After Acceptance: Reasoning About System Outputs

Dr. Stefanos Zachariadis

@thenewstef

https://moto.co.de

https://cyclema.ps

http://itv.com

OUTLINE

- Issues not typically caught by a CI environment
- What we'd like to test *after* the acceptance testing phase
- How we can achieve this systematically

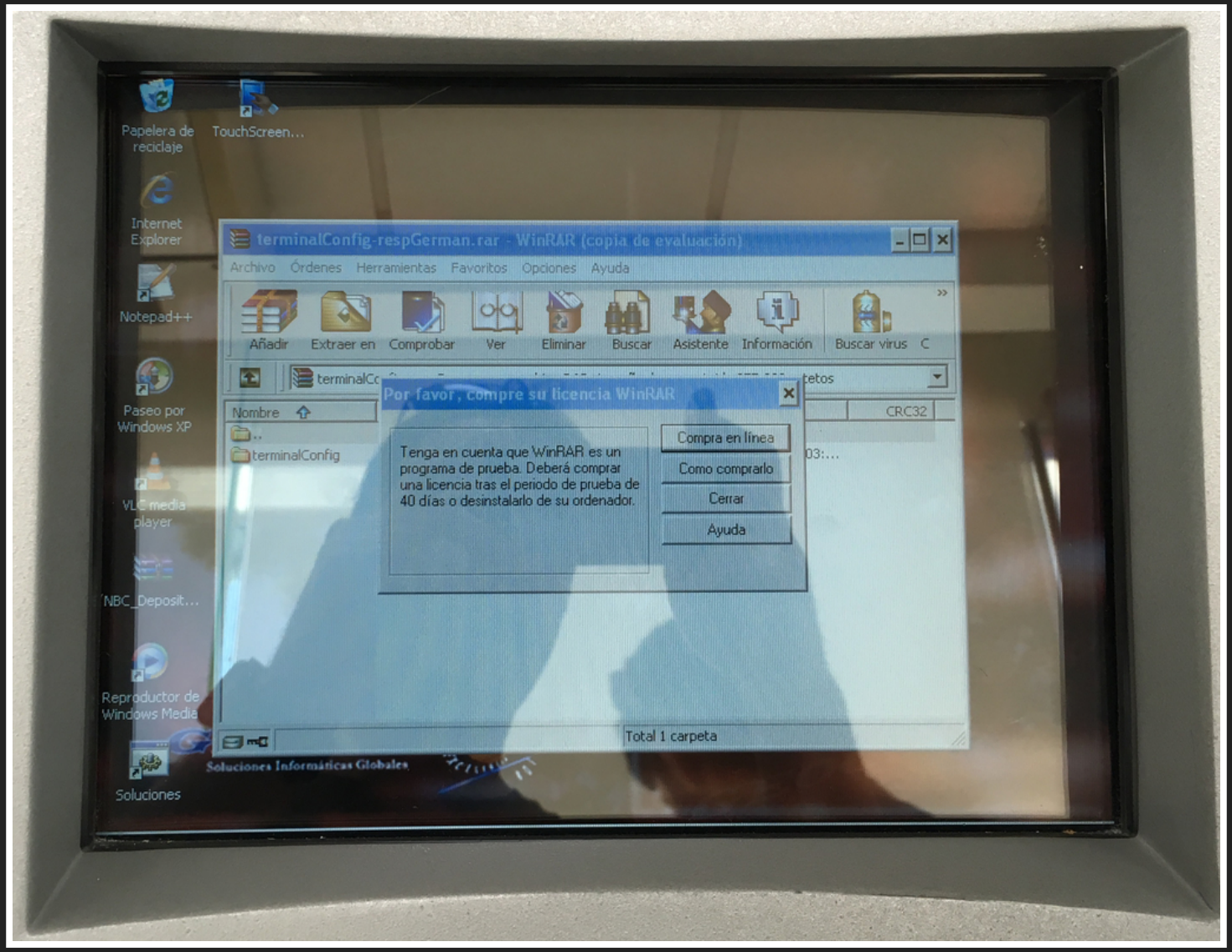
WHY ARE WE HERE


London 2012
London 2012
London 2012
London 2012
London 2012
London 2012


Ω OMEGA
London 2012
TRIYATNO TRIYATI
INA
13
188 KG
London 2012

actual photo of ATM in Uruguay taken Feb 2017

OH LOOK, TOUCHSCREEN STILL ACTIVE




£20
BANK of CTHULHU
£20
TWENTY
POUNDS
VIEW OF THE WEST FACE OF ARKHAM CATHEDRAL

```
enum AccountType {
     PERSONAL, BUSINESS, PRIVATE
}

public class Account {
       private final String name;
       private final AccountType type;
}
```

```
enum AccountType {
     PERSONAL, BUSINESS
}
```

```
public class Account {
        @NotNull
        private final String name;
        private final AccountType type;


        public boolean equals(Object obj) {
                  return name.equals(obj.name) &&
                    type.equals(obj.type);
        }

}
```

Rupert Jones

Friday 14 October 2016 16.30 BST

NatWest and Royal Bank of Scotland customers have reported having their debit cards declined in shops and at ATMs after the banking group was hit by yet another technical glitch.

The problems emerged at around 12.45pm on Friday, just as many people were popping out to buy a sandwich or do some lunch-hour shopping. The problem coincided with payday for many people, and customers voiced their frustrations on the banks' Twitter and Facebook pages.

```
public class Account {
	private final String name;
	private final AccountType type;
	private final CurrencyCode currency;
}
```

```java
@Test
public void getsTheCorrectBalanceAfterADeposit() {
        //given
        Account account = new Account();

        //when
        account.deposit(200);

        //then
        assertEquals(200, account.getBalance());
}
```

software is complicated

```
class StarWarsMovies {
      private boolean[] seen = new boolean[200];
}
```

2^200 + 1

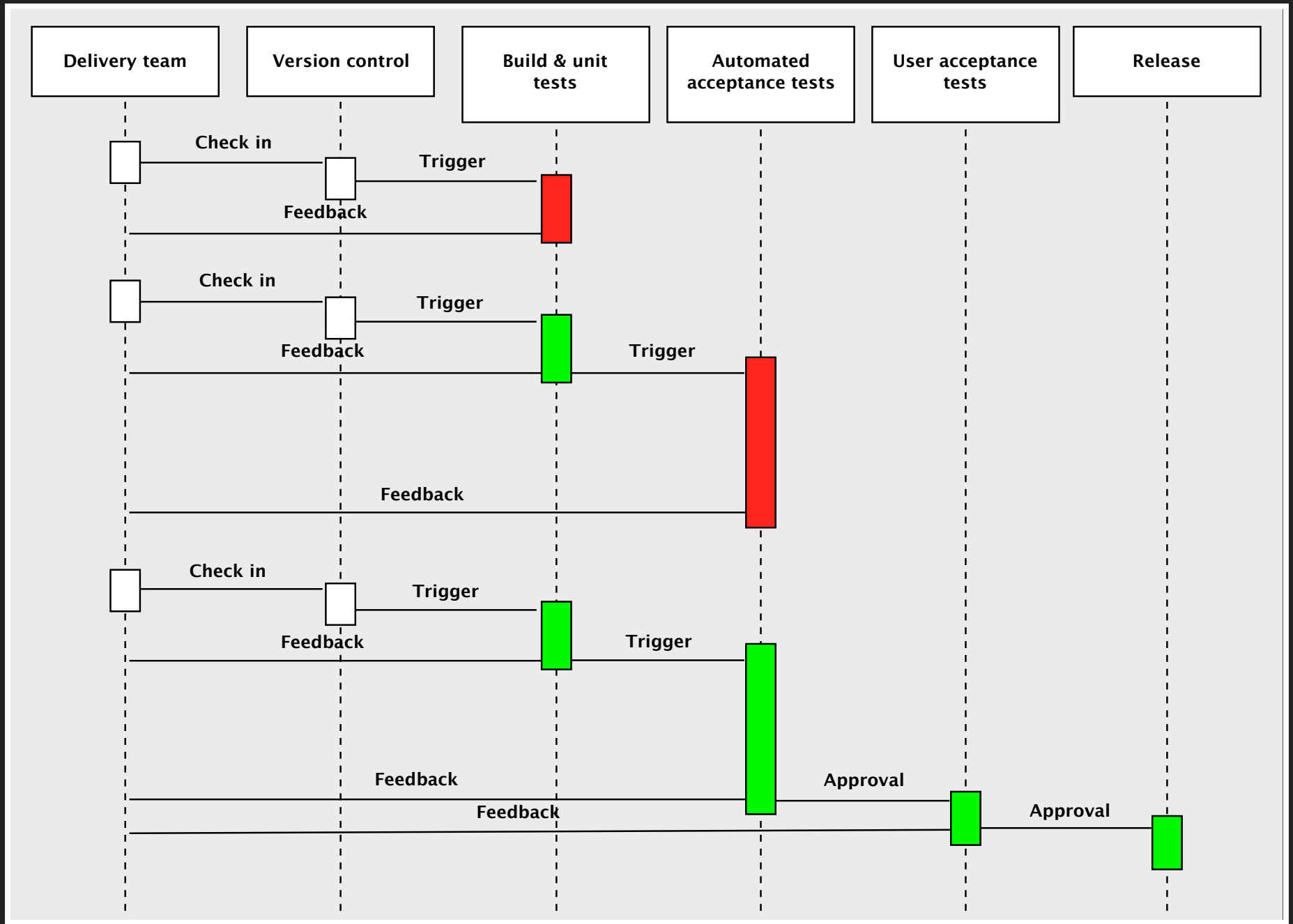
```
seen = null;
```

1 606 938 044 258
990 275 541 962 092
341 162 602 522 202
993 782 792 835 301
376 + 1

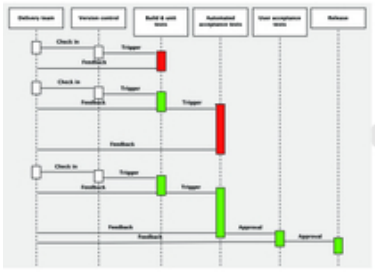
testing cannot be *exhaustive*

continuous delivery of stateful systems is hard

2^200


Delivery team
Version control
Build & unit tests
Automated acceptance tests
User acceptance tests
Release
Check in
Trigger
Feedback
Check in
Trigger
Feedback
Trigger
Feedback
Check in
Trigger
Feedback
Trigger
Feedback
Approval
Feedback
Approval

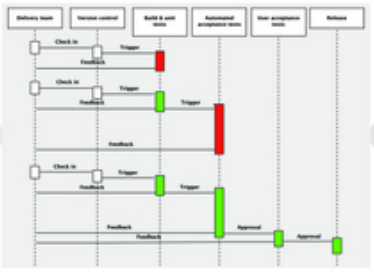
2^200



Version 1

*

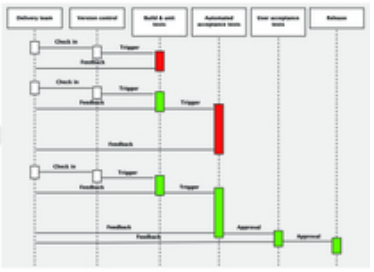
2^200



Version 2

*

2^200



Version 3

```java
@Test
public void getsTheCorrectBalanceAfterADeposit() {
        //given
        Account account = new Account();

        //when
        account.deposit(200);

        //then
        assertEquals(200, account.getBalance());
}
```

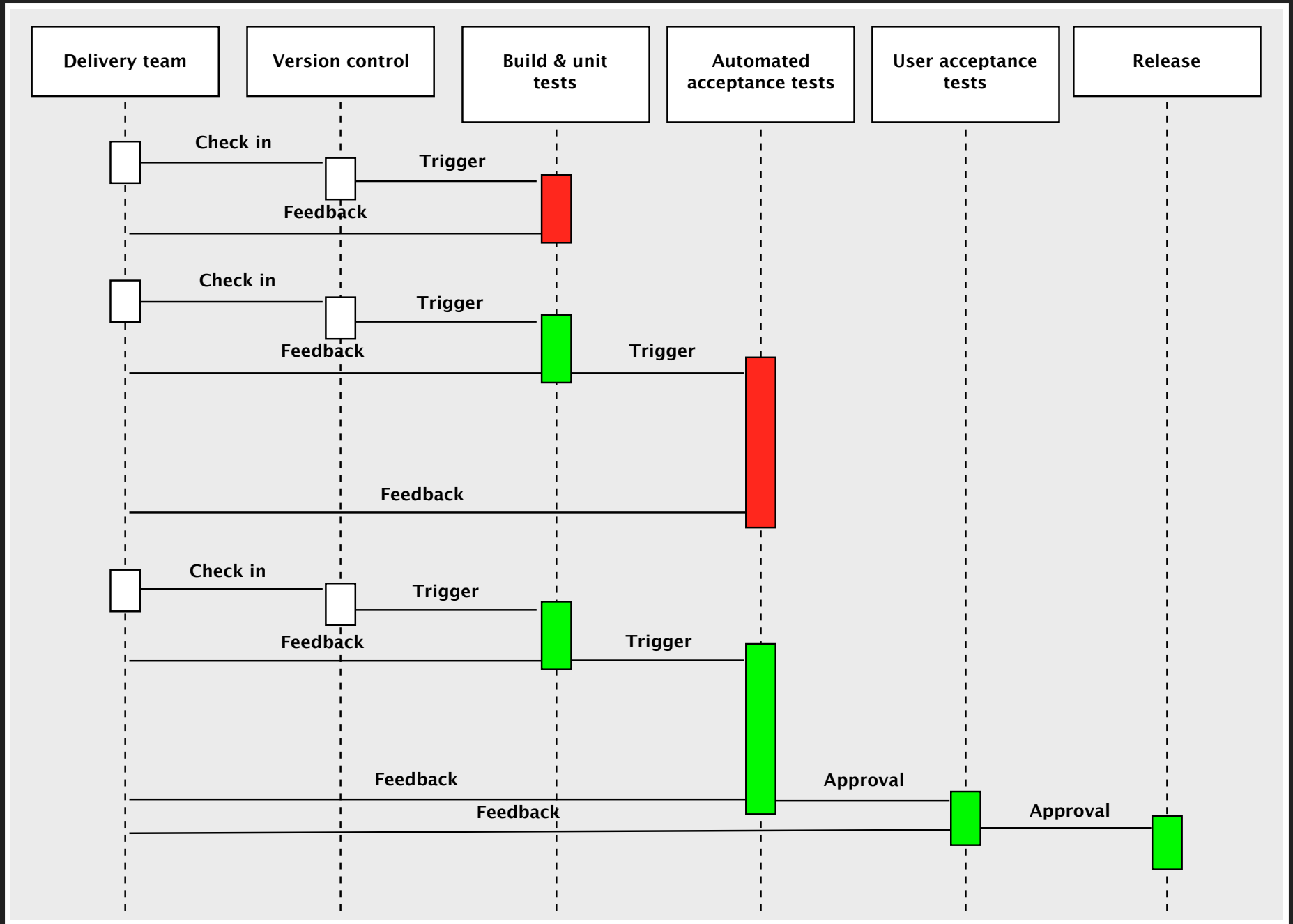
Your current account balance is a result of:

- Deposits and withdrawals
- Charges and deposits by other actors
- Data migrations
- Exchange rates
- Multiple system releases

over lots of time

WHY IT MAY BREAK

- system state
- over multiple releases


Delivery team
Version control
Build & unit tests
Automated acceptance tests
User acceptance tests
Release
Check in
Trigger
Feedback
Check in
Trigger
Feedback
Trigger
Feedback
Check in
Trigger
Feedback
Trigger
Feedback
Approval
Feedback
Approval

A NEW HOPE

WHAT TO TEST

Data validity

All data can be loaded

```
@Test
public void allAccountsAreReadable() {
        final AccountDao accountDao = new AccountDao();
        for(User user : users) {
                final Account account
                        = accountDao.loadAccountFor(user);
                verify(account);
        }
}
```

Business level validation

```
@Test
public void accountsBalanceAboveOverdraftLimit() {
        for(User user : users) {
                Account account =
                        accountDao.loadAccountFor(user);
                        assertTrue(account.getBalance()
                                > NEGATIVE_OVERDRAFT_LIMIT);
        }
}
```

WHAT TO TEST

Data invariance

1. capture invariants
2. upgrade
3. ???
4. verify invariants (& profit)

```
@Test
public void accountBalanceIsMaintained() {
        for(User user : users) {
                Account account =
                   accountDao.loadAccountFor(user);

                BigDecimal productionBalance =
                prodData.getBalance(account.getId());

                assertEqual(productionBalance,
                            account.getBalance());
        }
}
```

WHAT TO TEST

Migration integrity

```
@Test
public void allAccountsAreInUSD() {
        for(User user : users) {
                Account account =
                   accountDao.loadAccountFor(user);
                assertEqual(CurrencyCode.USD,
                            account.getCurrency())
        }
}
```

WHAT TO TEST

Data volume

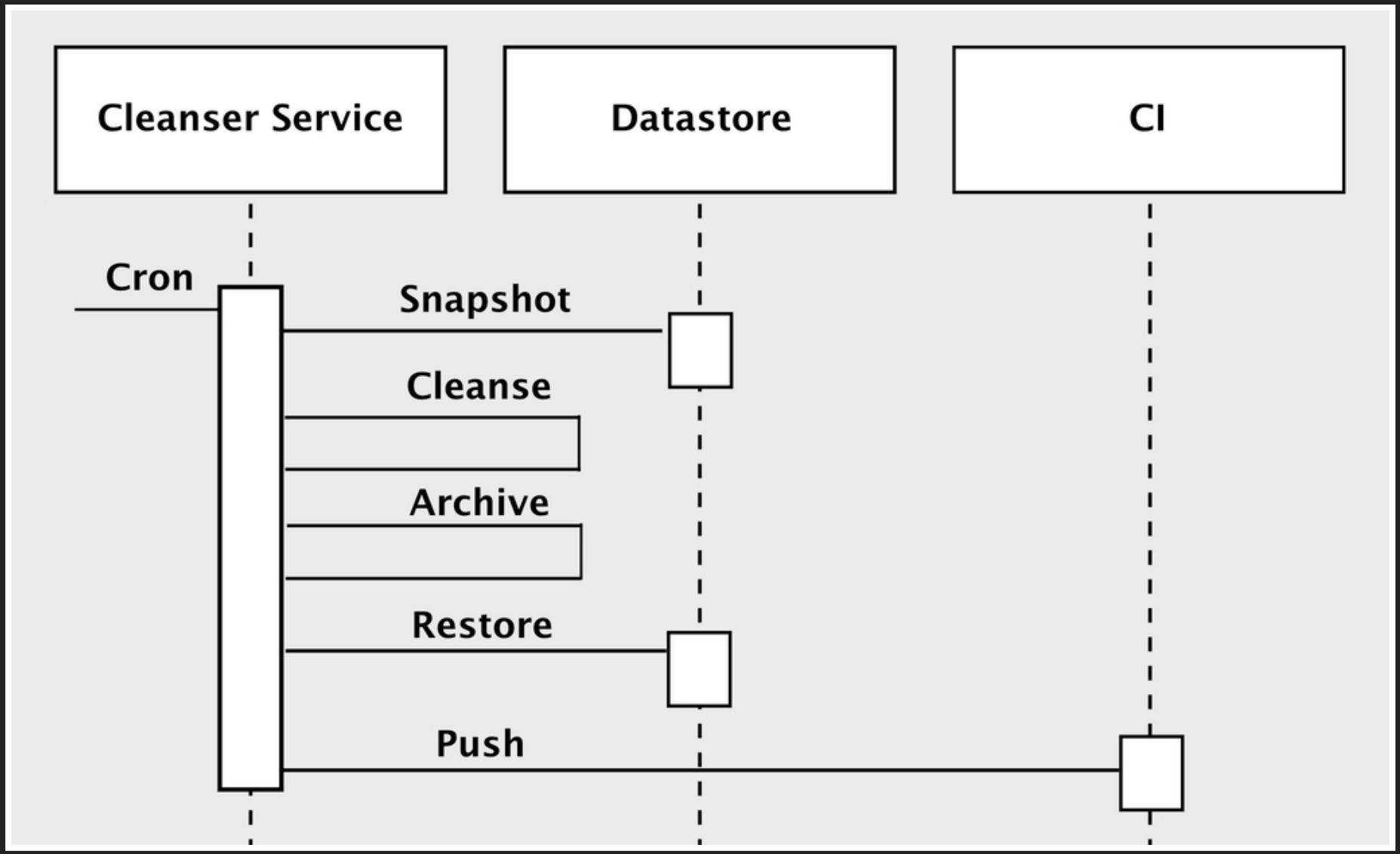
```
@Test
public void generatesARiskReport() {
        final AccountDao accountDao = new AccountDao();
        final RiskReportGenerator riskReportGenerator =
                new RiskReportGenerator(accountDao);
        time(riskReportGenerator::make, 60, TimeUnit.MINUTES)
}
```

DATA SANITISATION

first_name	last_name	account_id
John	Smith	7

↓

first_name	last_name	account_id
Name 3	Surname 2	7


Cleanser Service
Datastore
CI
Cron
Snapshot
Cleanse
Archive
Restore
Push

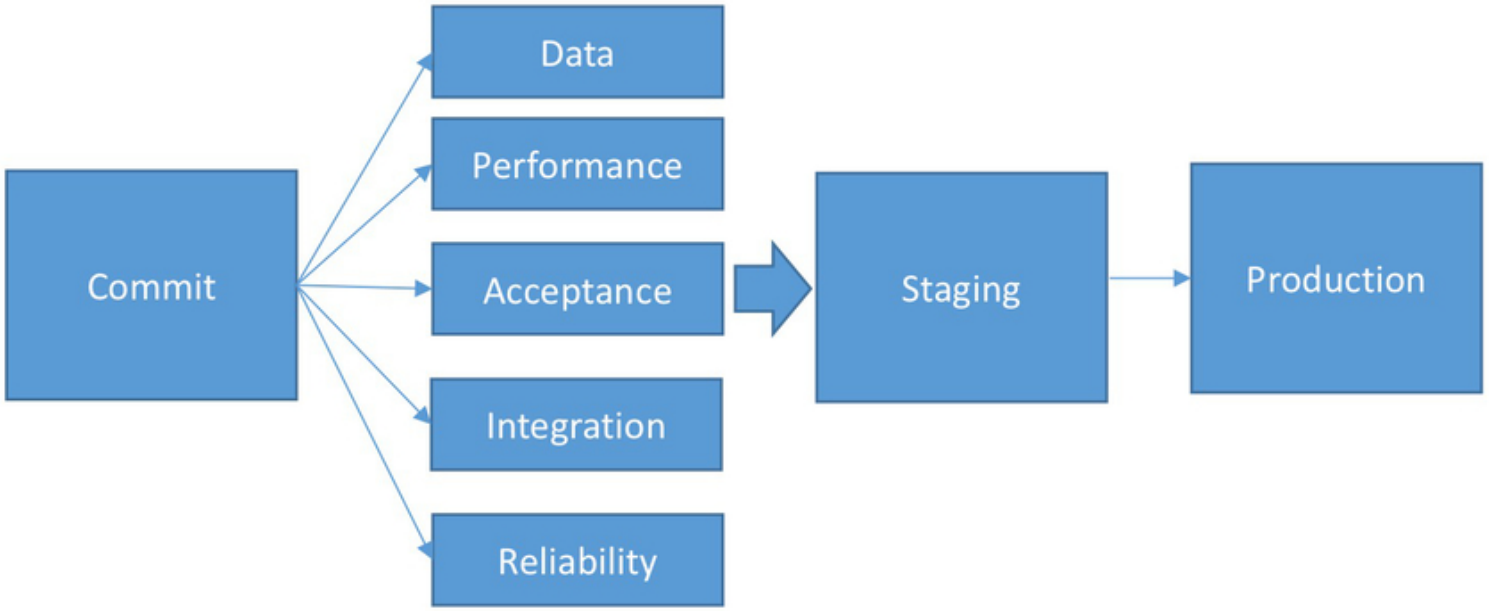
data migration

Production data → latest commit ?

Data is owned by the application. This means that the **migration process is owned by the application**, and *new migrations should ship with the application and be performed with the deployment of every new version*.

latestData = migration(prodData)

1. import the sanitised data that the cleanser produced
2. capture invariants (e.g. account balances)
3. migrate it to the latest version of the application
4. run the tests


Commit
Data
Performance
Acceptance
Integration
Reliability
Staging
Production

CONSIDERATIONS

- fight the power
- no downtime
- flip it around - send the tests to production
- frequency

MORE THAN JUST TESTING

- staging
- passwords

OH IT CAN BE SO SLOW

- Sampling
- Incremental updates
- Rolling back

CONCLUSIONS

- data testing: integration testing of commit with prod - like data
- bring production - like data into your CD pipeline
- use a cleanser to make this legal
- migration is integral to the application
- allows you to catch a whole new category of bugs
- facilitates frequent releases

Thank you!

https://moto.co.de

https://cyclema.ps

http://itv.com

pictures by @thenewstef apart from CD (wikipedia), Data, Jar Jar, bank of chthulhu (unattributed)