How we increased efficiency in unit test automation

Sanda Contiu / Jurian van de Laar

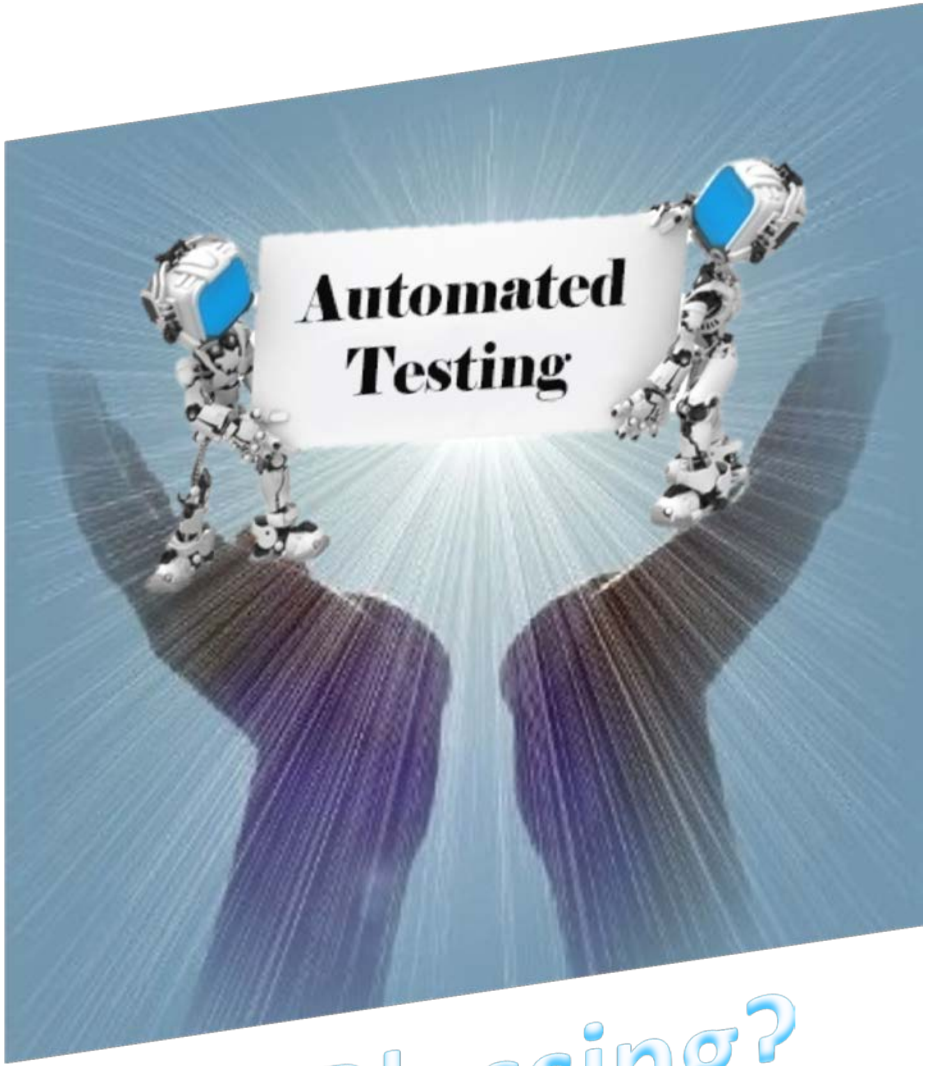
IGT Systems - Image Chain Cluster

September 21, 2015

Version 15

innovation you

PHILIPS

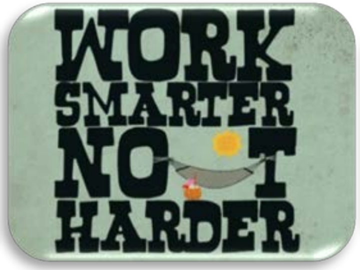


Blessing?

Headache?

Agenda

Context



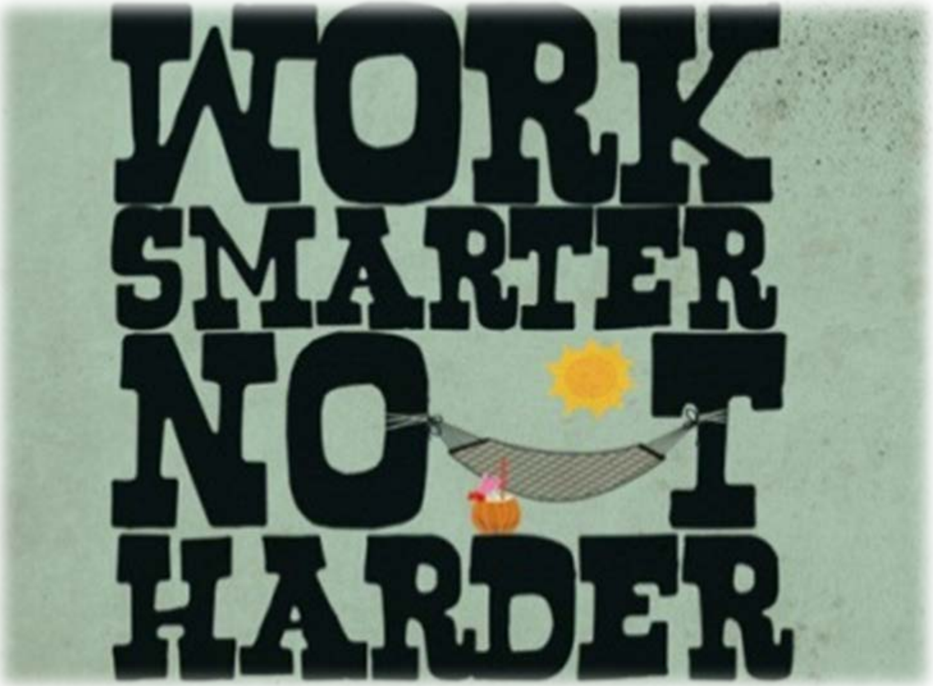
Challenges

Solution

Results

Vision

Agenda



Context

Challenges

Solution

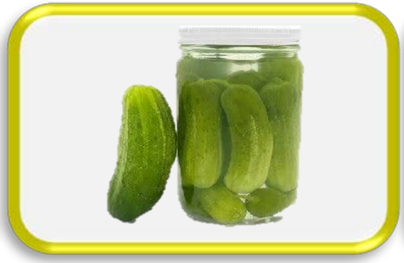
Results

Vision

Agenda

Context

Challenges

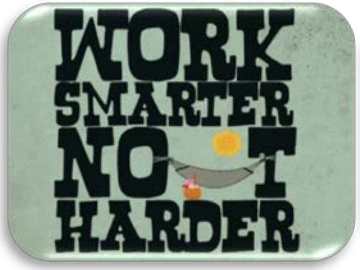


Solution

Results

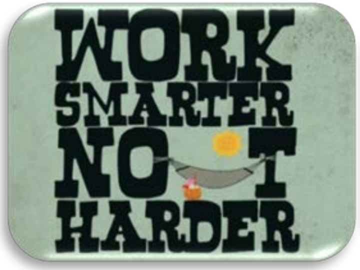
Vision

Agenda



Context Challenges Solution Results Vision

Agenda



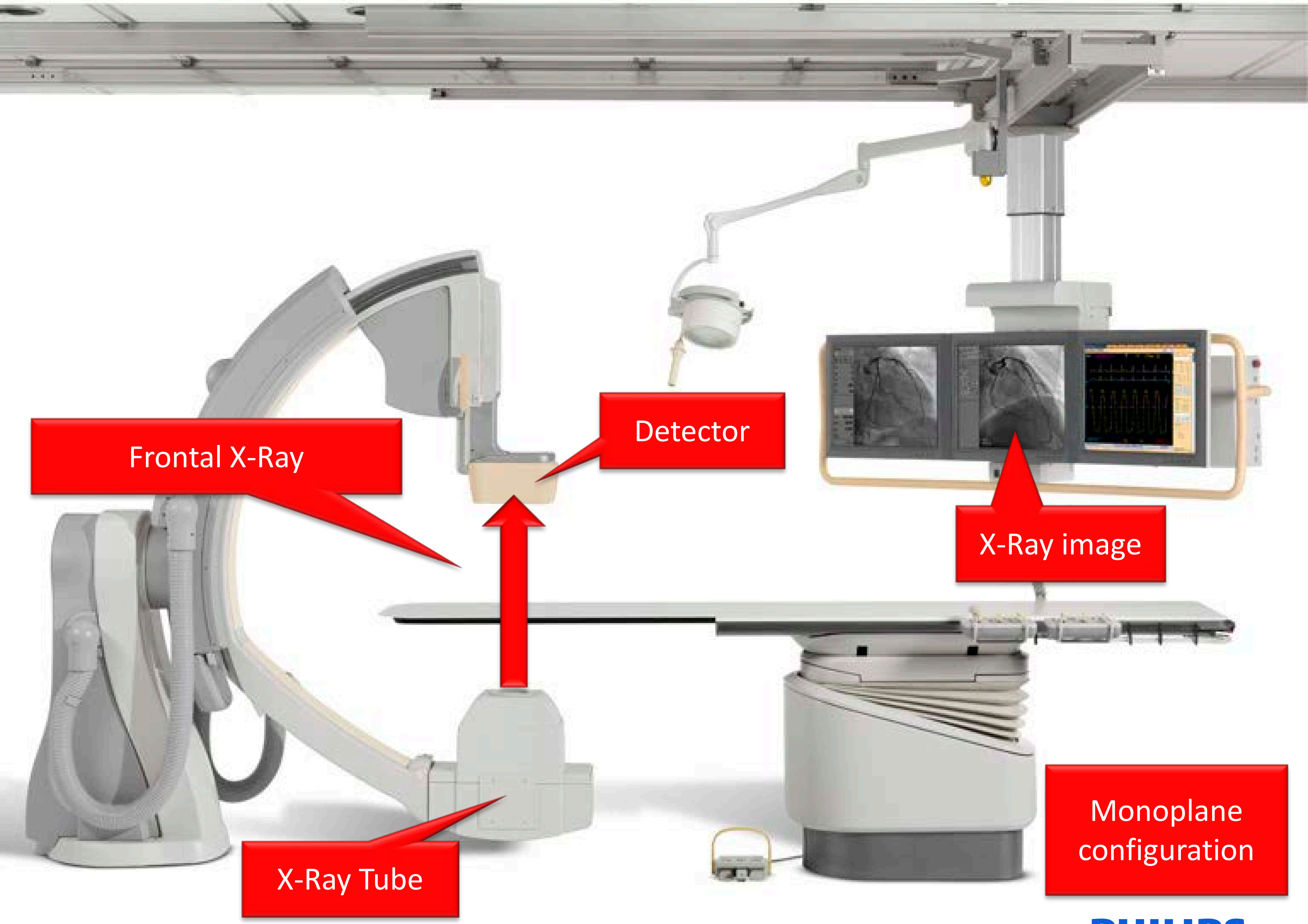
Context

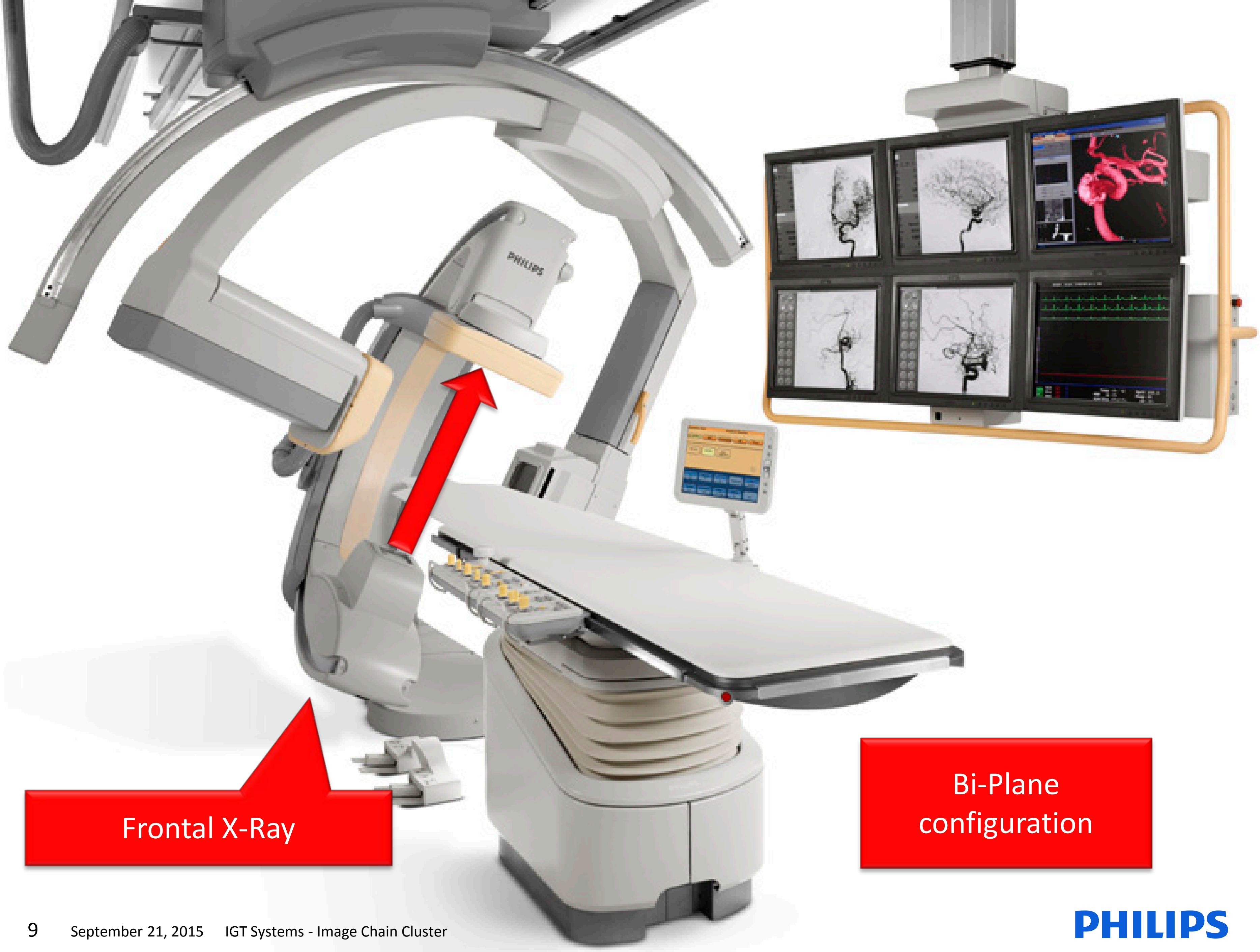
Challenges

Solution

Results

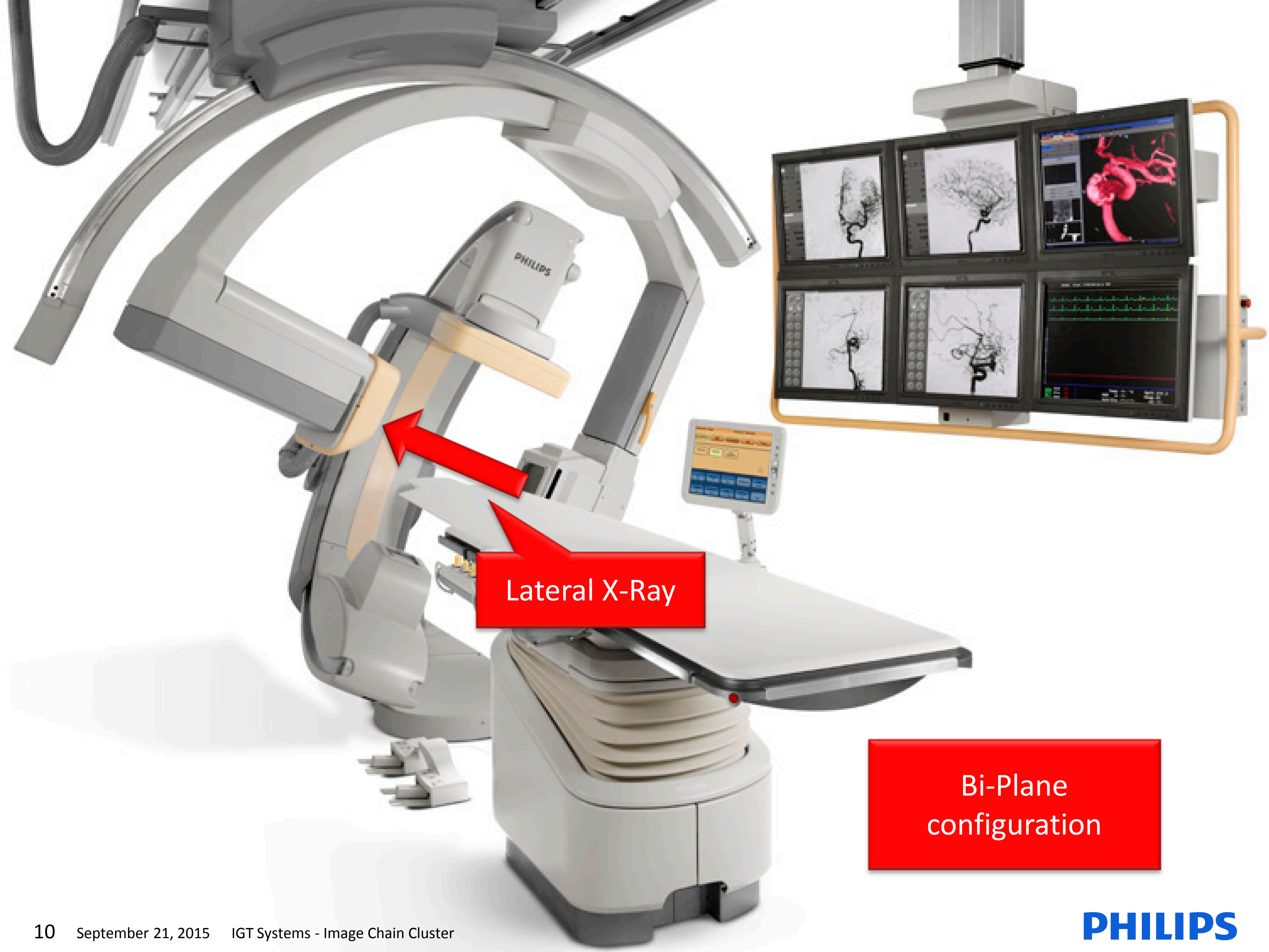
Vision


Frontal X-Ray
Detector
X-Ray image
X-Ray Tube
Monoplane configuration



Frontal X-Ray

Bi-Plane configuration



Lateral X-Ray

Bi-Plane configuration

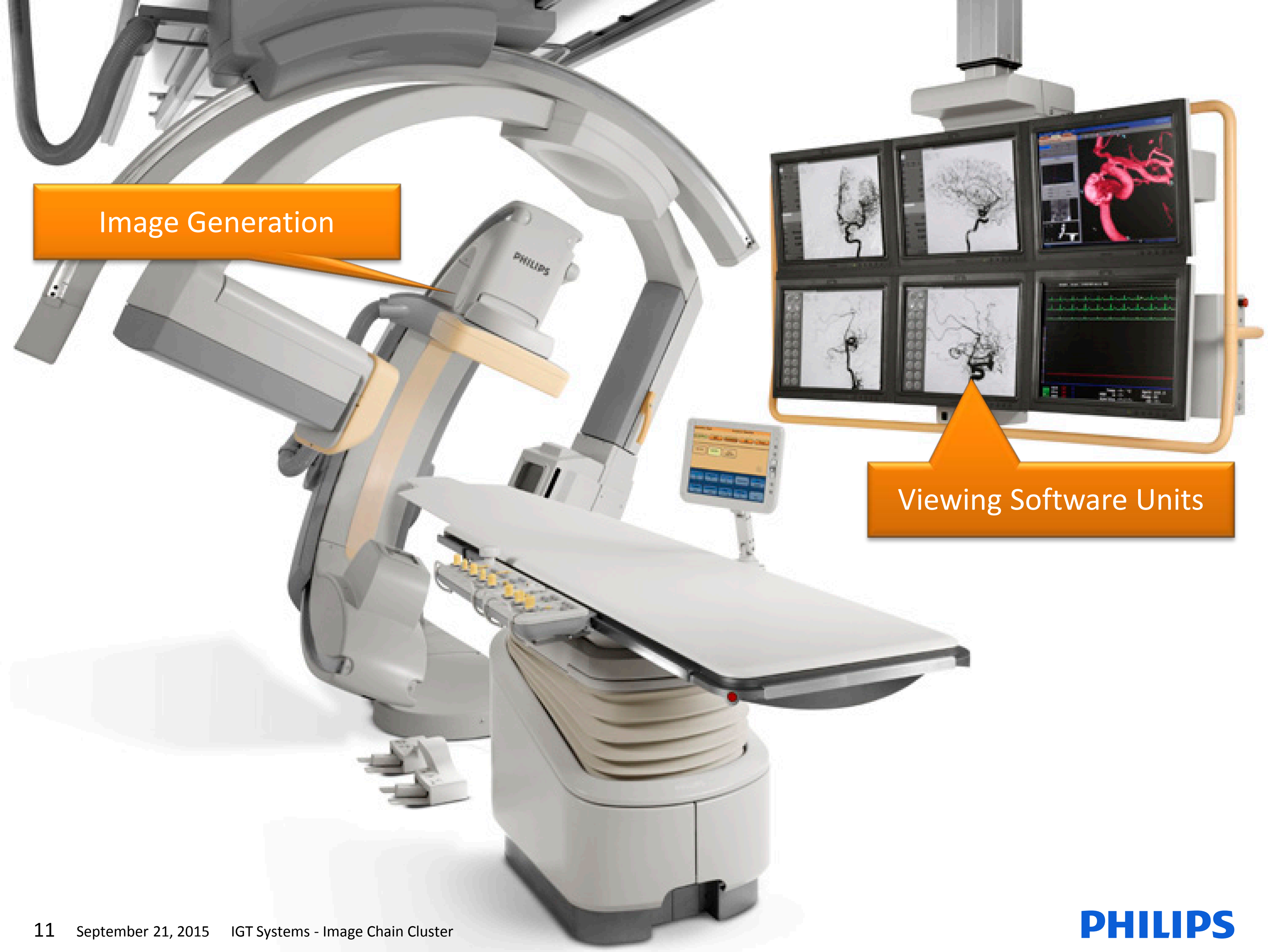
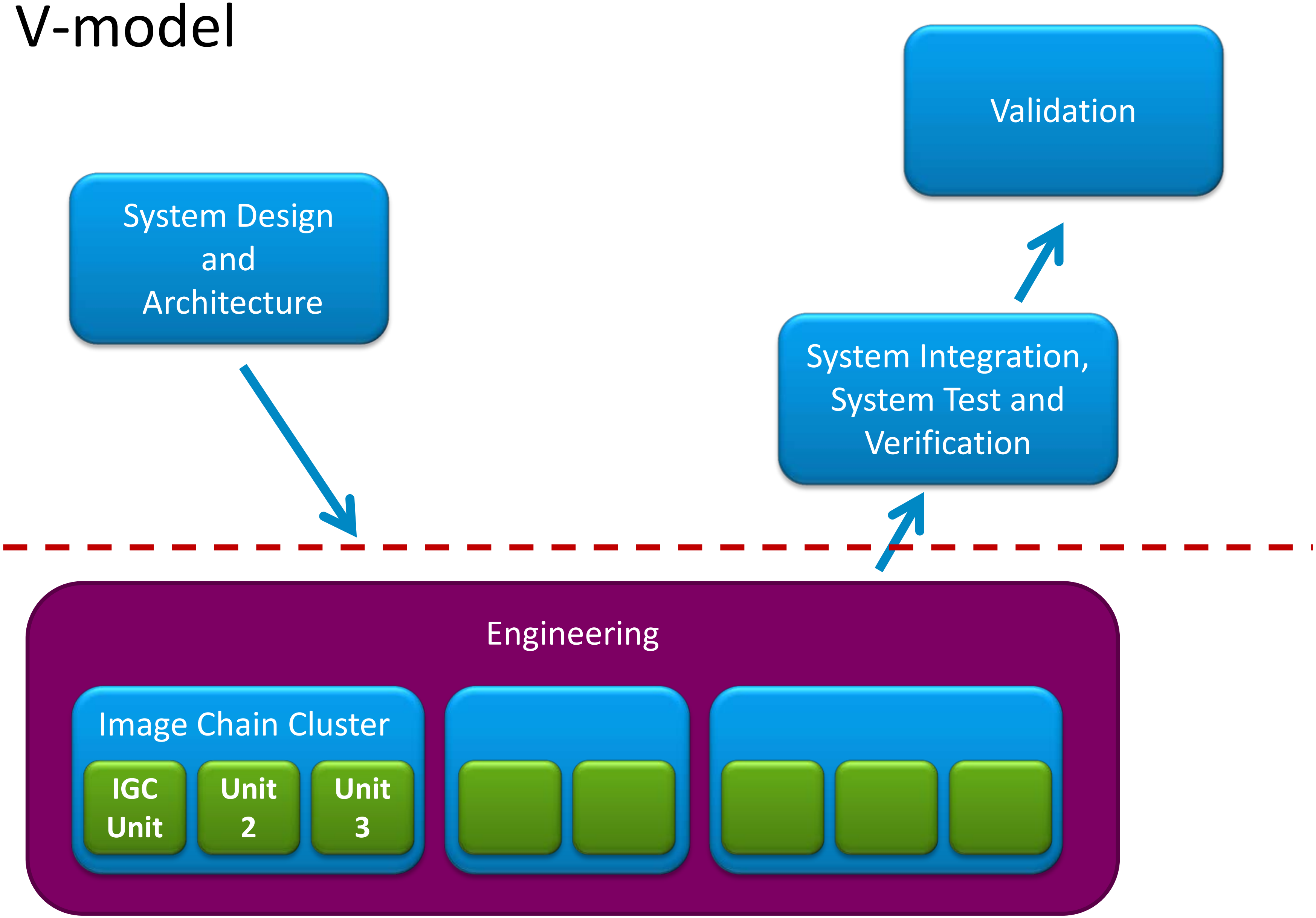


Image Generation

Viewing Software Units

V-model


Validation
System Design and Architecture
System Integration, System Test and Verification
Engineering
Image Chain Cluster
IGC Unit
Unit 2
Unit 3

Challenges

1. Resources

- Code size
- Maintenance
- Test documentation
- Maintaining in sync
- Engineers are scarce
- Need for speed
- Need for efficiency

Challenges

1. Resources

2. Quality

- Testing: a profession
- Difficult to review
- Developers biased
- Testers in driver’s seat

Challenges

1. Resources

2. Quality

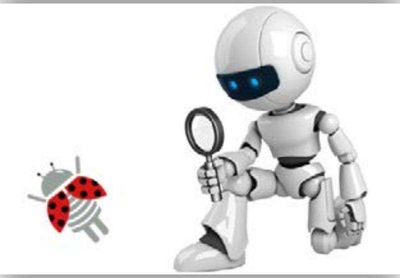
3. Automation

- Manual test execution
  - High degree
  - Time consuming
  - Error prone
  - Reproducibility

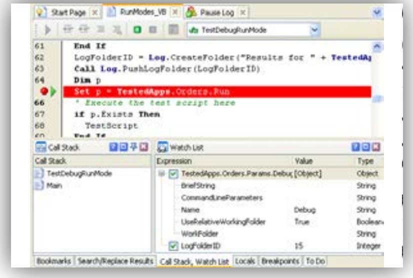
Challenges

1. Resources

2. Quality



3. Automation



```
End If
LogFolderID = Log.CreateFolder("Results for " + TestedA
Call Log.PushLogFolder(LogFolderID)
Dim p
Set p = TestedApps.Orders.Run
' Execute the test script here
if p.Exists Then
  TestScript
End If
```

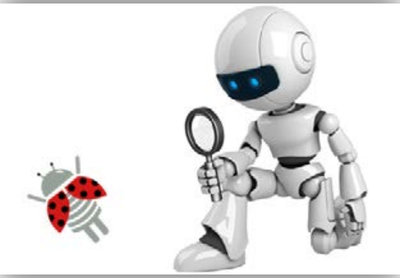
Call Stack
- TestDebugRunMode
- Main

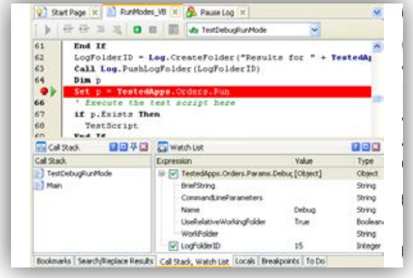
Expression	Value	Type
TestedApps.Orders.Params.Debu	[Object]	Object
BriefString		String
CommandLineParameters		String
Name	Debug	String
UseRelativeWorkingFolder	True	Boolea
WorkFolder		String
LogFolderID	15	Integer

4. Environment

- Need for integration
  - ***For testers:***
    - Readable script
    - Stubs
  - ***For developers:***
    - Debugging
    - Toggle

Challenges



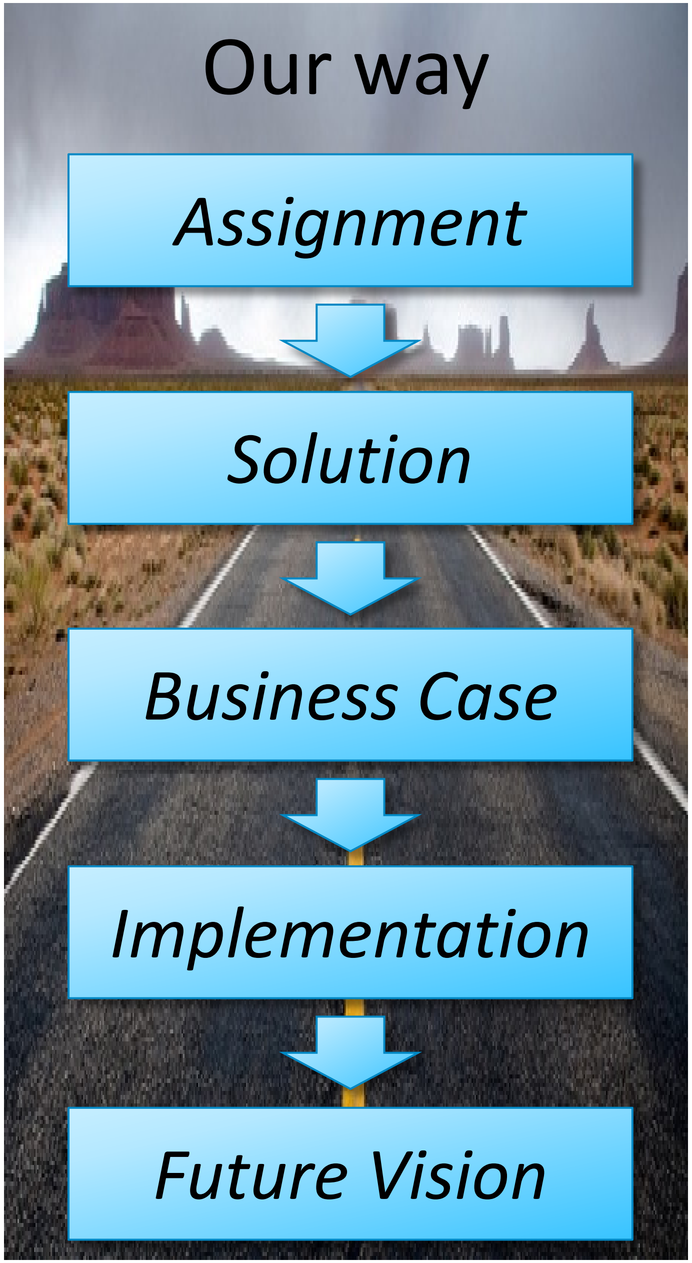


```
End If
LogFolderID = Log.CreateFolder("Results for " + TestedAp
Call Log.PushLogFolder(LogFolderID)
Dim p
Set p = TestedApps.Orders.Run
' Execute the test script here
if p.Exists Then
  TestScript
End If
```

Call Stack
TestDebugRunMode
Main

Expression	Value	Type
TestedApps.Orders.Params.Debuc	[Object]	Object
BriefString		String
CommandLineParameters		String
Name	Debug	String
UseRelativeWorkingFolder	True	Boolean
WorkFolder		String
LogFolderID	15	Integer

Our way



Assignment

Solution

Business Case

Implementation

Future Vision

Goal

Testers write test specification

- Different mental model
- Testing professionalism
- Save time for developers

Higher **efficiency**

- Reduce maintenance
- Higher degree of test automation

Assignment

specify **readable unit**-level **tests**

at a **higher abstraction** level
than current keyword-based scripts

to **minimize synchronization**
test specification versus test implementation

How?

- Identify **needs** testers, developers
- **Own** DSL for testing vs. **existing** tools
- Choose best **option** for **pilot**,
then implement **solution**

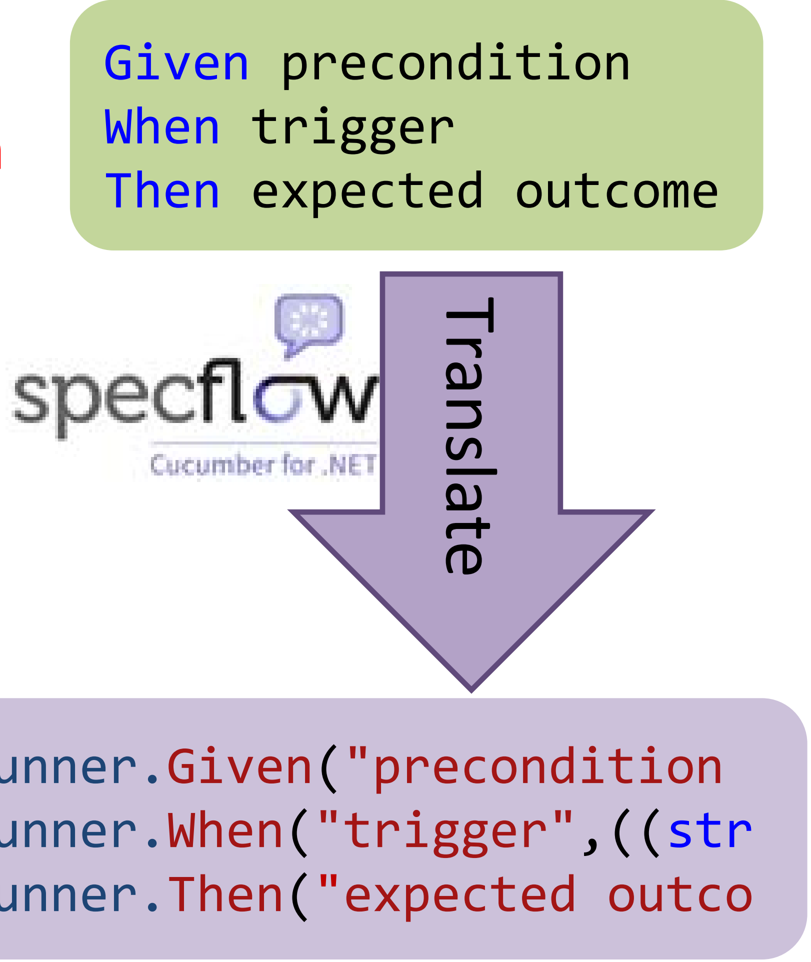
Solution: Gherkin + SpecFlow

Gherkin :

- extendable **DSL**
- natural language **test specification**
- behavior specification **by example**

SpecFlow:

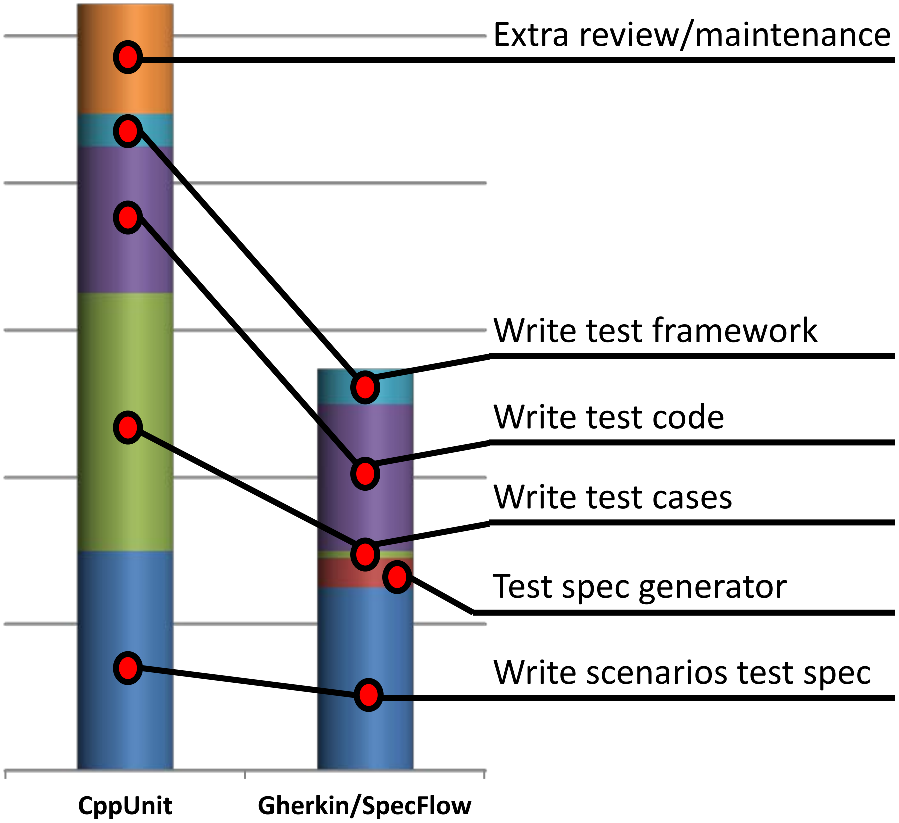
- Cucumber-family **tool**
- **write** test specs in Gherkin
- **generate** test code
- **integrated** in IDE
- NOT a test framework

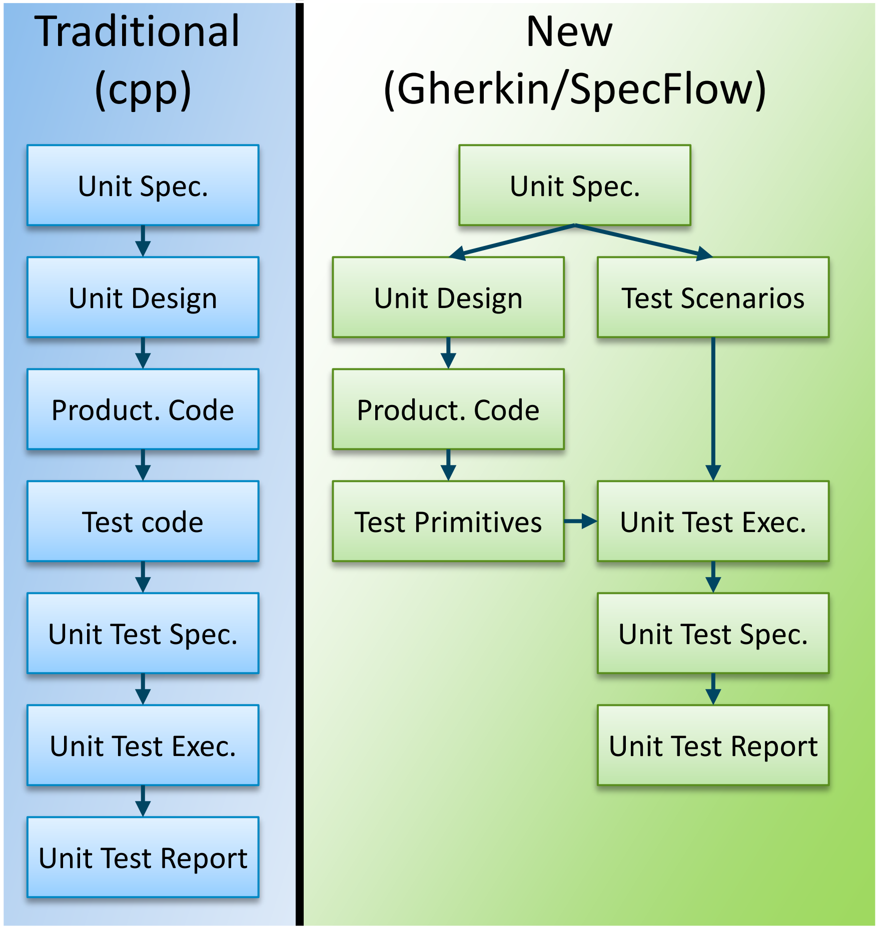


```
Given precondition
When trigger
Then expected outcome
```

```
testRunner.Given("precondition
testRunner.When("trigger",((str
testRunner.Then("expected outco
```

Business case





Traditional (cpp)

Unit Spec. → Unit Design → Product. Code → Test code → Unit Test Spec. → Unit Test Exec. → Unit Test Report

New (Gherkin/SpecFlow)

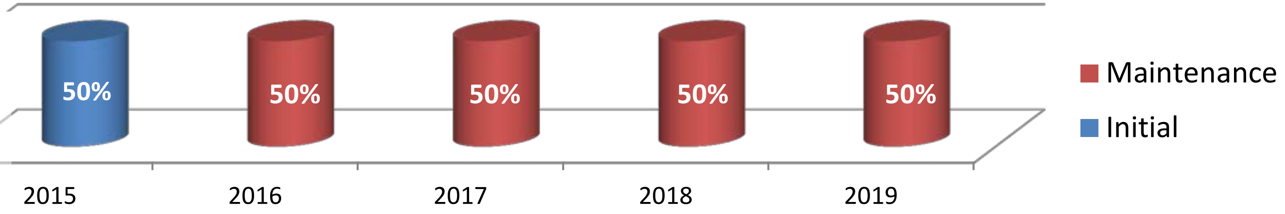
Unit Spec. → Unit Design, Test Scenarios

Unit Design → Product. Code → Test Primitives → Unit Test Exec.

Test Scenarios → Unit Test Exec. → Unit Test Spec. → Unit Test Report

Yearly savings maintenance costs:

CppUnit: 25% of code size
Gherkin: 12.5% for test primitives



Pilot

Goal: technical feasibility

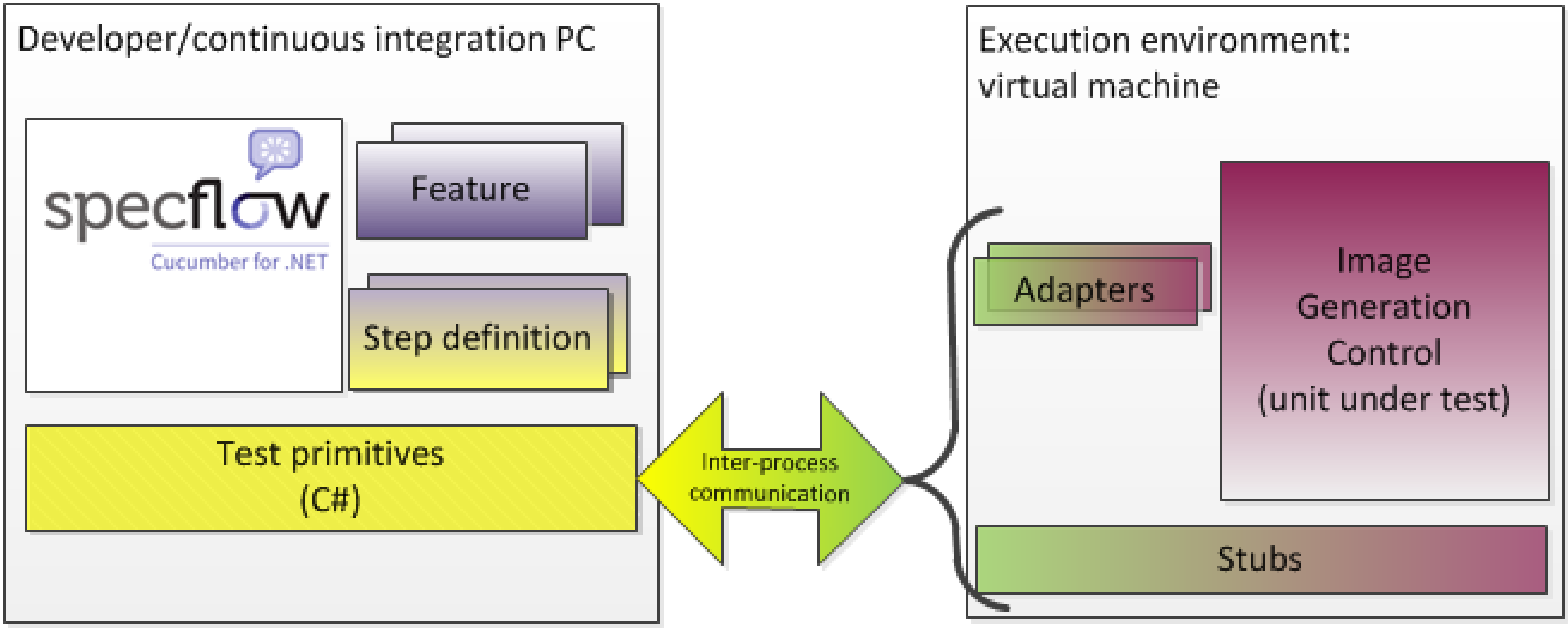
Time: 1 month

Content: implement test cases for 1 feature

Outcome:

- smooth **tooling integration**
- test **infrastructure**
- executable test spec **completely written** by **tester**
- **running tests** in virtual environment
- **answers** to technical questions
- decision: **use** this **approach for all** unit tests

Test setup 1: virtual machine (Developer PC)



Developer/continuous integration PC

specflow

Cucumber for .NET

Feature

Step definition

Test primitives
(C#)

Inter-process
communication

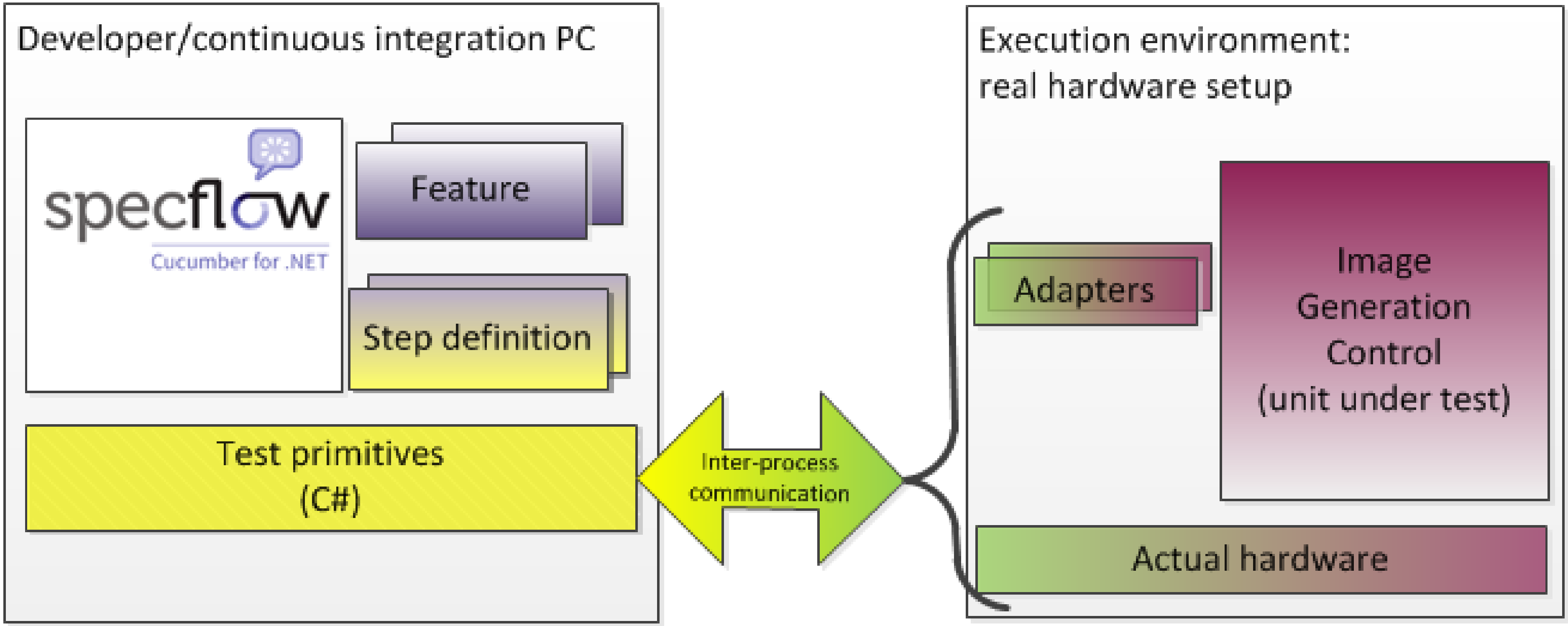
Execution environment:
virtual machine

Adapters

Image
Generation
Control
(unit under test)

Stubs

Test setup 2: Actual target hardware



Developer/continuous integration PC

specflow

Cucumber for .NET

Feature

Step definition

Test primitives (C#)

Inter-process communication

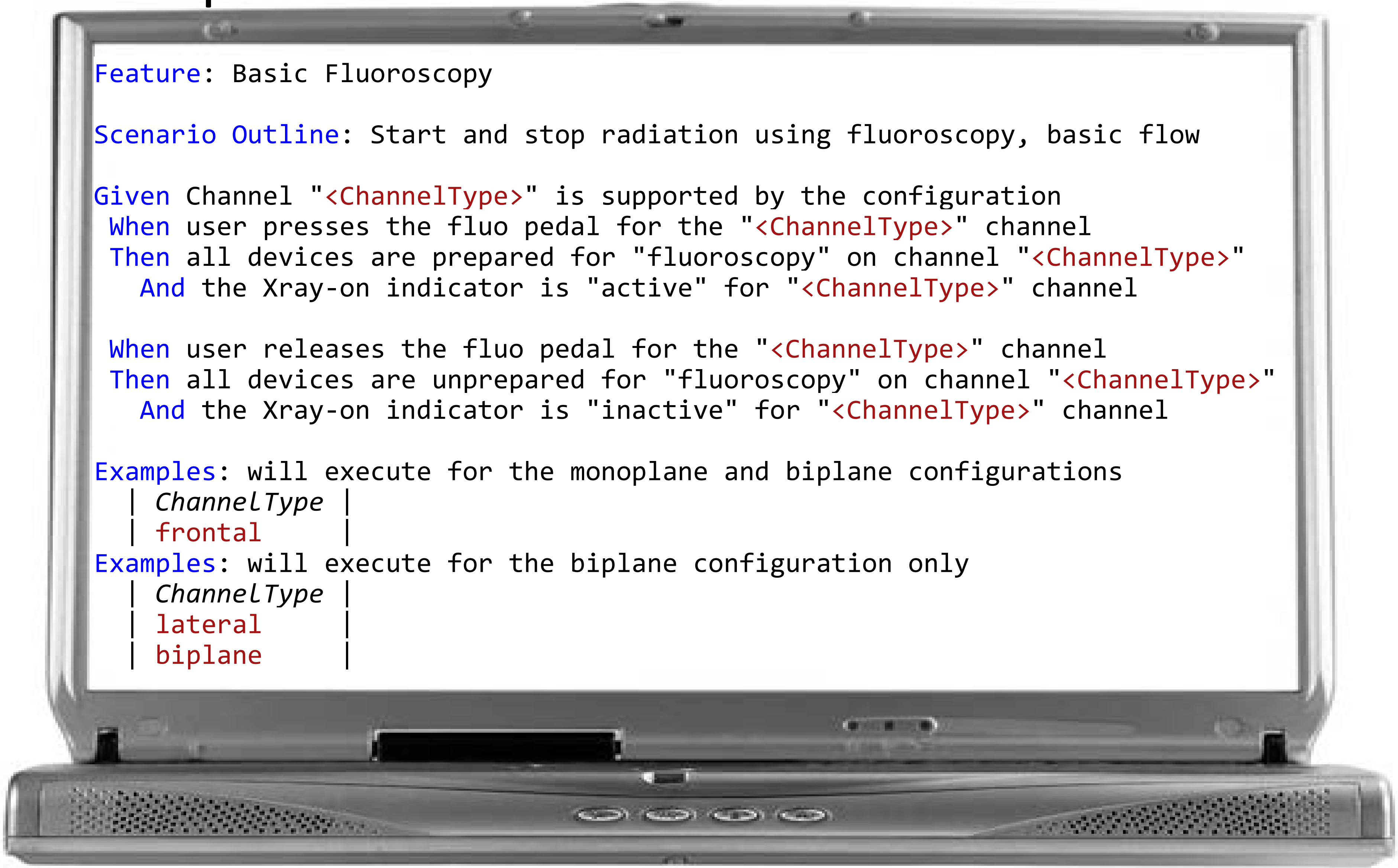
Execution environment: real hardware setup

Adapters

Image Generation Control (unit under test)

Actual hardware

Example: Feature File



```
Feature: Basic Fluoroscopy

Scenario Outline: Start and stop radiation using fluoroscopy, basic flow

Given Channel "<ChannelType>" is supported by the configuration
 When user presses the fluo pedal for the "<ChannelType>" channel
 Then all devices are prepared for "fluoroscopy" on channel "<ChannelType>"
   And the Xray-on indicator is "active" for "<ChannelType>" channel

 When user releases the fluo pedal for the "<ChannelType>" channel
 Then all devices are unprepared for "fluoroscopy" on channel "<ChannelType>"
   And the Xray-on indicator is "inactive" for "<ChannelType>" channel

Examples: will execute for the monoplane and biplane configurations
  | ChannelType |
  | frontal     |
Examples: will execute for the biplane configuration only
  | ChannelType |
  | lateral     |
  | biplane     |
```

Results

"During SW development of a new component as part of a large legacy code base, I discovered that the Cucumber way of testing enabled us to create a very accessible and clean test environment for this component with self-explaining test cases that can easily be understood and maintained by all team members and even beyond"

John Mulder

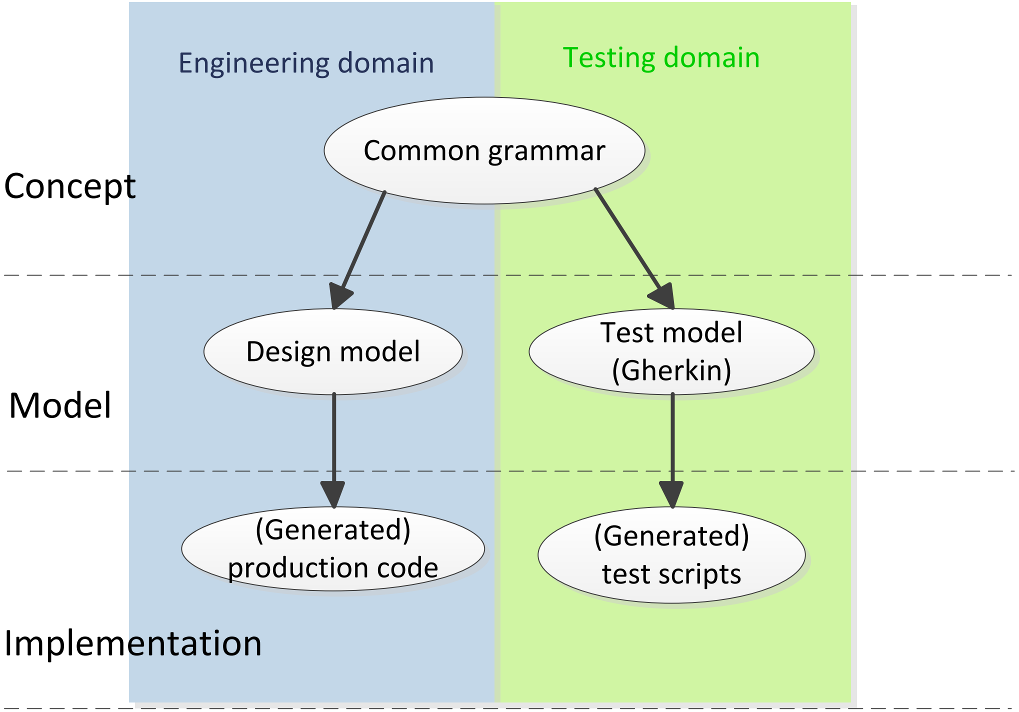
"Cucumber makes test specs writable by testers, test reports readable by managers and the test process in between is automated. R&D can focus on R&D. It makes life easier for all of us."

Vic Henderikx

- Reality vs. initial ideas
- Benefits
  - **Cooperation**
  - **Automation** level
  - **One** test spec for **multiple** configurations/exec env.
  - **Reduced** maintenance effort

Future vision

- **Requirements validation** earlier (in study)
- **Test case generation** from model (in study)
- **Integrate** Model Driven SE + Model Based Testing



Conclusions

Efficiency increased

- Lower initial and maintenance costs
- Testers and developers work in parallel
- Faster test scenarios (natural language)
- Specification by example (using tables)
- Generation of Unit Test Spec (keeping in sync)

Testability increased

- Tester in driver seat (reducing developer bias)
- Also embraced by developers (‘TDD’)
- Reproducibility (using SpecFlow)
- Stubs and debugging facilities
- Running on development & target system

Questions & Answers

PHILIPS

PHILIPS