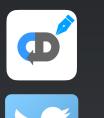
#### **Amplifying Creativity**

#### **Dave Farley**



https://www.davefarley.net



https://bit.ly/CDonYT

## Continuous Delivery Itd

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

# **Engineering for Software**

#### DAVID FARLEY

#### MODERN SOFTWARE ENGINEERING

Doing What Works to uild Better Software Faster Foreword by TRISHA GE

The Addison Wesley Signature Series

#### Continuous DELIVERY

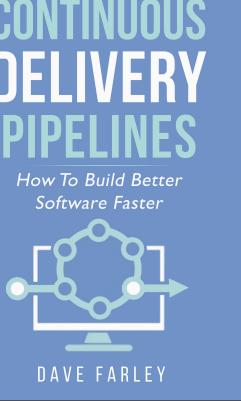
JEZ HUMBLE DAVID FARLEY

AN ANTIC

-----

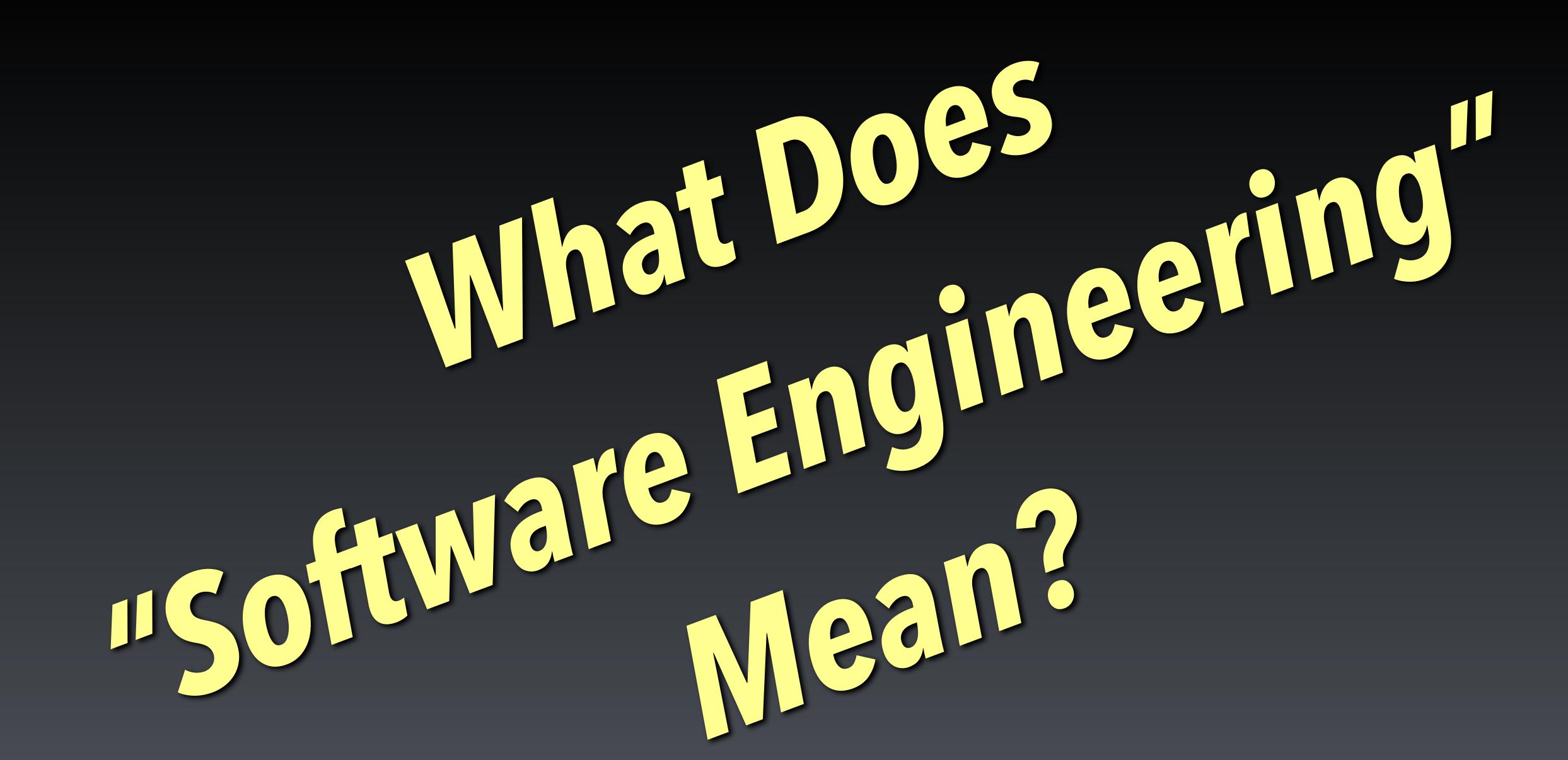
Foreword by Martin Fowler















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



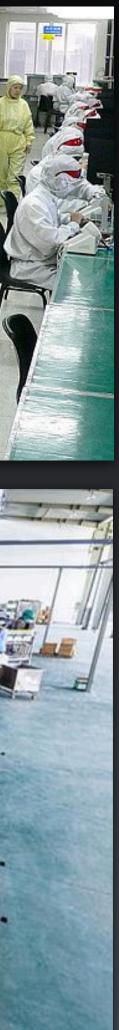


# "The Things We Can't Afford to Get Wrong"



















	•			accentar	nce-testing – BookShoppingDsl.java [cd-acceptance-testing_test]	
	cceptance-testing > src > test > java > com >	cd \ ec	centan			ତ 5 📭
					x C BookShoppingDsl.java X I BookShopDriver.java X C BookShopDrivers.java X C MyLocalBookStoreBookShopDriver.java	
<u>1</u> : Project						a 🔨 Cardijav
1: P	<ul> <li>acceptance-testing ~/dev/acceptanc</li> <li>acceptance-testing ~/dev/acceptanc</li> </ul>		pac	ckage com.cd.accep	prance.dsl;	
			imn	not com.cd.accent	tance.dsl.drivers.BookShopDrivers;	
	<ul> <li>build</li> </ul>		Tub			
				olic class BookSho	ppingDsl	
	▼ 🖿 src		{			
	► 📴 main			private final Bo	pokShopDrivers driver;	
	test test sources root					
	<ul> <li>acceptance</li> <li>acceptance</li> </ul>			public BookShopp	pingDsl(BookShopDrivers drivers)	
	▼ Com ▼ Com			{		
	▼ <b>D</b> acceptance			this.driver	= drivers;	
	<ul> <li>Encorptance</li> <li>Encorptance</li> </ul>			}		
	C ExampleAmaz					
	C ExampleBook:			public void sear	rchForBook(String args)	
	G ExampleMyLo			1		
	► 🗖 utils				ns = new Params(args);	
	* Java			String title	e = params.Optional( name: "title", defaultValue: "Continuous Delivery");	
		18 19		dojvon cooc	chForBook(title);	
	Cu			<pre>driver.searc }</pre>		
		22		public void sele	ectBook(String args)	
	© BookShopping			{		
				Params param	ns = new Params(args);	
	Channel				or = params.Optional( name: "author", defaultValue: "David Farley");	
				driver.seled	ctBook(author);	
				}		
	C Params C TestAnnotatio					
	feature			public void addS	<pre>SelectedItemToShoppingBasket() { driver.addSelectedItemToShoppingBasket(); }</pre>	
	resources					
	acceptance-testing.iml			public void asse	ertItemListedInShoppingBasket(String args)	
	R build.gradle			{		
	gradiew				ns = new Params(args);	
				String item	<pre>= params.Optional( name: "item", defaultValue: "Continuous Delivery");</pre>	
				deiver ees	atlistadInShappingPacket(itam);	
	A settings.gradie	40 41		driver.asser	rtListedInShoppingBasket(item);	
	Illi External Libraries			,		
				public void cher	ckOut(String args)	
				{	ntooc(ocrang di go)	
				Params param	ns = new Params(args);	
					= params.Optional( name: "item", defaultValue: "Continuous Delivery");	
ture					e = params.Optional( name: "price", defaultValue: "£10.00");	
Structure		48	-		parseCard(params.Optional( name: "card", defaultValue: "1234 5678 9101 0001 12/23 007"));	
5						
•				driver.check	kOut(item, price, card);	
SB			þ	}		
Favorites						
2:Fav				public void asse	ertItemPurchased(String args)	
*			þ	{		
	: <u>≣ 6:</u> TODO 🔨 Build 🔰 <u>9</u> : Git 🚺 <u>8</u> : Servi	rices D	Z Termi		ne - naw Daname(ange):	3 Event
	Intellij IDEA 2020.2.1 available: // Update (yesterda				48:82 LF UTF-8 4 spa	







				acceptance-testing – BookShoppingDsl.java [cd-acceptance-testing_test]		
	java 〉com 〉c	d ) acc	eptance	→ dsl 〉 @ BookShoppingDsl 〉 @ checkOut	Git: 🖌 🗸 🗸	5 📭
				rBookShopDriver.java × 🖸 BookShoppingDsl.java × 💶 BookShopDriver.java × G BookShopDrivers.java × G MyLocal		Card.jav
▼ acceptance-testing ~/de				ge com.cd.acceptance.dsl;		
🖬 🕨 🖿 .gradle						
idea			impor	t com.cd.acceptance.dsl.drivers.BookShopDrivers;		
build						
► In gradle ▼ In src			publi	c class BookShoppingDsl		
► <b>In main</b>			1	tivete finel BackCharDeivers deivert		
Test test sources ro			P	rivate final BookShopDrivers driver;		
🔻 🖿 acceptance				ublic BookShoppingDsl(BookShopDrivers drivers)		
V 🗖 com		10	í f			
▼ 🗖 cd				this.driver = drivers;		
accepta		<b>12</b> (	¢ }			
▼ 🖿 exam ooi E	npies xampleAmaz					
	xampleBook:		P	ublic void searchForBook(String args)		
	xampleMyLo <sup>1</sup>	.5	¢ {			
► 🗖 utils				Params params = new Params(args);		
▼ 🖿 java				<pre>String title = params.Optional( name: "title", defaultValue: "Continuous Delivery");</pre>		
▼ Im com		18 19		daivaa aaaabEanBaak(titla).		
V 🗖 cd		19 20 (	ρ }	driver.searchForBook(title);		
▼ D∎ accepta ▼ D∎ dsl		20 I 21				
► La dsi			C	ublic void selectBook(String args)		
	ookShopping 2					
				Params params = <mark>new</mark> Params(args);		
				<pre>String author = params.Optional( name: "author", defaultValue: "David Farley");</pre>		
C C C D	N-1			driver.selectBook(author);		
			¢ }			
	estAnnotatio				+() - )	
File feature			± p	<pre>ublic void addSelectedItemToShoppingBasket() { driver.addSelectedItemToShoppingBaske</pre>	t(); }	
resources				ublic void assertItemListedInShoppingBasket(String args)		
acceptance-testing.iml		55 36 (		Socie void assertitementsteurnsnoppingbasket(string args)		
and build.gradle				Params params = new Params(args);		
▶ gradlew		38		String item = params.Optional( name: "item", defaultValue: "Continuous Delivery");		
॑ gradlew.bat ☑ run-selenium.sh						
er run-selenium.sn				driver.assertListedInShoppingBasket(item);		
<ul> <li>IIII External Libraries</li> </ul>			ģ }			
🗞 Scratches and Consoles						
			P	ublic void checkOut(String args)		
				Params params = new Params(args);		
e				<pre>String item = params.Optional( name: "item", defaultValue: "Continuous Delivery"); String price = params.Optional( name: "price", defaultValue: "S10 00");</pre>		
Structure		47 48	•	String price = params.Optional( name: "price", defaultValue: "£10.00"); Card card = parseCard(params.Optional( name: "card", defaultValue: "1234 5678 9101 00	001 12/23 007" <b>))</b> .	
Z: Str		4 <b>0</b> 49		our a cur a - pur secur a (par ans. operonal ( some, car a , decontrate. 1234 3676 9101 0	···· ··· ··· ··· ··· ··· ··· ··· ··· ·	
3				driver.checkOut(item, price, card);		
φ.			□ }			
vorite						
2: Favorites				ublic void assertItemPurchased(String args)		
*						
:≣ <u>6</u> : TODO <b>≺</b> Build  ≄ <u>9</u> : Git	t 🗗 8: Servic	es D	Terminal	Baname paname - naw Baname(ange):		3 Event
IntelliJ IDEA 2020.2.1 available: // Upd					48:82 LF UTF-8 4 spaces	

· 🔒 👲

Q

11100 0010010101000100100100100 001 1001010 1010 0011010100011 I A A 1 A A 1 A A A 1 A A 1 A A 1 A A 1 A 1 A A 1 A A 1 A A 1 A A 1 A A 1 A A 1 A A 1 A A 1 A A 1 A A 1 A A 1 A





				acceptance-testing – BookShoppingDsl.java [cd-acceptance-testing_test]		
	java 〉com 〉c	d ) acc	eptance	→ dsl 〉 @ BookShoppingDsl 〉 @ checkOut	Git: 🖌 🗸 🗸	5 📭
				rBookShopDriver.java × 🖸 BookShoppingDsl.java × 💶 BookShopDriver.java × G BookShopDrivers.java × G MyLocal		Card.jav
▼ acceptance-testing ~/de				ge com.cd.acceptance.dsl;		
🖬 🕨 🖿 .gradle						
idea			impor	t com.cd.acceptance.dsl.drivers.BookShopDrivers;		
build						
► In gradle ▼ In src			publi	c class BookShoppingDsl		
► <b>In main</b>			1	tivete finel BackCharDeivers deivert		
Test test sources ro			P	rivate final BookShopDrivers driver;		
🔻 🖿 acceptance				ublic BookShoppingDsl(BookShopDrivers drivers)		
V 🗖 com		10	í f			
▼ 🗖 cd				this.driver = drivers;		
accepta		<b>12</b> (	¢ }			
▼ 🖿 exam ooi E	npies xampleAmaz					
	xampleBook:		P	ublic void searchForBook(String args)		
	xampleMyLo <sup>1</sup>	.5	¢ {			
► 🗖 utils				Params params = new Params(args);		
▼ 🖿 java				<pre>String title = params.Optional( name: "title", defaultValue: "Continuous Delivery");</pre>		
▼ Im com		18 19		daivaa aaaabEanBaak(titla).		
V 🗖 cd		19 20 (	ρ }	driver.searchForBook(title);		
▼ In accepta ▼ In dsl		20 I 21				
► La dsi			C	ublic void selectBook(String args)		
	ookShopping 2					
				Params params = <mark>new</mark> Params(args);		
				<pre>String author = params.Optional( name: "author", defaultValue: "David Farley");</pre>		
C C C D	N-1			driver.selectBook(author);		
			¢ }			
	estAnnotatio				+() - )	
File feature			± p	<pre>ublic void addSelectedItemToShoppingBasket() { driver.addSelectedItemToShoppingBaske</pre>	t(); }	
resources				ublic void assertItemListedInShoppingBasket(String args)		
acceptance-testing.iml		55 36 (		Socie void assertitementsteurnsnoppingbasket(string args)		
and build.gradle				Params params = new Params(args);		
▶ gradlew		38		String item = params.Optional( name: "item", defaultValue: "Continuous Delivery");		
॑ gradlew.bat ☑ run-selenium.sh						
er run-selenium.sn				driver.assertListedInShoppingBasket(item);		
<ul> <li>IIII External Libraries</li> </ul>			ģ }			
🗞 Scratches and Consoles						
			P	ublic void checkOut(String args)		
				Params params = new Params(args);		
e				<pre>String item = params.Optional( name: "item", defaultValue: "Continuous Delivery"); String price = params.Optional( name: "price", defaultValue: "S10 00");</pre>		
Structure		47 48	•	String price = params.Optional( name: "price", defaultValue: "£10.00"); Card card = parseCard(params.Optional( name: "card", defaultValue: "1234 5678 9101 00	001 12/23 007" <b>))</b> .	
Z: Str		4 <b>0</b> 49		our a cur a - pur secur a (par ans. operonal ( some, car a , decontrate. 1234 3676 9101 0	···· ··· ··· ··· ··· ··· ··· ··· ··· ·	
3				driver.checkOut(item, price, card);		
φ.			□ }			
vorite						
2: Favorites				ublic void assertItemPurchased(String args)		
*						
:≣ <u>6</u> : TODO <b>≺</b> Build  ≄ <u>9</u> : Git	t 🗗 8: Servic	es D	Terminal	Baname paname - naw Baname(ange):		3 Event
IntelliJ IDEA 2020.2.1 available: // Upd					48:82 LF UTF-8 4 spaces	

· 🔒 👲

Q

11100 0010010101000100100100100 001 1001010 1010 0011010100011 I A A 1 A A 1 A A A 1 A A 1 A A 1 A A 1 A 1 A A 1 A A 1 A A 1 A A 1 A A 1 A A 1 A A 1 A A 1 A A 1 A A 1 A A 1 A





010|0|110|10|1001010100100100100|001001110010|00101010011010100011 





010|0|110|10|1001010100100100100|001001110010|00101010011010100011 





01010 100100100101010010101010010010010110010 101001001001010010010010010100000010100 









### DEPLOY

 $30 \quad 001 \\ 001 \\ 010 \\ 100 \\ 100 \\ 100 \\ 110 \\ 110 \\ 100 \\ 110 \\ 100 \\ 010 \\ 110 \\ 100 \\ 010 \\ 110 \\ 100 \\ 010 \\ 110 \\ 100 \\ 010 \\ 110 \\ 100 \\ 000 \\ 101 \\ 100 \\ 000 \\ 101 \\ 100 \\ 000 \\ 101 \\ 100 \\$ 81 | 1 | 1 0 0 0 0 1 0 0 1 0 1 0 0 0 1 0 0 1 0 0 0 0 0 0 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 0 0 1 1 00100100111001010010101001101010100011 









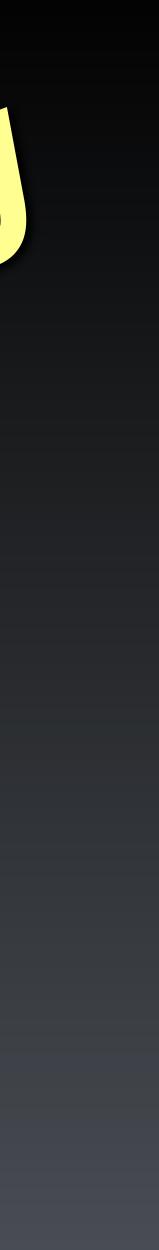






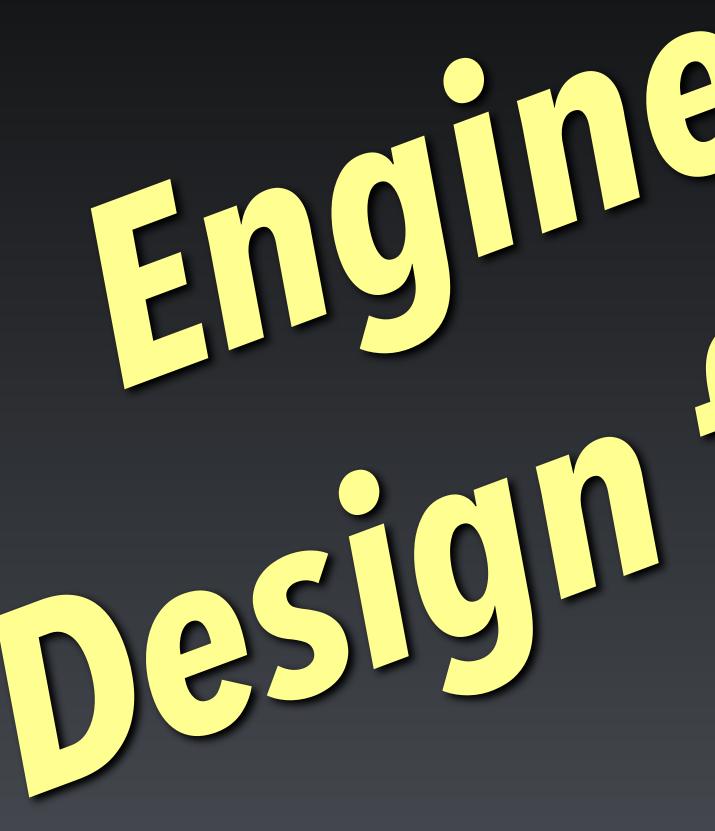
# "Designing, Building & Repairing things in a Principled way"











# Engineering: Engineering: Engineering: Design for Failure Design

# Alan Kal





# All Engineering is not the same!



















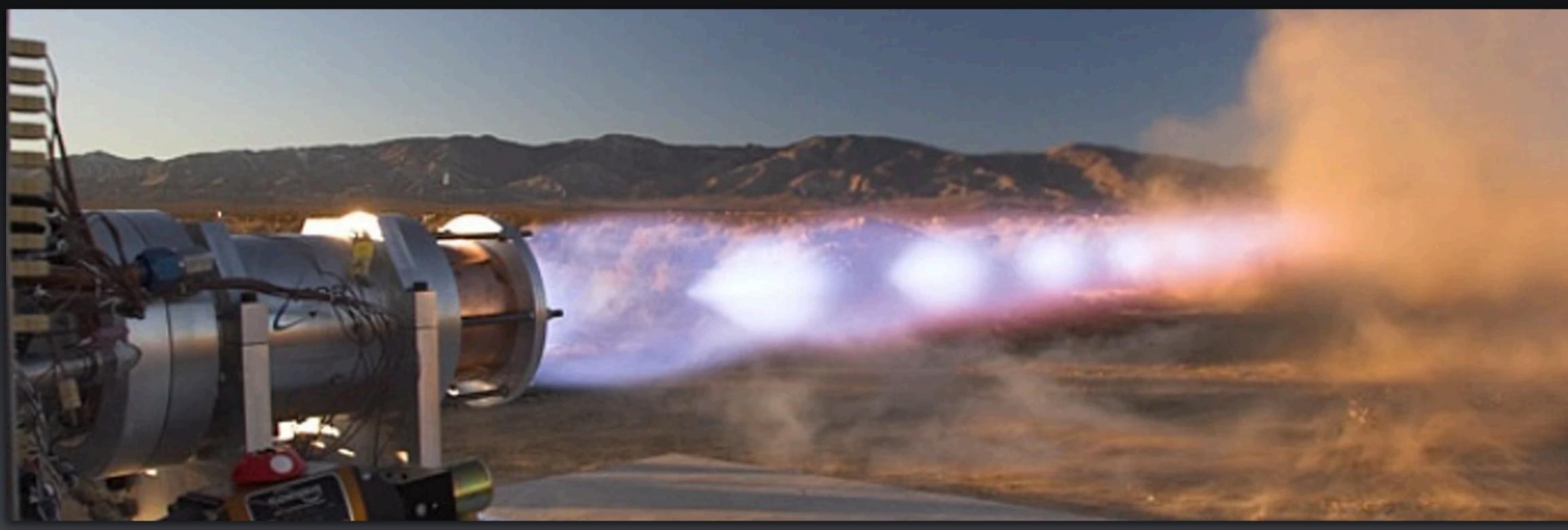












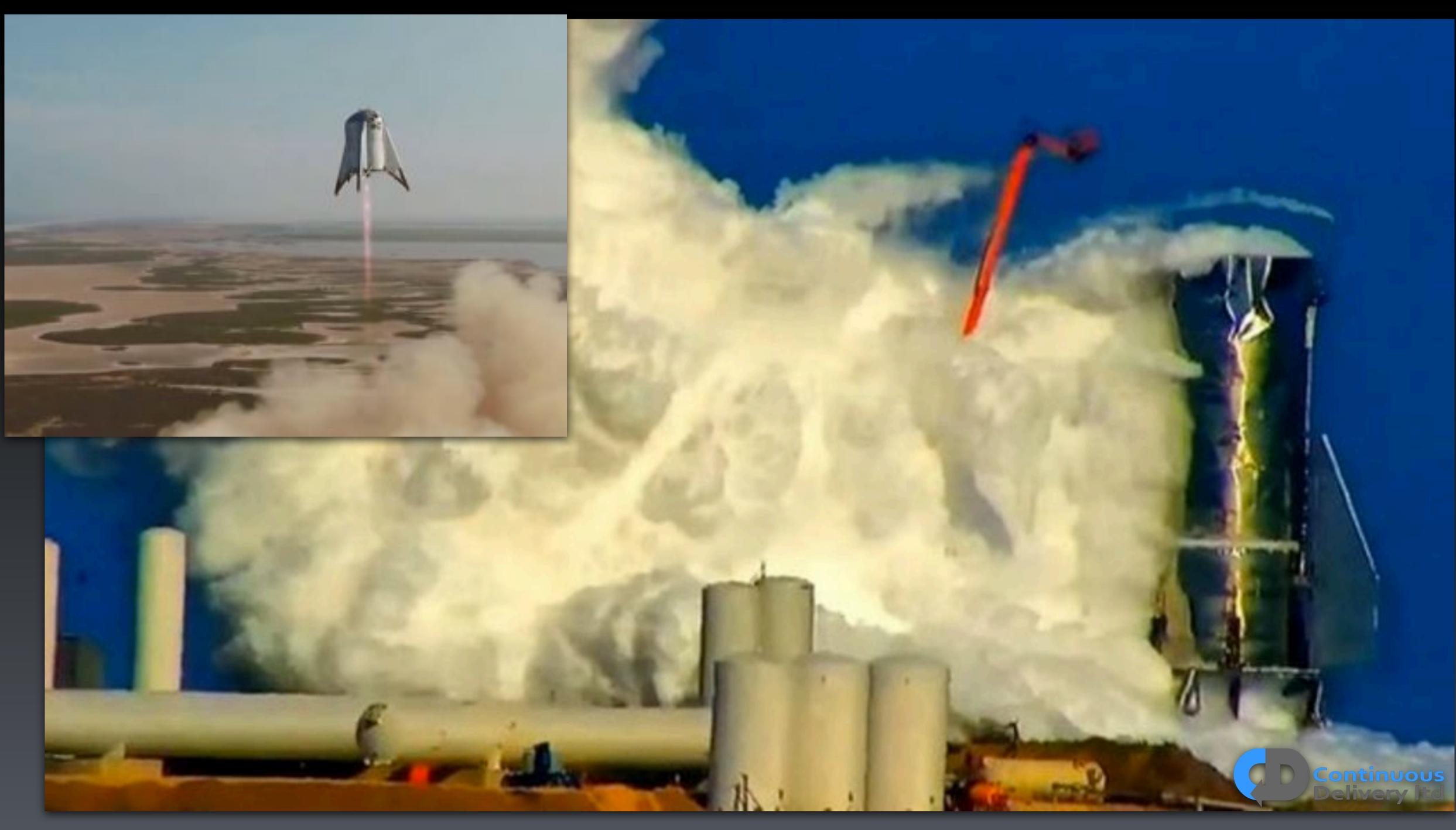




































## Optimise for Learning





# Iteration

## Optimise for Learning





# Optimise for Learning Iteration Feedback







# Iteration Feedback Incremental Optimise for Learning





# Iteration Feedback Feedback Incremental Optimise for Learning Experimental





## Optimise for Learning Iteration Feedback ncrementa Experimental Empirical









# Software Development is also about Managing Complexity









## Optimise to Manage Complexity





## Optimise to Manage Complexity Modularity







# Optimise to Manage Complexity Modularity Abstraction





# Modularity Abstraction Separation of Concerns







# Optimise to Manage Complexity Modularity Abstraction Separation of Concerns Loose Coupling







## Optimise to Manage Complexity Modularity Abstraction Separation of Concerns the coupling







## Principles of Applying Engineering Thinking

- Optimise for Learning
- Optimise to Manage/Limit Complexity
- Control the Variables
- Make Evidence Based Decisions (Run the Experiments)
- Never Assume You Have the Correct Answer
- Find Ways to Falsify Ideas Simply (More Experiments!)























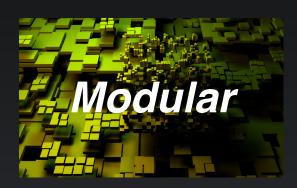
• It Has to Work!





• It Has to Work!

• Modular



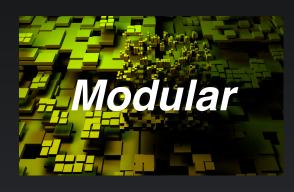




• It Has to Work!

• Modular

Loosely-coupled

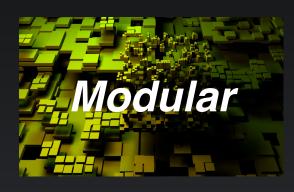








- It Has to Work!
- Modular
- Loosely-coupled
- High-Cohesion



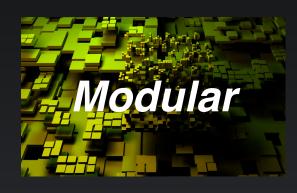








- It Has to Work!
- Modular
- Loosely-coupled
- High-Cohesion
- Good Separation of Concerns















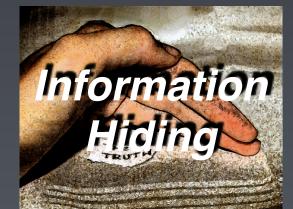
- It Has to Work!
- Modular
- Loosely-coupled
- High-Cohesion
- Good Separation of Concerns
- Exhibits Information Hiding















#### What Drives Quality? - Before TDD

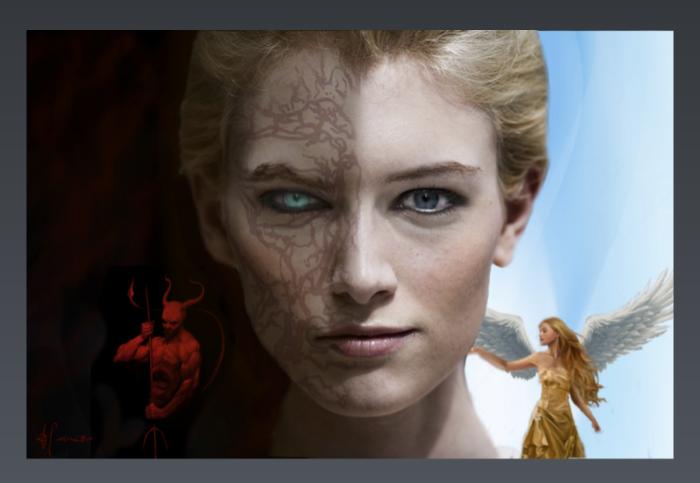




#### What Drives Quality? - Before TDD



#### The Skill, Experience and Integrity of an individual programmer.





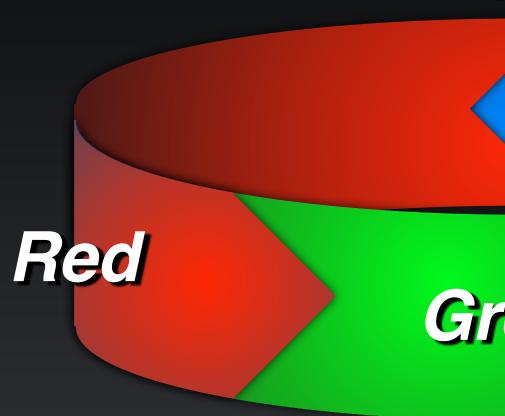




## TDD is...







#### TDD is...

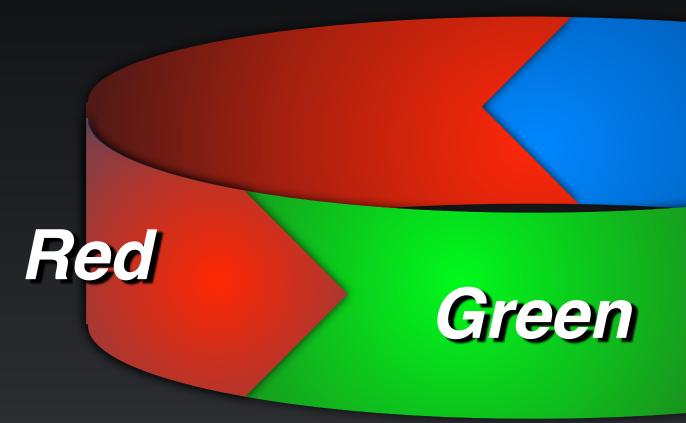
#### Repeat!



#### Refactor







#### • Write a Test - See it Fail

#### TDD is...

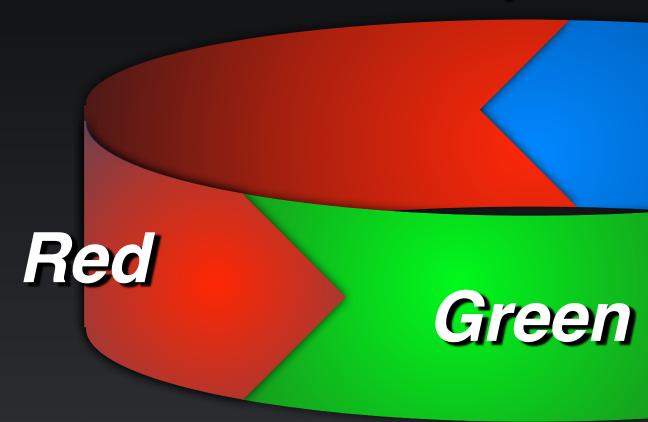
#### Repeat!

#### Refactor









• Write a Test - See it Fail

#### TDD is...

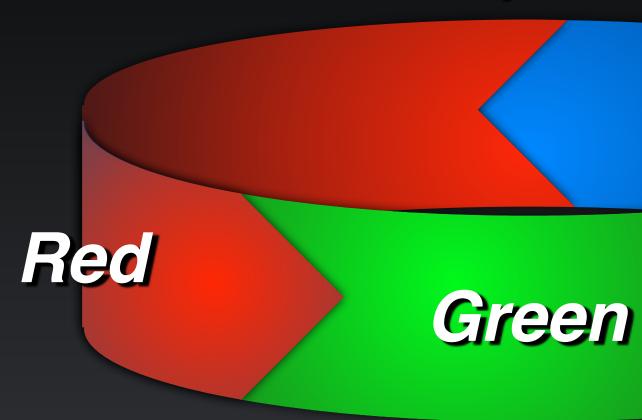
#### Repeat!



## • Write Code to Make the Test Pass - See it Pass







- Write a Test See it Fail

## TDD is...

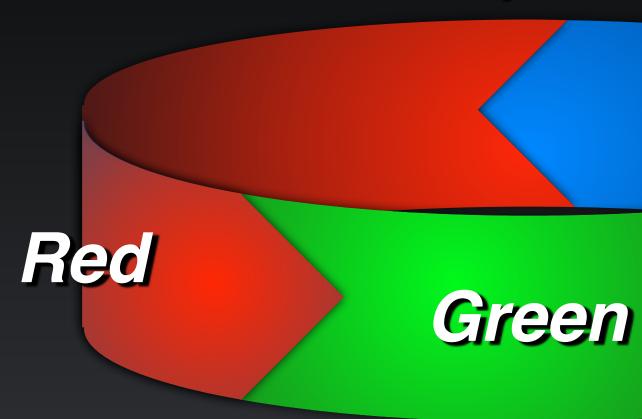
### **Repeat!**



# • Write Code to Make the Test Pass - See it Pass Modify the Code to Make it Clean and Elegant







- Write a Test See it Fail

- Next Test...

## TDD is...

### **Repeat!**



# • Write Code to Make the Test Pass - See it Pass Modify the Code to Make it Clean and Elegant









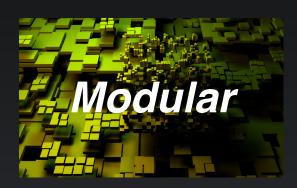
• It Has to Work!





• It Has to Work!

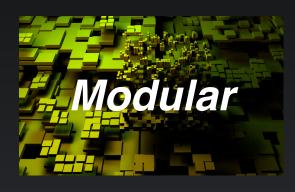
• Modular







- It Has to Work!
- Modular
- Loosely-coupled

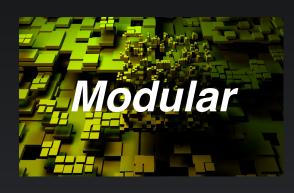








- It Has to Work!
- Modular
- Loosely-coupled
- High-Cohesion











- It Has to Work!
- Modular
- Loosely-coupled
- High-Cohesion
- Good Separation of Concerns













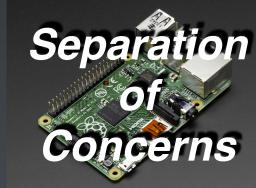


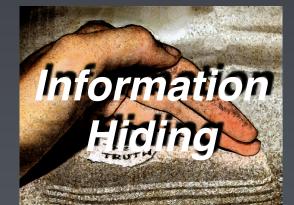
- It Has to Work!
- Modular
- Loosely-coupled
- High-Cohesion
- Good Separation of Concerns
- Exhibits Information Hiding









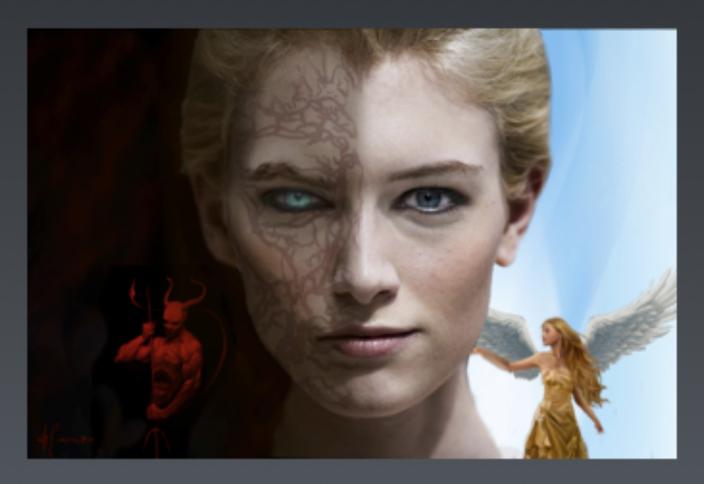








## The Skill, Experience and Integrity of an individual programmer.



(C)opyright Dave Farley 2015









(C)opyright Dave Farley 2015



The Skill, Experience and Integrity of an individual programmer.













The Skill, Experience and Integrity of an individual programmer.









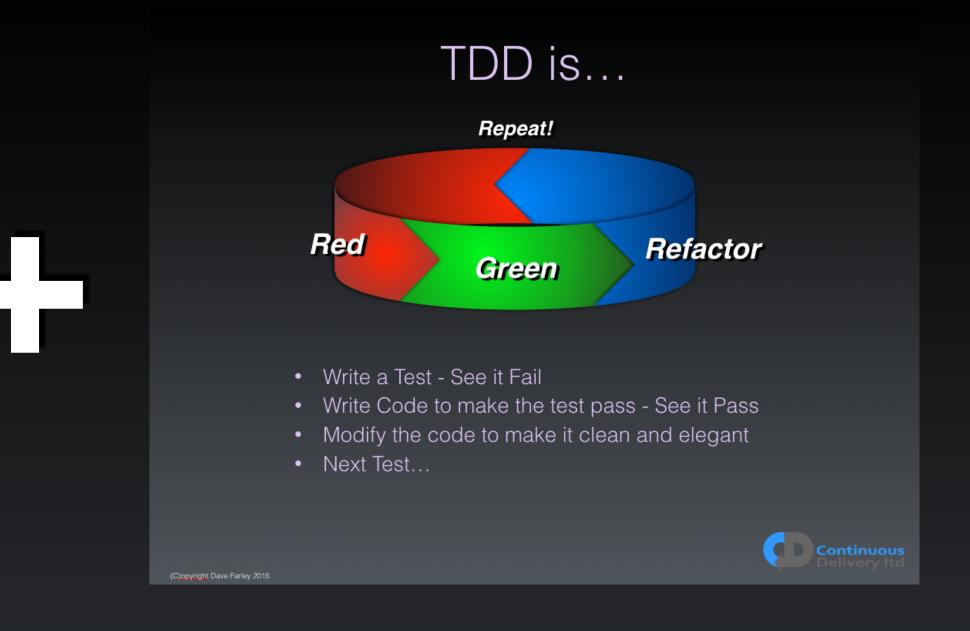




The Skill, Experience and Integrity of an individual programmer.







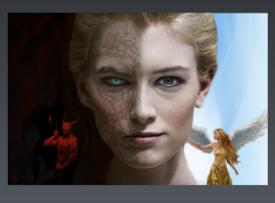








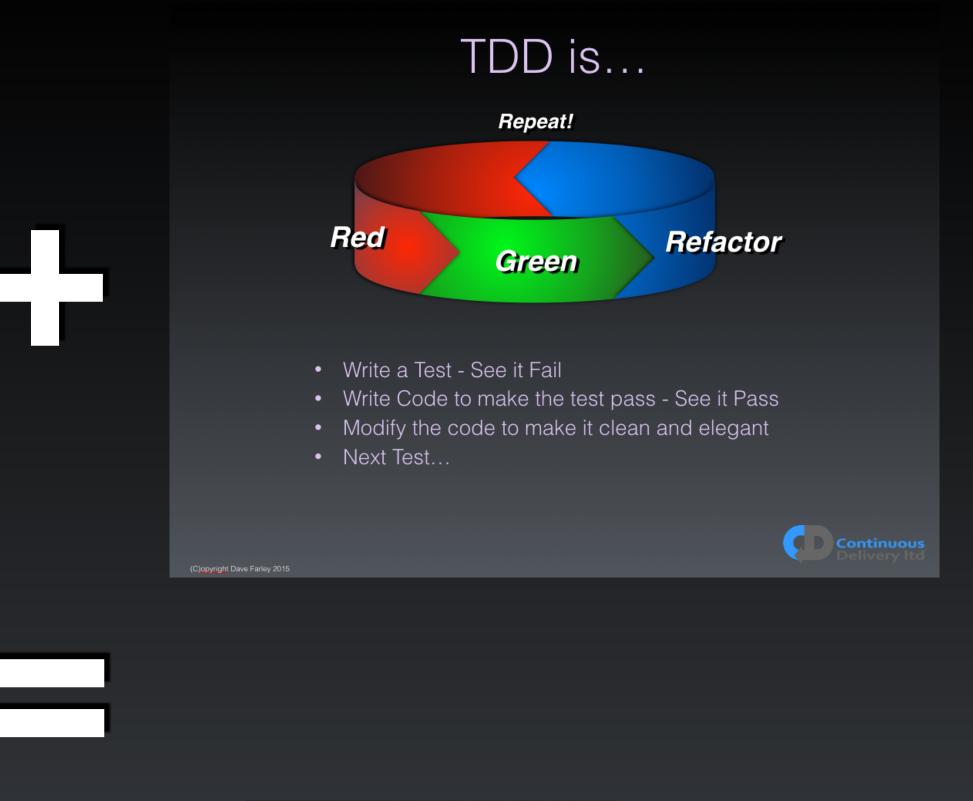
The Skill, Experience and Integrity of an individual programmer.

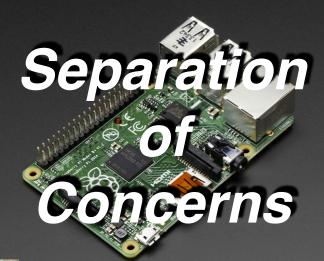












Information Hitcling

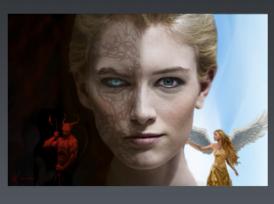


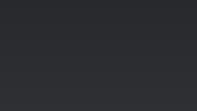






The Skill, Experience and Integrity of an individual programmer.

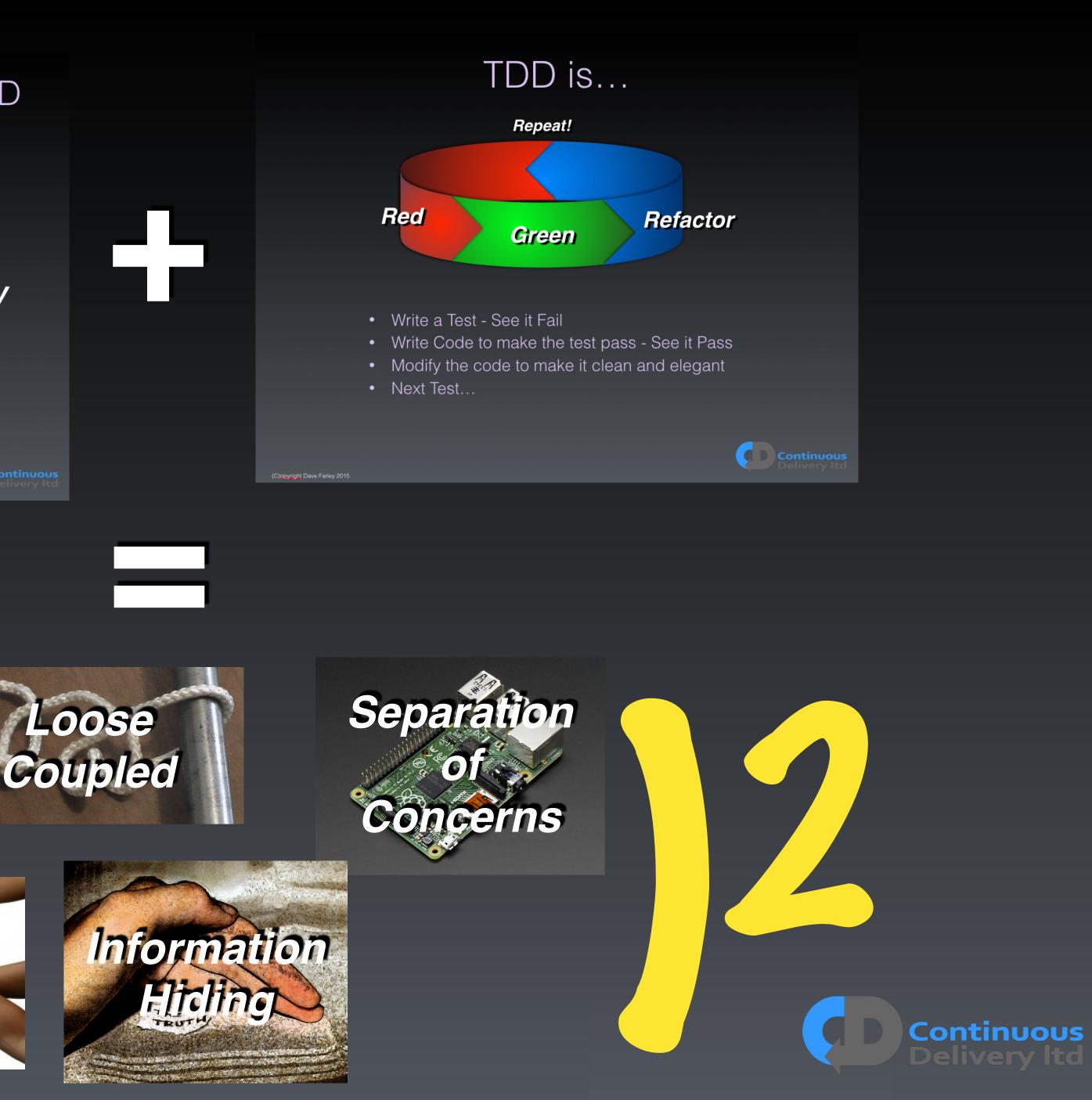




Continuous













public class Car {

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

private void applyBrakes() { }

private void putIntoNeutral() {

### private final Engine engine = new PetrolEngine();





public class Car {

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

private void applyBrakes() { }

private void putIntoNeutral() {

### private final Engine engine = new PetrolEngine();





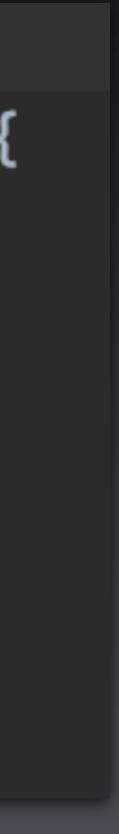
```
public class Car {
   private final Engine engine = new PetrolEngine();
   public void start() {
        putIntoNeutral();
        applyBrakes();
        this.engine.start();
    }
   private void applyBrakes() {
    }
   private void putIntoNeutral() {
    }
```

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

car.start();

// Nothing to assert!!







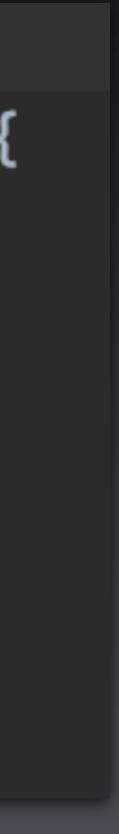
```
public class Car {
    private final Engine engine = new PetrolEngine();
    public void start() {
        putIntoNeutral();
        applyBrakes();
        this.engine.start();
    }
    private void applyBrakes() {
    }
    private void putIntoNeutral() {
    }
```

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

car.start();

// Nothing to assert!!















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

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

private void applyBrakes() { }

private void putIntoNeutral() {





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

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

private void applyBrakes() { }

private void putIntoNeutral() {





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

annlyBrakes(); public class C

### private final Engine engine = new PetrolEngine();





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

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

private void applyBrakes() { }

private void putIntoNeutral() {





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

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

private void applyBrakes() { }

private void putIntoNeutral() {





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

### public void shouldStartBetterCarEngine() {

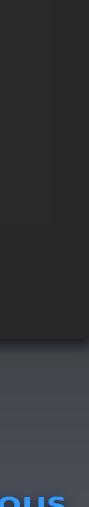




```
public class BetterCar {
    private final Engine engine;
    public BetterCar(Engine engine) {
                                        @Test
        this.engine = engine;
    }
    public void start() {
        putIntoNeutral();
        applyBrakes();
        this.engine.start();
    private void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

public void shouldStartBetterCarEngine() { FakeEngine engine = new FakeEngine();





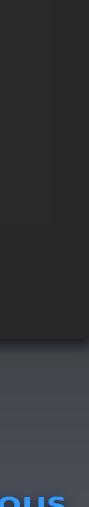


```
public class BetterCar {
   private final public class FakeEngine implements Engine {
                     private boolean started = false;
   public Better
      this.engi
   }
                     @Override
                     public void start() {
   public void s
                          started = true;
      putIntoNe
      applyBrak
      this.engi
   private void
                          return started;
                     ł
   private void
```

# public boolean startedSuccessfully() {

ine() { ine();



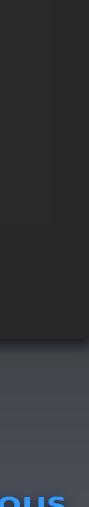




```
public class BetterCar {
    private final Engine engine;
    public BetterCar(Engine engine) {
                                        @Test
        this.engine = engine;
    }
    public void start() {
        putIntoNeutral();
        applyBrakes();
        this.engine.start();
    private void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

public void shouldStartBetterCarEngine() { FakeEngine engine = new FakeEngine();







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



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



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



### public void shouldStartBetterCarEngine() { FakeEngine engine = new FakeEngine(); BetterCar car = new BetterCar(engine);

car.start();



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



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

car.start();

assertTrue(engine.startedSuccessfully());



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

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

private void applyBrakes() { }

private void putIntoNeutral() {





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

### public void createCars() {





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; }

### 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());









# Working Experimentally



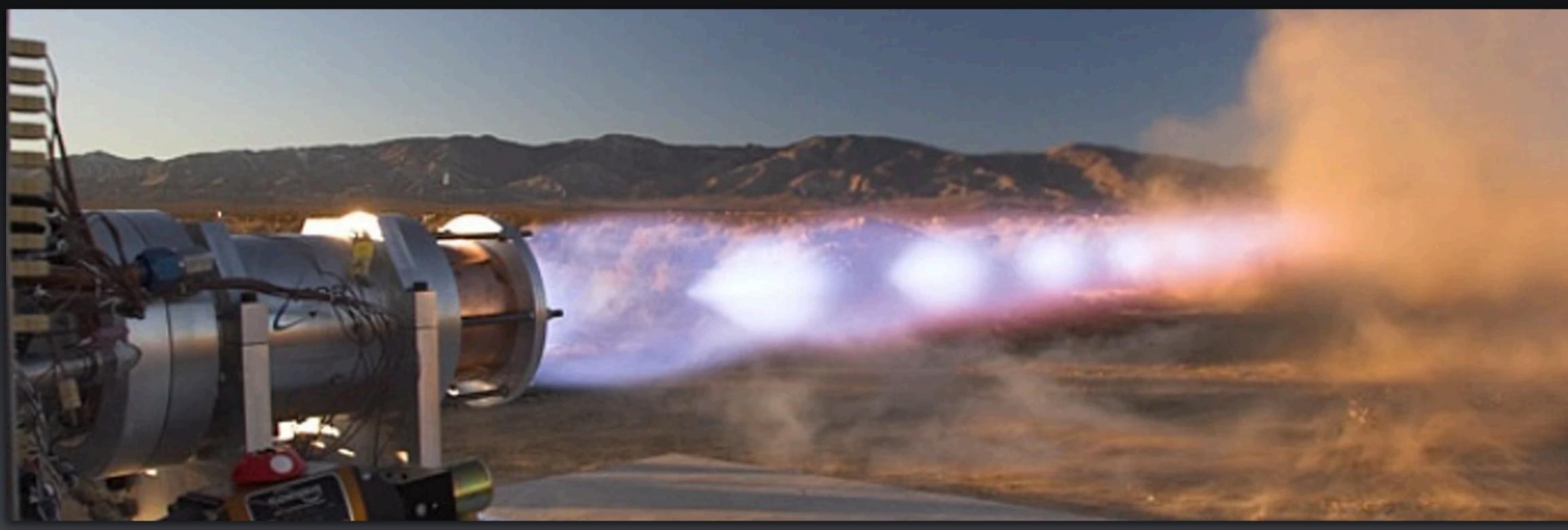












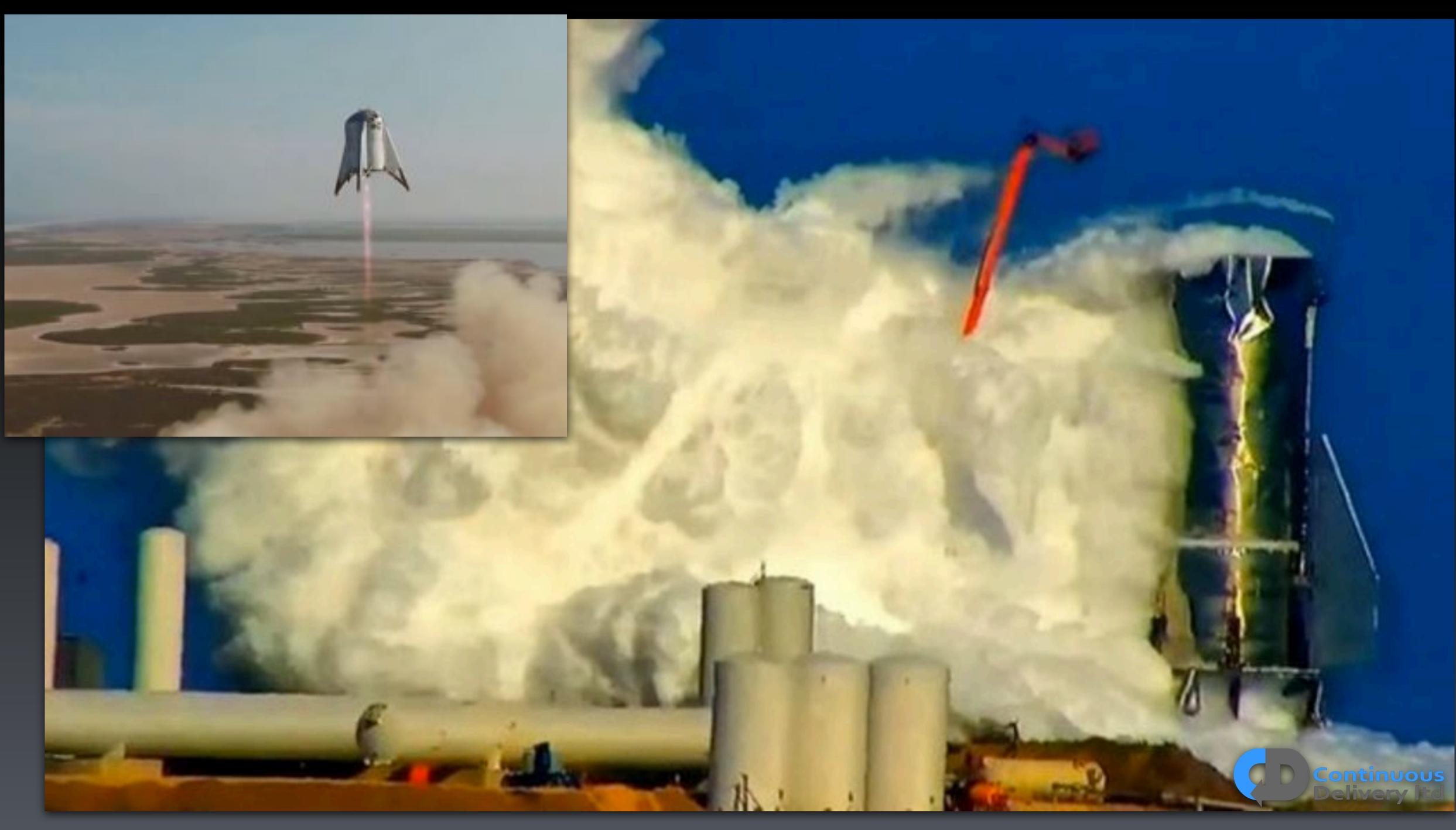




























# Gather Feedback





# Predict the Results







# **Control the Variables**







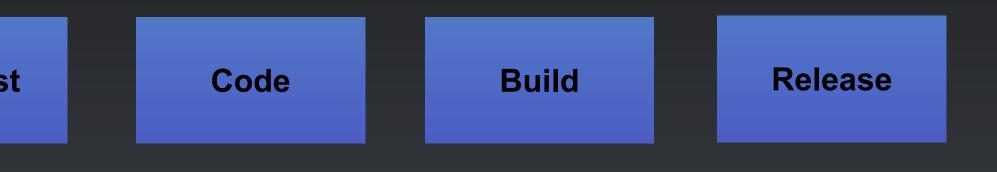




## What Really Works?

#### Smart Automation - a repeatable, reliable process for releasing software



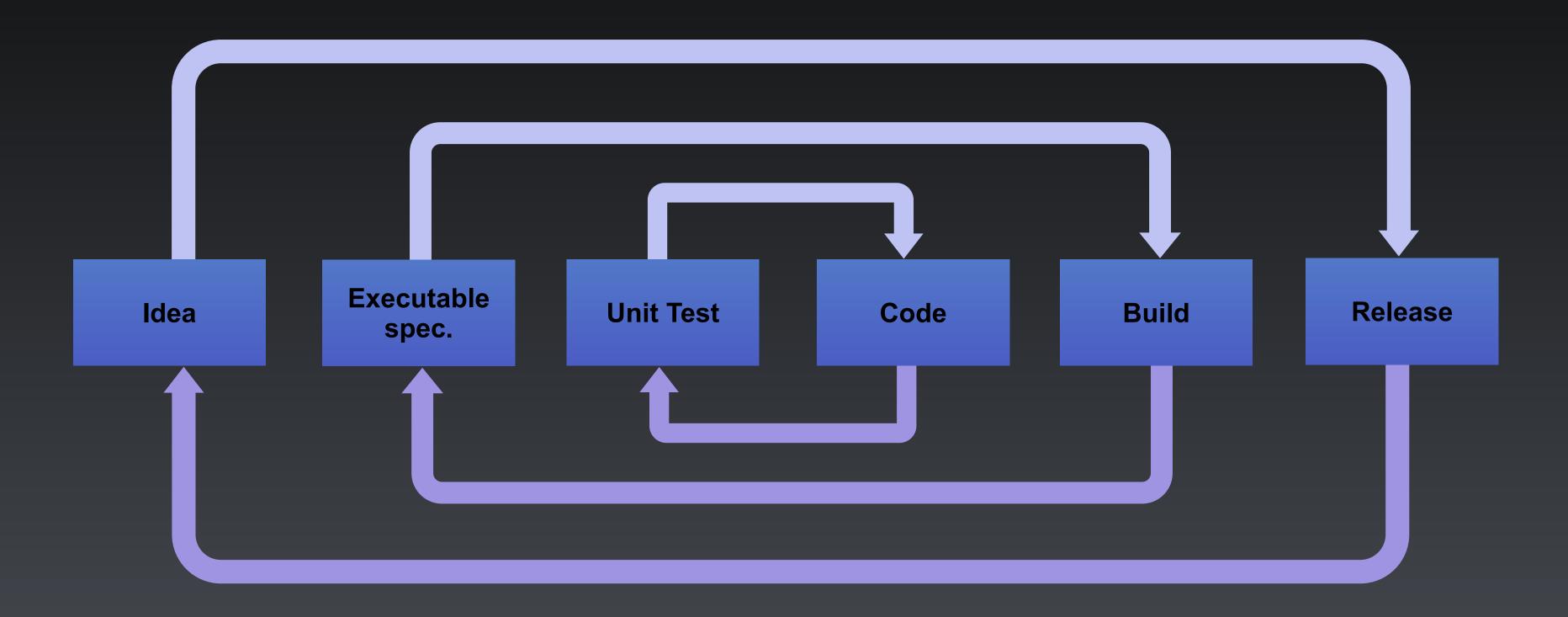






## What Really Works?

#### Smart Automation - a repeatable, reliable process for releasing software







## What Really Works?

### "It doesn't matter how intelligent you are, if you guess and that guess cannot be backed up by experimental evidence – then it is still a guess!"

**Richard Feynman** 













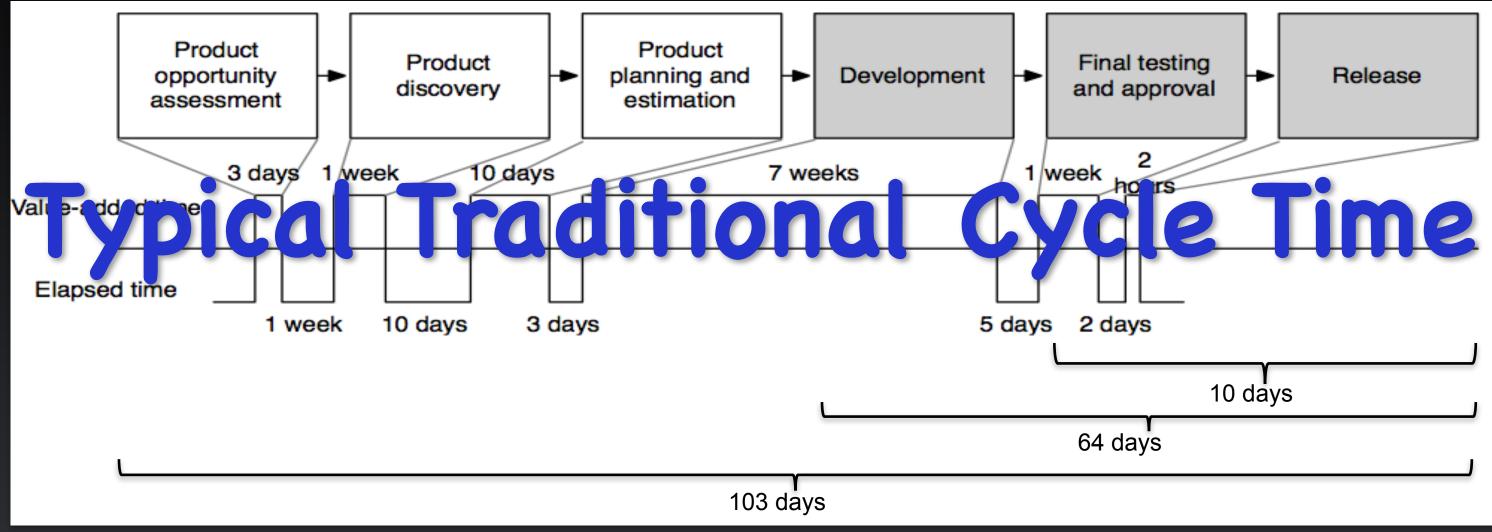








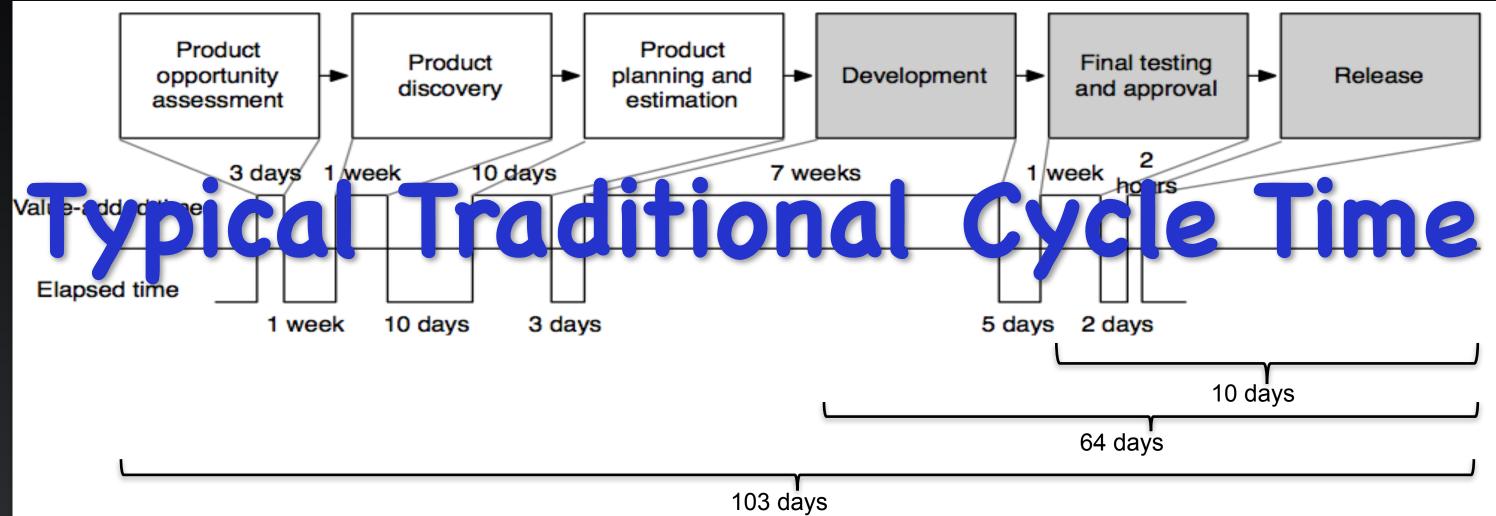
# Cycle-Time

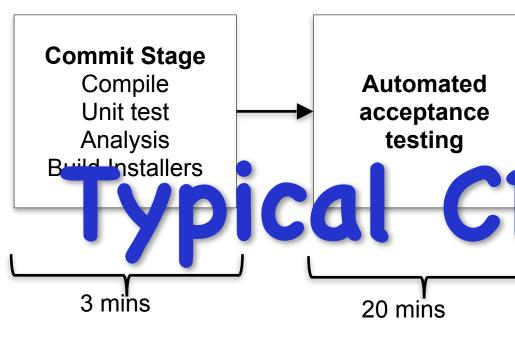






# Cycle-Time





Automated performance testing Release 20 mins me Manual testing 4 mins . 30 mins



57 mins



# What Is Continuous Delivery?

"Our highest priority is to satisfy the customer through early and continuous delivery of valuable software."

- The first principle of the agile manifesto.
- The logical extension of continuous integration.
- A holistic approach to development.
- **Every commit creates a release candidate.**
- Finished means released into the hands of users, delivering value!  $\bigcirc$





# The Principles of Continuous Delivery

- Create a repeatable, reliable process for releasing software.
- Automate almost everything.
- Keep everything under version control.
- If it hurts, do it more often bring the pain forward.
- Build quality in.
- Done means released.
- Everybody is responsible for the release process.
- Improve continuously.





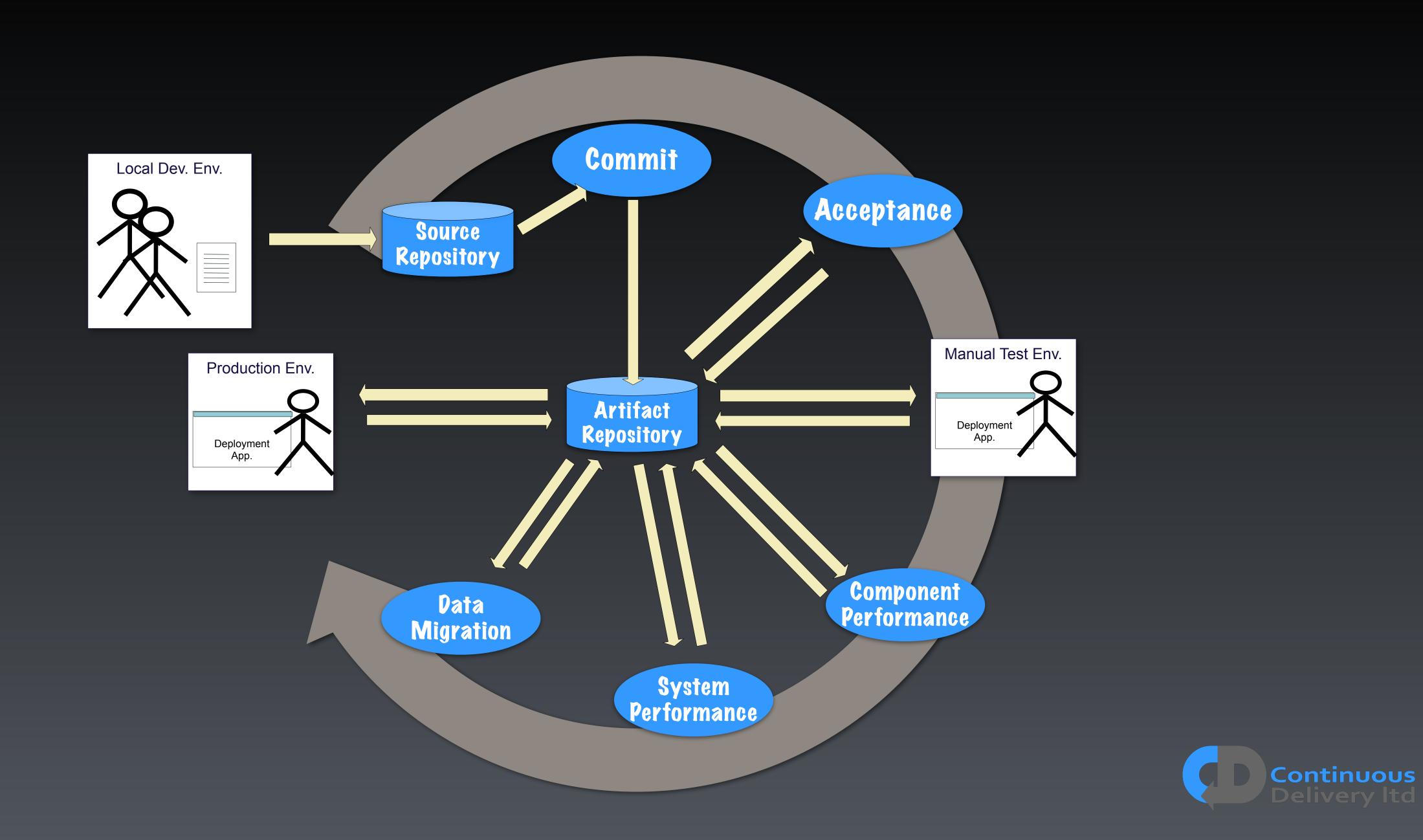
# The Principles of Continuous Delivery

Create a repeatable, reliable process for releasing software. "If Agile software development was the opening act to a great performance, Continuous Delivery is responsible for the release process. Delivery is the headliner."

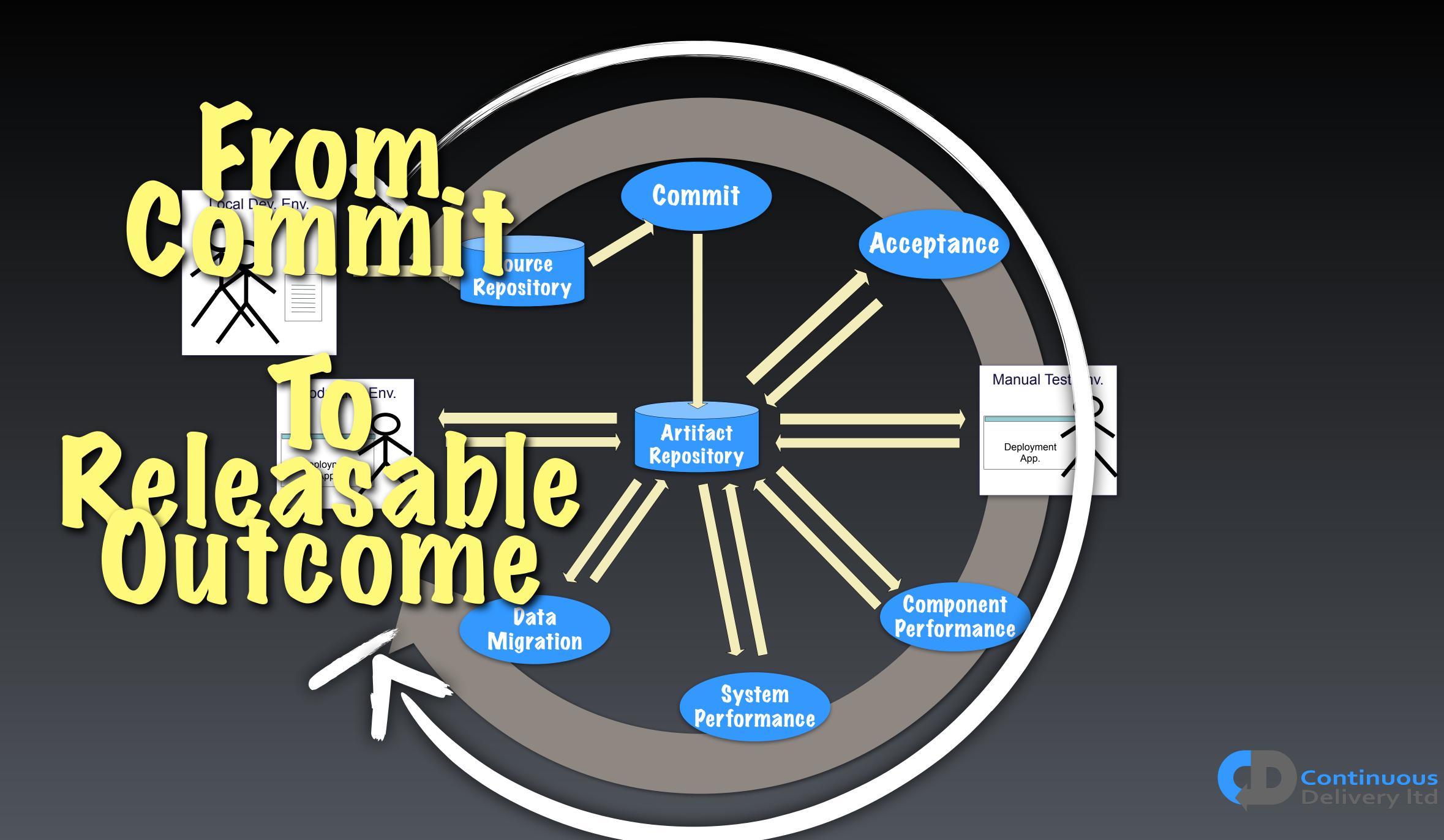
**Forrester Research 2013** 



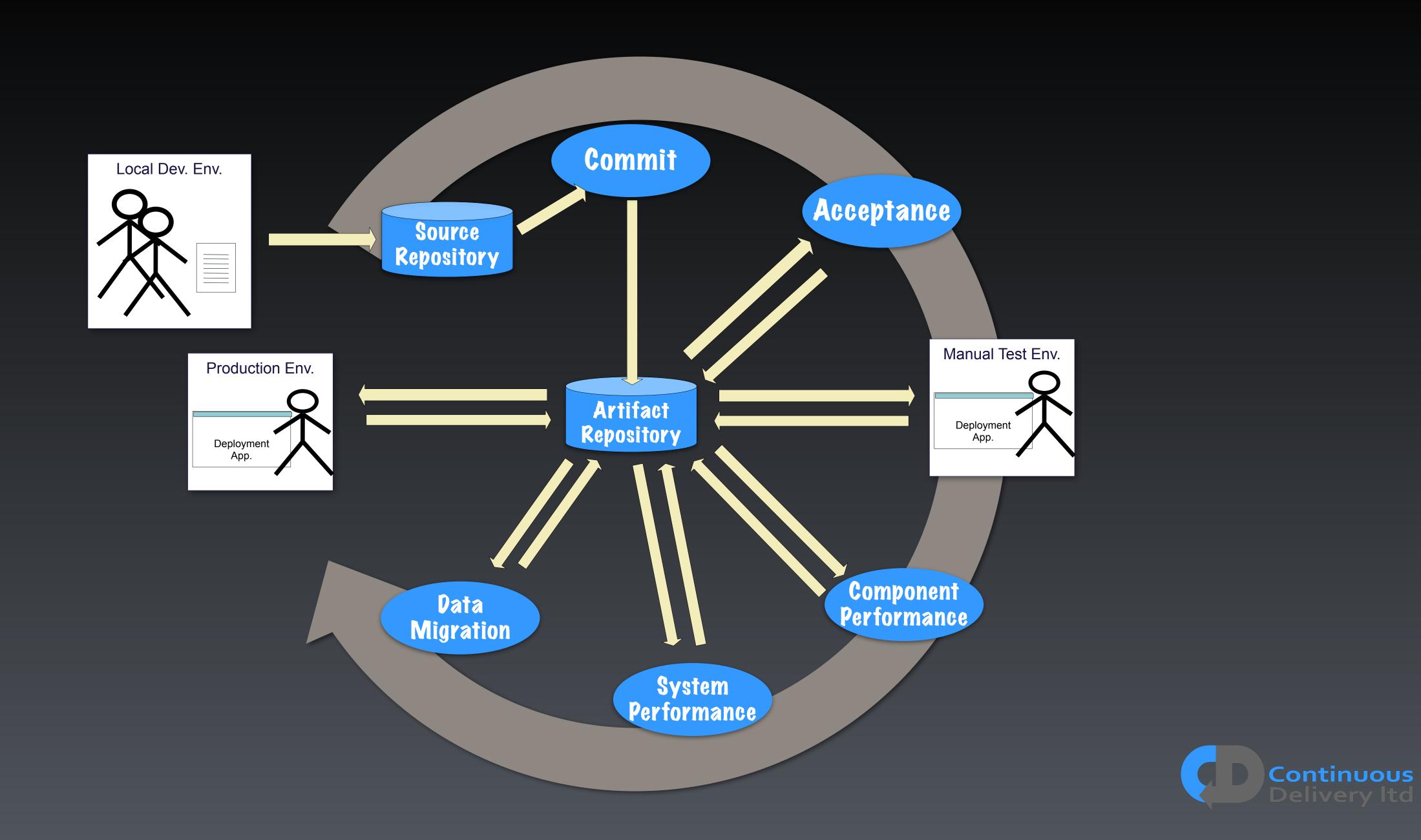




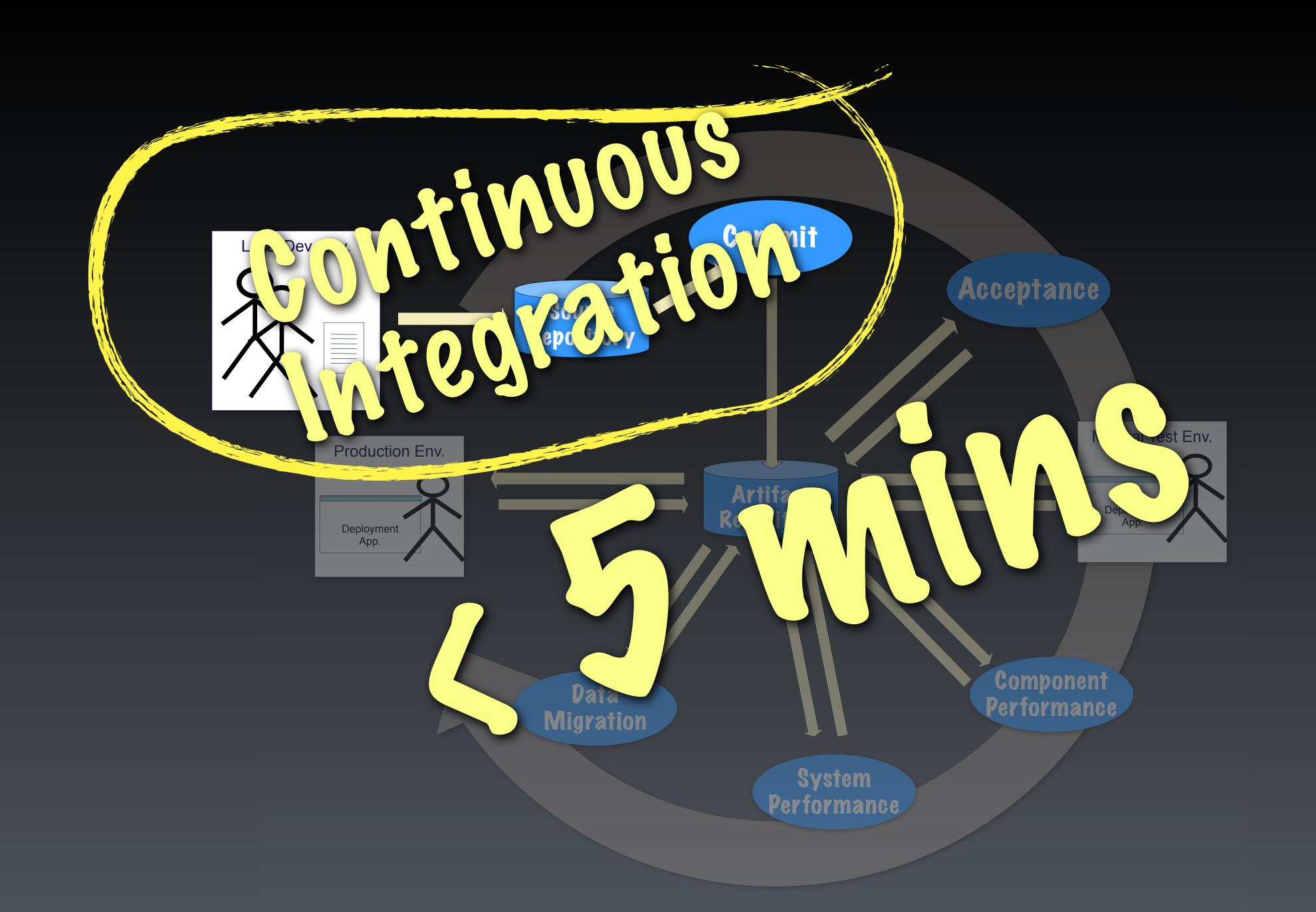






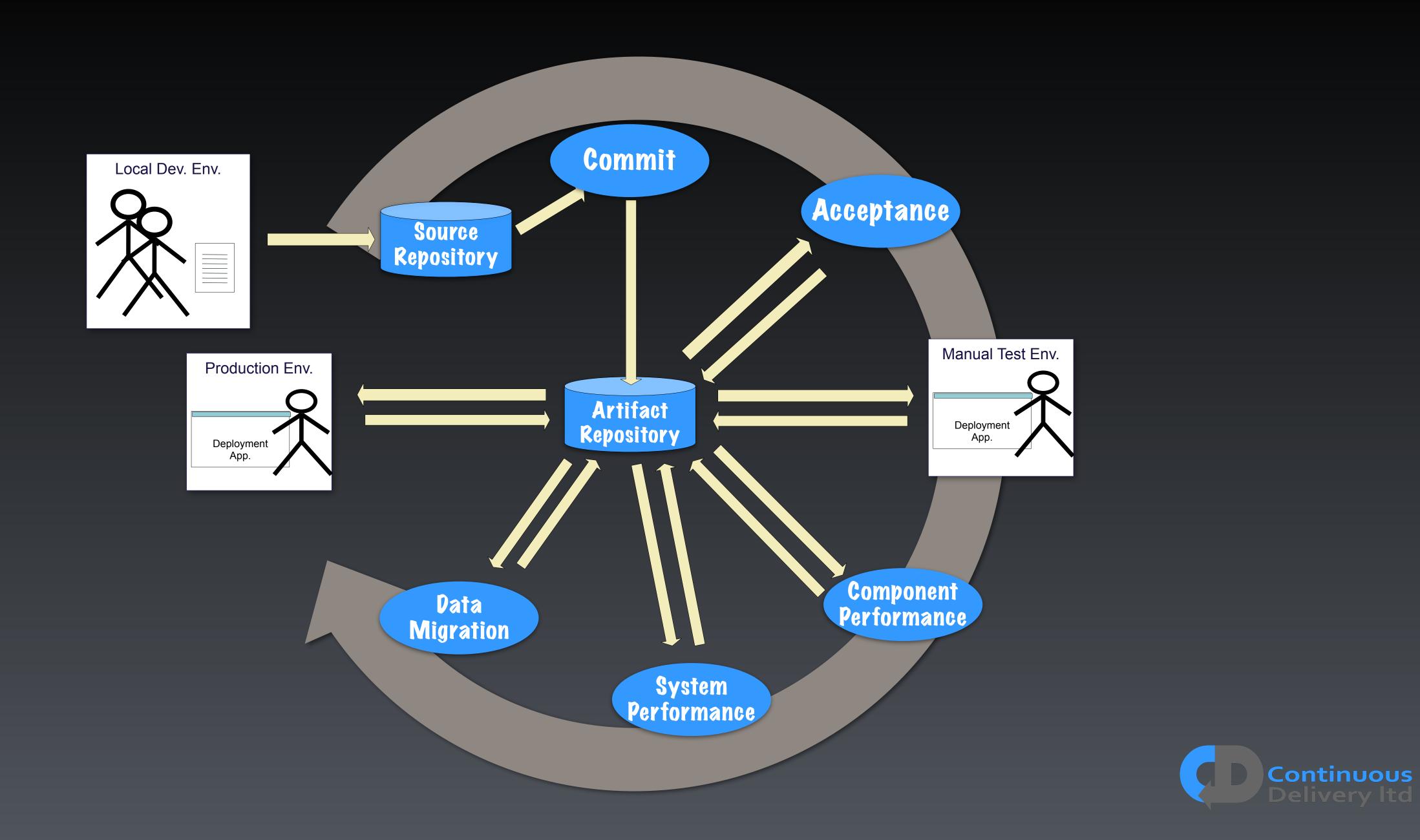




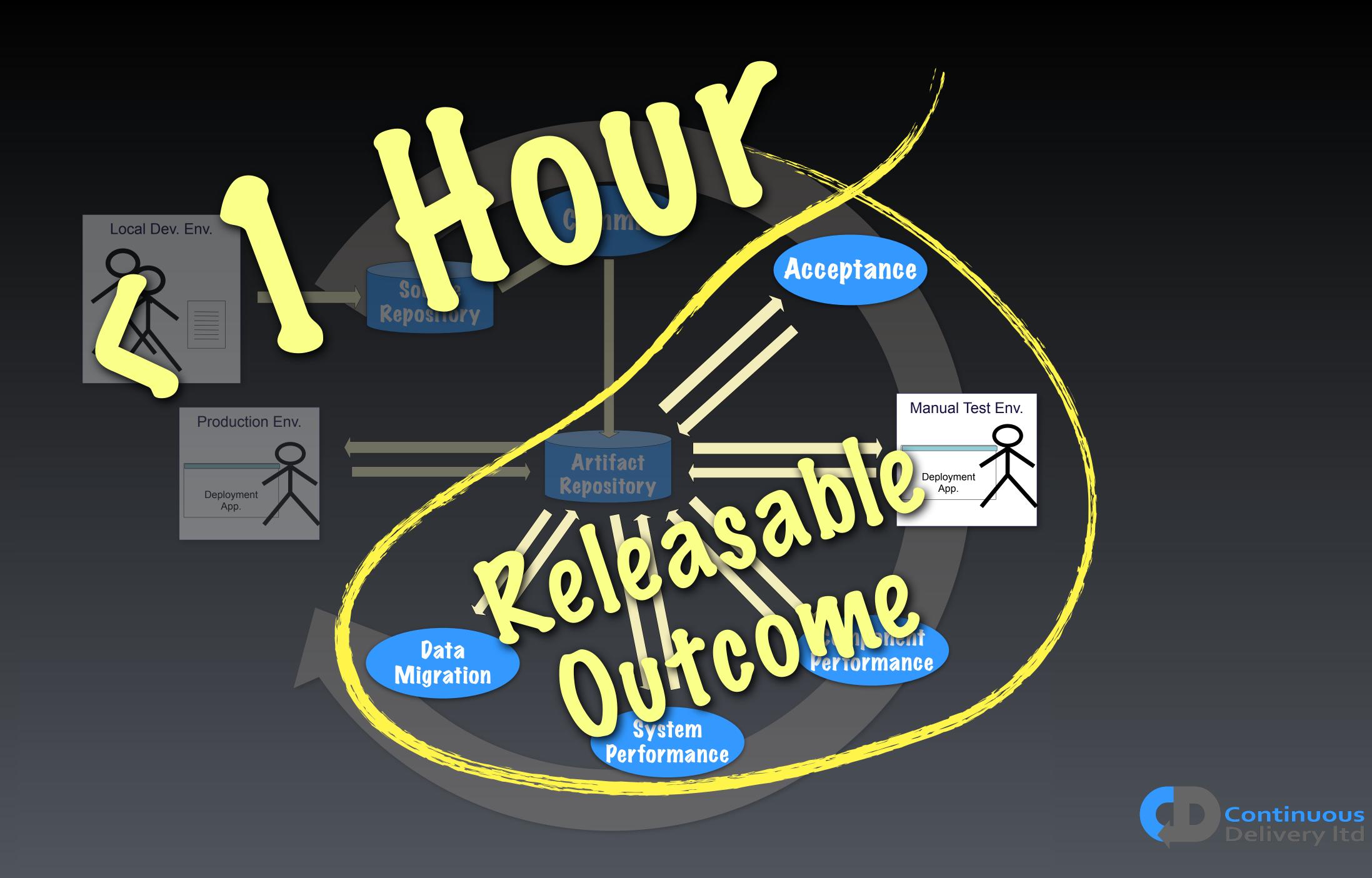








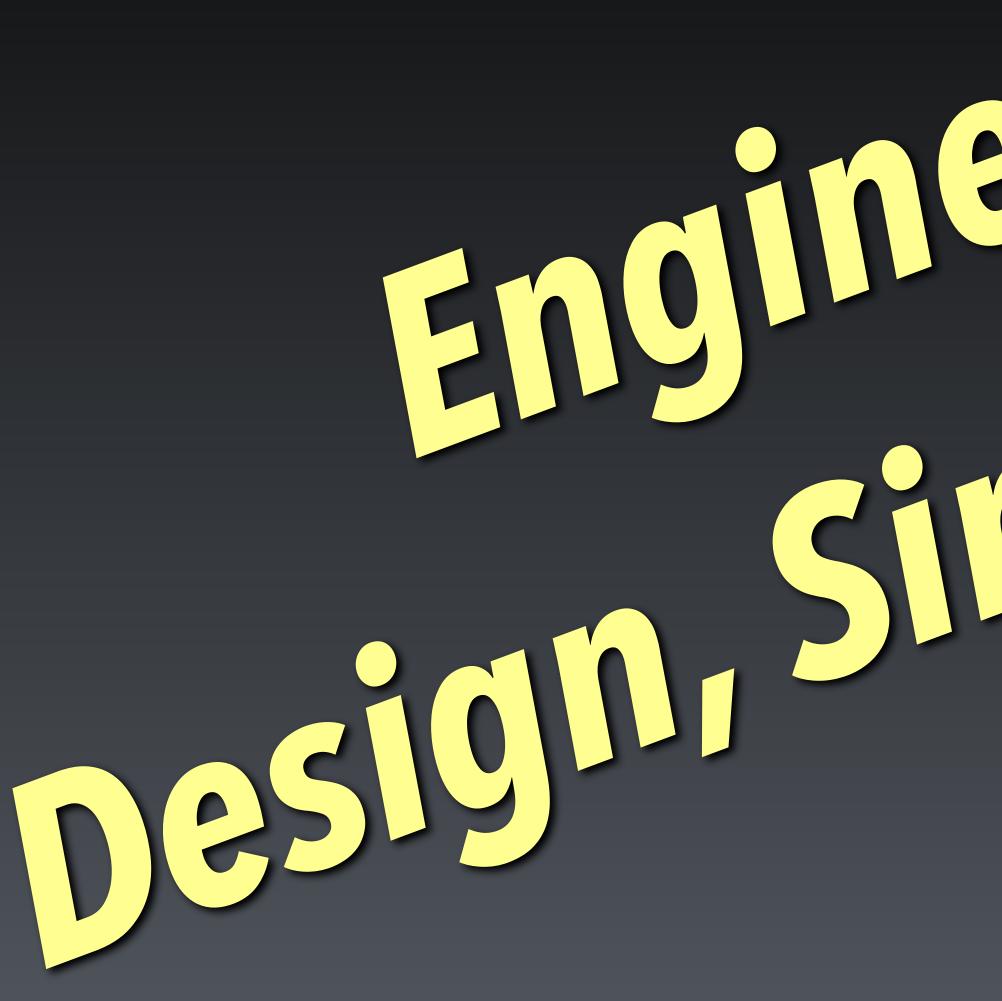






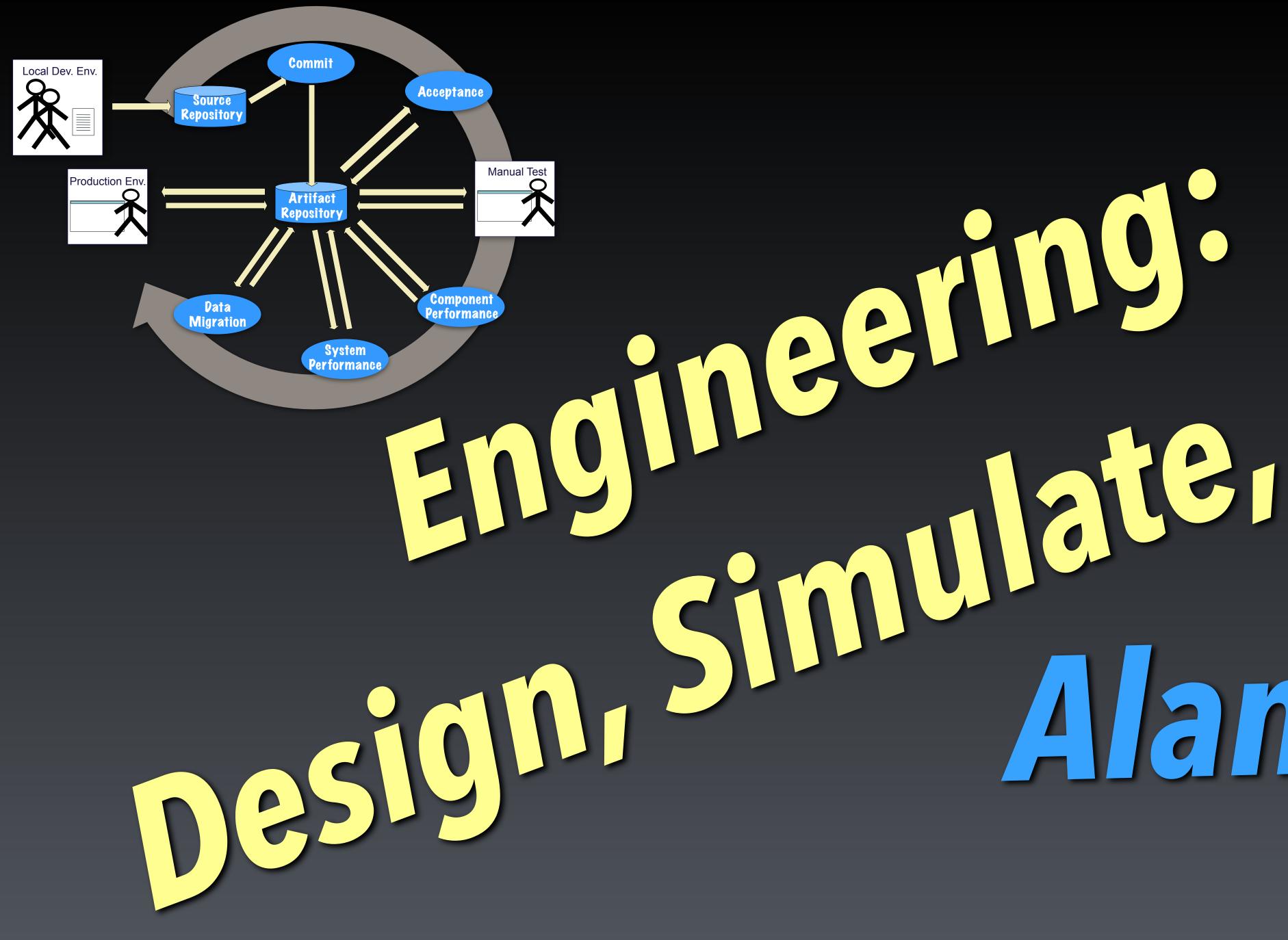






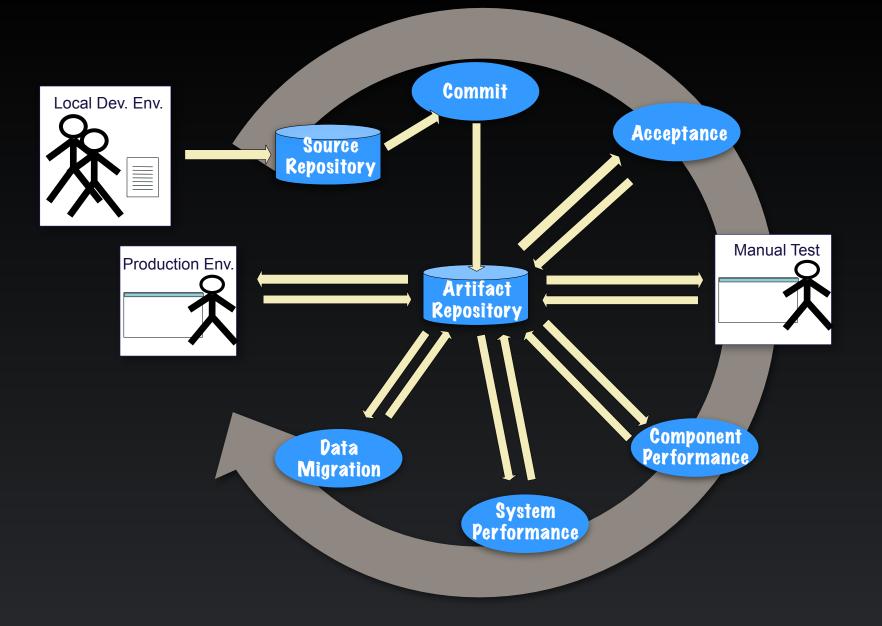
# engineering. nesign, simulate, Build





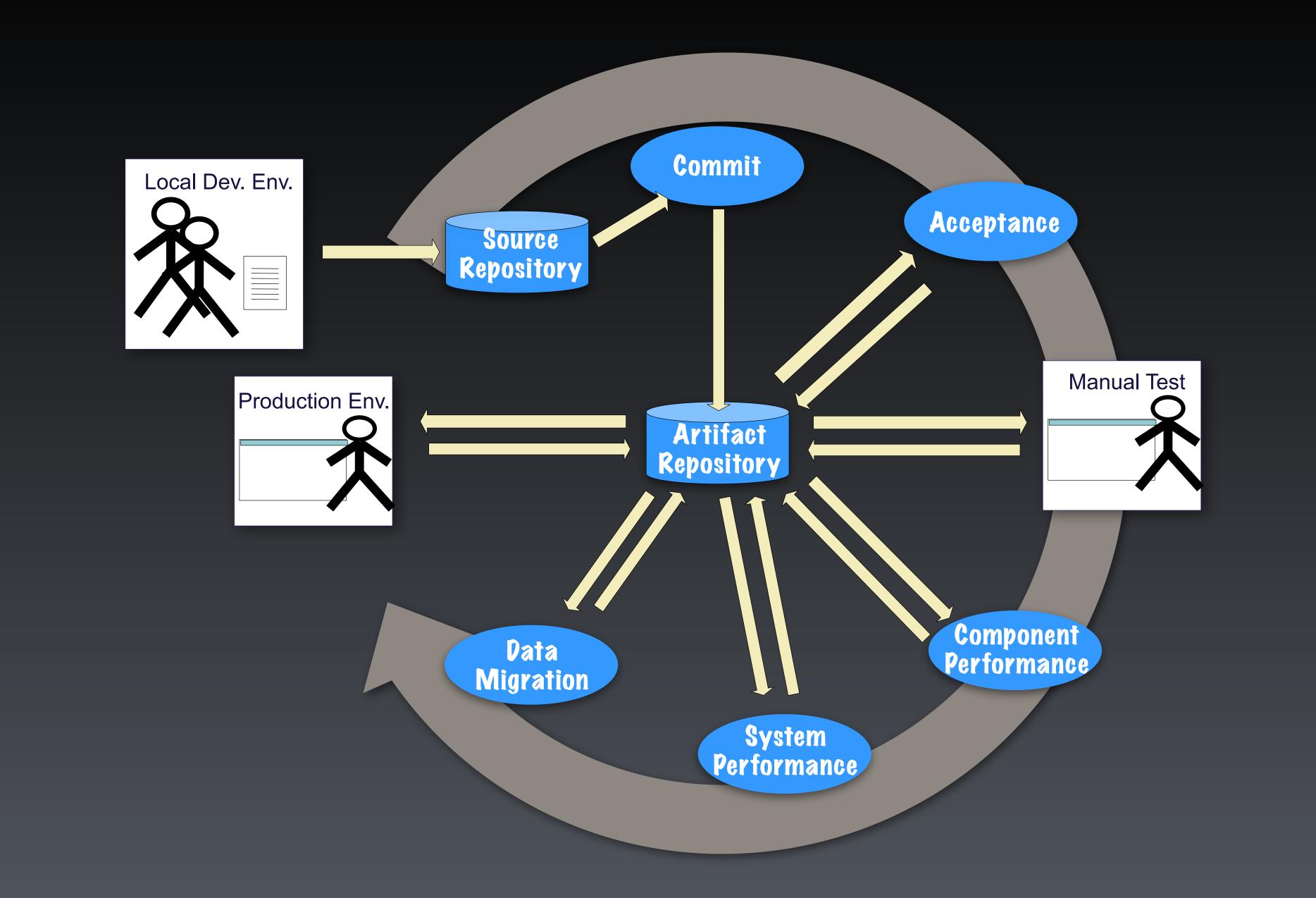
# cinclater Alan





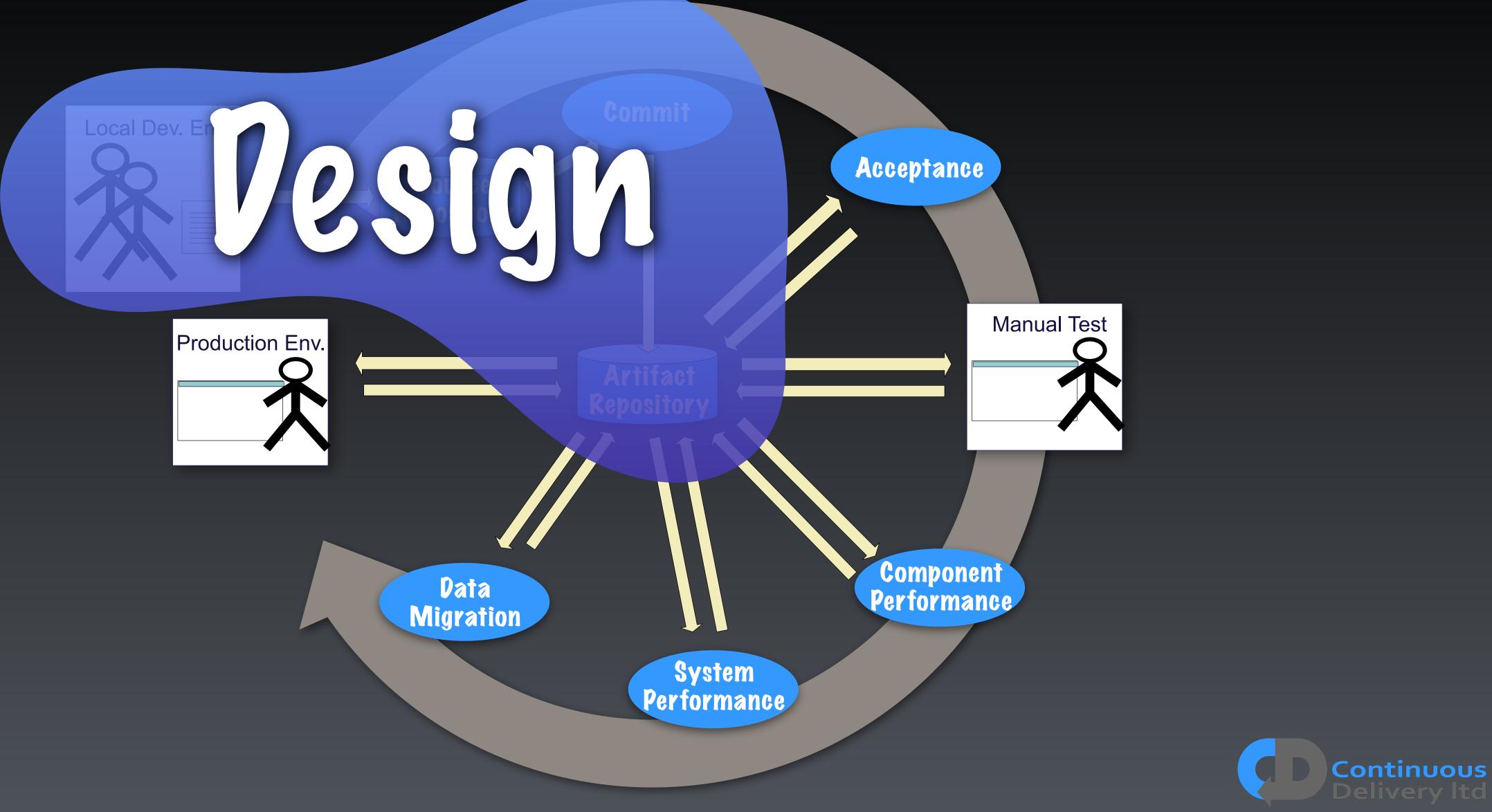




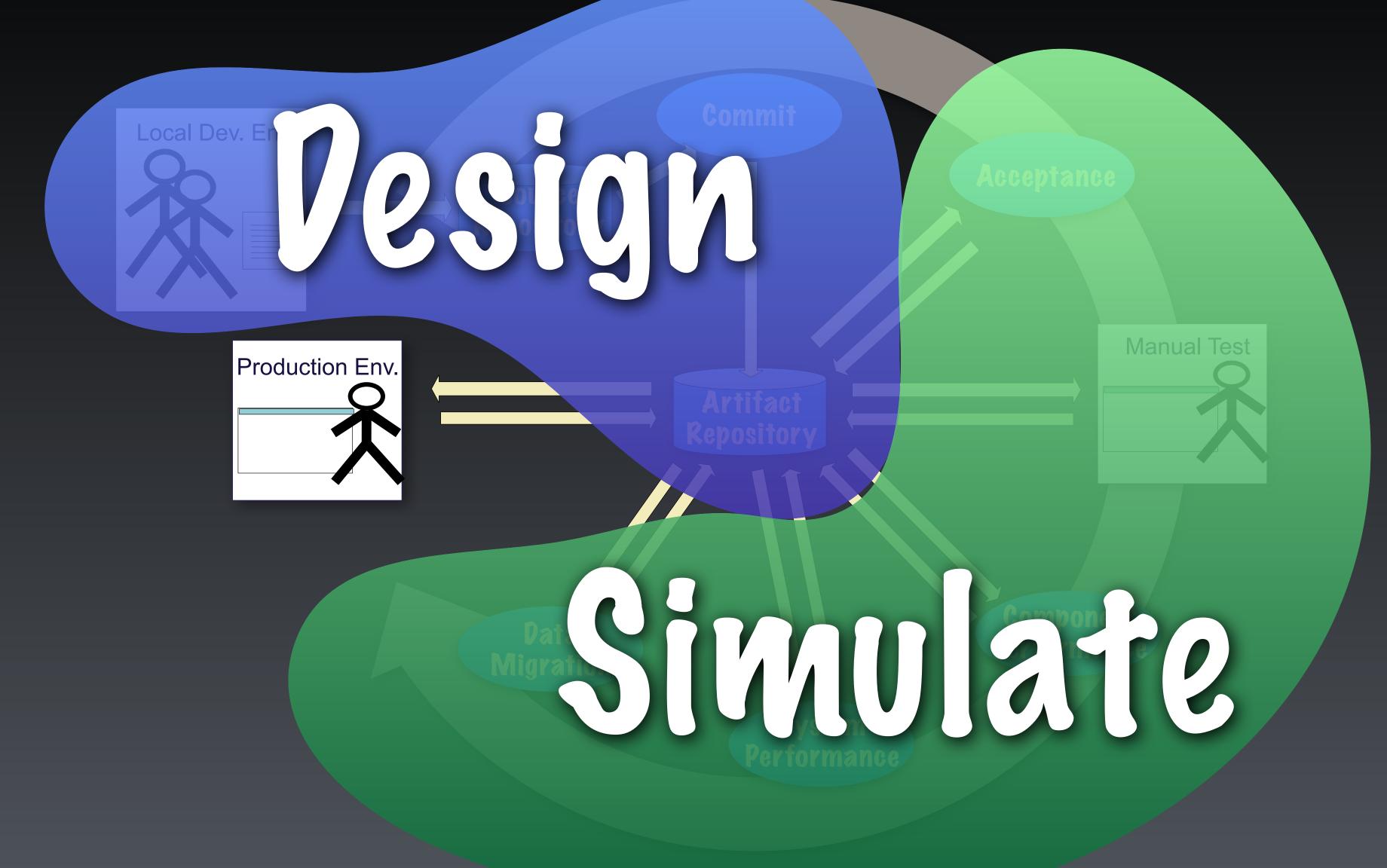






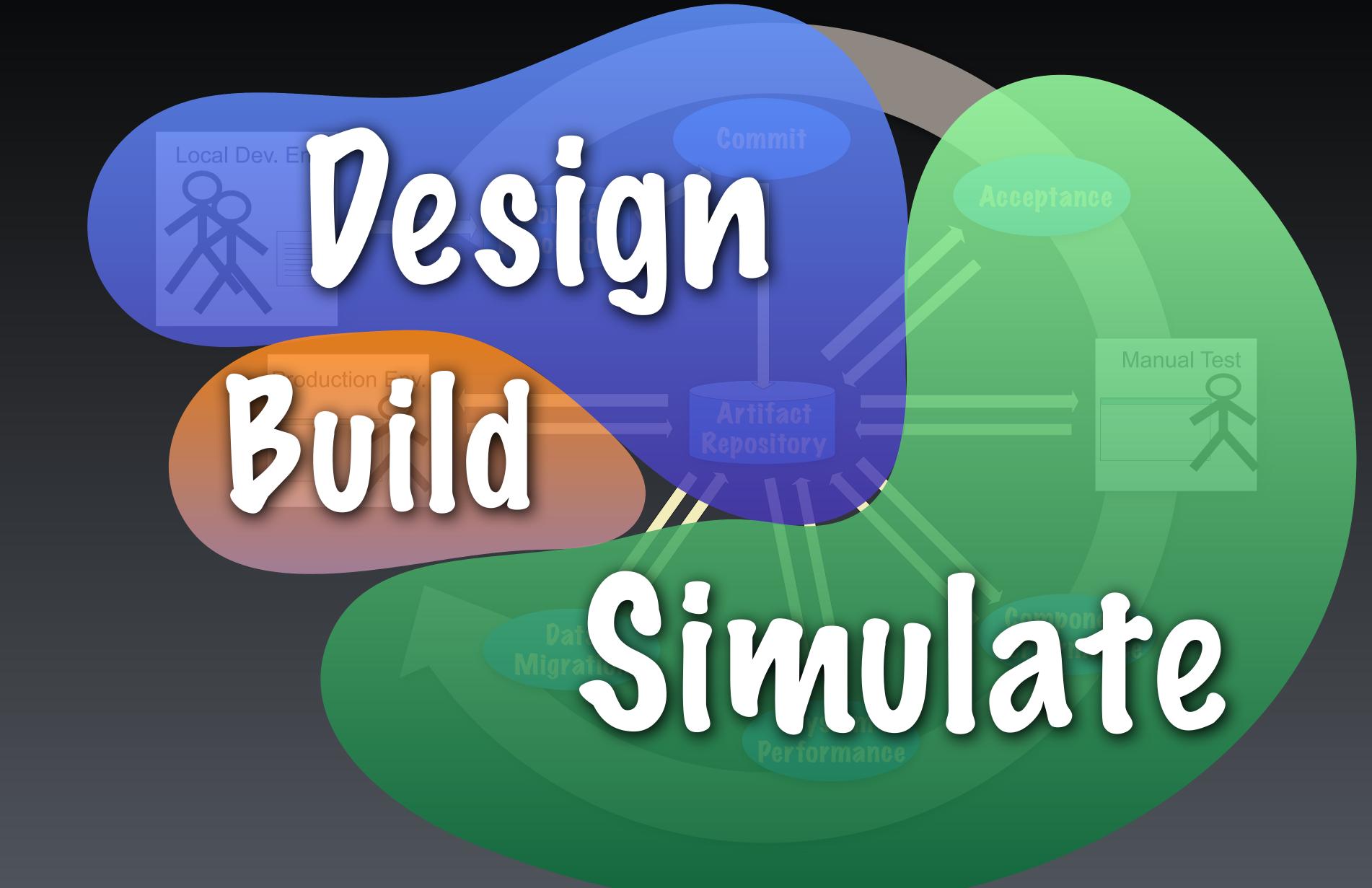
















## https://bit.ly/CDonYT

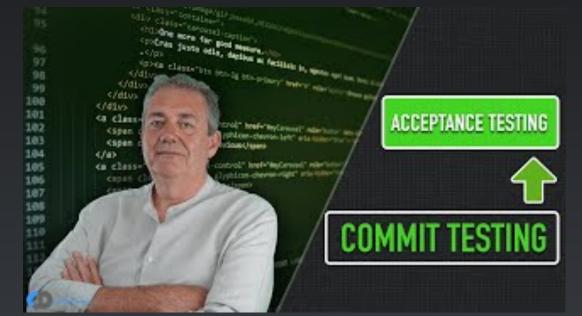




## https://bit.ly/CDonYT



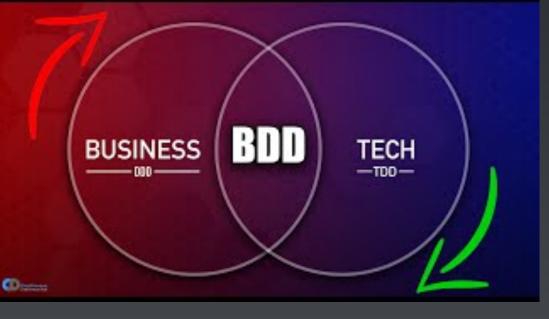














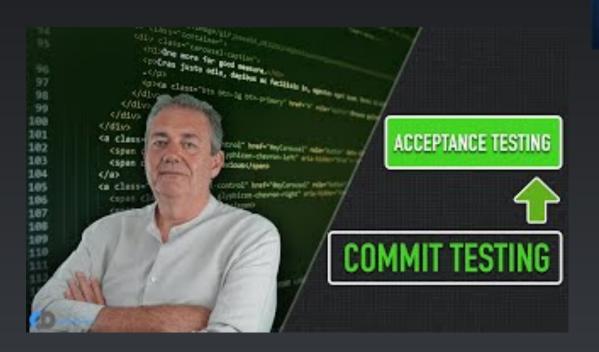


## https://bit.ly/CDonYT





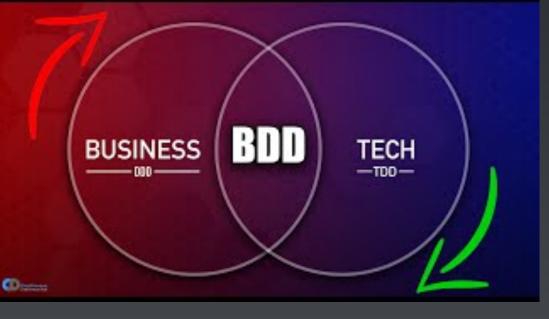
















DAVID FARLEY

### MODERN SOFTWARE ENGINEERING

Doing What Works to **Build Better Software Faster** 

Foreword by TRISHA GEE

**•** 





DAVID FARLEY

### MODERN SOFTWARE ENGINEERING

Doing What Works to

 $\bullet$ 

# Use the Code: MODERNSWENG For 35% Off!

# Pre-Order https://informit.com/modernsweng













#### **Dave Farley**

https://www.davefarley.net @davefarley77 https://bit.ly/CDonYT



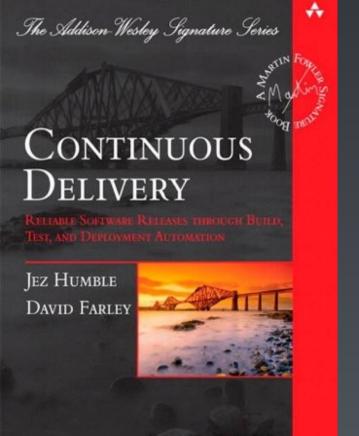
#### **Continuous Delivery Itd**

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

#### DAVID FARLEY 🔶

#### MODERN SOFTWARE ENGINEERING

Doing What Works to Build Better Software Faster Foreword by TRISHA GEE



Foreword by Martin Fowler

CONTINU DELIVE DELIVE PIPELIN

How To Build Better Software Faster



DAVE FARLEY

