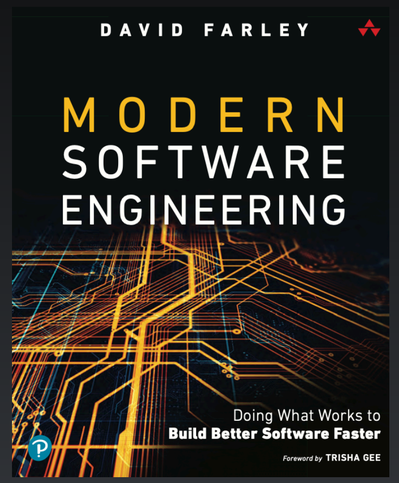
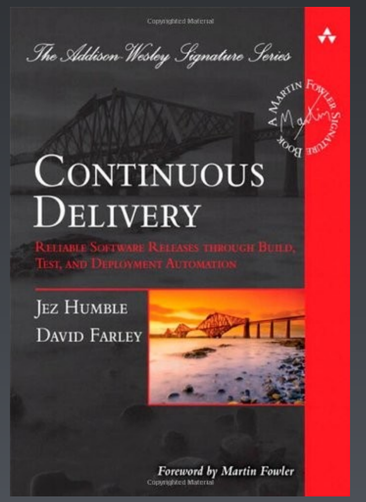
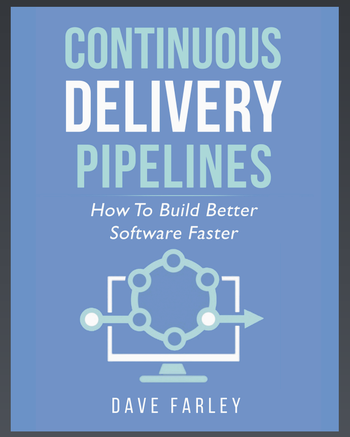
Engineering for Software

How to Amplify Creativity







Dave Farley

https://www.davefarley.net

@davefarley77

https://bit.ly/CDonYT

Continuous Delivery ltd

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


Continuous
Delivery ltd

How Do You Think of Your Work?

Continuous Delivery ltd

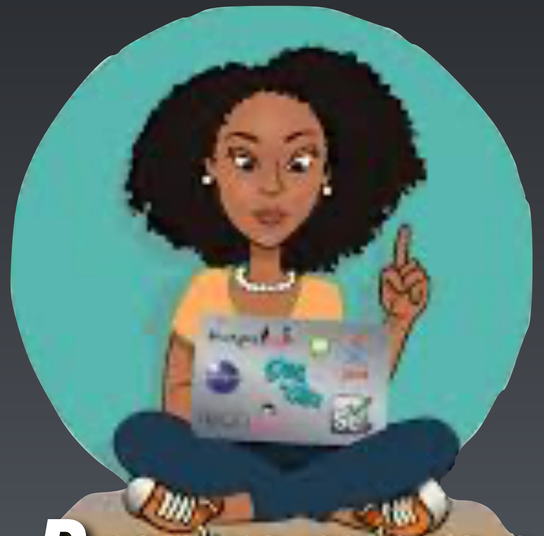


Programmer

Continuous Delivery ltd



Software Developer



Programmer

Continuous Delivery ltd



Programmer



Craftsperson

Continuous Delivery ltd



Programmer



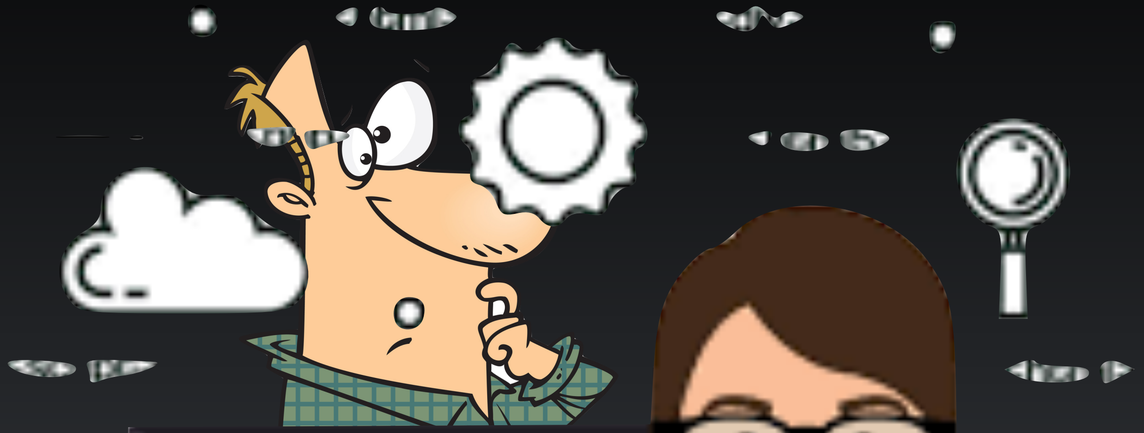
Meme Wrangler

loper

Continuous Delivery ltd



Meme Wrangler





Programmer

Software Engineer

Developer

Continuous Delivery ltd



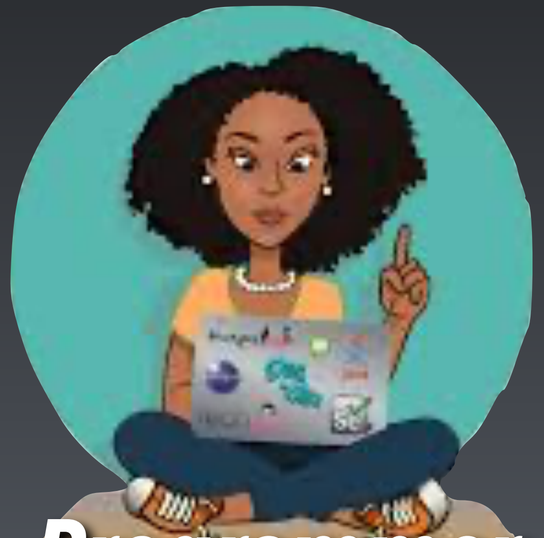
Meme Wrangler



Craftsperson



Software Engineer



Programmer



Software Developer

Continuous Delivery ltd

Who Cares?

Continuous Delivery ltd

Who Cares?

Some of These Things are More Than Only Labels

Continuous Delivery ltd


Continuous
Delivery ltd

- ***What is Engineering***

Continuous Delivery ltd

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

Continuous Delivery ltd

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

Continuous Delivery ltd

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

Continuous Delivery ltd

What is Engineering

Continuous Delivery ltd


Continuous
Delivery ltd

- *Engineering in Other Disciplines is:*

  "The Stuff That Works"

Continuous Delivery ltd

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

Continuous Delivery ltd

The Stuff That Works

Continuous Delivery ltd

The Stuff That Works

- ***What are the Ideas that Increase Our Chances of Success?***

Continuous Delivery ltd

The Stuff That Works

- ***What are the Ideas that Increase Our Chances of Success?***
- ***Whatever the Nature of the Software***

Continuous Delivery ltd

The Stuff That Works

- ***What are the Ideas that Increase Our Chances of Success?***
- ***Whatever the Nature of the Software***
- ***What Ideas Represent the:***

"Foundations of Our Discipline"

Continuous Delivery ltd

All Engineering is not the same!

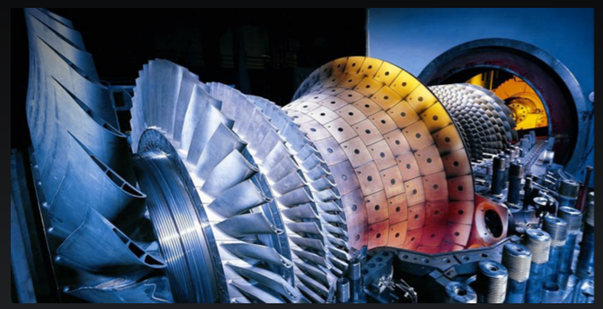
Continuous Delivery ltd

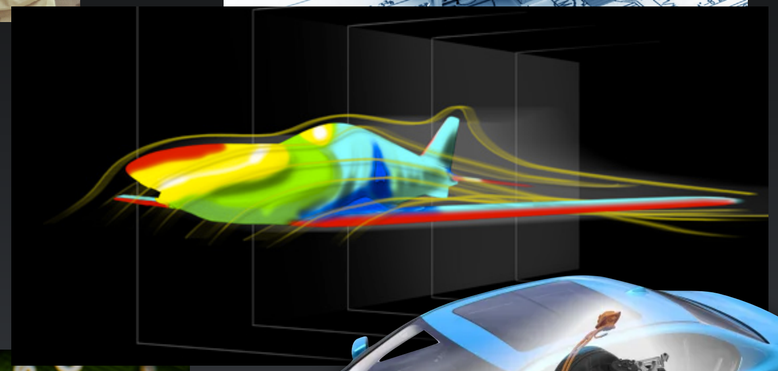


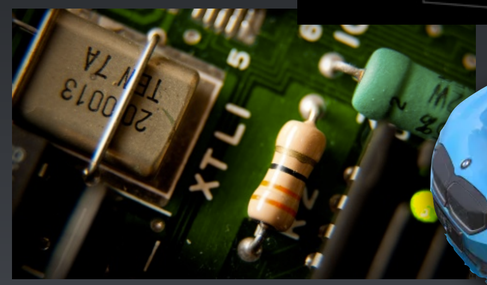
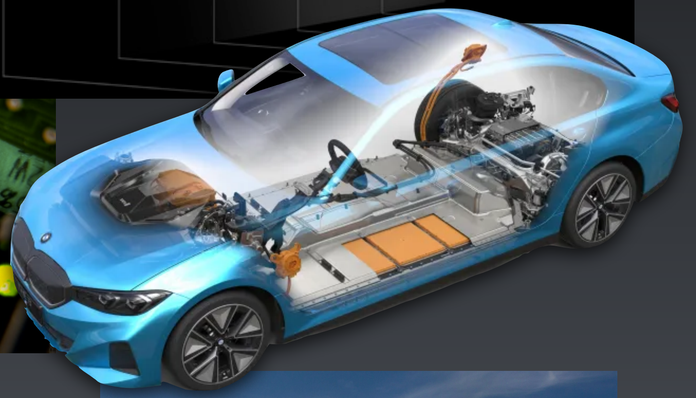


All Engineering is not the same!







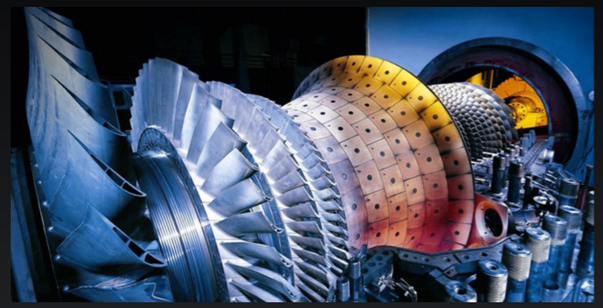



Continuous Delivery ltd

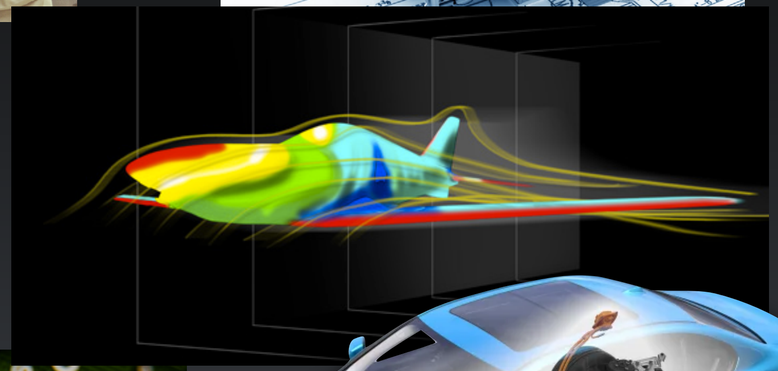
Learning From Other Engineering Disciplines





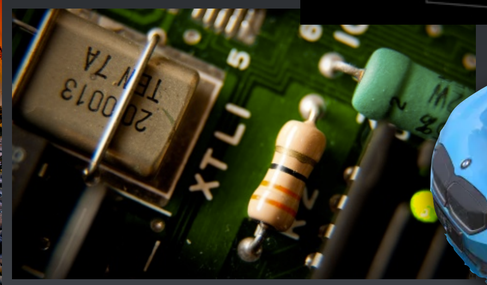


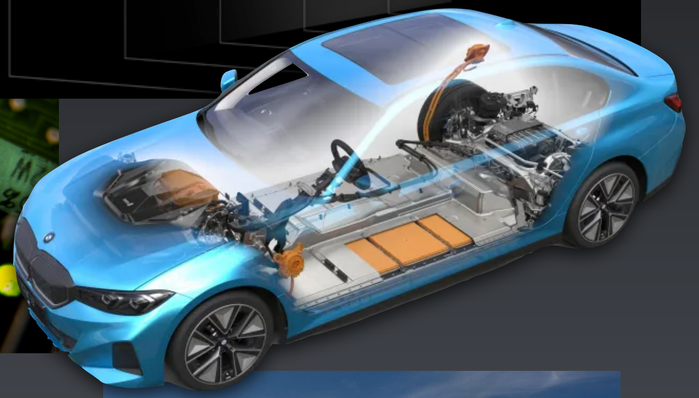












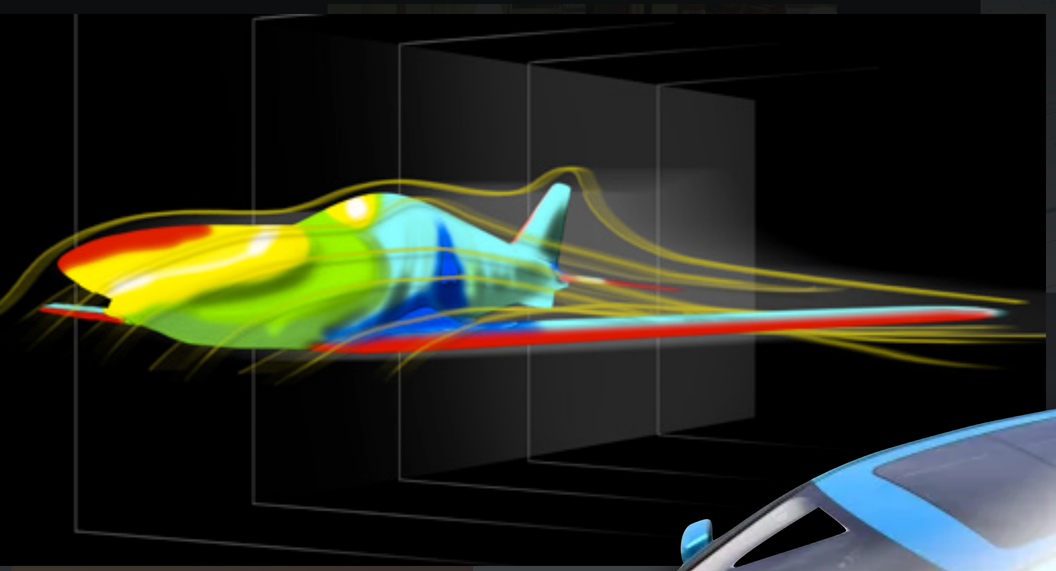




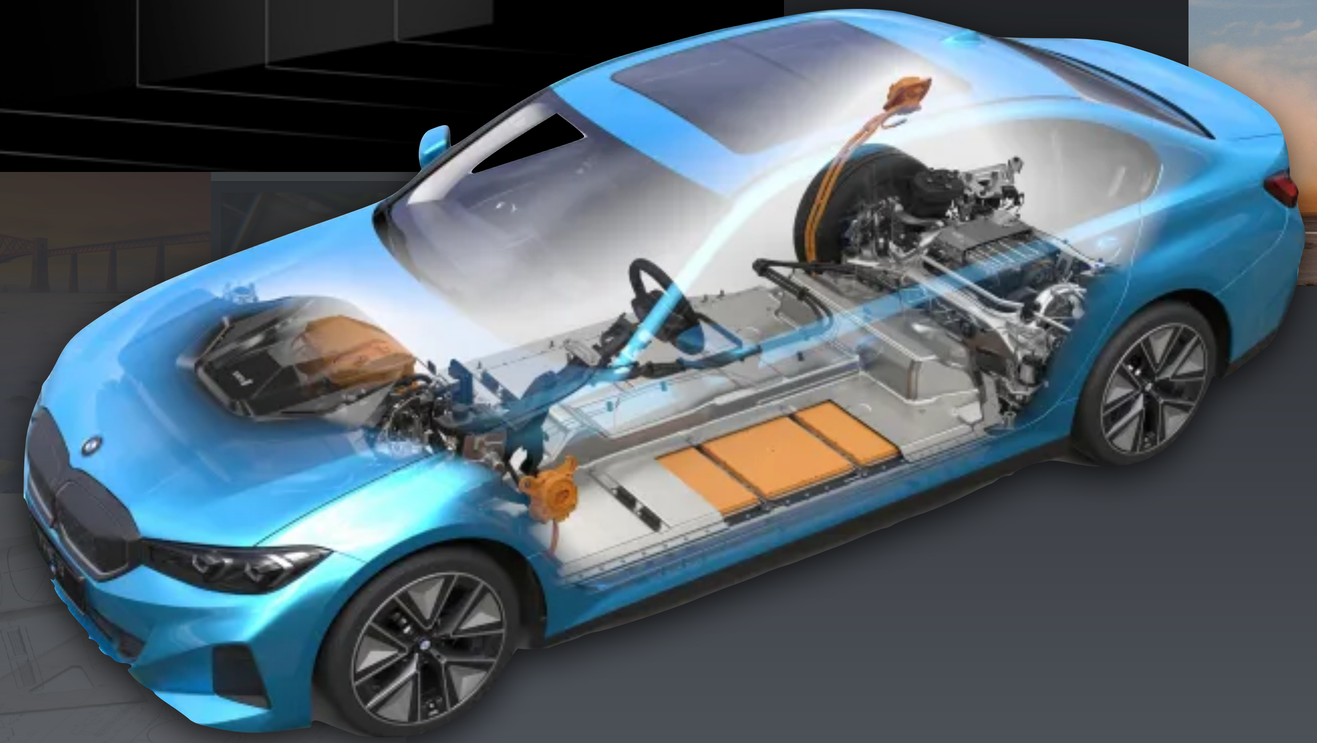


Continuous Delivery ltd

Learning From Other Engineering Disciplines

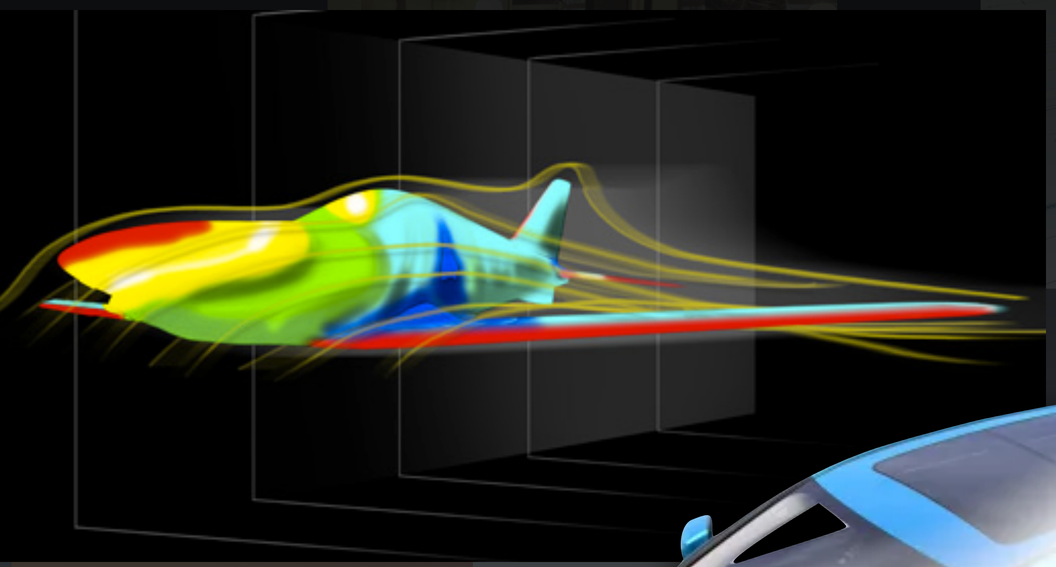




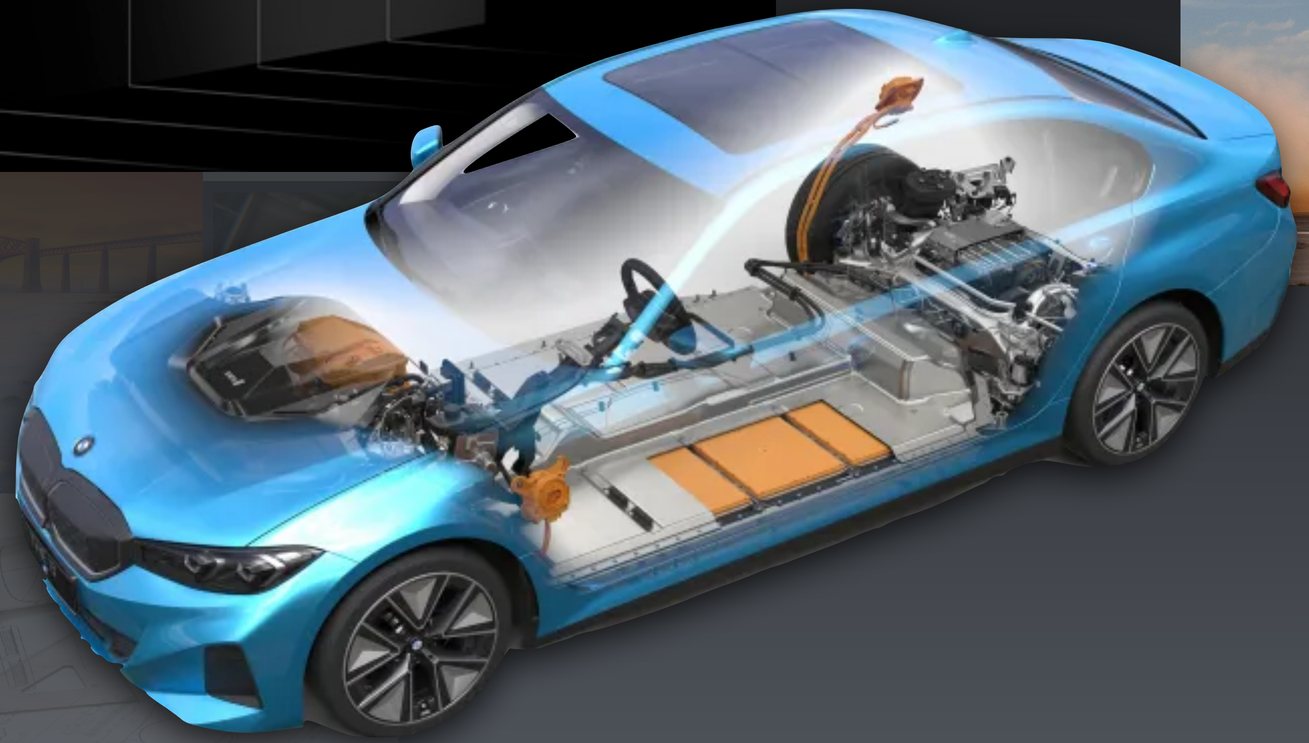


Continuous Delivery ltd

Learning From Other Engineering Disciplines



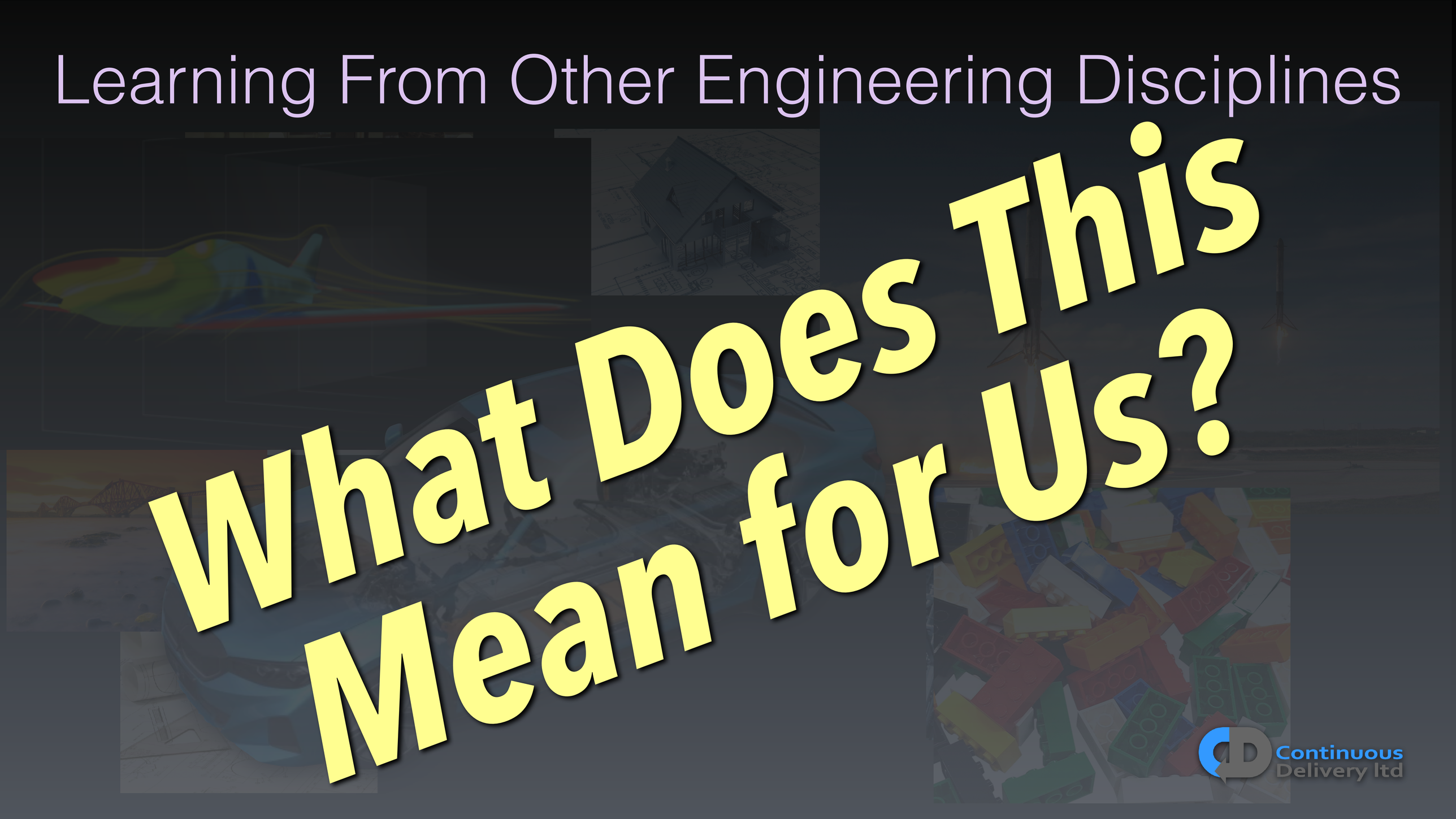






Continuous Delivery ltd

Learning From Other Engineering Disciplines



What Does This Mean for Us?

Continuous Delivery ltd

What is ‘Engineering’?

Continuous Delivery ltd

What is ‘Engineering’?

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

(Dave Farley - Just Now!)

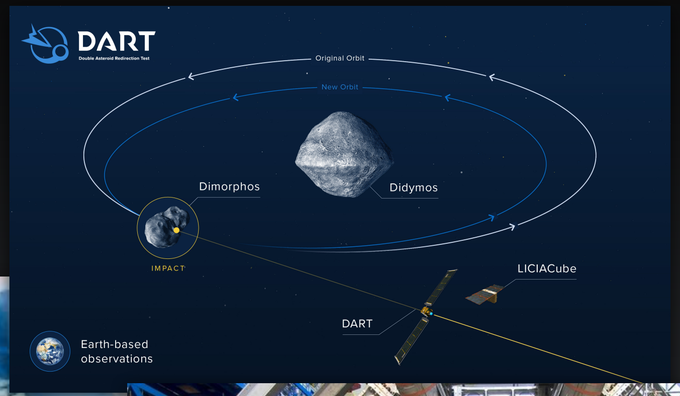
Continuous Delivery ltd

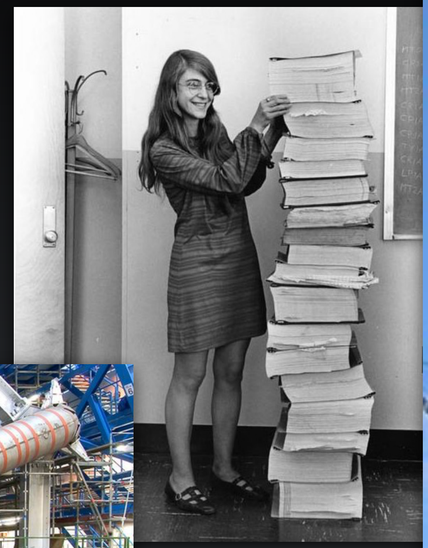
To Put It More Simply…

Continuous Delivery ltd

To Put It More Simply…

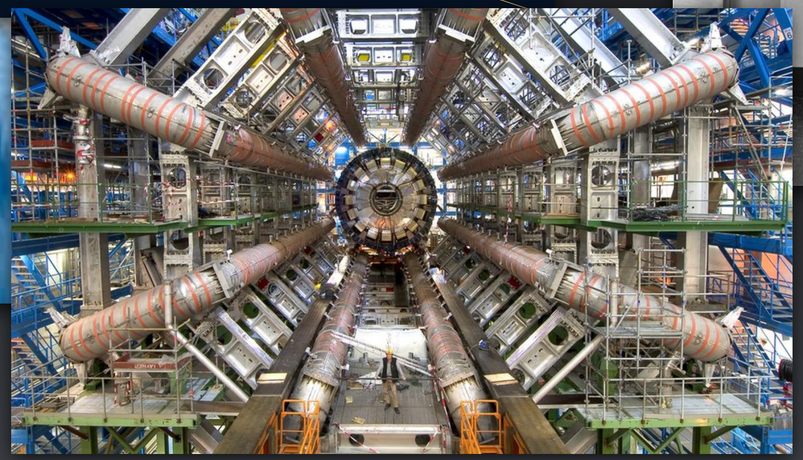












Engineering

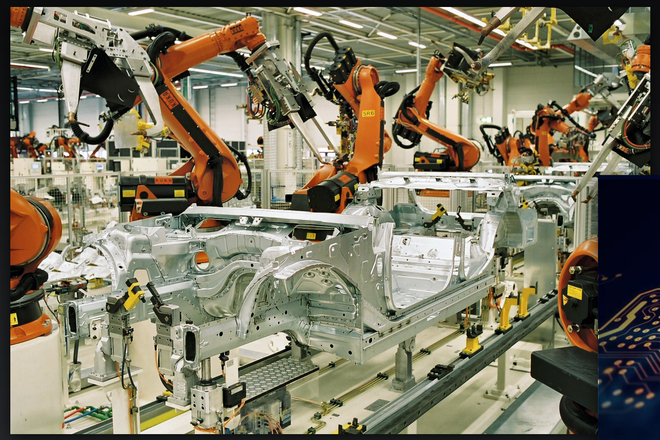
is How We Solve the Hard Problems!

Continuous Delivery ltd

To Put It More Simply…

Continuous Delivery ltd

To Put It More Simply…









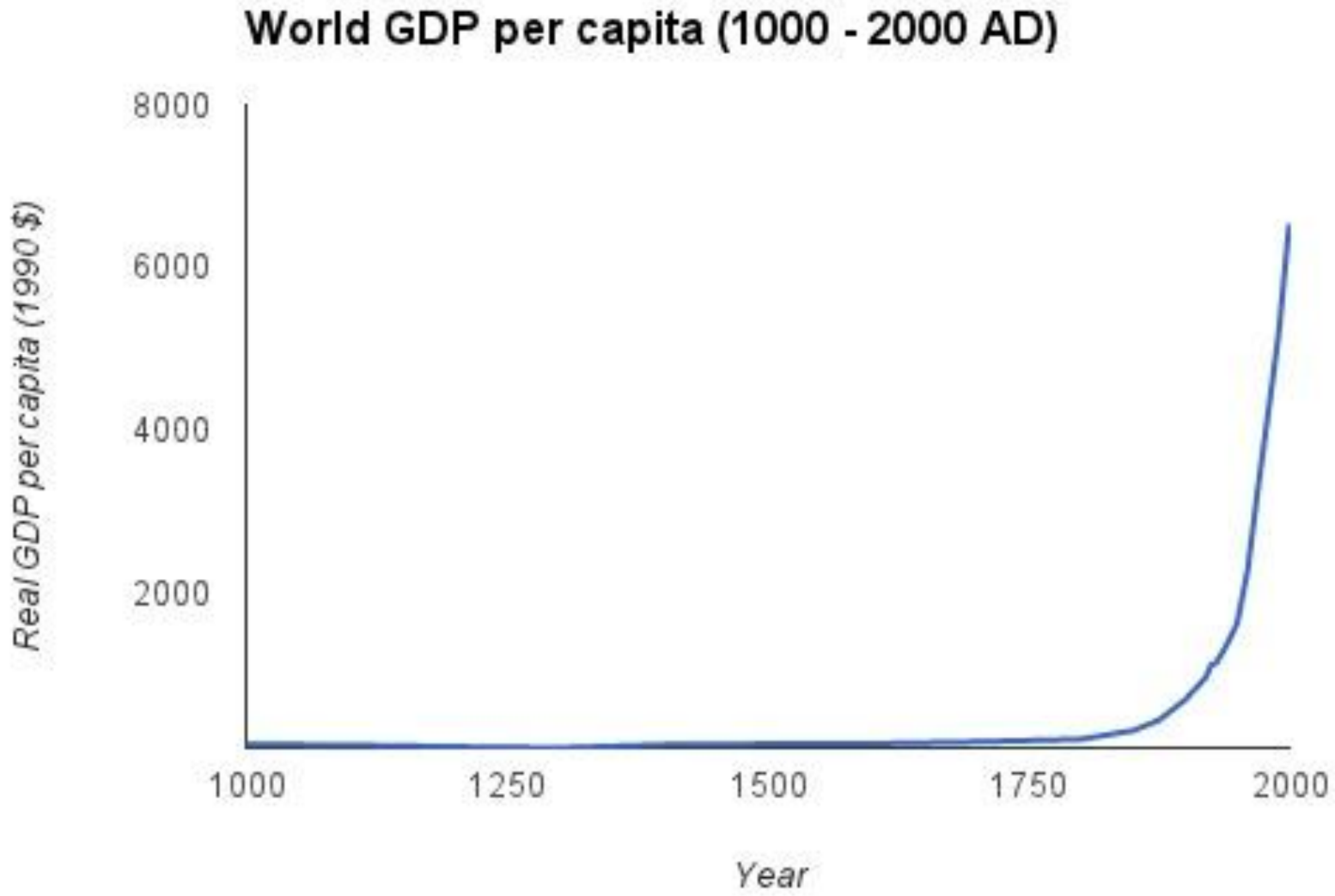
Engineering

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

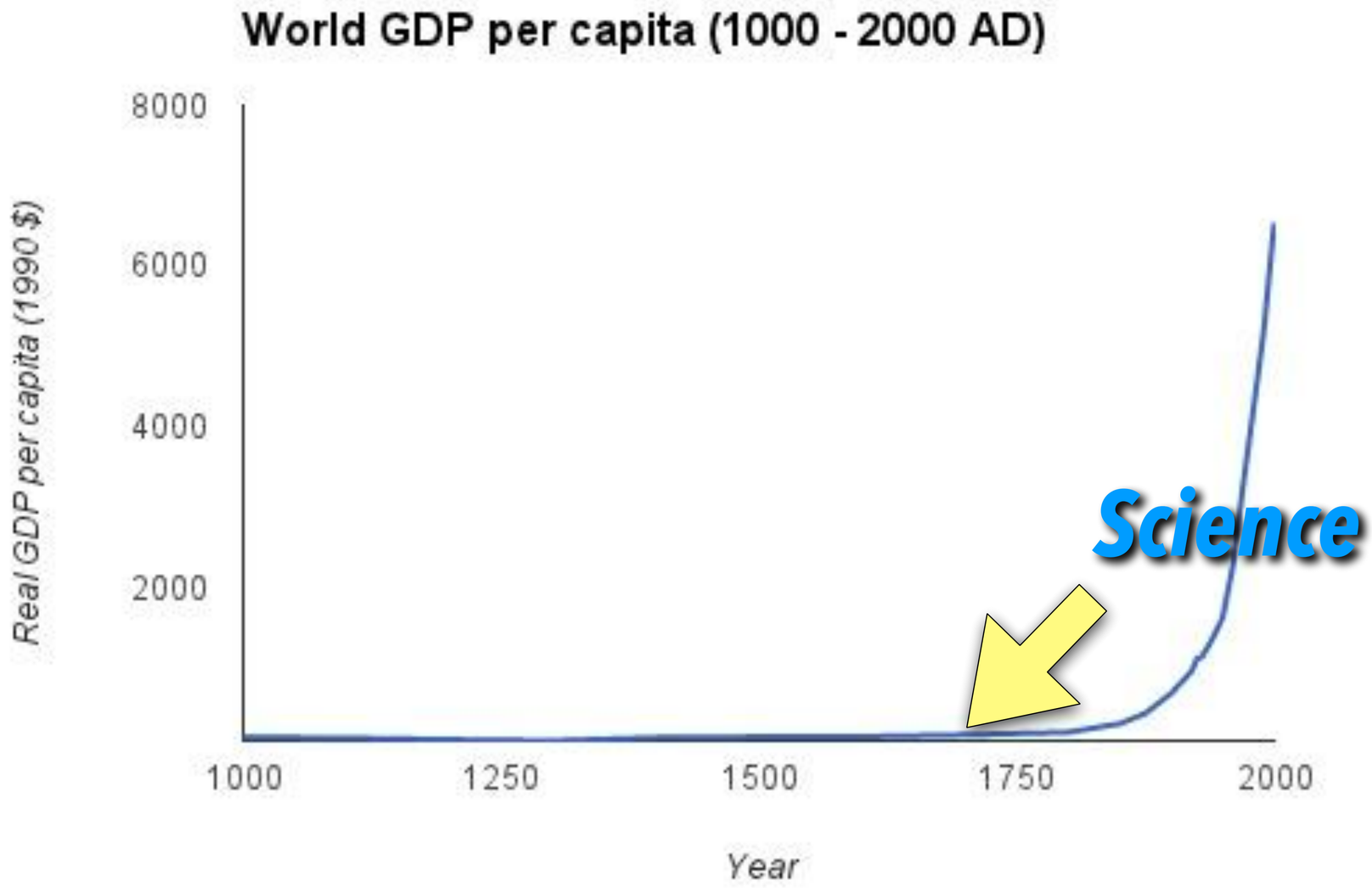
Continuous Delivery ltd

The Impact of Science & Engineering

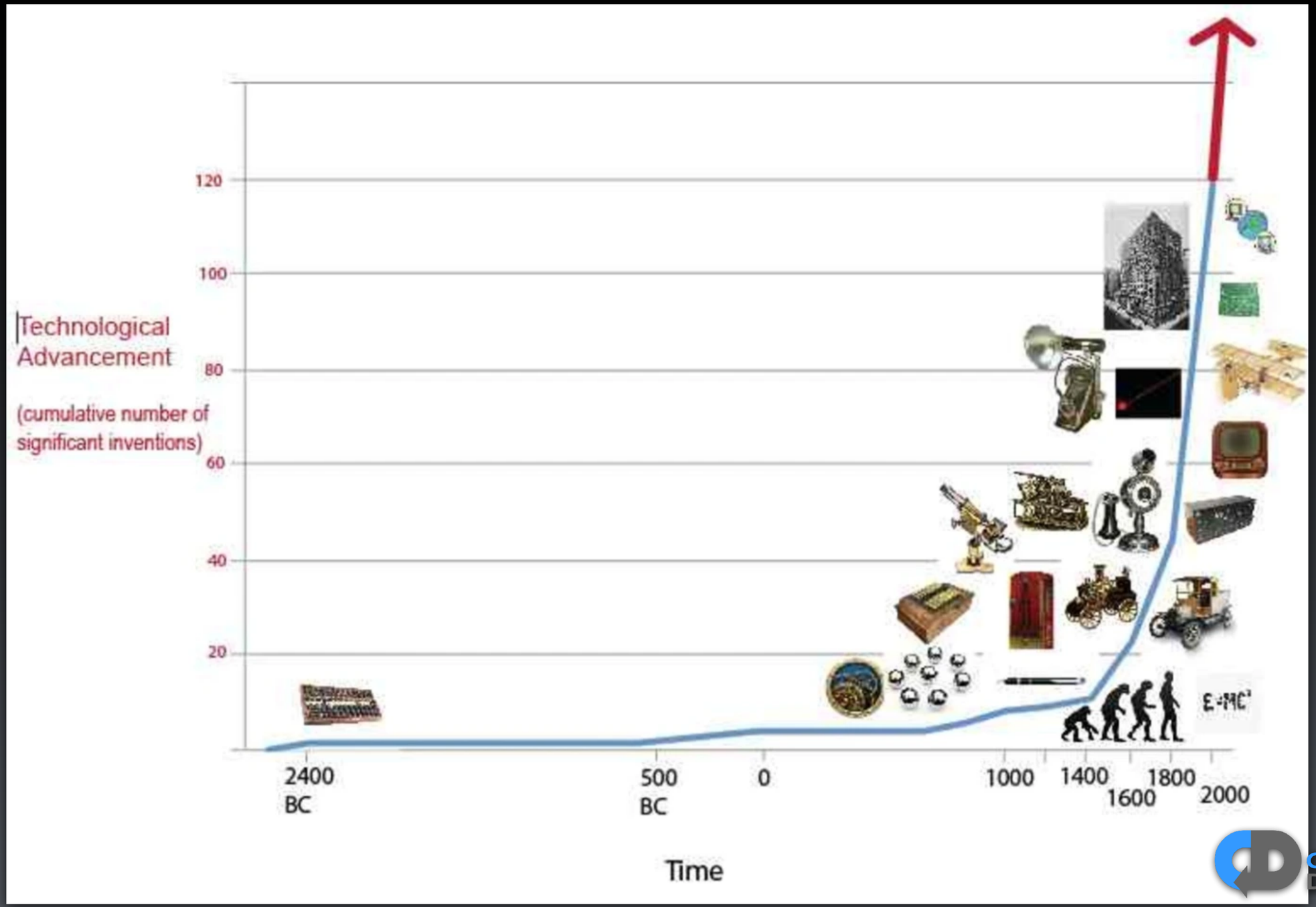
Continuous Delivery ltd



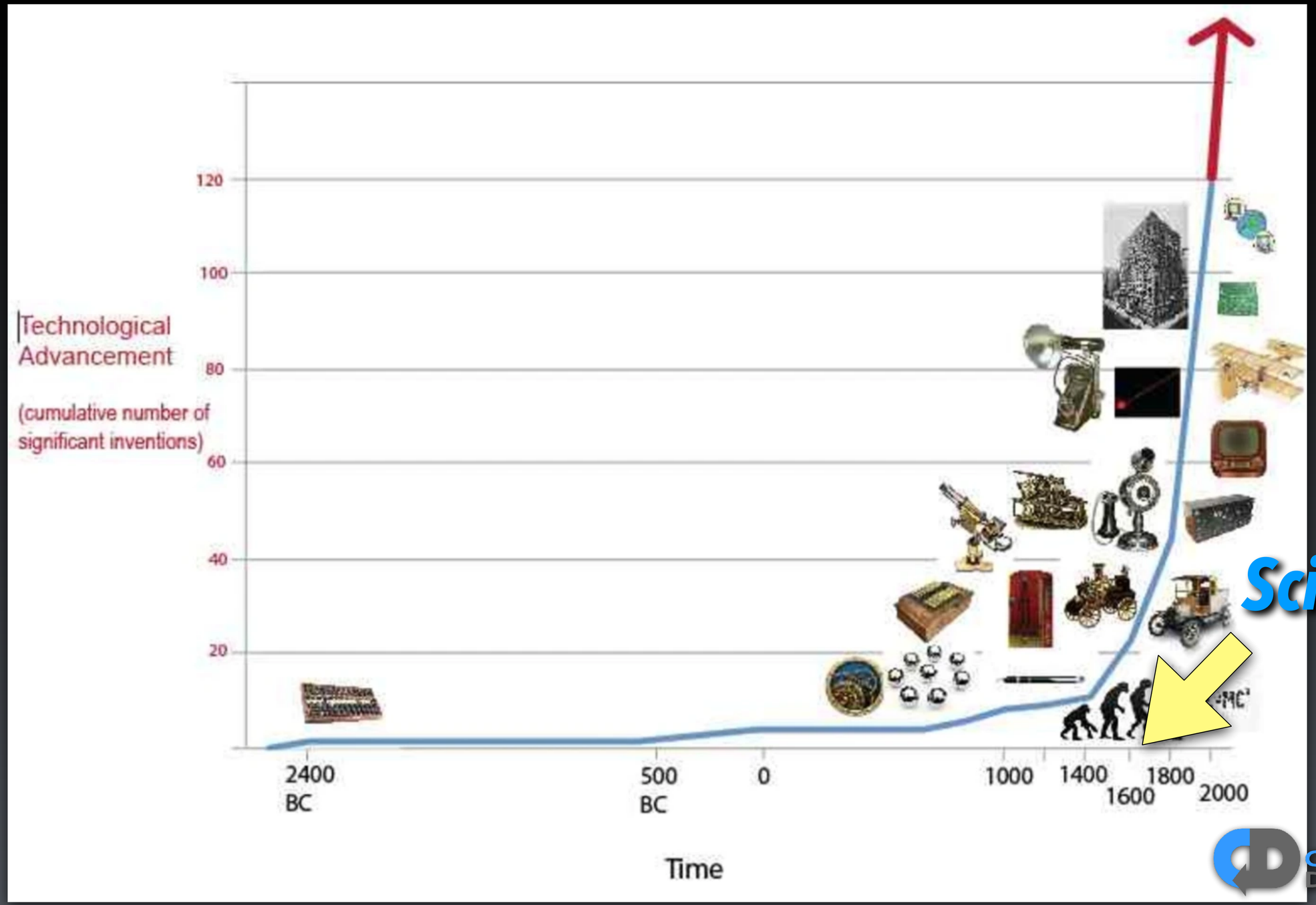
Continuous Delivery ltd



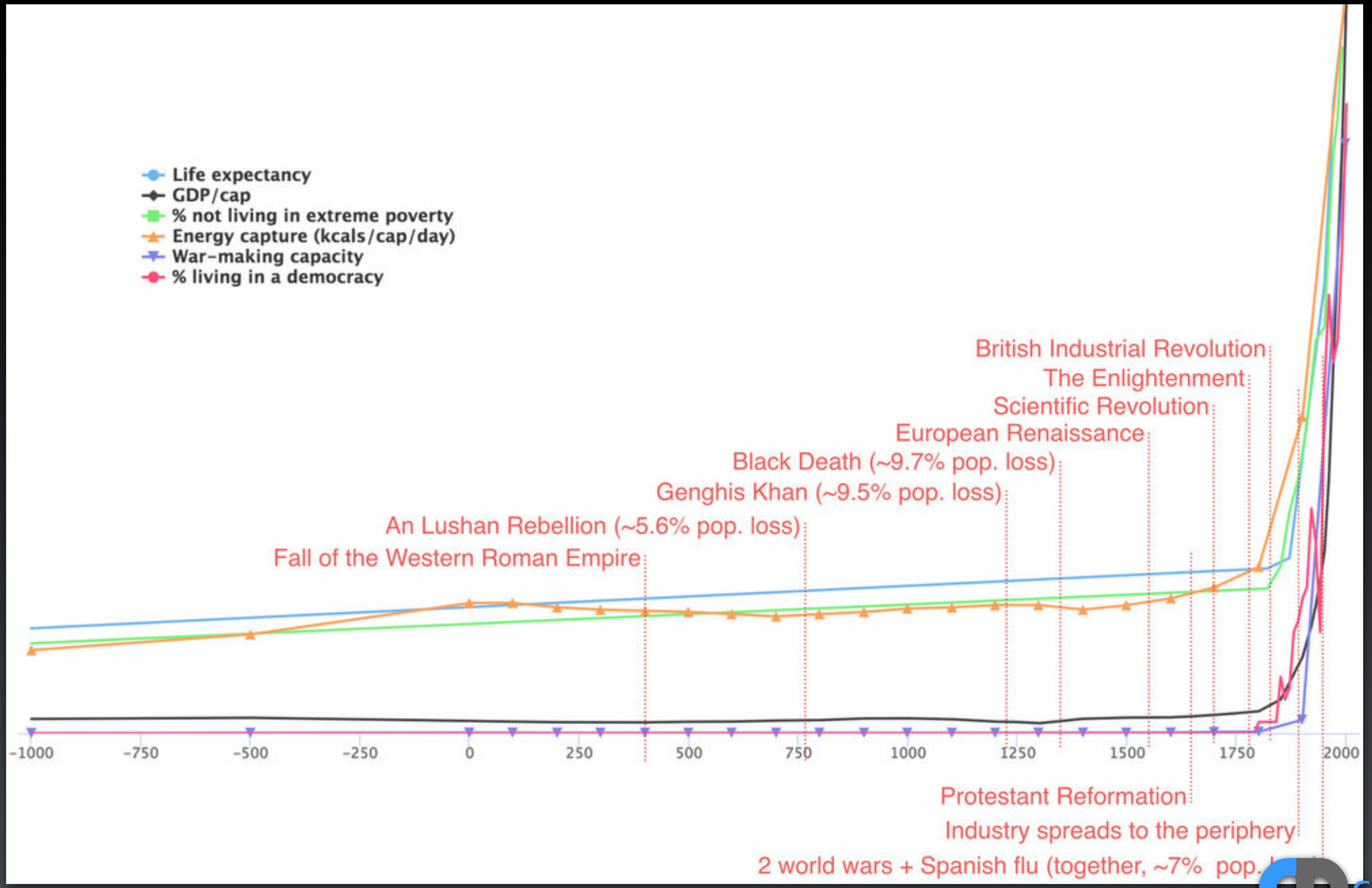
Continuous Delivery ltd



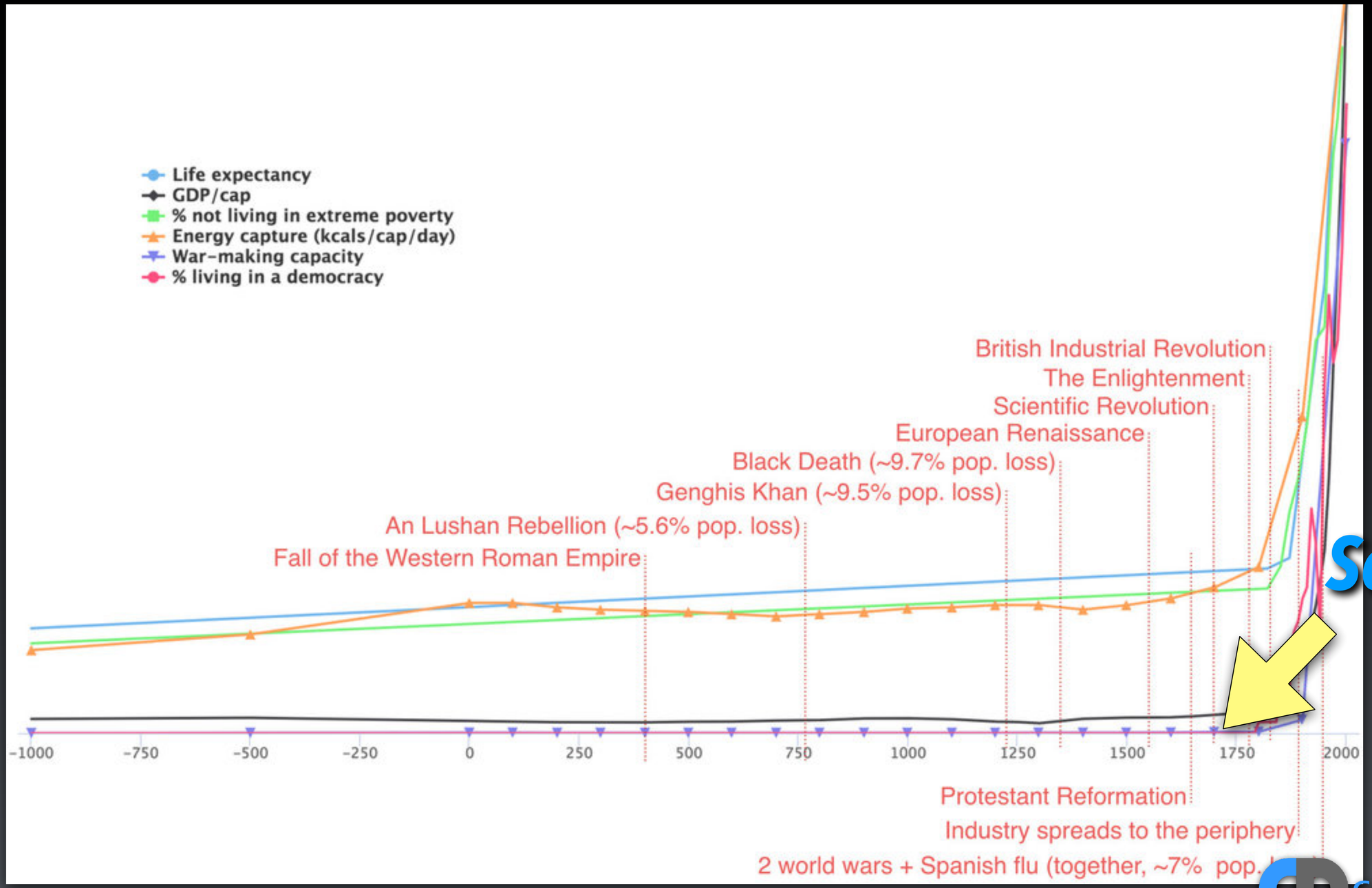
Continuous Delivery ltd



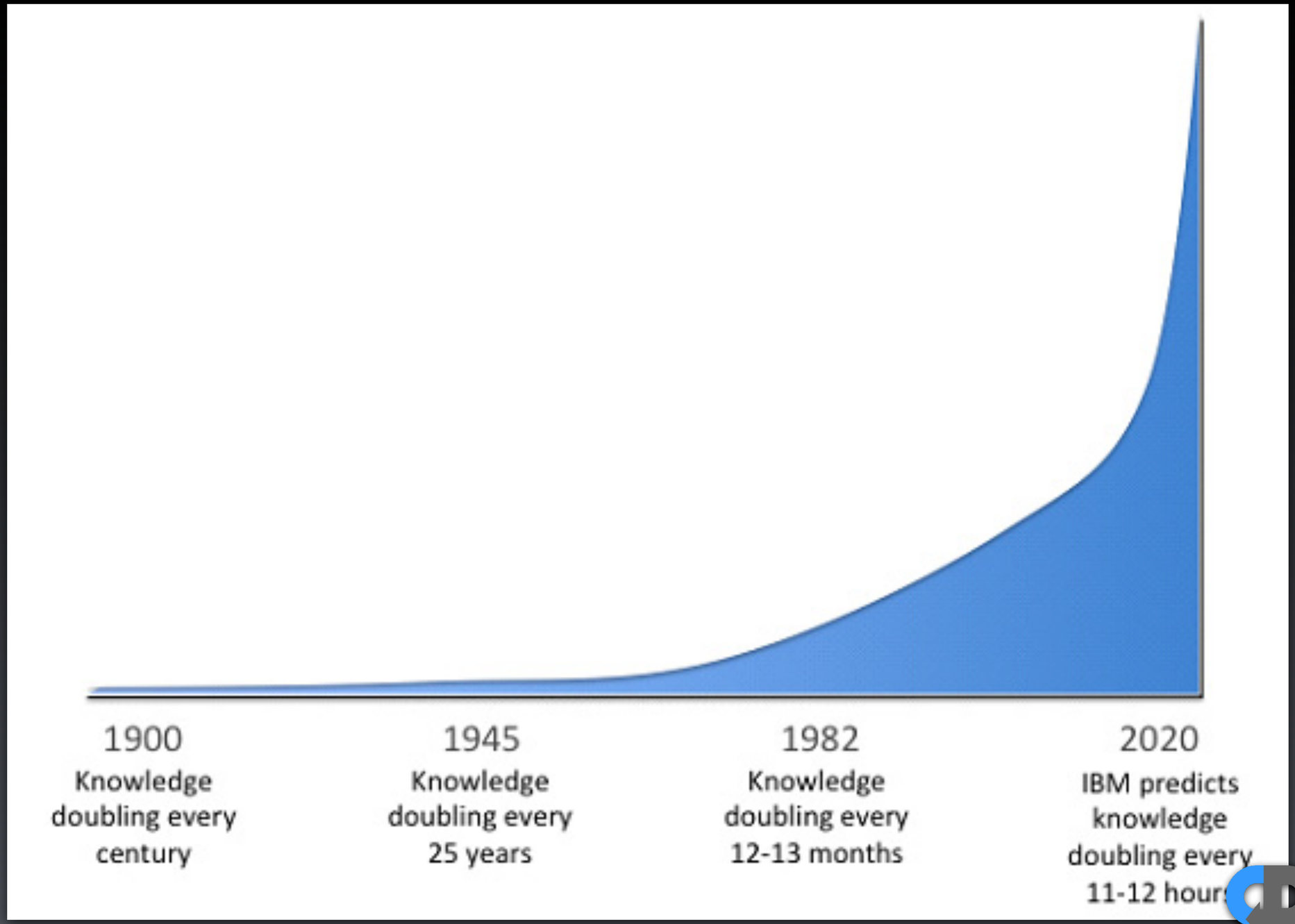
Continuous Delivery ltd



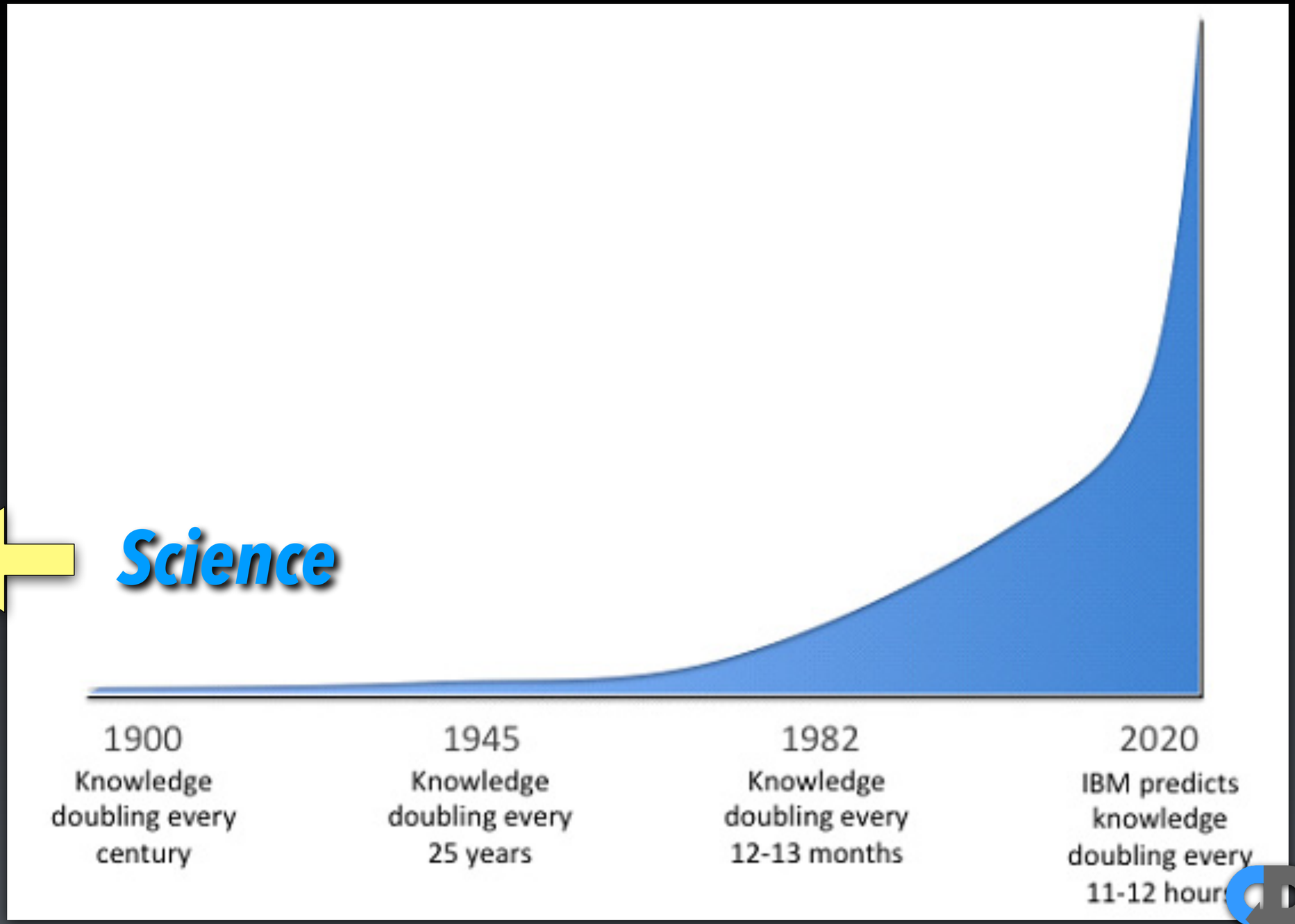
Continuous Delivery ltd



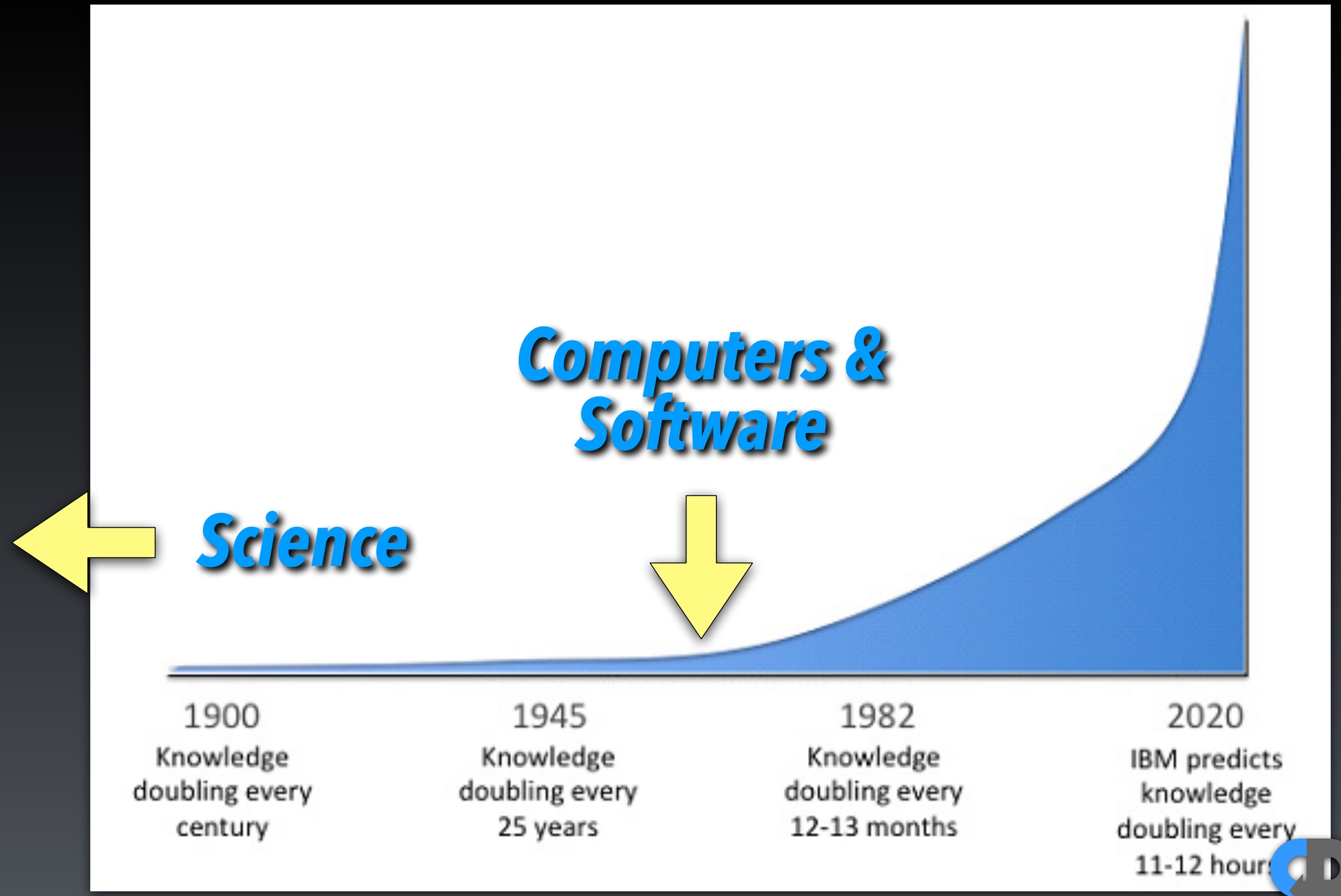
Continuous Delivery ltd



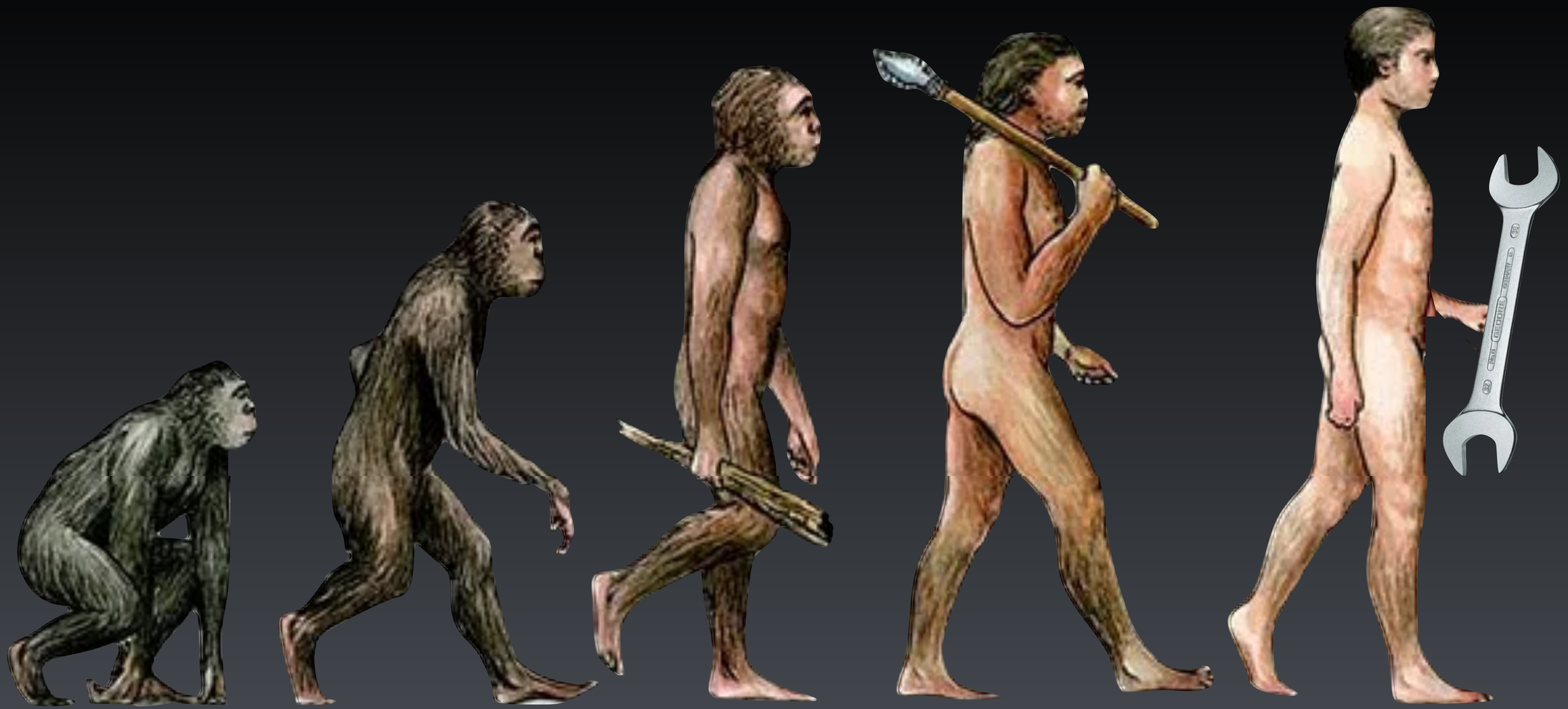
Continuous Delivery ltd



Continuous Delivery ltd



Continuous
Delivery ltd



Continuous Delivery ltd


Continuous
Delivery ltd

What is Engineering Really For?

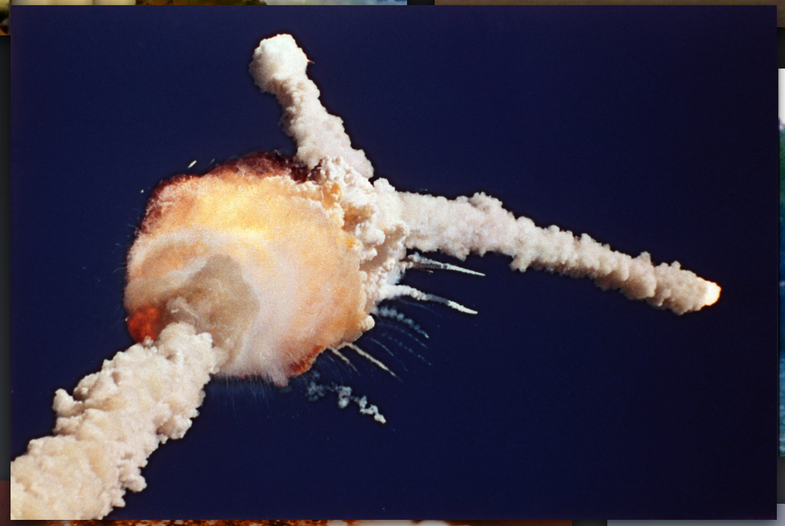
Continuous Delivery ltd


Continuous
Delivery ltd

















Continuous Delivery ltd


It's Not to Define Solutions!
AIR FRANCE
Continuous Delivery ltd













It Rules Out Bad Ideas!





Continuous Delivery ltd

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



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





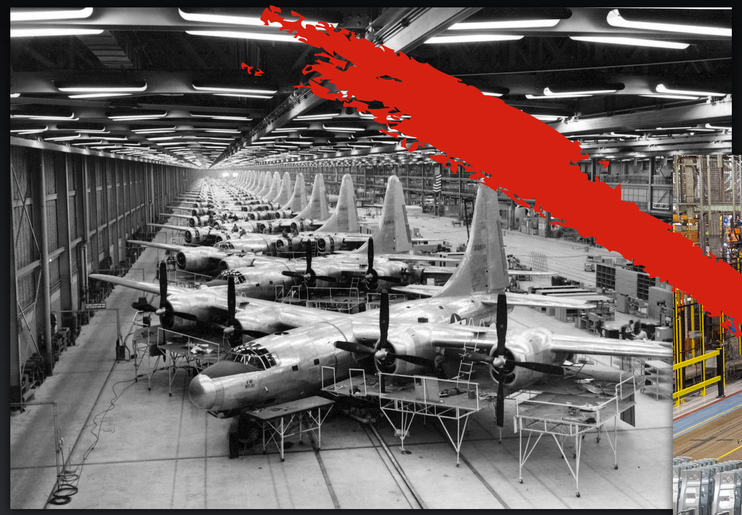






Continuous Delivery ltd

"The Things We Can't Afford to Get Wrong"







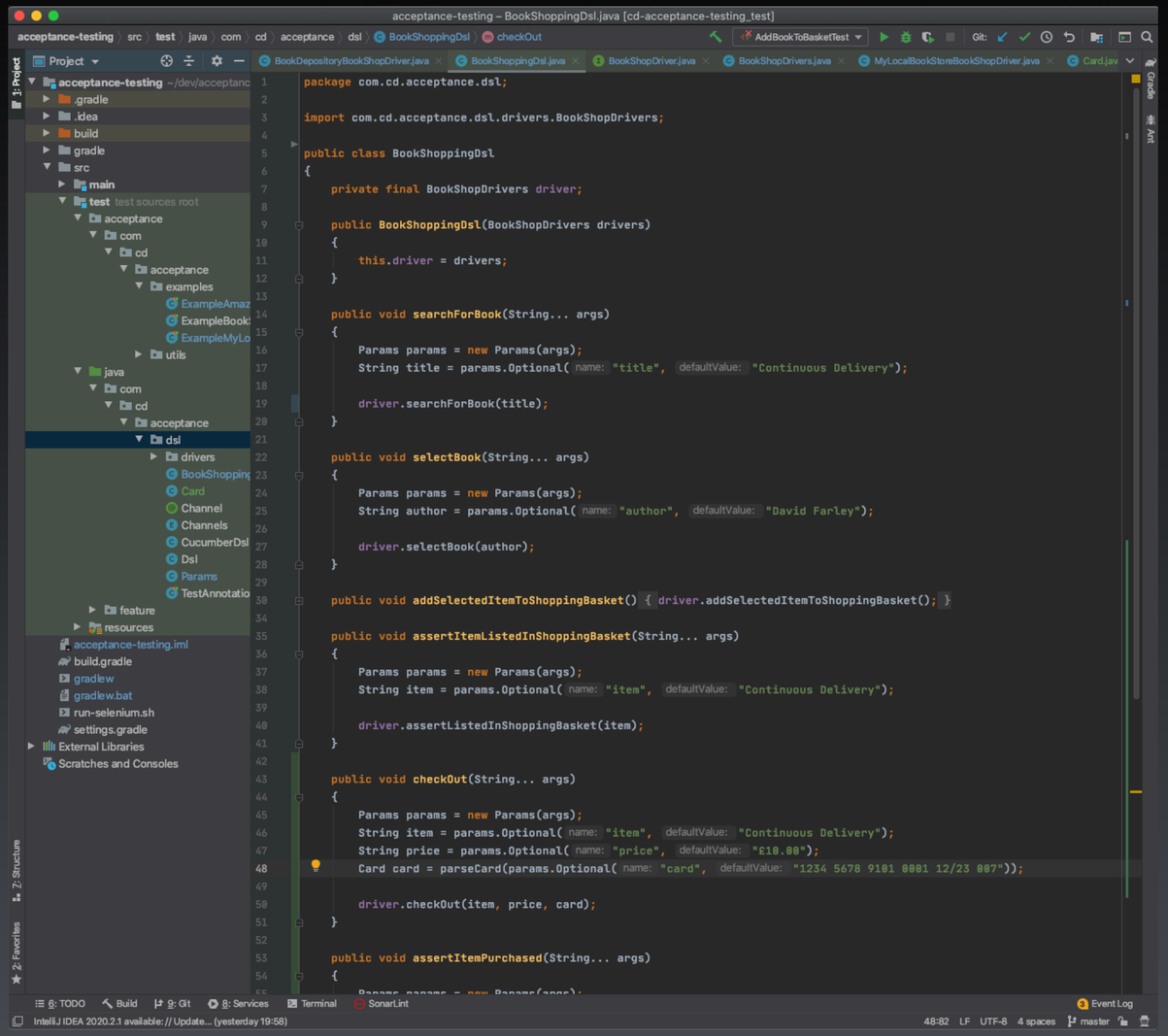




This is NOT Our Problem!

Continuous Delivery ltd


Continuous
Delivery ltd




```
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");

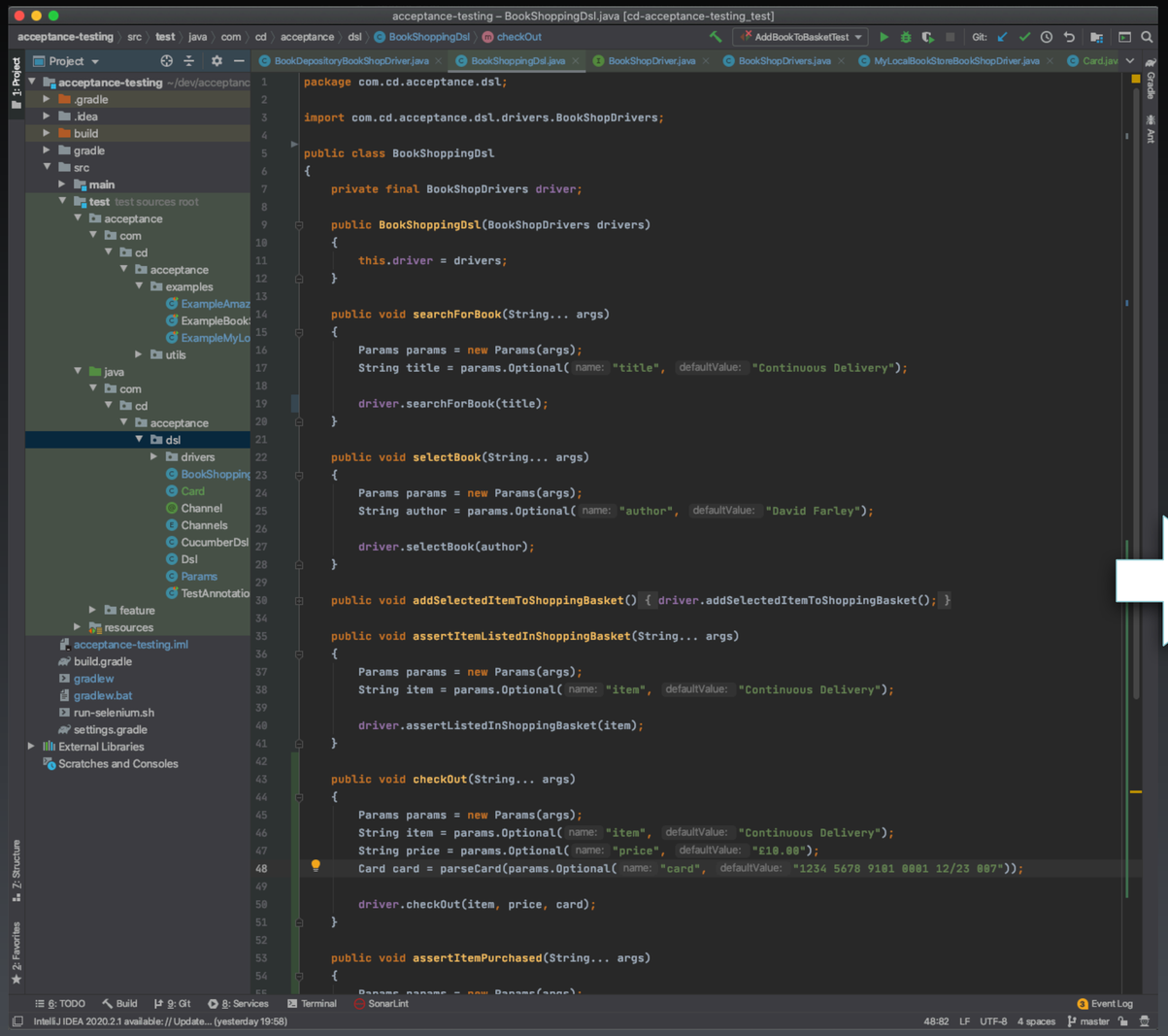
        driver.assertListedInShoppingBasket(item);
    }

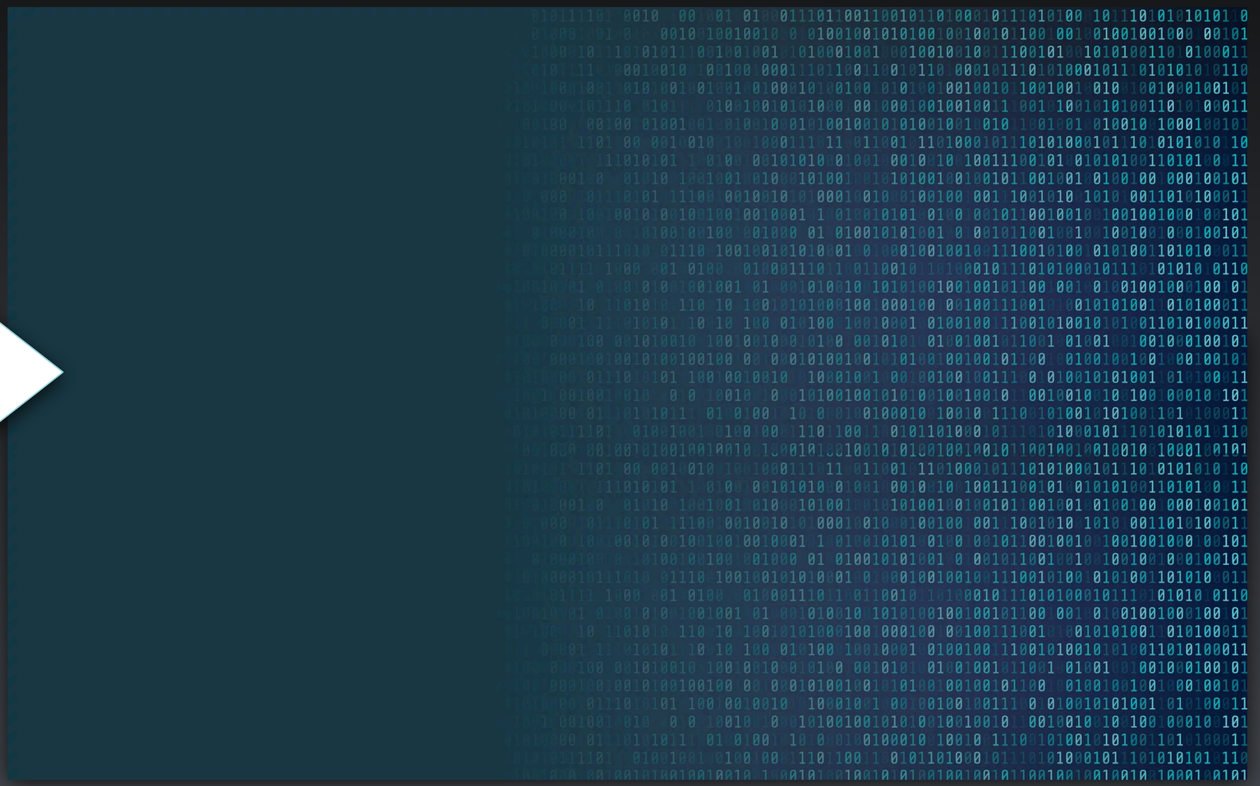
    public void checkOut(String... args)
    {
        Params params = new Params(args);
        String item = params.Optional( name: "item",  defaultValue: "Continuous Delivery");
        String price = params.Optional( name: "price",  defaultValue: "£10.00");
        Card card = parseCard(params.Optional(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

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

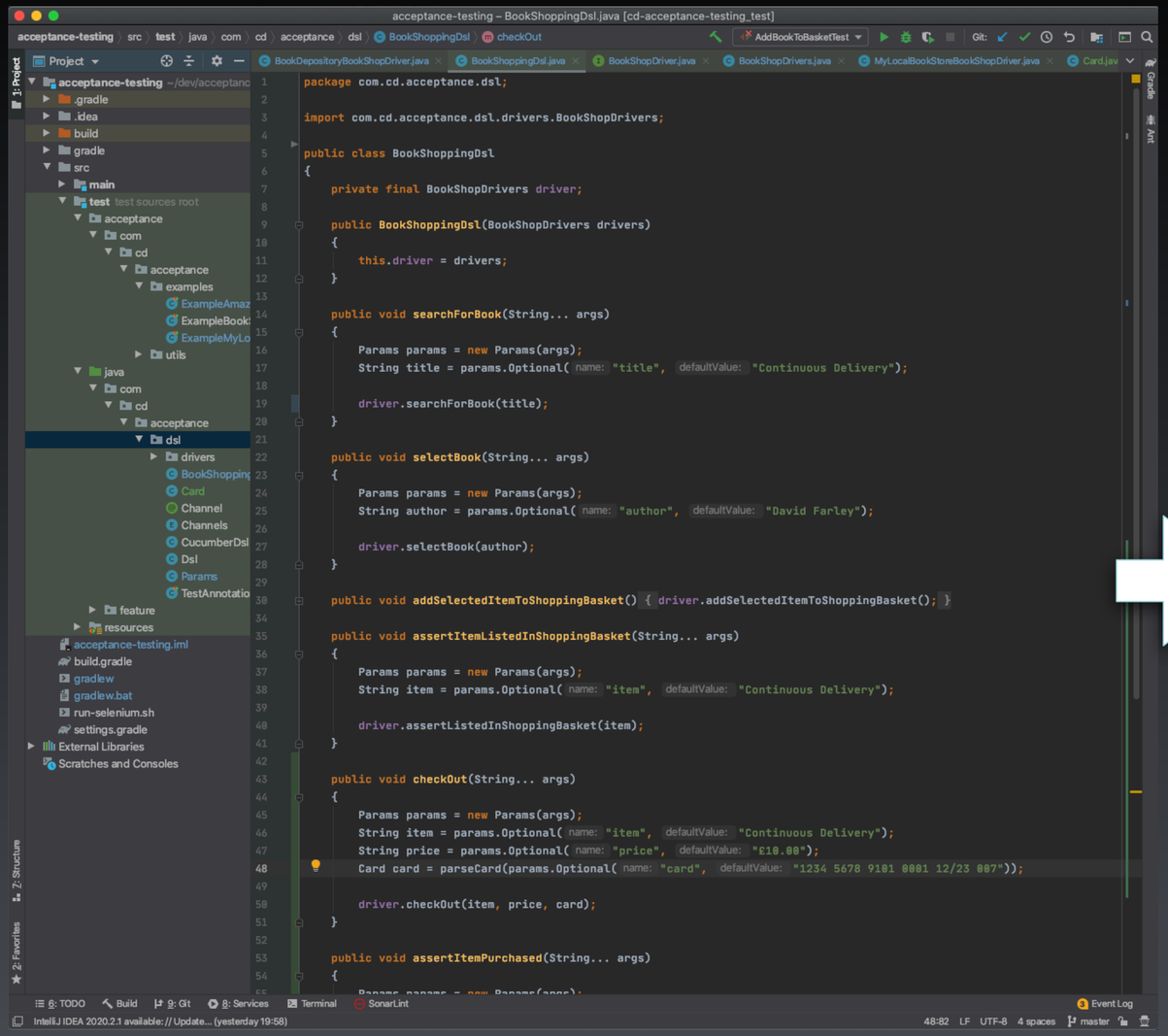
    public void assertItemPurchased(String... args)
    {
```

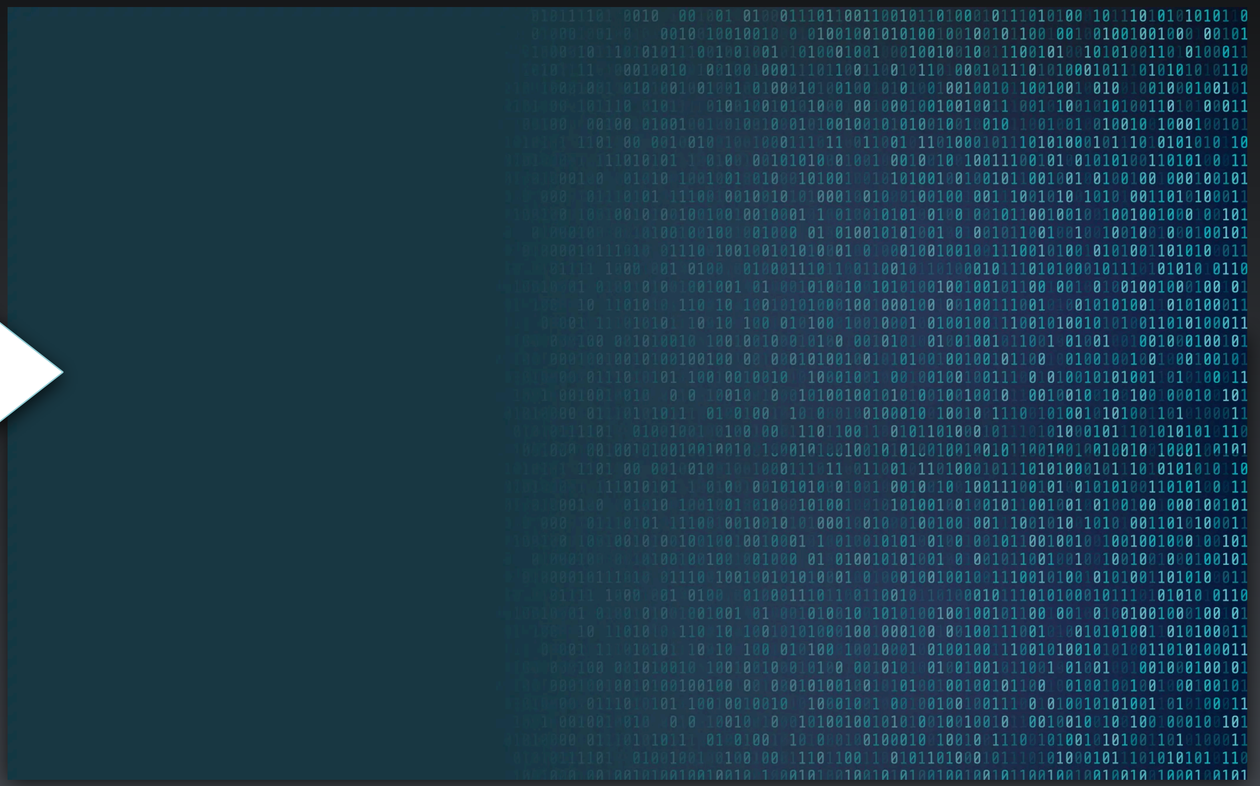
Continuous Delivery ltd



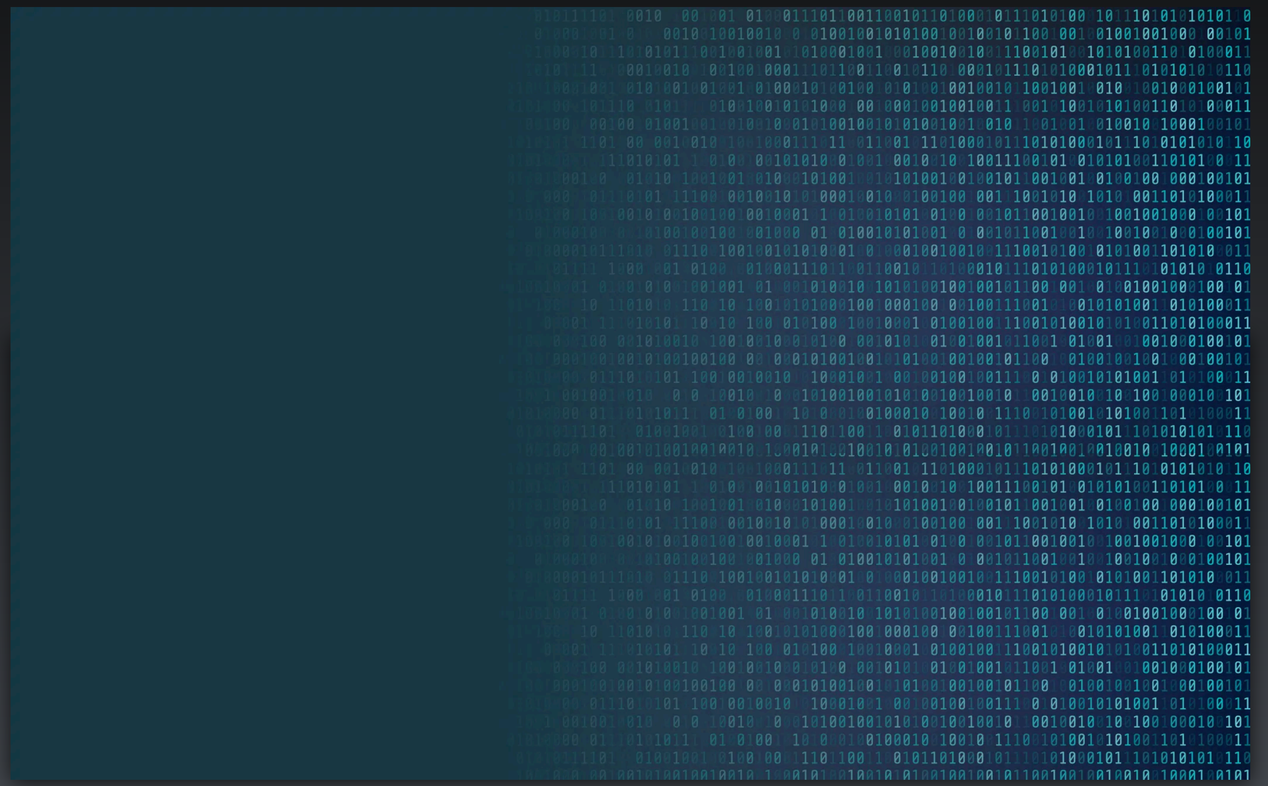


Continuous Delivery ltd

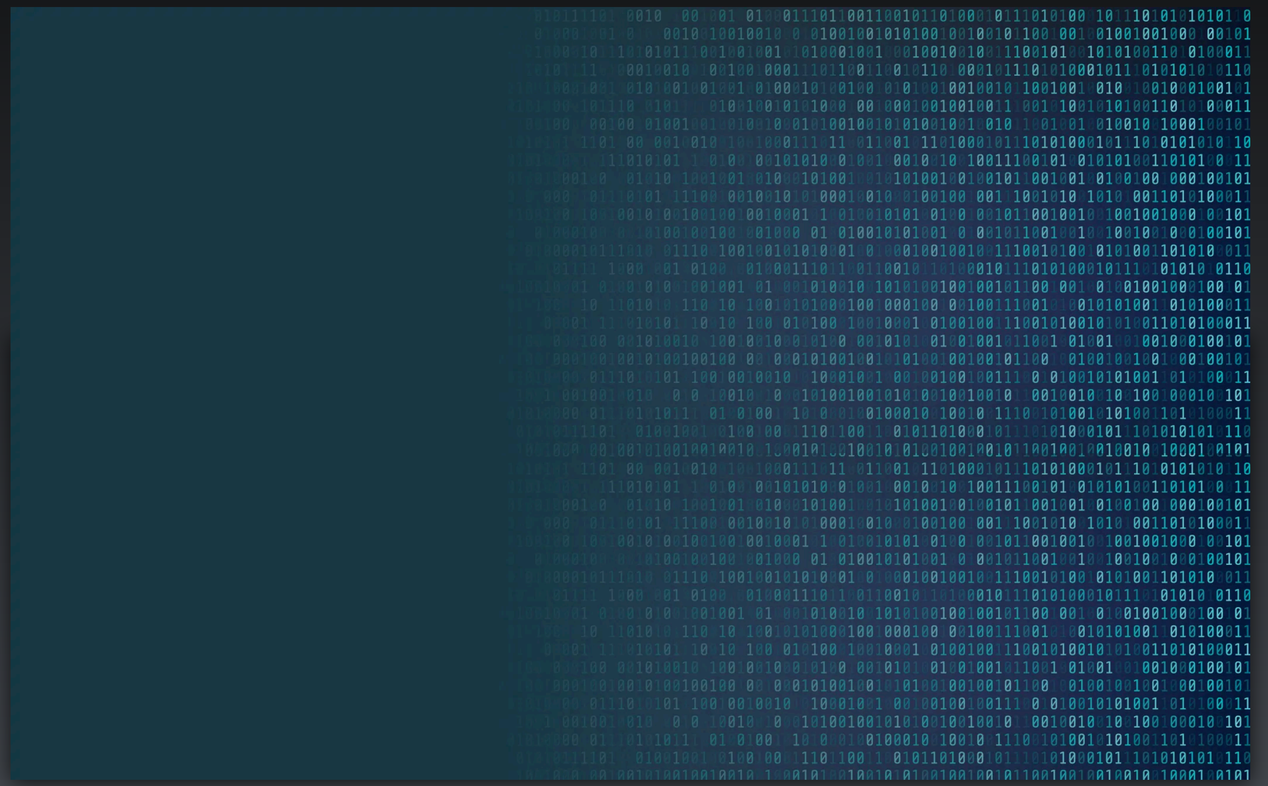




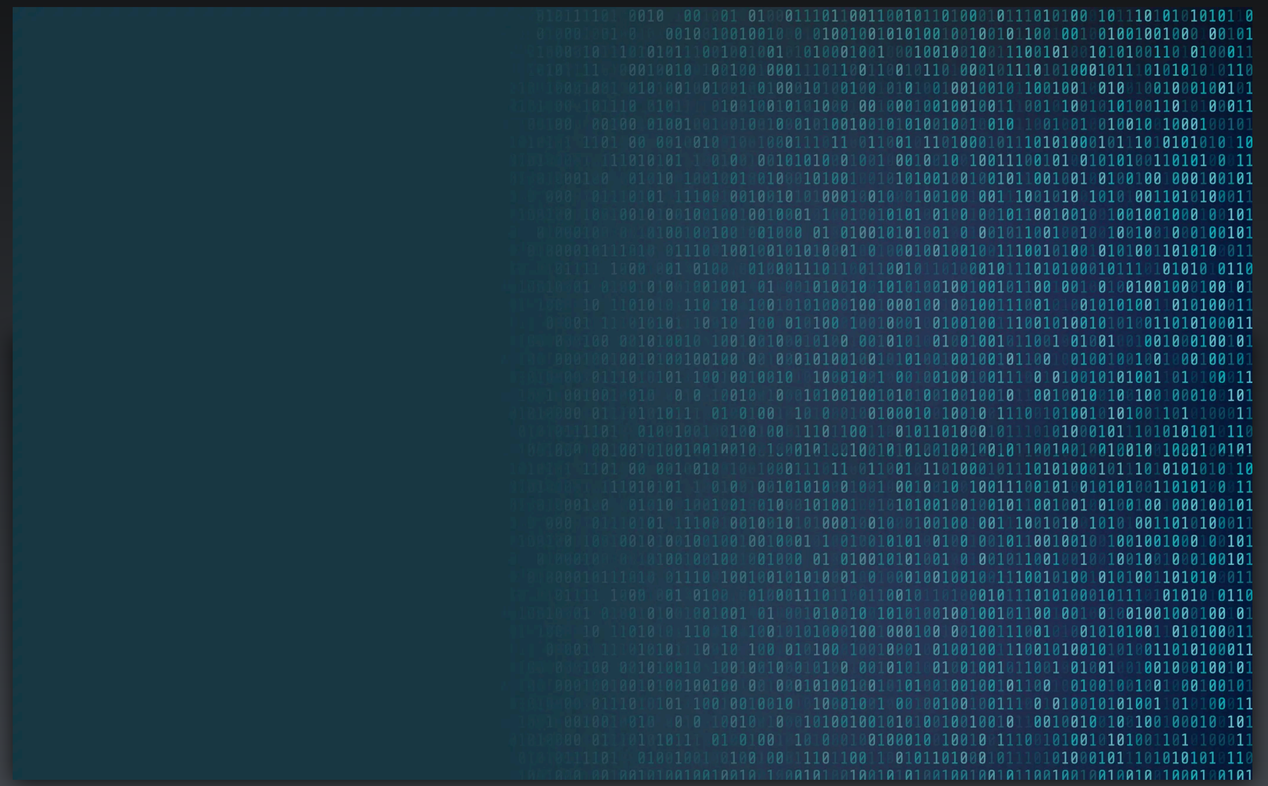
Continuous Delivery ltd



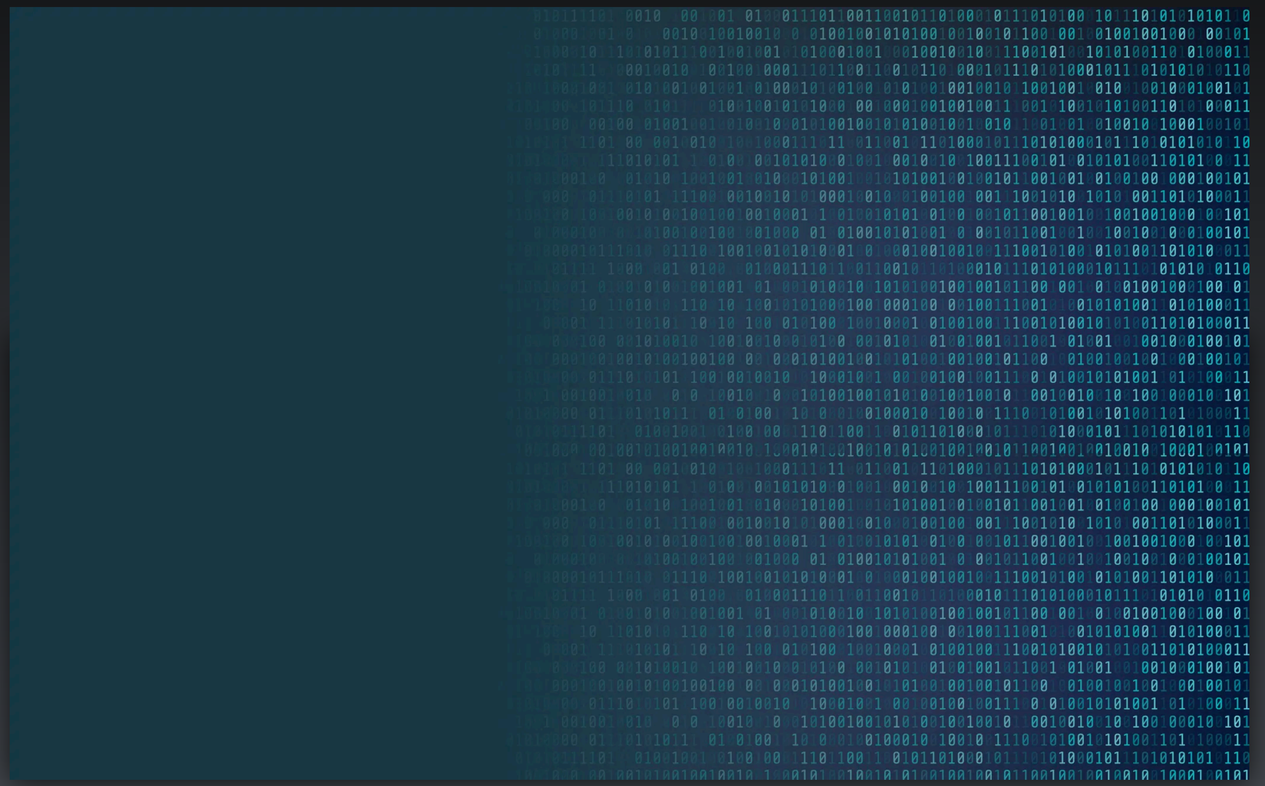
Continuous Delivery ltd



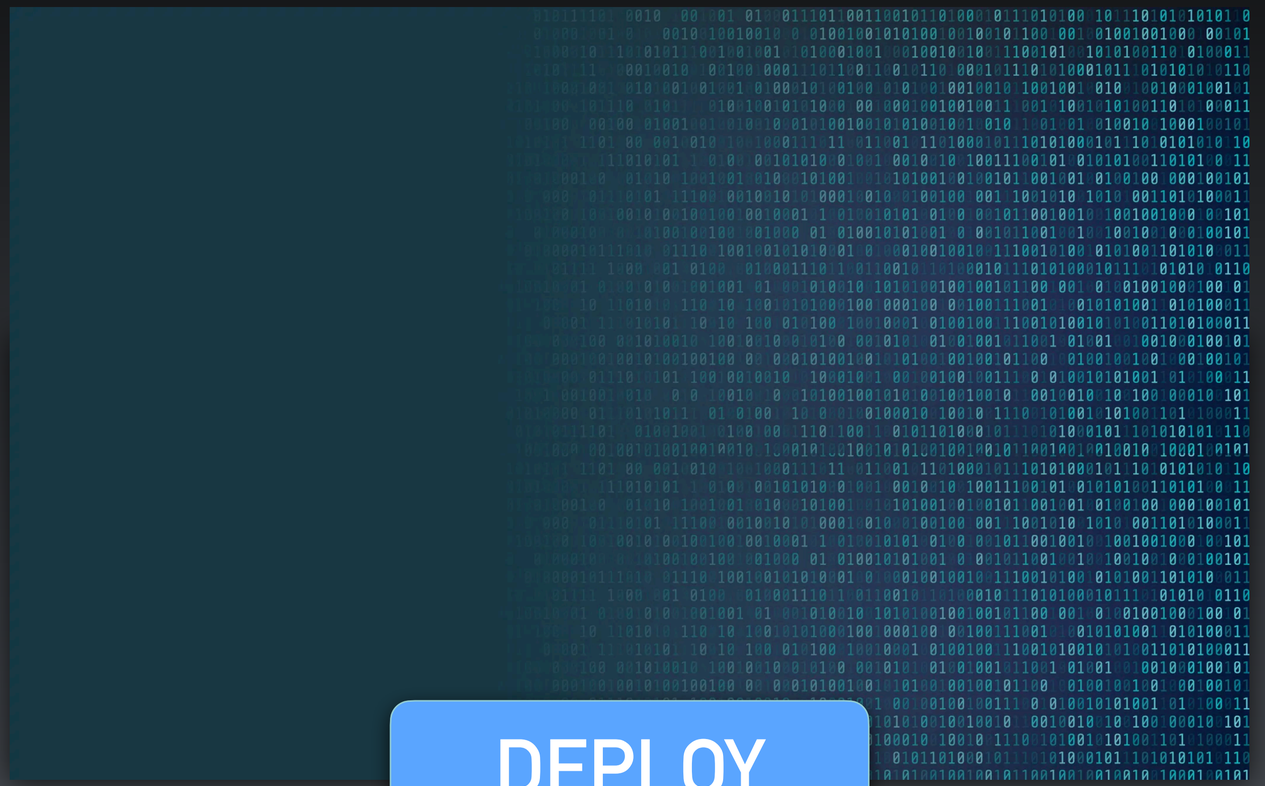
Continuous Delivery ltd



Continuous Delivery ltd



Continuous Delivery ltd



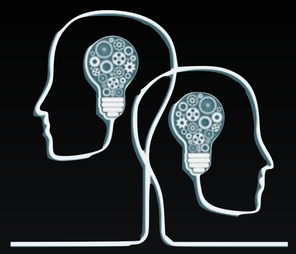
Continuous Delivery ltd

We Can Recreate Entire Systems for Free!



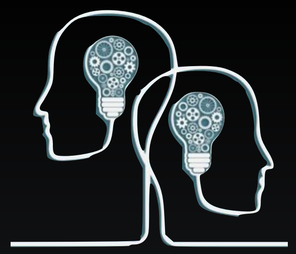
Continuous Delivery ltd


Continuous
Delivery ltd



Engineering *IS* About Exploration & Discovery

Continuous Delivery ltd



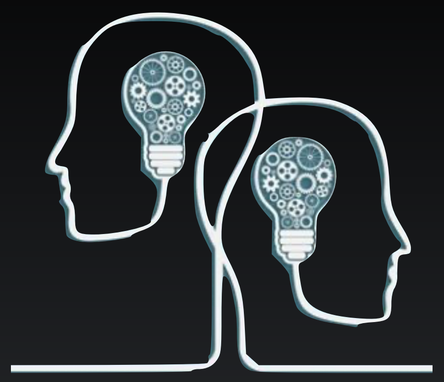
Software Development *IS* About Exploration & Discovery

Continuous Delivery ltd

The Stuff That Works

Continuous Delivery ltd

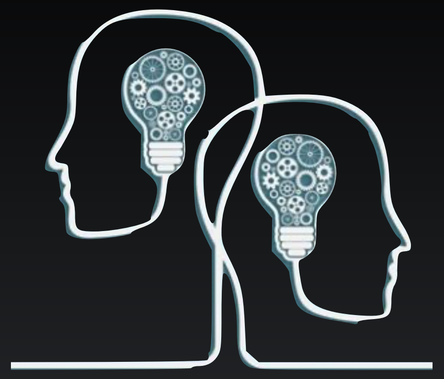
The Stuff That Works



Optimise for Learning

Continuous Delivery ltd

The Stuff That Works





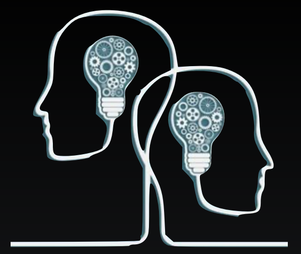
Optimise for Learning

Optimise for Managing Complexity

Continuous Delivery ltd


Continuous
Delivery ltd

Optimise for Learning



Iterative Experimental

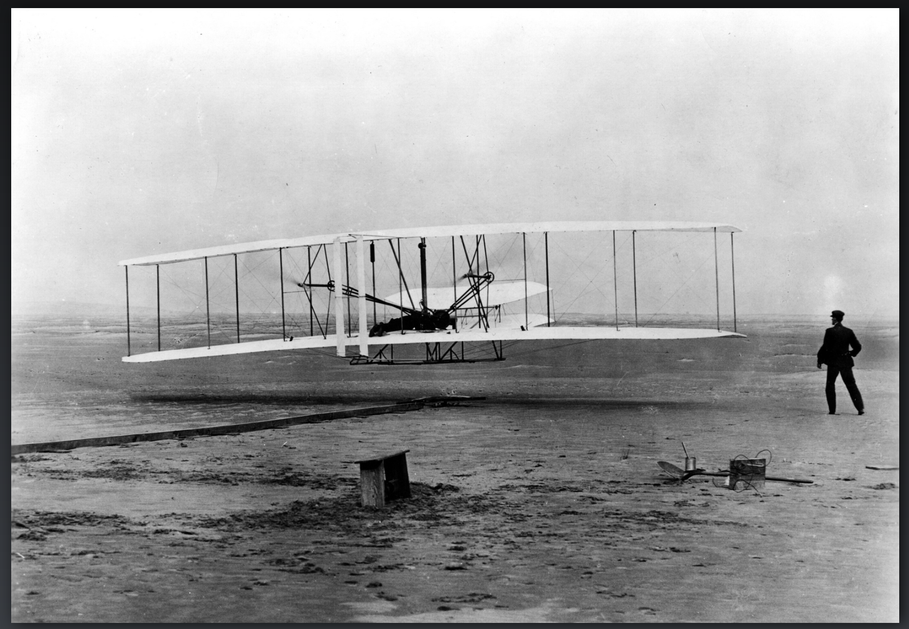
Feedback Driven

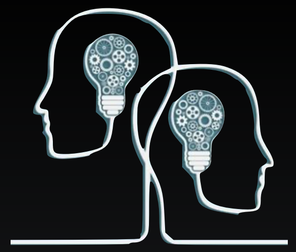
Incremental Empirical

Continuous Delivery ltd


Continuous
Delivery ltd

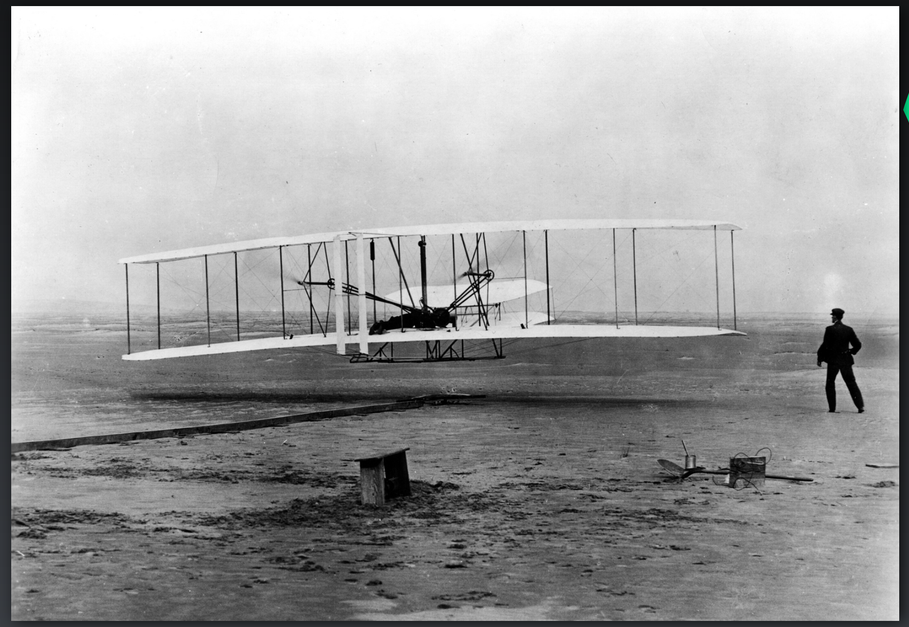
Iteration



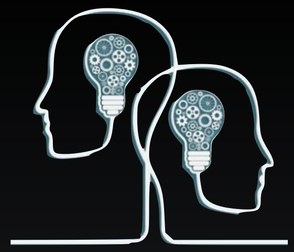


Continuous Delivery ltd

Iteration

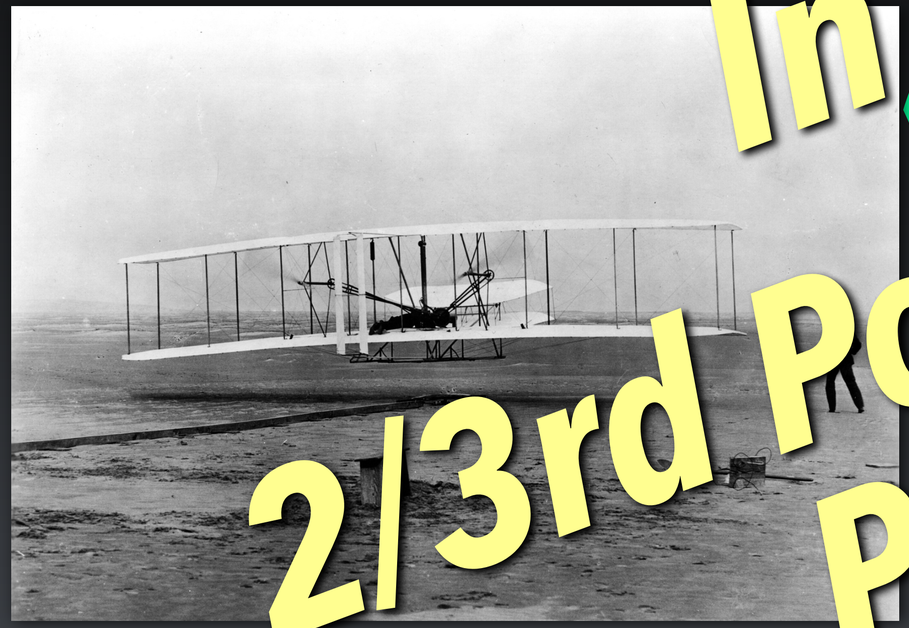






Continuous Delivery ltd

Iteration



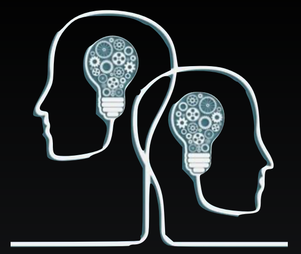


In 2017
2/3rd Population of the Planet Flew
With no Fatalities!



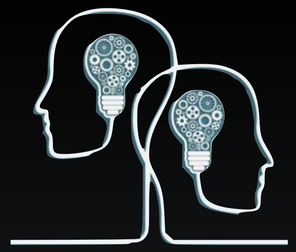
Continuous Delivery ltd

Iteration



Continuous Delivery ltd

Iteration

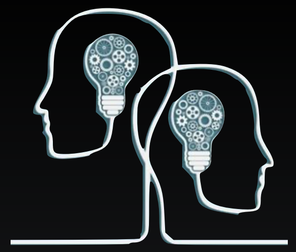




The Goal

Continuous Delivery ltd

Iteration

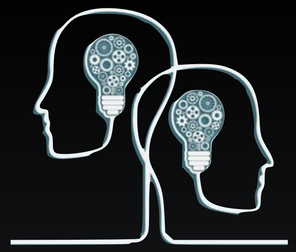


The Goal

Start

Continuous Delivery ltd

Iteration





The Goal



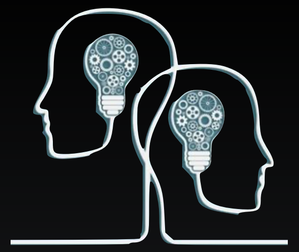
Start



Fitness Function

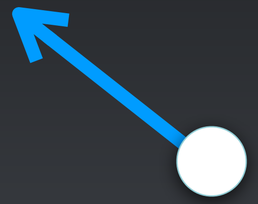
Continuous Delivery ltd

Iteration





The Goal



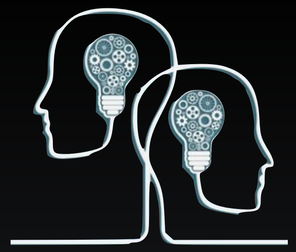
Start



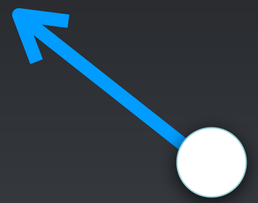
Fitness Function

Continuous Delivery ltd

Iteration



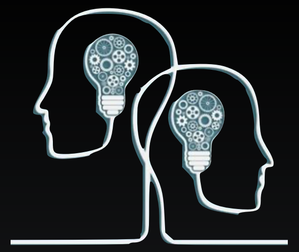


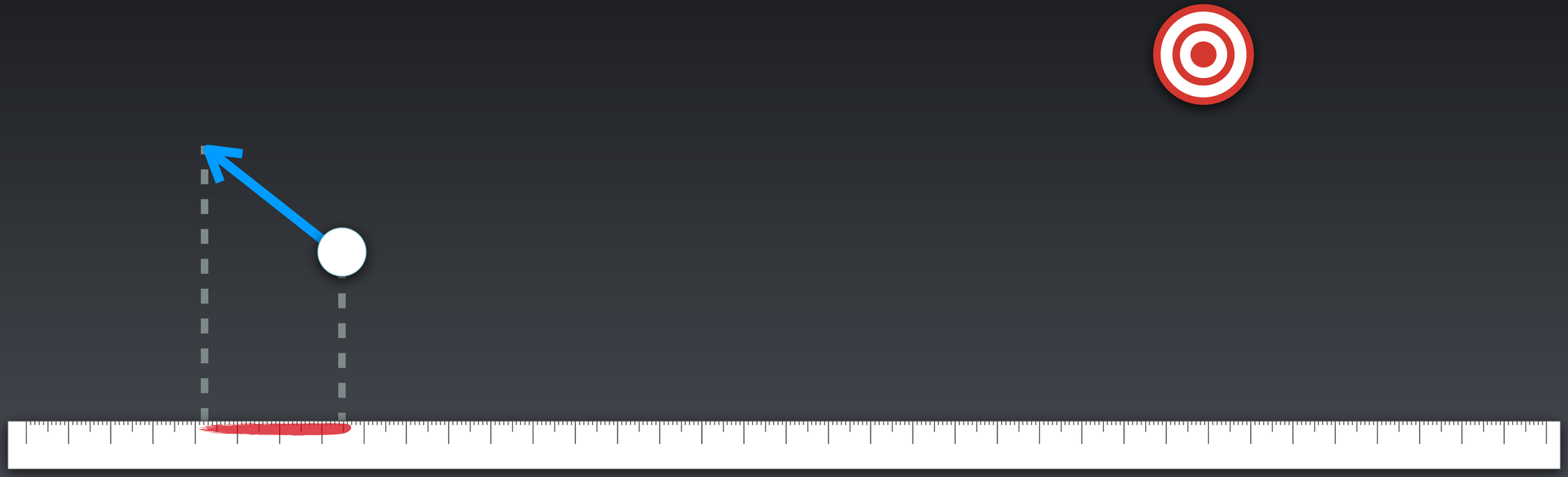




Continuous Delivery ltd

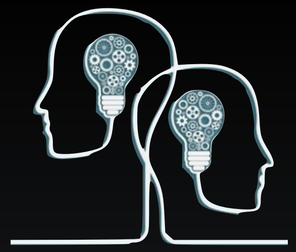
Iteration





Continuous Delivery ltd

Iteration



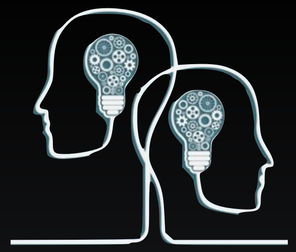


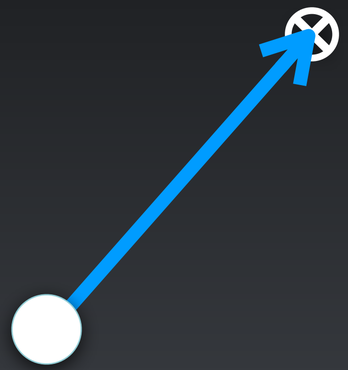




Continuous Delivery ltd

Iteration



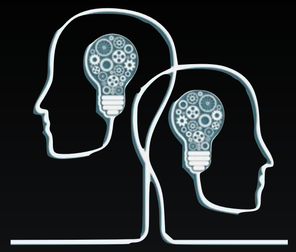


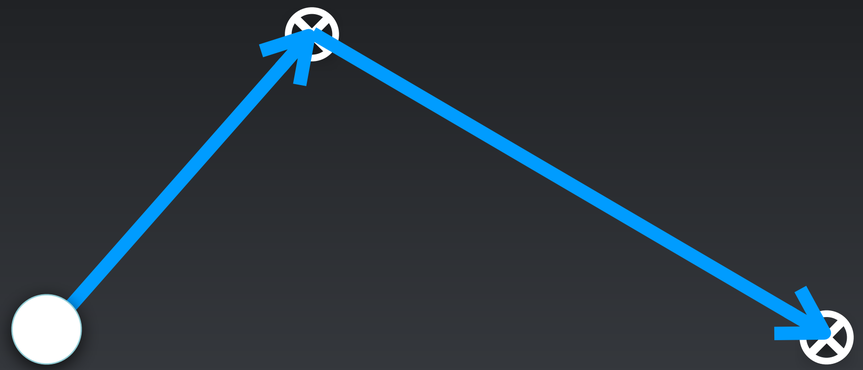




Continuous Delivery ltd

Iteration



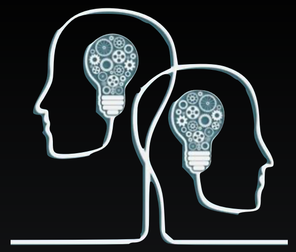


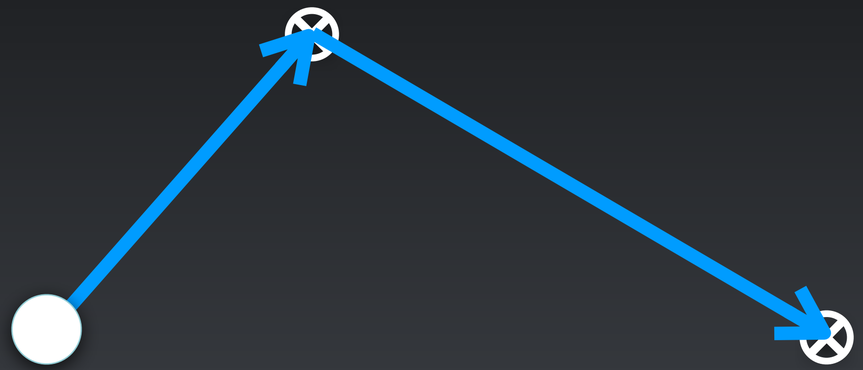




Continuous Delivery ltd

Iteration



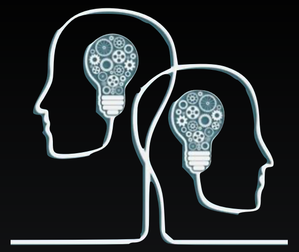


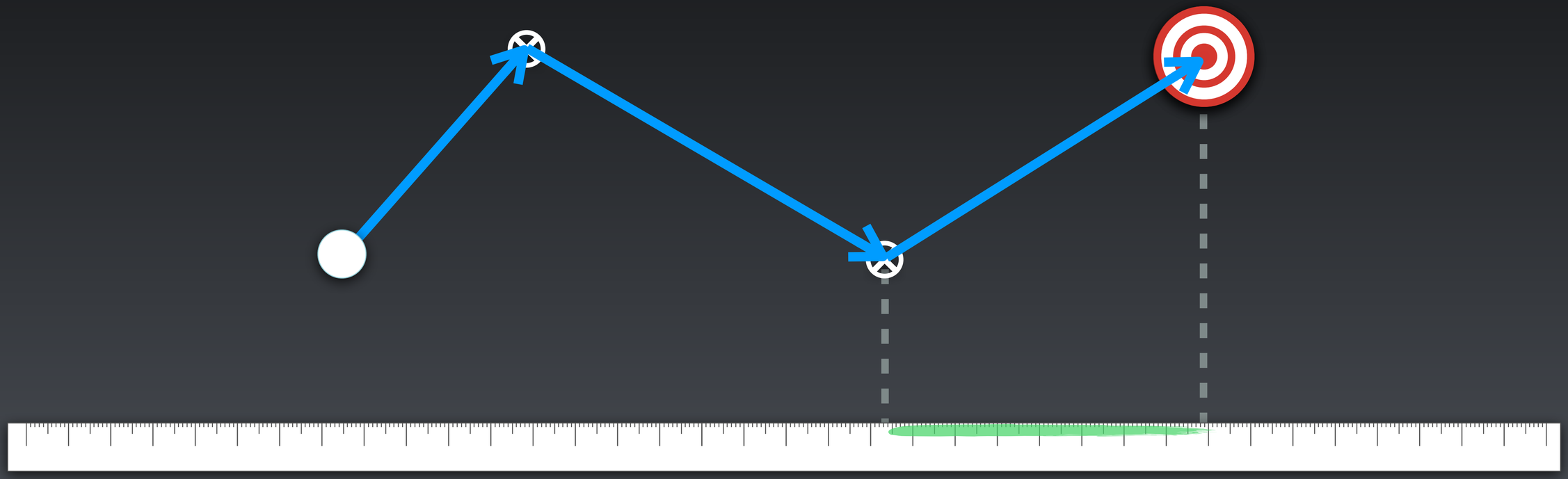




Continuous Delivery ltd

Iteration

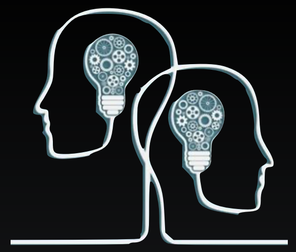


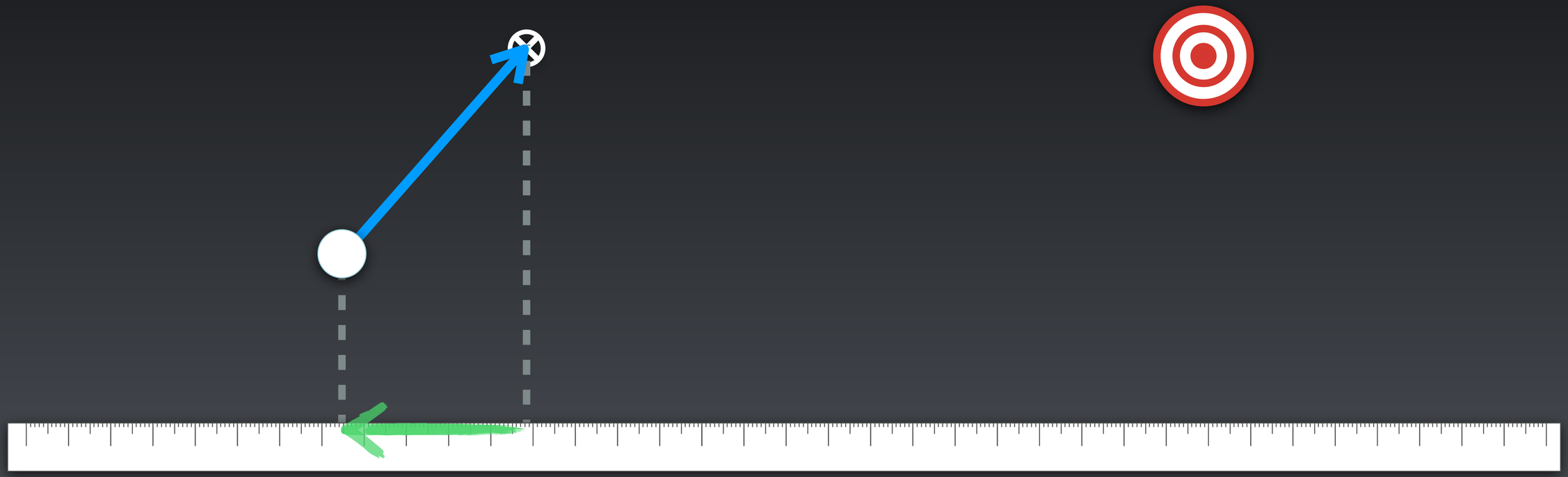


Continuous Delivery ltd


Continuous
Delivery ltd

Feedback

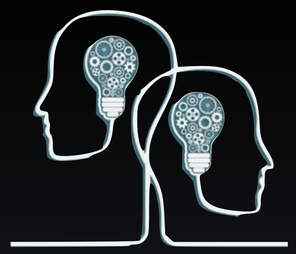




Feedback!

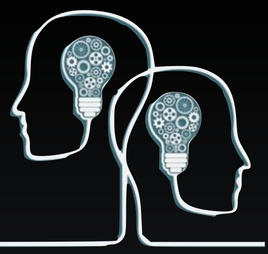
Continuous Delivery ltd

Feedback



Continuous Delivery ltd

Feedback



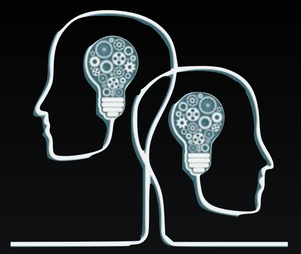
How Do You Balance a Broom?



Continuous Delivery ltd

Feedback

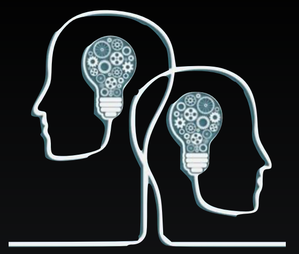




Continuous Delivery ltd

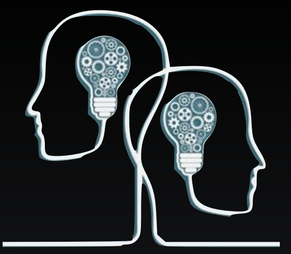
Feedback





Continuous Delivery ltd

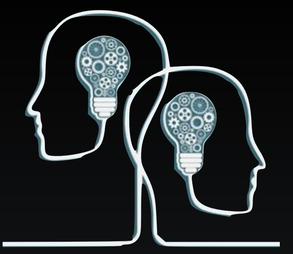
Feedback





Continuous Delivery ltd

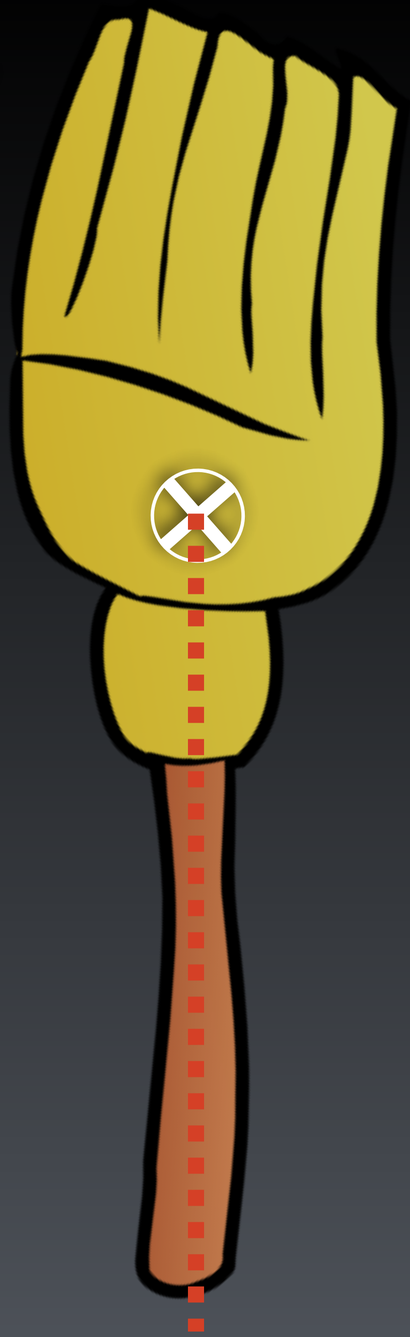
Feedback

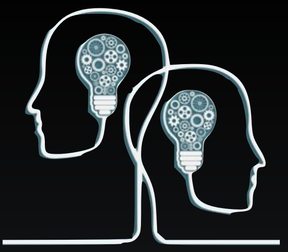




Continuous Delivery ltd

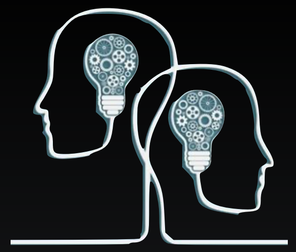
Feedback



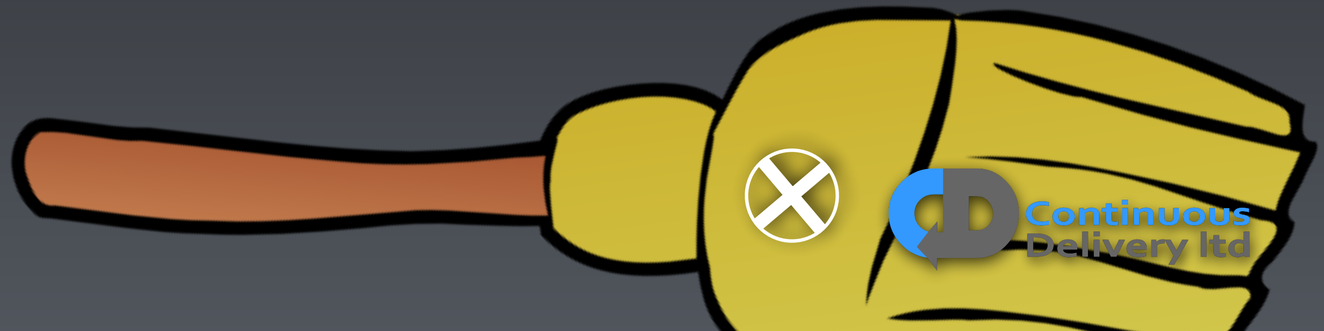


Continuous Delivery ltd

Feedback

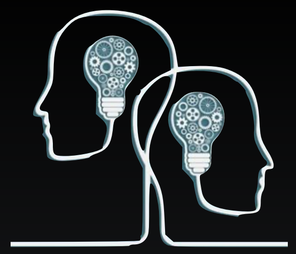


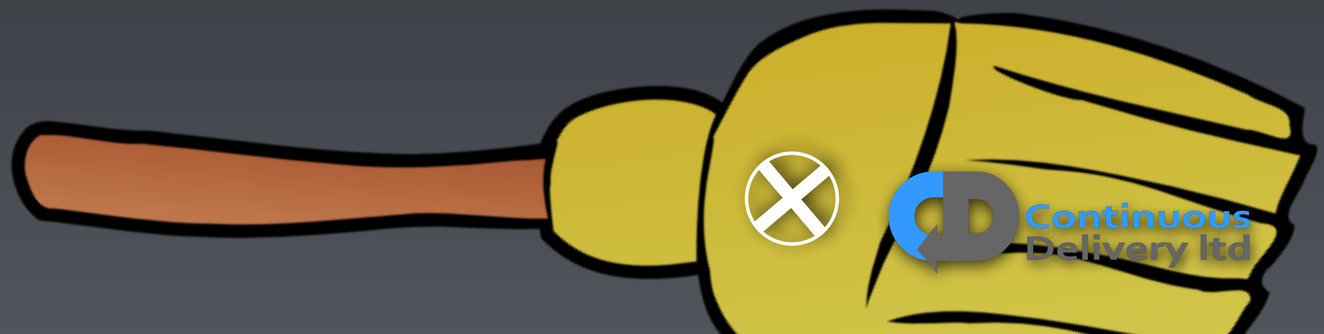
There's Only One Solution to This!



Continuous Delivery ltd

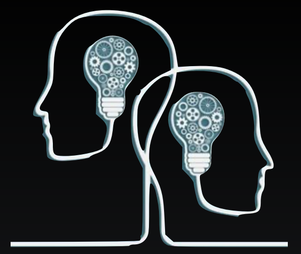
Feedback





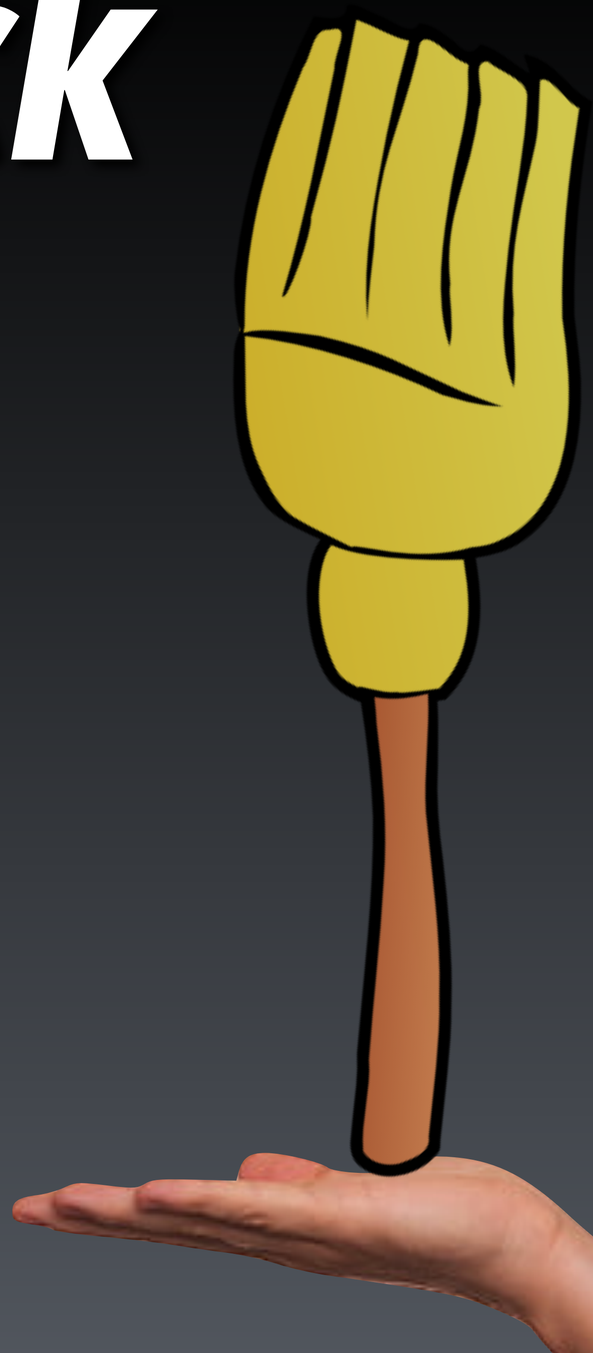
Feedback

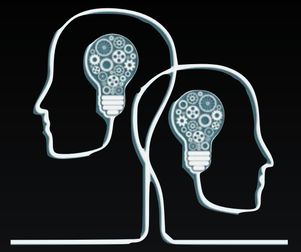




Continuous Delivery ltd

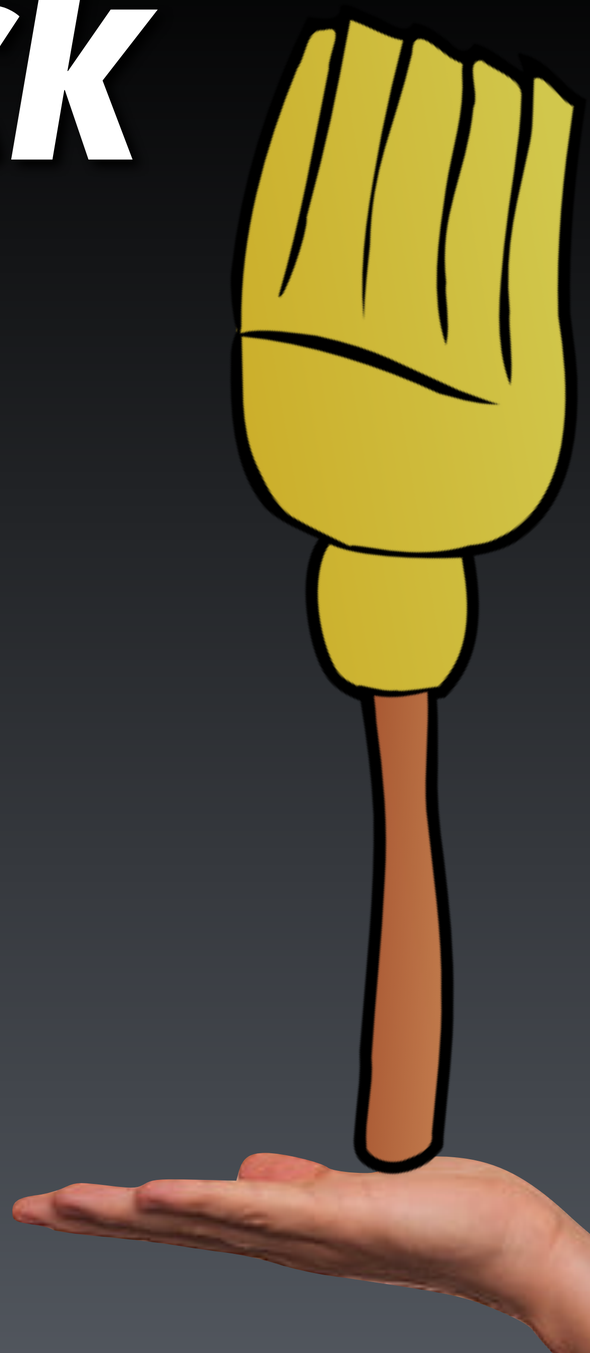
Feedback

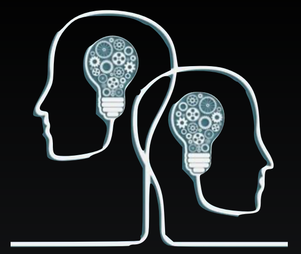




Continuous Delivery ltd

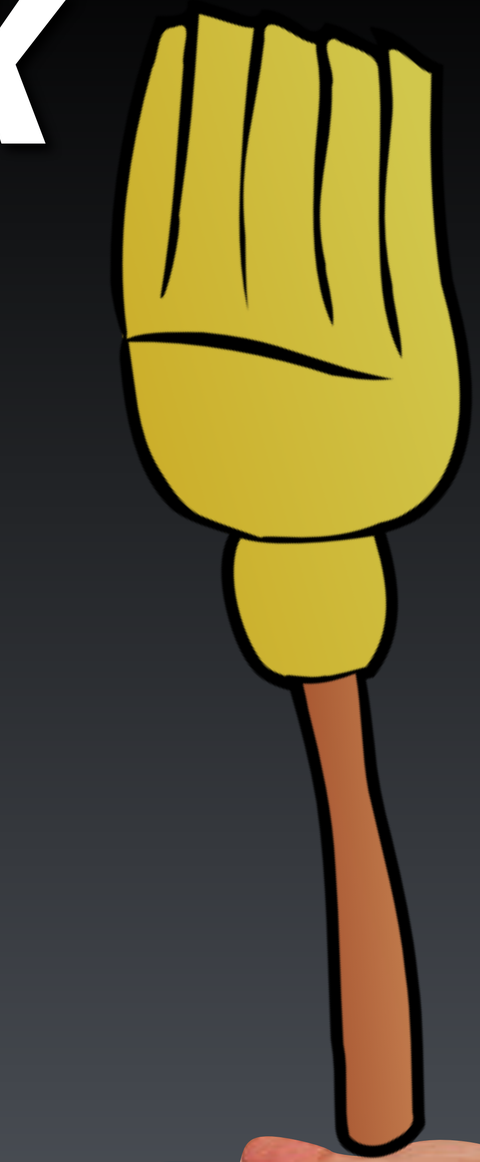
Feedback

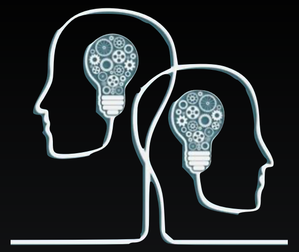




Continuous Delivery ltd

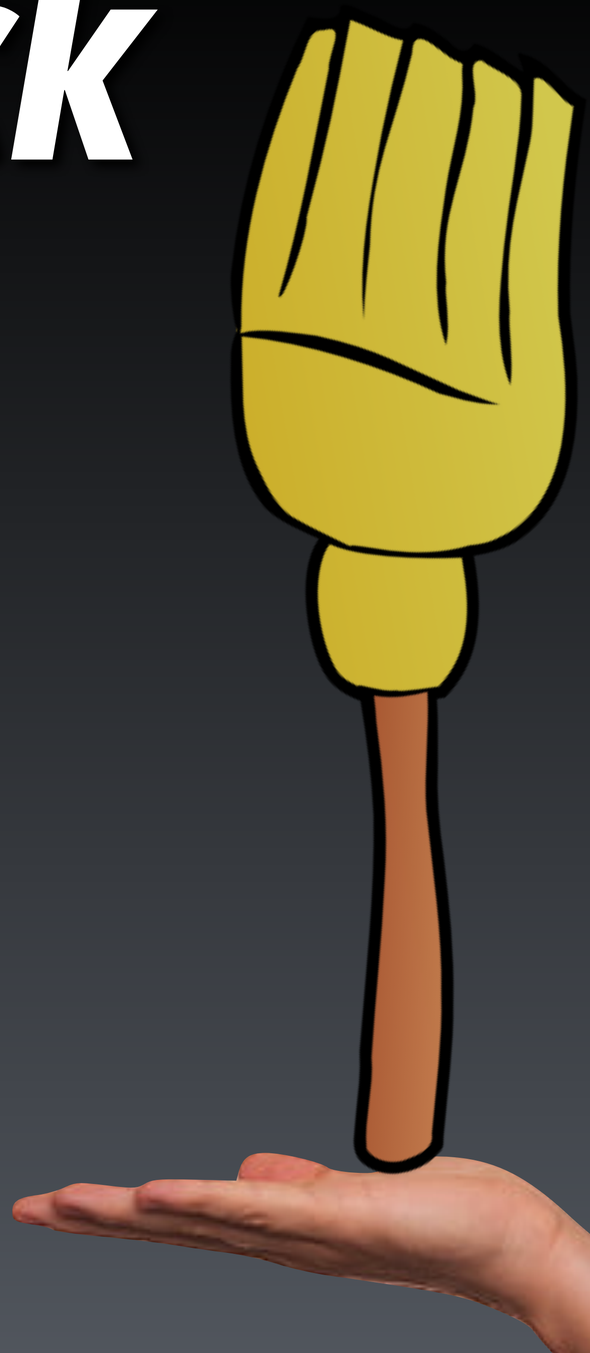
Feedback

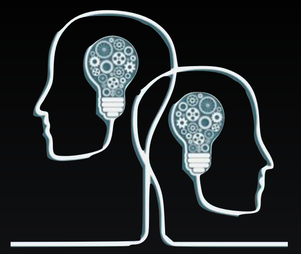




Continuous Delivery ltd

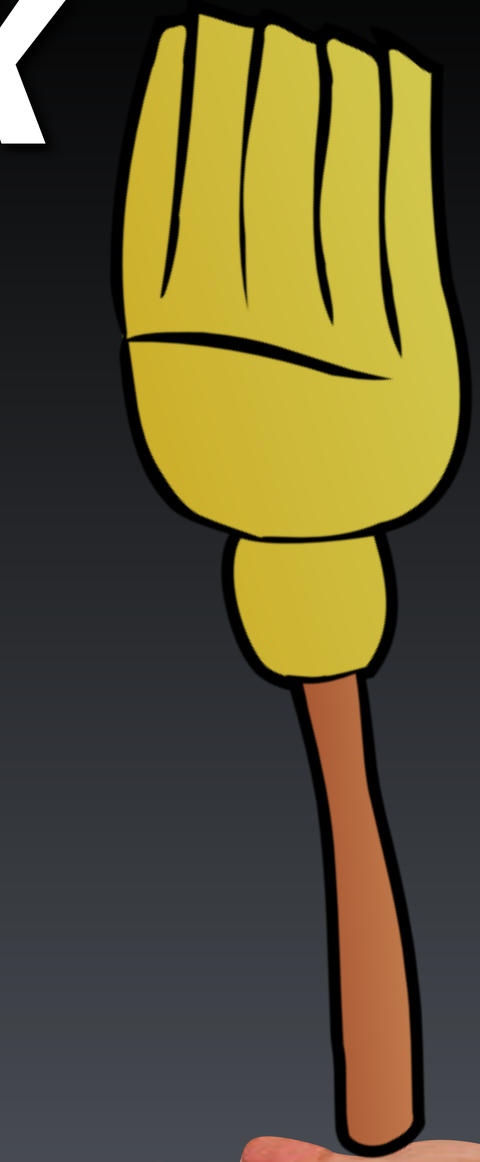
Feedback

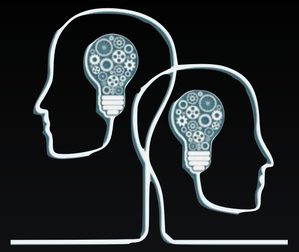




Continuous Delivery ltd

Feedback

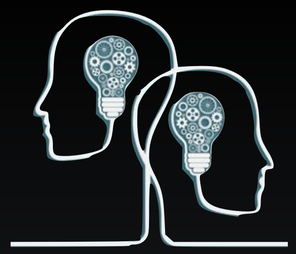




Continuous Delivery ltd

Feedback

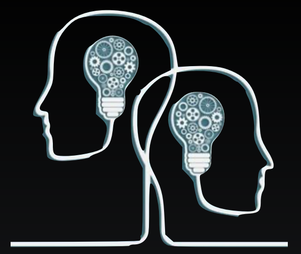




Continuous Delivery ltd


Continuous
Delivery ltd

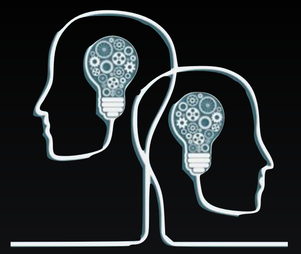
Incremental



Continuous Delivery ltd

Incremental





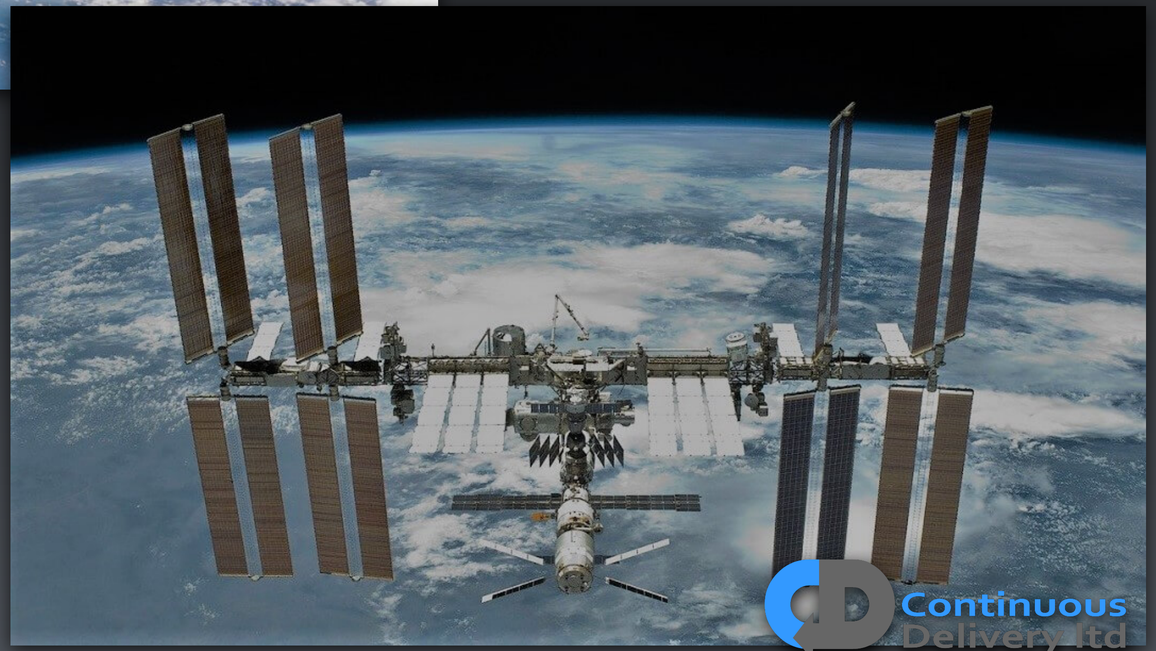
Continuous Delivery ltd

Incremental

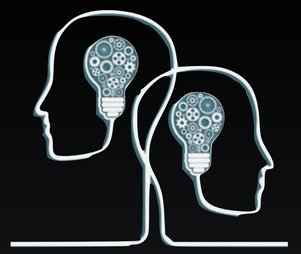








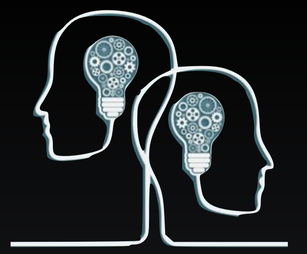
Incremental



Continuous Delivery ltd

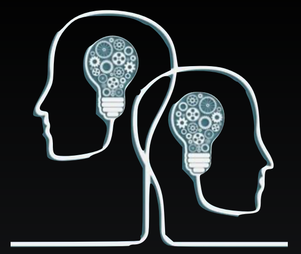
Incremental





Continuous Delivery ltd

Incremental

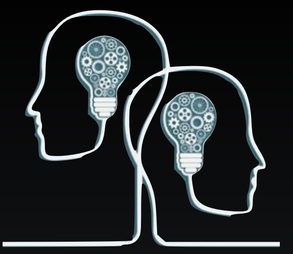




No 3rd Party Apps
No GPS
No Video
No Facetime

Continuous Delivery ltd

Incremental





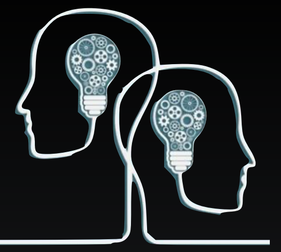


No 3rd Party Apps
No GPS
No Video
No Facetime

No 4G
No Siri
No Face Recognition
No Apple Pay

Continuous Delivery ltd

Incremental









No 3rd Party Apps
No GPS
No Video
No Facetime

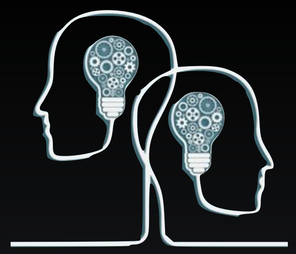
No 4G
No Siri
No Face Recognition
No Apple Pay

What Comes Next?

Continuous Delivery ltd


Continuous
Delivery ltd

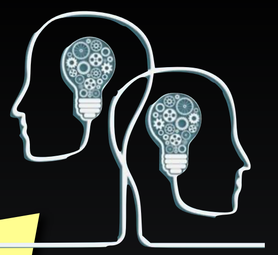
Experimental



Scary!

Continuous Delivery ltd

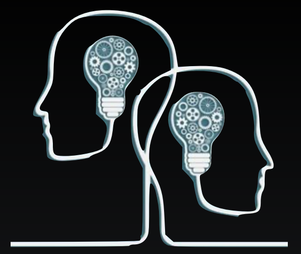
Experimental



How Else Can We Make Progress?

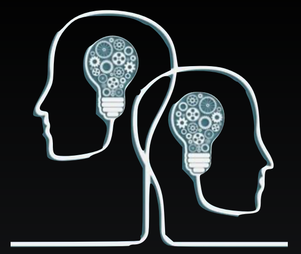
Continuous Delivery ltd

Experimental



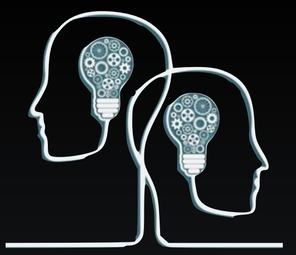
Continuous Delivery ltd

Experimental



Continuous Delivery ltd

Experimental

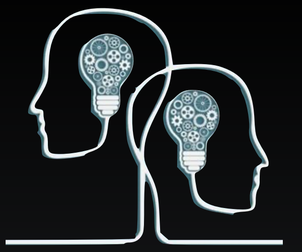


Controlling the Variables!

Continuous Delivery ltd

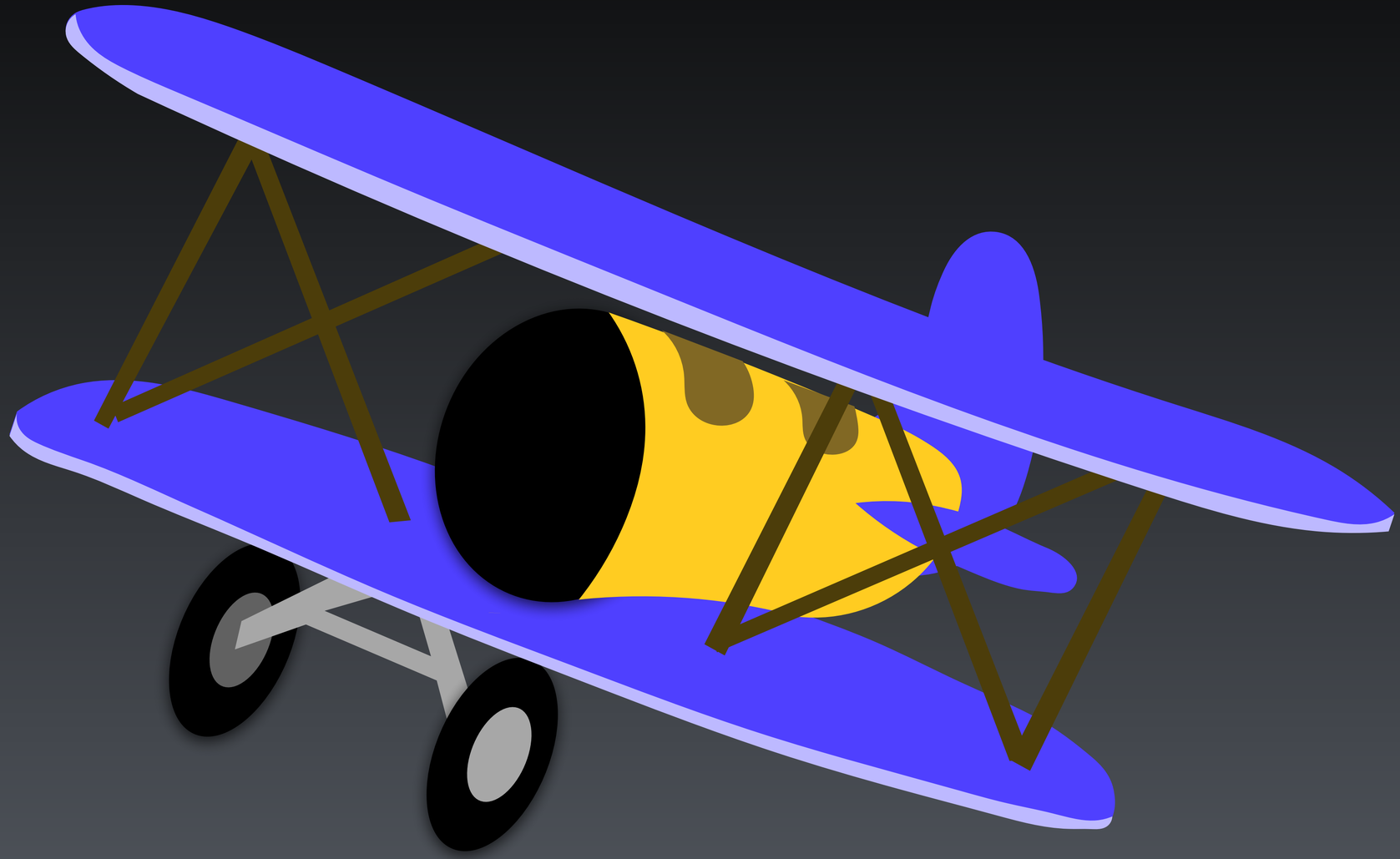
Experimental

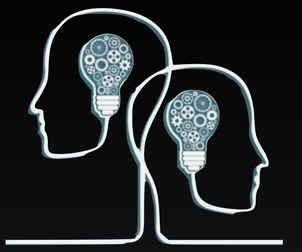




Continuous Delivery ltd

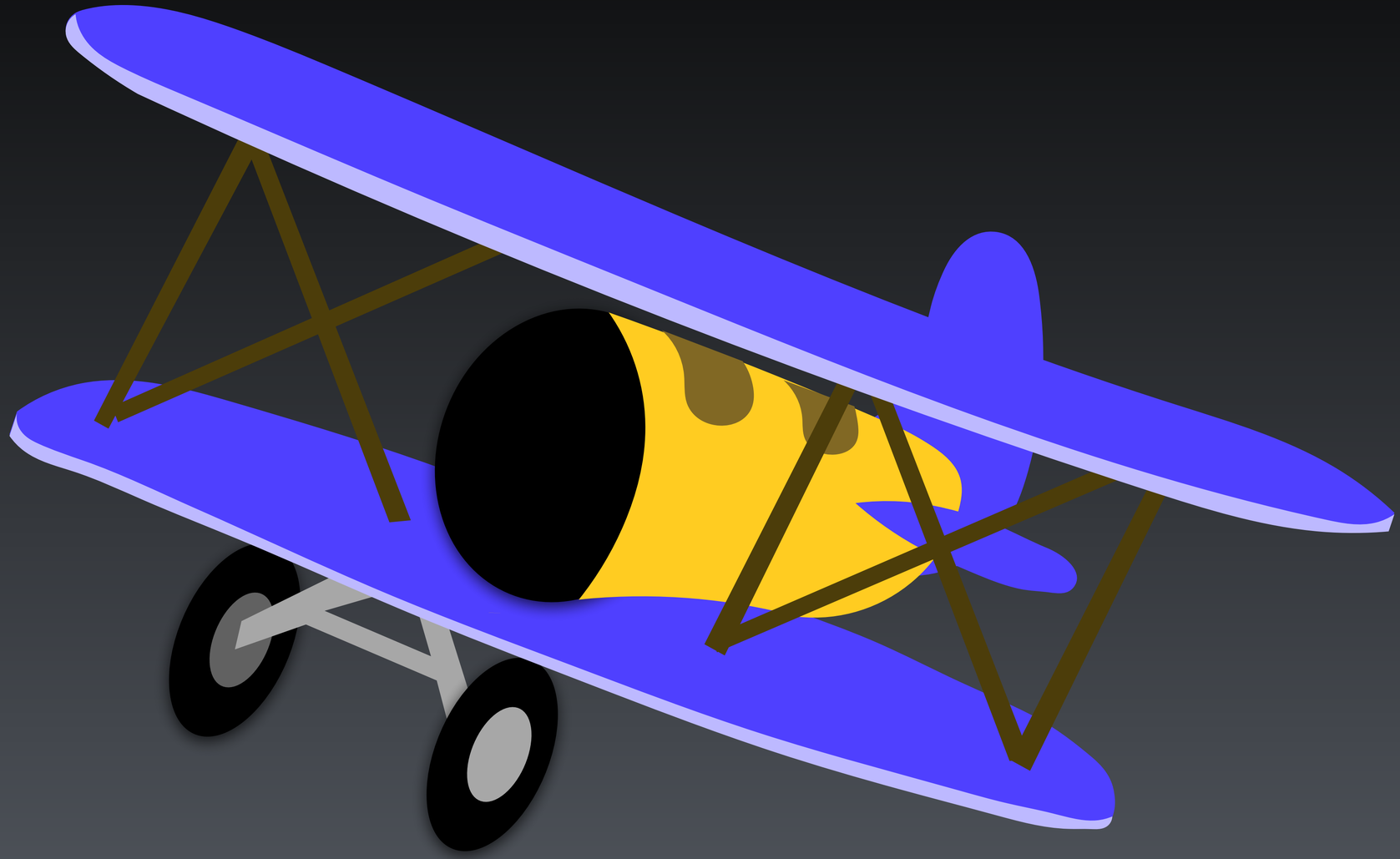
Experimental

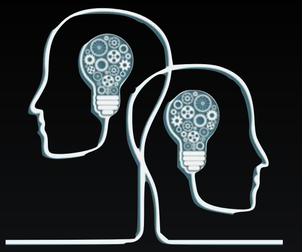




Continuous Delivery ltd

Experimental

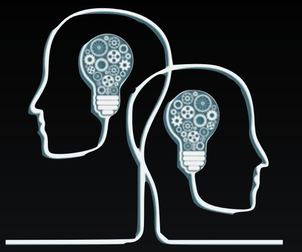




Continuous Delivery ltd

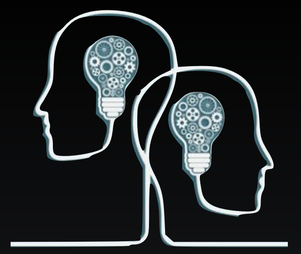
Experimental





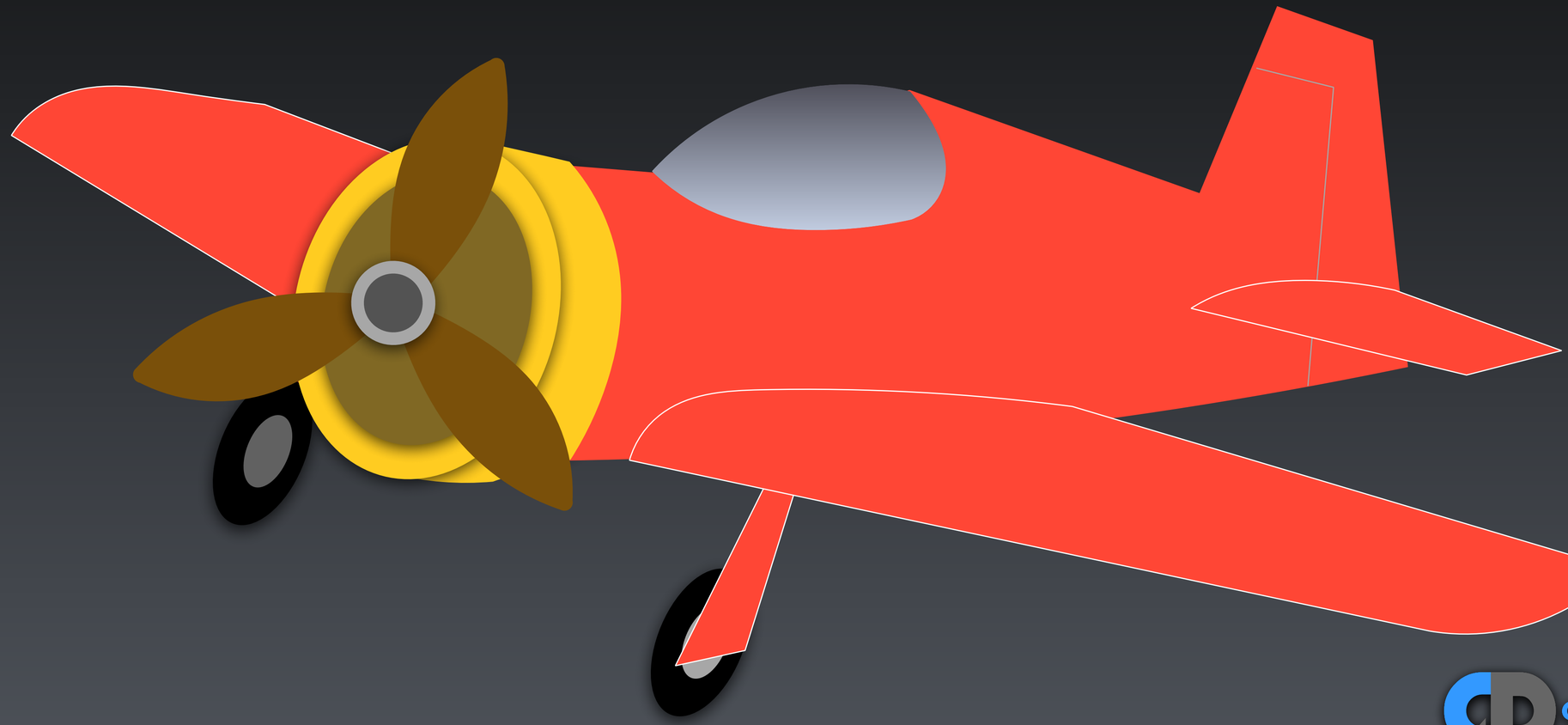
Continuous Delivery ltd

Experimental



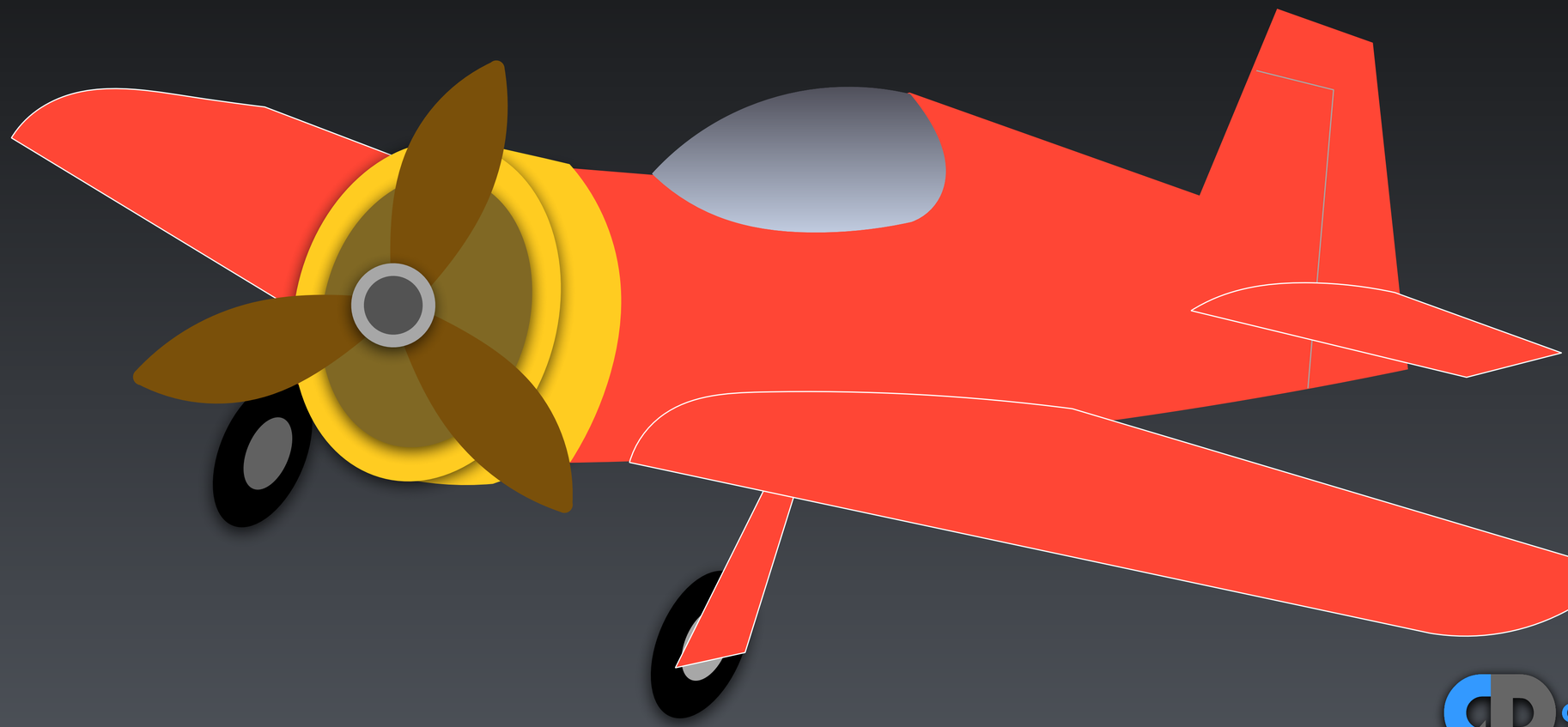
Continuous Delivery ltd

Experimental



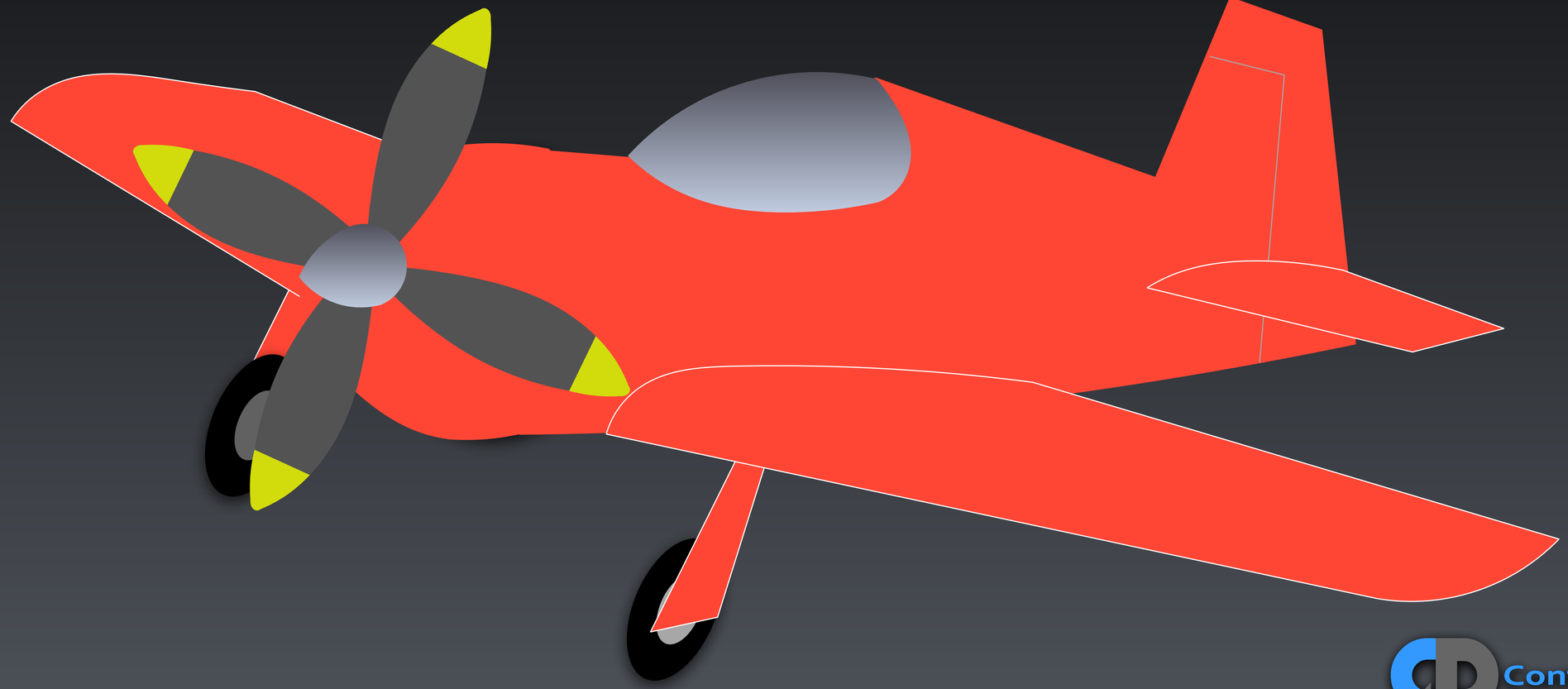
Continuous Delivery ltd

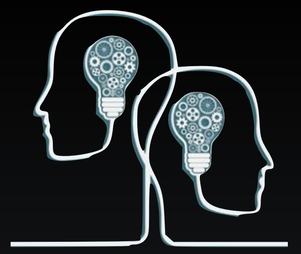
Experimental



Continuous Delivery ltd

Experimental

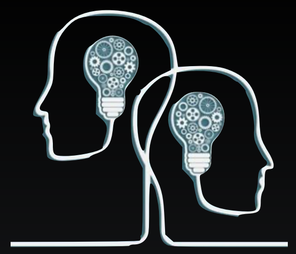




Continuous Delivery ltd


Continuous
Delivery ltd

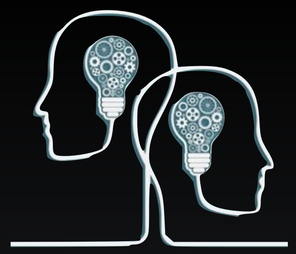
Empirical



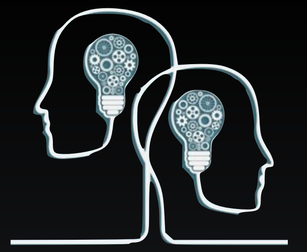
Continuous Delivery ltd

Empirical





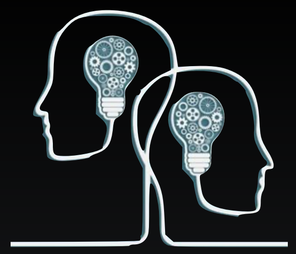
Continuous Delivery ltd



Continuous Delivery ltd

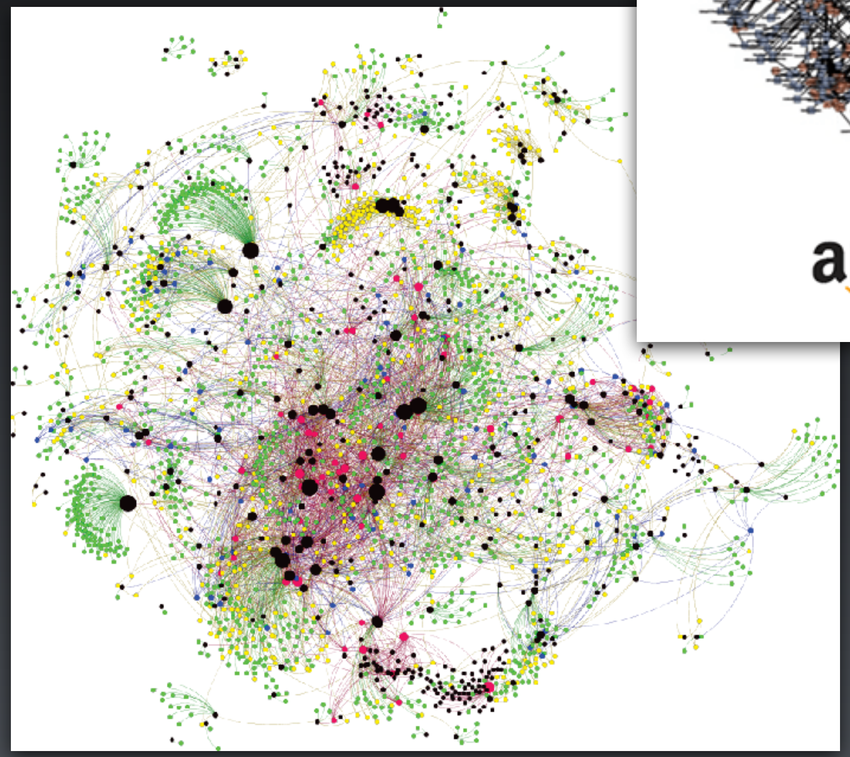
Empirical



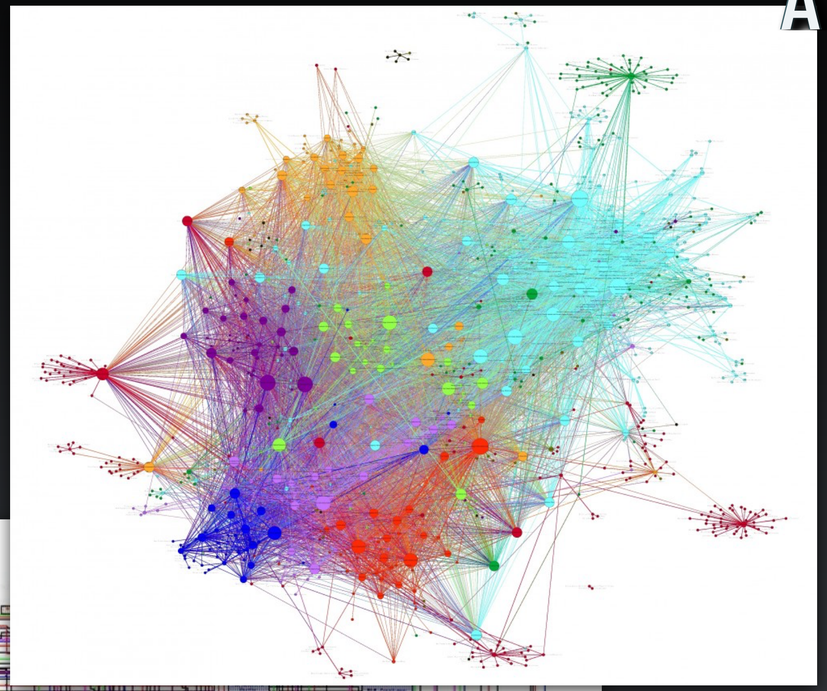


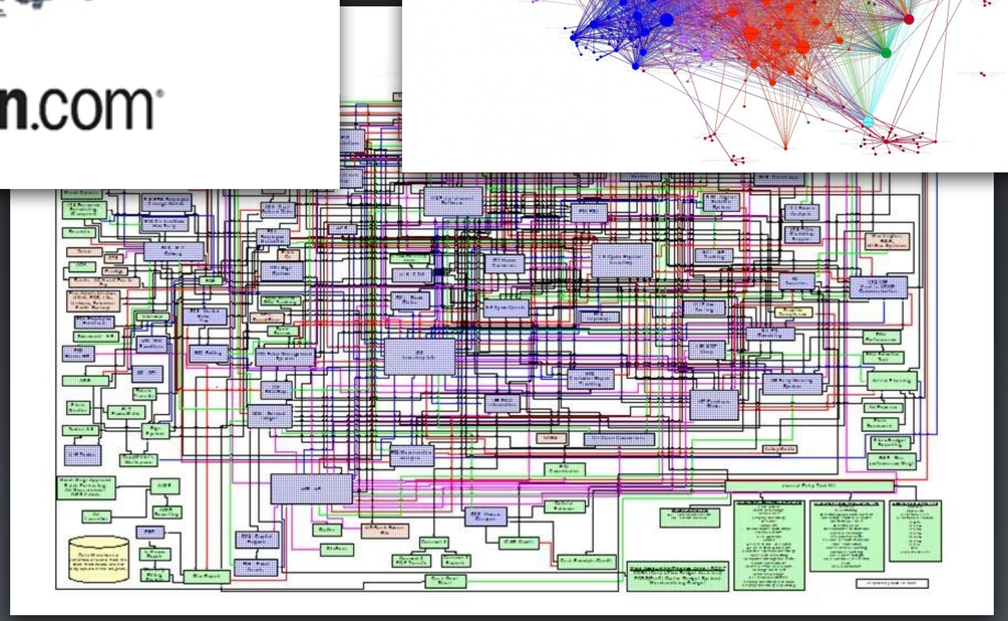
Continuous Delivery ltd


Continuous
Delivery ltd









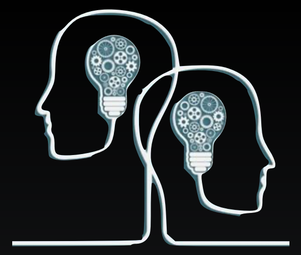
A B
A B
Continuous Delivery ltd



Modern Software is Often Complex!

A B
A B
Continuous Delivery ltd


Continuous
Delivery ltd





Software Development is about Learning & Managing Complexity

Continuous Delivery ltd



Managing Complexity

Continuous Delivery ltd



Managing Complexity

Modularity

Cohesion

Separation of Concerns

Abstraction

Coupling

Continuous Delivery ltd


Continuous
Delivery ltd

Modularity



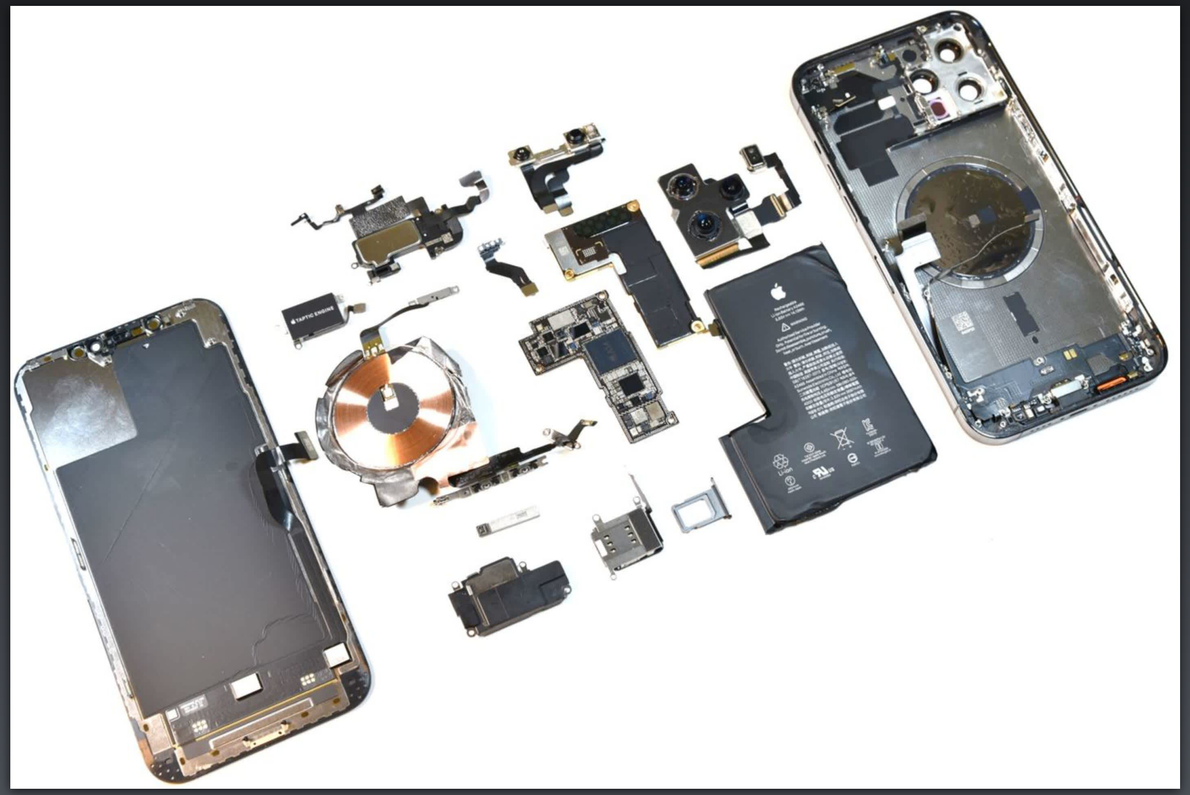
Continuous Delivery ltd

Modularity



Continuous Delivery ltd

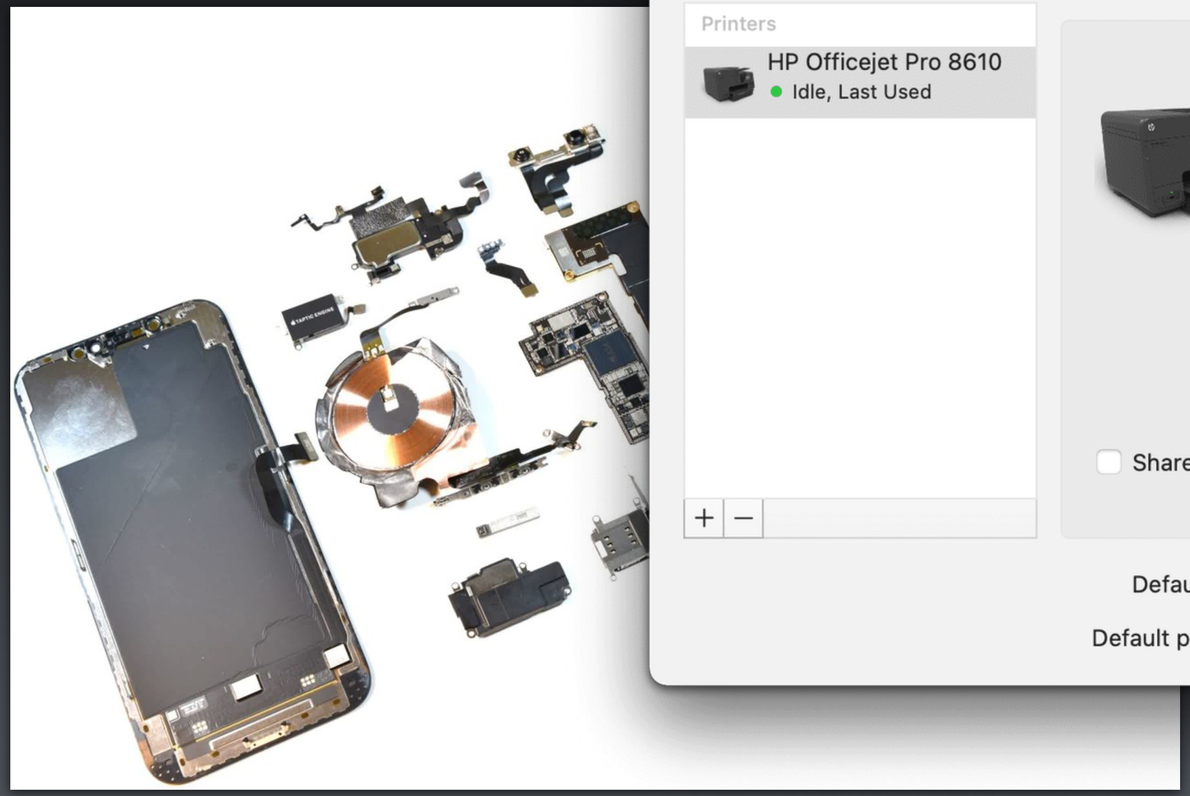
Modularity

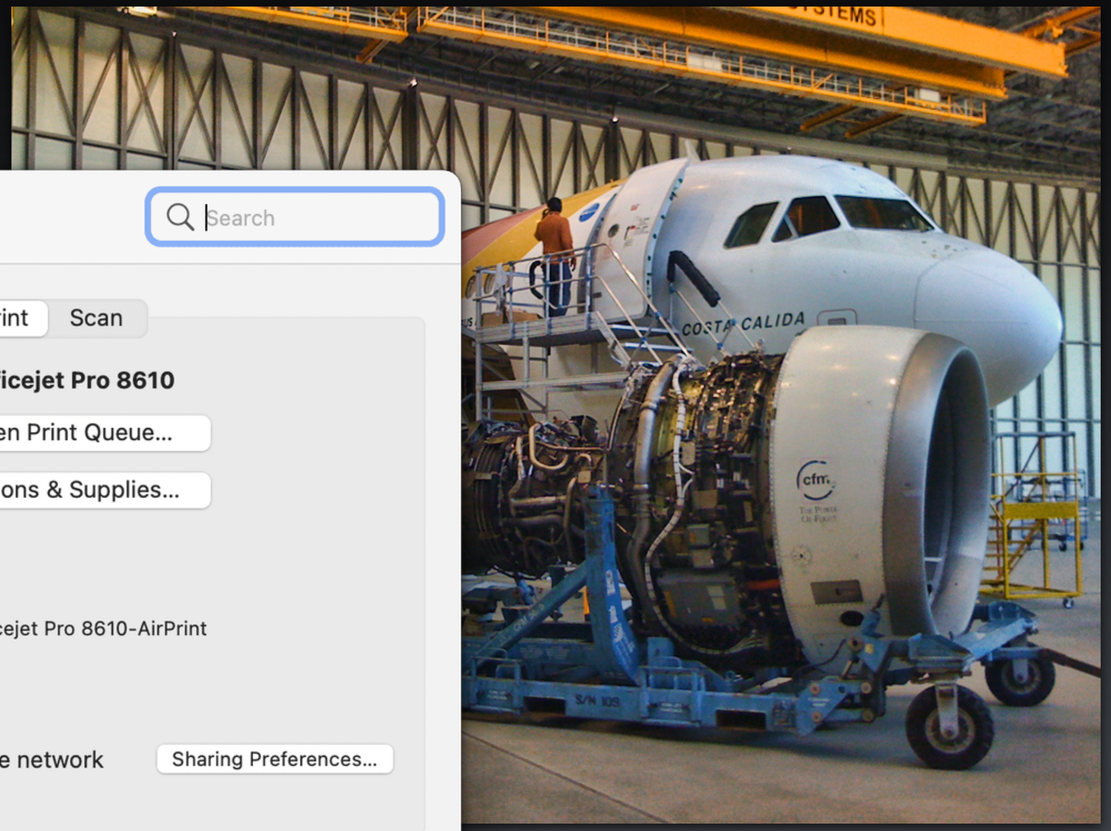


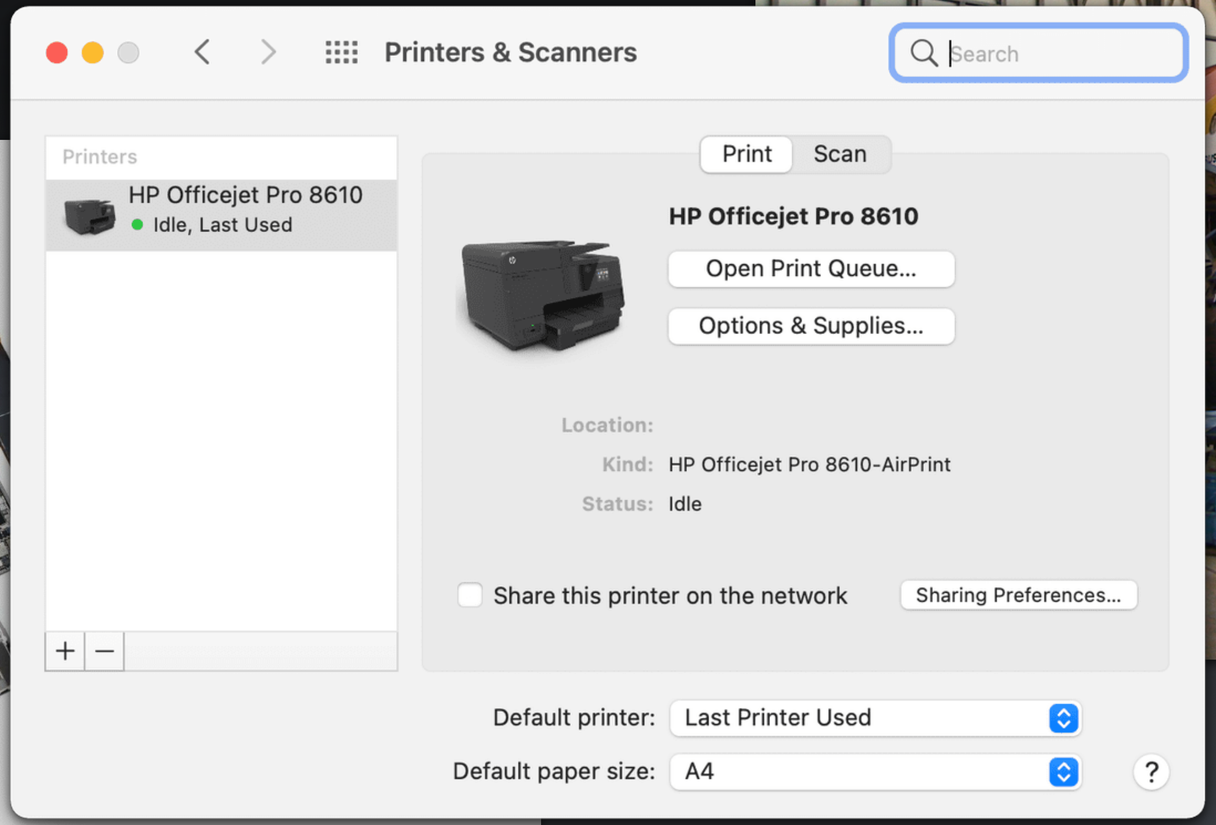


Continuous Delivery ltd

Modularity






Printers & Scanners
Search
Printers
HP Officejet Pro 8610
Idle, Last Used
Print
Scan
HP Officejet Pro 8610
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




```
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




```
60 ▼    if (unlikely(!ci)) {
61          struct ecm_db_mapping_instance *src_
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



Continuous Delivery ltd

Modularity



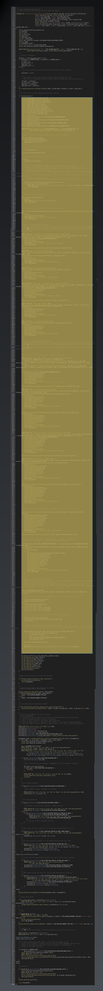

```
 * ecm_nss_non_ported_ipv4_process()
 *  Process a protocol that does not have ported identifiers
 */
unsigned int ecm_nss_non_ported_ipv4_process(struct net_device *out_dev, struct net_device *out_dev_nat,
                            struct net_device *in_dev, struct net_device *in_dev_nat,
                            uint8_t *src_node_addr, uint8_t *src_node_addr_nat,
                            uint8_t *dest_node_addr, uint8_t *dest_node_addr_nat,
                            bool can_accel, bool is_routed, bool is_l2_encap, struct sk_buff *skb,
                            struct ecm_tracker_ip_header *ip_hdr,
                            struct nf_conn *ct, ecm_tracker_sender_type_t sender, ecm_db_direction_t ecm_dir,
                            struct nf_conntrack_tuple *orig_tuple, struct nf_conntrack_tuple *reply_tuple,
                            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)
{
    struct ecm_db_connection_instance *ci;
    int protocol;
    int src_port;
    int src_port_nat;
    int dest_port;
    int dest_port_nat;
    ip_addr_t match_addr;
    struct ecm_classifier_instance *assignments[ECM_CLASSIFIER_TYPES];
    int aci_index;
    int assignment_count;
    ecm_db_timer_group_t ci_orig_timer_group;
    struct ecm_classifier_process_response prevalent_pr;

    DEBUG_TRACE("Non-ported protocol src: " ECM_IP_ADDR_DOT_FMT ", dest: " ECM_IP_ADDR_DOT_FMT "\n",
                ECM_IP_ADDR_TO_DOT(ip_src_addr), ECM_IP_ADDR_TO_DOT(ip_dest_addr));

    /*
```



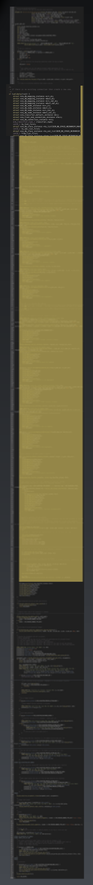
Modularity





Continuous Delivery ltd

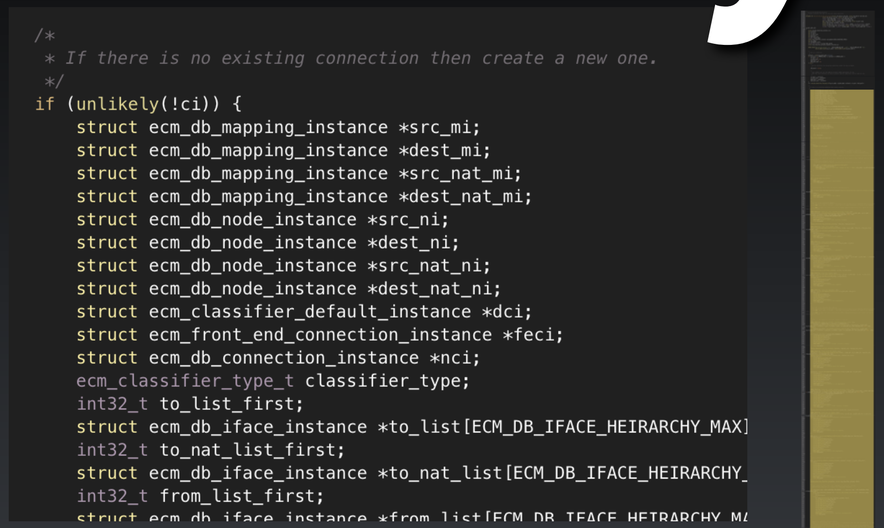
Modularity





Continuous Delivery ltd

Modularity



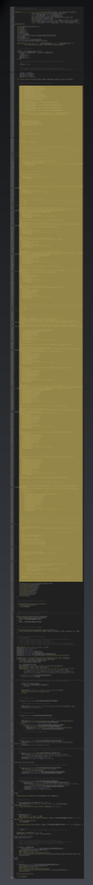
```
/*
 * If there is no existing connection then create a new one.
 */
if (unlikely(!ci)) {
    struct ecm_db_mapping_instance *src_mi;
    struct ecm_db_mapping_instance *dest_mi;
    struct ecm_db_mapping_instance *src_nat_mi;
    struct ecm_db_mapping_instance *dest_nat_mi;
    struct ecm_db_node_instance *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 *feci;
    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_MA
```






Modularity

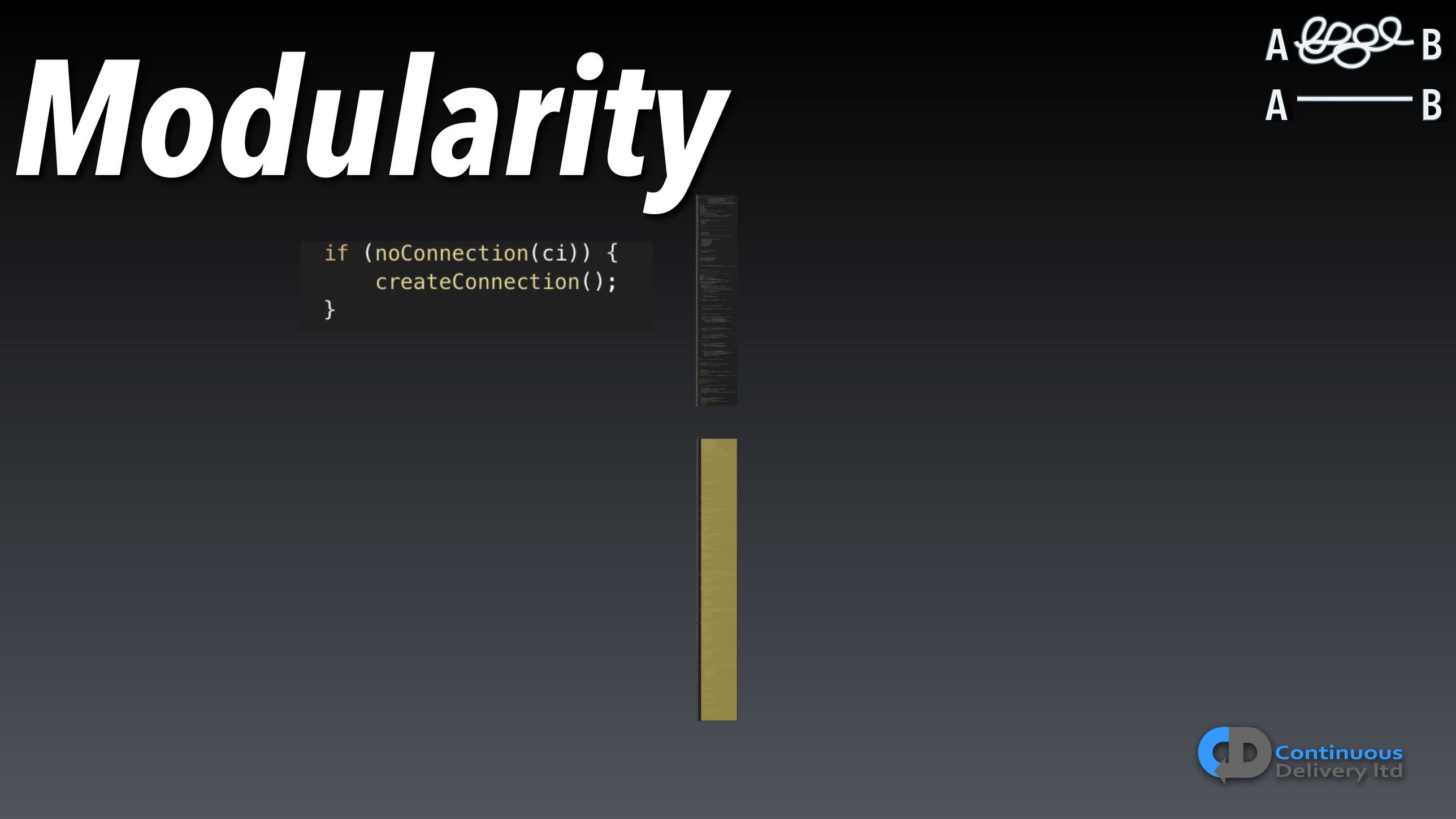
```
if (noConnection(ci)) {
    createConnection();
}
```





Continuous Delivery ltd

Modularity




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


Continuous Delivery ltd


Continuous
Delivery ltd



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

- Kent Beck

Continuous Delivery ltd



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

- Kent Beck

Continuous Delivery ltd



Modularity

things that aren't related further apart

Cohesion

things that are related closer together

Continuous Delivery ltd

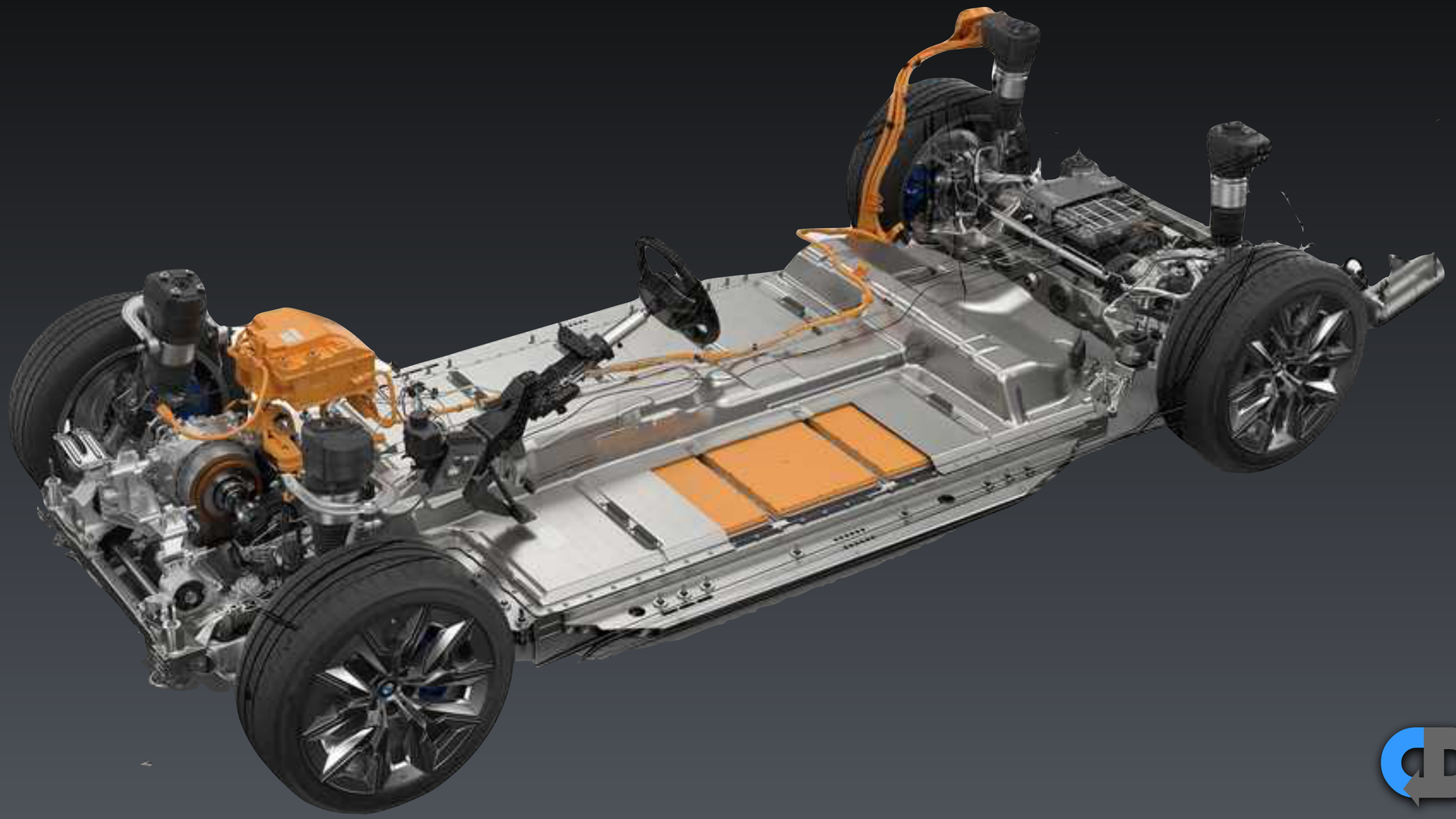

Continuous
Delivery ltd

Separation of Concerns



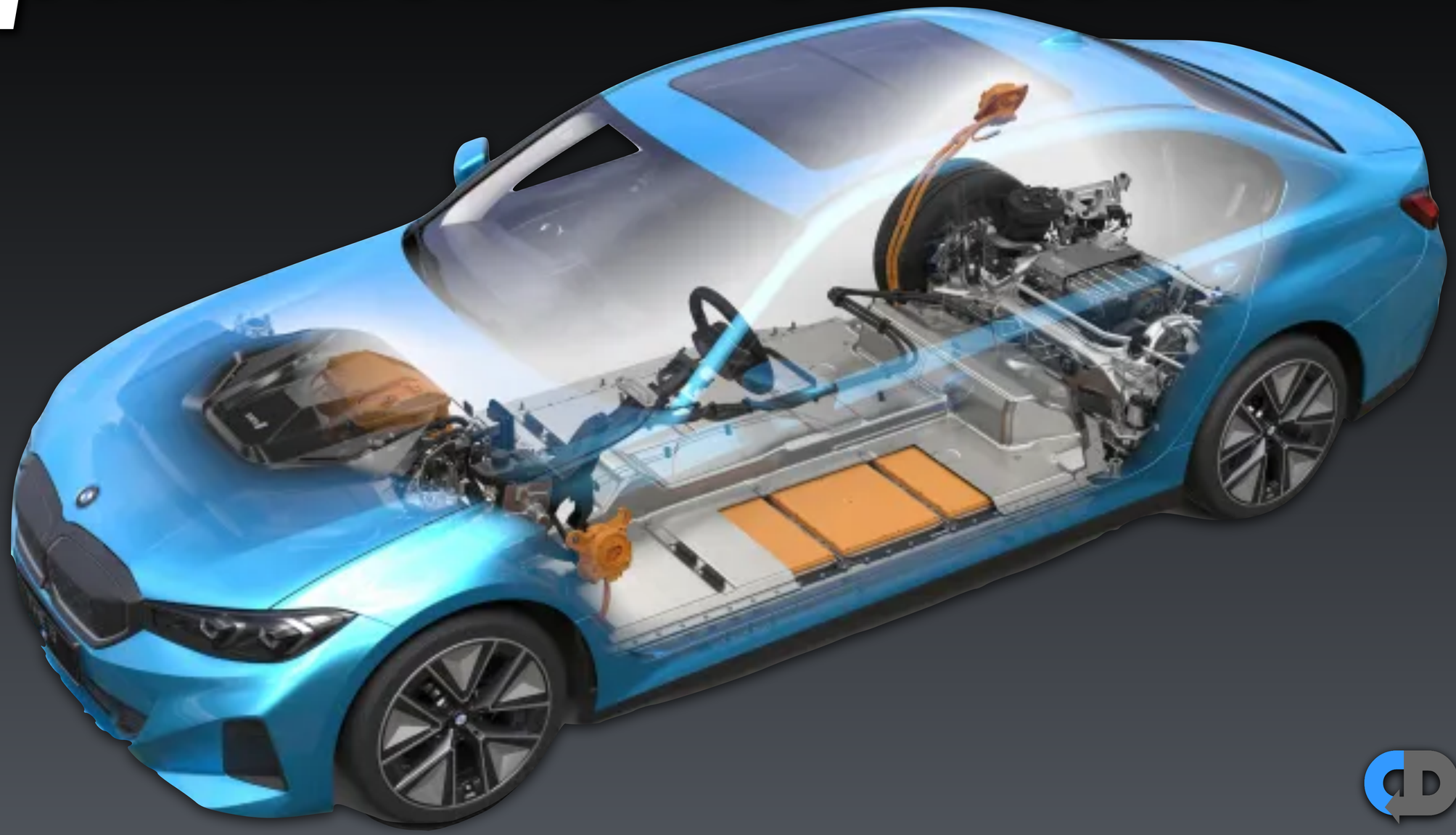
Continuous Delivery ltd

Separation of Concerns



Continuous Delivery ltd

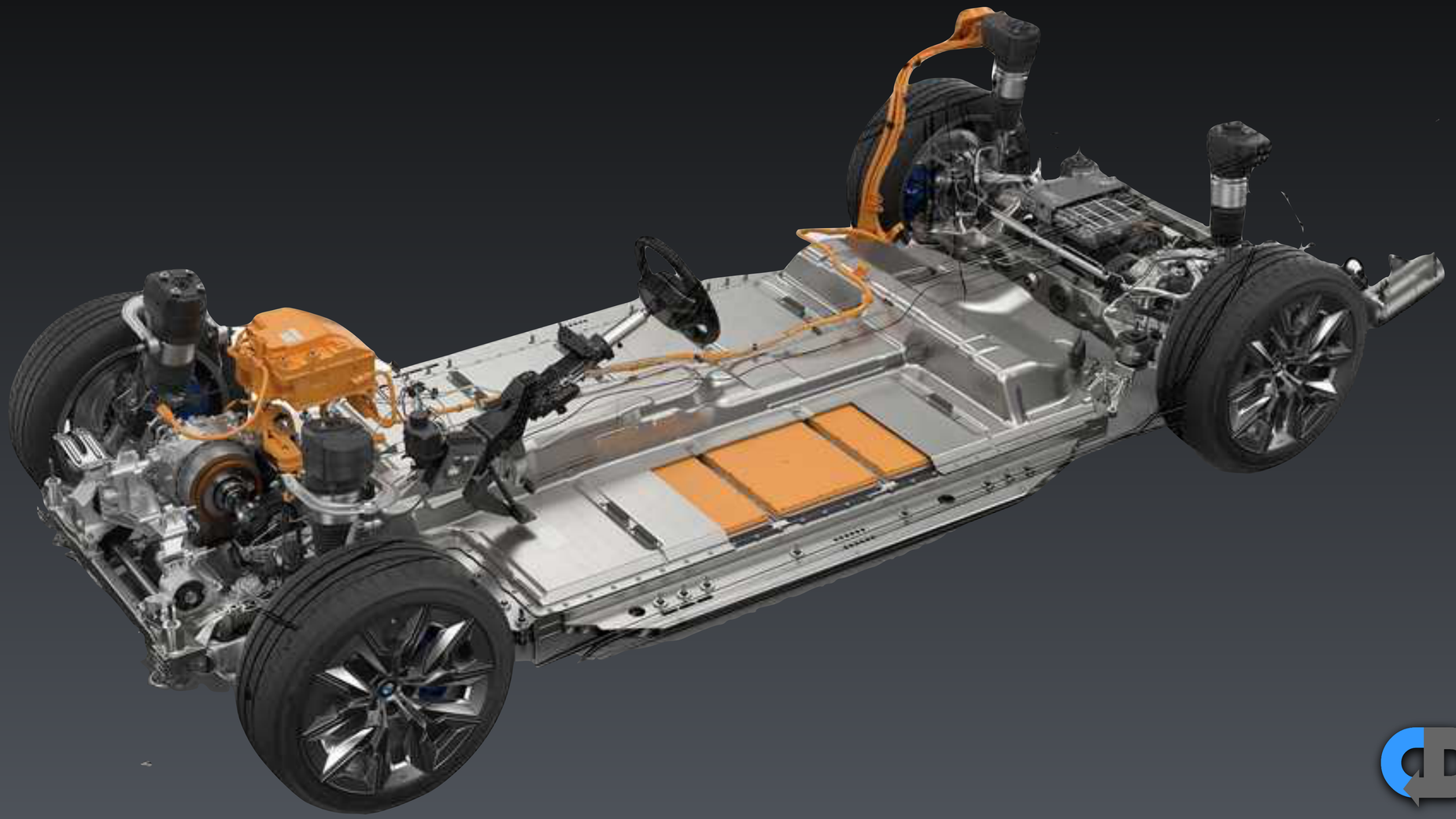
Separation of Concerns





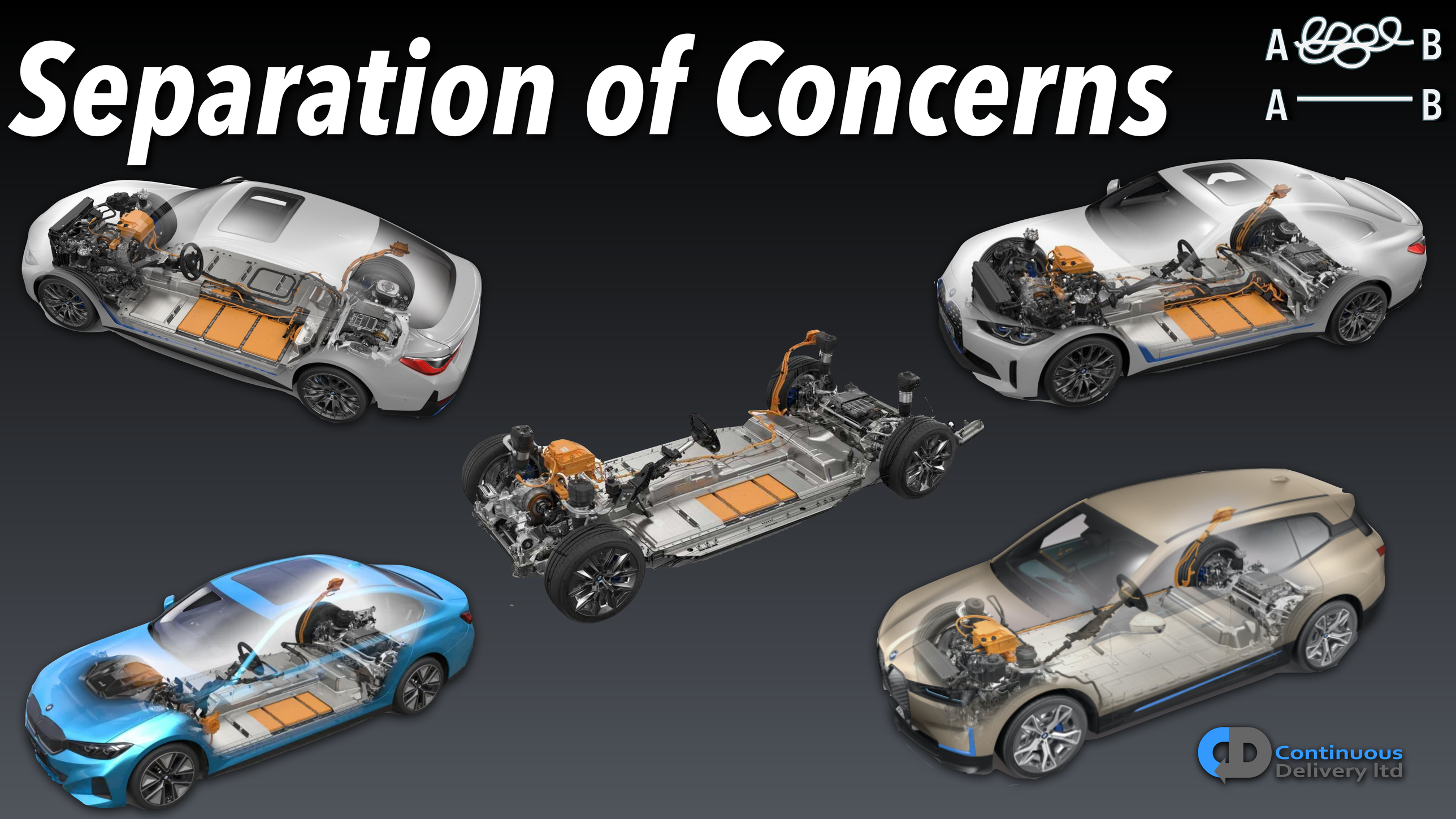
Continuous Delivery ltd

Separation of Concerns



Continuous Delivery ltd

Separation of Concerns



Separation of Concerns



Continuous Delivery ltd

Separation of Concerns



Focus & Clarity

Continuous Delivery ltd

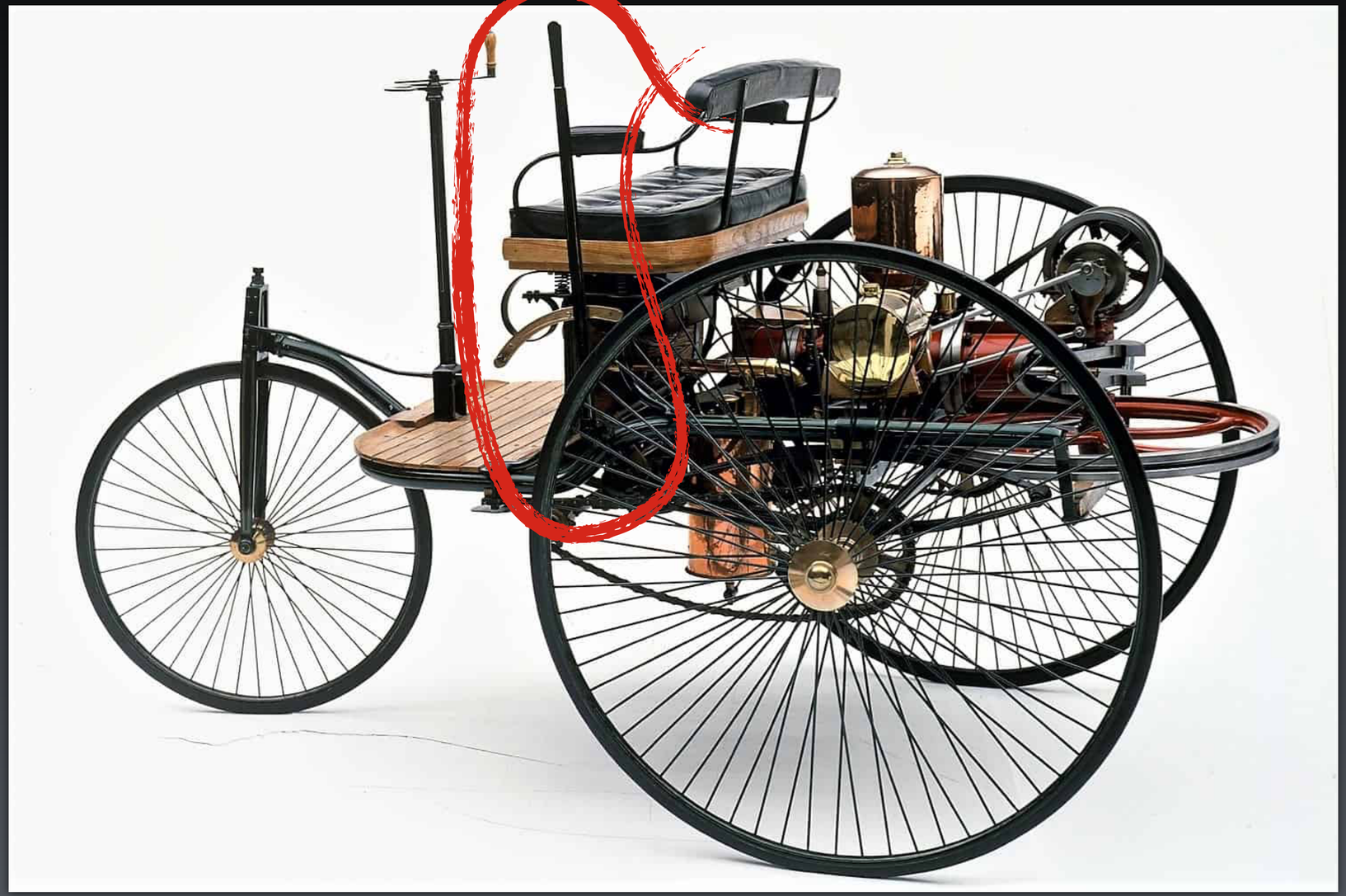

Continuous
Delivery ltd

Abstraction



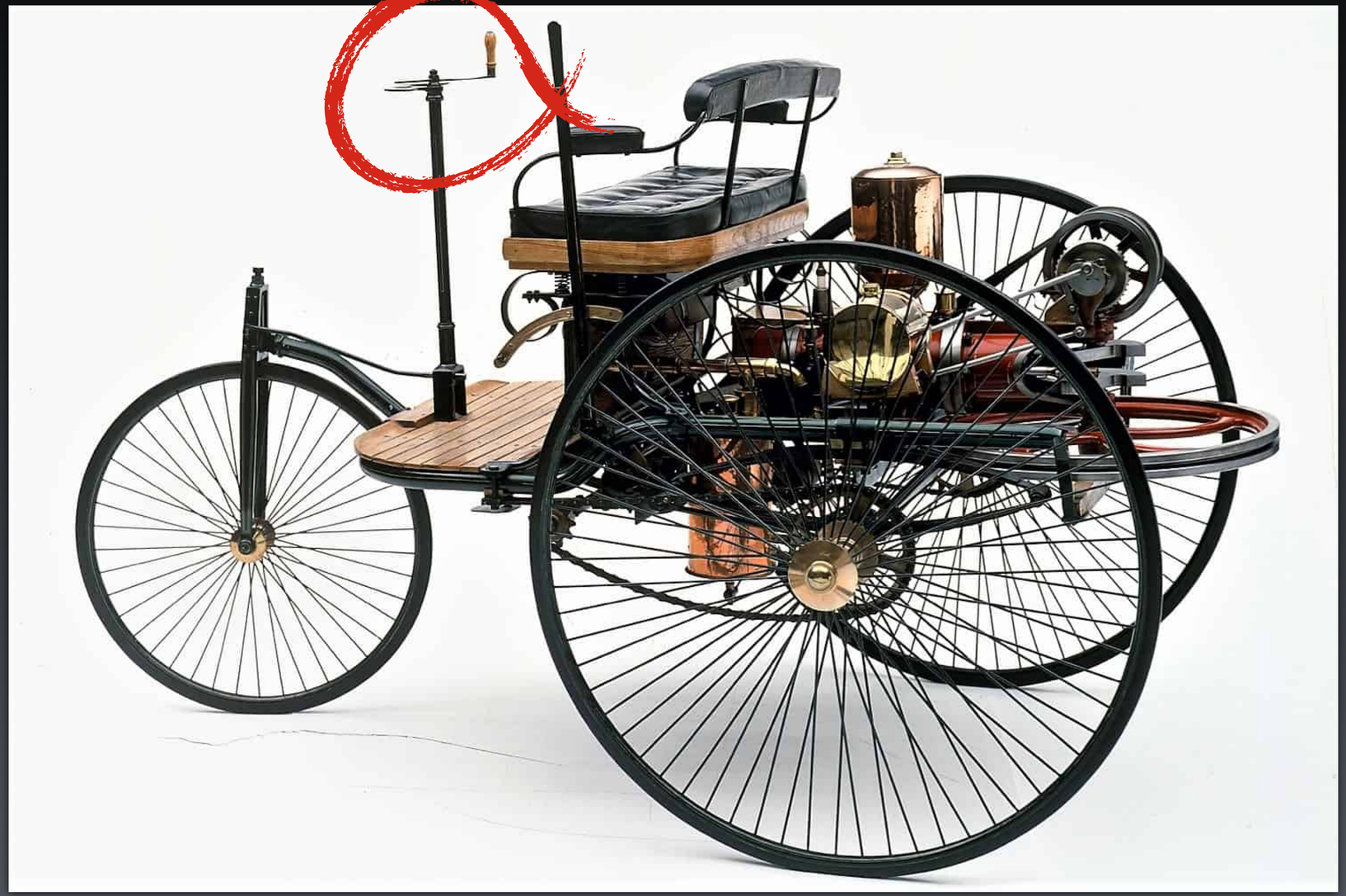
Continuous Delivery ltd

Abstraction



A B
A B
Continuous Delivery ltd

Abstraction



A B
A B

Continuous Delivery ltd

Abstraction



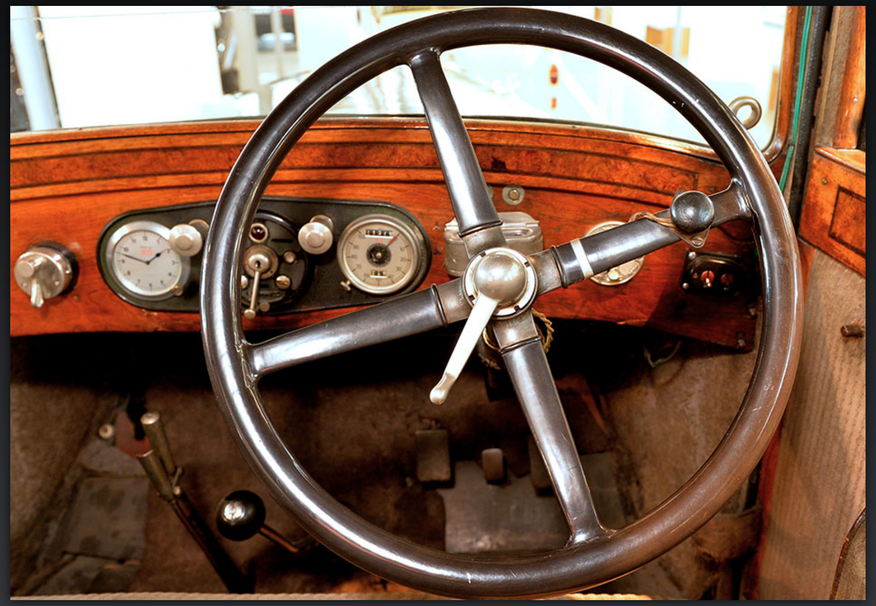
Continuous Delivery ltd

Abstraction



Continuous Delivery ltd

Abstraction





Continuous Delivery ltd

Abstraction











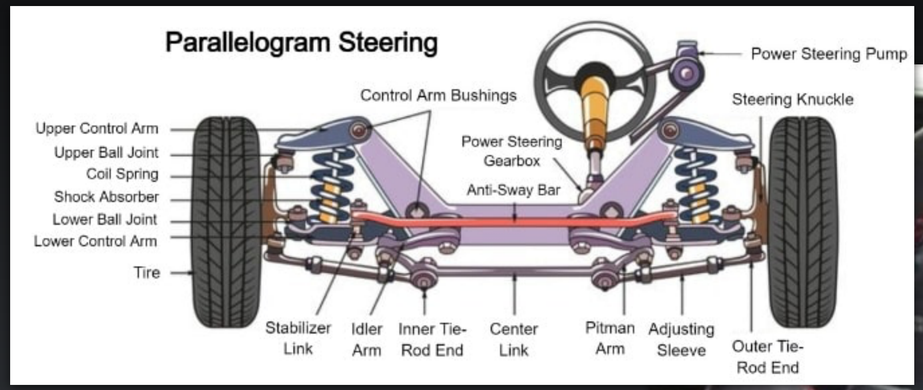
Continuous Delivery ltd

Abstraction

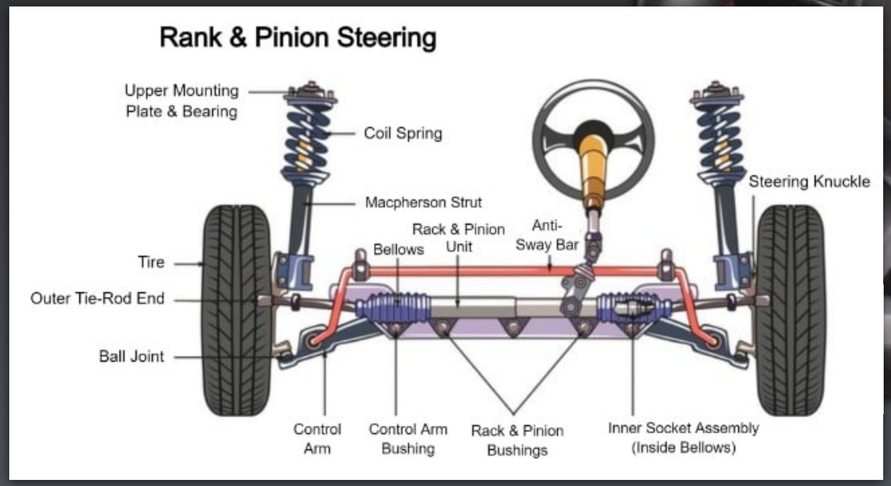


Source: engineeringlearn.com

Abstraction

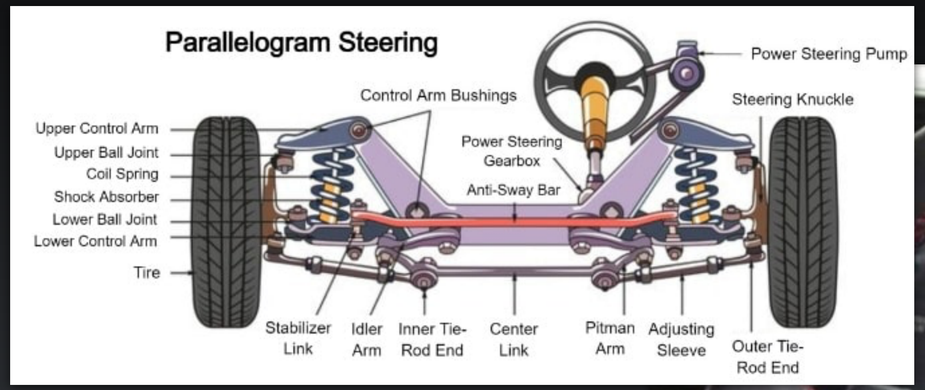




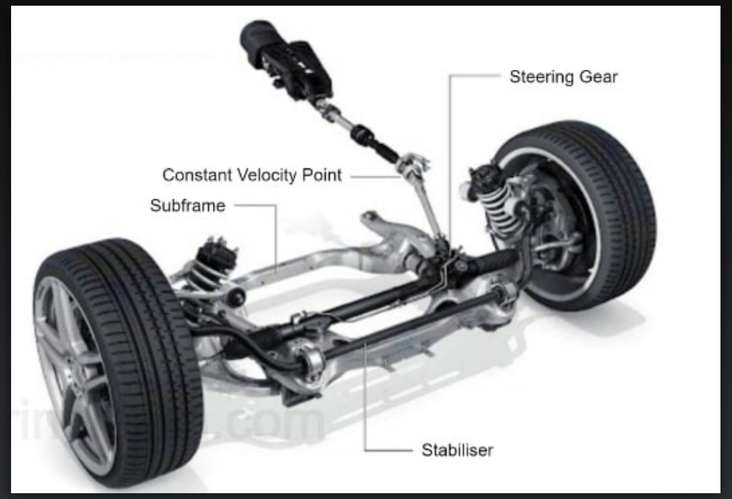


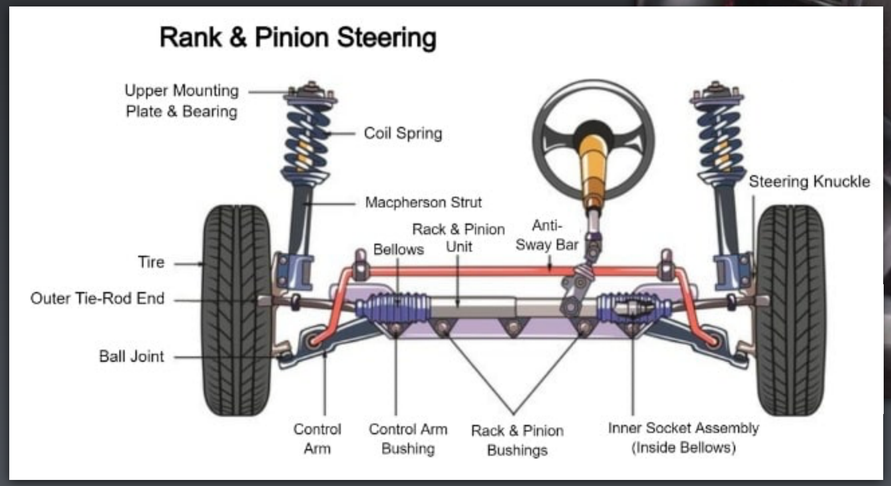
Source: engineeringlearn.com

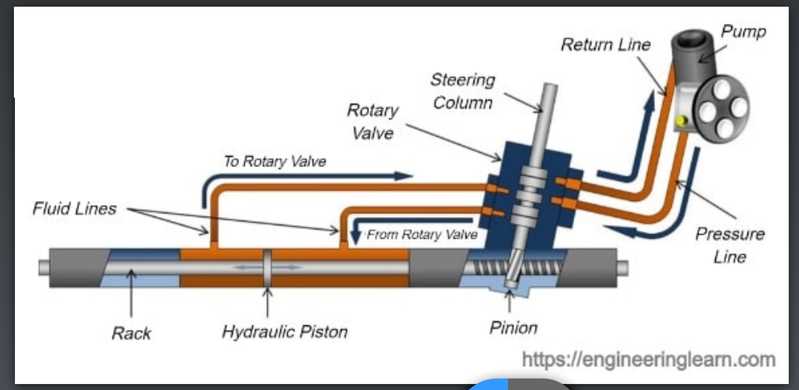
Abstraction











Source: engineeringlearn.com

Abstraction



Continuous Delivery ltd

Abstraction





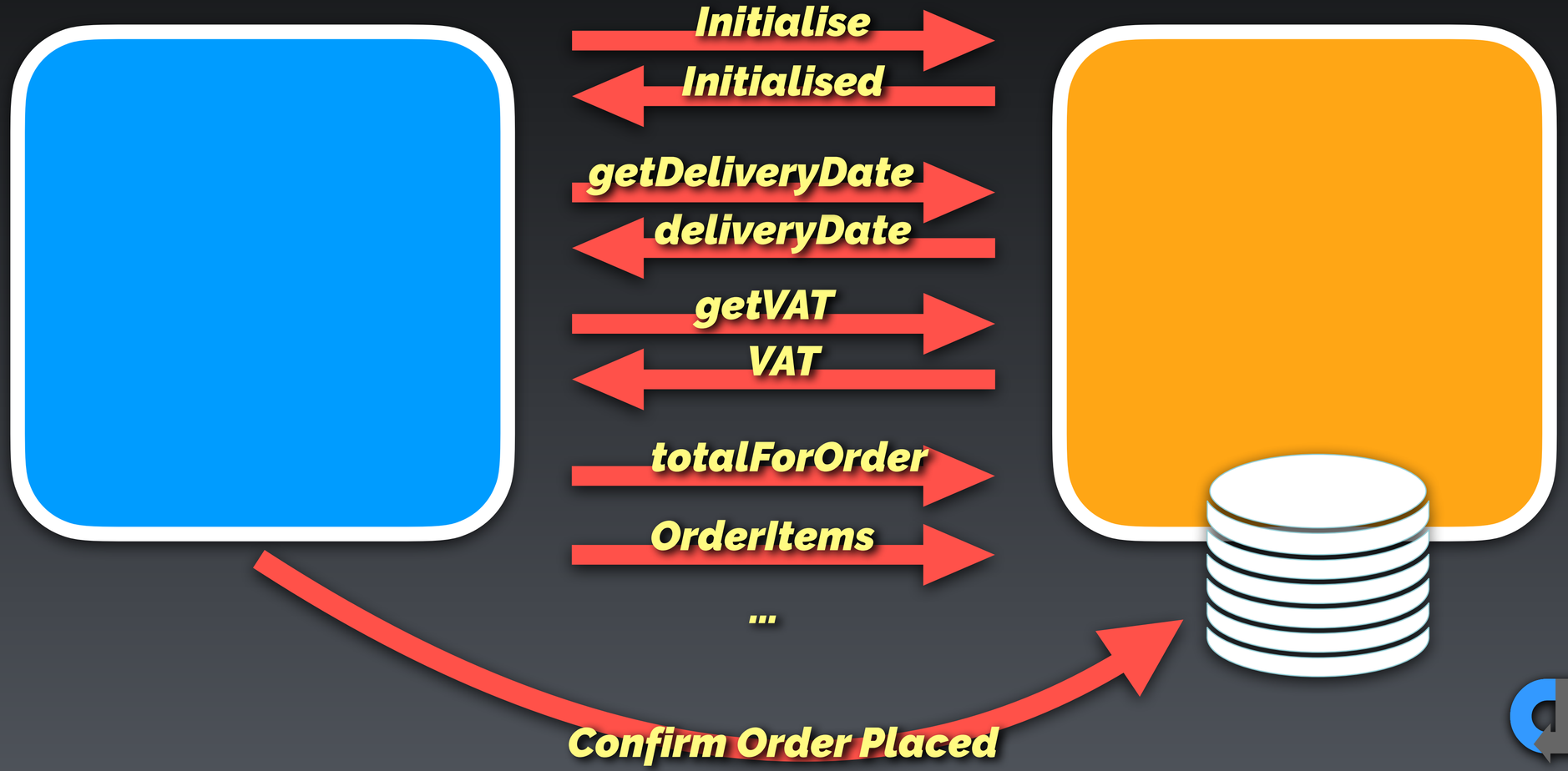
Continuous Delivery ltd

Abstraction



Continuous Delivery ltd

Abstraction

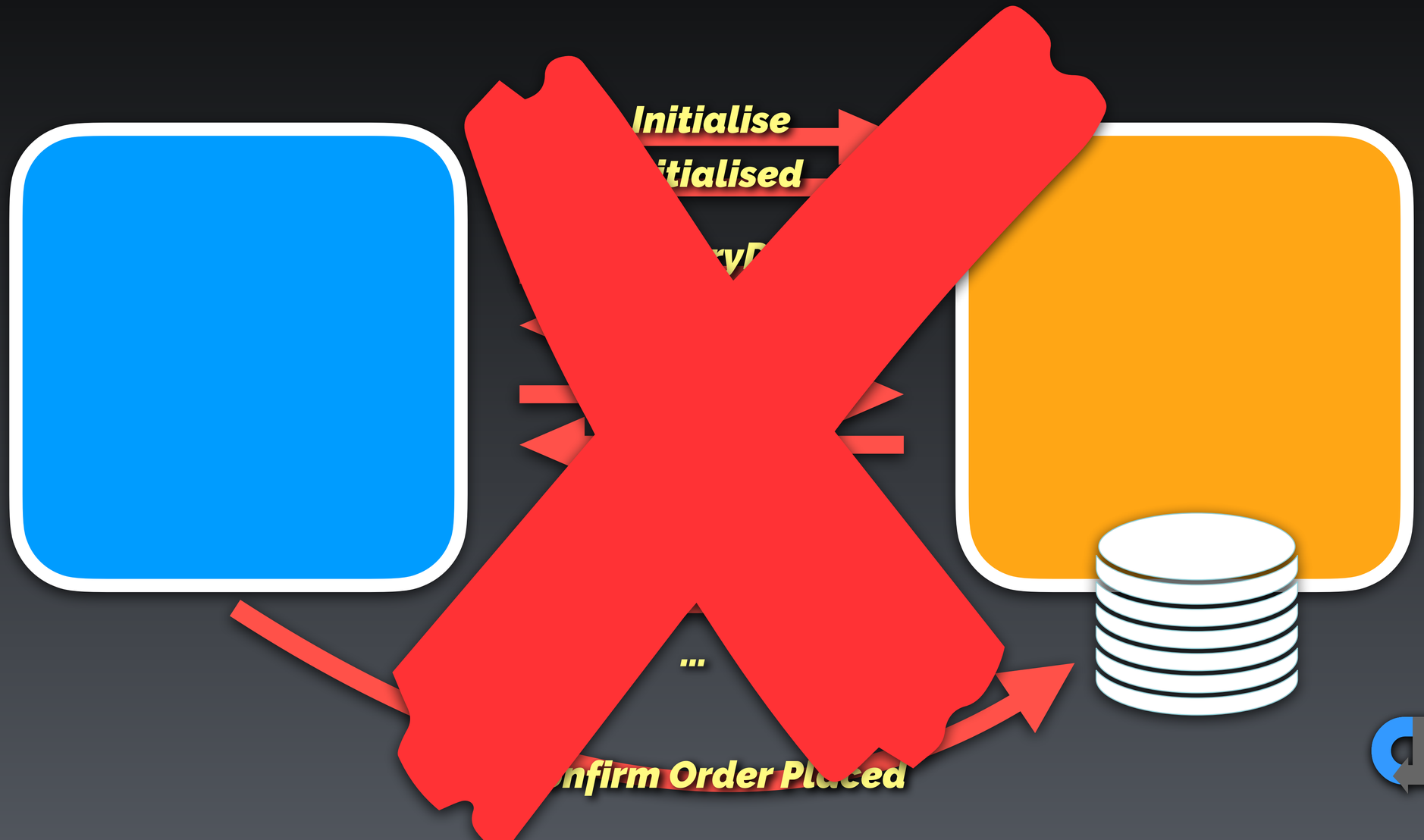


Abstraction



Continuous Delivery ltd

Abstraction



Continuous Delivery ltd

Abstraction



Continuous Delivery ltd

Abstraction



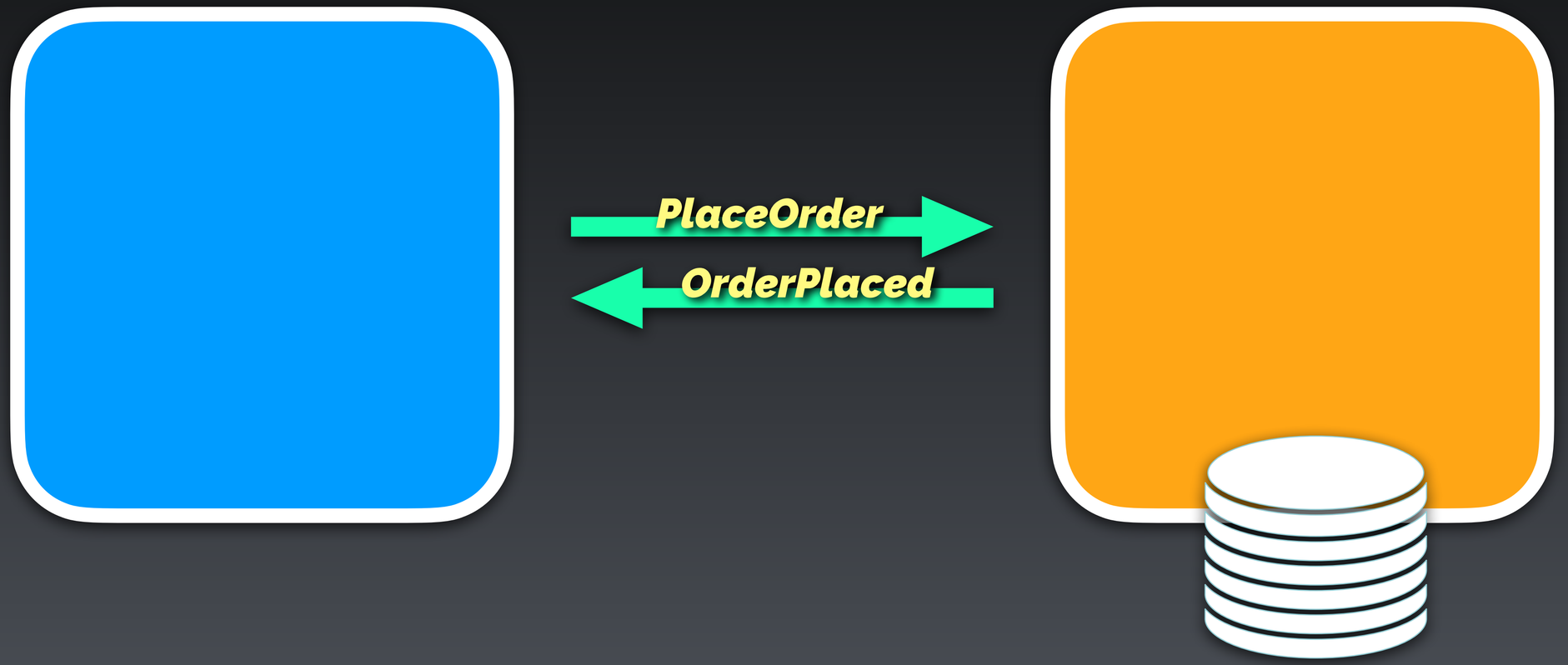
Continuous Delivery ltd

Abstraction





Abstraction



Continuous Delivery ltd

Abstraction




Continuous
Delivery ltd

Coupling







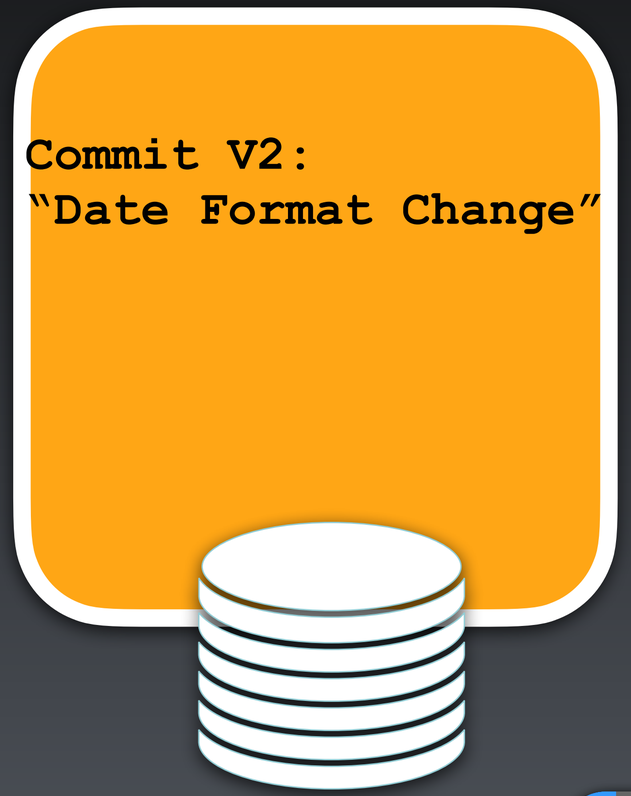
Continuous Delivery ltd



Continuous Delivery ltd

Coupling



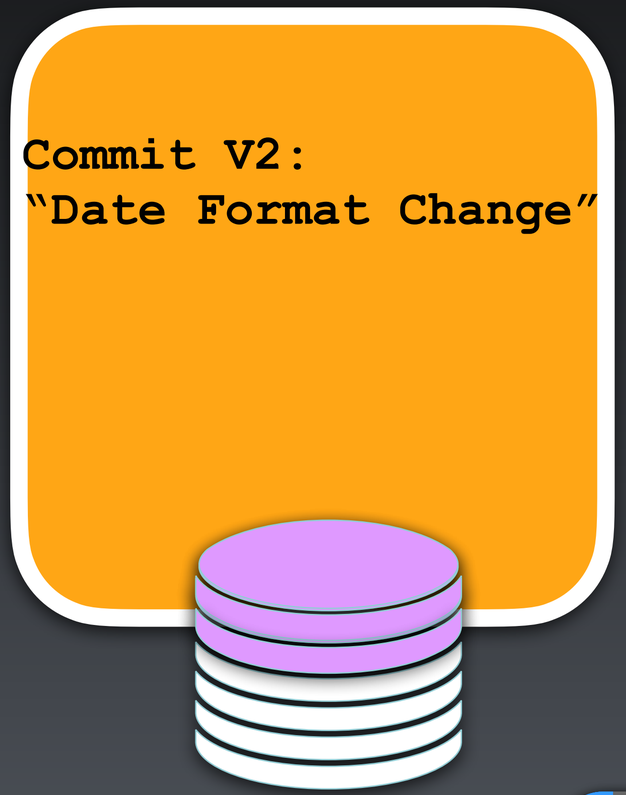


Continuous Delivery ltd

Coupling





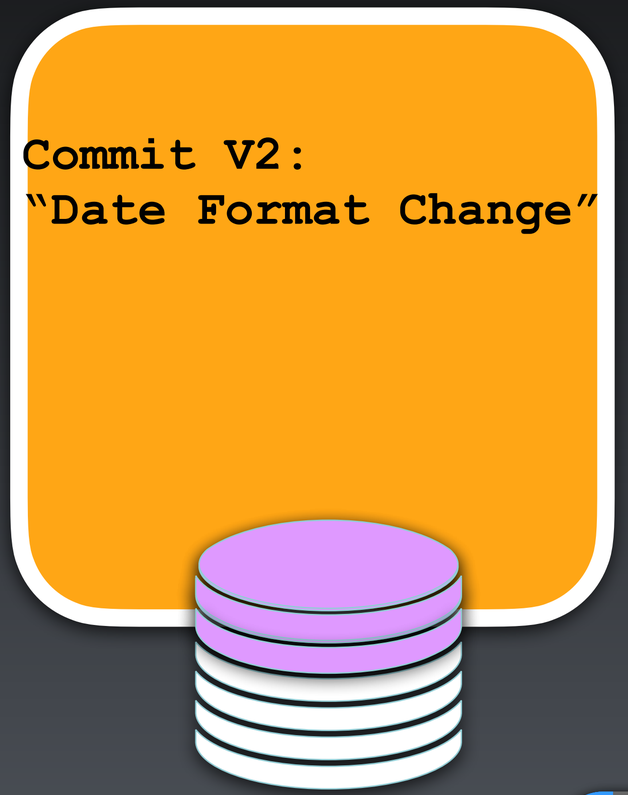


Continuous Delivery ltd

Coupling







Continuous Delivery ltd

Coupling



Continuous Delivery ltd

Coupling



Continuous Delivery ltd

Coupling



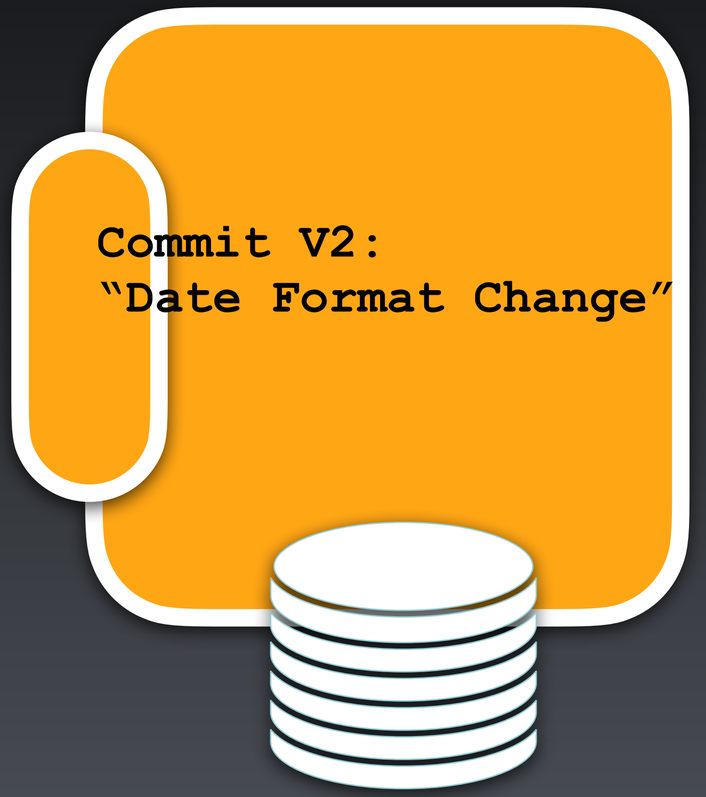




Continuous Delivery ltd

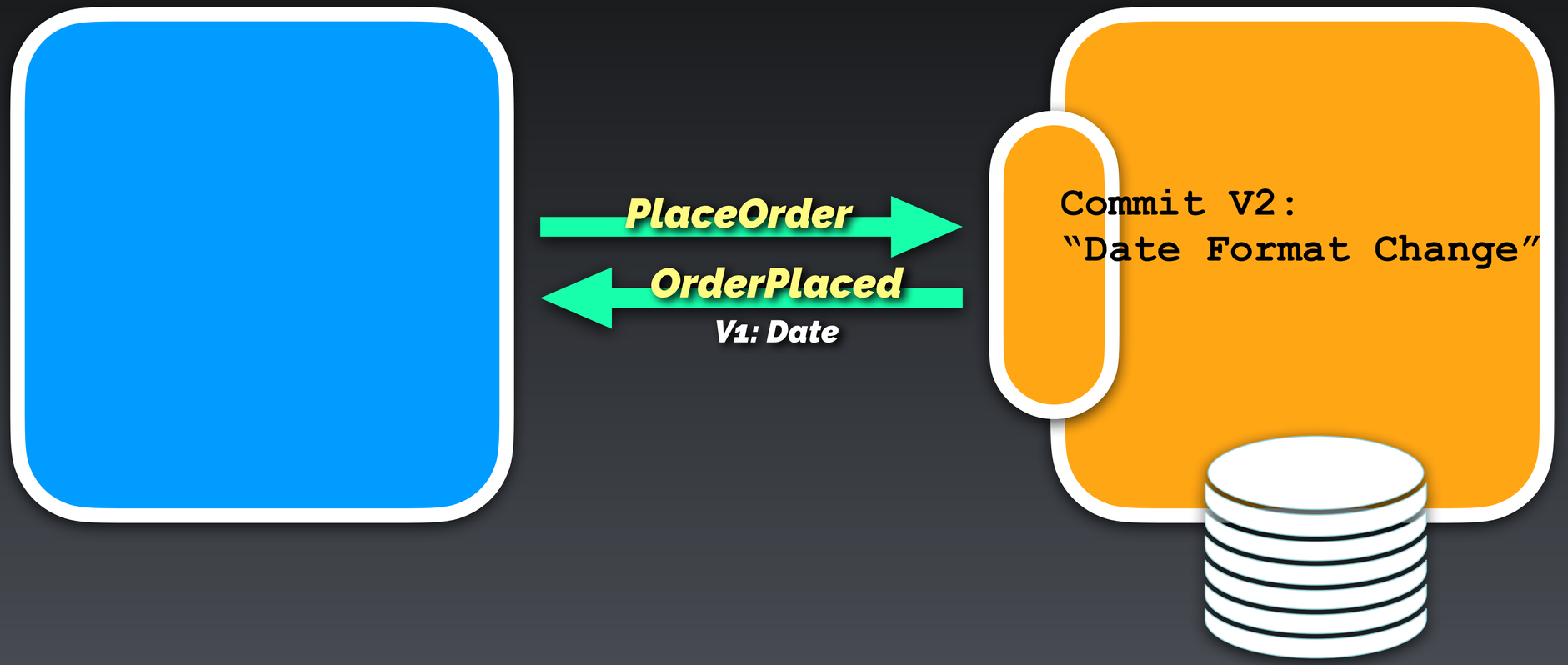
Coupling





Continuous Delivery ltd

Coupling



Continuous Delivery ltd

Applying the Guidelines

Continuous Delivery ltd

Managing Complexity

Continuous Delivery ltd

Managing Complexity

Modularity

Cohesion

Separation of Concerns

Abstraction

Coupling

Continuous Delivery ltd

Managing Complexity

Modularity

Cohesion

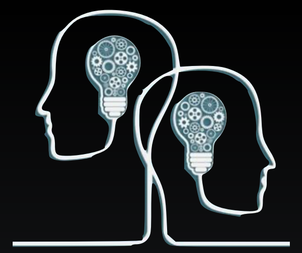
Separation of Concerns

Abstraction

Coupling

Just Motherhood & Apple Pie?

Continuous Delivery ltd



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

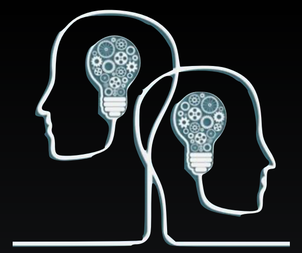
    public void start() {
        putIntoNeutral();
        applyBrakes();
        this.engine.start();
    }

    private void applyBrakes() {
    }

    private void putIntoNeutral() {
    }
}
```



Continuous Delivery ltd



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

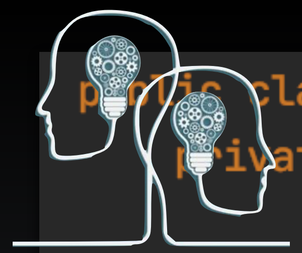
    public void start() {
        putIntoNeutral();
        applyBrakes();
        this.engine.start();
    }

    private void applyBrakes() {
    }

    private void putIntoNeutral() {
    }
}
```



Continuous Delivery ltd



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

    public void start() {
        putIntoNeutral();
        applyBrakes();
        this.engine.start();
    }

    private void applyBrakes() {
    }

    private void putIntoNeutral() {
    }
}
```

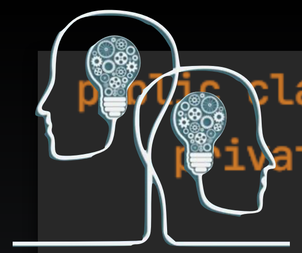
```
@Test
public void shouldStartCarEngine() {
    Car car = new Car();

    car.start();

    // Nothing to assert!!
}
```



Continuous Delivery ltd



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

    public void start() {
        putIntoNeutral();
        applyBrakes();
        this.engine.start();
    }

    private void applyBrakes() {
    }

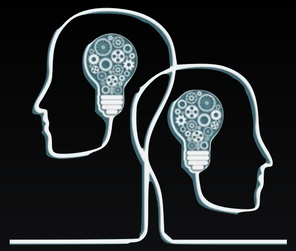
    private void putIntoNeutral() {
    }
}
```

```java
@Test
public void shouldStartCarEngine() {
    Car car = new Car();

    car.start();

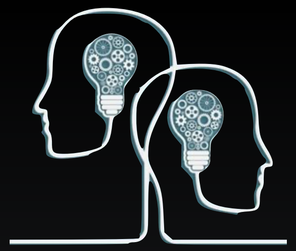
    // Nothing to assert!!
}
```

Continuous Delivery ltd



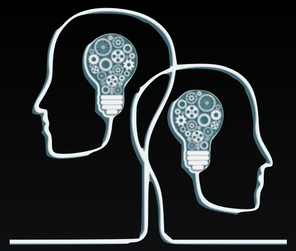


Continuous Delivery ltd





Continuous Delivery ltd

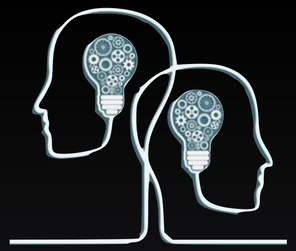





```
@Test
public void shouldStartBetterCarEngine() {


}
```

Continuous Delivery ltd

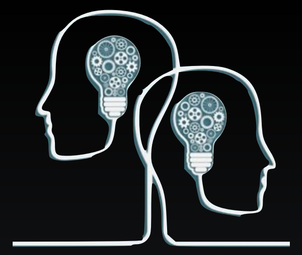





```
@Test
public void shouldStartBetterCarEngine() {


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

Continuous Delivery ltd



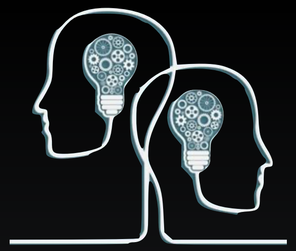



```
@Test
public void shouldStartBetterCarEngine() {


    car.start();

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

Continuous Delivery ltd





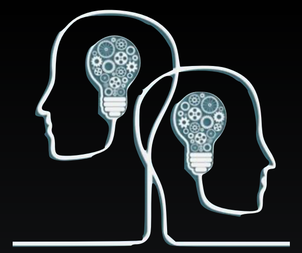

```
@Test
public void shouldStartBetterCarEngine() {

    BetterCar car = new BetterCar(engine);

    car.start();

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

Continuous Delivery ltd



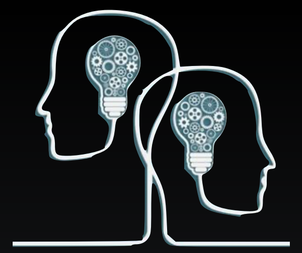



```
@Test
public void shouldStartBetterCarEngine() {
    FakeEngine engine = new FakeEngine();
    BetterCar car = new BetterCar(engine);

    car.start();

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

Continuous Delivery ltd



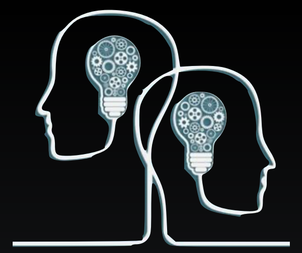
```
public class FakeEngine implements Engine {
    private boolean started = false;

    @Override
    public void start() {
        started = true;
    }

    public boolean startedSuccessfully() {
        return started;
    }
}
```



Continuous Delivery ltd



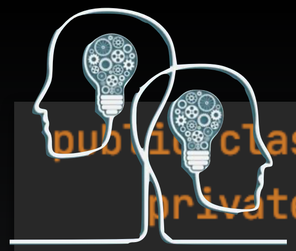



```
@Test
public void shouldStartBetterCarEngine() {
    FakeEngine engine = new FakeEngine();
    BetterCar car = new BetterCar(engine);

    car.start();

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

Continuous Delivery ltd



```
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() {
    }
}
```

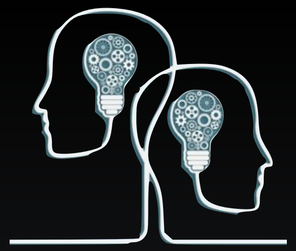



```
@Test
public void shouldStartBetterCarEngine() {
    FakeEngine engine = new FakeEngine();
    BetterCar car = new BetterCar(engine);

    car.start();

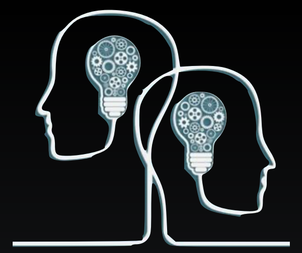
    assertTrue(engine.startedSuccessfully());
}
```

Continuous Delivery ltd





Continuous Delivery ltd



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

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

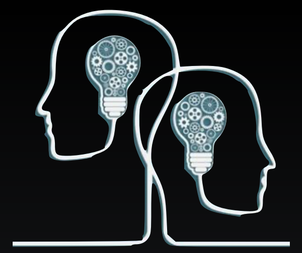
    public void start() {
        putIntoNeutral();
        applyBrakes();
        this.engine.start();
    }

    private void applyBrakes() {
    }

    private void putIntoNeutral() {
    }
}
```



Continuous Delivery ltd



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

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

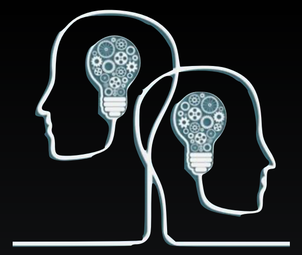
    public void start() {
        putIntoNeutral();
        applyBrakes();
        this.engine.start();
    }

    private void applyBrakes() {
    }

    private void putIntoNeutral() {
    }
}
```



Continuous Delivery ltd



A B

A B

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

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

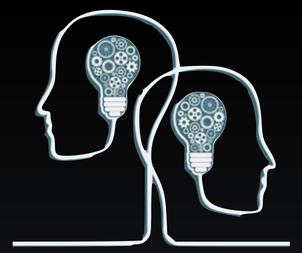
```java
public class Car {
    private final Engine engine = new PetrolEngine();
```

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

    private void applyBrakes() {
    }

    private void putIntoNeutral() {
    }
}
```

Continuous Delivery ltd



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

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

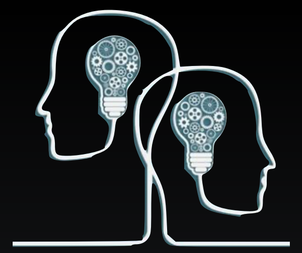
    public void start() {
        putIntoNeutral();
        applyBrakes();
        this.engine.start();
    }

    private void applyBrakes() {
    }

    private void putIntoNeutral() {
    }
}
```



Continuous Delivery ltd



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

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

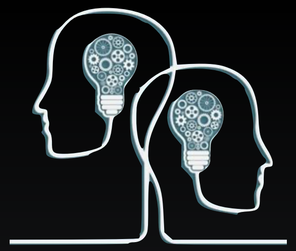
    public void start() {
        putIntoNeutral();
        applyBrakes();
        this.engine.start();
    }

    private void applyBrakes() {
    }

    private void putIntoNeutral() {
    }
}
```



Continuous Delivery ltd



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

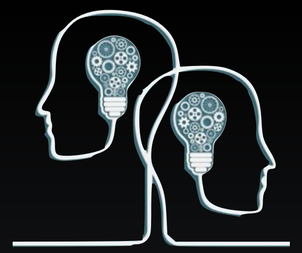
    public BetterCar(Engine engine) {
        this.engine = engine;
    }
```




```
public void createCars() {

}
```

Continuous Delivery ltd



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

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

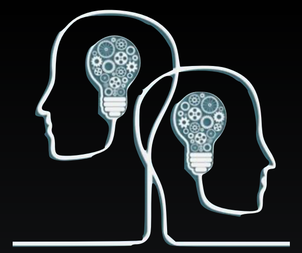



```
public void createCars() {

    BetterCar petrolCar = new BetterCar(new PetrolEngine());

}
```

Continuous Delivery ltd



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

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



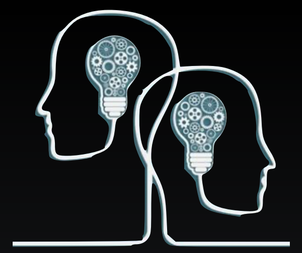

```
public void createCars() {

    BetterCar petrolCar = new BetterCar(new PetrolEngine());

    BetterCar electricCar = new BetterCar(new ElectricEngine());


}
```

Continuous Delivery ltd



```
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());

}
```

Continuous Delivery ltd


Continuous
Delivery ltd

The Tools of Our Trade!

Optimise for Learning

Iteration
Feedback
Incremental
Experimental
Empirical

Optimise to Manage Complexity

Modularity
Cohesion
Separation of Concerns
Abstraction
Coupling

Continuous Delivery ltd


Continuous
Delivery ltd

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

Continuous Delivery ltd

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

Continuous Delivery ltd

Embrace Change!

Continuous Delivery ltd

Embrace Change!

JS Frameworks Change All the Time!

Requirements Change

Technologies Change

Businesses Change

People Change

Plans Change

Markets Change

Our Designs Change

Customers Change

Our Understanding Changes

Continuous Delivery ltd


Continuous
Delivery ltd

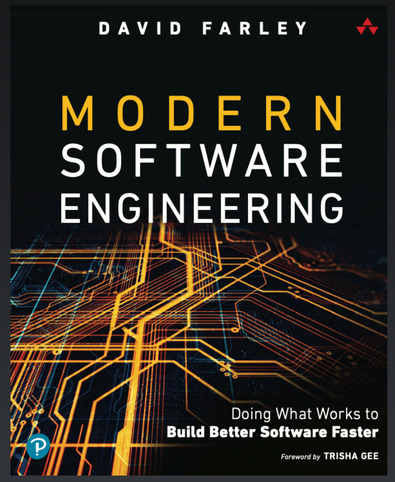
The Quality of a System is Defined by Our Ability to Change it!

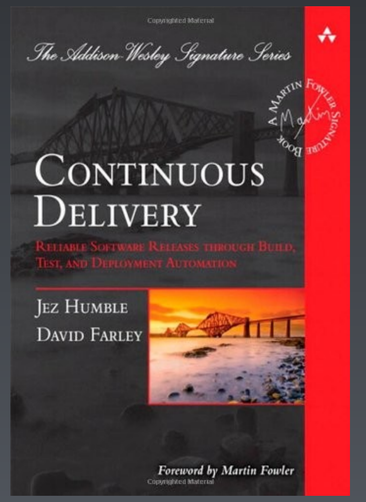
Continuous Delivery ltd

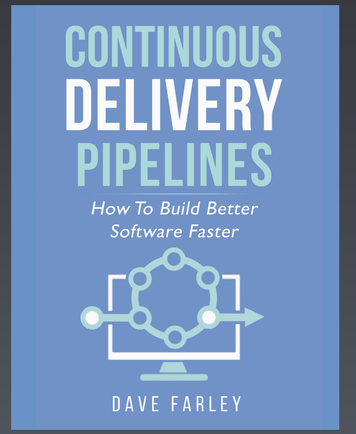
Q&A

Continuous Delivery ltd

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







Dave Farley

https://www.davefarley.net

@davefarley77

https://bit.ly/CDonYT