

UNIVERSITY OF TWENTE.

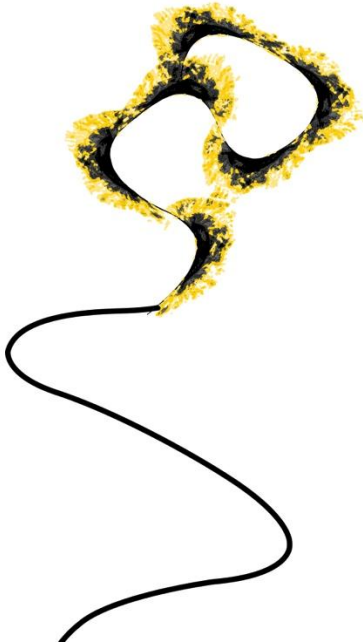
everett.

TRUSTED TO KNOW | HOW



MOVING TOWARDS CONTINUOUS DELIVERY

INTRODUCING TEST AUTOMATION WITH FITNESS
IN SYSTEM INTEGRATION PROJECTS



SANDRA DRENTZEN

s.drenthen@student.utwente.nl /

s.drenthen@alumnus.utwente.nl

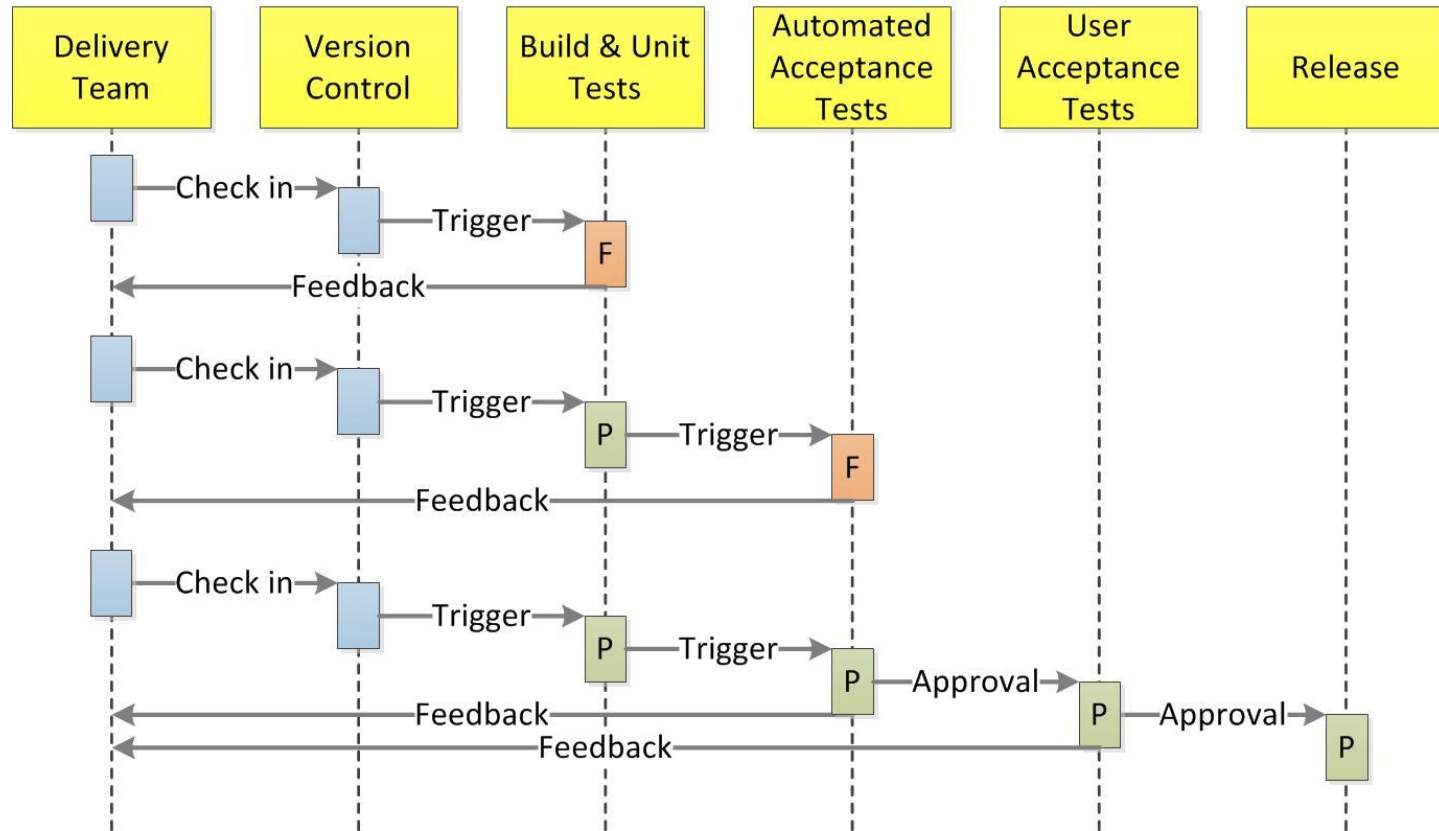
CONTENT

- Background
- Test Strategy
- Case Study
- Conclusion
- Questions

BACKGROUND

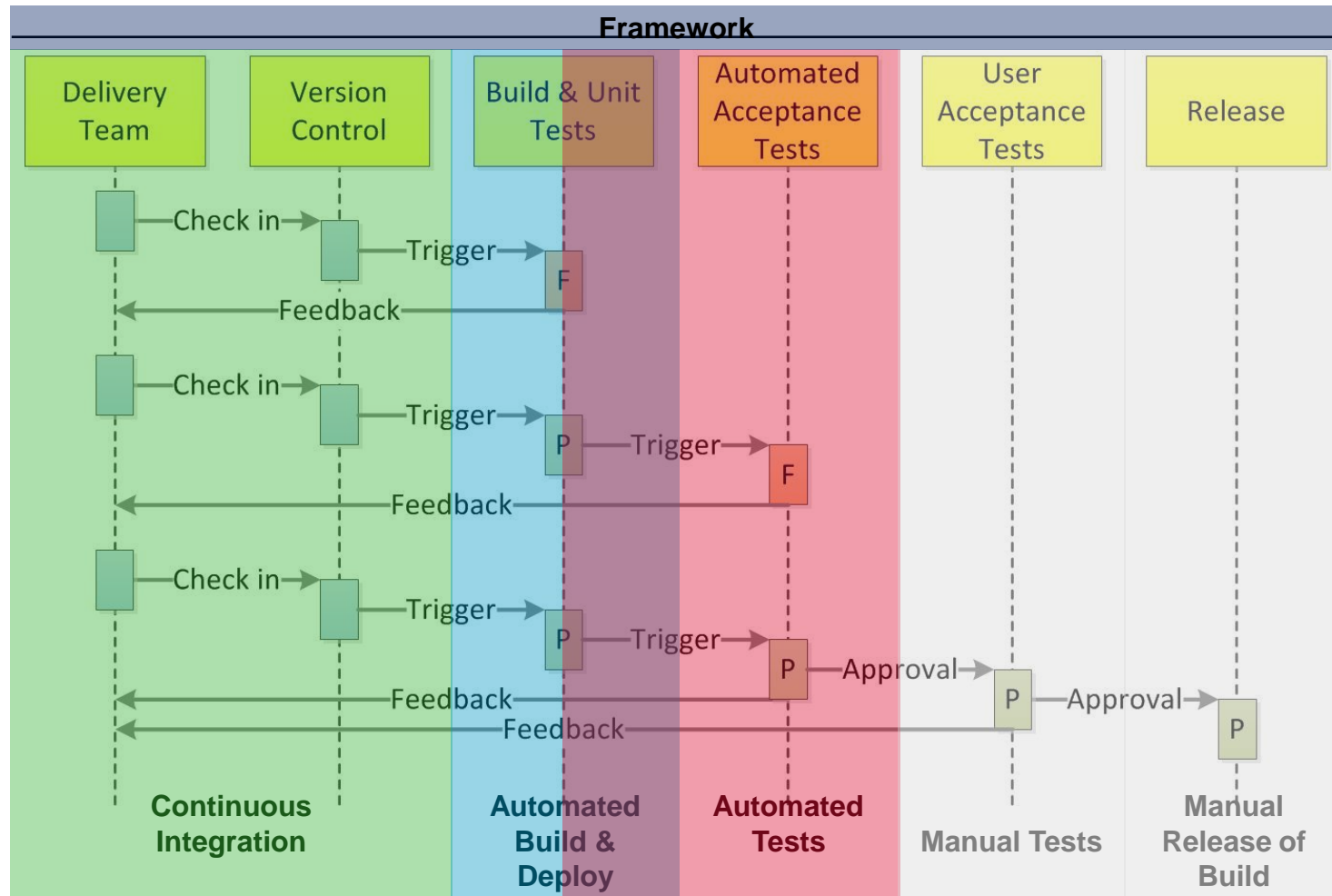


CONTINUOUS DELIVERY THEORY



[source]: J. Humble and D. Farley, *Continuous delivery: reliable software releases through build, test, and deployment automation.*, Addison-Wesley Professional, 2010.

CONTINUOUS DELIVERY THEORY



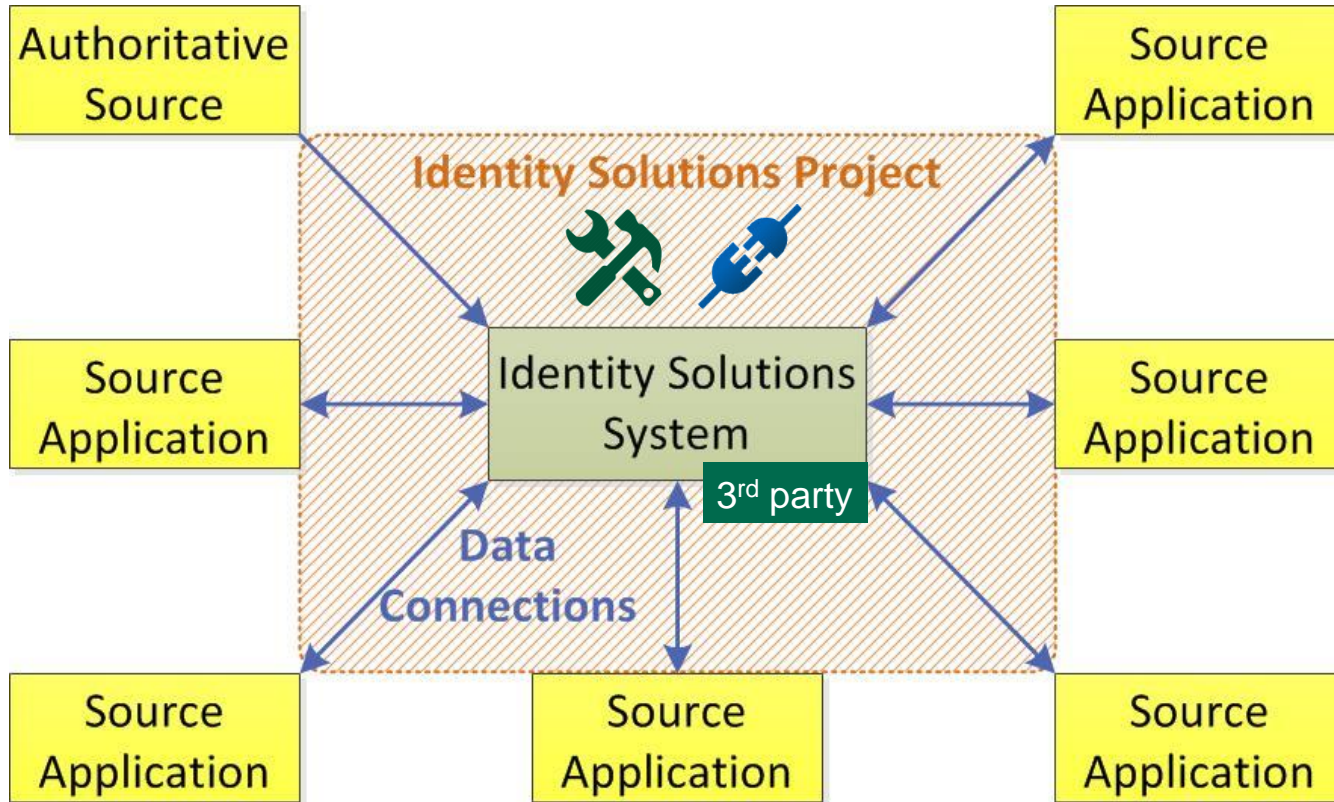
IDENTITY SOLUTIONS

IDENTITY & ACCESS GOVERNANCE

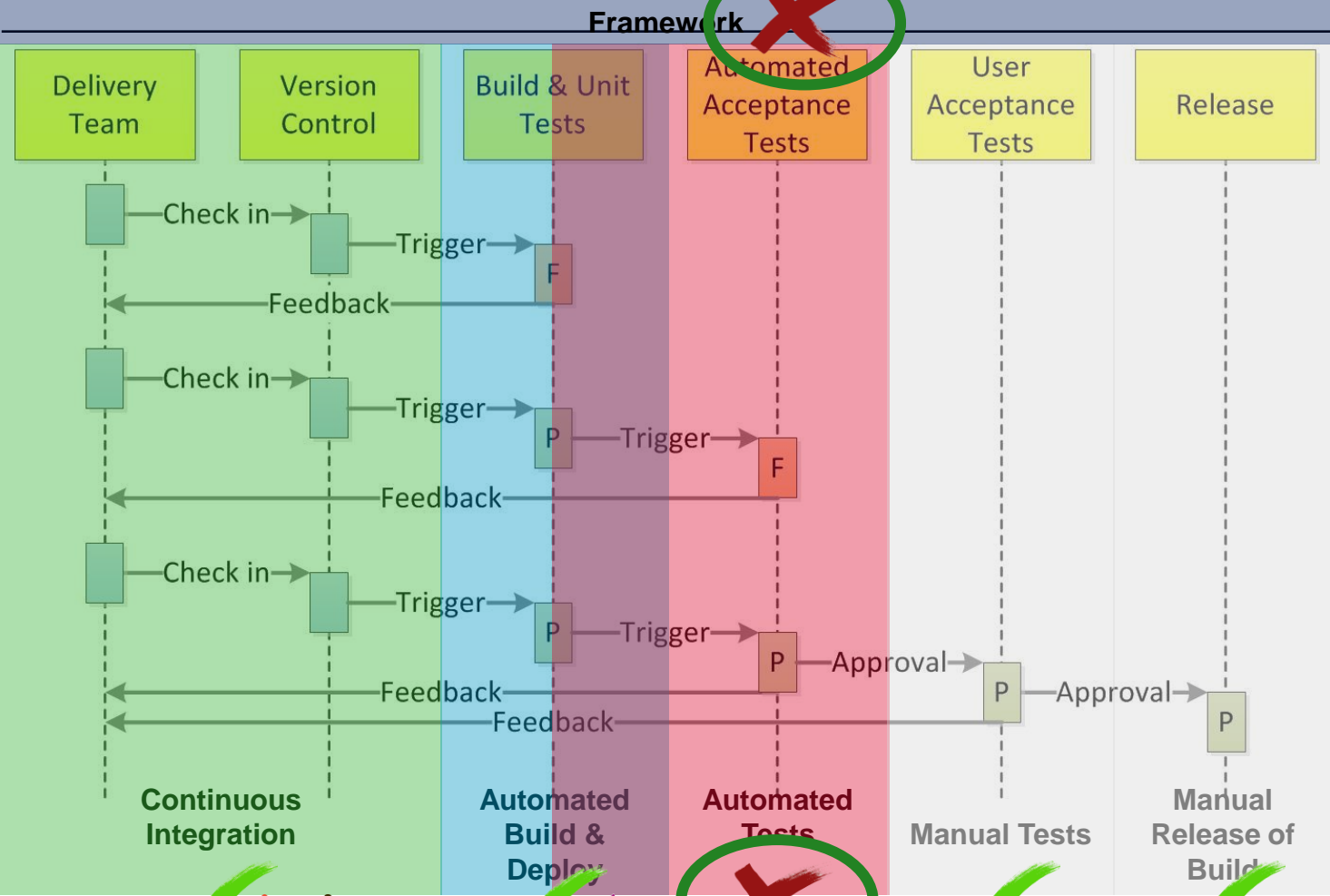
- **Main goal:**
 - **Be in control** of access rights & being able to **prove it**
- **Offers the client:**
 - **Correlate data**
 - Specify/check **rules**
 - Specify/run **certifications**
 - Possibility to implement **provisioning**
 - Possibility to define **Workflows**

IDENTITY SOLUTIONS

IDENTITY & ACCESS GOVERNANCE

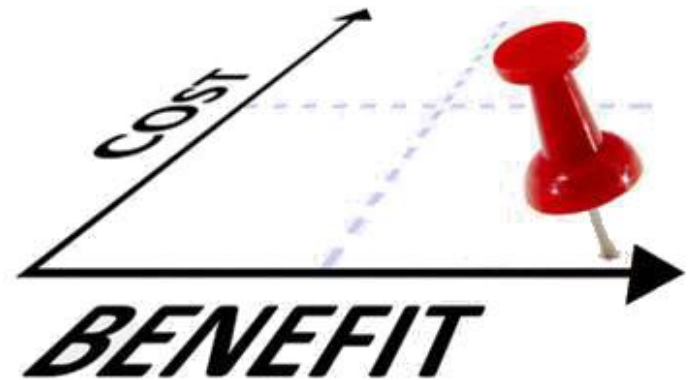


CONTINUOUS DELIVERY THEORY



GOALS

- **Automate and improve process of software delivery**
 - By introducing Continuous Delivery
- **Reduce costs of testing**
 - By re-running/re-using tests
- **Enhance quality of product**
 - By finding faults earlier
 - By having better test-documentation



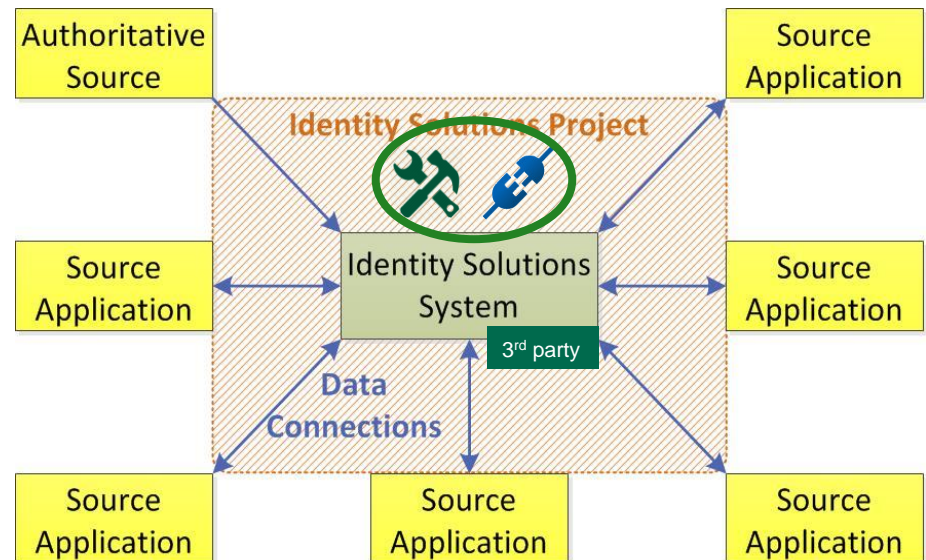
TEST STRATEGY



TEST AUTOMATION

WHAT DO YOU WANT TO AUTOMATE?

- In our case, most risks are in:
 - Configurations
 - Integrations
- Automated Tests on:
 - Functional level



[must-read]: B. Marnick, "When Should a Test Be Automated?," in *Proceedings of The 11th International Software/Internet Quality Week*, San Francisco, 1998.

TEST AUTOMATION

TEST DATA

- **Type of test-data**
 - **Production** data
 - **Masked** data
 - **Generated/Fictional** data

- **In our case:**
 - **Production** data with extra constraints

TEST AUTOMATION

HOW DO YOU COMMUNICATE WITH SUT?

- **Communicate with SUT**
 - **Find tools** that are capable do this and try a **proof of concept**.



- **Our choice:**
 - Because **functional**, communicate with the **API** of the SUT
 - Use the **FitNesse tool**



TEST AUTOMATION

FITNESSE SELLING POINTS

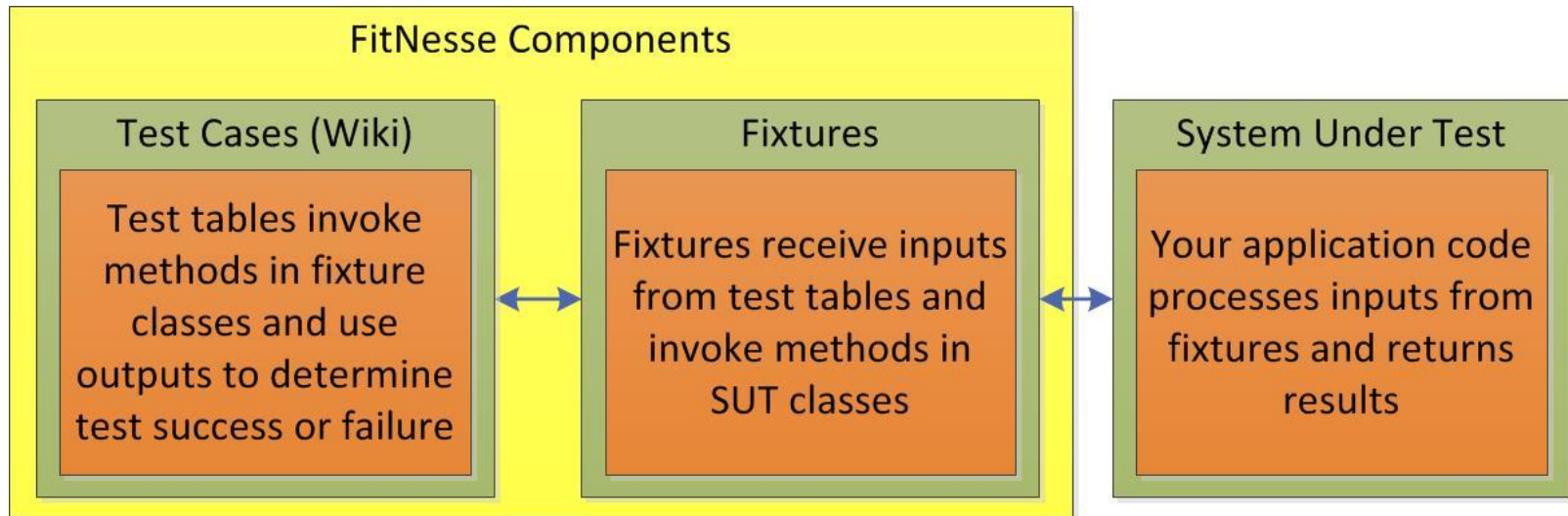
- **Automated acceptance testing framework**
 - Use it **everywhere**
cross-platform, lightweight, open-source, standalone
 - **Easy** to use and learn
wiki and java syntax
 - **Readable** tests
Tests are wiki tables
 - Define tests **together**
business users can write tests





TEST AUTOMATION

FITNESSE COMPONENTS



[source]: M. Sorens, "Acceptance Testing With FitNesse, The Overview," Juli 2013. [Online]. Available: <https://www.simple-talk.com/dotnet/.net-tools/acceptance-testing-with-fitnessse,-the-overview/>



TEST AUTOMATION

FITNESSE EXAMPLE

variable defined: `TEST_SYSTEM=slim`

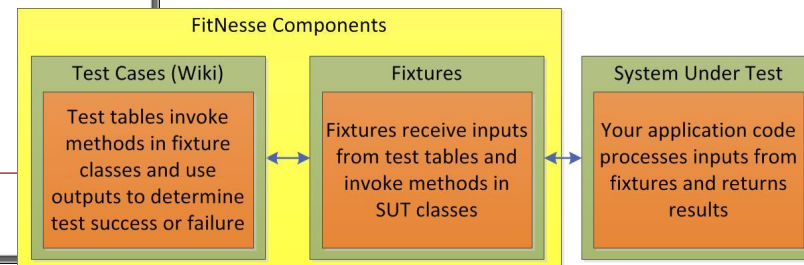
for using Eclipse with standard settings:
classpath: `../Tutorial/bin/`

```
import
jukebox.fixtures
```

Tests with script table

script	current account	
check	cash balance should be	0.0
deposit	1.0	
check	cash balance should be	1.0
\$balance=	total deposits	
ensure	withdraw	\$balance
check	cash balance should be	0.0
note	account should not allow negative balance	
reject	withdraw	1.0
check	cash balance should be	0.0
note	faults are colored red	
check	cash balance should be	8.0

[Front Page](#) | [User Guide](#)
[root](#) (for global lpath's, etc.)





TEST AUTOMATION

FITNESSE EXAMPLE

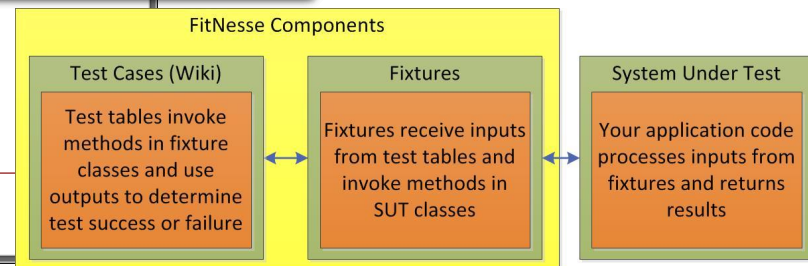
The screenshot shows the FitNesse web interface for a test suite named 'PaymentTest'. The browser address bar shows 'localhost:8080/JukeboxSuite.PaymentTest'. The page includes a 'Test' button and navigation links for 'Edit', 'Add', and 'Tools'. Under the 'Contents:' section, there is a variable definition: `variable defined: TEST_SYSTEM=slim` and instructions for using Eclipse with standard settings: `classpath: ../Tutorial/bin/`. Below this is an 'import' section with `jukebox.fixtures`. The 'Tests with script table' section contains a table of test cases:

script	current account
check	cash balance should be 0.0
deposit	1.0
check	cash balance should be 1.0
\$balance=	total deposits
ensure	withdraw \$balance
check	cash balance should be 0.0
note	account should not allow negative balance
reject	withdraw 1.0
check	cash balance should be 0.0
note	faults are colored red
check	cash balance should be 8.0

At the bottom of the screenshot, there are links for 'Front Page' and 'User Guide root (for global /path's, etc.)'.

```
package jukebox.fixtures;
public class CurrentAccount {
    //calls to SUT
}
public double cashBalanceShouldBe() {
    //calls to SUT
}
public double totalDeposits() {
    //calls to SUT
}
public void deposit(double amount) {
    //calls to SUT
}
public boolean withdraw(double amount) {
    //calls to SUT
}
}
```

System(s)
Under Test





TEST AUTOMATION

FITNESSE EXAMPLE

Test Results: JukeboxSuite x
localhost:8080/JukeboxSuite.PaymentTest?test

JukeboxSuite
PaymentTest

Tests Executed OK Test Edit Add Tools

Failure Navigator: < of 1 >

Assertions: 6 right, 1 wrong, 0 ignored, 0 exceptions (0,104 seconds)

Contents:

variable defined: TEST_SYSTEM=slim

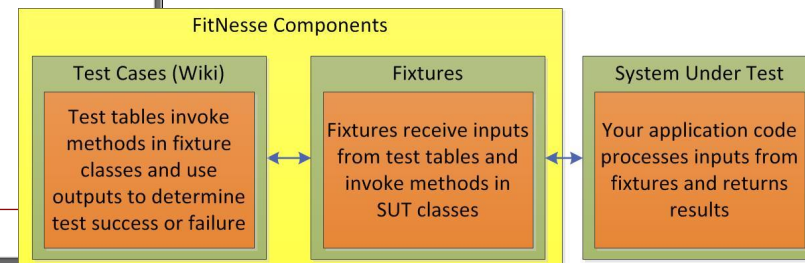
for using Eclipse with standard settings:
classpath: ../Tutorial/bin/

```
import
jukebox.fixtures
```

Tests with script table

script	current account
check	cash balance should be 0.0
deposit	1.0
check	cash balance should be 1.0
\$balance<- [1.0]	total deposits
ensure	withdraw \$balance->[1.0]
check	cash balance should be 0.0
note	account should not allow negative balance
reject	withdraw 1.0
check	cash balance should be 0.0
note	faults are colored red
check	cash balance should be [0.0] expected [8.0]

[Front Page](#) | [User Guide](#)
root (for global 'path's, etc.)





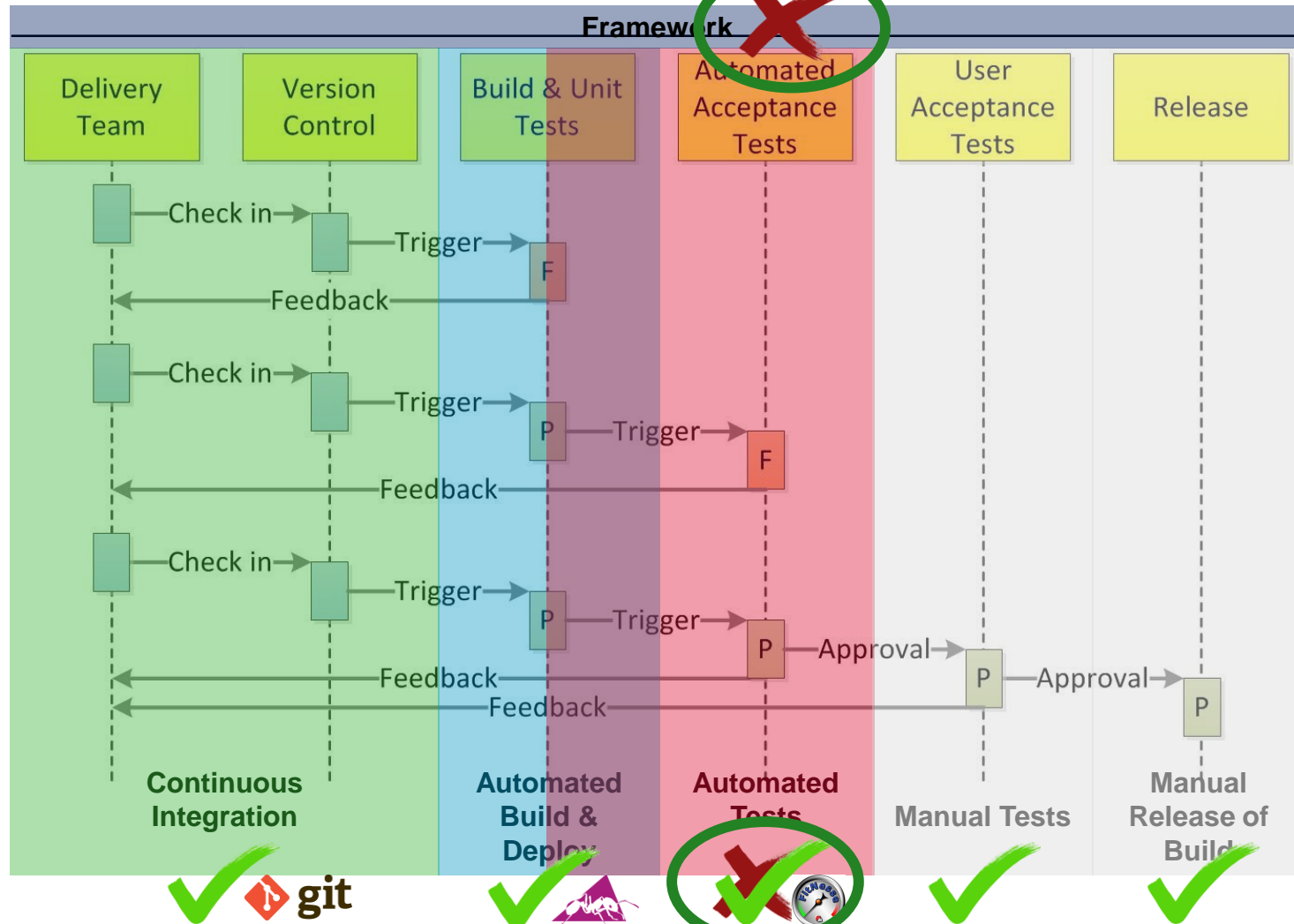
TEST AUTOMATION

FITNESSE USAGE

- **FitNesse Guidelines**
 - Combine with **Test Driven Development**
 - Use FitNesse's **SLIM-test system** (instead of FIT)
 - Use FitNesse's **SetUp/TearDown** pages
 - **Group** tests in suites, based on **functionality**
 - **Work together** by using **GIT** for FitNesse as well

CONTINUOUS DELIVERY

STATUS IN EVERETT'S PROJECTS





AUTOMATED BUILD, TEST AND DEPLOY

OUR CHOICE: JENKINS

Jenkins

sdrenthen | Afmelden

ENABLE AUTO REFRESH

Jenkins > build-war >

- Terug naar Dashboard
- Status
- Wijzigingen
- Werkplaats
- Start nu een bouwpoqing
- Verwijder Project
- Configureer

Project build-war

voeg omschrijving toe

Deactiveer project

Werkplaats

Laatste succesvol gebouwde artefacten

- .war 151,64 MB
- .war.MD5 34 B

Recente wijzigingen

Laatste testresultaten (geen gefaalde testen)

Latest FitNesse Results (AllTestSuite, 18 pages: 6 wrong or with exceptions, 6 ignored)

Overzicht Bouwpoqing (trend)

- #94 30-okt-2013 7:00:44
- #93 29-okt-2013 7:01:18
- #92 28-okt-2013 7:01:18
- #91 27-okt-2013 7:01:18
- #90 26-okt-2013 7:01:18
- #14 27-aug-2013 7:01:04

RSS alle RSS enkel gefaalde

Permanente referenties

- Laatste bouwpoqing (#94), sinds 3 uren 11 minuten
- Laatste stabiele bouwpoqing (#14), sinds 2 maanden 4 dagen
- Laatste succesvolle bouwpoqing (#90), sinds 4 dagen 4 uren
- Laatst gefaalde bouwpoqing (#94), sinds 3 uren 11 minuten
- Last unstable build (#90), sinds 4 dagen 4 uren
- Last unsuccessful build (#94), sinds 3 uren 11 minuten

Trend testresultaten

Build #	Count
#14	4
#90	4
#91	4
#92	4
#93	4
#94	4

(toon enkel fouten) ver groot

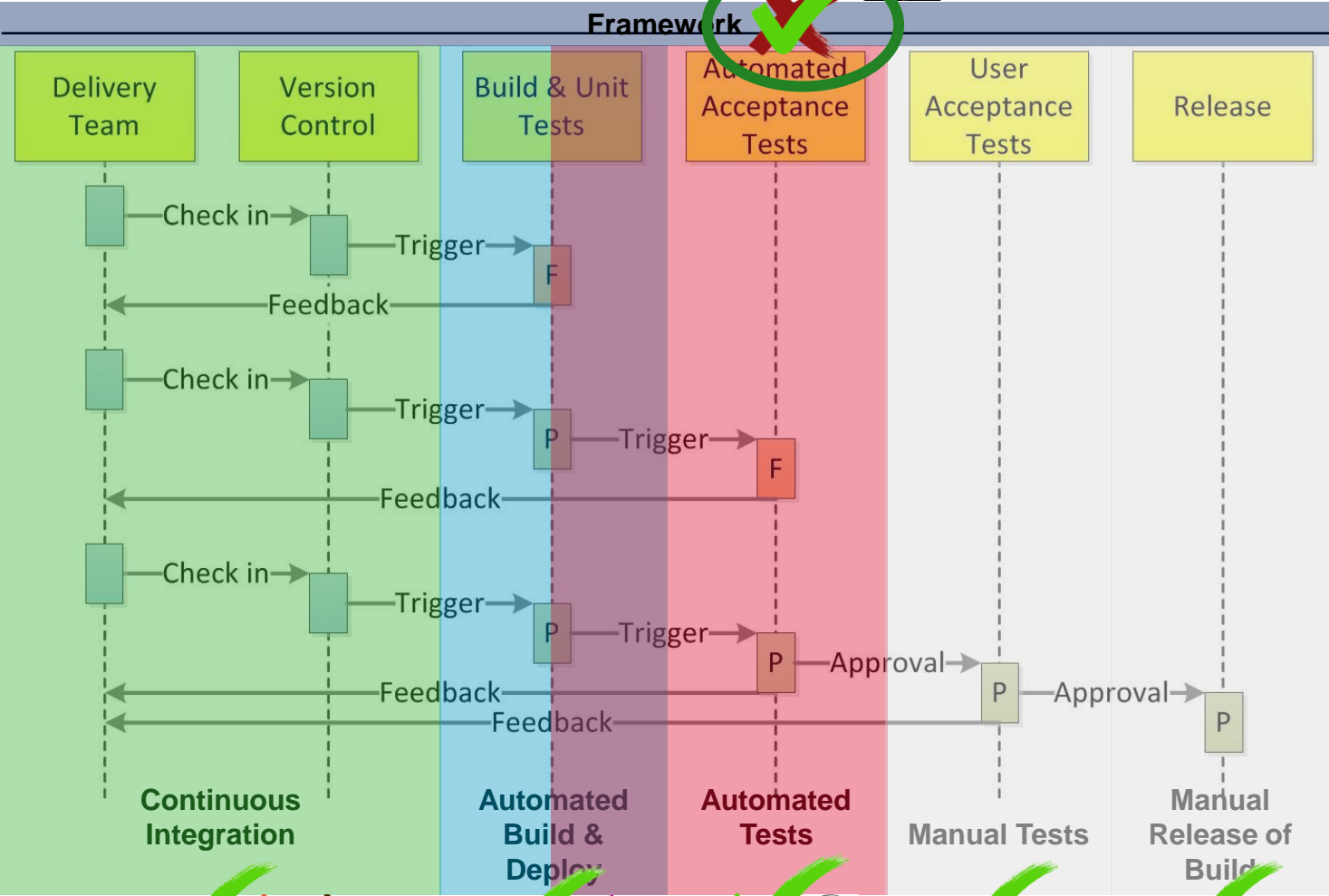
FitNesse Results Trend

Build #	Count
#14	0
#90	18

Pagina aangemaakt: 30-okt-2013 10:12:06 REST API Jenkins ver. 1.528

CONTINUOUS DELIVERY

STATUS IN EVERETT'S PROJECTS



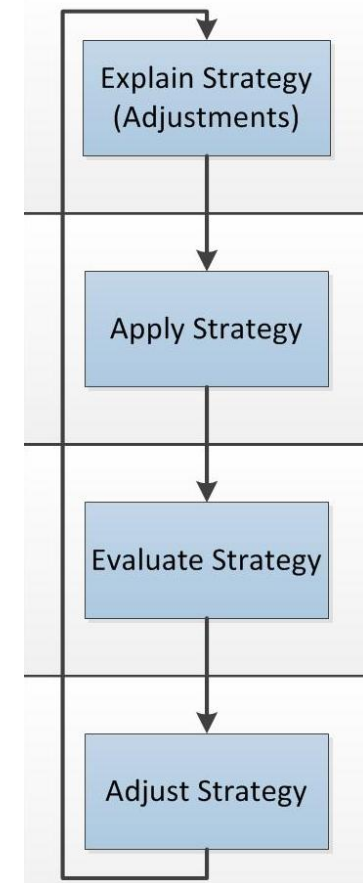
CASE STUDY



CASE STUDY

PROJECT & METHOD

- **Project:**
 - **At a client** of Everett
 - **6 team-members** (3 of Everett, 3 of client)
 - 3 days in a week, **sprints of 2 weeks**
- **Method**
 - Used **5 sprints (= 5x2 weeks)**
 - **Iterative**
 - **Evaluate:** Survey + Interviews



CASE STUDY

ITERATIONS

- **First iteration:**
 - Started with **basic strategy**
 - I created first test with team-member
 - Problem: **public API/Console**
- **Second Iteration**
 - Use **private API (created framework)**
 - I created **demo-tests** on demo environment
 - Chose & Install **Jenkins**

CASE STUDY

ITERATIONS

- **Third iteration:**
 - Team created **first tests** on project
 - Problem: running **wiki isn't updated** automatically
- **Fourth Iteration**
 - **Automatically** pull/push wiki **updates** to GIT
- **Fifth iteration:**
 - Held **Final Interviews**

RESULTS (1)

- **Amount of tests created**
 - **10** (by 4 team-members and me)
- **Benefits**
 - Short **learning curve**
 - **Regression** tests can now be automated
- **Costs**
 - **Invest time** to create framework, tailor the strategy, automate tests
 - Need **test criteria** early in the sprints
 - Need a **dedicated person** for the continuous delivery process.

RESULTS (2)

- **Compared with previous projects :**
 - Did not yet **spend less time on testing**
 - Due to needed time investments
 - Did not yet **found bugs earlier / found more bugs**
 - Due to few automated tests and still lot of manual tests
 - In a **new project**, the strategy is promising for these goals
- **Reuse strategy and tests in other projects**
 - **Strategy:** **yes, Except API-framework** when using other software
 - **Test cases:** **no, Except some parts** when using the same software

CONCLUSIONS



GOALS ACHIEVED?

✓ **Automate and improve process of software delivery**

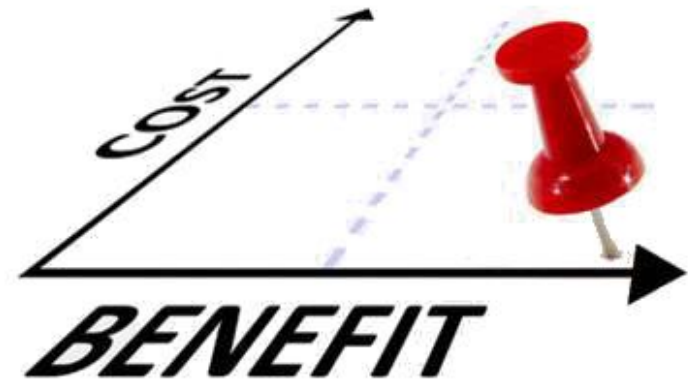
- By introducing Continuous Delivery

✗ **Reduce costs of Testing**

- By re-running/re-using tests

✓ **Enhance quality of product**

- By finding faults earlier
- By having better test-documentation



QUESTIONS



UNIVERSITY OF TWENTE.

everett.

TRUSTED TO KNOW | HOW



MOVING TOWARDS CONTINUOUS DELIVERY

INTRODUCING TEST AUTOMATION WITH FITNESS
IN SYSTEM INTEGRATION PROJECTS



SANDRA DRENTHEN

s.drenthen@student.utwente.nl /

s.drenthen@alumnus.utwente.nl