



TNO **ESI**

Powered by industry
and academia

TAKE THE NEXT STEP IN TEST AUTOMATION WITH MBT

Debjyoti Bera | TNO-ESI | Email: debjyoti.bera@tno.nl

AGENDA

- Testing context and challenges in industry
 - Best practices in industry
 - Limitations of BDD test automation frameworks
 - **Break + Reflections and Discussions**
 - Take the next step in test automation with MBT
 - What do we mean by models and what is MBT
 - Collaborative specification of systems in context with BPMN4S
 - Automated Test Generation and Coverage
 - Live demo of a print shop
 - How did we evaluate the methodology and show value, insights into adoption at ASML
 - **Break + Reflections and Discussions**
 - Research and Development Roadmap
 - Can AI potentially help to improve systems testing?
 - **Takeaways and Wrap-up**
-

AGENDA

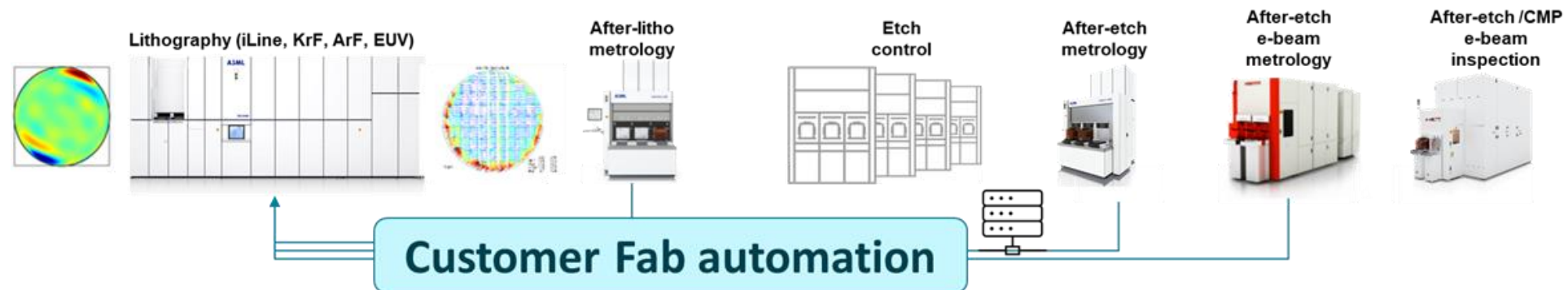
- Testing context and challenges in industry
 - Best practices in industry
 - Limitations of BDD test automation frameworks
 - **Break + Reflections and Discussions**
 - Take the next step in test automation with MBT
 - What do we mean by models and what is MBT
 - Collaborative specification of systems in context with BPMN4S
 - Automated Test Generation and Coverage
 - Live demo of a print shop
 - How did we evaluate the methodology and show value, insights into adoption at ASML
 - Break + Reflections and Discussions
 - Research and Development Roadmap
 - Can AI potentially help to improve systems testing?
 - Takeaways and Wrap-up
-

BACKGROUND OF THIS TALK

From 2023 to 2025, TNO-ESI and ASML *collaborated* in a **public-private partnership** to develop a **MBT** methodology to **improve efficiency and effectiveness of system(-of-systems) testing processes** in industry.

Open-source tooling offering a **collaborative modeling framework**, with capabilities for **simulation** and **automatic generation of test scripts** in various languages such as Cucumber, Robot, JUnit, and GTest.

Successful validation through several pilots, the methodology and tools are being **matured for adoption** into a new way of working.



RISING COMPLEXITY OF CYBER-PHYSICAL SYSTEMS

Increasing role and complexity of software in cyber-physical systems

Ensuring promised system quality is a huge challenge

- Multi-disciplinary with intrinsic rising complexity
- Uncertainty in operational context (environment)
- Heterogeneity in systems, SW platforms and standards
- Product-line variability and long-lifetimes,
- Impact of updates and upgrades
- New trends such as AI, cloud deployments, IoT etc.

Testing is a key activity to assess and improve system quality but not much has changed in how we do it in the last 30 years...



Products

Azurion 3F12
Azurion 3F15
Azurion 5C12
Azurion 5C20
Azurion 5F20
Azurion 7F12
Azurion 7C12
Azurion 7C20
Azurion 7B 12-12
Azurion 7B 20-12
Azurion 7B 20-15



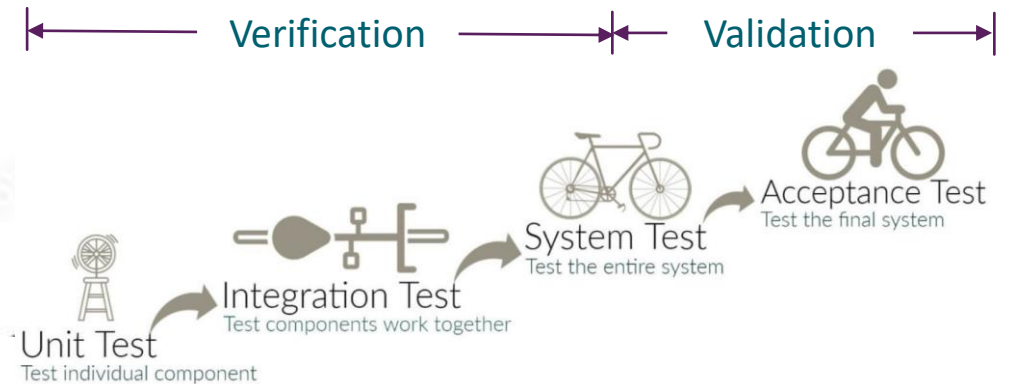
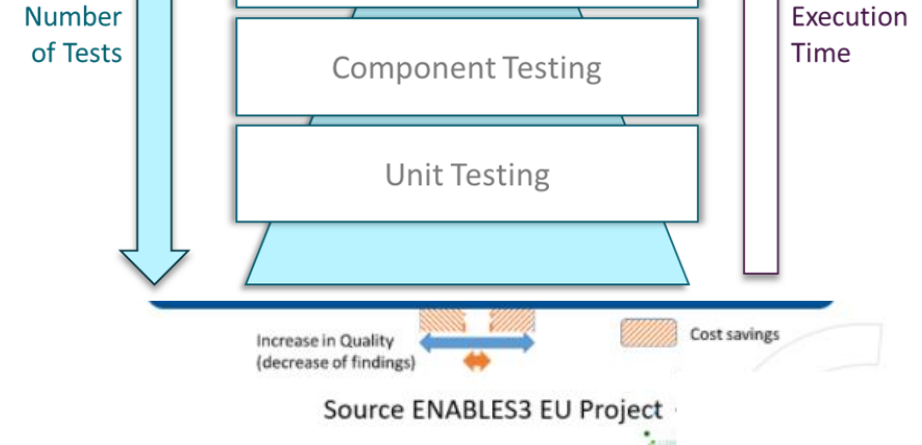
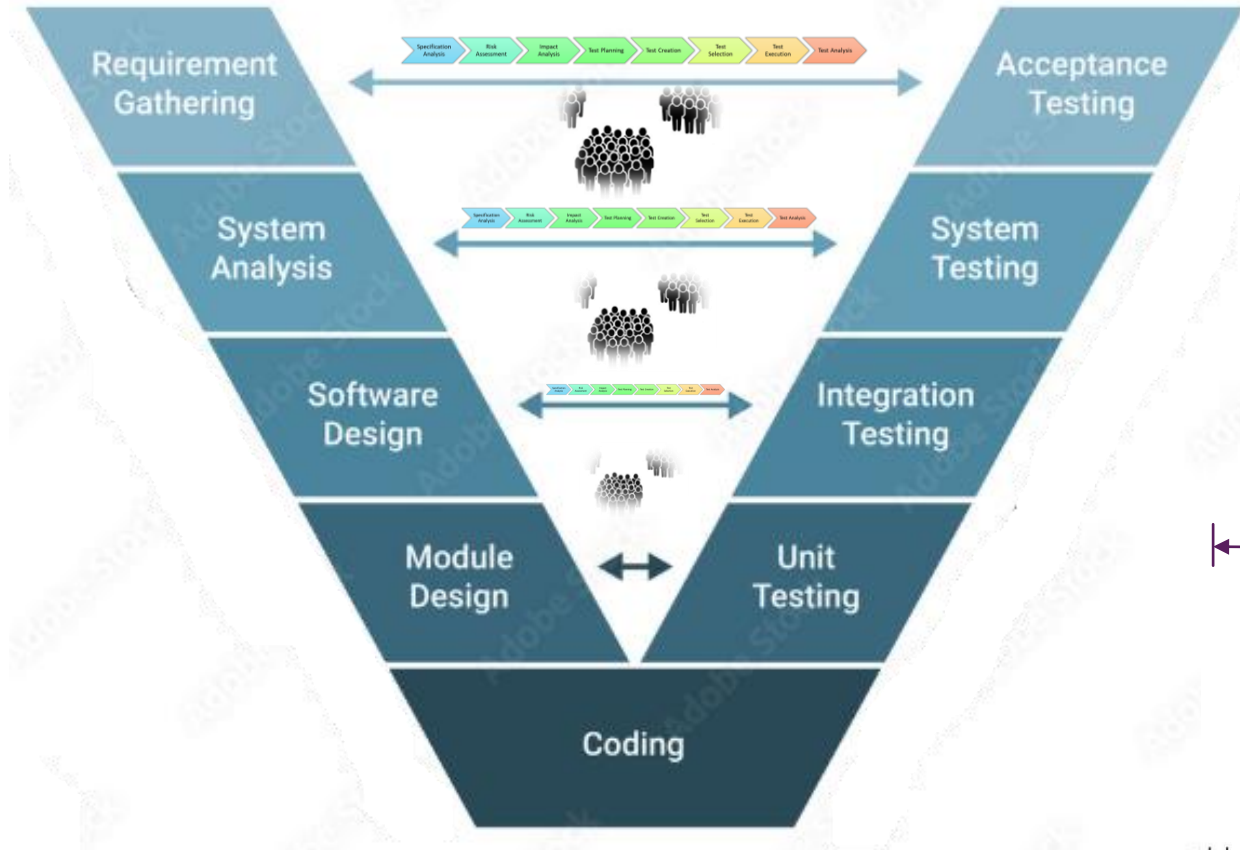
SW Versions

R1.0.0, R1.1.0, SP 1.0... R14.0.0

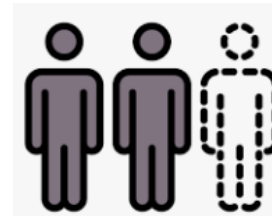


SYSTEMS TESTING

The cost of detecting software defects grows exponentially



TESTING CHALLENGES



Lack of Trust



How to enable faster time to market with promised quality and lower costs?

Automation of Manual Testing	Requirements Gathering and Stakeholder Management	Costs of Test Infrastructure	Lack of Available Hardware
Handcrafting of Test Suites	Traceability of Specifications and Tests	Maintenance of Test Suites	Test Execution Analysis
Dealing with Updates and Upgrades	Selection of Tests and Configurations	Coverage and Quality Metrics	Risk Management and Mitigation
Scarcity of People			

BEST PRACTICES IN INDUSTRY

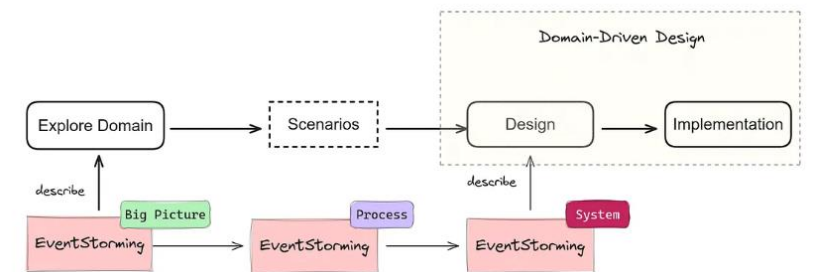
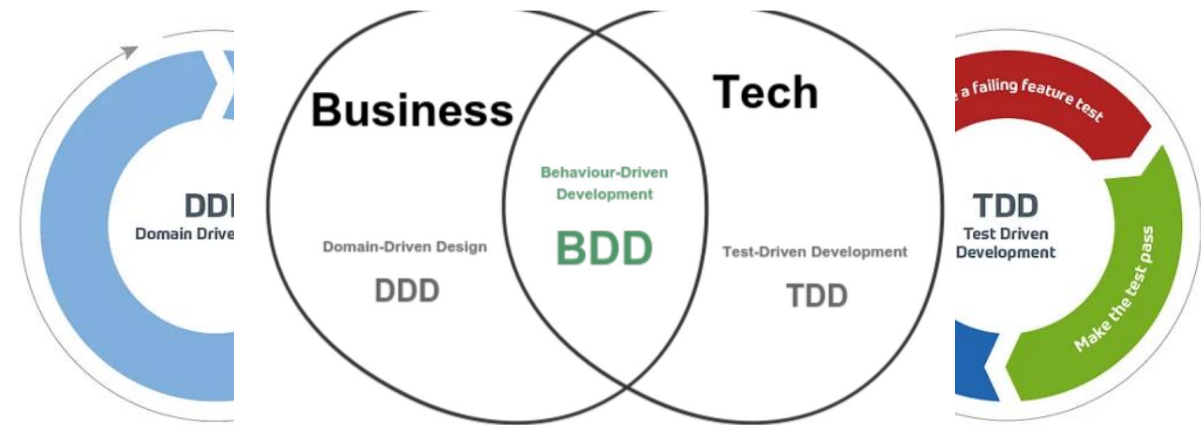
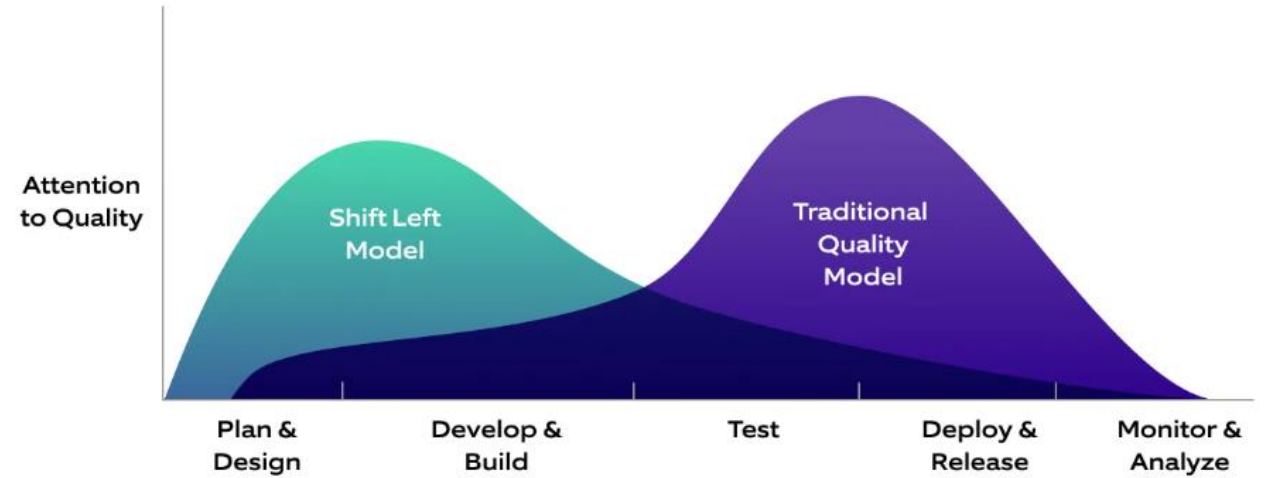
Shift-left strategy

DDD: Create shared understanding and language between business and technical concepts (ubiquitous language in context)

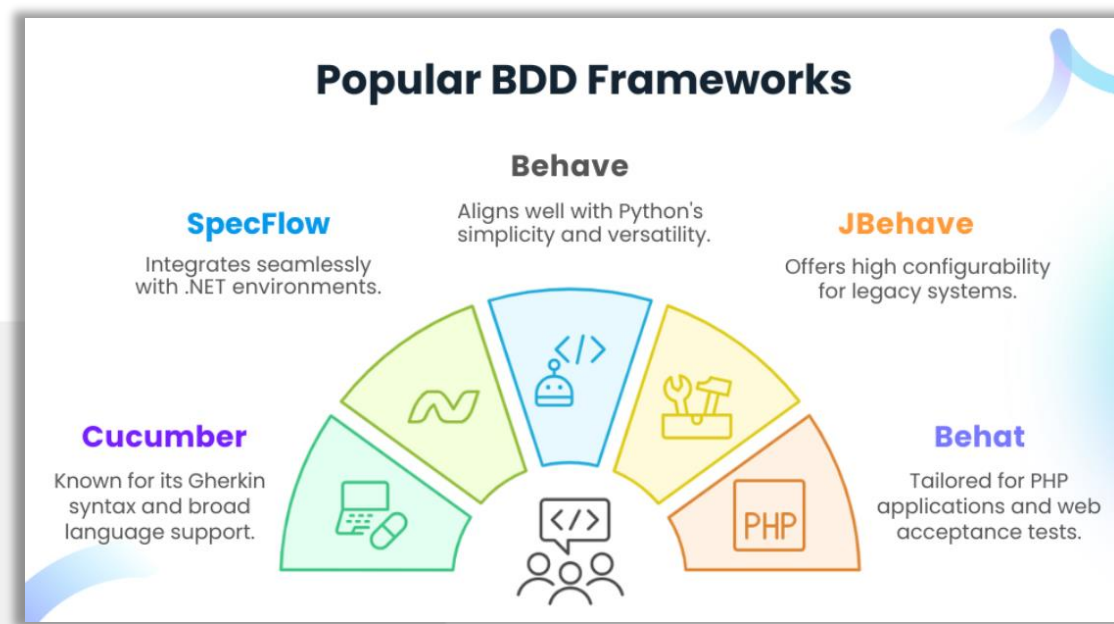
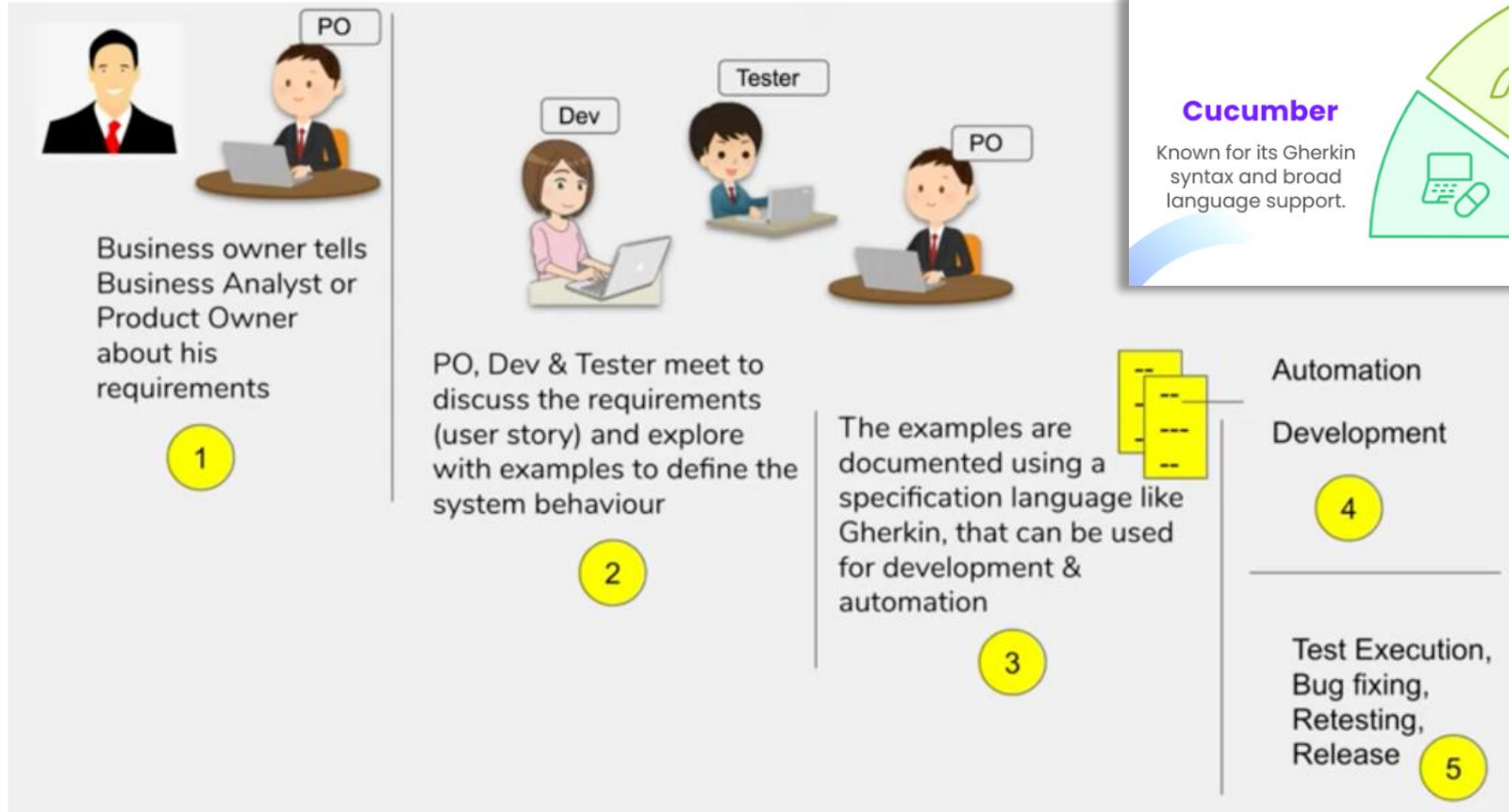
BDD: Focus on collaborative specification of *expected* system behavior as scenarios

TDD: Failing tests before systems development

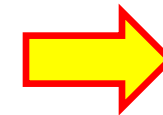
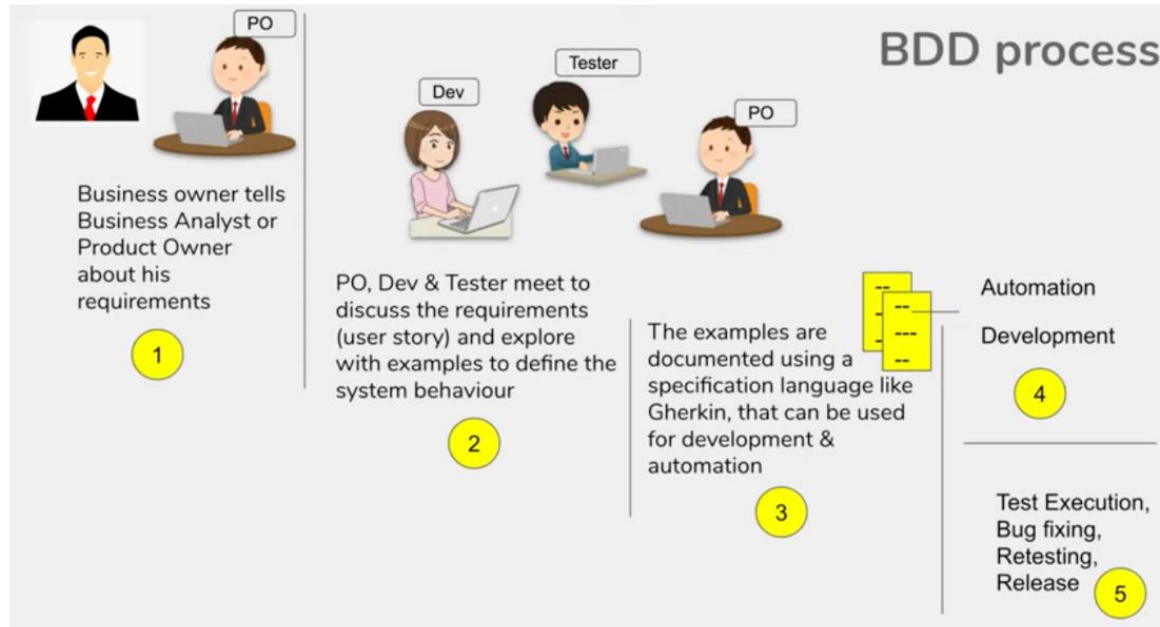
DDD + BDD + TDD (combinations of them) are widely used methodologies in industry



BEHAVIOR-DRIVEN DEVELOPMENT



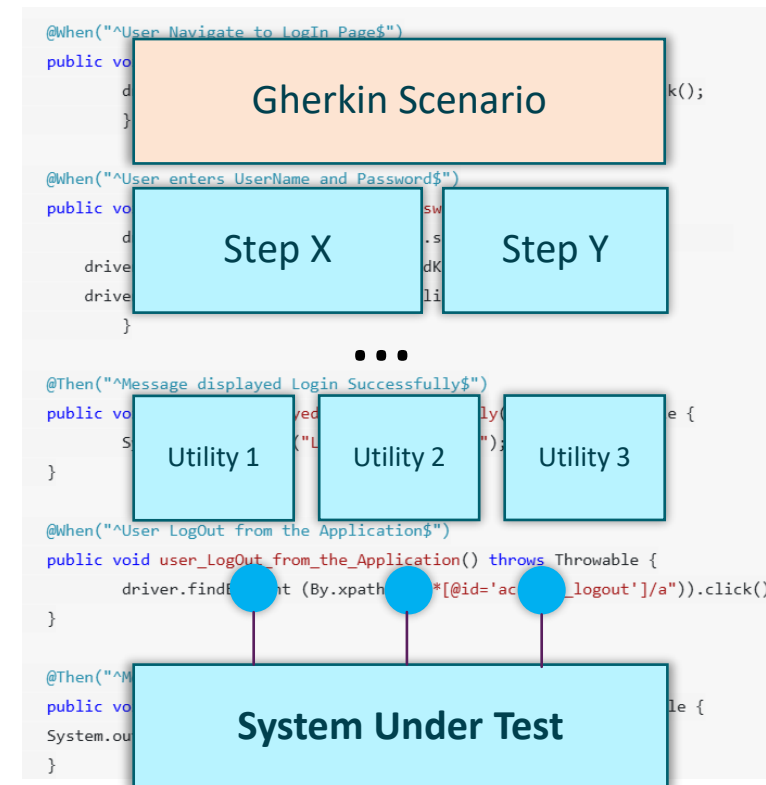
COLLABORATIVE SPECIFICATION WITH GHERKIN



```

Feature: Login
  As a user
  I want to be able to log in to the application
  So that I can access my account information

Scenario: Successful login
  Given I am on the login page
  When I enter my username and password
  And I click the login button
  Then I should be taken to the dashboard page
    
```

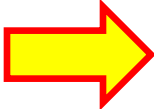
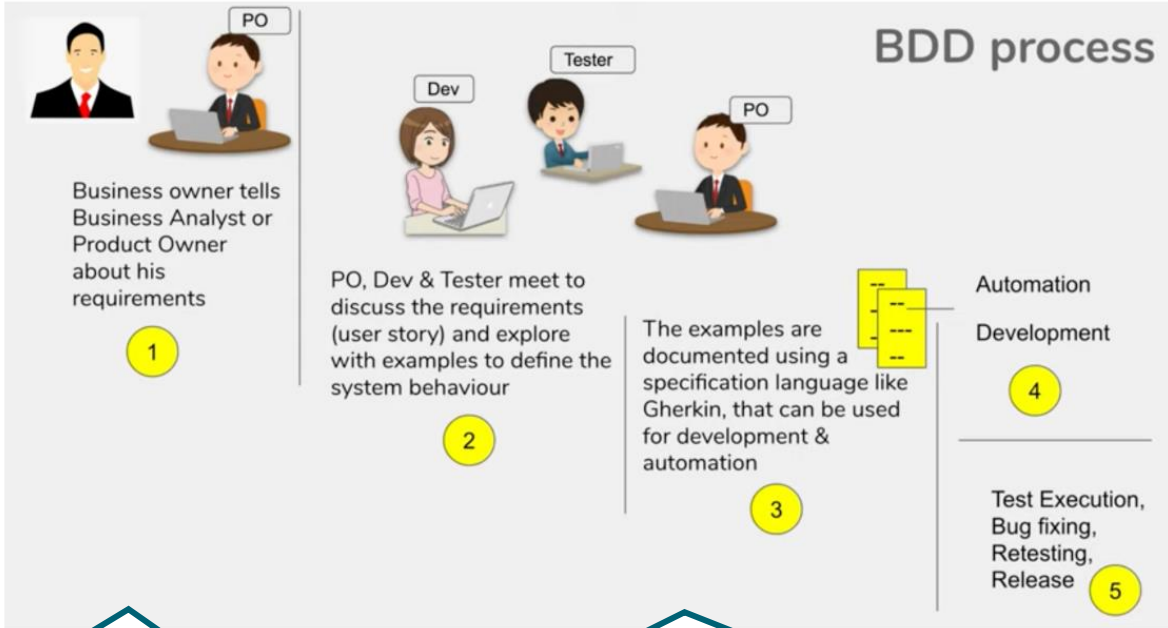


BDD CHALLENGES

What can we really say about coverage of specification and risks?
What does a keyword really do?

```
Feature: Login
As a user
I want to be able to log in to the application
So that I can access my account information

Scenario: Successful login
Given I am on the login page
When I enter my username and password
And I click the login button
Then I should be taken to the dashboard page
```



```
@When("User Navigate to LogIn Page$")
public void user_Navigate_to_LogIn_Page() throws Throwable {
    driver.findElement(By.xpath(".*[@id='account']/a")).click();
}

@When("User enters UserName and Password$")
public void user_enters_UserName_and_Password() throws Throwable {
    driver.findElement(By.id("log")).sendKeys("testuser_1");
    driver.findElement(By.id("pwd")).sendKeys("Test@123");
    driver.findElement(By.id("login")).click();
}

@Then("Message displayed Login Successfully$")
public void message_displayed_Login_Successfully() throws Throwable {
    System.out.println("Login Successfully");
}

@When("User LogOut from the Application$")
public void user_LogOut_from_the_Application() throws Throwable {
    driver.findElement (By.xpath(".*[@id='account_logout']/a")).click();
}

@Then("Message displayed Logout Successfully$")
public void message_displayed_Logout_Successfully() throws Throwable {
    System.out.println("Logout Successfully");
}
```

High effort, cost and lead time

Lack support for dealing with changes (updates/upgrades)

High maintenance effort

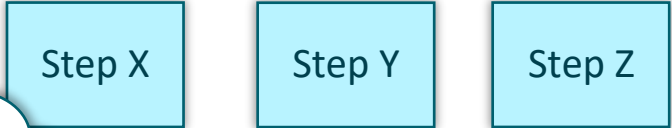
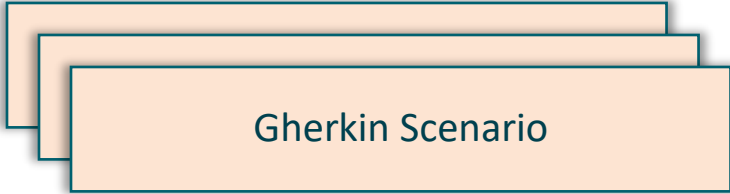
How to answer these questions: What is the coverage of my test suite? How much of each risk do each of my test cover? What does it mean to cover? etc...

LIMITATIONS OF GHERKIN

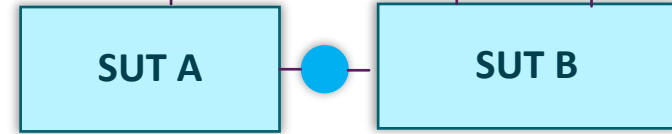
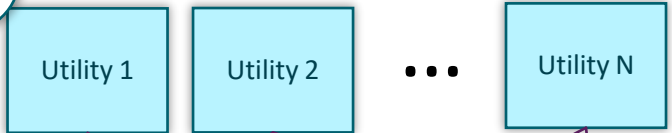
Limited Language Expressivity

- Limited data types (int, real, bool and string)
- Deterministic, i.e. no branching, e.g., if-else, loops, etc.
- No support for configuration variability

Simple Data Types



Simple Data Types Mapped to Complex Types



```
@Coin @Coffeebeans
Scenario Outline: The user orders Coffee and Pays by Coins for it successfully the Coffee is delivered
    Given the Vending Machine is initialized with mode "<Mode>"
    When the user selects startUp
    Then the vending machine is Ready
    And the vending machine presents the product menu
    When the user selects Coffee
    And the user selects strength
    Then the vending machine presents payment screen
    When the user pays "<C>" euro by coins
    Then the vending machine delivers Coffee
    And the vending machine displays thank you screen
    And the vending machine is Ready
```

```
@Coin @Coffeebeans
Scenario Outline: The user orders Coffee and Pays for it unsuccessfully
    Given the Vending Machine is initialized with mode "<Mode>"
    When the user selects startUp
    Then the vending machine is Ready
    And the vending machine presents the product menu
    When the user selects Coffee
    And the user selects strength
    Then the vending machine presents payment screen
    When the user pays "<C>" euro by coins
    Then the vending machine displays payment failure
    And the vending machine is Ready
```

```
@Card @MilkCan
Scenario Outline: The user orders Capuccino and Pays for it successfully by card and the Capuccino is delivered
    Given the Vending Machine is initialized with mode "<Mode>"
    When the user selects startUp
    Then the vending machine is Ready
    And the vending machine presents the product menu
    When the user selects Capuccino
    And the user selects strength
    Then the vending machine presents payment screen
    When the user pays "<C>" euro by card
    Then the vending machine delivers Capuccino
    And the vending machine displays thank you screen
    And the vending machine is Ready
```

```
Examples:
| Mode |
| Eco  |
| Normal |
| Fast |

@WaterDispenser
Scenario Outline: The user orders Water and it is delivered without payment
    Given the Vending Machine is initialized with mode "<Mode>"
    When the user selects startUp
    Then the vending machine is Ready
    And the vending machine presents the product menu
    When the user selects Water
    Then the vending machine delivers Water
    And the vending machine displays thank you screen
    And the vending machine is Ready
```

```
Examples:
| Mode |
| Eco  |
| Normal |
| Fast |

@Coin @Coffeebeans
Scenario Outline: The user orders Coffee and Pays more than needed the money is returned and Coffee is delivered
    Given the Vending Machine is initialized with mode "<Mode>"
    When the user selects startUp
    Then the vending machine is Ready
    And the vending machine presents the product menu
    When the user selects Coffee
    And the user selects strength
    Then the vending machine presents payment screen
    When the user pays "<C>" euro by coins
    Then the vending machine returns "<R>" euro
    And the vending machine delivers Coffee
    And the vending machine displays thank you screen
    And the vending machine is Ready
```

```
Examples:
| Mode |
| Eco  |
| Normal |
| Fast |
```

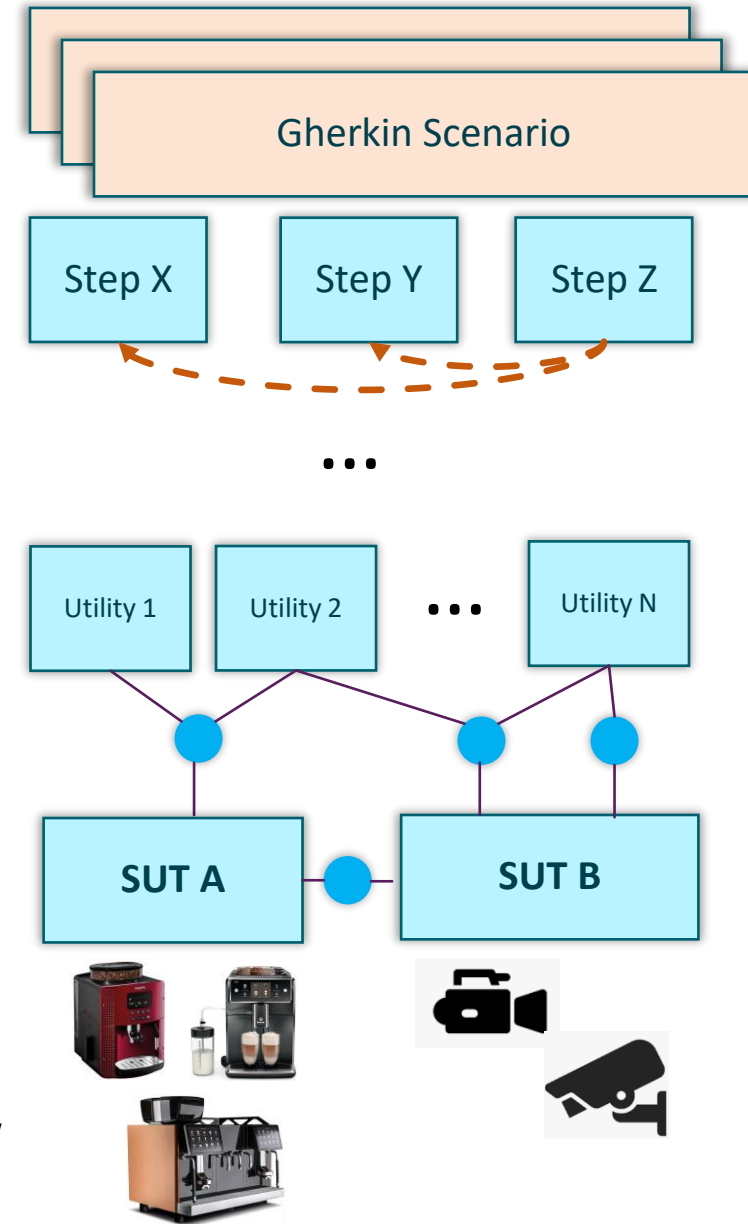
LIMITATIONS OF GHERKIN

Limited Language Expressivity

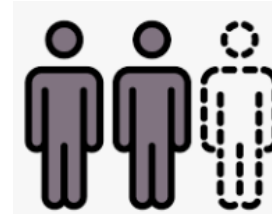
- Limited data types (int, real, bool and string)
- No branching, e.g., if-else, loops, etc.
- No support for configuration variability
- No support for references to step outputs

Leads-to

- **Maintenance burden over time**
 - Exploding number of test cases times system variants
 - Growing burden of keyword governance
 - How to prevent clones?
- **High complexity, lead time and missing overview**
 - Do all stakeholders have the same interpretation of steps?
 - Long scenarios with data relations between steps, implicit in their implementations
 - Multiple abstraction layers, written in different languages makes it difficult to get a global view



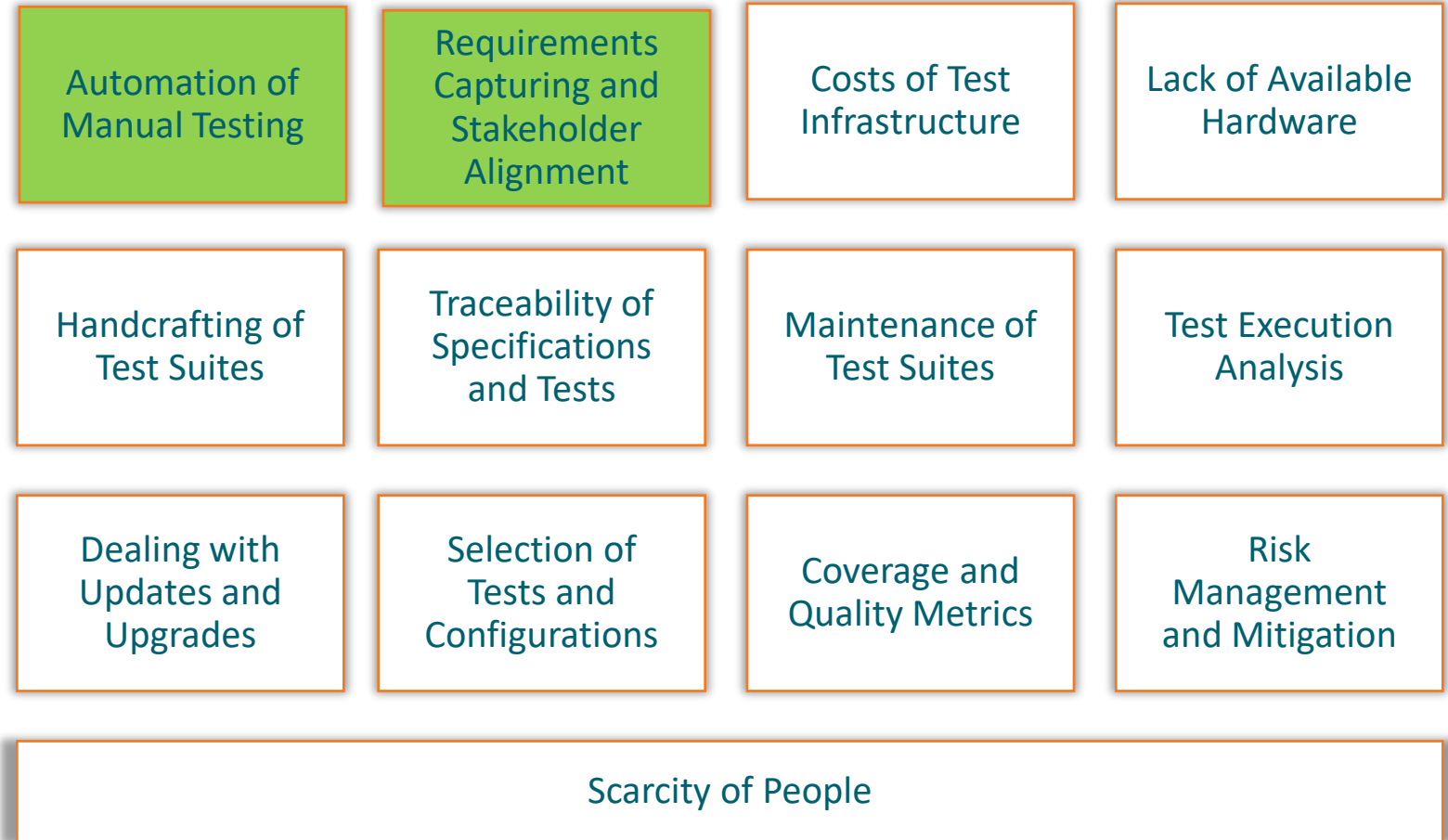
INNOVATIONS ARE NOT KEEPING UP



Lack of Trust



Off-the-shelf test automation frameworks **have not kept pace** with the **needs** to improve *efficiency* and *effectiveness* of testing complex systems.



REFLECTIONS

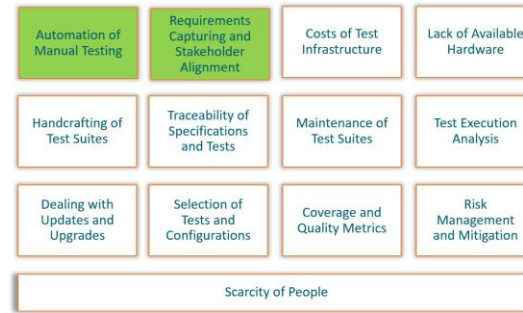


We would love to hear your thoughts on

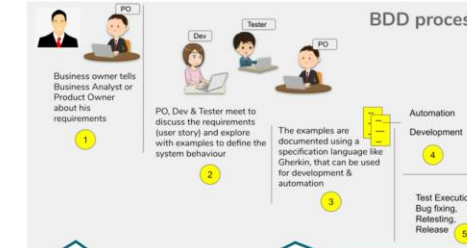
- How do you experience BDD in practice?
- Do you recognize these challenges?
- Do you think there are more, or less?
- Where do you feel the biggest need for innovation?

INNOVATIONS ARE NOT KEEPING UP

Off-the-shelf test automation frameworks **have not kept pace** with the *needs to improve efficiency and effectiveness* of testing complex systems.



BDD CHALLENGES



- High effort, cost and lead time
- Lack support for dealing with changes (updates/upgrades)
- High maintenance effort

What can we really say about coverage of specification and risks?
What does a keyword really do?

```

Feature: Login
  As a user
  I want to be able to log in to the application
  So that I can access my account information

Scenario: Successful login
  Given I am on the login page
  When I enter my username and password
  And I click the login button
  Then I should be taken to the dashboard page

@then["User navigates to Login Page"]
public void user_navigate_to_login_page() throws Throwable {
    driver.findElement(By.xpath("//input[@id='account']/a")).click();
}

@when["User enters Username and Password"]
public void user_enters_username_and_password() throws Throwable {
    driver.findElement(By.id("log")).sendKeys("testuser_17");
    driver.findElement(By.id("pwd")).sendKeys("123qaz!@WSX");
}

@then["Message displayed Login Successfully"]
public void message_displayed_login_successfully() throws Throwable {
    System.out.println("Login Successfully");
}

@when["User Logout from the Application"]
public void user_logout_from_the_application() throws Throwable {
    driver.findElement(By.xpath("//input[@id='account_logout']/a")).click();
}

@then["Message displayed Logout Successfully"]
public void message_displayed_logout_successfully() throws Throwable {
    System.out.println("Logout Successfully");
}
    
```

How to answer these questions: What is the coverage of my test suite? How much of each risk do each of my test cover? What does it mean to cover? etc...

October 21, 2025

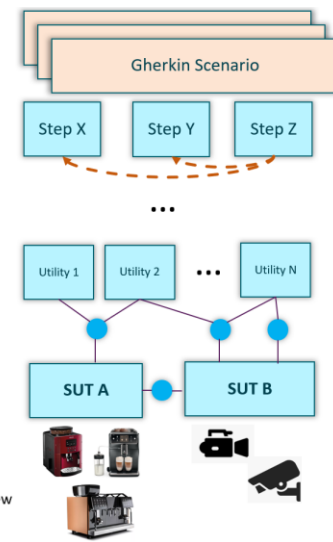
LIMITATIONS OF GHERKIN

Limited Language Expressivity

- Limited data types (int, real, bool and string)
- No branching, e.g., if-else, loops, etc.
- No support for configuration variability
- No support for references to step outputs

Leads-to

- Maintenance burden over time
 - Exploding number of test cases times system variants
 - Growing burden of keyword governance
 - How to prevent clones?
- High complexity and missing overview
 - Do all stakeholders have the same interpretation of steps?
 - Long scenarios with data relations between steps, implicit in their implementations
 - Multiple abstraction layers, written in different languages makes it difficult to get a global view




BREAK

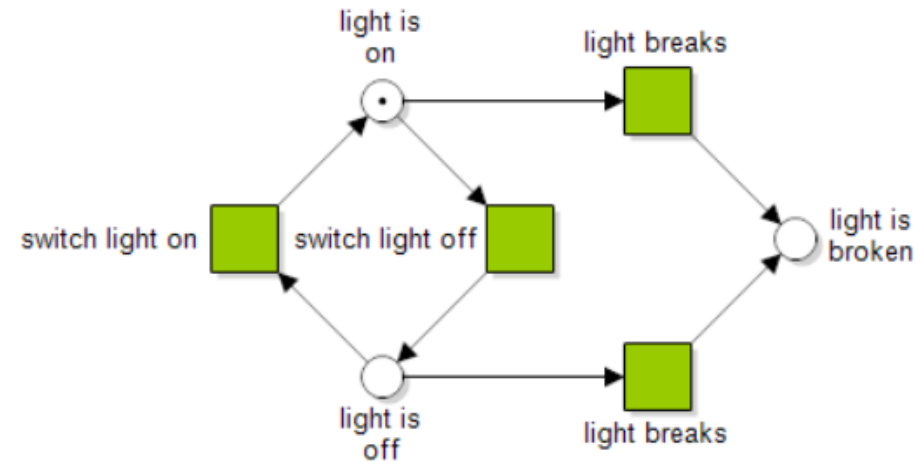


AGENDA

- Testing context and challenges in industry
 - Best practices in industry
 - Limitations of BDD test automation frameworks
 - Break + Reflections and Discussions
 - Take the next step in test automation with MBT
 - What do we mean by models and what is MBT
 - Collaborative specification of systems in context with BPMN4S
 - Automated Test Generation and Coverage
 - Live demo of a print shop
 - How did we evaluate the methodology and show value, insights into adoption at ASML
 - Break + Reflections and Discussions
 - Research and Development Roadmap
 - Can AI potentially help to improve systems testing?
 - Takeaways and Wrap-up
-

NON-DETERMINISTIC MODELS VERSUS SCENARIOS

A Model of a Switch and Light Bulb



Represents an Infinite Number of Scenarios

Switch Light On; Switch Light Off;

Switch Light On; Light Breaks

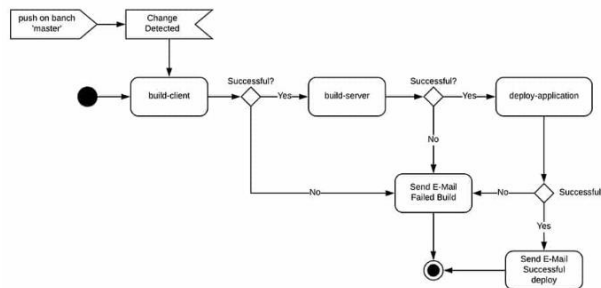
Switch Light On; Switch Light Off; Light Breaks

Switch Light On; Switch Light Off; Switch Light On; Switch Light Off; ...

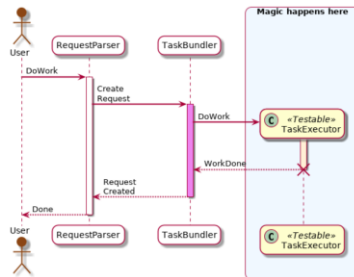
Switch Light On; Switch Light Off; Switch Light On; Switch Light Off; Light Breaks ...

...

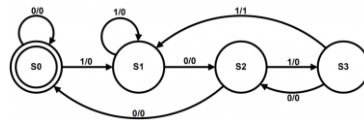
TYPES OF MODELS



Control-Flow View
Activity Diagram
Event-driven Process Chain
BPML



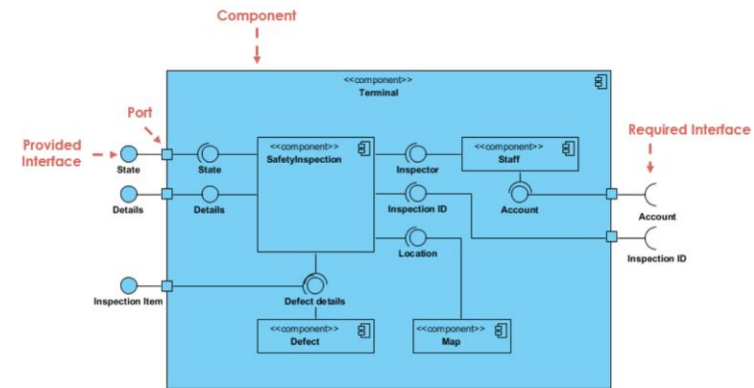
Scenario View
Sequence Diagram, MSC
Communication Diagram



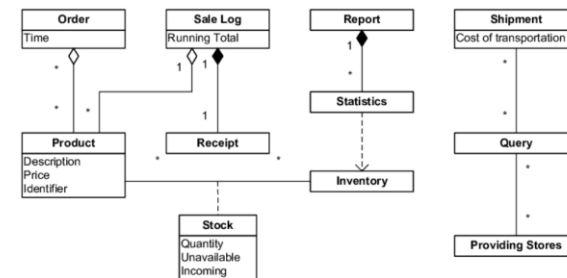
State View
State machine diagram
Finite Automaton
Statecharts

if the customer does act-seq turn on then act-seq ready should be shown by the coffeeMachine immediately-after

Constraint View
SCL, FRET, etc.

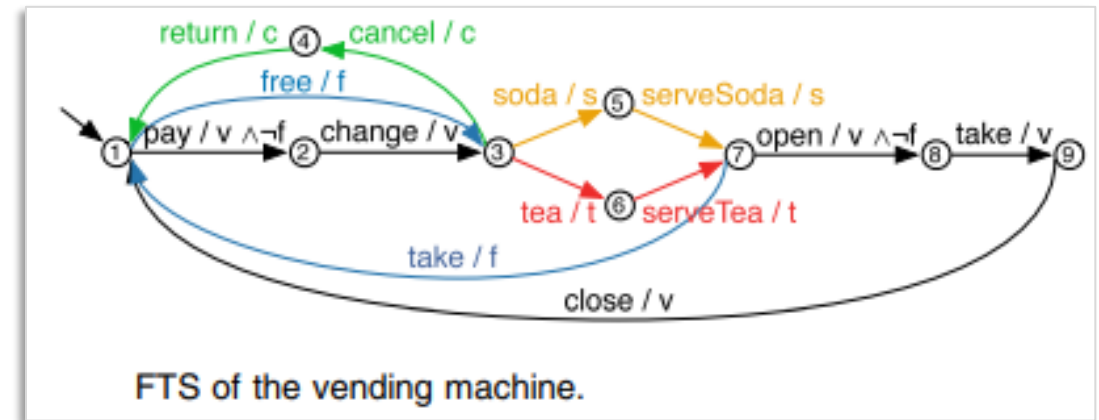
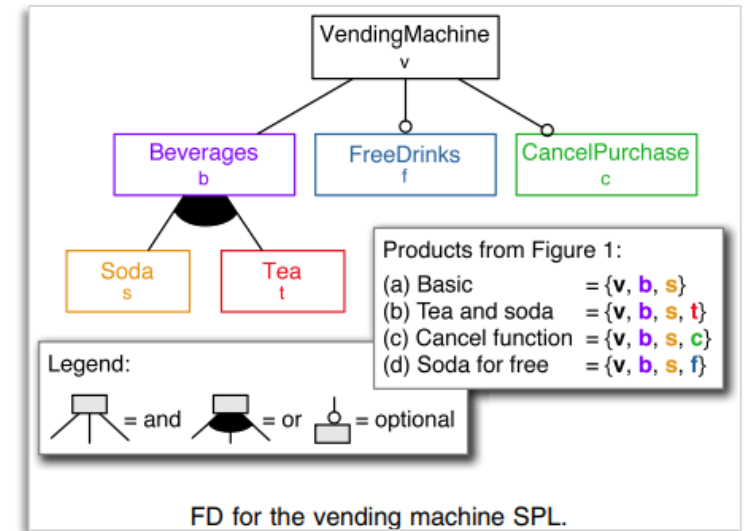
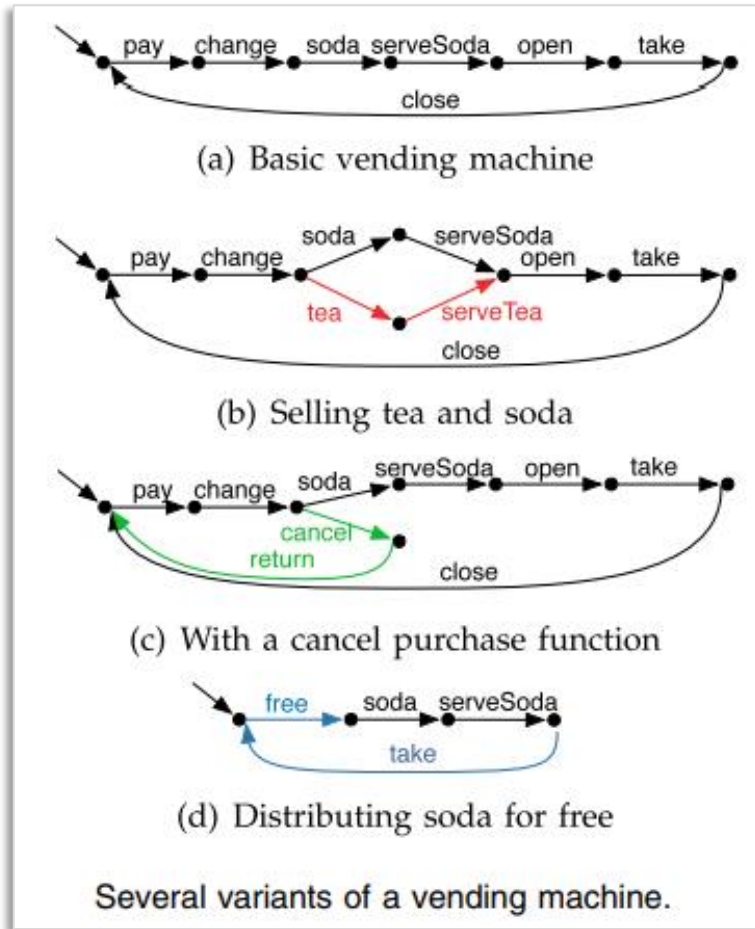


Structural View
Component Diagram



Data View
ER/Class Diagram

TYPES OF MODELS: VARIABILITY MODELING



WHAT IS MODEL-BASED TESTING

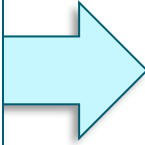
Offline MBT



Test Purpose



Model-based Test Selection and Generation



Abstract Test Suite

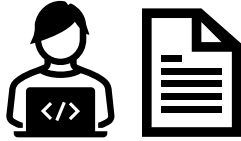
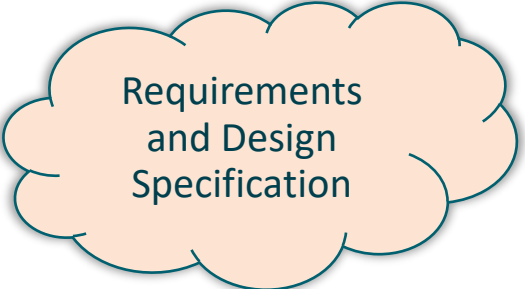
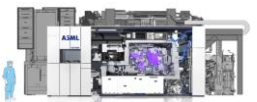


Concrete Test Suite

Test Runner



System Under Test



Specification Models



Conformance?

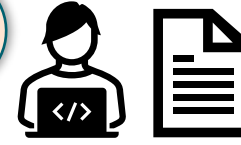
Verdict



WHAT IS MODEL-BASED TESTING

Online MBT

Risks,
Priorities,
Test Goals



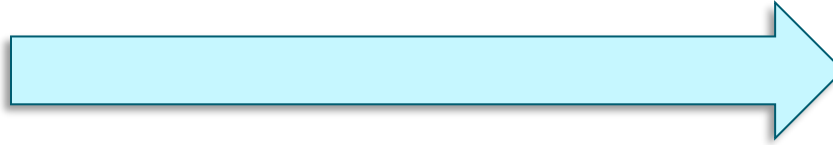
Test Purpose



Requirements
and Design
Specification



Specification Models



Model-based Test
Selection and
Generation



Test Adapter

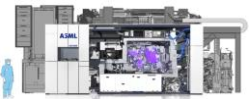
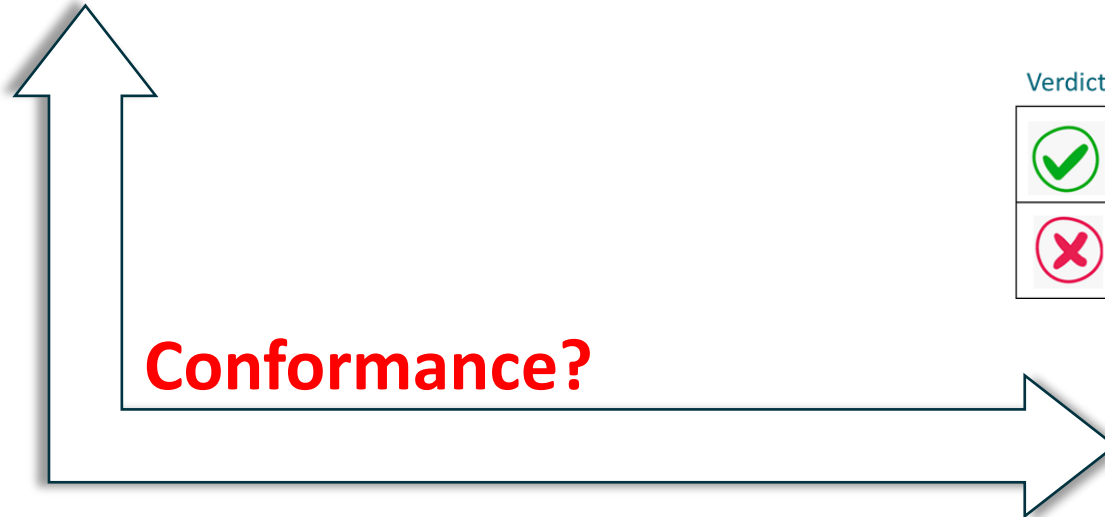
Verdict



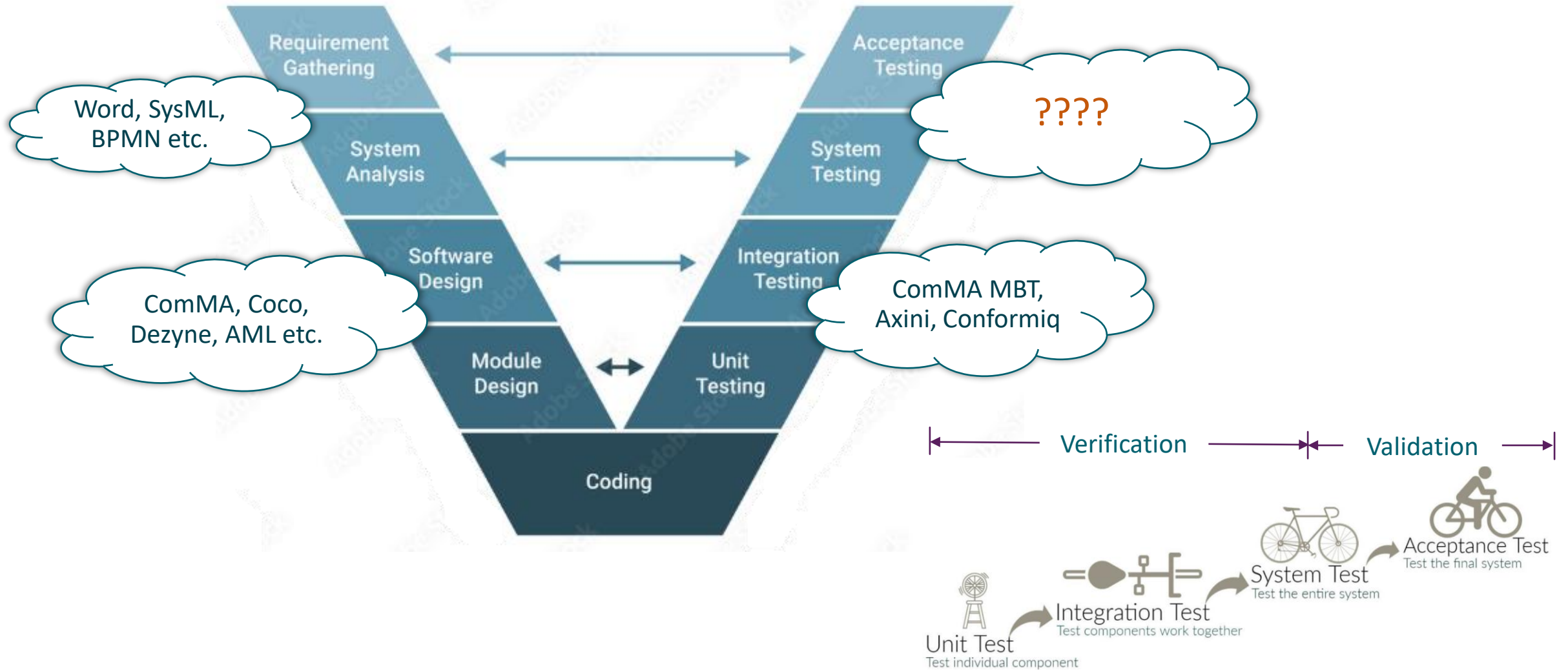
System Under Test



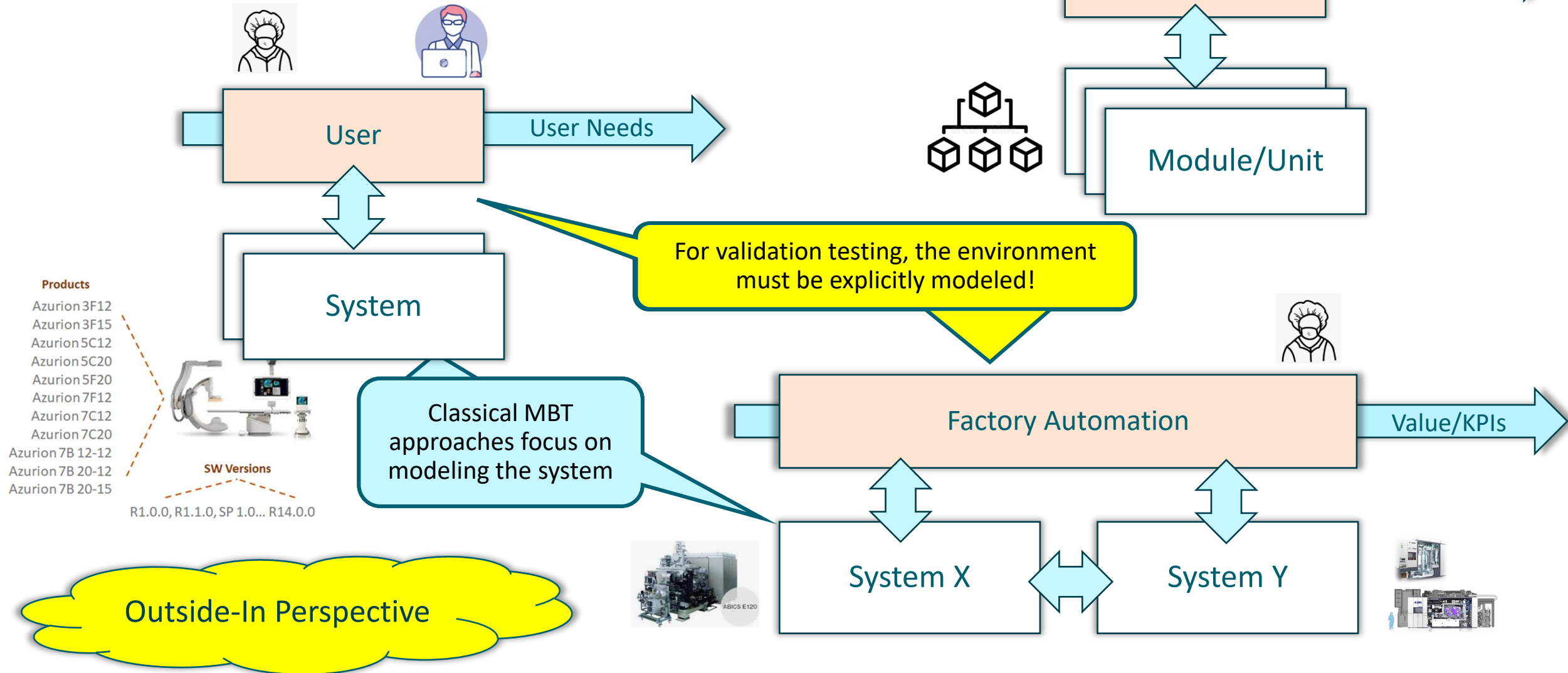
Conformance?



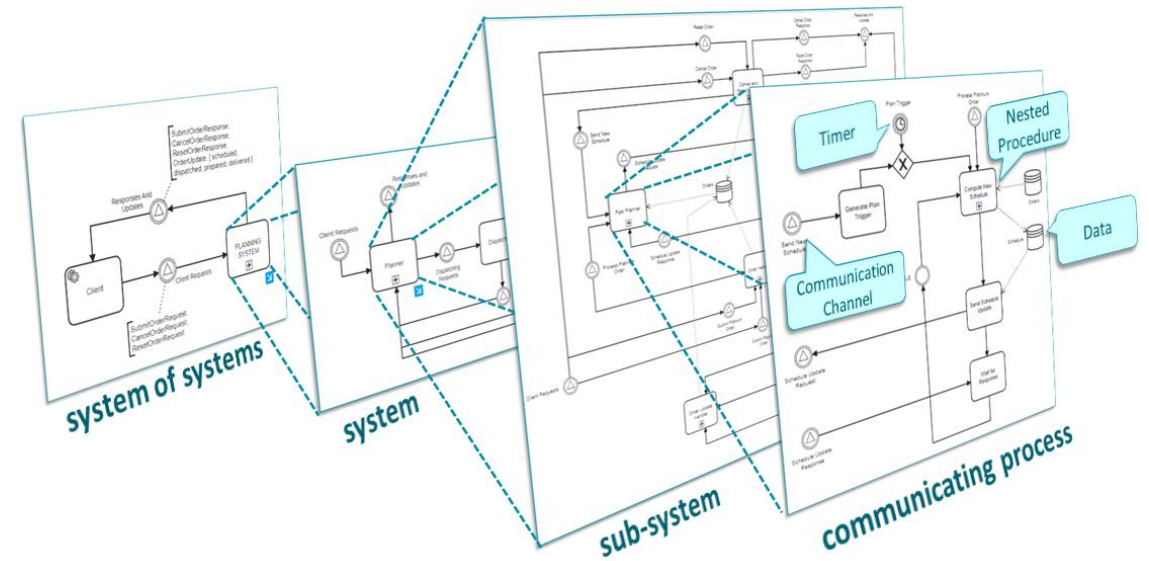
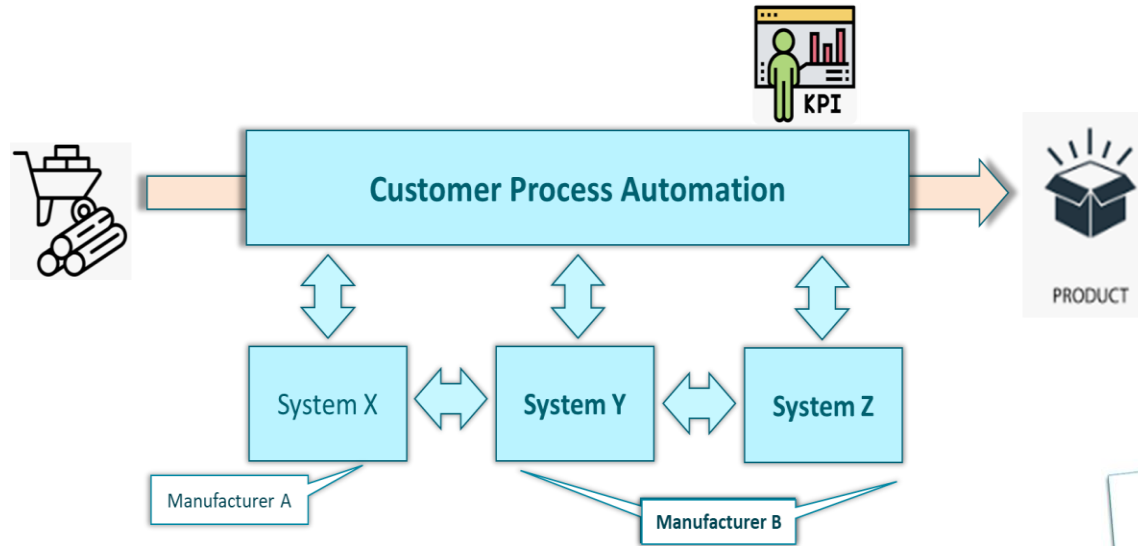
WHAT IS OUT THERE AND WHAT DO WE MISS?



VALIDATION OF SYSTEMS IN CONTEXT






SYSTEM OF SYSTEMS MODELING WITH BPMN4S

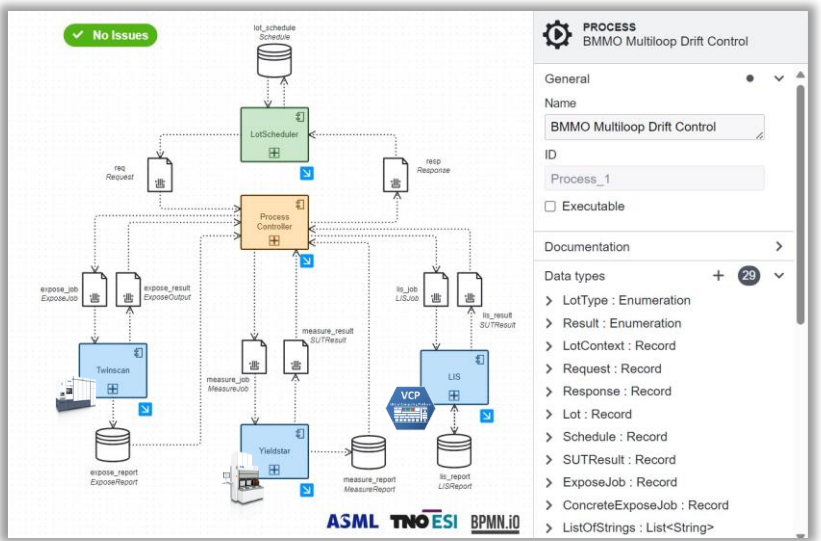


Methodology to Improve efficiency and effectiveness of system(-of-systems) testing processes by leveraging MBT technology

OFFLINE MBT FOR SOS IN CONTEXT

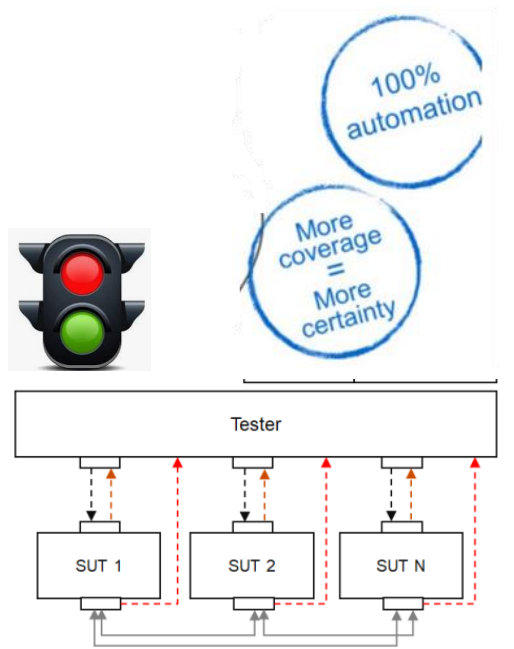
The BPMN4S Modeling Framework

- System Configurations 
- Concrete Test Data 
- Job Recipes Test Purposes 
- 



BDD Test Suite

-  Gherkin Scenario
-  C# Step Definitions

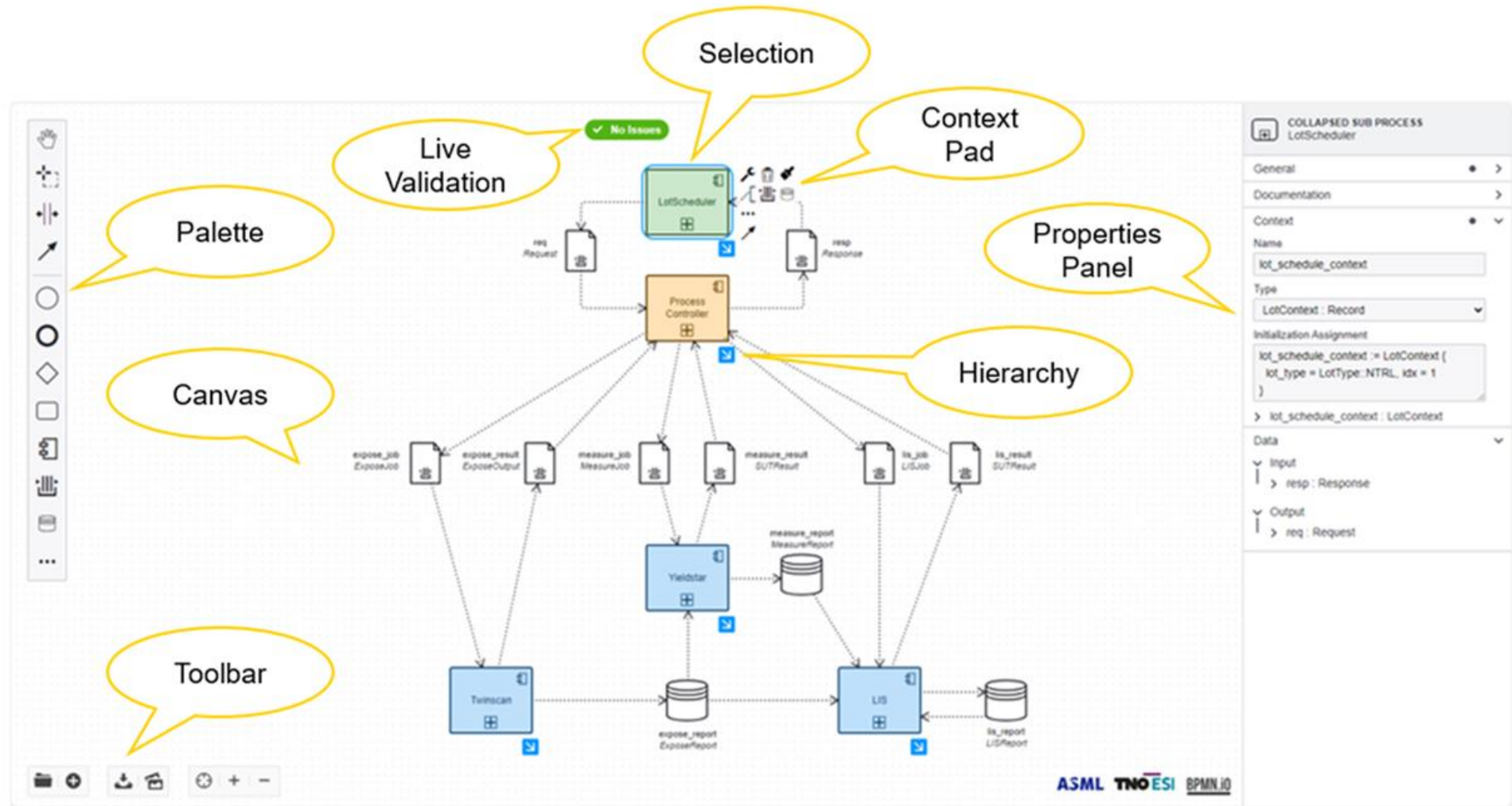


Collaborative specification and simulation of expected system behavior in context

- Automation *reduces lead time* by *eliminating* the need to hand-craft tests
- Models grow *linearly* as compared to test scenarios (easy maintenance)
- Provides sound basis to reason about *coverage*
- Intrinsic *traceability* between specifications and tests
- Sound basis for *automated impact analysis and regression test selection*

THE BPMN4S EDITOR

Extends the OMG standard for **BPMN 2.0**
Builds on the open source **bpmn.io** editor



SIMULATION OF BPMN4S MODELS

The screenshot displays a BPMN simulation interface. The main workspace shows a process diagram for a 'Production Controller' with the following elements:

- Start Event:** A circle leading to an XOR gateway.
- Task:** 'Send Production Request' (rounded rectangle) with a data store 'production_status' (cylinder) connected to it. A data object 'requests Request' is also connected to this task.
- Task:** 'Wait for Response' (rounded rectangle) with a data object 'Responses Response' connected to it.
- Task:** 'Prepare for Next Request' (rounded rectangle).
- End Event:** A circle reached from the 'Prepare for Next Request' task.
- Flow:** An arrow connects the 'Prepare for Next Request' task back to the XOR gateway, creating a loop.

On the right side, a 'Simulation State' window is open, showing a log of process events:

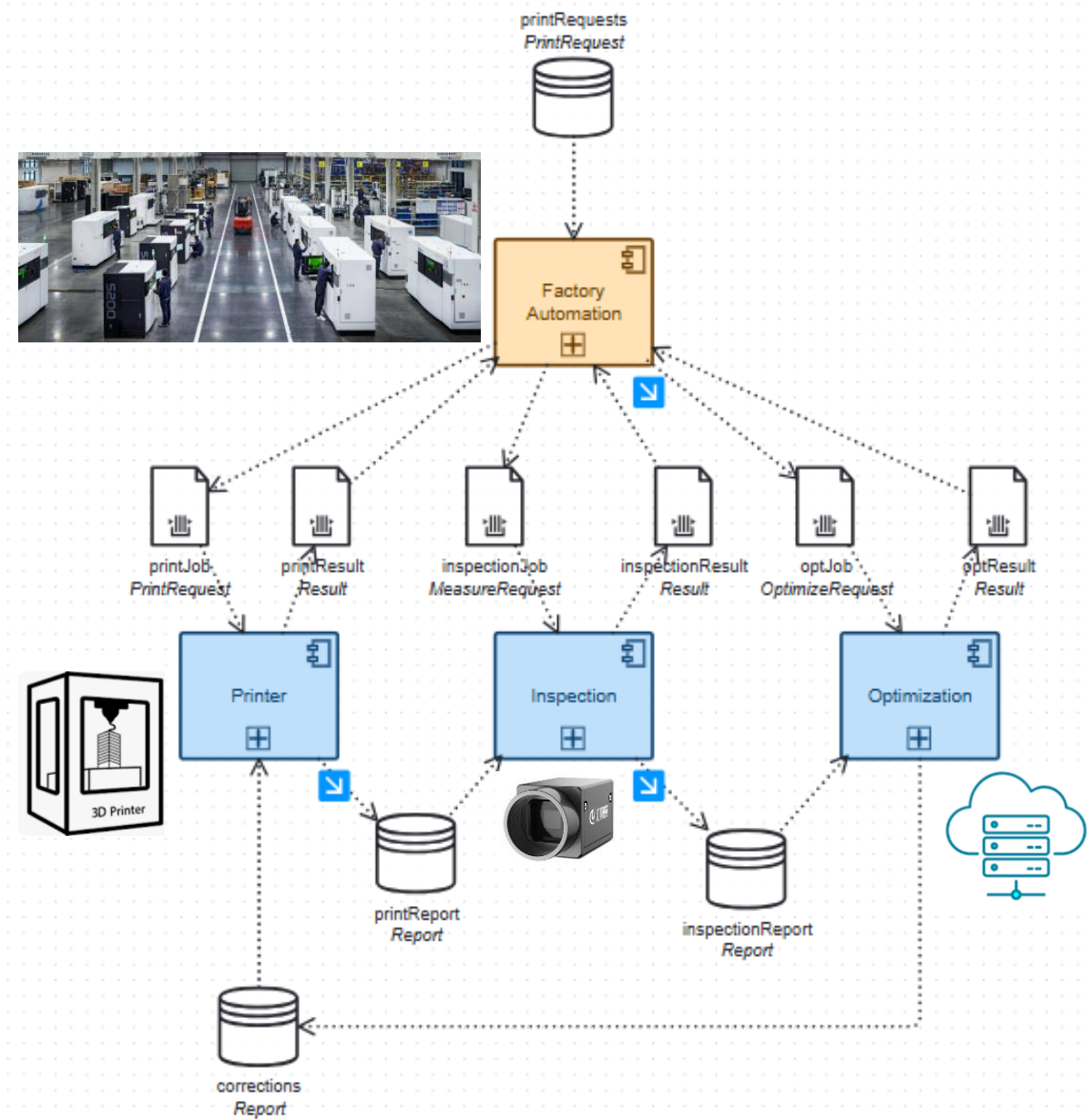
```

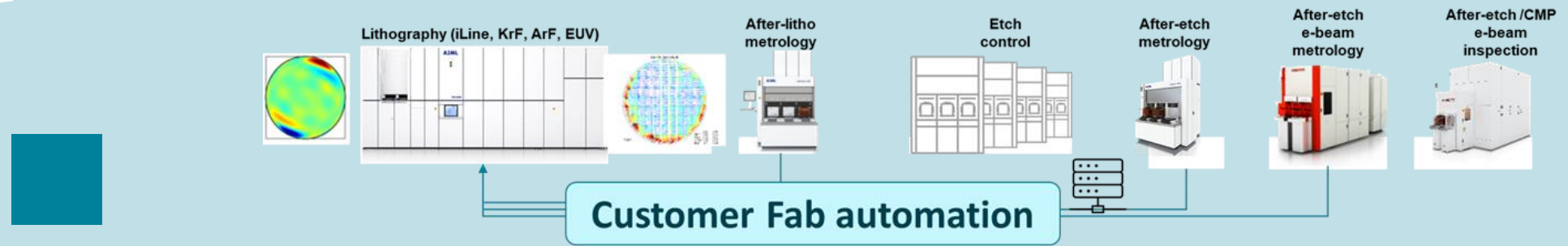
Simulation State
├── Process_1.Controller.Production Controller [1f63m6g]
│   ├── { "id": 0, "payload": "REQ1" }
│   └── { "id": 1, "payload": "REQ2" }
├── Process_1.Controller.production_status [1y4vneug]
│   └── { "run_production": true }
├── Process_1.responses [11ye1v9]
│   └── { "result": "Result::OK", "id": 1 }
└── Process_1.responses [1m01e44]
    └── { "result": "Result::OK", "id": 0 }
    
```

The interface includes a 'Token Simulation' control bar at the top with a play button and several colored status indicators. A breadcrumb trail at the top left reads 'Process_1 > Controller > Production Controller'. The bottom right corner features logos for ASML, TNO ESI, and BPMN.io.

DEMO

Simple 3D Printshop Model and Simulation





HOW DID WE EVALUATE THE METHODOLOGY AND SHOW VALUE AT ASML

REFLECTIONS



We would love to hear your thoughts on

- What did you think about the BPMN4S language and its simulation features? Is this a step up in fostering collaboration and creating shared understanding?
- Do you agree that with such an approach the focus will be more on understanding and capturing the domain and risks?
- Do you see such an approach resulting in higher efficiency and effectiveness? If not, what concerns you the most?

Models



BDD Test Suite



Gherkin
Scenario

C# Step
Definitions





BREAK



AGENDA

- Testing context and challenges in industry
 - Best practices in industry
 - Limitations of BDD test automation frameworks
 - Break + Reflections and Discussions
 - Take the next step in test automation with MBT
 - What do we mean by models and what is MBT
 - Collaborative specification of systems in context with BPMN4S
 - Automated Test Generation and Coverage
 - Live demo of a print shop
 - Break + Reflections and Discussions
 - How did we evaluate the methodology and show value, insights into adoption at ASML
 - Research and Development Roadmap
 - Can AI potentially help to improve systems testing?
 - **Takeaways and Wrap-up**
-

RESEARCH AND DEVELOPMENT ROADMAP

Further enhance collaborative specification

- Version management, reviewing, model comparisons, replay of test cases in simulation

Focus testing efforts where it matters

- Reason about product risks and priorities
- Exploit usage profiles, test execution logs and defects

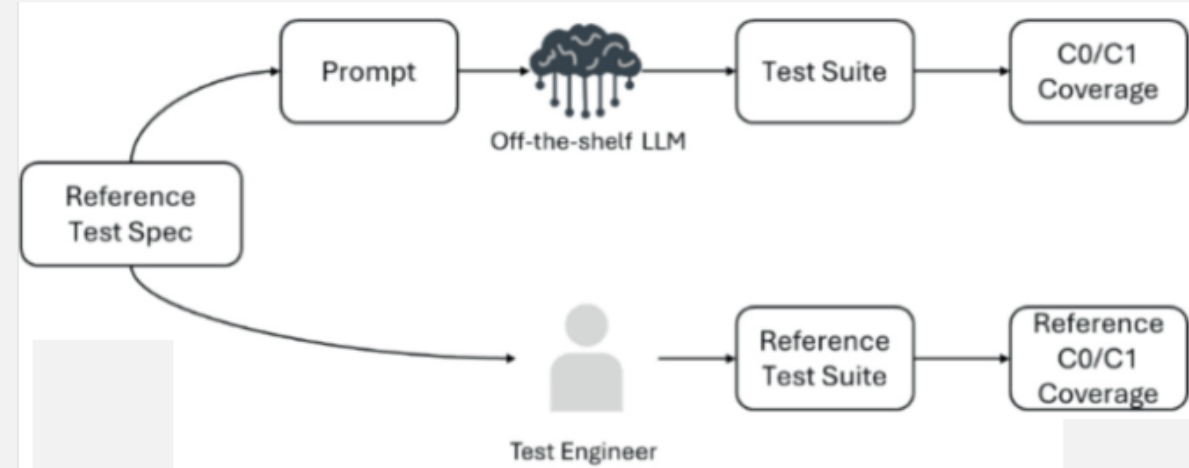
Faster root-cause analysis

- Model-assisted root-cause analysis of failing tests
- Coverage analysis and test reporting

Higher coverage with online model-based testing

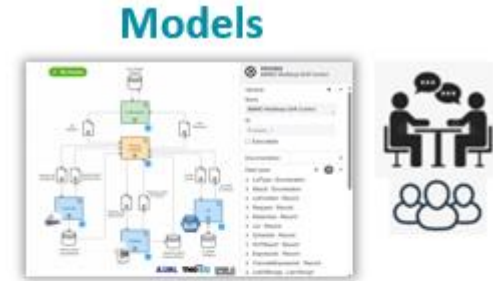
- On-the-fly test generation and execution
- More representative testing of real-world dynamics

CAN AI AUGMENT BDD + MBT?

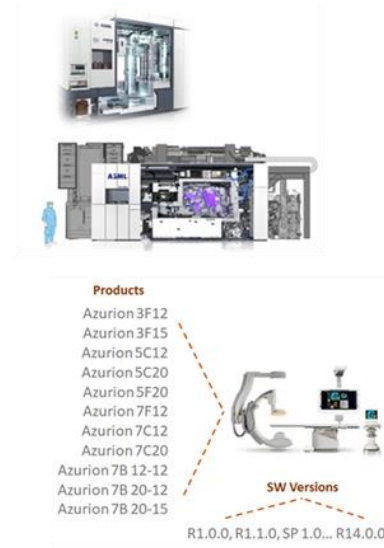


TAKE AWAYS

- Systems are getting more complex
- Current approach to testing is not scaling
- BDD truly triggered the shift-left movement
- But there is an urgent need for innovation in technologies supporting BDD
- Model-based testing can augment BDD by taking collaboration to the next level and improve test coverage
- LLMs may hold the answer to the modeling hurdle in MBT
- Finding the right application of AI and embedding in existing tech/WoW is not trivial



BDD Test Suite



How to enable faster time to market with promised quality and lower costs?

Automation of Manual Testing	Risk Management and Mitigation	Costs of Test Infrastructure	Lack of Available Hardware
Handcrafting of Test Suites	Traceability of Specifications and Tests	Maintenance of Test Suites	Test Execution Analysis
Dealing with Updates and Upgrades	Selection of Tests and Configurations	Coverage and Quality Metrics	Defect Leakage and Exploiting Usage Profiles
Scarcity of People			



TNO **ESI**

Powered by industry
and academia