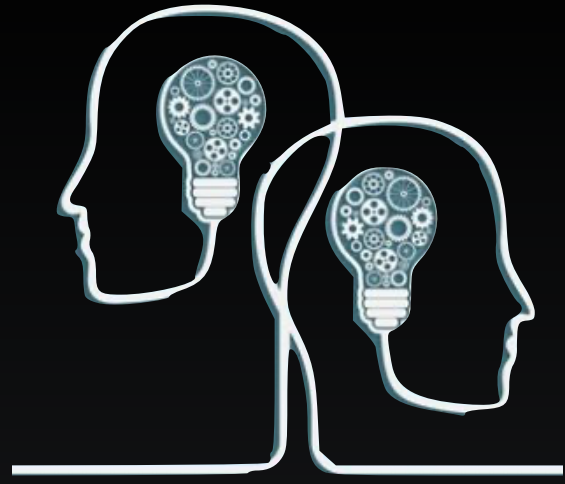
Feedback



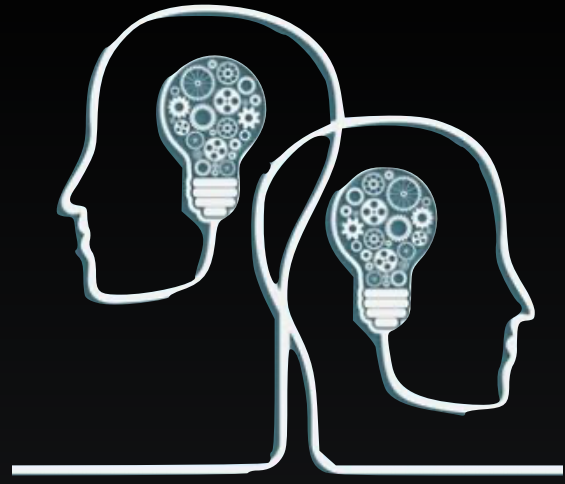
How Do You Balance a
Broom?



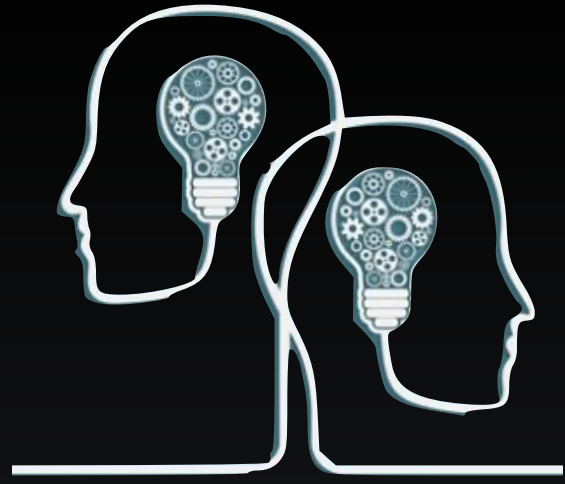
Feedback



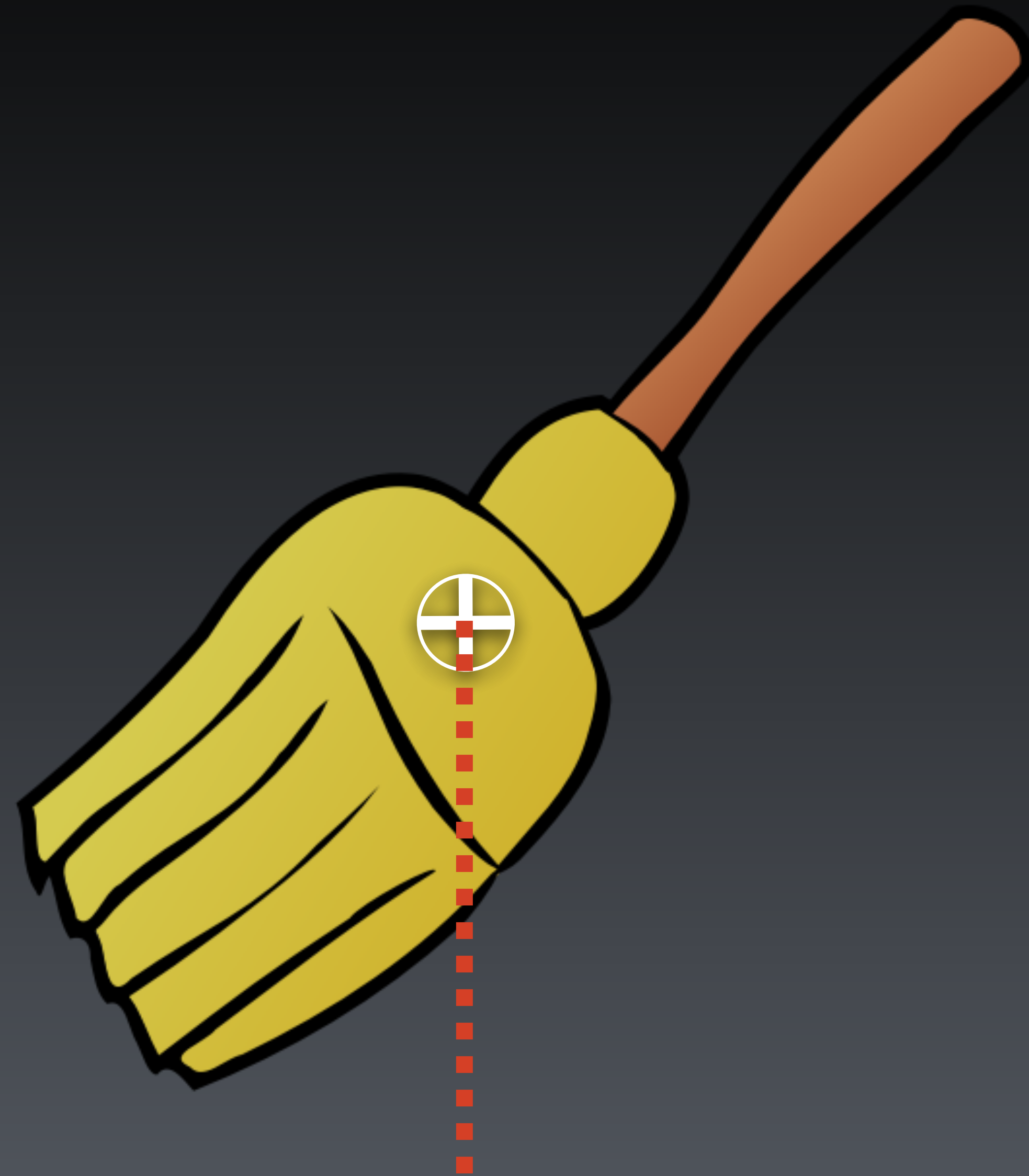
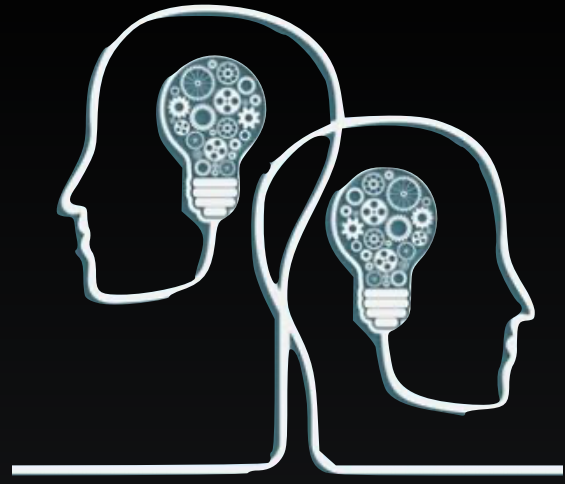
Feedback



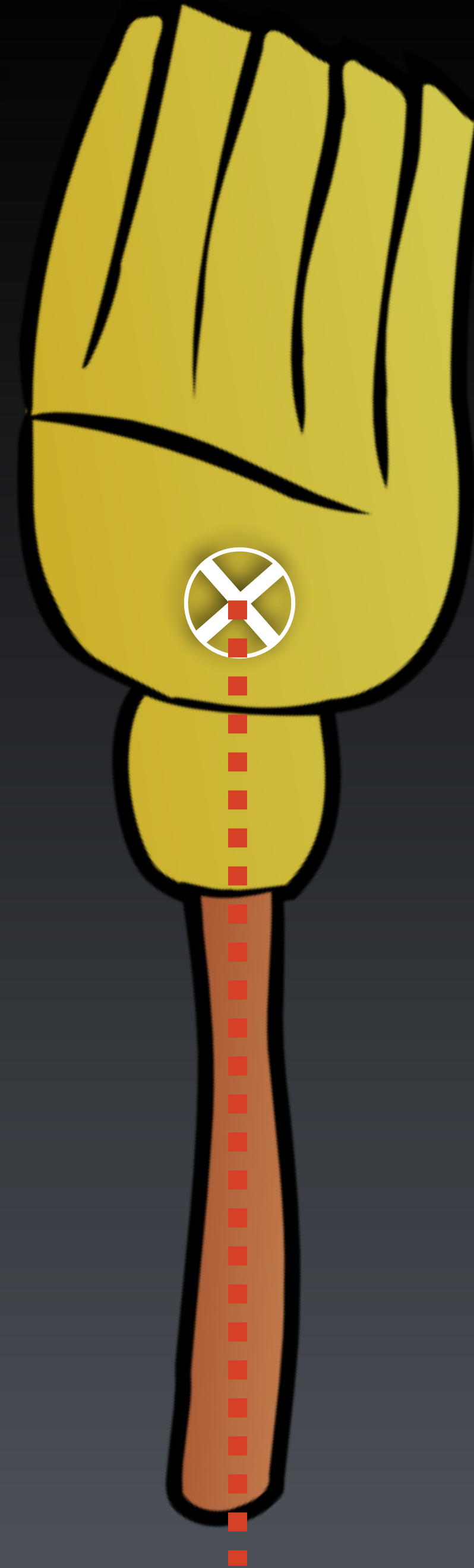
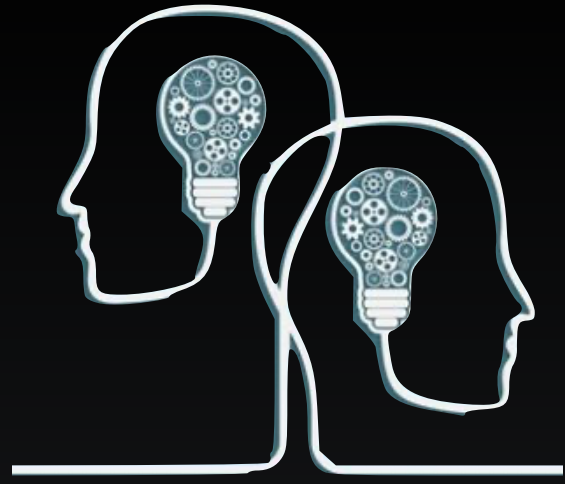
Feedback



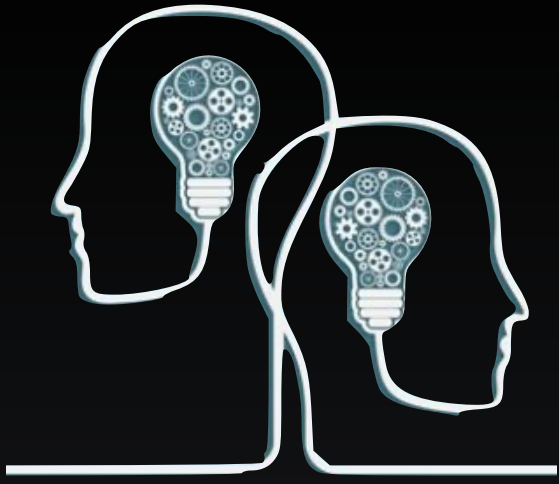
Feedback



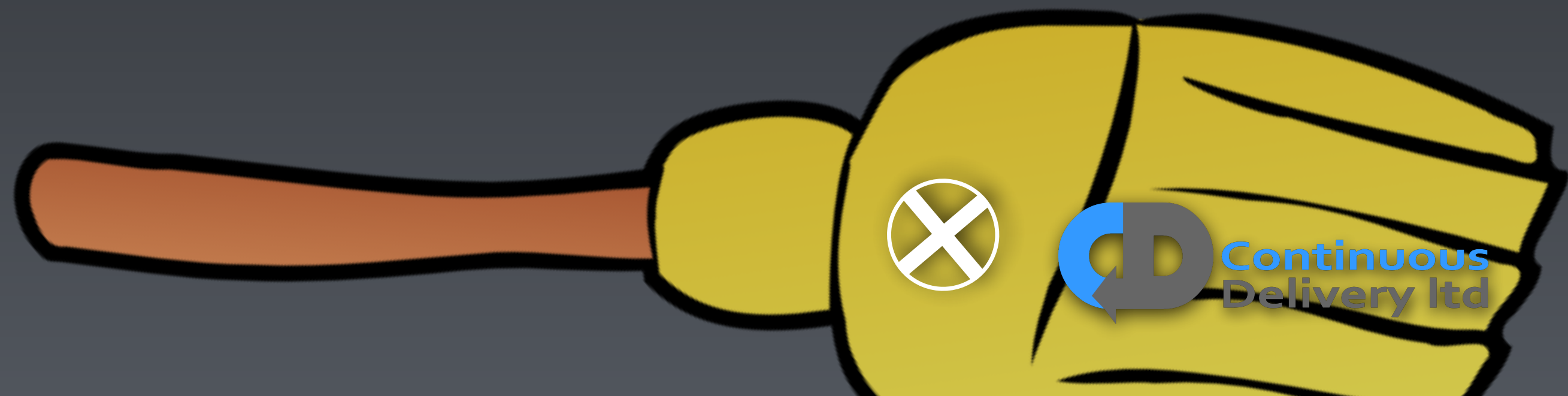
Feedback



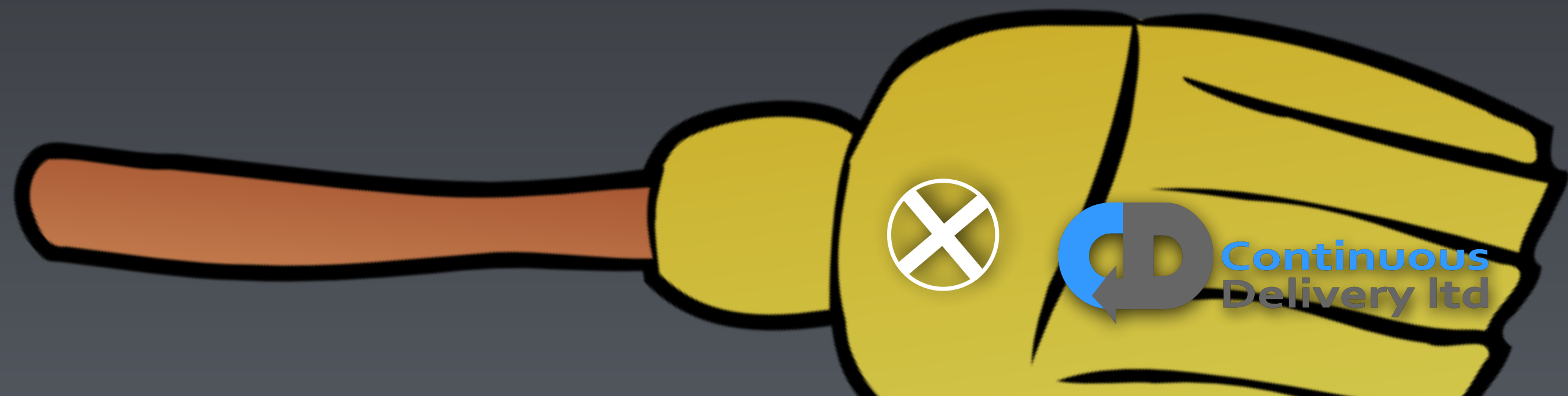
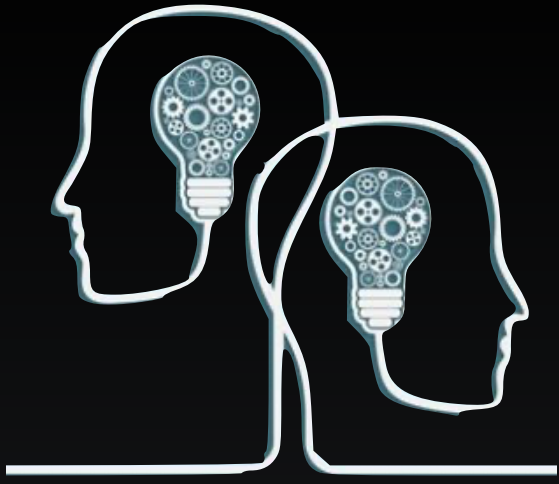
Feedback



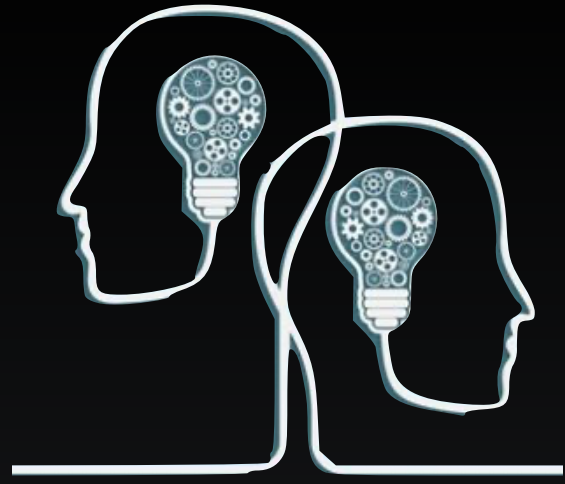
**There's Only One
Solution to This!**



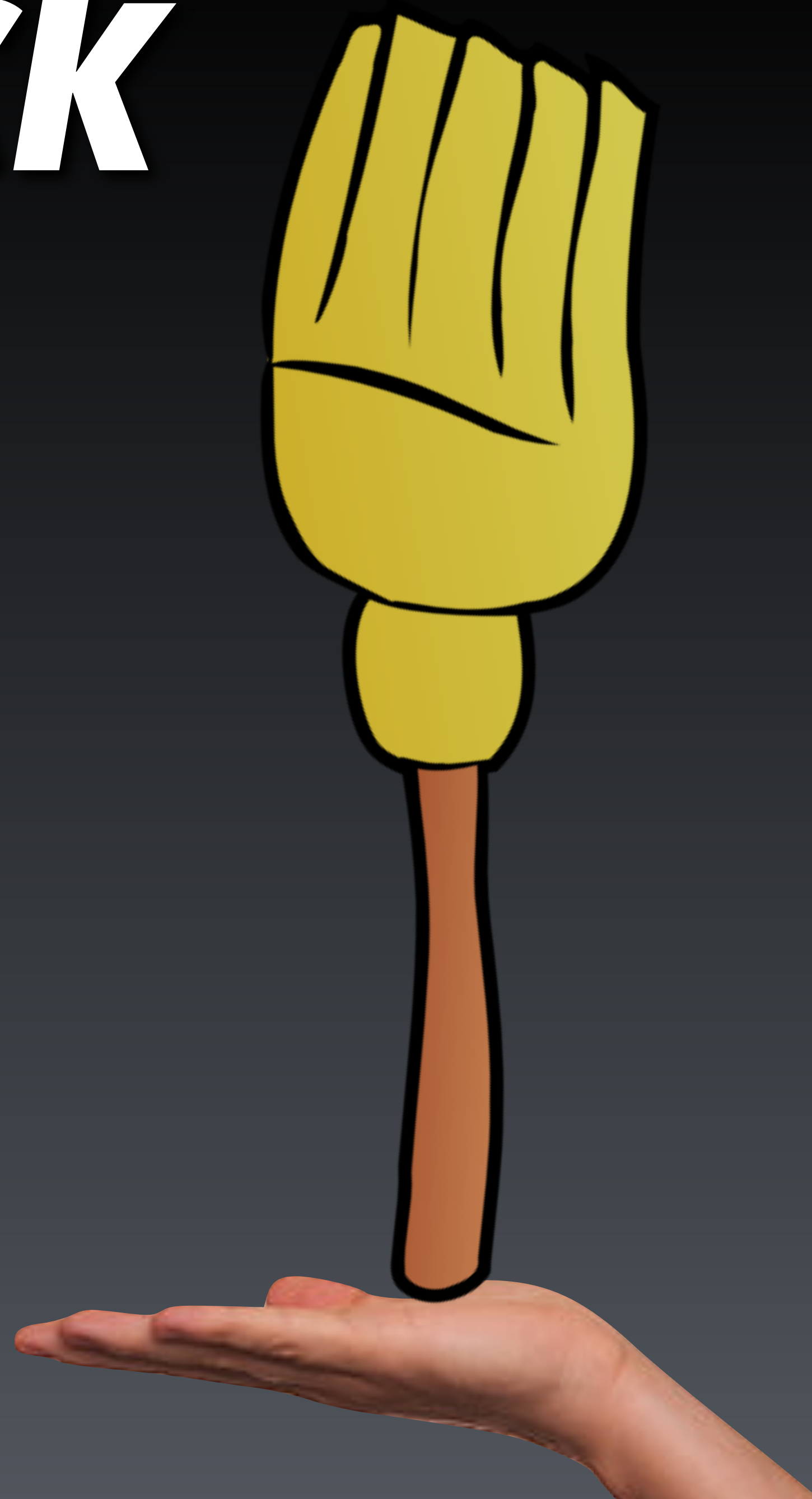
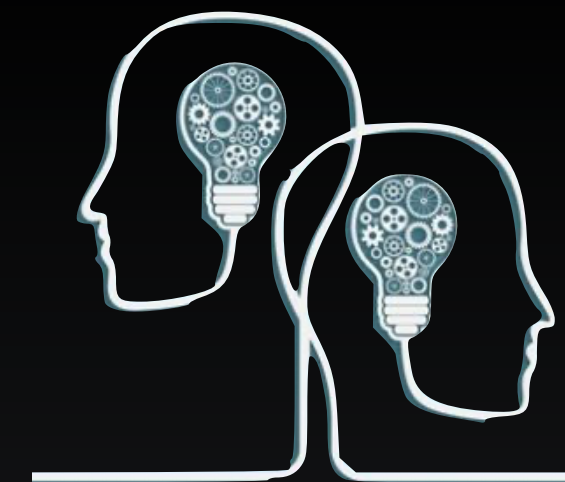
Feedback



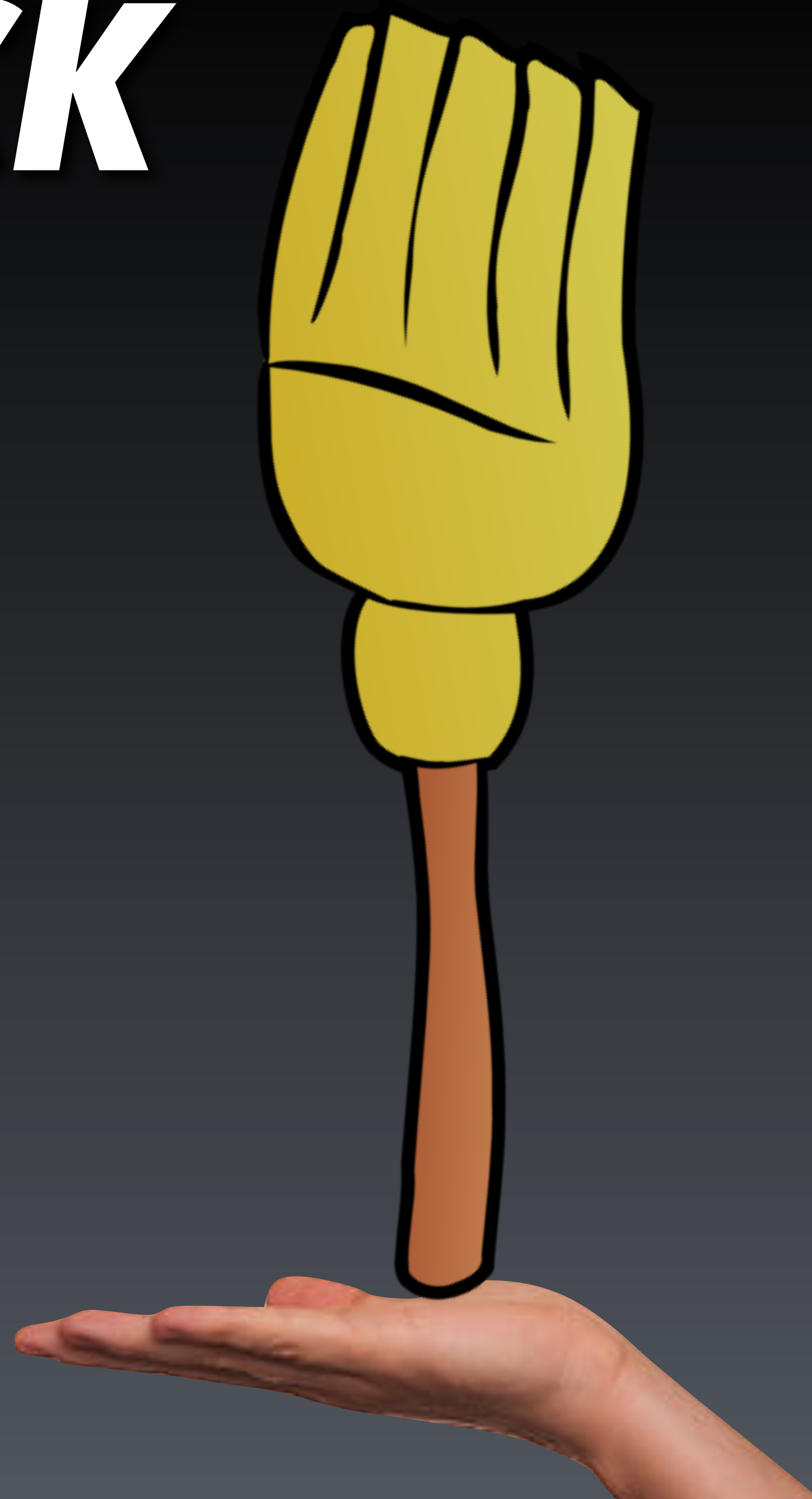
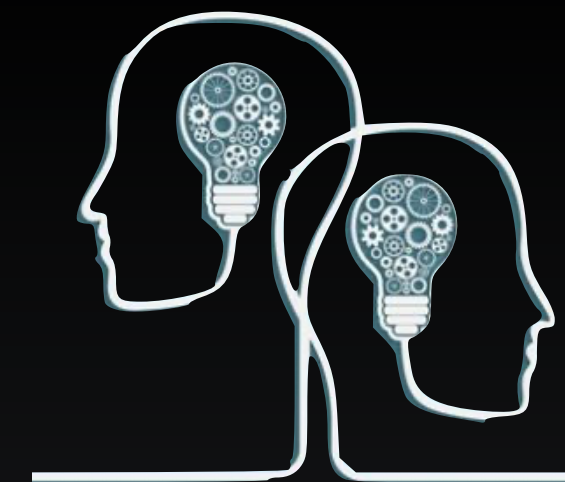
Feedback



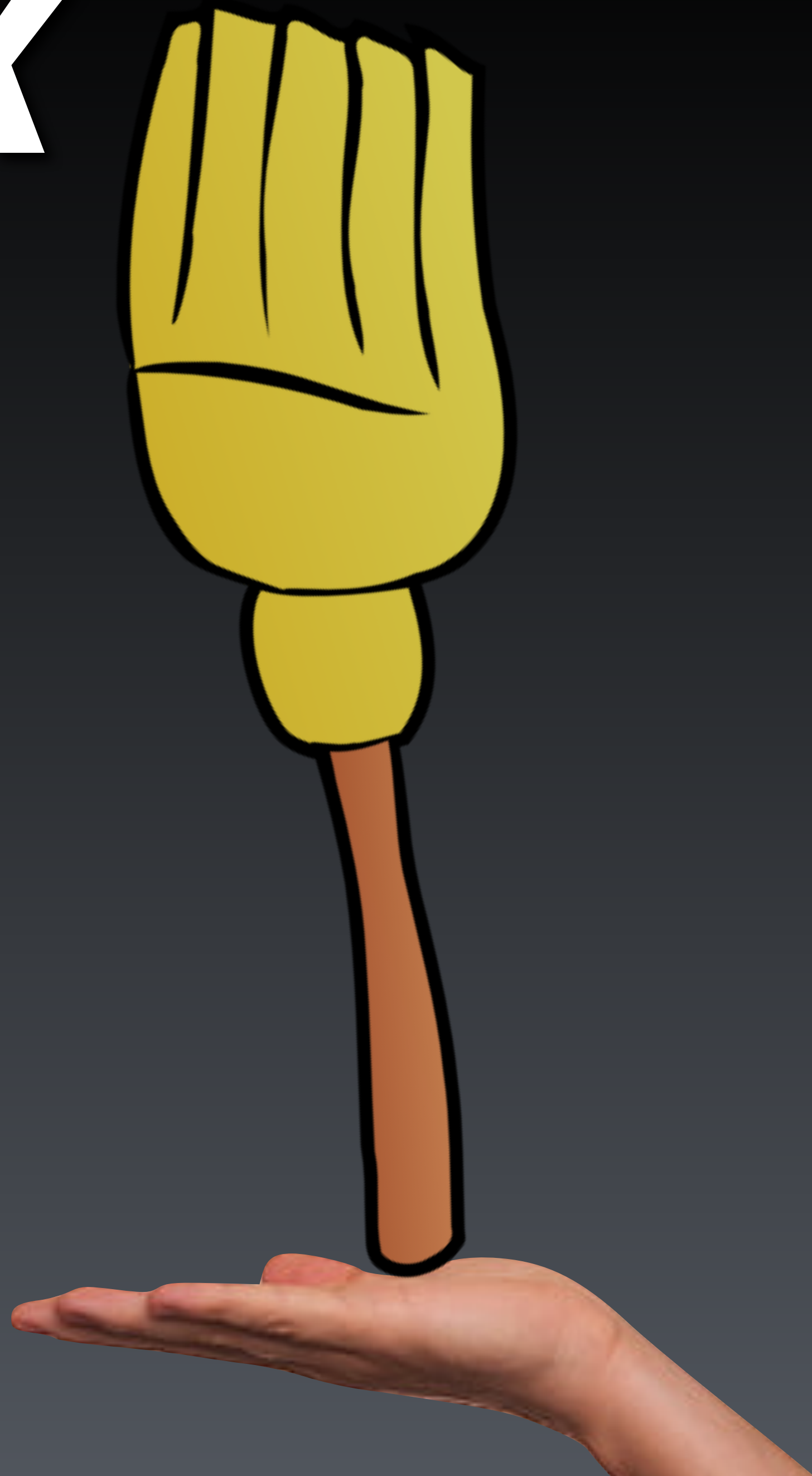
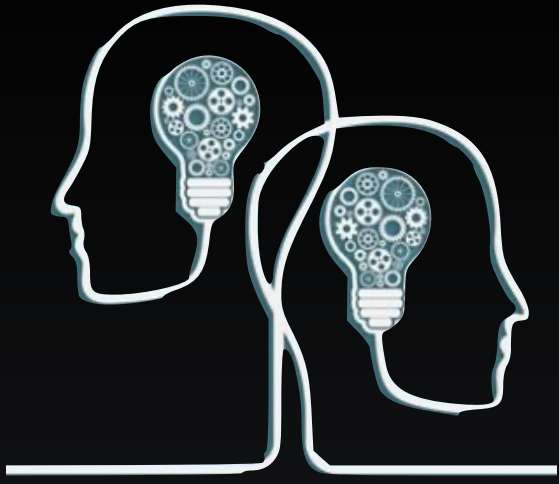
Feedback



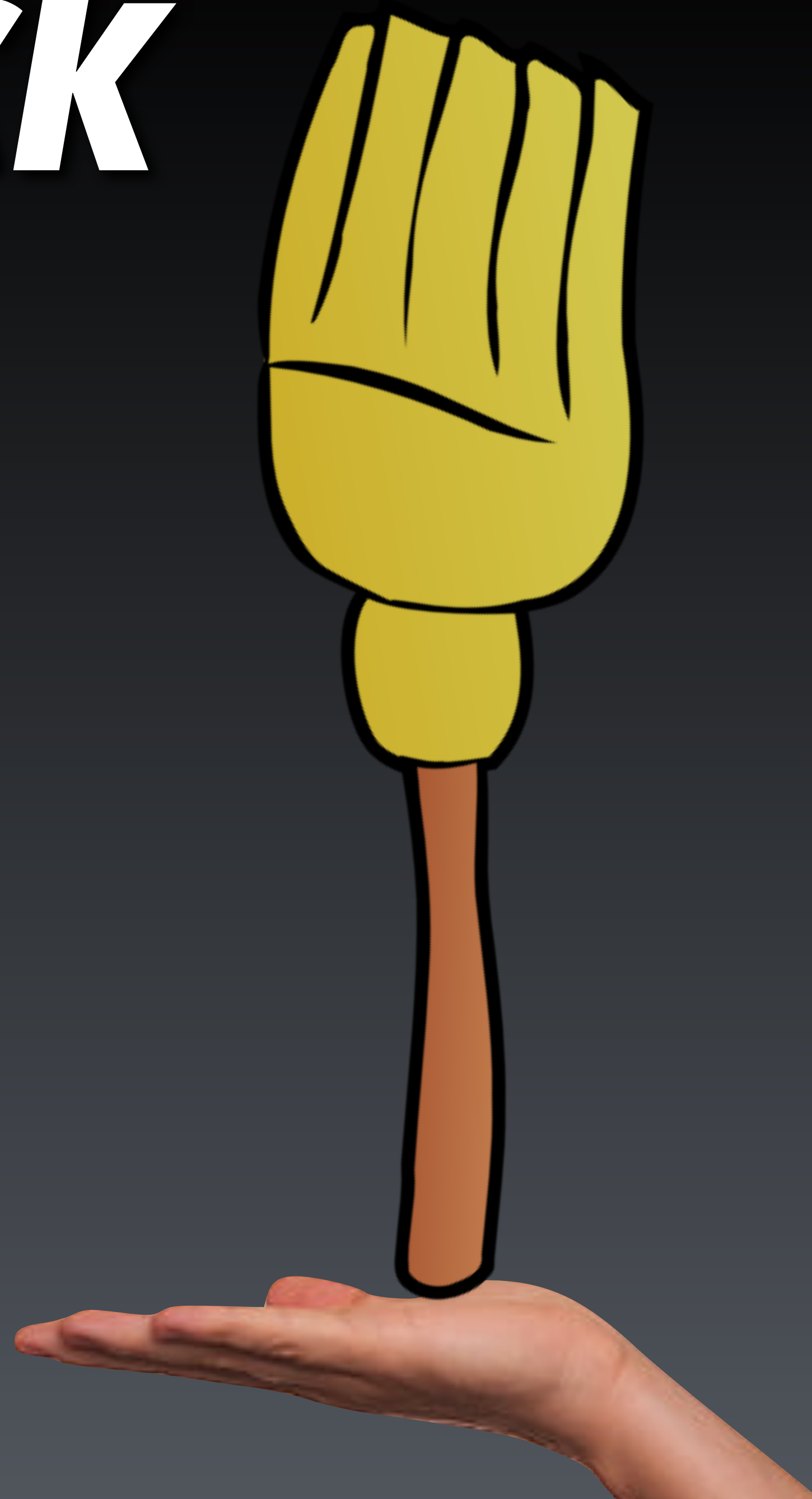
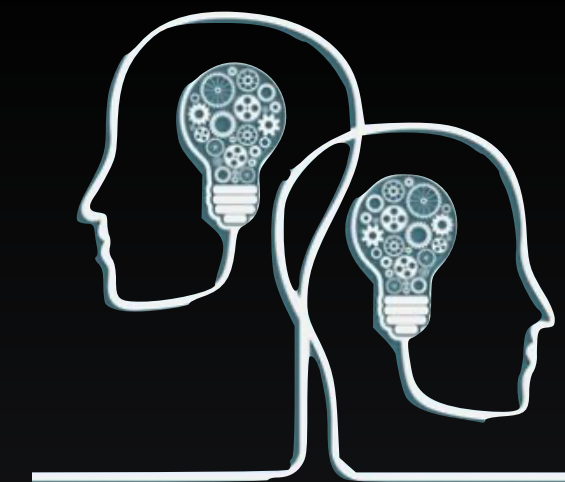
Feedback



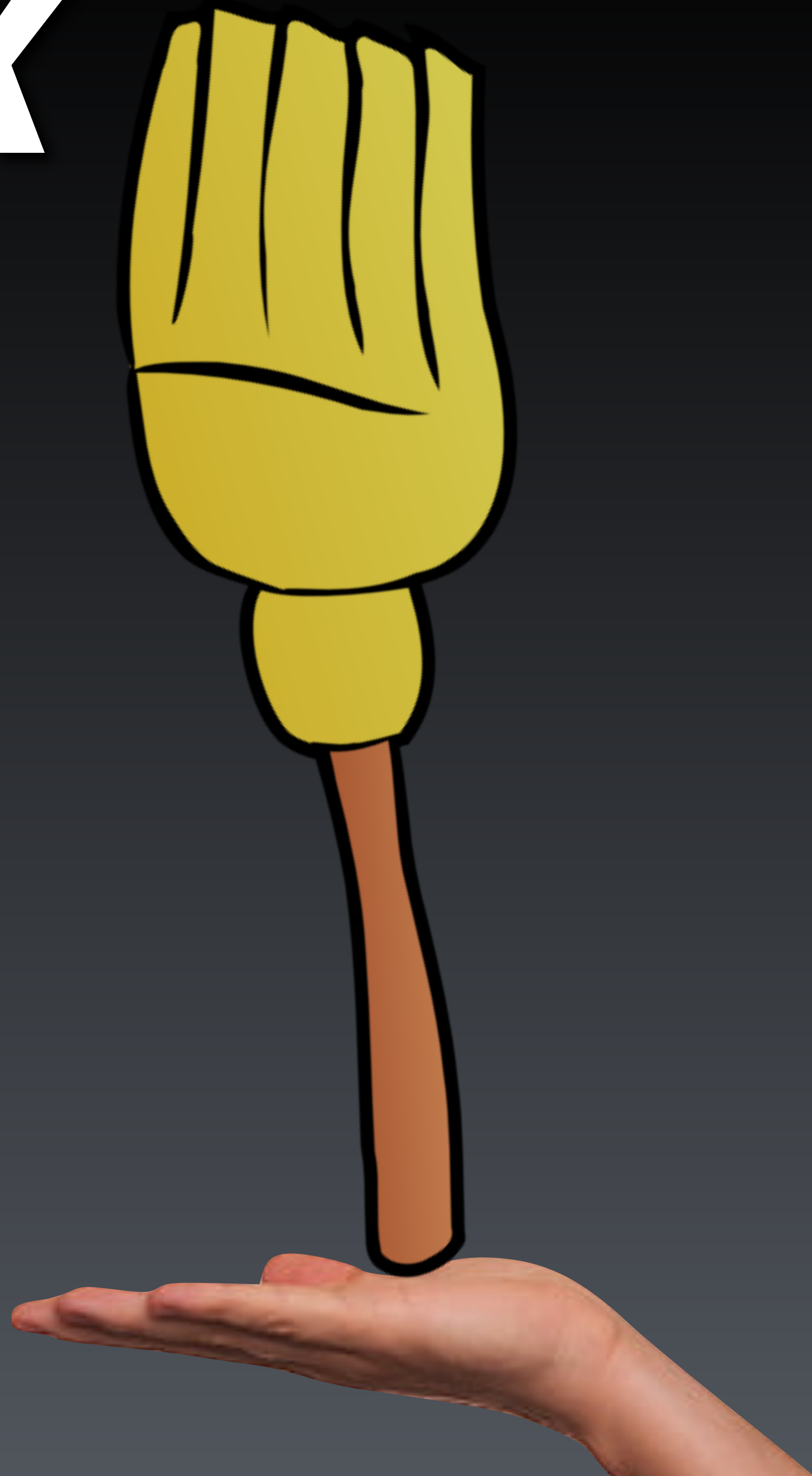
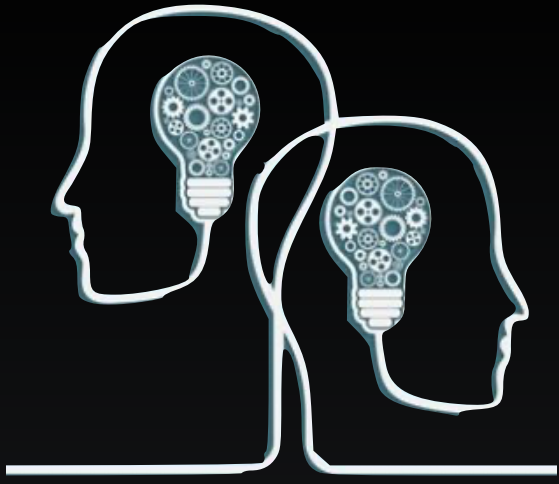
Feedback



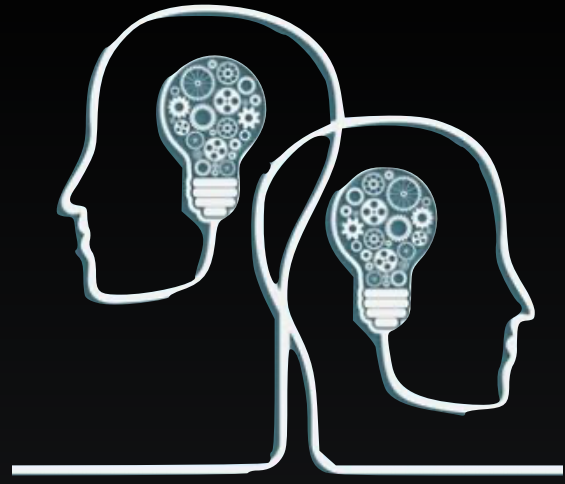
Feedback



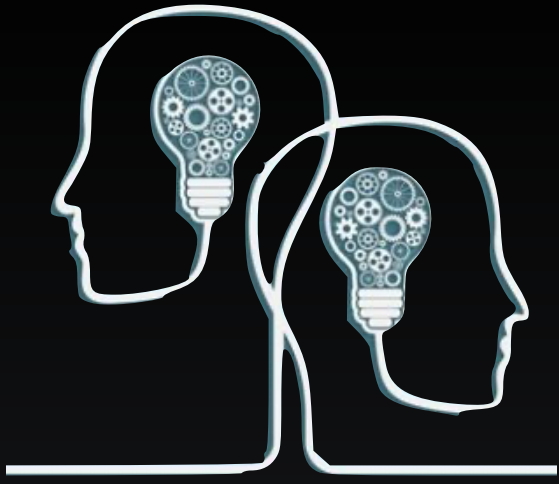
Feedback



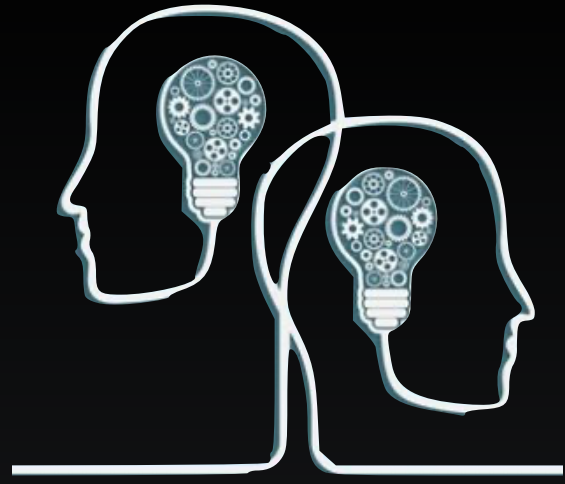
Feedback



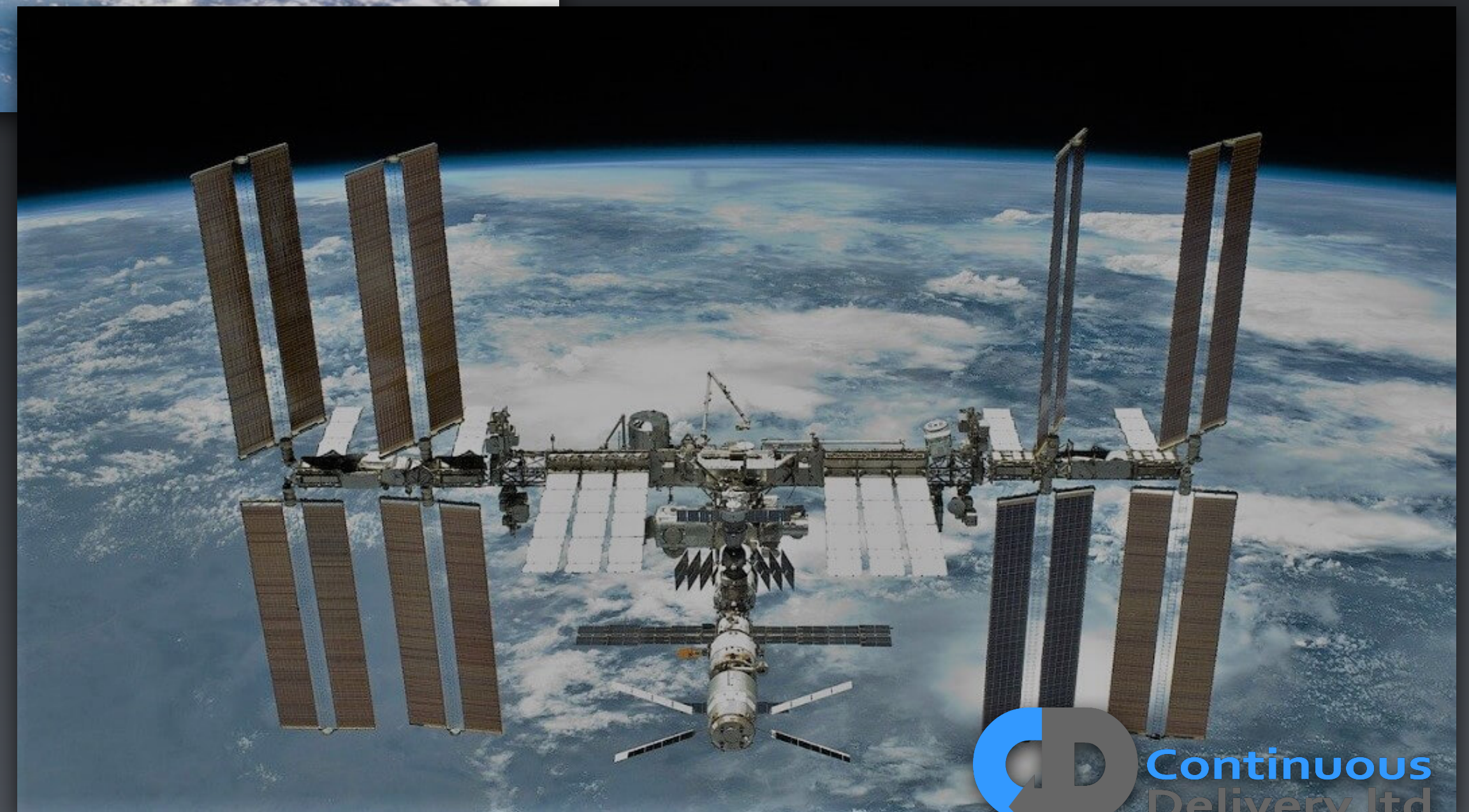
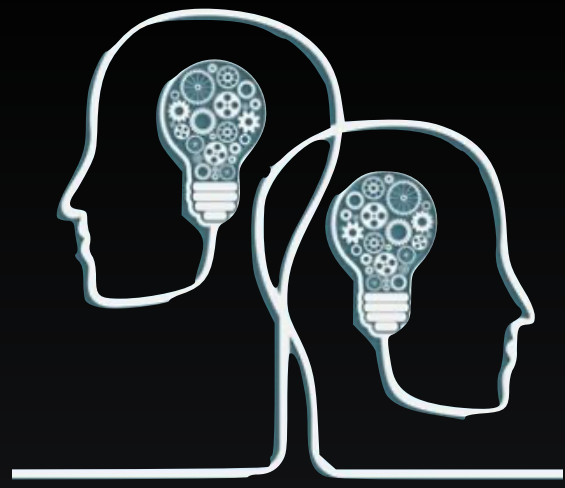
Incremental



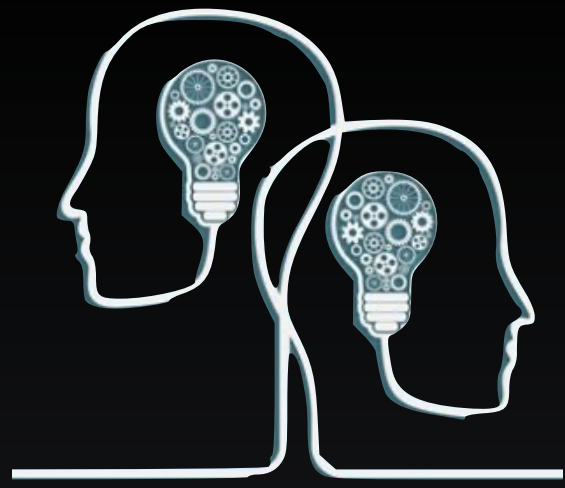
Incremental



Incremental

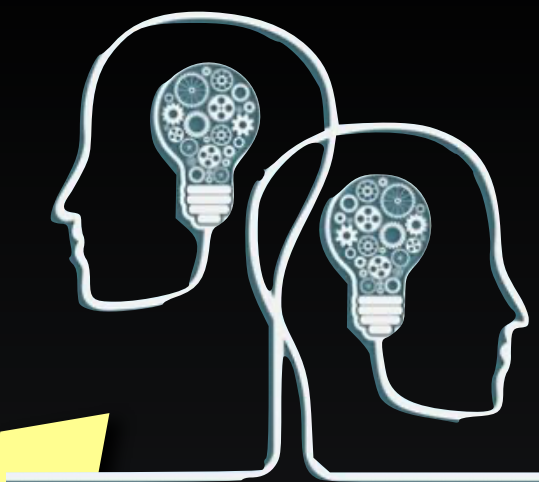


Experimental



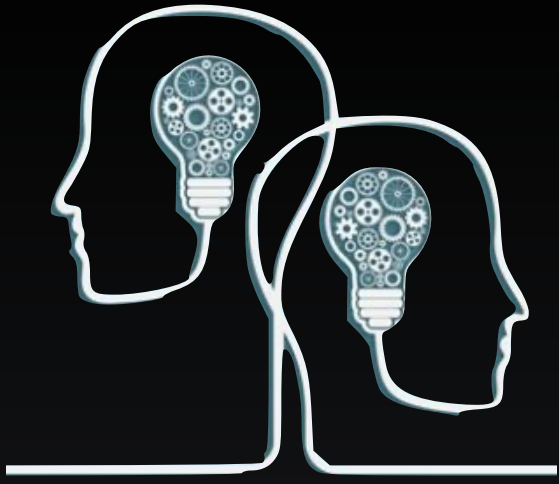
Scary!

Experimental

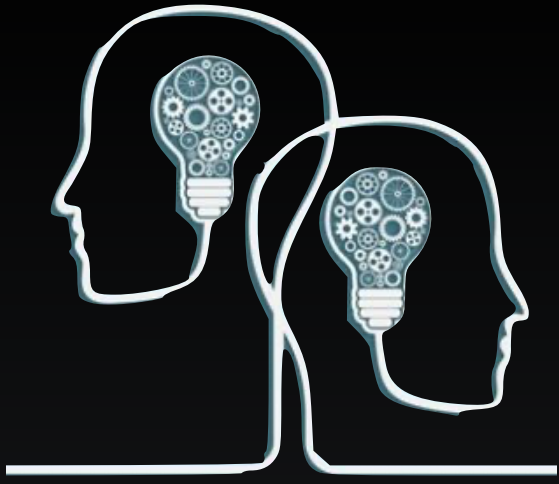


**How Else Can We
Make Progress?**

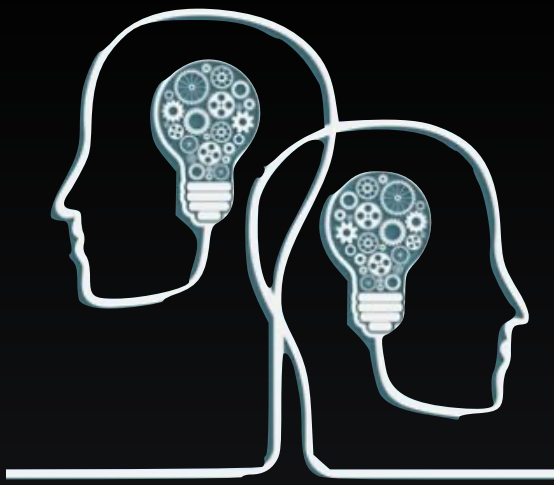
Experimental



Experimental

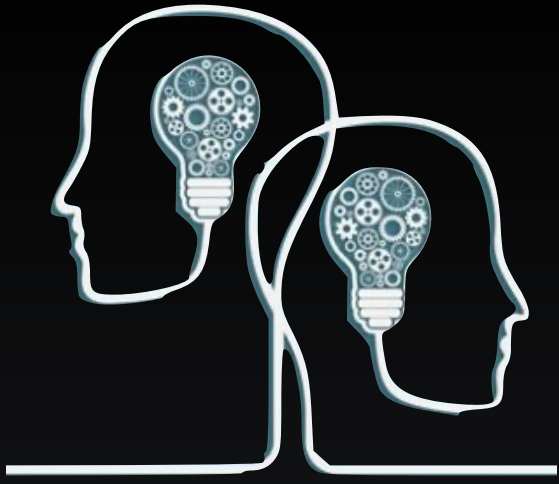


Experimental

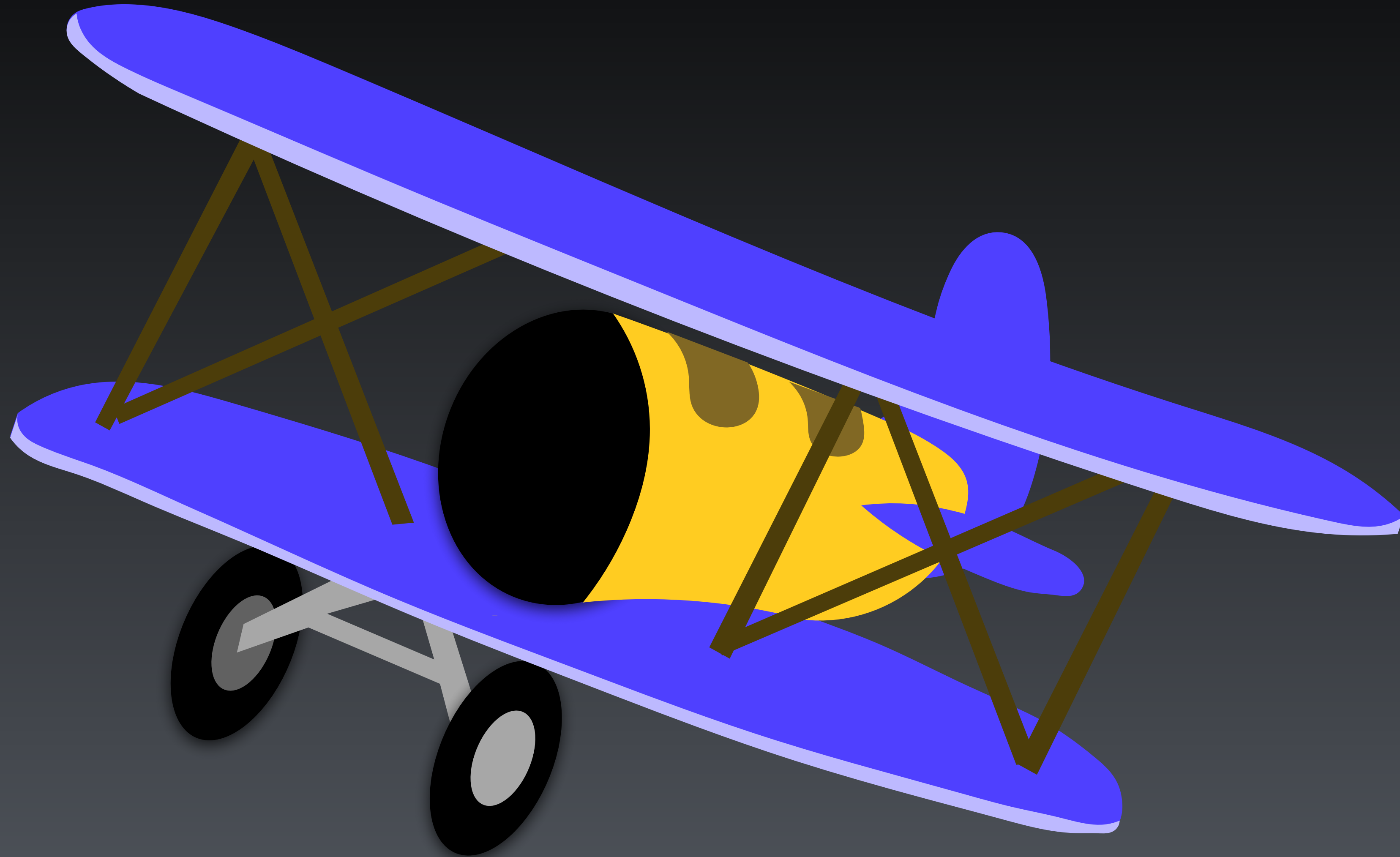
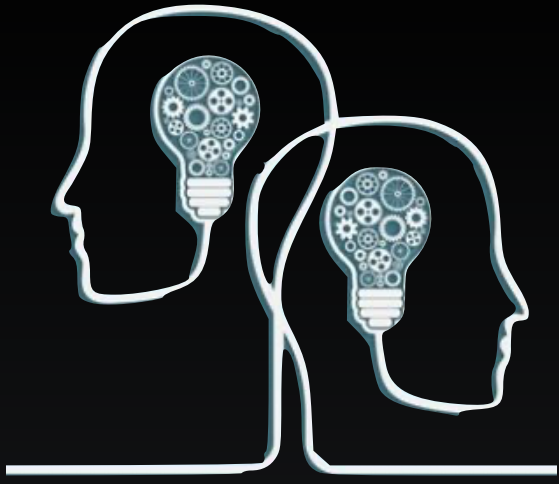


**Controlling the
Variables!**

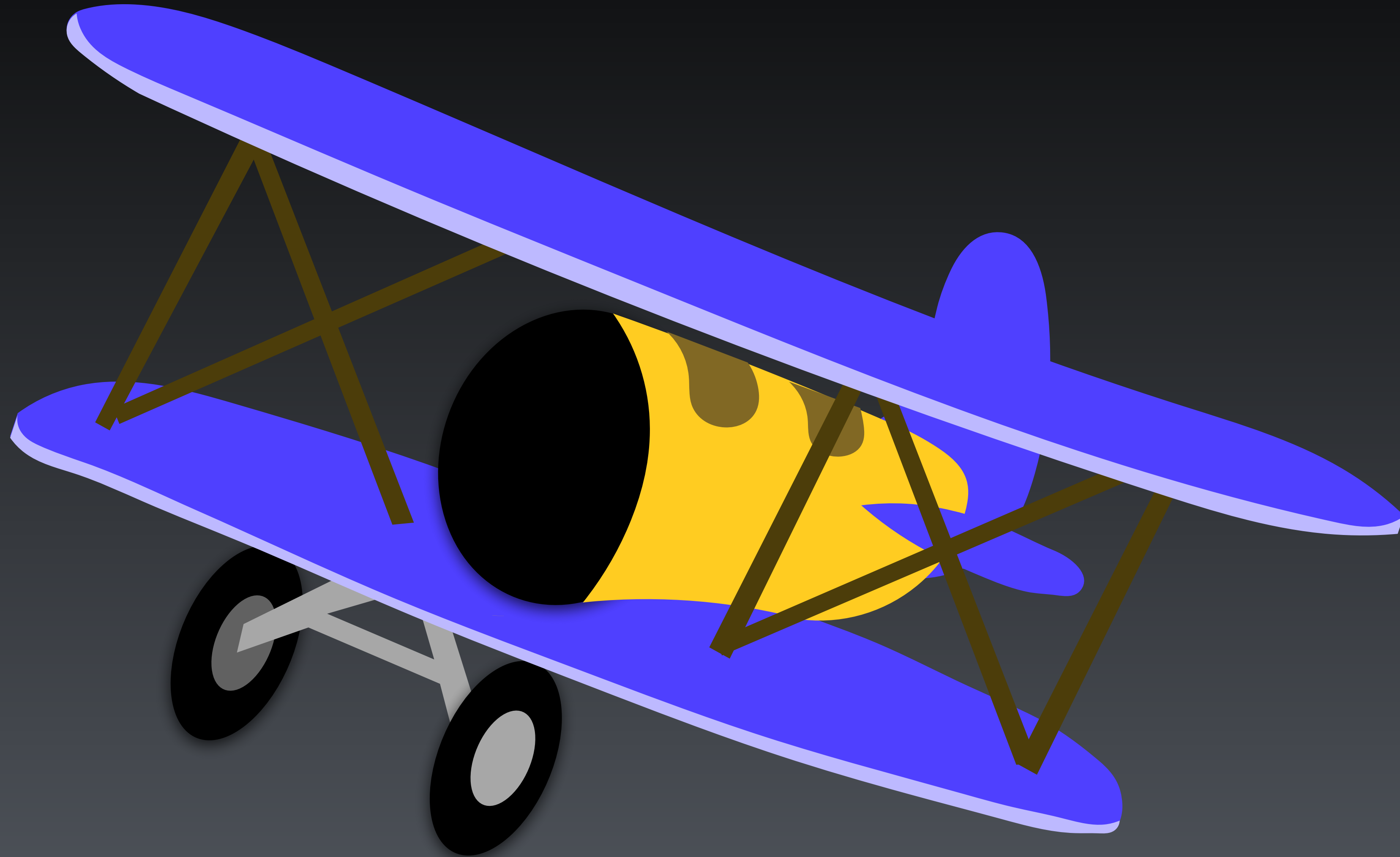
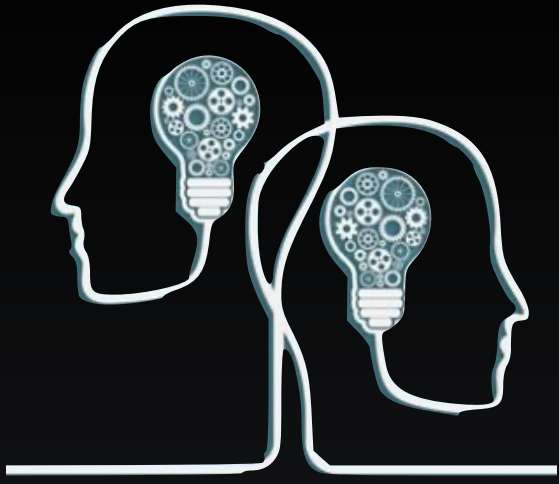
Experimental



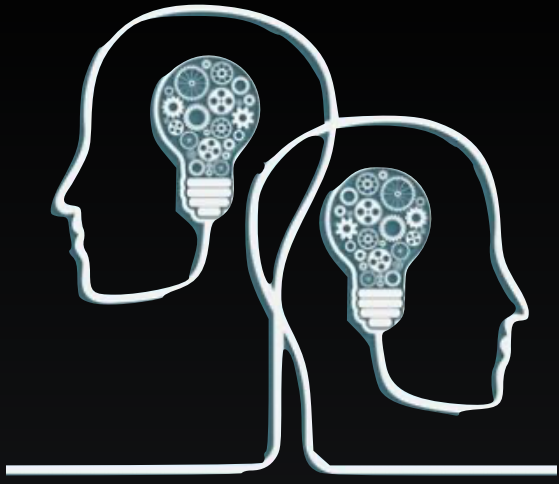
Experimental



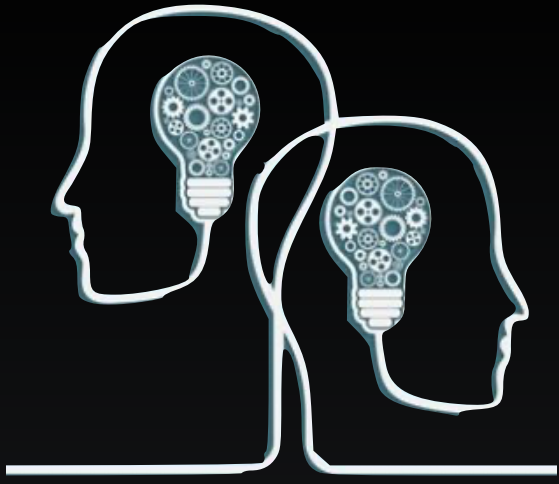
Experimental



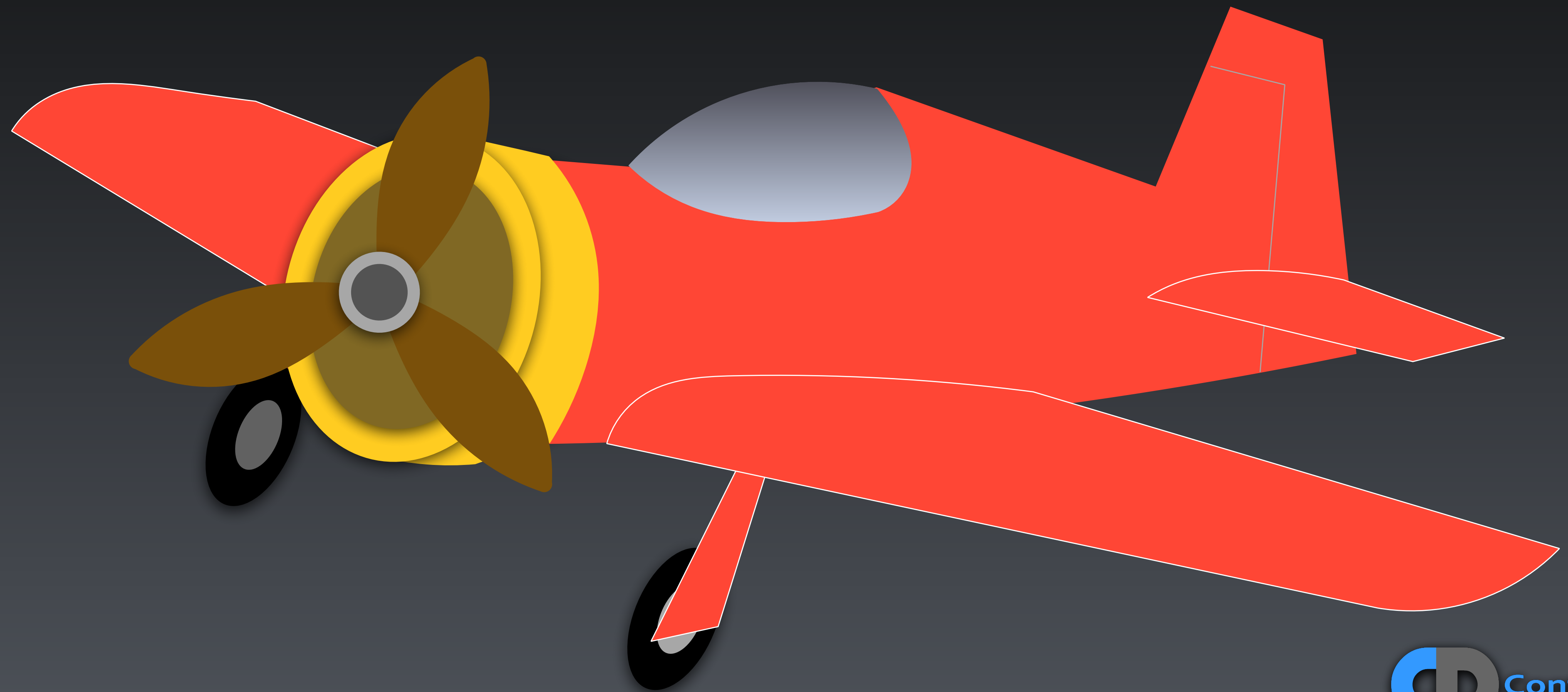
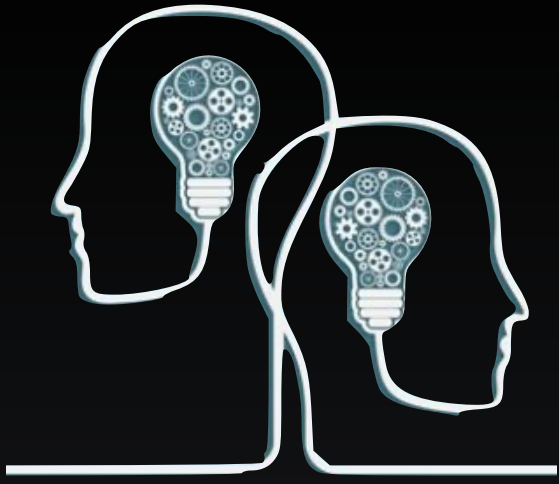
Experimental



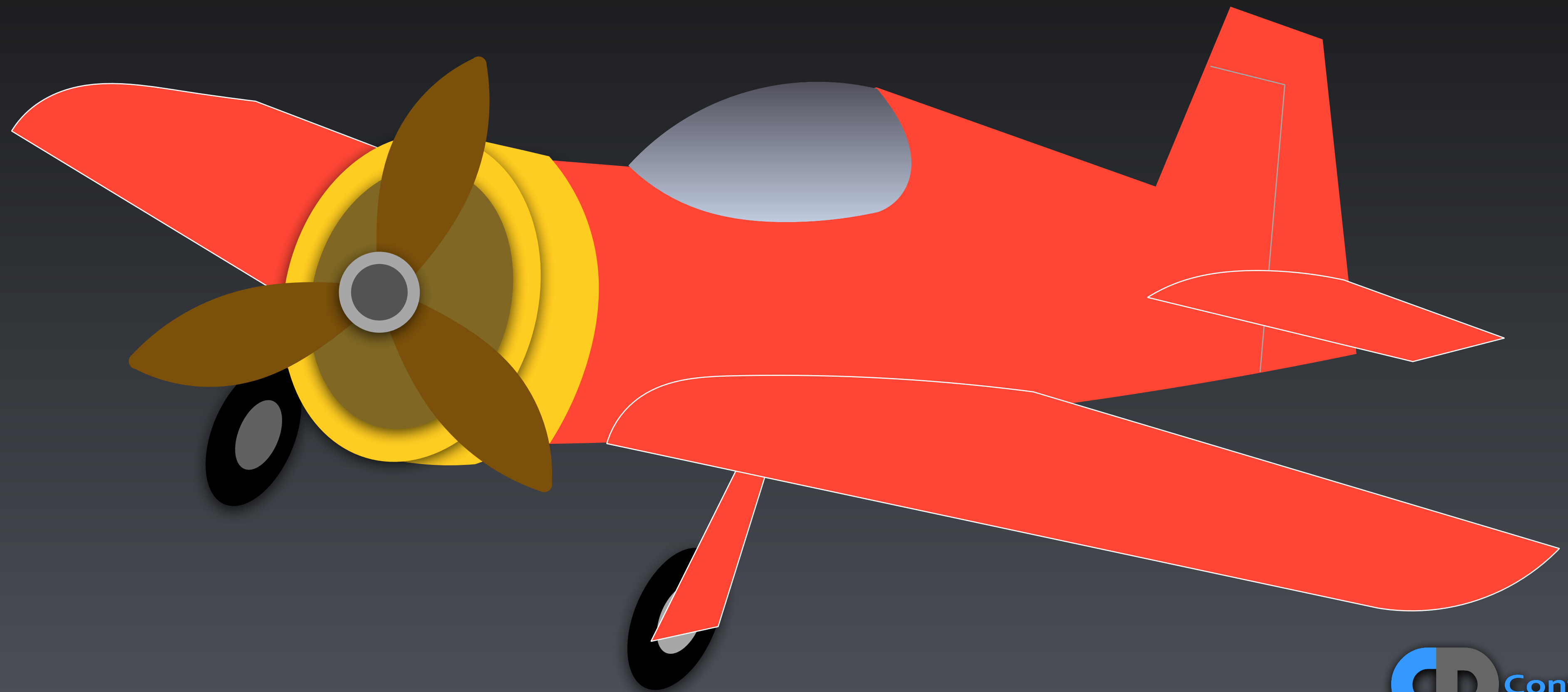
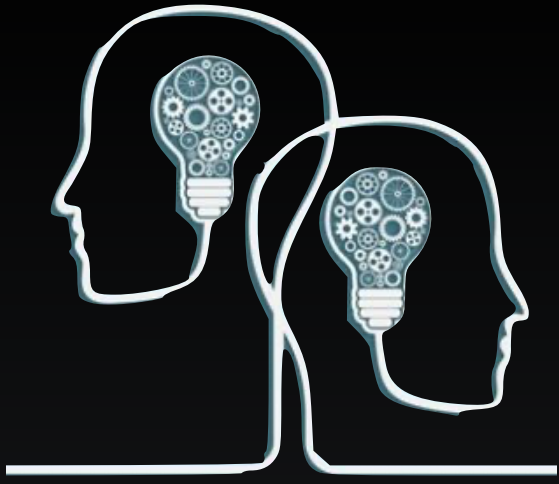
Experimental



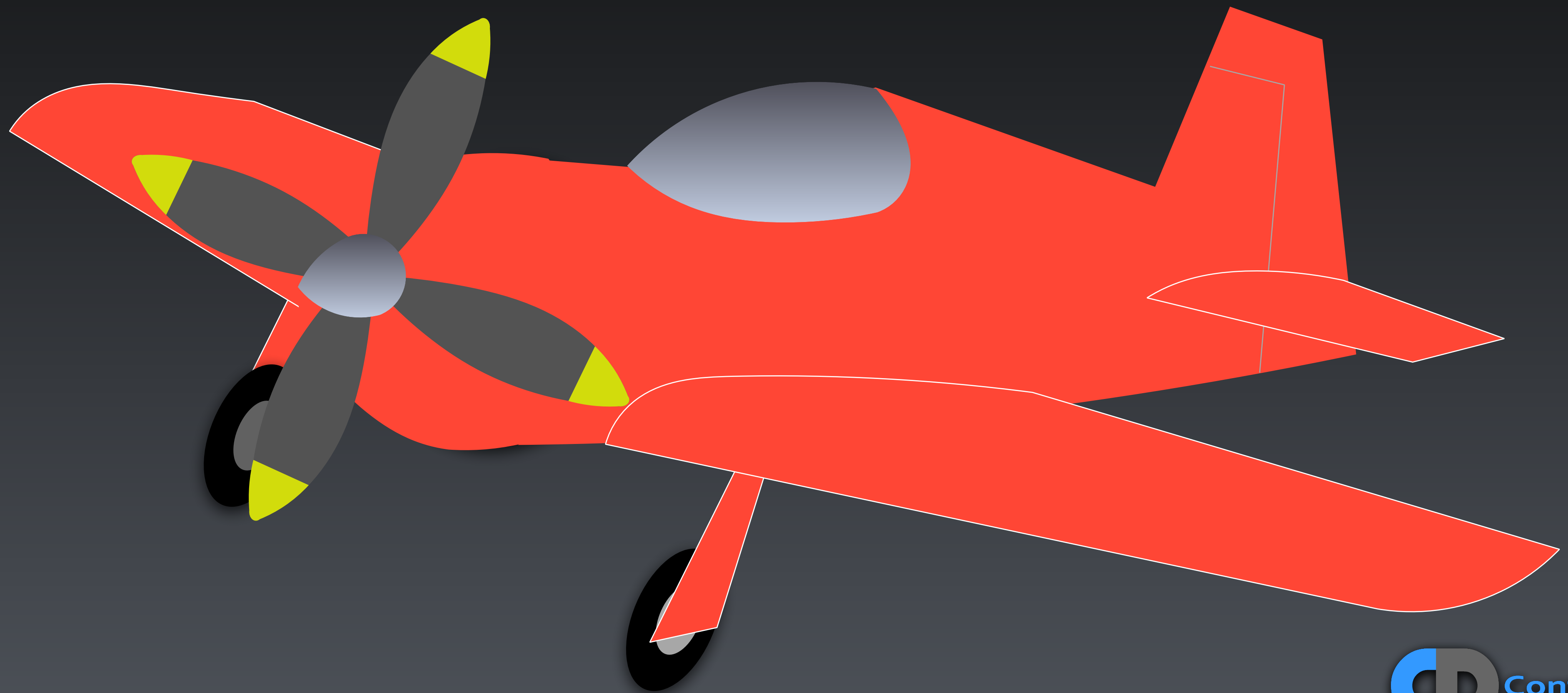
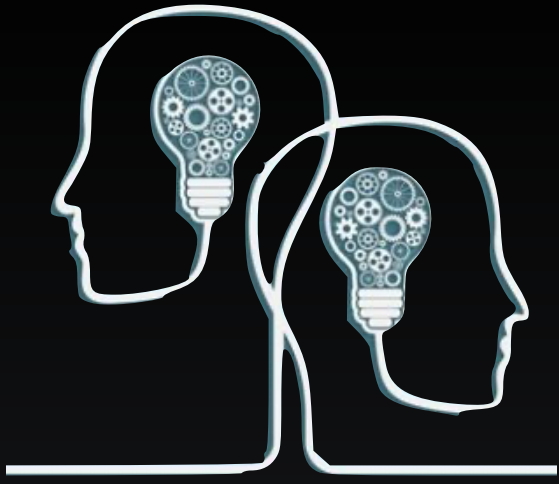
Experimental



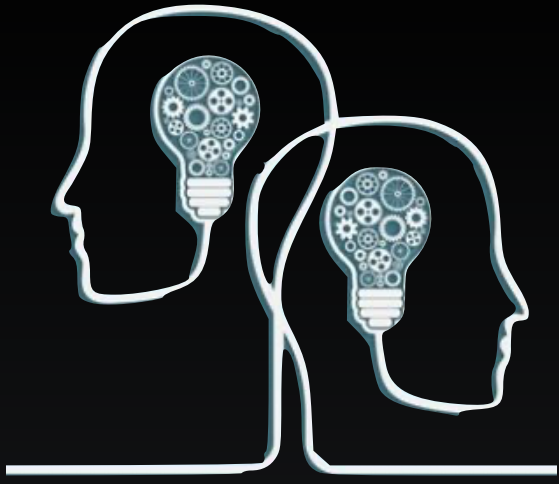
Experimental



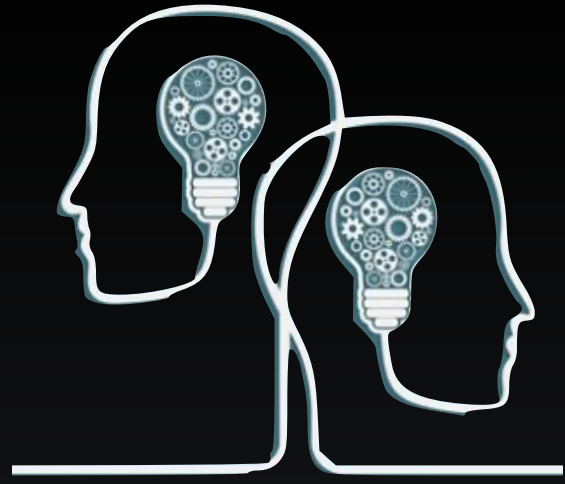
Experimental

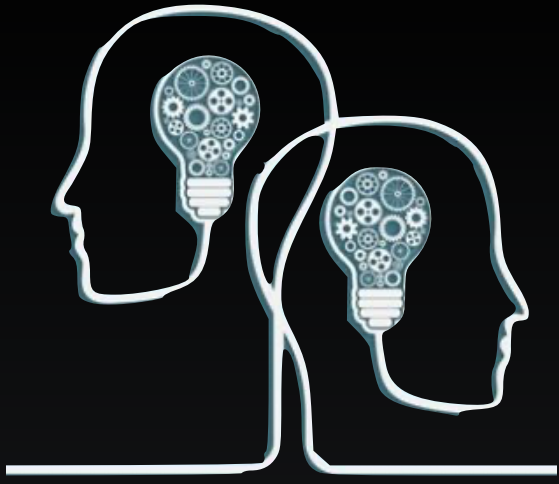


Empirical

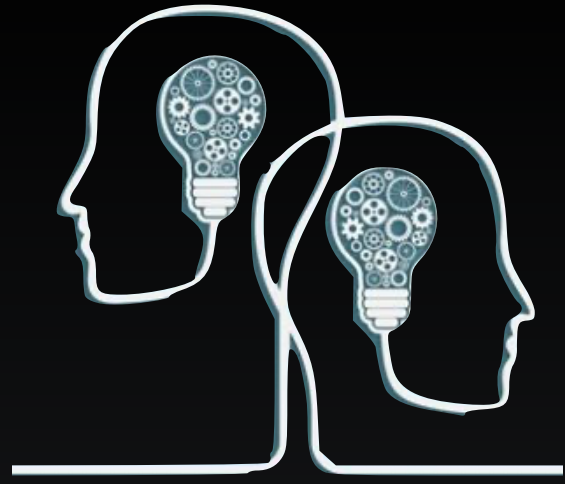


Empirical



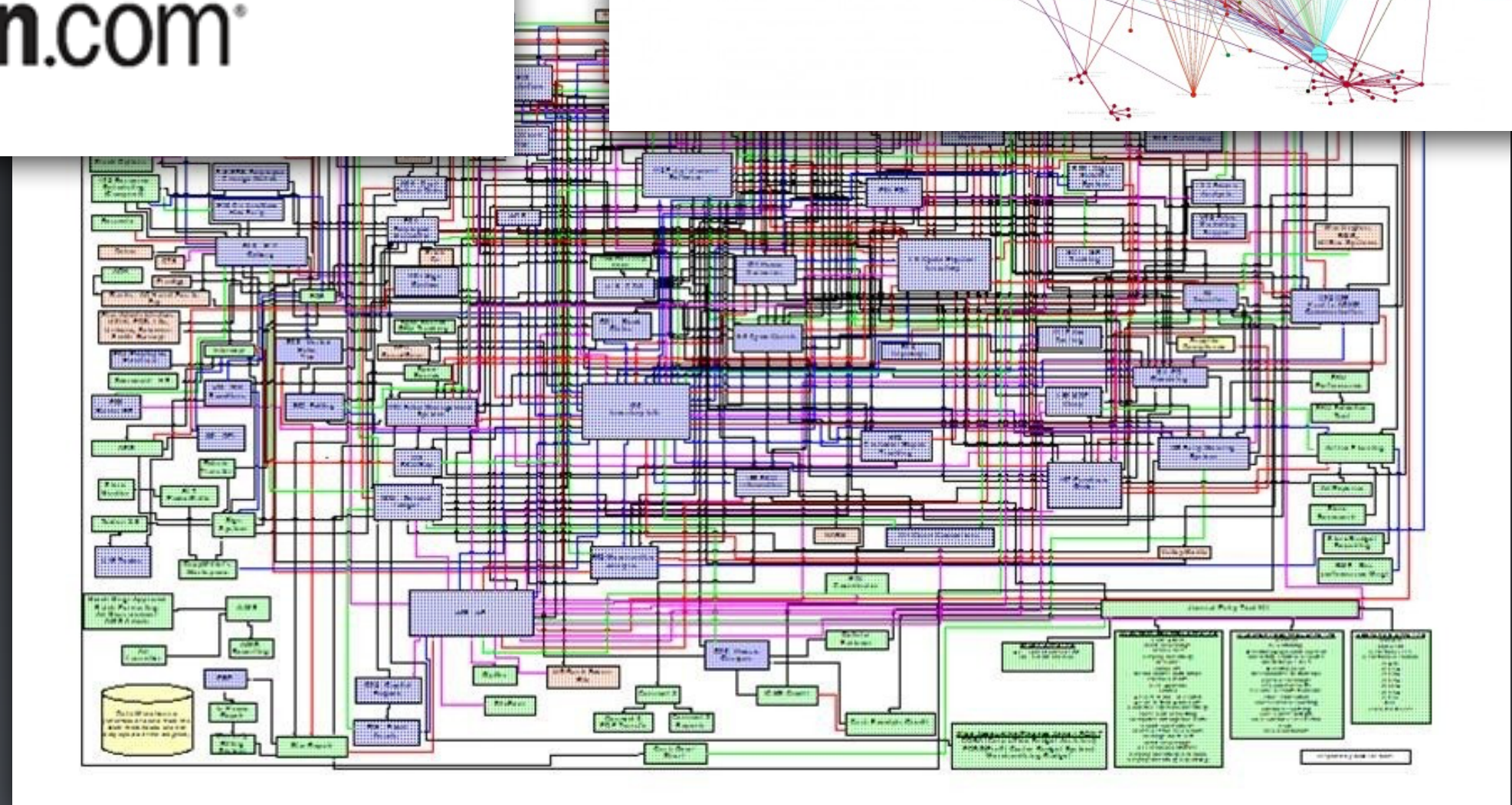
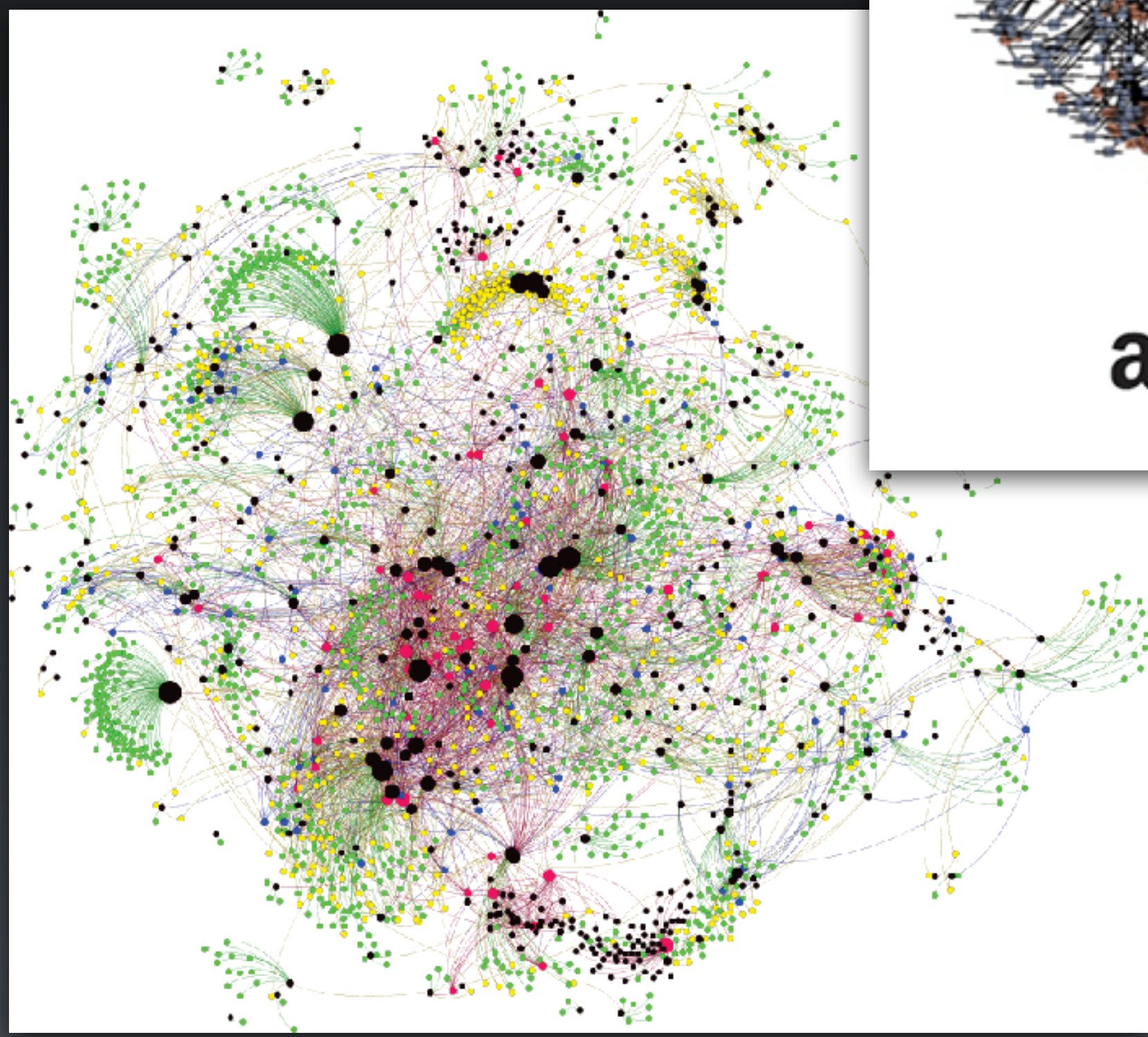
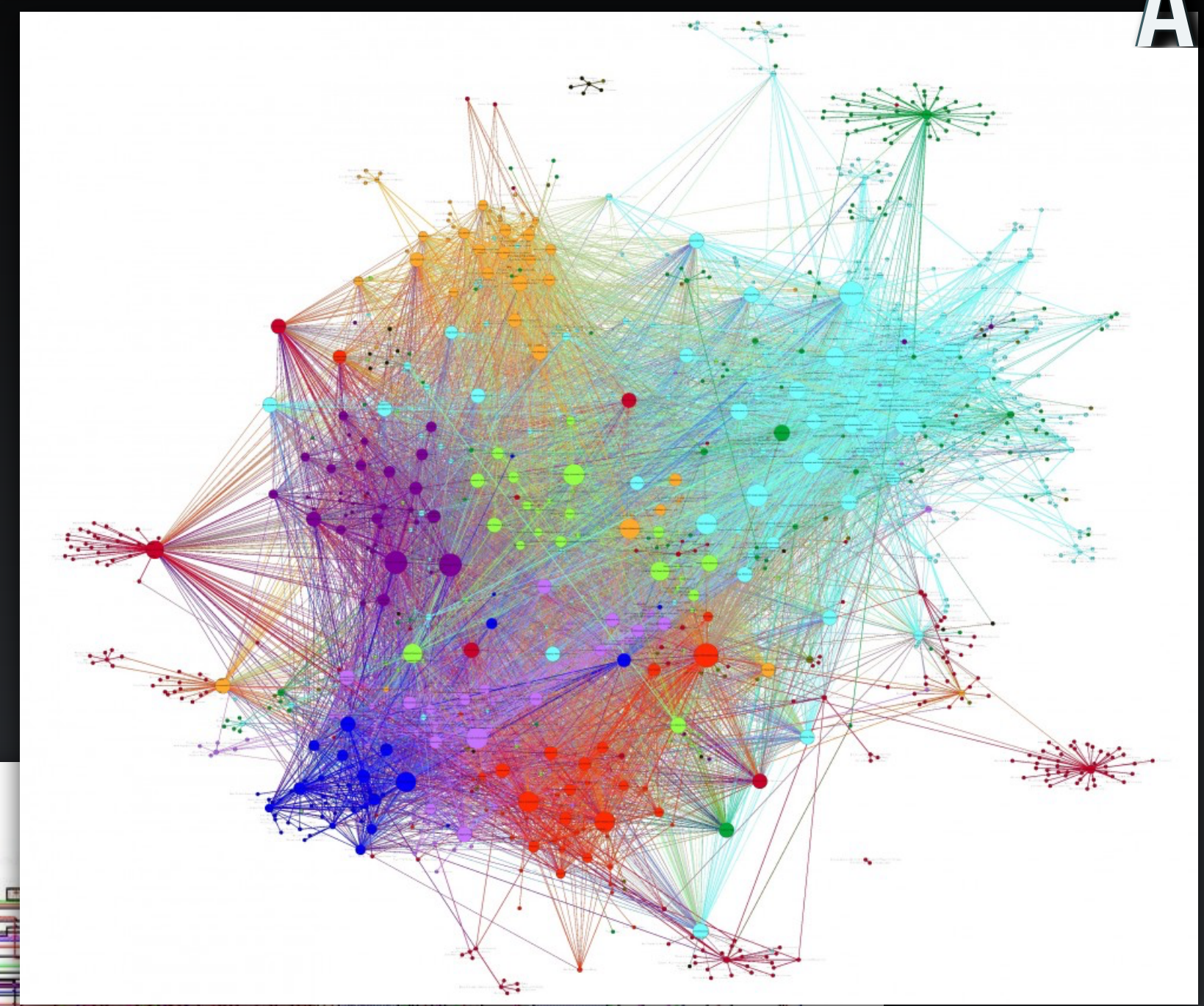


Empirical

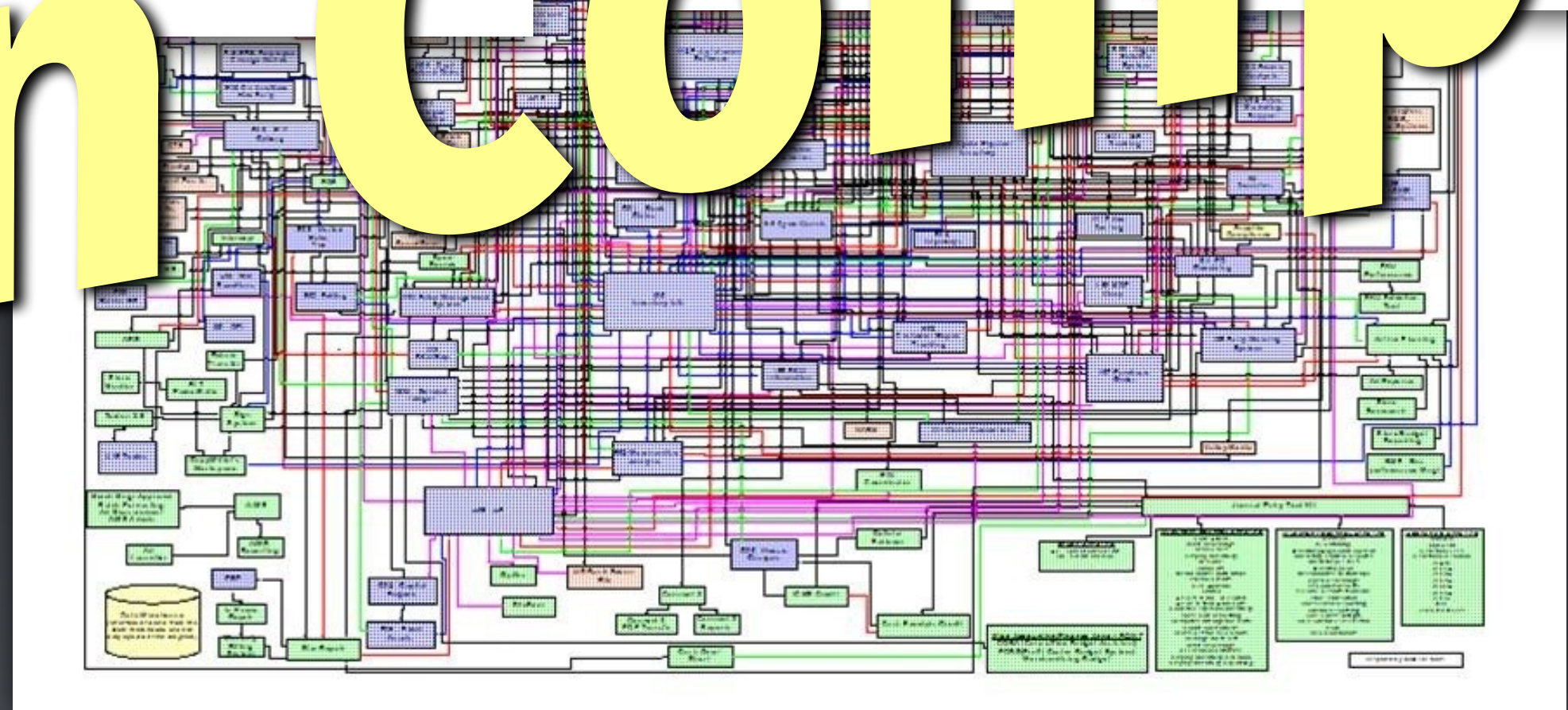
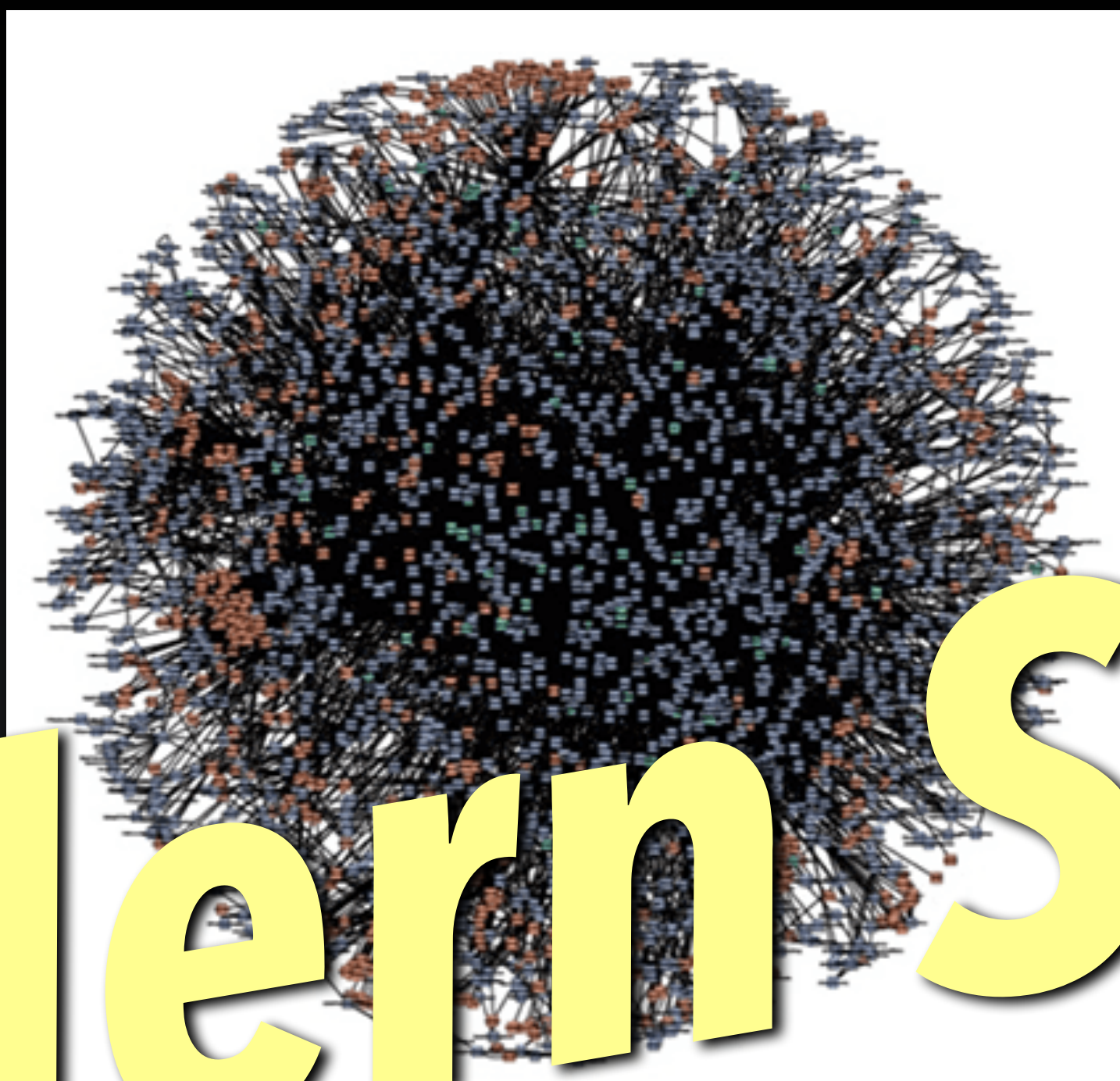




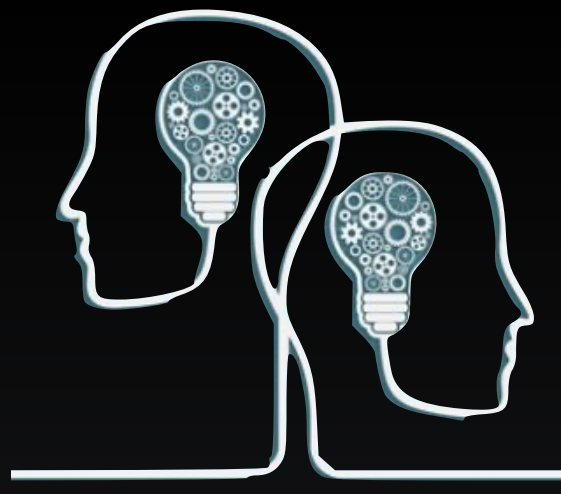
amazon.com



A  B
A  B



Modern Software is Often Complex!

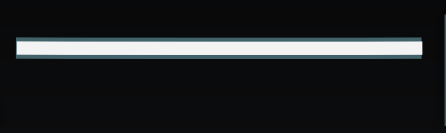


A  B

A ——— B

**Software Development
is about Learning &
Managing Complexity**

A  B

A  B

Managing Complexity



Managing Complexity

Modularity

Cohesion

Separation of Concerns

Abstraction

Coupling

Modularity



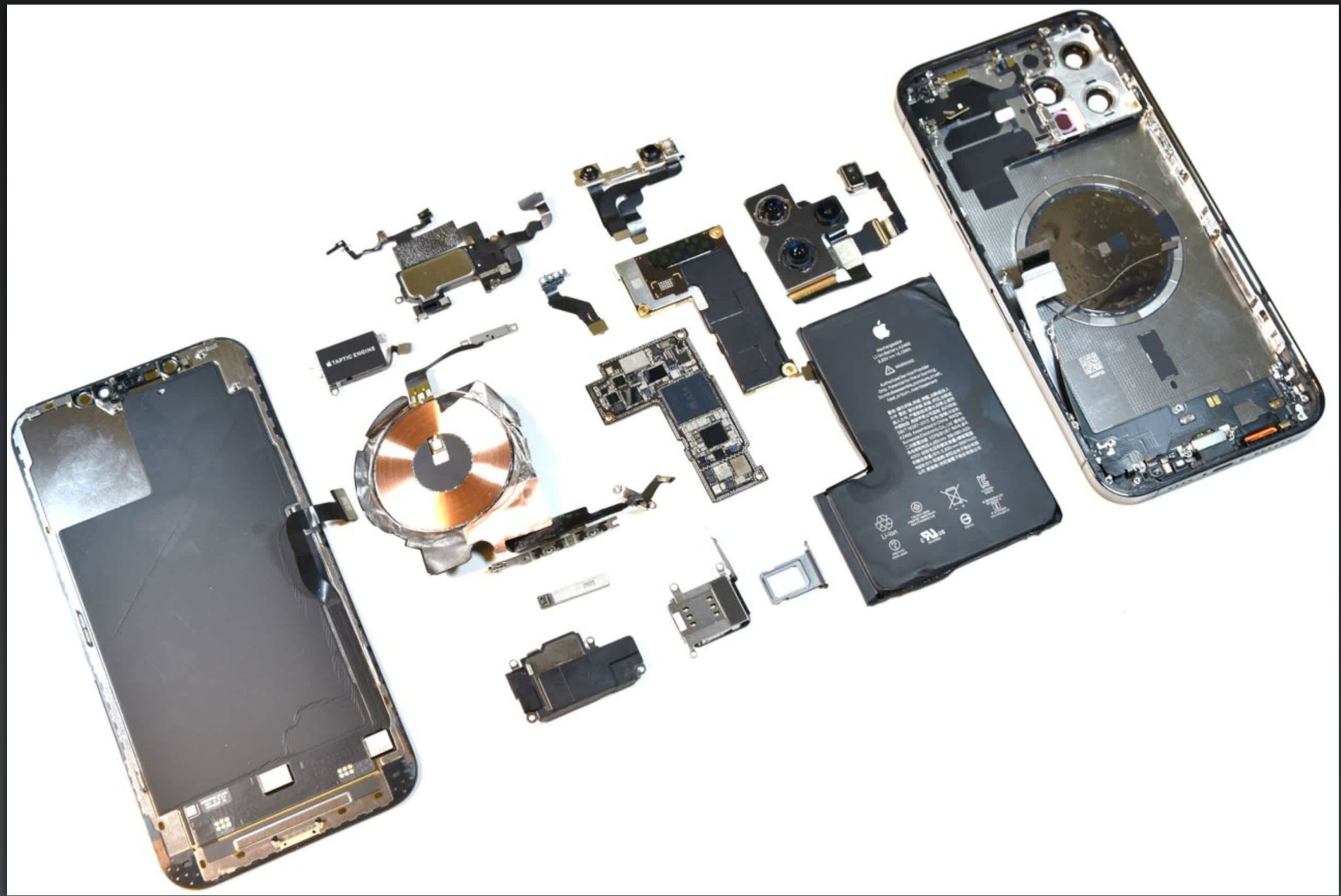
Modularity

A  B
A ——— B



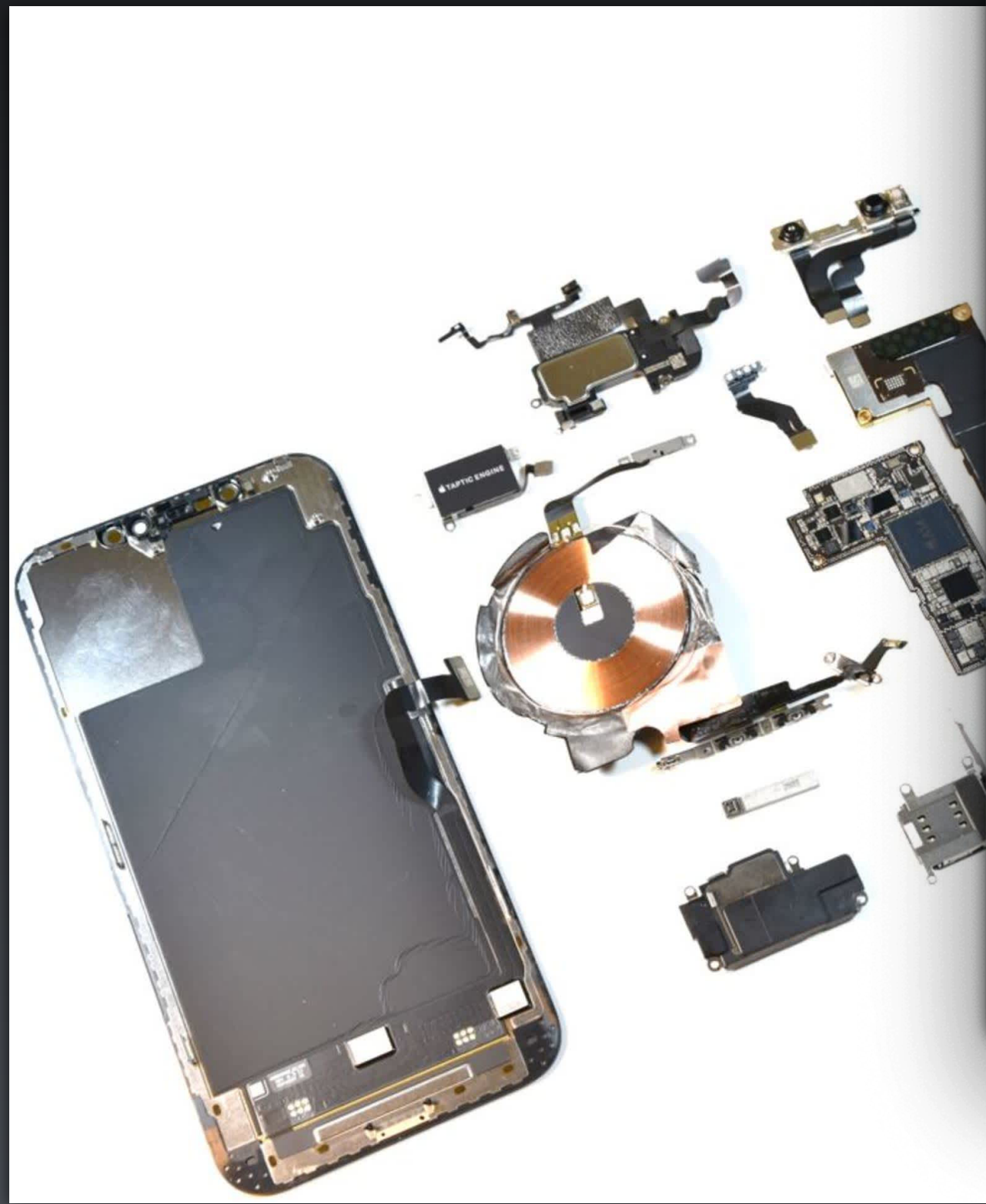
Modularity

A  B
A ——— B



Modularity

A  B
A ——— B



Printers & Scanners

Printers

- HP Officejet Pro 8610
Idle, Last Used

HP Officejet Pro 8610

Print Scan

Open Print Queue...

Options & Supplies...

Location:

Kind: HP Officejet Pro 8610-AirPrint

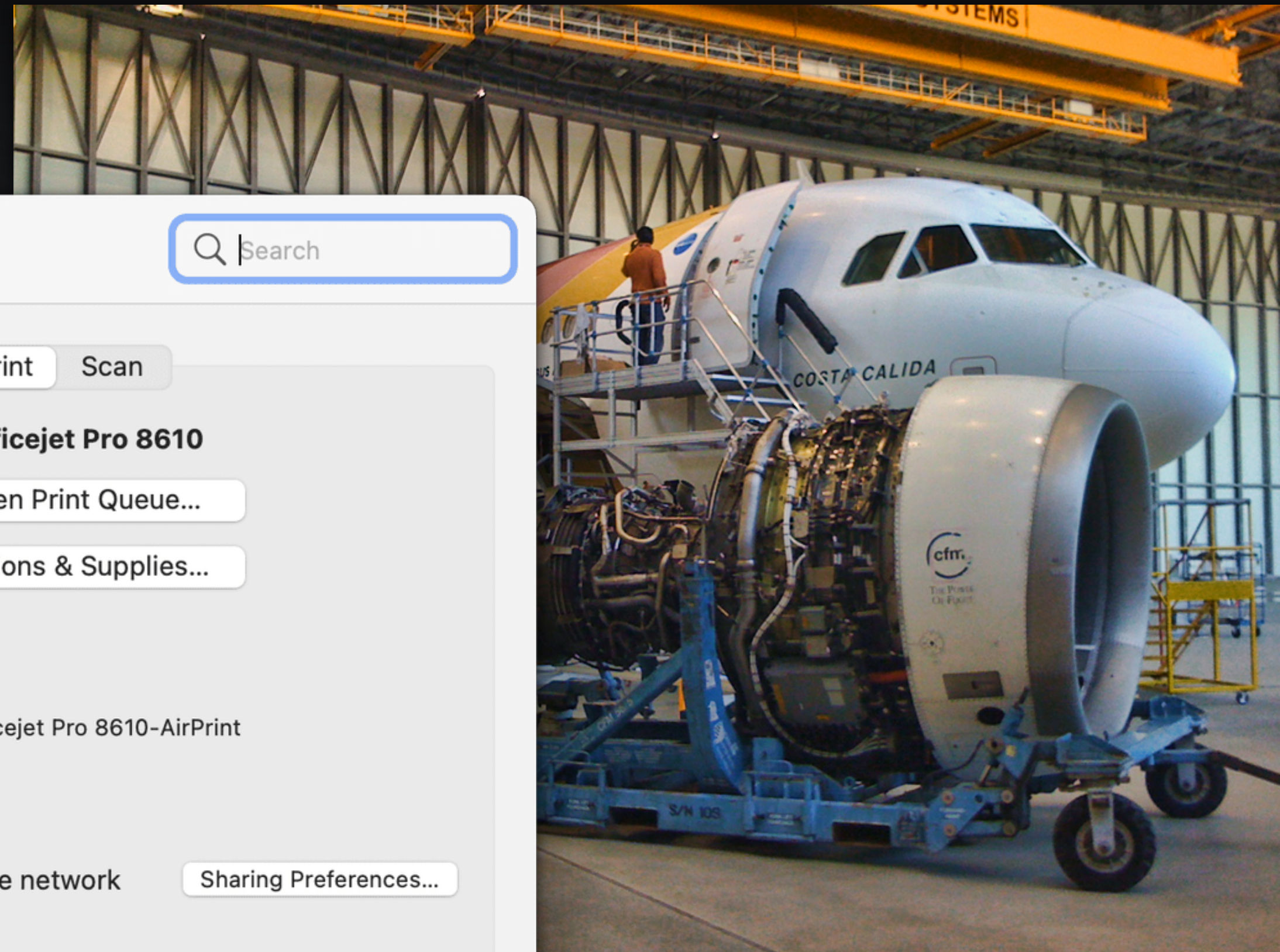
Status: Idle

Share this printer on the network

Sharing Preferences...

Default printer: Last Printer Used

Default paper size: A4



A  B

A ——— B

```
1  /*
2  * ecm_nss_non_ported_ipv4_process()
3  * Process a protocol that does not have port based identifiers
4  */
5  unsigned int ecm_nss_non_ported_ipv4_process(struct net_device *out_dev, struct net_device *out_dev_nat,
6  struct net_device *in_dev, struct net_device *in_dev_nat,
7  uint8_t *src_node_addr, uint8_t *src_node_addr_nat,
8  uint8_t *dest_node_addr, uint8_t *dest_node_addr_nat,
9  bool can_accel, bool is_routed, bool is_l2_encap, struct sk_buff *skb,
10 struct ecm_tracker_ip_header *ip_hdr,
11 struct nf_conn *ct, ecm_tracker_sender_type_t sender, ecm_db_direction_t ecm_dir,
12 struct nf_conntrack_tuple *orig_tuple, struct nf_conntrack_tuple *reply_tuple,
13 ip_addr_t ip_src_addr, ip_addr_t ip_dest_addr, ip_addr_t ip_src_addr_nat, ip_addr_t
. ip_dest_addr_nat)
14 {
15     struct ecm_db_connection_instance *ci;
16     int protocol;
17     int src_port;
18     int src_port_nat;
19     int dest_port;
20     int dest_port_nat;
21     ip_addr_t match_addr;
22     struct ecm_classifier_instance *assignments[ECM_CLASSIFIER_TYPES];
23     int aci_index;
24     int assignment_count;
25     ecm_db_timer_group_t ci_orig_timer_group;
26     struct ecm_classifier_process_response prevalent_pr;
27
28     DEBUG_TRACE("Non-ported protocol src: " ECM_IP_ADDR_DOT_FMT ", dest: " ECM_IP_ADDR_DOT_FMT "\n",
29                 ECM_IP_ADDR_TO_DOT(ip_src_addr), ECM_IP_ADDR_TO_DOT(ip_dest_addr));
30
```


Modularity

A  B

A ——— B

```
60 ▼ if (unlikely(!ci)) {
61     struct ecm_db_mapping_instance *src_mi;
62     struct ecm_db_mapping_instance *dest_mi;
63     struct ecm_db_mapping_instance *src_nat_mi;
64     struct ecm_db_mapping_instance *dest_nat_mi;
65     struct ecm_db_node_instance *src_ni;
66     struct ecm_db_node_instance *dest_ni;
67     struct ecm_db_node_instance *src_nat_ni;
68     struct ecm_db_node_instance *dest_nat_ni;
69     struct ecm_classifier_default_instance *dci;
70     struct ecm_front_end_connection_instance *feci;
71     struct ecm_db_connection_instance *nci;
72     ecm_classifier_type_t classifier_type;
73     int32_t to_list_first;
74     struct ecm_db_iface_instance *to_list[ECM_DB_IFACE_HEIRARCHY_MAX];
75     int32_t to_nat_list_first;
76     struct ecm_db_iface_instance *to_nat_list[ECM_DB_IFACE_HEIRARCHY_MAX];
77     int32_t from_list_first;
78     struct ecm_db_iface_instance *from_list[ECM_DB_IFACE_HEIRARCHY_MAX];
79     int32_t from_nat_list_first;
80     struct ecm_db_iface_instance *from_nat_list[ECM_DB_IFACE_HEIRARCHY_MAX];
81
82     DEBUG_INFO("New connection from " ECM_IP_ADDR_DOT_FMT " to " ECM_IP_ADDR_DOT_FMT "\n",
83               ECM_IP_ADDR_TO_DOT(ip_src_addr), ECM_IP_ADDR_TO_DOT(ip_dest_addr));
84
85     /*
86      * Before we attempt to create the connection are we being terminated?
87      */
88     spin_lock_bh(&ecm_nss_ipv4_lock);
89 ▼   if (ecm_nss_ipv4_terminate_pending) {
90       spin_unlock_bh(&ecm_nss_ipv4_lock);
```

Modularity

A  B

A — B

```
394 */
395 ecm_db_connection_add(nci, feci, src_mi, dest_mi, src_nat_mi, dest_nat_mi,
396 src_ni, dest_ni, src_nat_ni, dest_nat_ni,
397 4, protocol, ecm_dir,
398 NULL /* final callback */,
399 ecm_nss_non_ported_ipv4_connection_defunct_callback,
400 tg, is_routed, nci);
401
402 spin_unlock_bh(&ecm_nss_ipv4_lock);
403
404 ci = nci;
405 DEBUG_INFO("%p: New Non-porting protocol %d connection created\n", ci, protocol);
406 }
407
408 /*
409 * No longer need references to the connection created
410 */
411 dci->base->deref(ecm_classifier_instance);
412 ecm_db_mapping_deref(dest_nat_mi);
413 ecm_db_node_deref(dest_nat_ni);
414 ecm_db_mapping_deref(src_nat_mi);
415 ecm_db_node_deref(src_nat_ni);
416 ecm_db_mapping_deref(dest_mi);
417 ecm_db_node_deref(dest_ni);
418 ecm_db_mapping_deref(src_mi);
419 ecm_db_node_deref(src_ni);
420 feci->deref(feci);
421 }
422
423 /*
```

361 Lines in an if statement

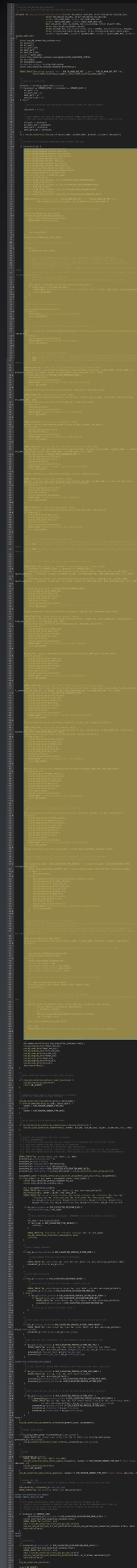
Modularity

A  B

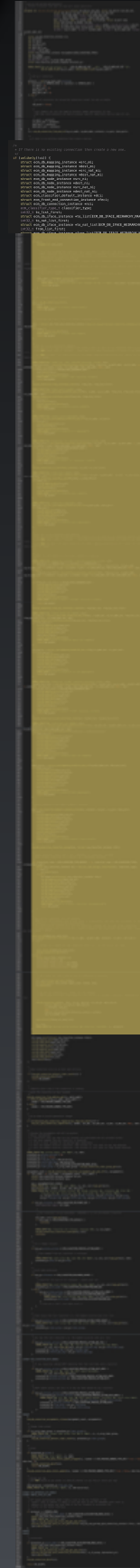
A — B

```
2  * ecm_nss_non_ported_ipv4_process()
3  * Process a protocol that does not have ported identifiers
4  */
5  unsigned int ecm_nss_non_ported_ipv4_process(struct net_device *out_dev, struct net_device *out_dev_nat,
6                                             struct net_device *in_dev, struct net_device *in_dev_nat,
7                                             uint8_t *src_node_addr, uint8_t *src_node_addr_nat,
8                                             uint8_t *dest_node_addr, uint8_t *dest_node_addr_nat,
9                                             bool can_accel, bool is_routed, bool is_l2_encap, struct sk_buff *skb,
10                                            struct ecm_tracker_ip_header *ip_hdr,
11                                            struct nf_conn *ct, ecm_tracker_sender_type_t sender, ecm_db_direction_t ecm_dir,
12                                            struct nf_conntrack_tuple *orig_tuple, struct nf_conntrack_tuple *reply_tuple,
13                                            ip_addr_t ip_src_addr, ip_addr_t ip_dest_addr, ip_addr_t ip_src_addr_nat, ip_addr_t
14                                            ip_dest_addr_nat)
15  {
16      struct ecm_db_connection_instance *ci;
17      int protocol;
18      int src_port;
19      int src_port_nat;
20      int dest_port;
21      int dest_port_nat;
22      ip_addr_t match_addr;
23      struct ecm_classifier_instance *assignments[ECM_CLASSIFIER_TYPES];
24      int aci_index;
25      int assignment_count;
26      ecm_db_timer_group_t ci_orig_timer_group;
27      struct ecm_classifier_process_response prevalent_pr;
28
29      DEBUG_TRACE("Non-ported protocol src: " ECM_IP_ADDR_DOT_FMT ", dest: " ECM_IP_ADDR_DOT_FMT "\n",
30                  ECM_IP_ADDR_TO_DOT(ip_src_addr), ECM_IP_ADDR_TO_DOT(ip_dest_addr));
31
32      /*
```

Modularity

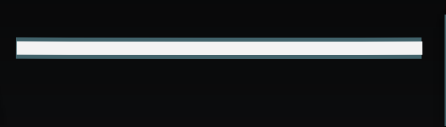


Modularity



Modularity

A  B

A  B

```
/*  
 * If there is no existing connection then create a new one.  
 */  
if (unlikely(!ci)) {  
    struct ecm_db_mapping_instance *src_mi;  
    struct ecm_db_mapping_instance *dest_mi;  
    struct ecm_db_mapping_instance *src_nat_mi;  
    struct ecm_db_mapping_instance *dest_nat_mi;  
    struct ecm_db_node_instance *src_ni;  
    struct ecm_db_node_instance *dest_ni;  
    struct ecm_db_node_instance *src_nat_ni;  
    struct ecm_db_node_instance *dest_nat_ni;  
    struct ecm_classifier_default_instance *dci;  
    struct ecm_front_end_connection_instance *fec_i;  
    struct ecm_db_connection_instance *nci;  
    ecm_classifier_type_t classifier_type;  
    int32_t to_list_first;  
    struct ecm_db_iface_instance *to_list[ECM_DB_IFACE_HEIRARCHY_MAX]  
    int32_t to_nat_list_first;  
    struct ecm_db_iface_instance *to_nat_list[ECM_DB_IFACE_HEIRARCHY_  
    int32_t from_list_first;  
    struct ecm_db_iface_instance *from_list[ECM_DB_IFACE_HEIRARCHY_M
```

Modularity

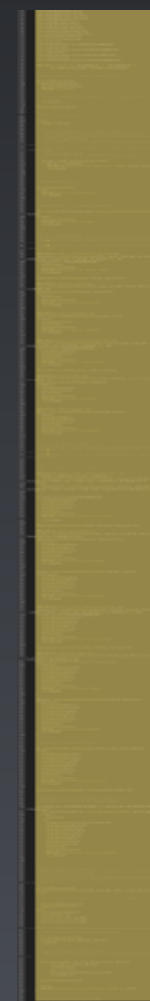


```
if (noConnection(ci)) {  
    createConnection();  
}
```

Modularity



```
if (noConnection(ci)) {  
    createConnection();  
}
```



A  B

A ——— B

"Good Design is about moving the things that are related closer together, and the things that aren't related further apart" - Kent Beck

A  B

A ——— B

"Good Design is about moving the things that are related closer together, and the things that aren't related further apart." - Kent Beck



Modularity

***things that aren't related
further apart***

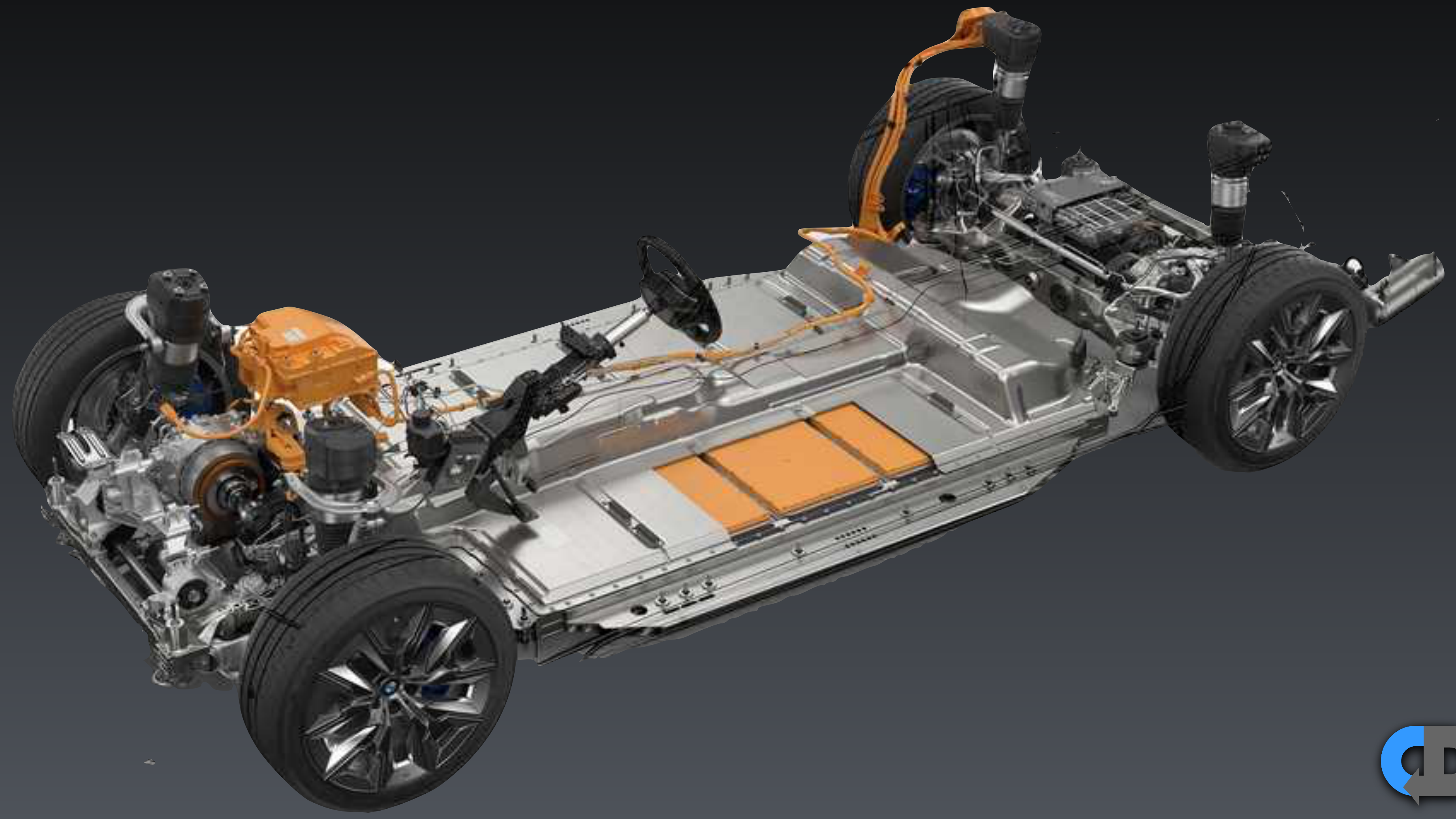
Cohesion

things that are related closer together

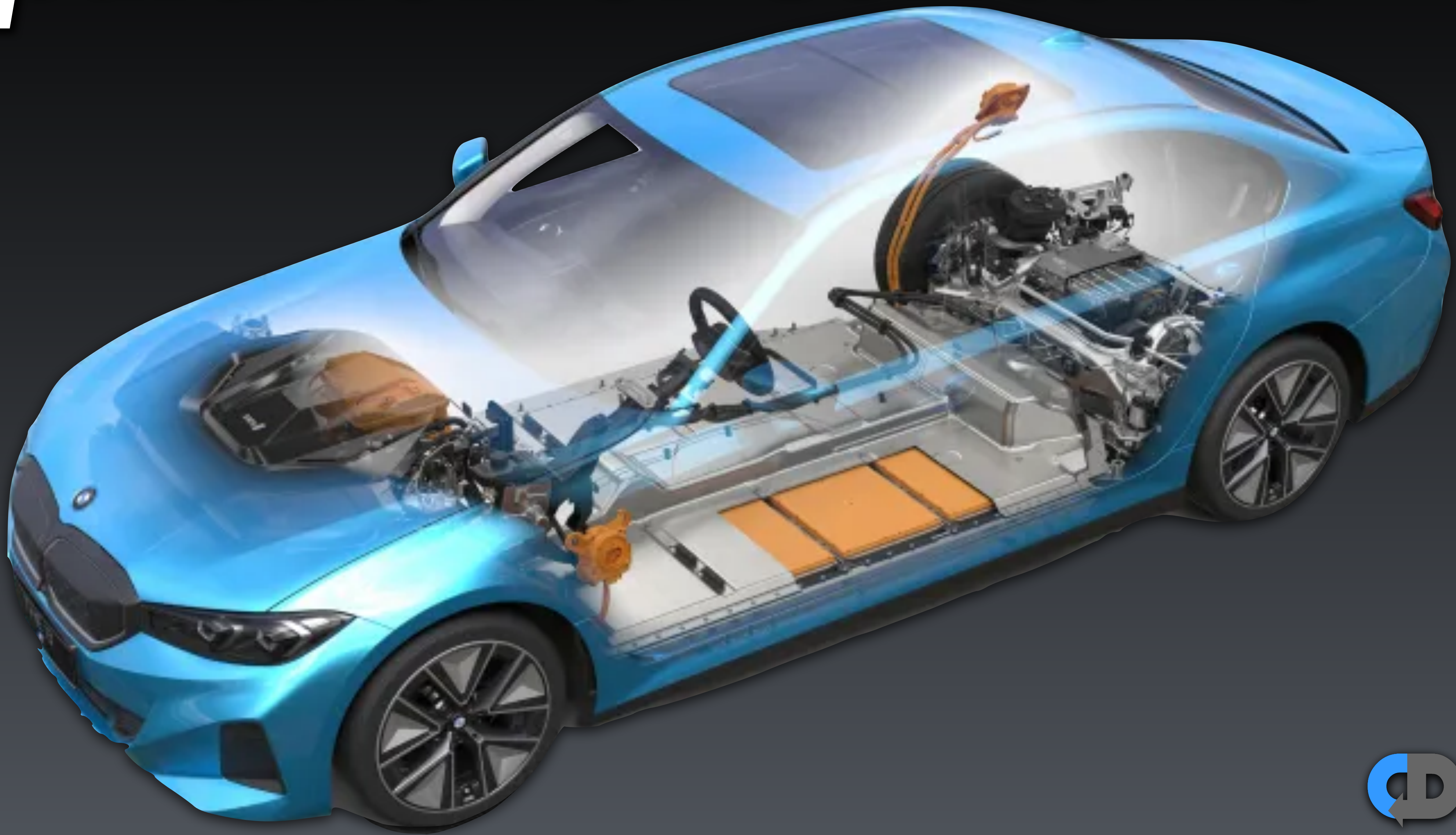
Separation of Concerns



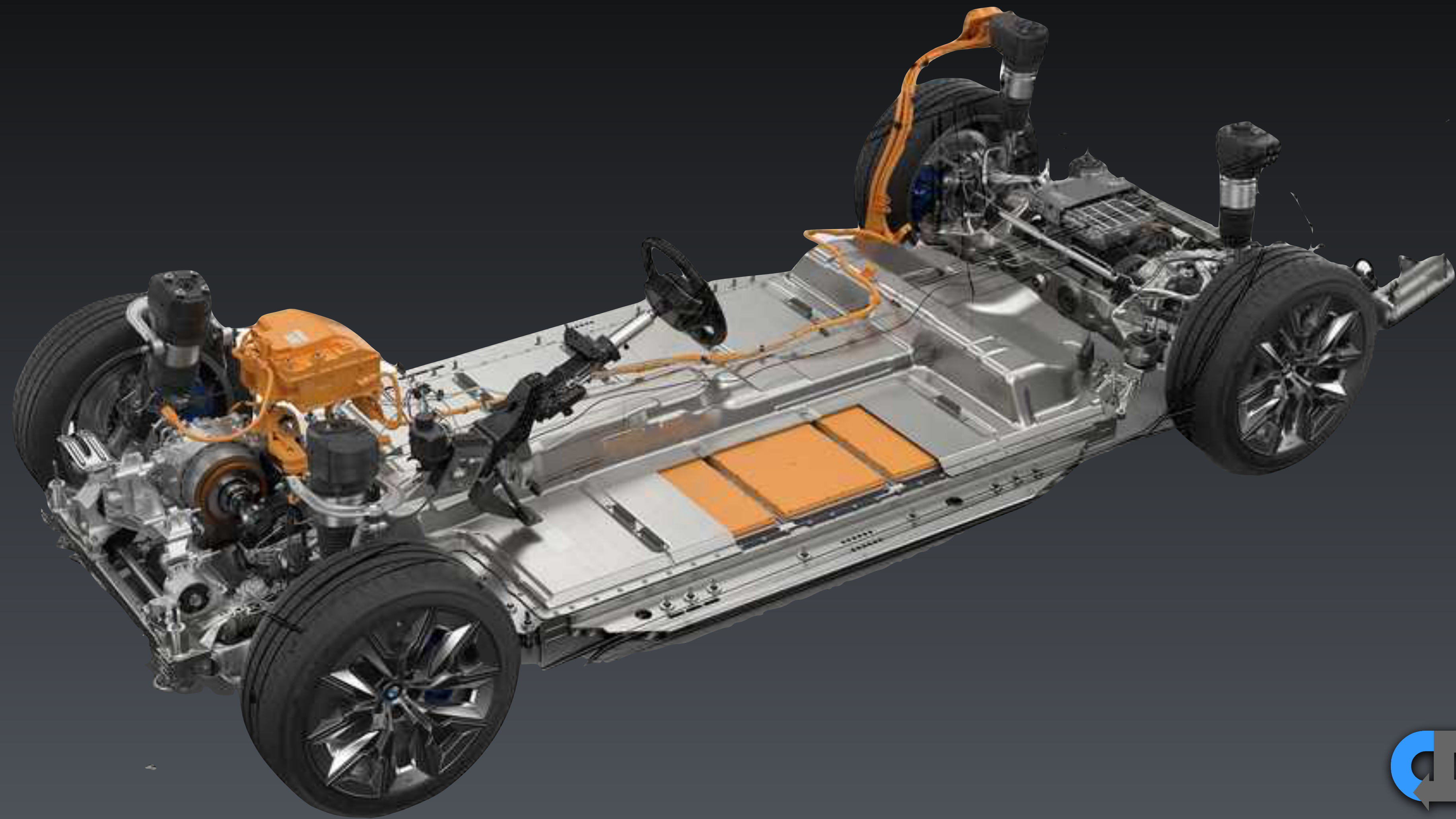
Separation of Concerns



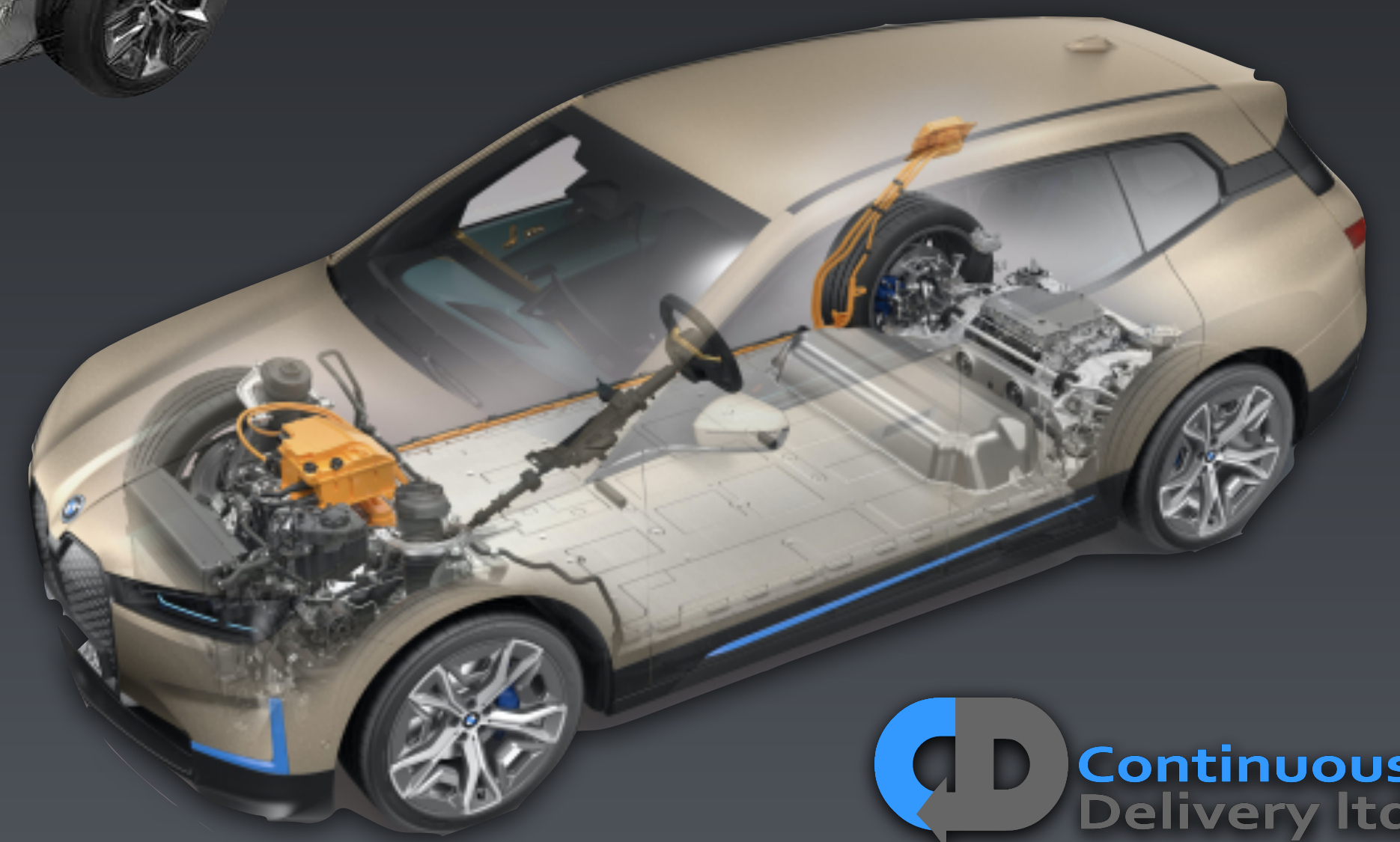
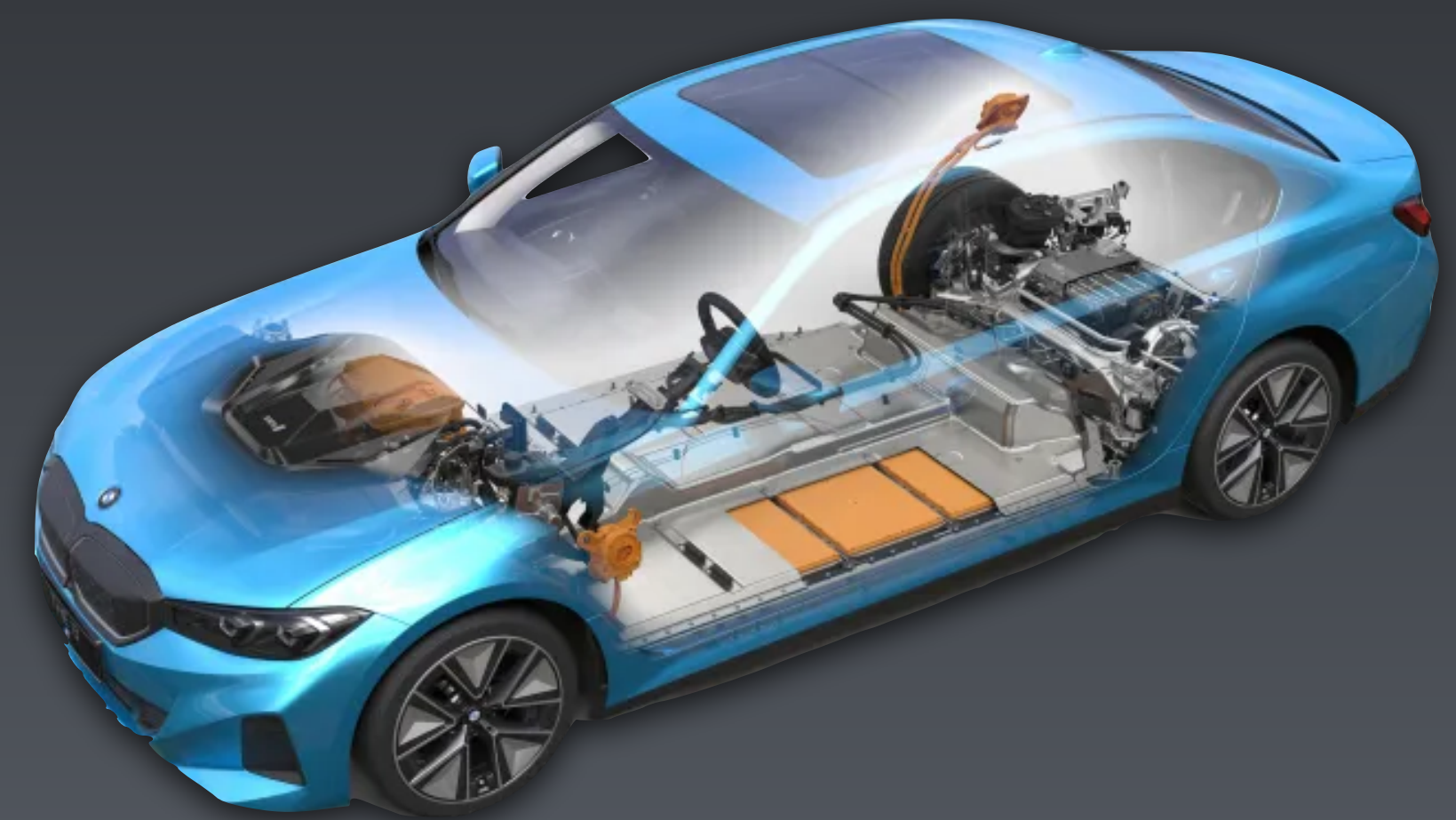
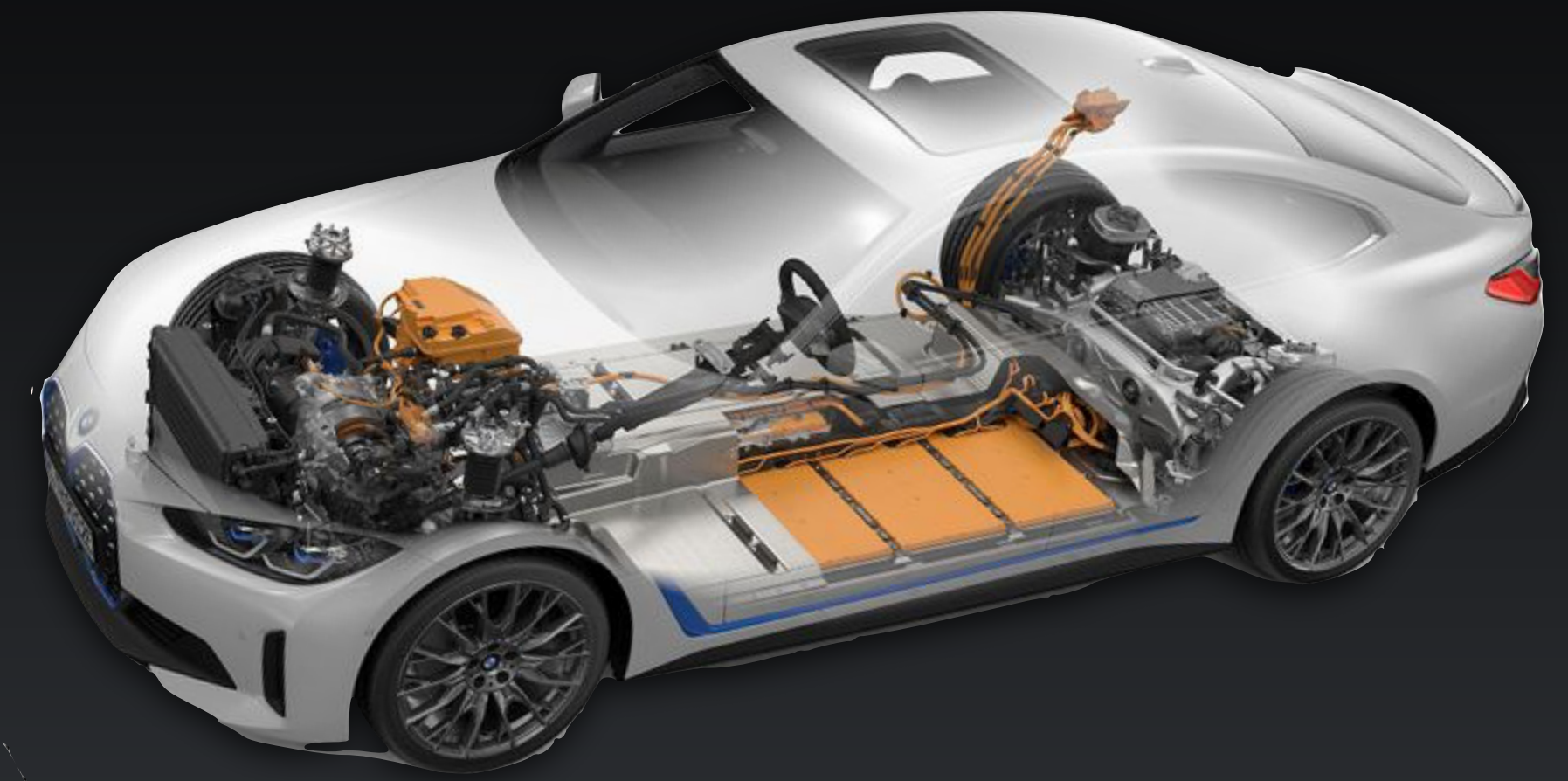
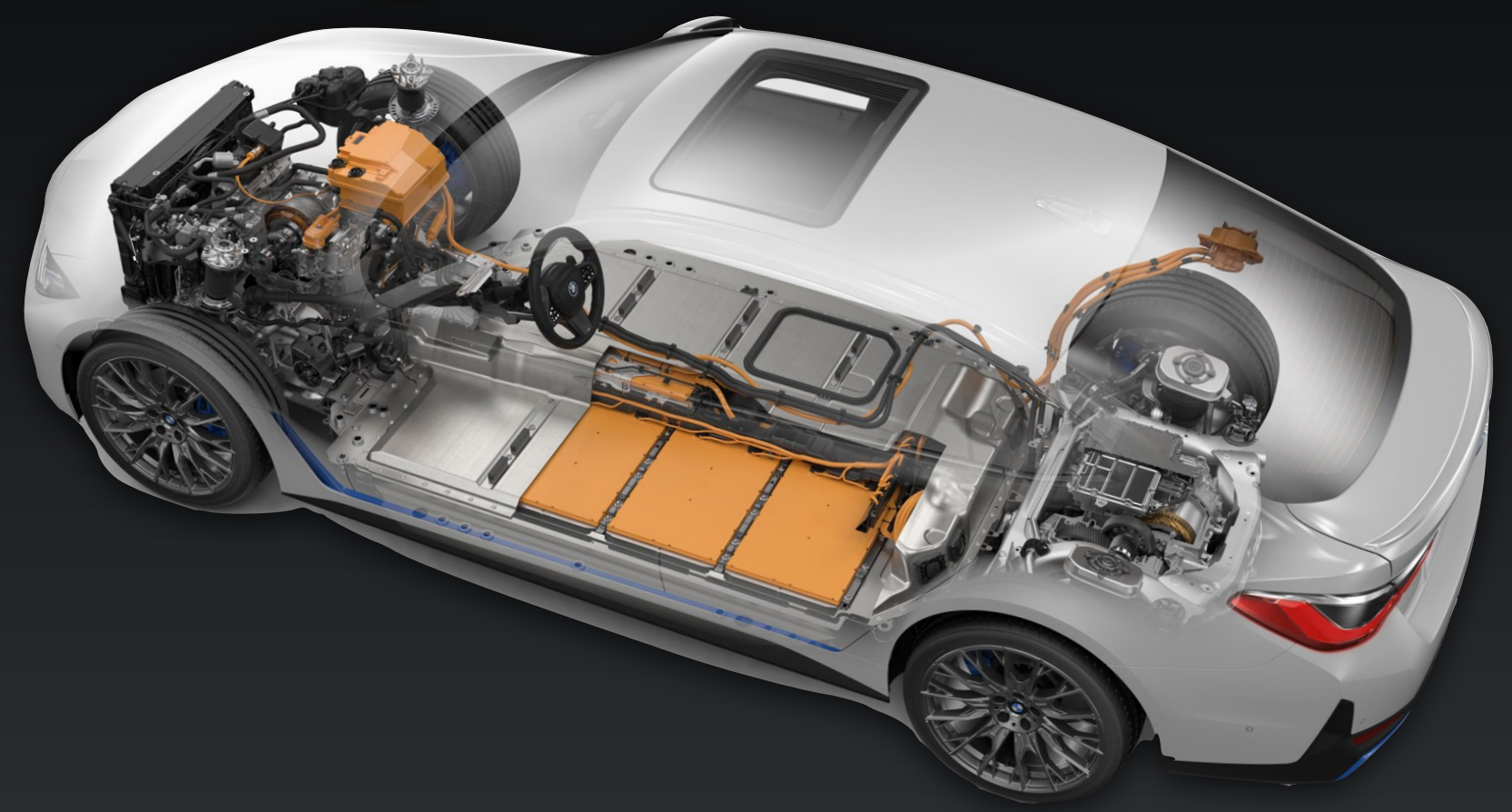
Separation of Concerns



Separation of Concerns



Separation of Concerns



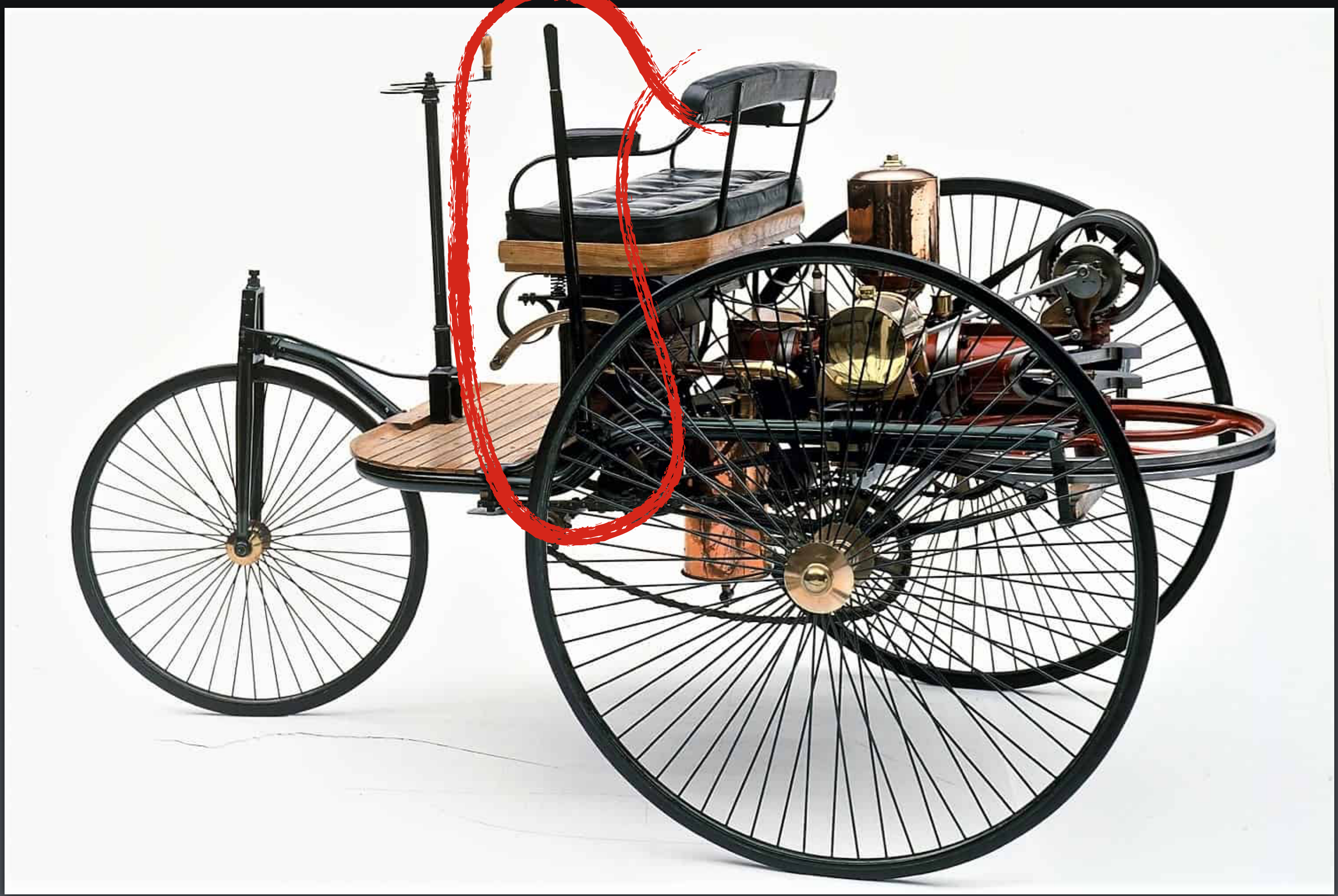
Abstraction

A  B
A  B



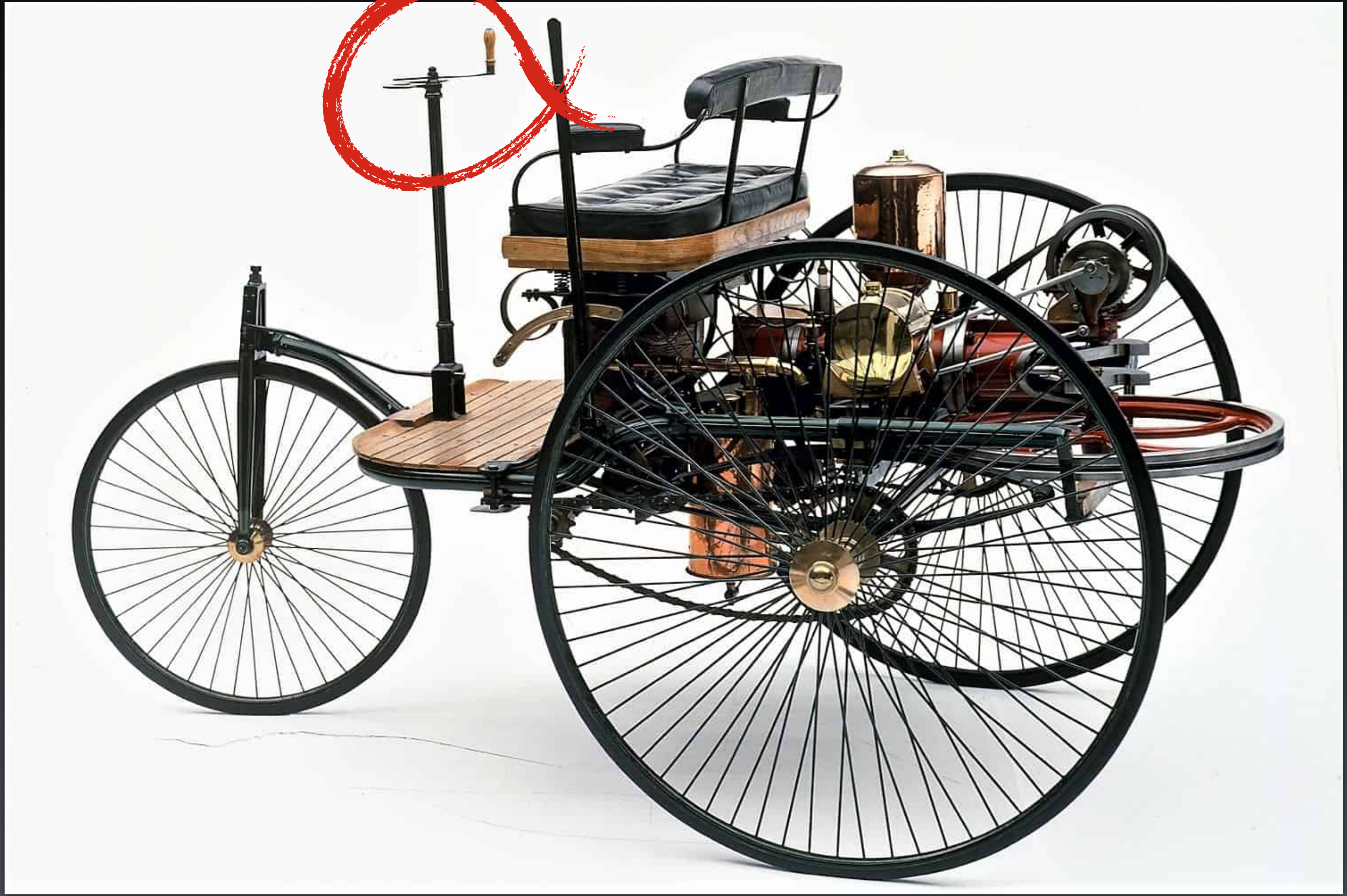
Abstraction

A  B
A  B



Abstraction

A  B
A  B



Abstraction

A  B
A  B



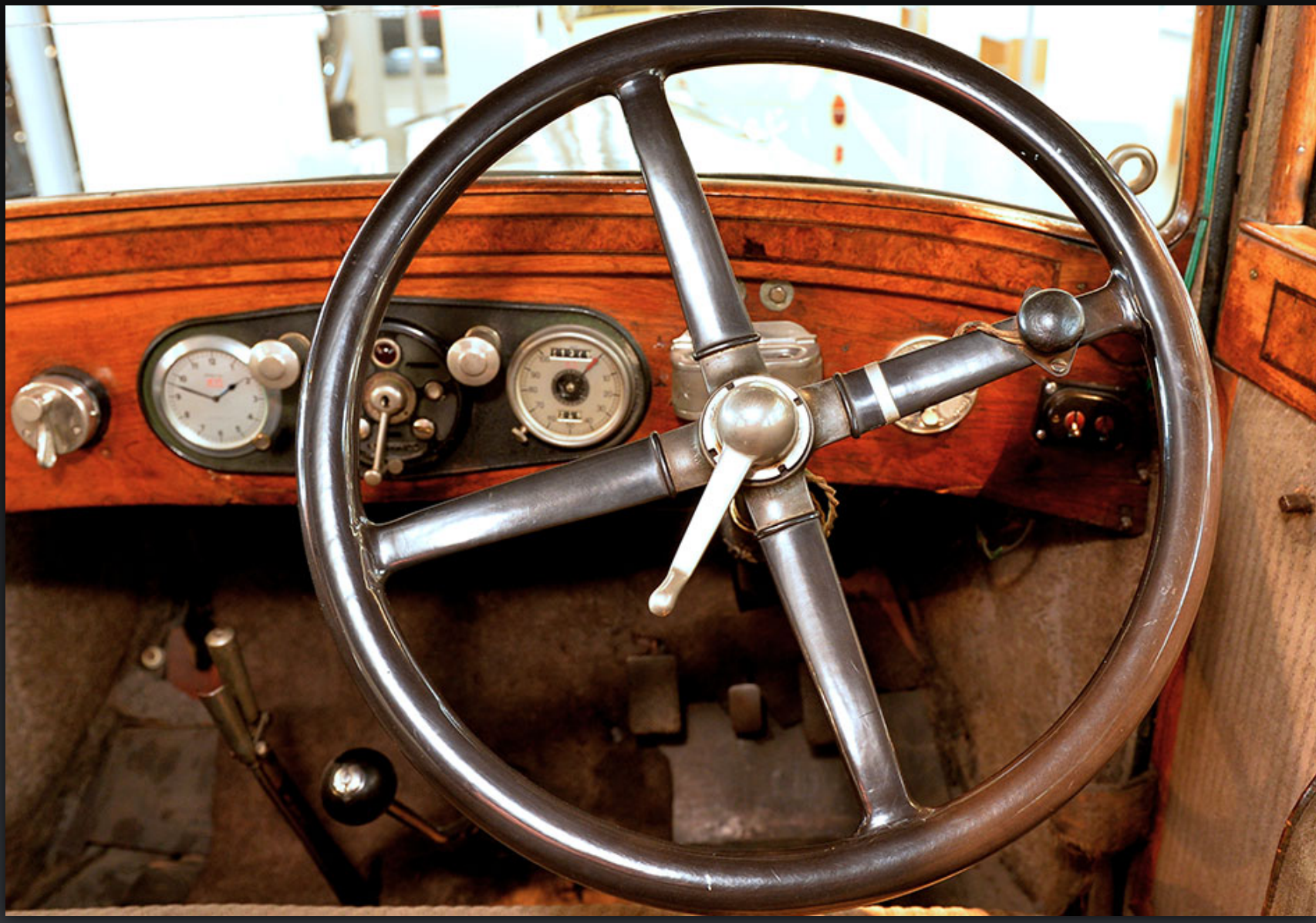
Abstraction

A  B
A  B



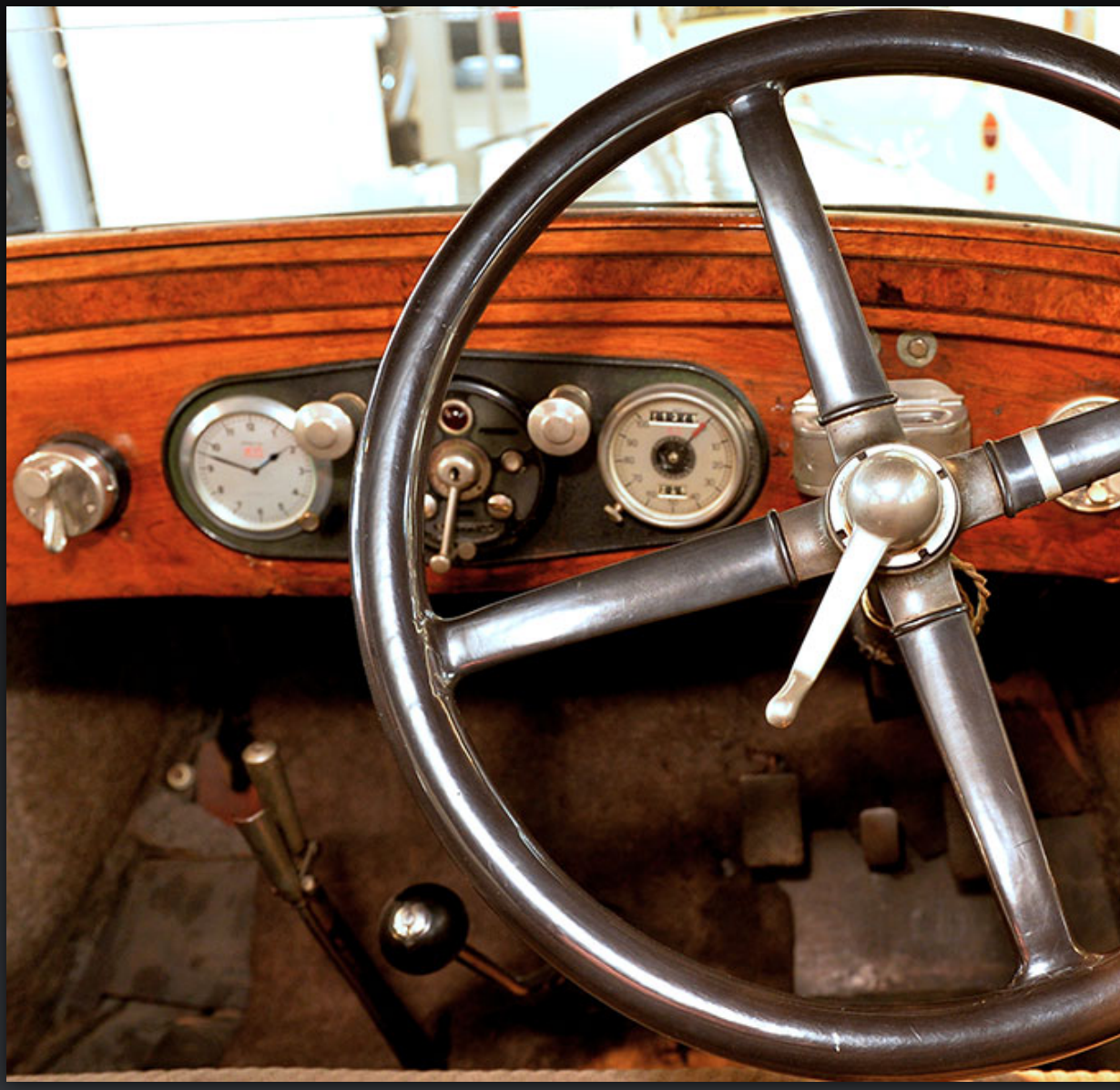
Abstraction

A *egg* B
A ——— B



Abstraction

A  B
A  B

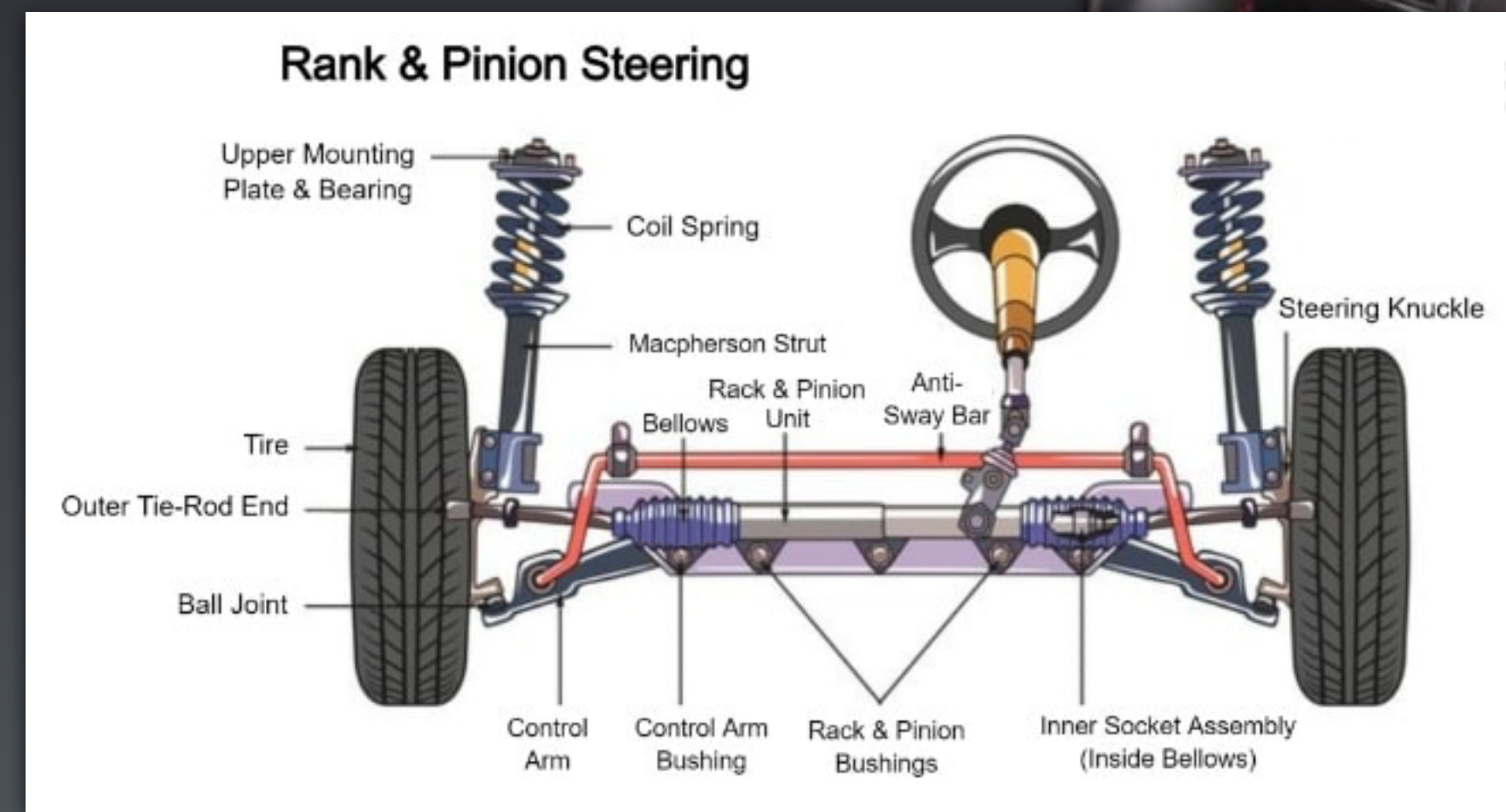
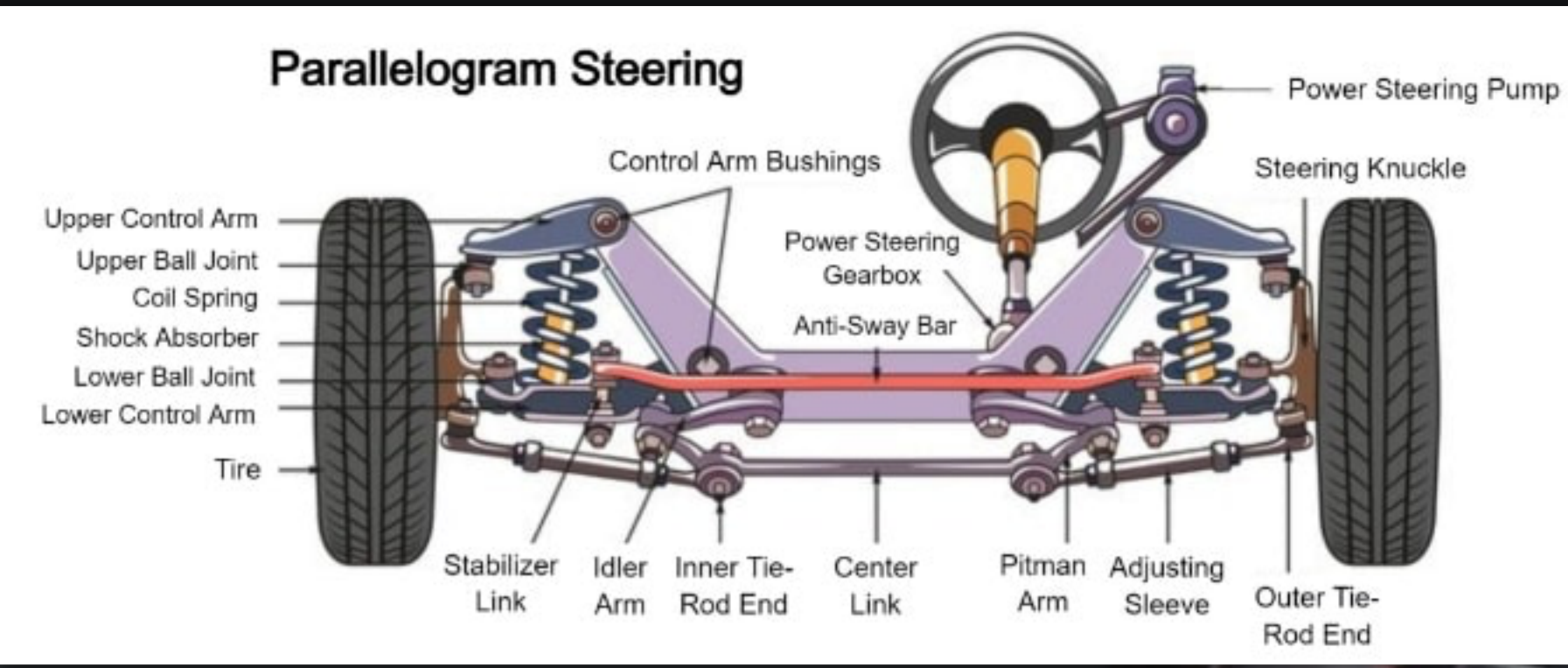


Abstraction



Abstraction

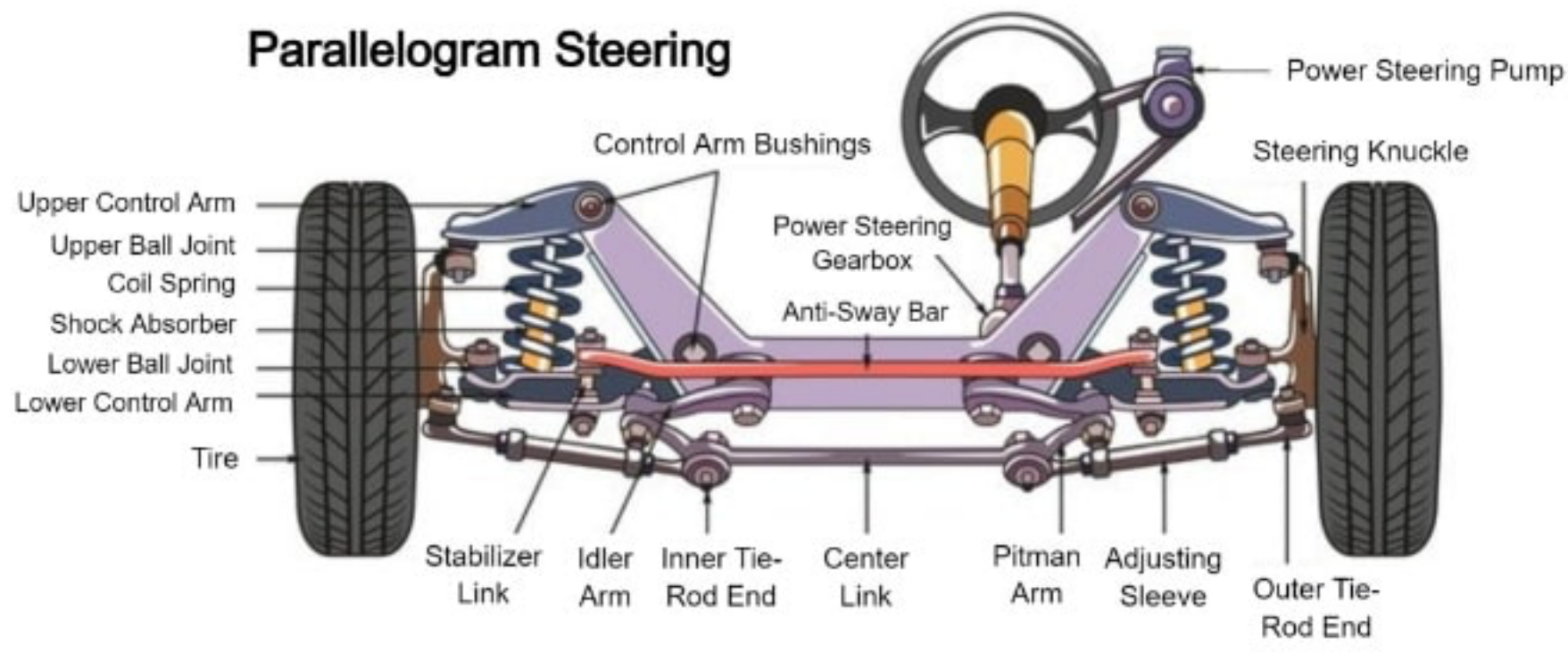
A  B
A  B



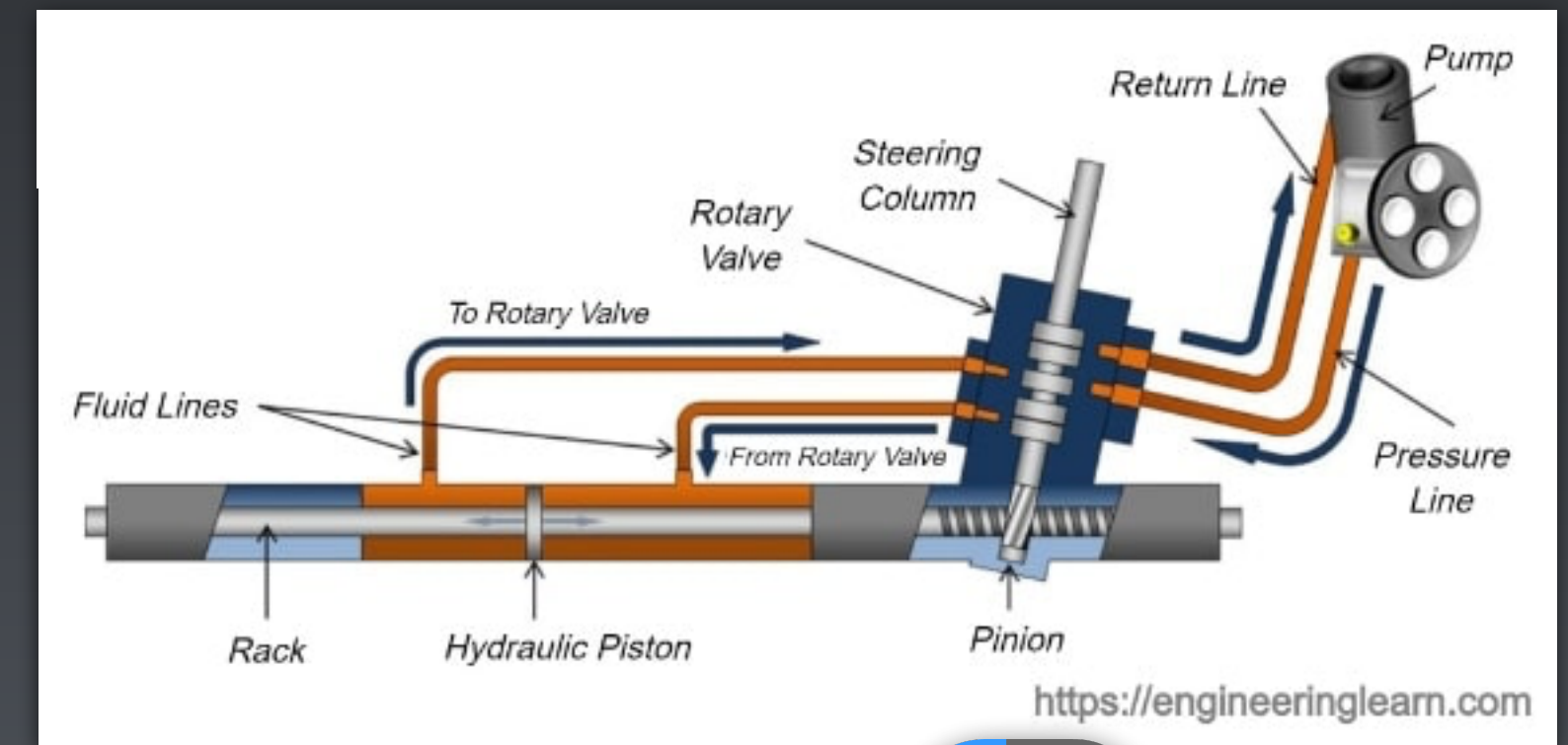
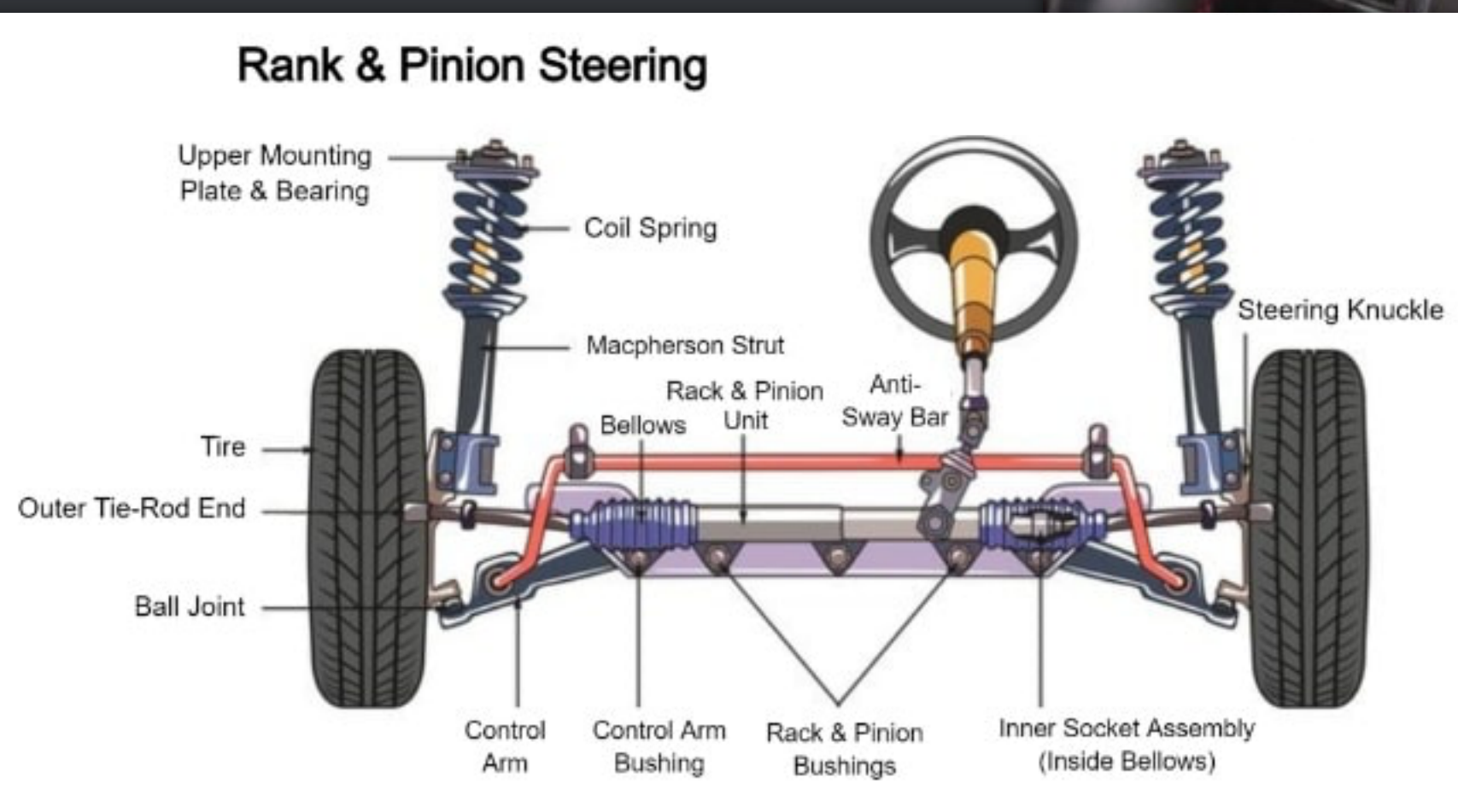
Abstraction

A  B
A ——— B

Parallelogram Steering



Rack & Pinion Steering



Abstraction



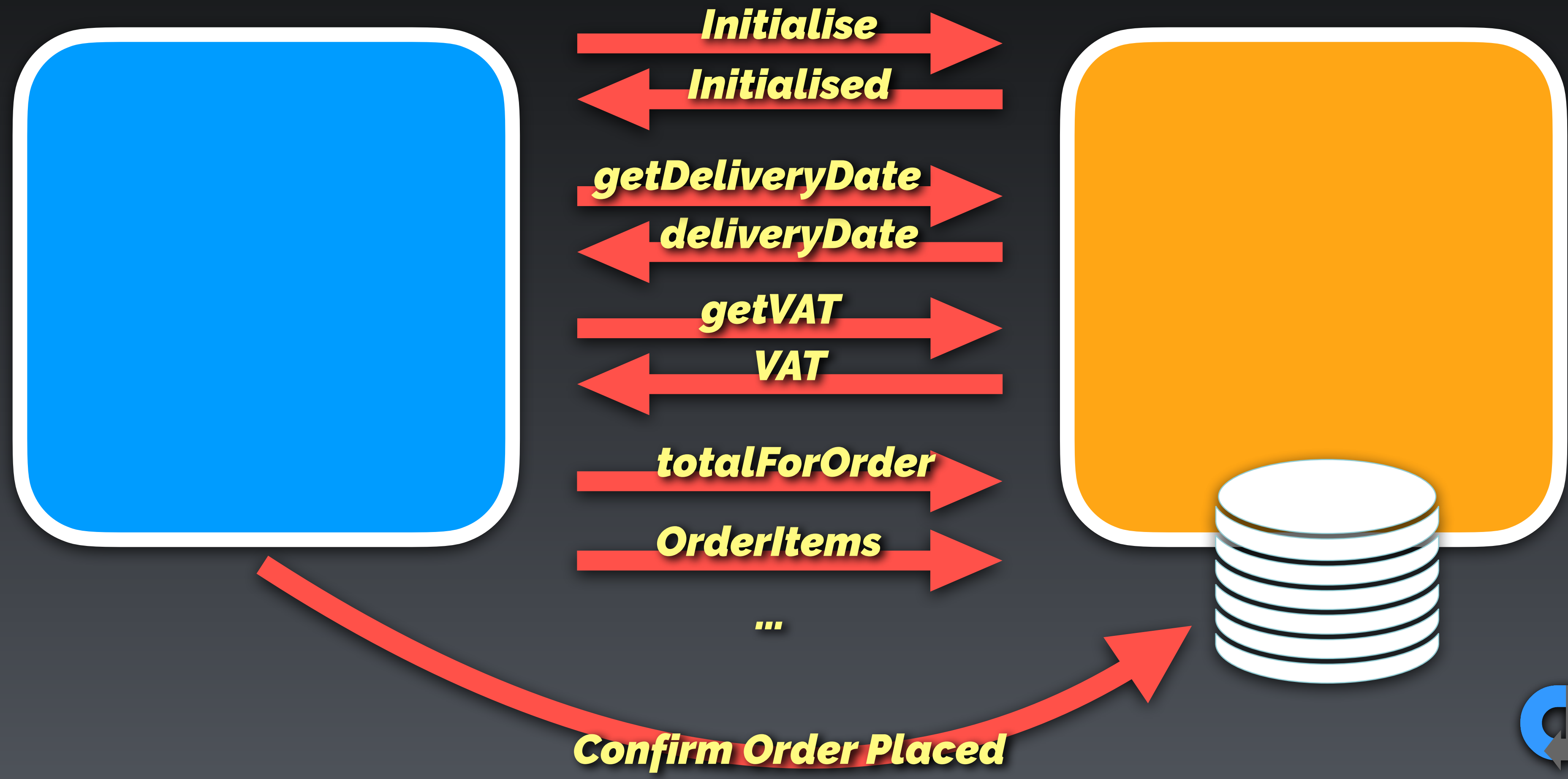
Abstraction



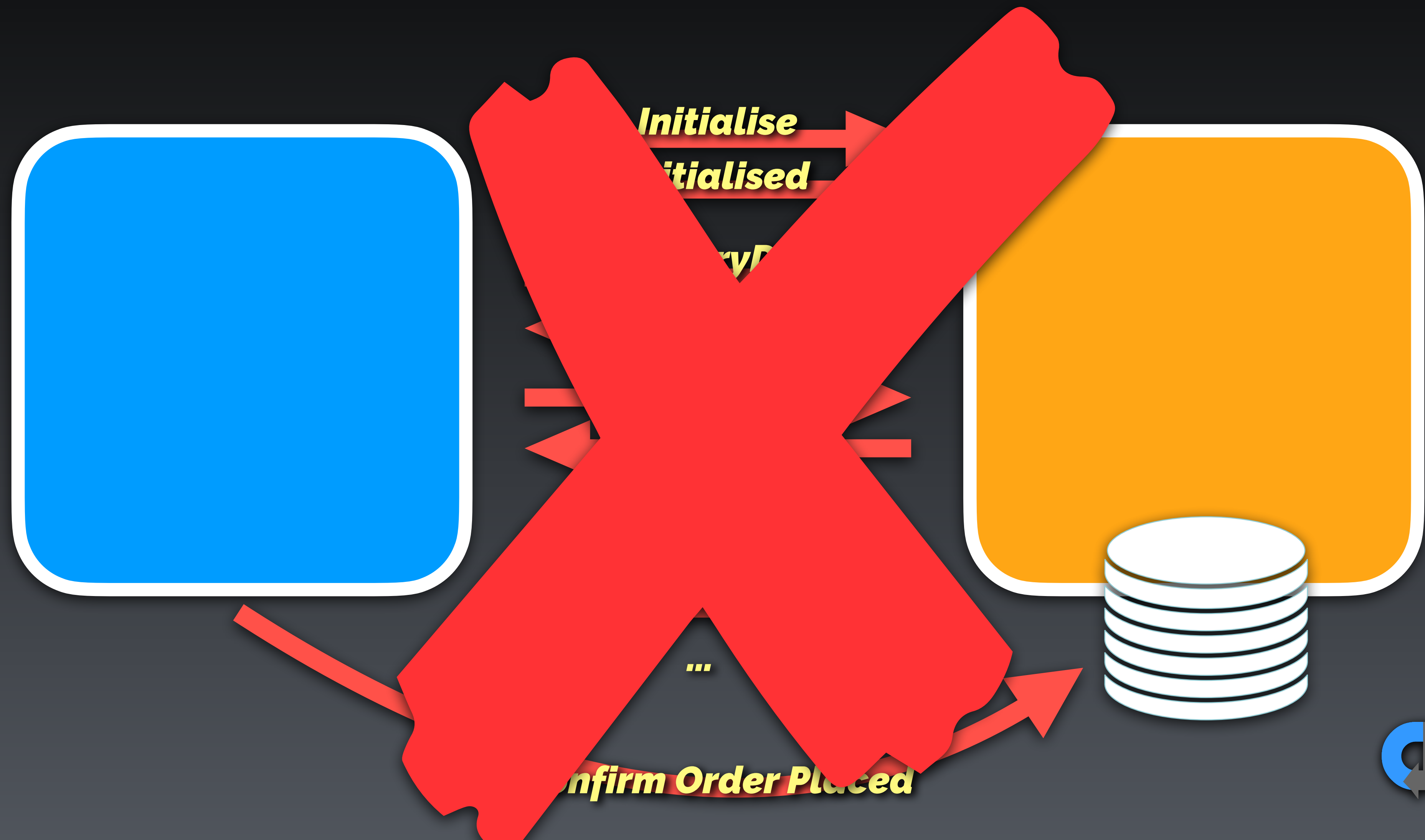
Abstraction



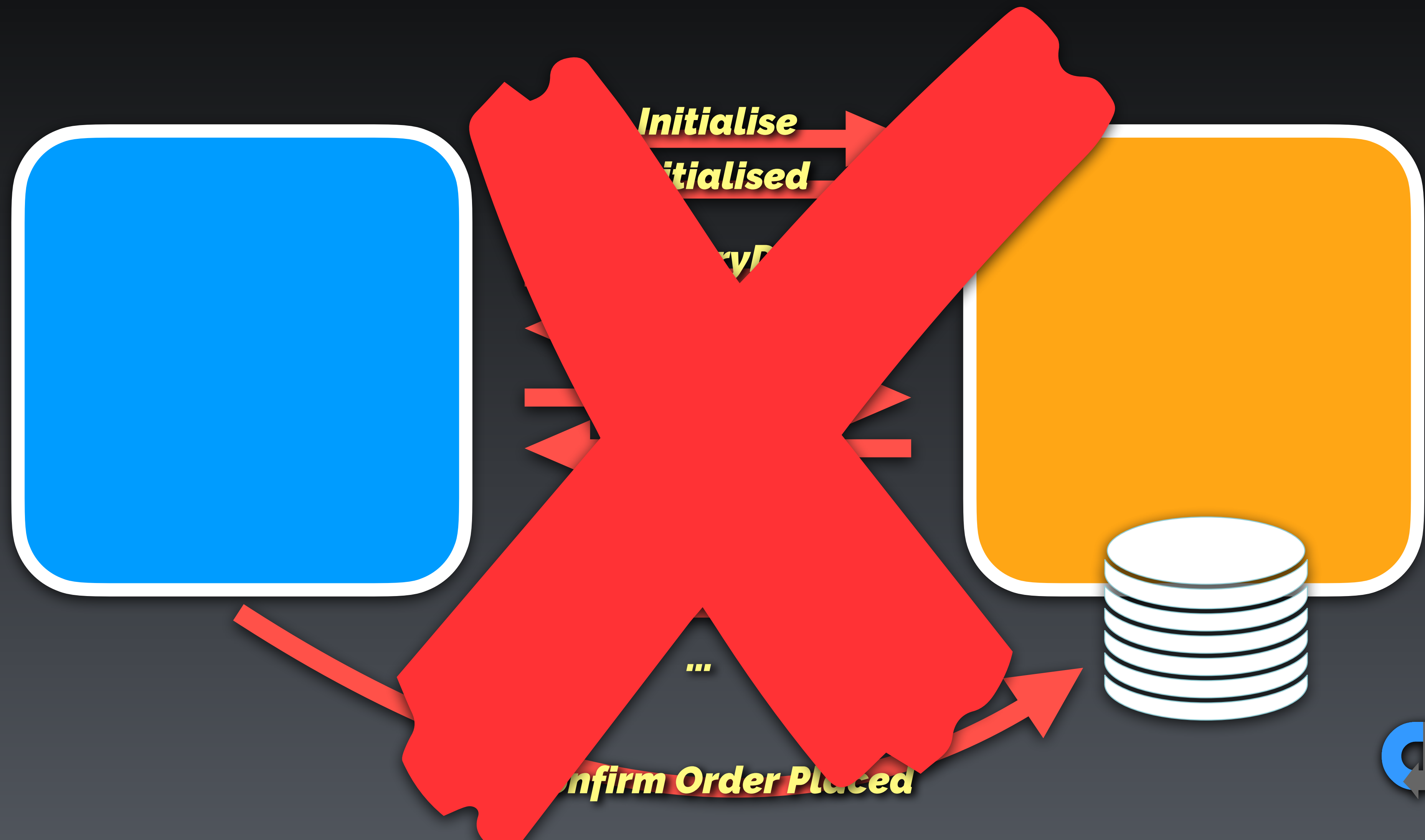
Abstraction



Abstraction



Abstraction



Abstraction



Abstraction



Abstraction



Abstraction



Abstraction




Keep Secrets!



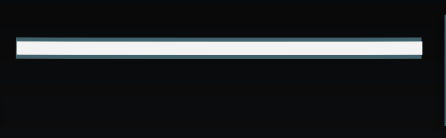
Coupling

A  B

A  B

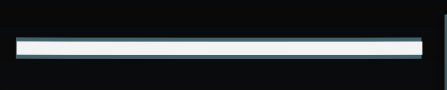


A  B

A  B

Coupling

A  B

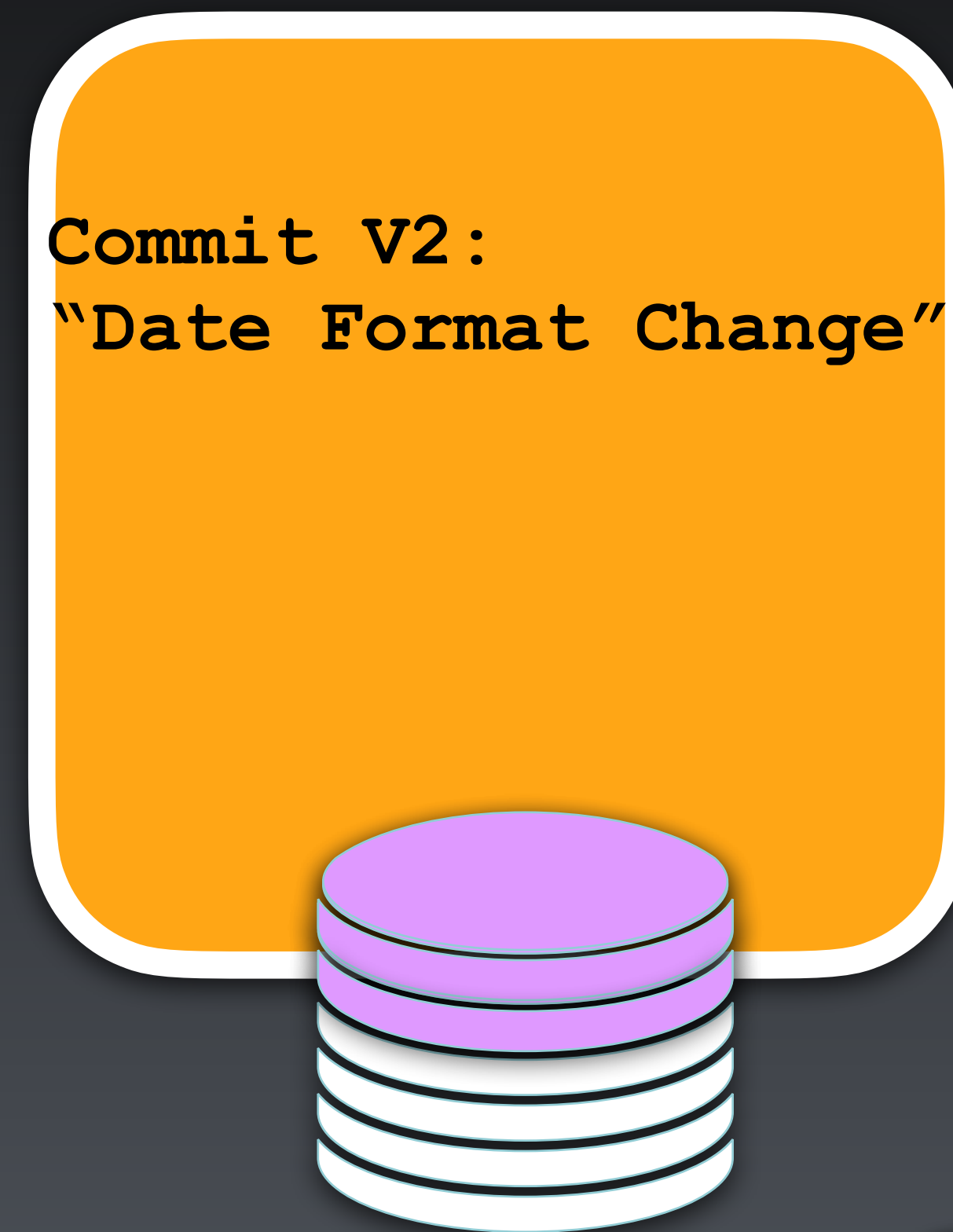
A  B



Commit V2:
"Date Format Change"



Coupling



Coupling



Coupling



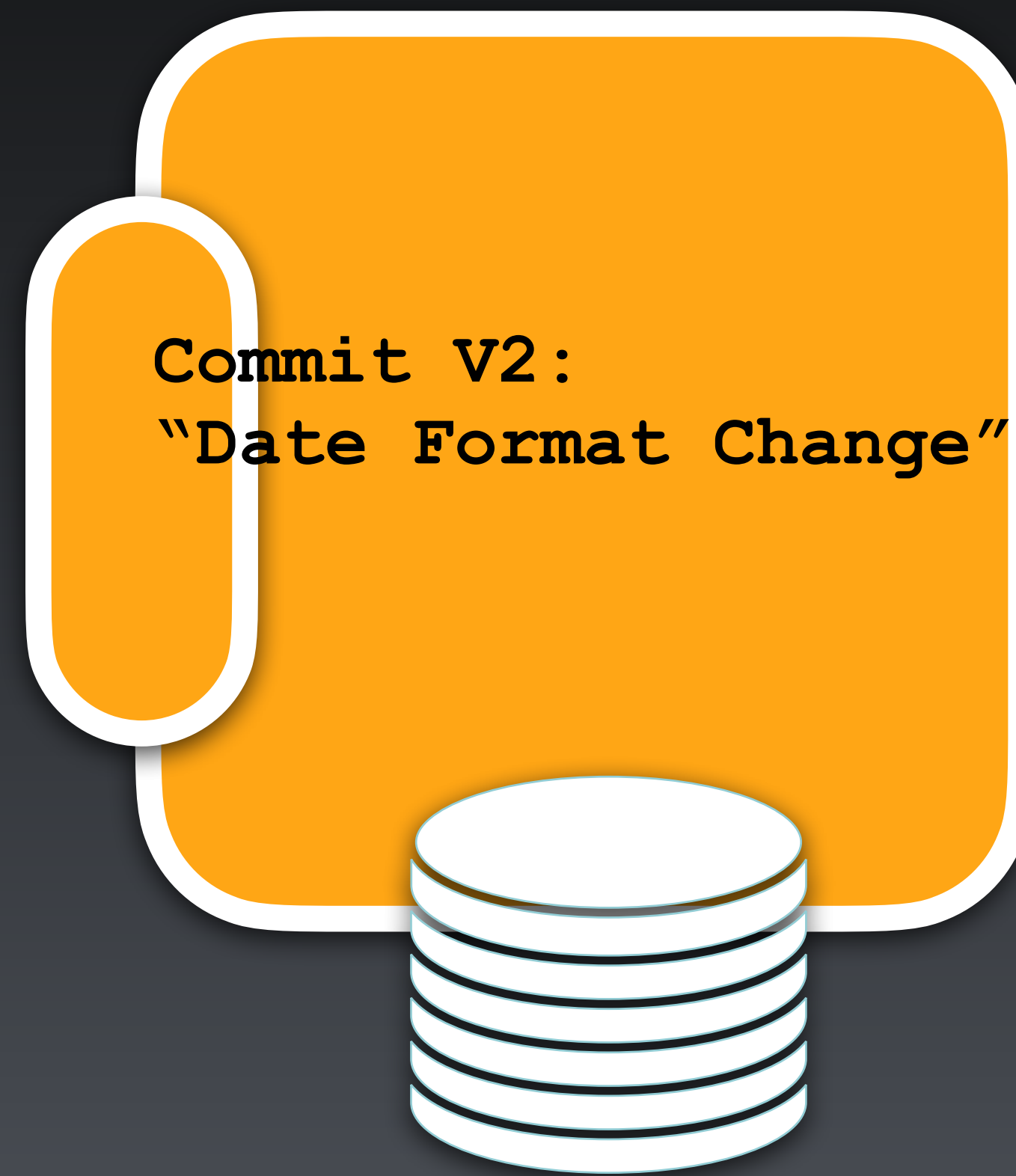
Coupling



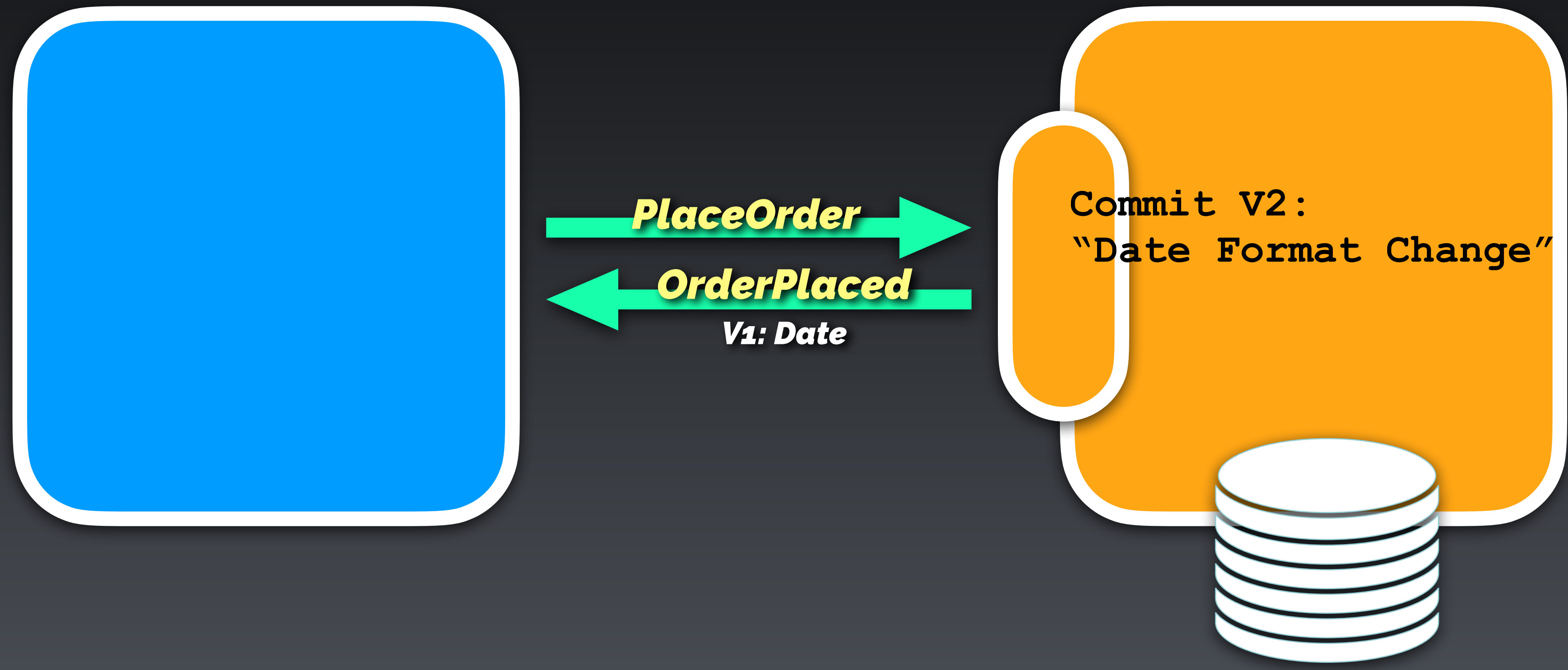
Coupling



Coupling



Coupling



Applying the Guidelines

Managing Complexity

Managing Complexity

Modularity

Cohesion

Separation of Concerns

Abstraction

Coupling

Managing Complexity

Modularity

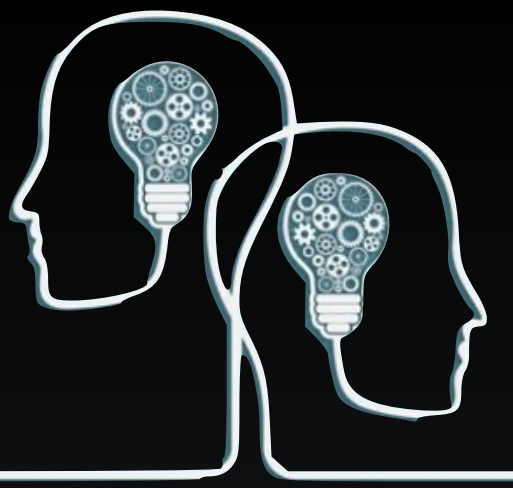
Cohesion

**Just Motherhood
& Apple Pie?**

Separation of Concerns

Abstraction

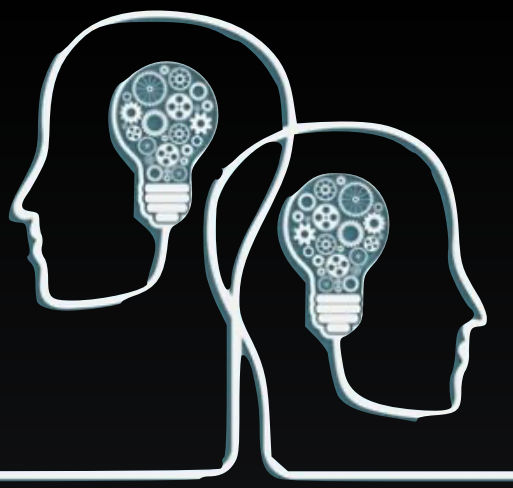
Coupling



```
public class Car {  
    private final Engine engine = new PetrolEngine();  
  
    public void start() {  
        putIntoNeutral();  
        applyBrakes();  
        this.engine.start();  
    }  
  
    private void applyBrakes() {  
    }  
  
    private void putIntoNeutral() {  
    }  
}
```

A  B

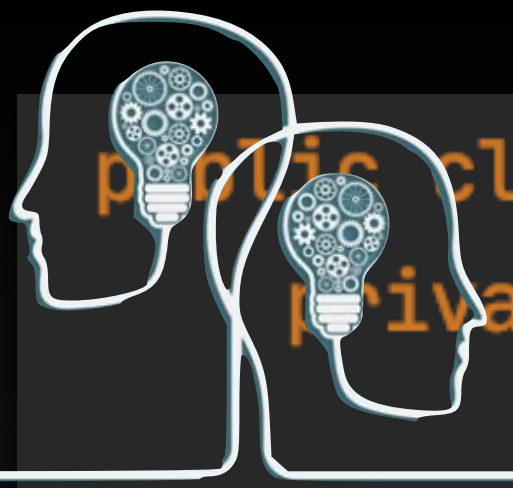
A ——— B



```
public class Car {  
    private final Engine engine = new PetrolEngine();  
  
    public void start() {  
        putIntoNeutral();  
        applyBrakes();  
        this.engine.start();  
    }  
  
    private void applyBrakes() {  
    }  
  
    private void putIntoNeutral() {  
    }  
}
```

A  B

A ——— B

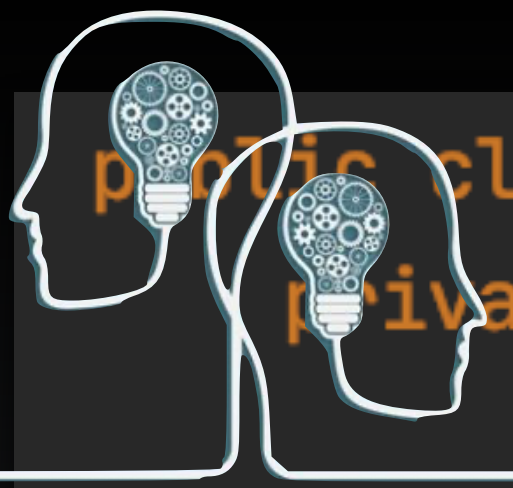


```
public class Car {  
    private final Engine engine = new PetrolEngine();  
  
    public void start() {  
        putIntoNeutral();  
        applyBrakes();  
        this.engine.start();  
    }  
  
    private void applyBrakes() {  
    }  
  
    private void putIntoNeutral() {  
    }  
}
```

A  B
A  B

```
@Test
```

```
public void shouldStartCarEngine() {  
    Car car = new Car();  
  
    car.start();  
  
    // Nothing to assert!!  
}
```

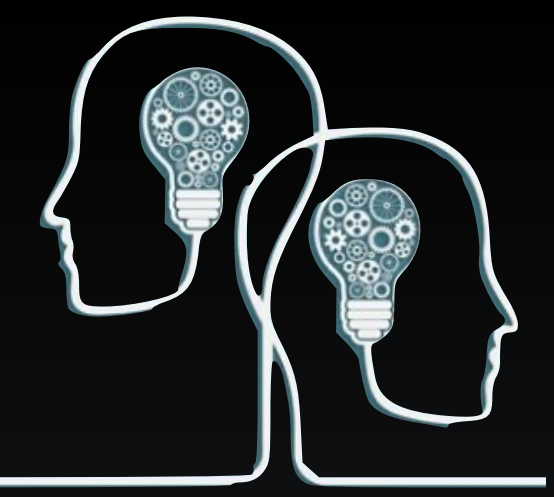


```
public class Car {  
    private final Engine engine = new PetrolEngine();  
  
    public void start() {  
        putIntoNeutral();  
        applyBrakes();  
        this.engine.start();  
    }  
  
    private void applyBrakes() {  
    }  
  
    private void putIntoNeutral() {  
    }  
}
```

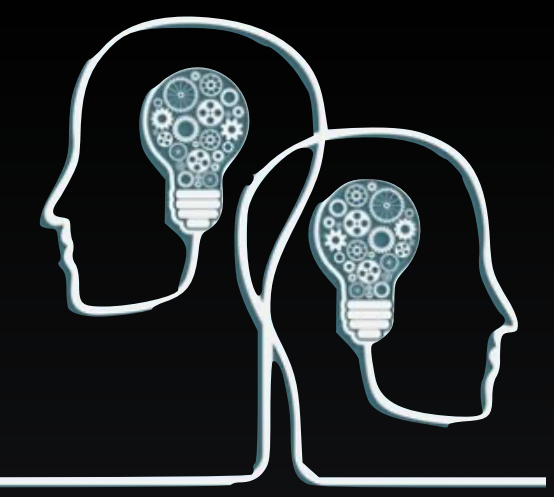
A  B
A  B

```
@Test
```

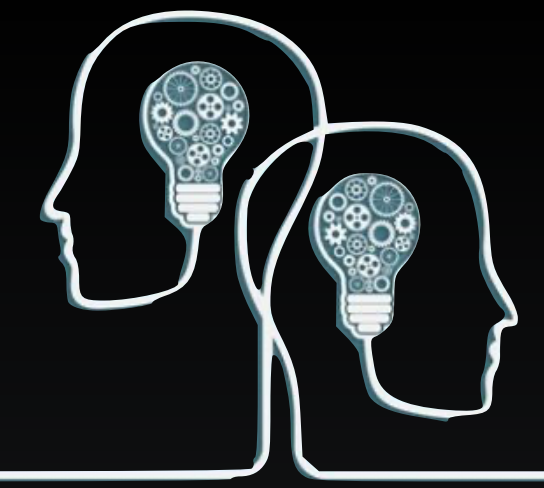
```
public void shouldStartCarEngine() {  
    Car car = new Car();  
  
    car.start();  
  
    // Nothing to assert!!  
}
```



A *loopy* B
A ——— B



A *loopy* B
A ——— B



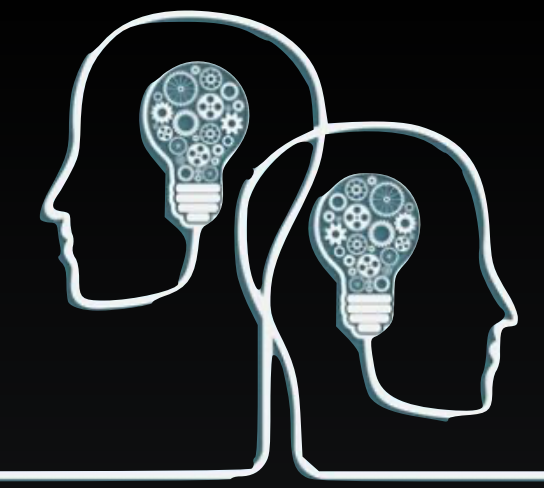
A *loopy* B

A — B

```
@Test
```

```
public void shouldStartBetterCarEngine() {
```

```
}
```



A  B

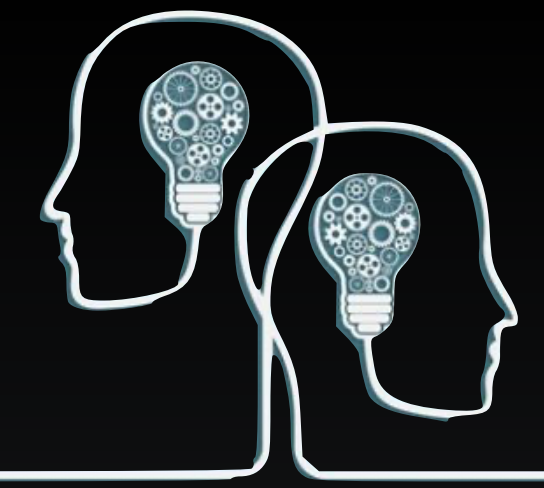
A — B

```
@Test
```

```
public void shouldStartBetterCarEngine() {
```

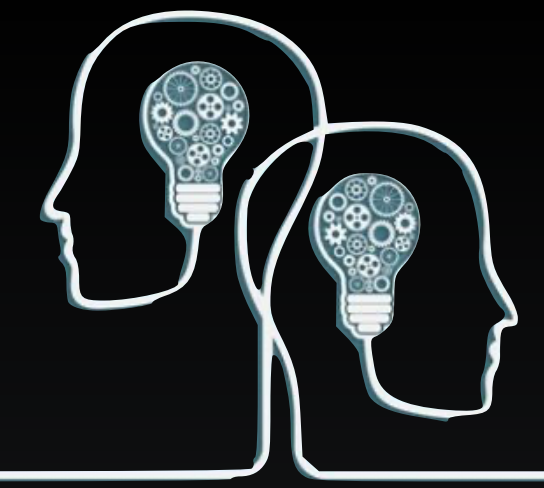
```
    assertTrue(engine.startedSuccessfully());
```

```
}
```

A  B
A ——— B

```
@Test  
public void shouldStartBetterCarEngine() {  
  
    car.start();  
  
    assertTrue(engine.startedSuccessfully());  
}
```



A  B
A ——— B

```
@Test
```

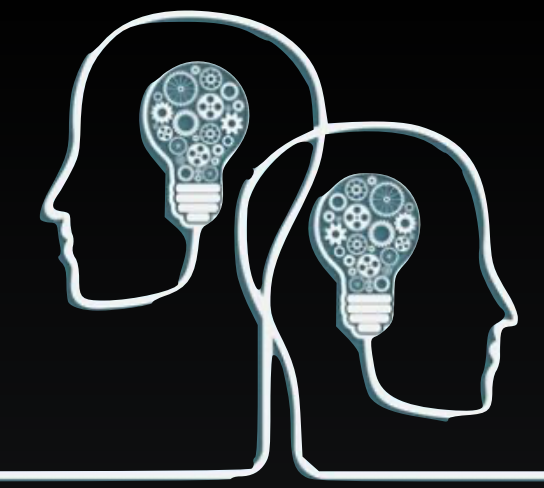
```
public void shouldStartBetterCarEngine() {
```

```
    BetterCar car = new BetterCar(engine);
```

```
    car.start();
```

```
    assertTrue(engine.startedSuccessfully());
```

```
}
```

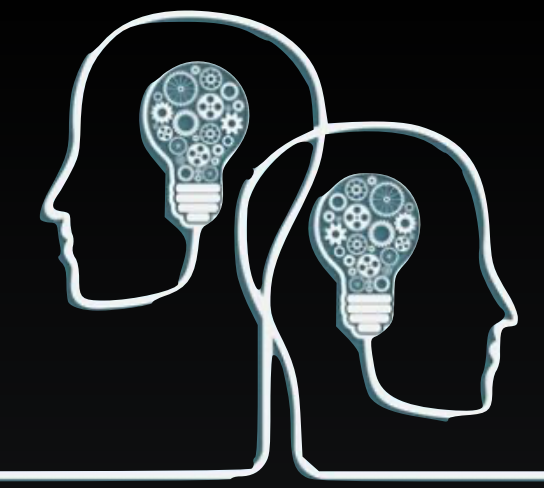


A  B

A ——— B

```
@Test
```

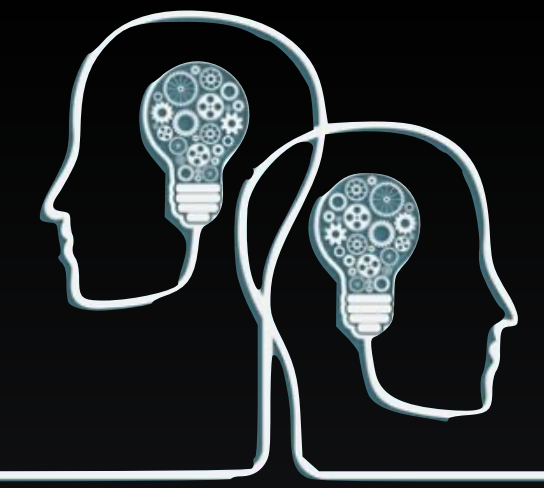
```
public void shouldStartBetterCarEngine() {  
    FakeEngine engine = new FakeEngine();  
    BetterCar car = new BetterCar(engine);  
  
    car.start();  
  
    assertTrue(engine.startedSuccessfully());  
}
```



```
public class FakeEngine implements Engine {  
    private boolean started = false;  
  
    @Override  
    public void start() {  
        started = true;  
    }  
  
    public boolean startedSuccessfully() {  
        return started;  
    }  
}
```

A  B
A ————— B

```
{  
;  
);  
y());
```

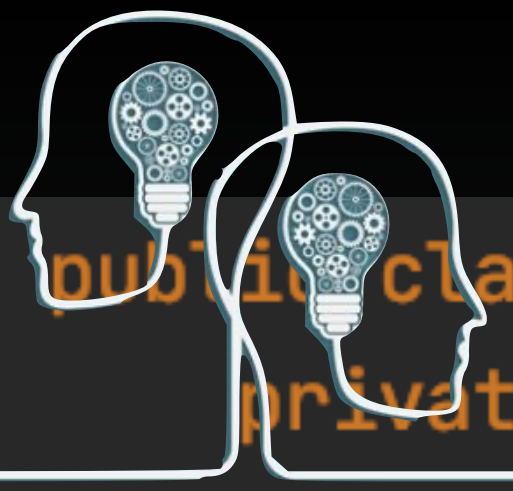


A  B

A ——— B

```
@Test
```

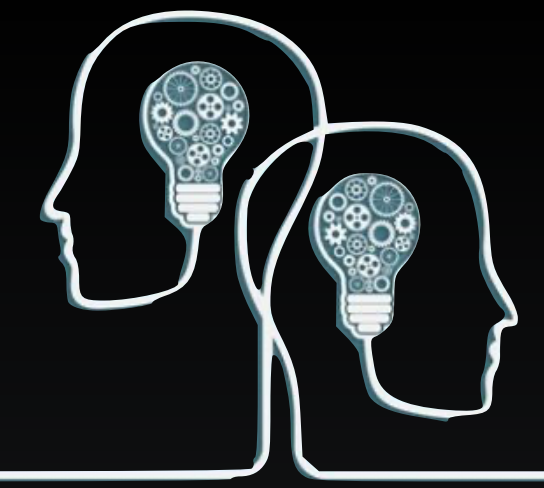
```
public void shouldStartBetterCarEngine() {  
    FakeEngine engine = new FakeEngine();  
    BetterCar car = new BetterCar(engine);  
  
    car.start();  
  
    assertTrue(engine.startedSuccessfully());  
}
```



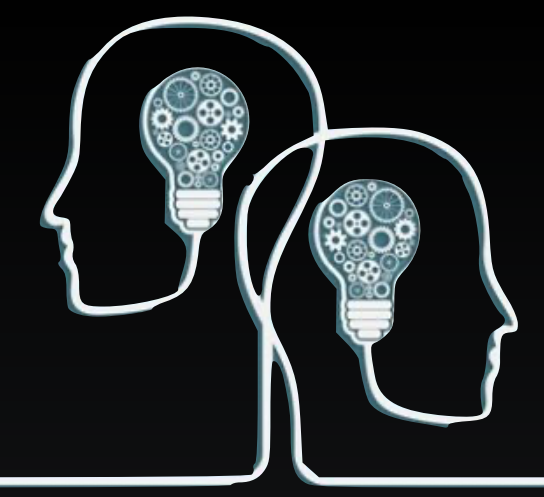
A  B
A ——— B

```
public class BetterCar {  
    private final Engine engine;  
  
    public BetterCar(Engine engine) {  
        this.engine = engine;  
    }  
  
    public void start() {  
        putIntoNeutral();  
        applyBrakes();  
        this.engine.start();  
    }  
  
    private void applyBrakes() {  
    }  
  
    private void putIntoNeutral() {  
    }  
}
```

```
@Test  
public void shouldStartBetterCarEngine() {  
    FakeEngine engine = new FakeEngine();  
    BetterCar car = new BetterCar(engine);  
  
    car.start();  
  
    assertTrue(engine.startedSuccessfully());  
}
```



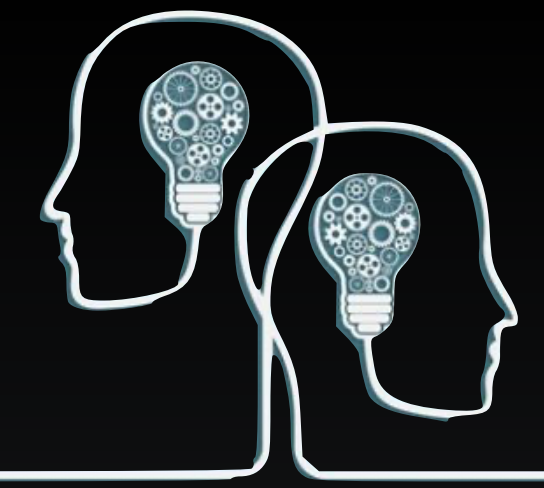
A *loopy* B
A ——— B



```
public class BetterCar {  
    private final Engine engine;  
  
    public BetterCar(Engine engine) {  
        this.engine = engine;  
    }  
  
    public void start() {  
        putIntoNeutral();  
        applyBrakes();  
        this.engine.start();  
    }  
  
    private void applyBrakes() {  
    }  
  
    private void putIntoNeutral() {  
    }  
}
```

A  B

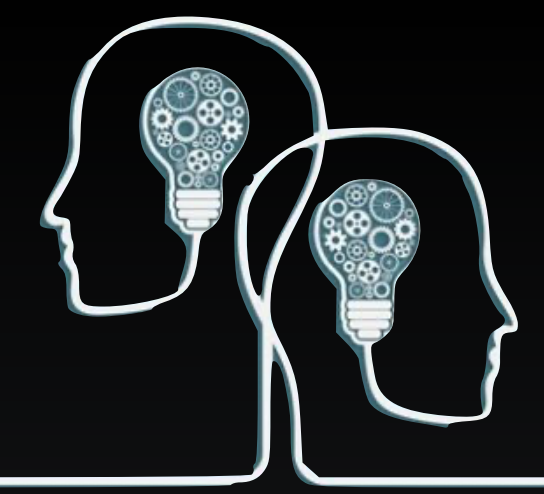
A — B



```
public class BetterCar {  
    private final Engine engine;  
  
    public BetterCar(Engine engine) {  
        this.engine = engine;  
    }  
  
    public void start() {  
        putIntoNeutral();  
        applyBrakes();  
        this.engine.start();  
    }  
  
    private void applyBrakes() {  
    }  
  
    private void putIntoNeutral() {  
    }  
}
```

A  B

A ——— B



```
public class BetterCar {  
    private final Engine engine;
```

```
public BetterCar(Engine engine) {  
    this.engine = engine;  
}
```

```
public void start() {  
    putIntoNeutral();  
    applyBrakes();
```

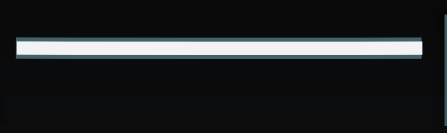
```
public class Car {  
    private final Engine engine = new PetrolEngine();
```

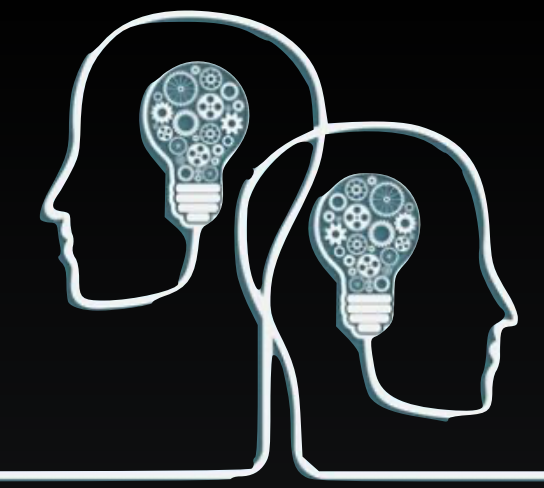
```
private void applyBrakes() {  
}
```

```
private void putIntoNeutral() {  
}
```

```
}
```

A  B

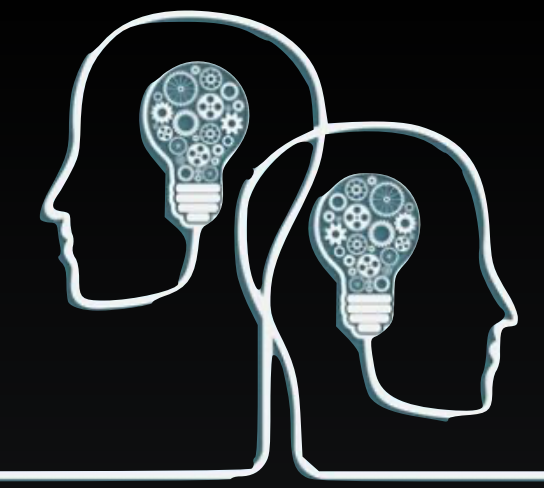
A  B



```
public class BetterCar {  
    private final Engine engine;  
  
    public BetterCar(Engine engine) {  
        this.engine = engine;  
    }  
  
    public void start() {  
        putIntoNeutral();  
        applyBrakes();  
        this.engine.start();  
    }  
  
    private void applyBrakes() {  
    }  
  
    private void putIntoNeutral() {  
    }  
}
```

A  B

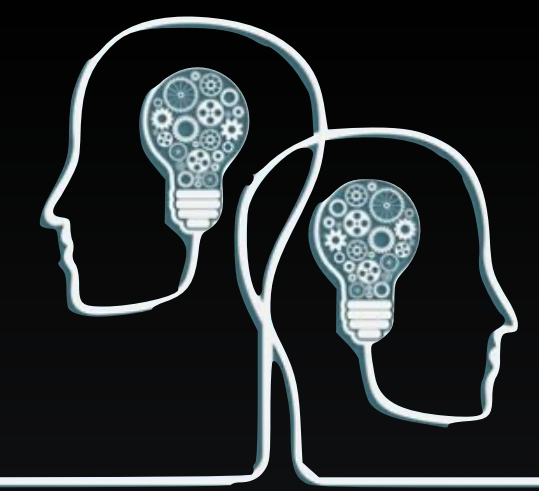
A ——— B



```
public class BetterCar {  
    private final Engine engine;  
  
    public BetterCar(Engine engine) {  
        this.engine = engine;  
    }  
  
    public void start() {  
        putIntoNeutral();  
        applyBrakes();  
        this.engine.start();  
    }  
  
    private void applyBrakes() {  
    }  
  
    private void putIntoNeutral() {  
    }  
}
```

A  B

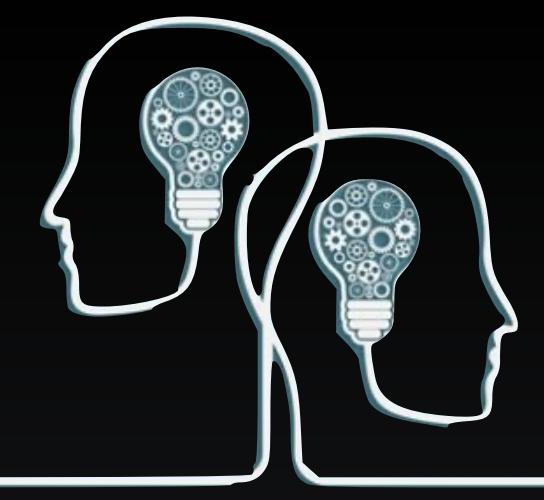
A ——— B



```
public class BetterCar {  
    private final Engine engine;  
  
    public BetterCar(Engine engine) {  
        this.engine = engine;  
    }  
}
```



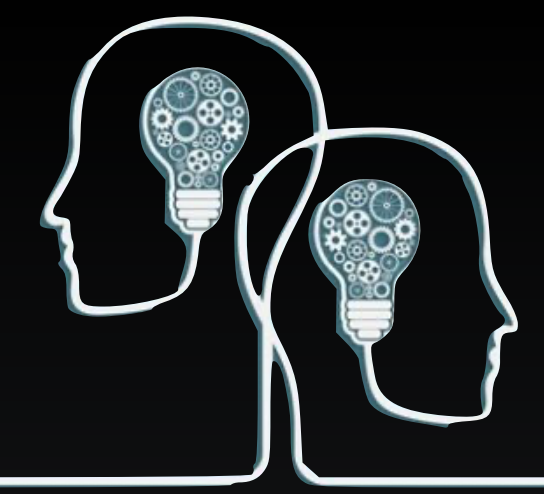
```
public void createCars() {  
  
    BetterCar petrolCar = new BetterCar(new PetrolEngine());  
  
}
```



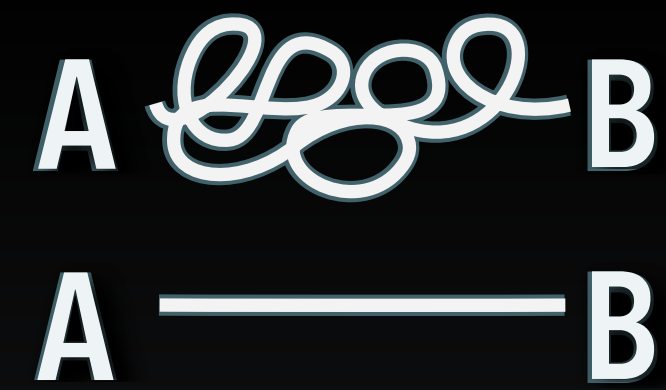
```
public class BetterCar {  
    private final Engine engine;  
  
    public BetterCar(Engine engine) {  
        this.engine = engine;  
    }  
}
```

A  B
A ——— B

```
public void createCars() {  
  
    BetterCar petrolCar = new BetterCar(new PetrolEngine());  
  
    BetterCar electricCar = new BetterCar(new ElectricEngine());  
  
}
```



```
public class BetterCar {  
    private final Engine engine;  
  
    public BetterCar(Engine engine) {  
        this.engine = engine;  
    }  
}
```



```
public void createCars() {  
  
    BetterCar petrolCar = new BetterCar(new PetrolEngine());  
  
    BetterCar electricCar = new BetterCar(new ElectricEngine());  
  
    BetterCar jetCar = new BetterCar(new JetEngine());  
  
}
```


The Tools of Our Trade!

Optimise for Learning

Iteration

Feedback

Incremental

Experimental

Empirical

Optimise to Manage Complexity

Modularity

Cohesion

Separation of Concerns

Abstraction

Coupling

- ***These Things Matter for 1 Reason***

- ***These Things Matter for 1 Reason***
- ***They Allow us to Change Our Software***

Embrace Change!

Embrace Change!

JS Frameworks Change All the Time!

Requirements Change

Technologies Change

Businesses Change

People Change

Markets Change

Plans Change

Customers Change

Our Designs Change

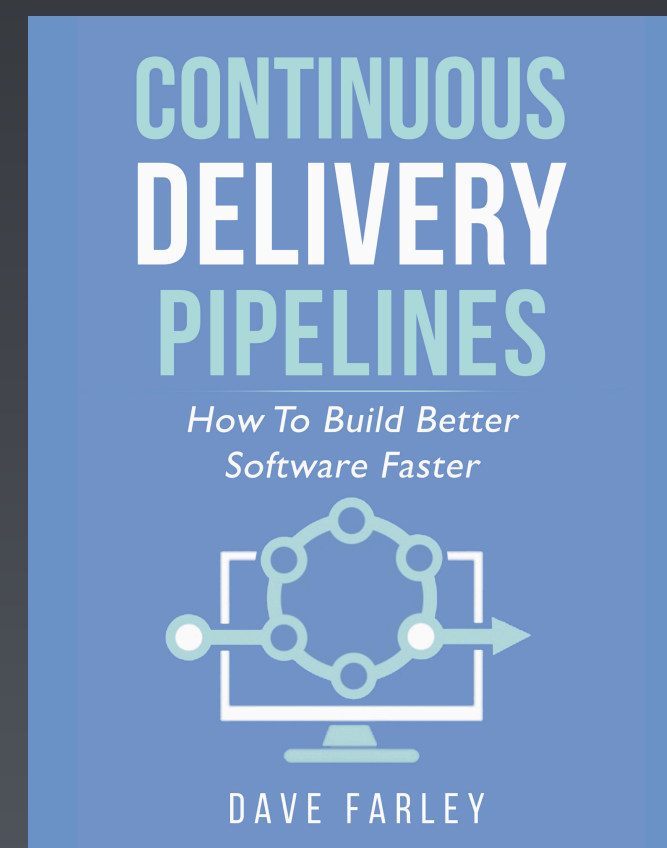
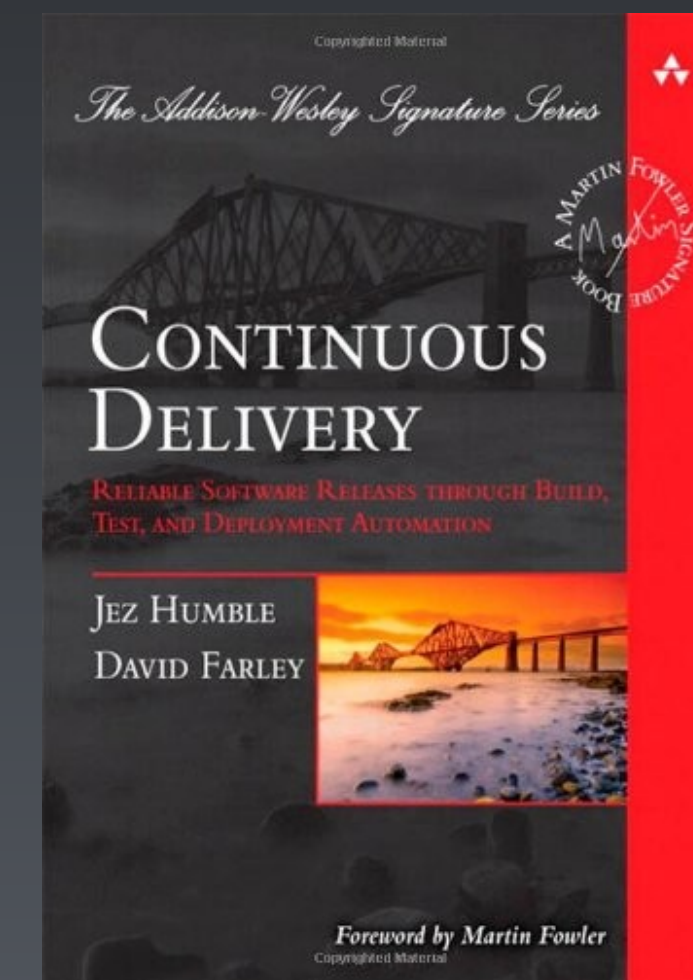
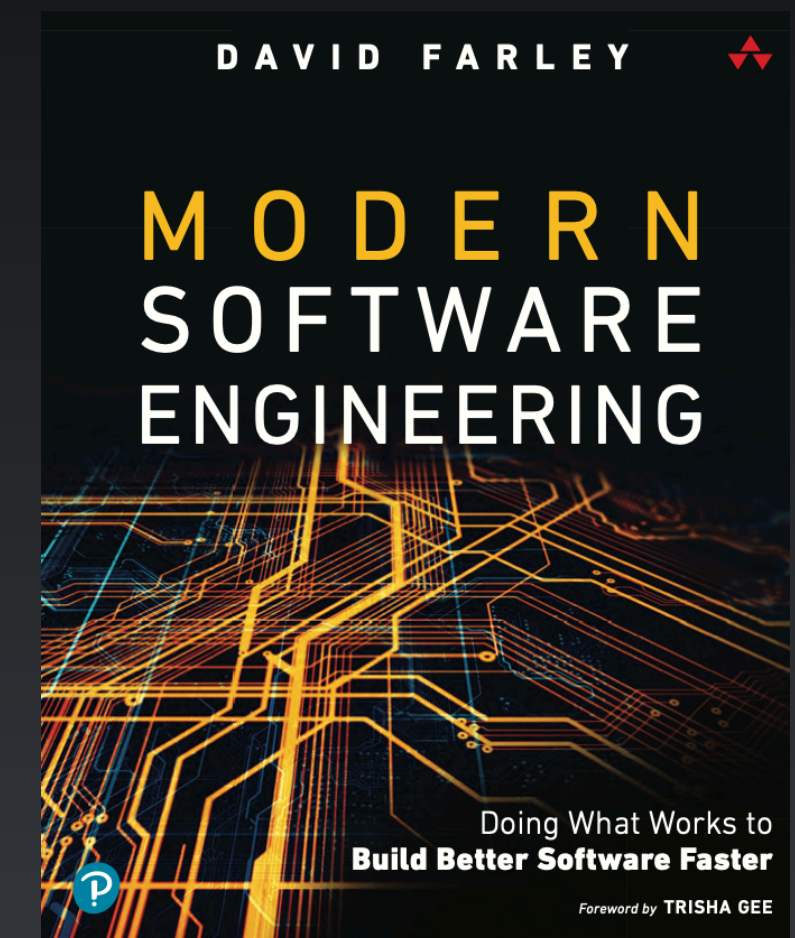
Our Understanding Changes

**The Quality of a System
is Defined by
Our Ability to Change it!**

Q&A



<http://www.continuous-delivery.co.uk>



Dave Farley



<https://www.davefarley.net>



@davefarley77



<https://bit.ly/CDonYT>