Integrating Executable Requirements in Prototyping

Yann Le Goff
Pierre Laborde
Éric Le Pors
Mickaël Kerboeuf
Alain Plantec

www.thalesgroup.com

www.univ-brest.fr

www.labsticc.fr

THALES
Building a future we can all trust
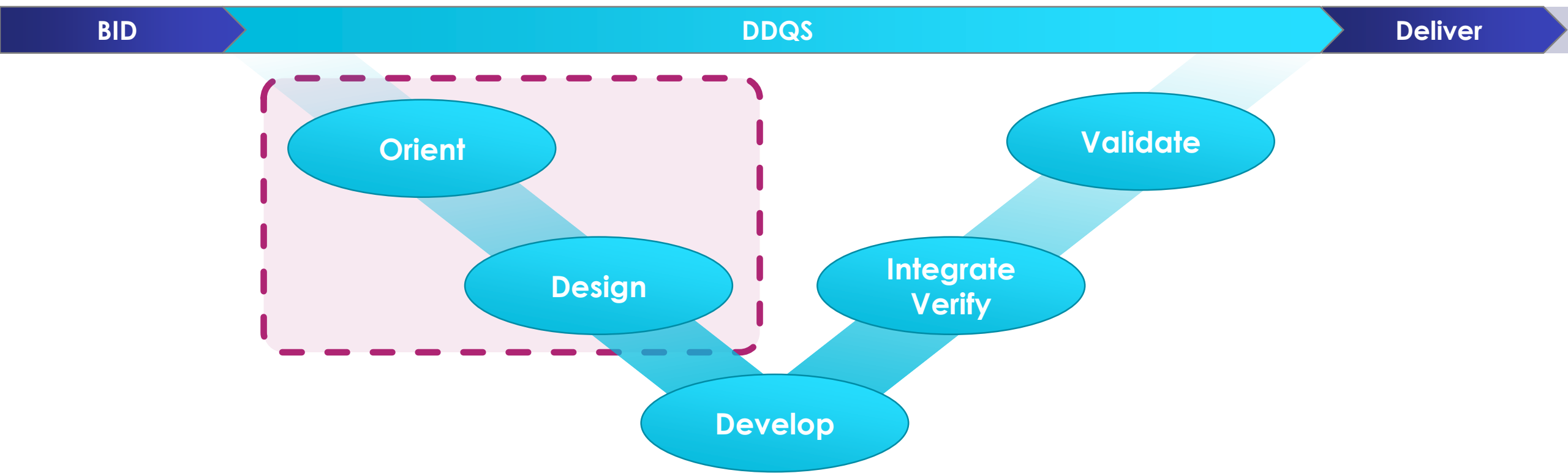
UBO
Université de Bretagne Occidentale

Lab-STICC
UMR 6285

Table of Contents

Project Lifecycle at Thales

Different Teams working together
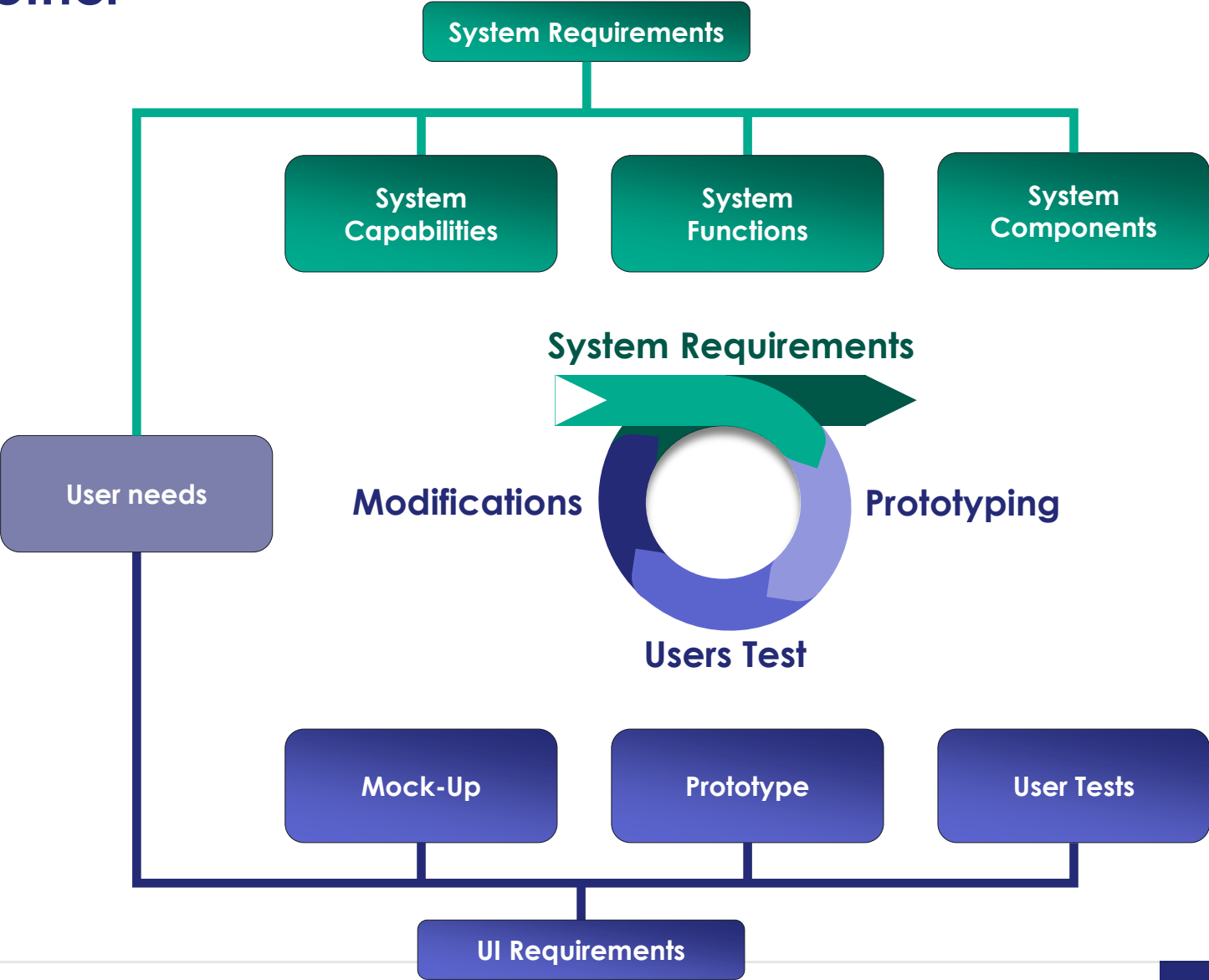
> System Engineering

- ‣ **Definition -** The interdisciplinary approach governing the total technical effort required to transform a requirement into a system solution.
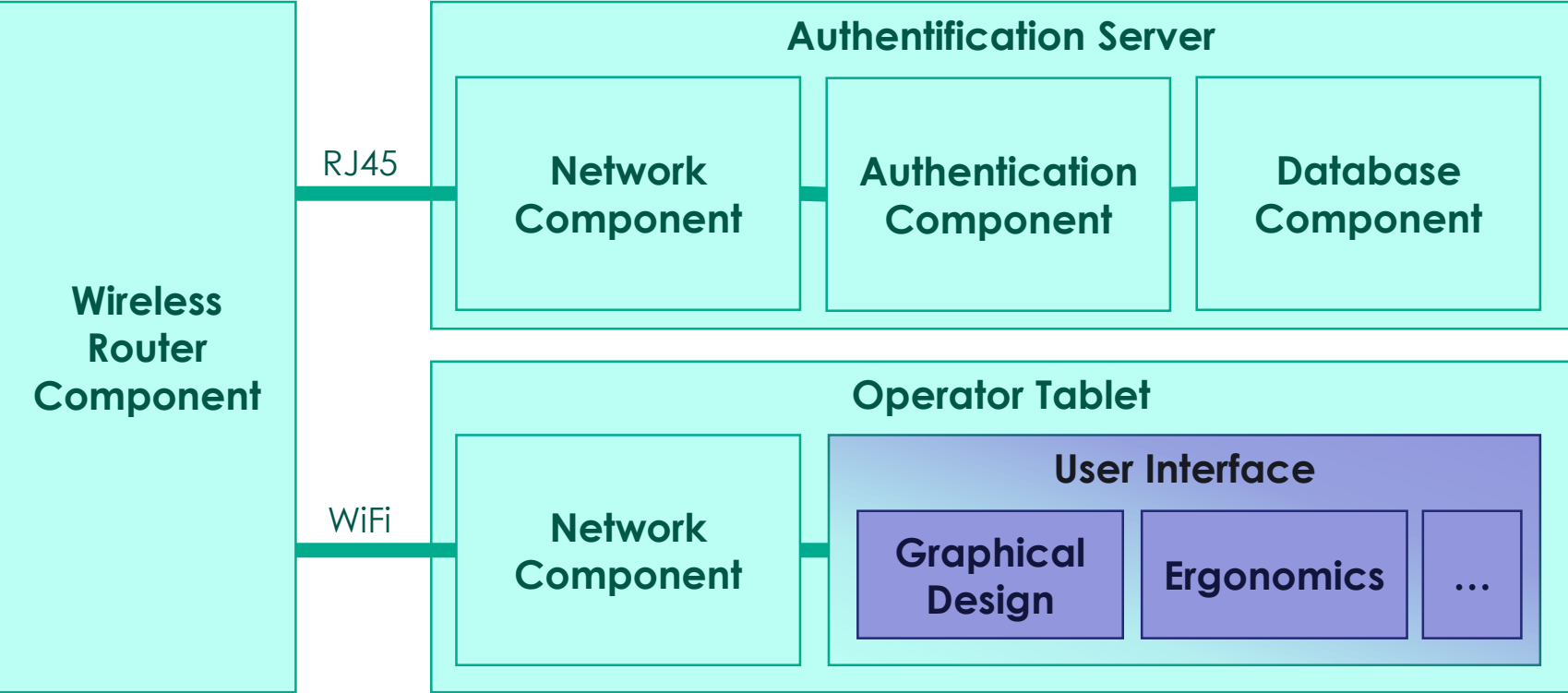
Source: Thales

> User Experience

- ‣ **Definition -** Person's perceptions and responses that result from the use or anticipated use of a product, system or service
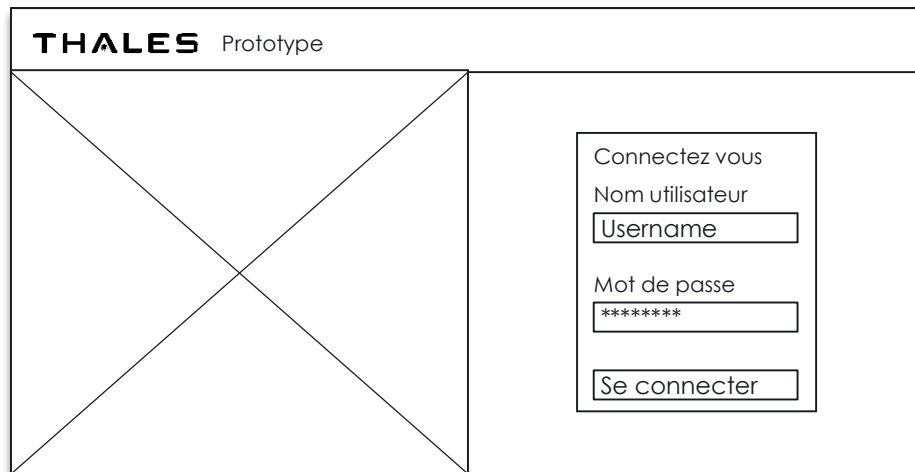
Source: ISO 24765 et Thales

System Requirements

System Capabilities — **System Functions** — **System Components**

System Requirements

User needs

Modifications — **Prototyping**

Users Test

Mock-Up — **Prototype** — **User Tests**

UI Requirements

An example

Authentification Server
- Network Component
- Authentication Component
- Database Component

Wireless Router Component

RJ45 — connection to Authentification Server

WiFi — connection to Operator Tablet

Operator Tablet
- Network Component
- **User Interface**
  - Graphical Design
  - Ergonomics
  - …

SYS System Engineering
UX UX Engineering

THALES — Building a future we can all trust

UBO — Université de Bretagne Occidentale

Lab-STICC UMR 6285

A prototype to specify the User Interfaces

> Prototype

‣ Preliminary type, form, or instance of a system that serves as a model for later stages or for the final, complete version of the system.

‣ **A prototype is used to get feedback from users for improving and specifying a complex human interface, for feasibility studies, or for identifying requirements.**

  – *Definition ISO 24765*

Low Fidelity (LoFi) : WireFrames, Mock-Ups
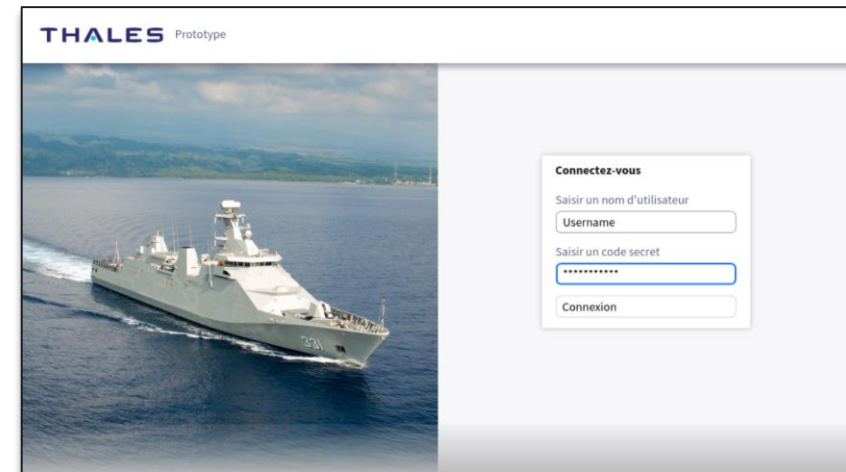
High Fidelity (HiFi) : Dynamic Prototypes

Table of Contents

THALES
Building a future we can all trust

UBO
Université de Bretagne Occidentale

Lab-STICC
UMR 6285

Avoid regression on requirements

System Requirement — **The UI must display an error message in case of an invalid login.**

Prototype : Version 1

Prototype : Version 2

Prototype : Version 3
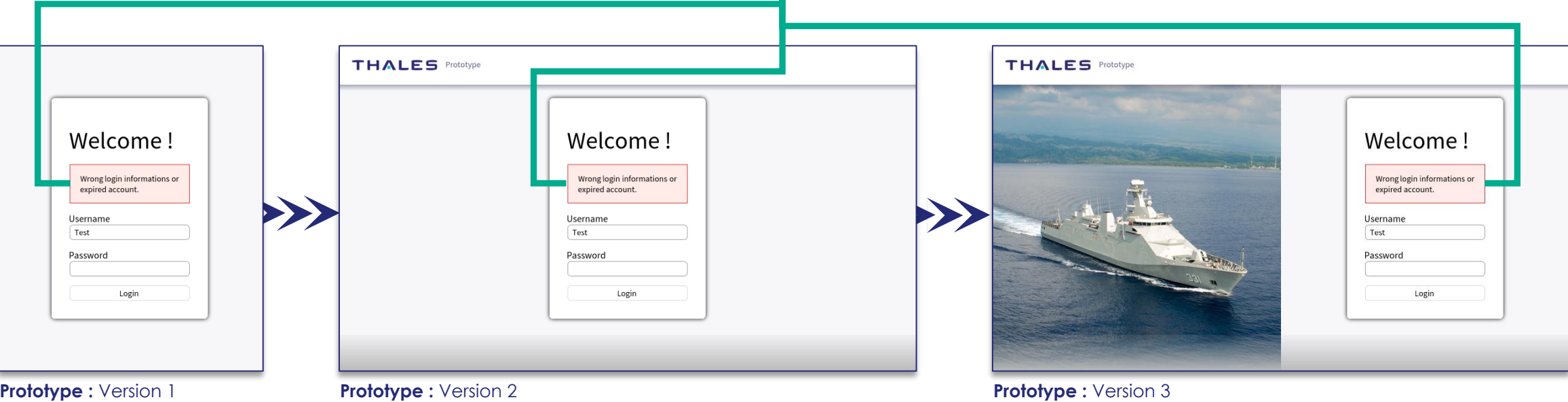
Table of Contents

> Context

> Issue

> **Proposal**

> **Conclusion**

Meta-model

Link the requirements to the prototype (1/3)

System Requirement | The UI must display an error message in case of an invalid login.

*Requirement 1 :***Requirement**

+ titre: 'The UI must display an error message in case of an invalid login.'
+ description: '…'
+ source: example.CAPELLA
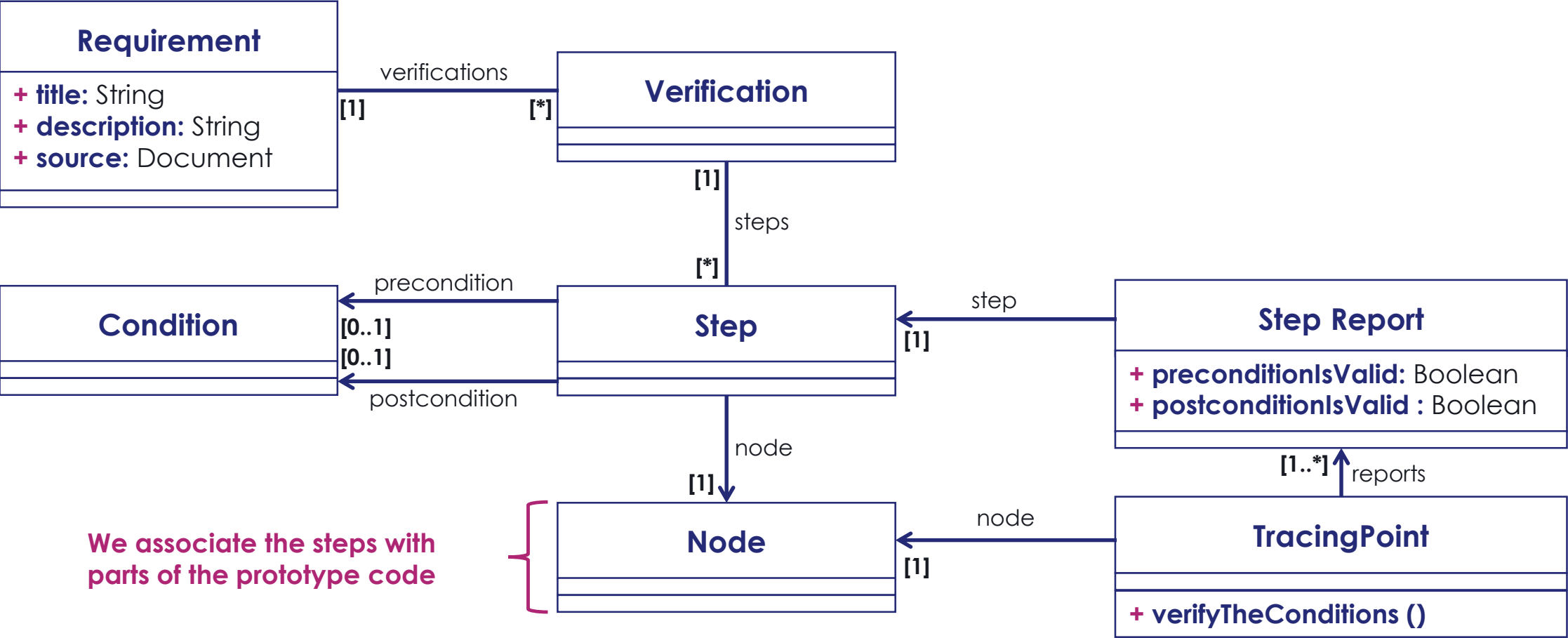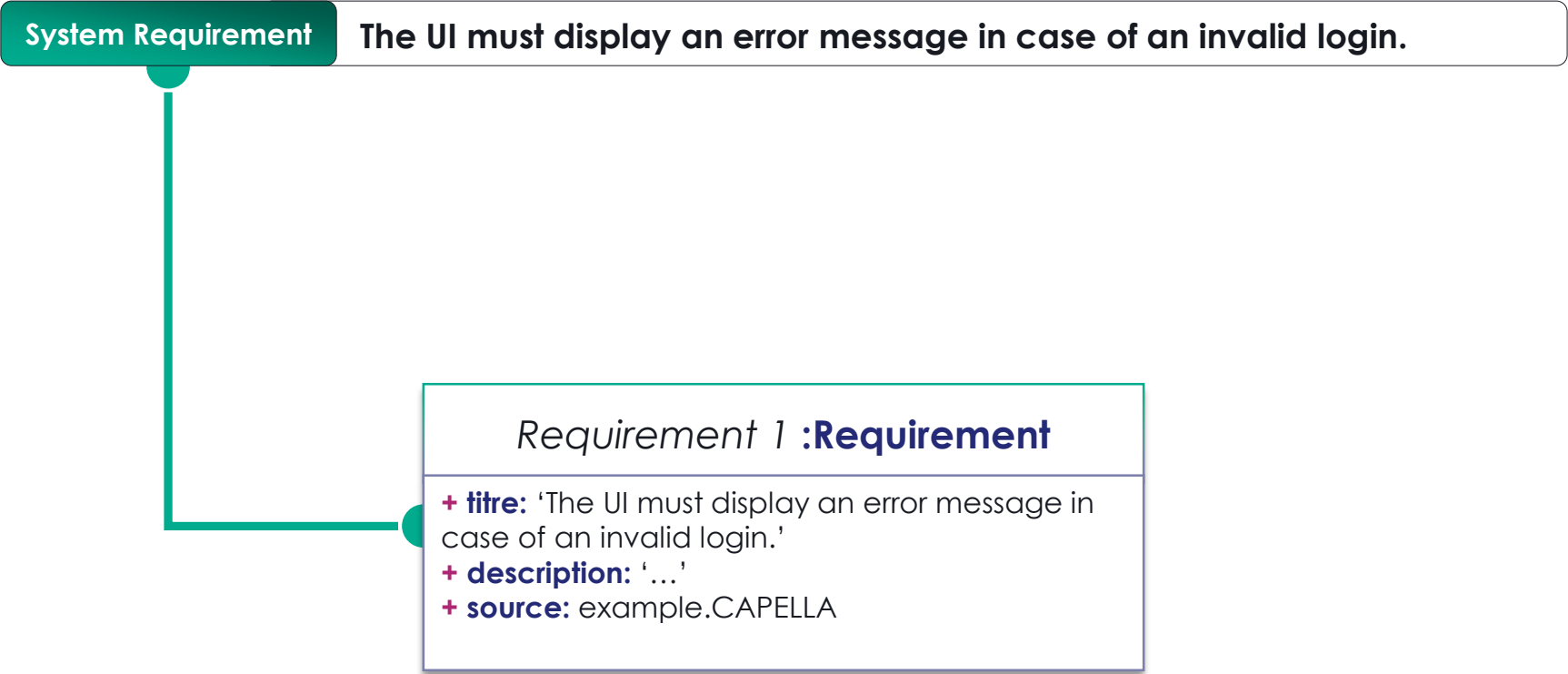
Link the requirements to the prototype (2/3)

Requirement 1 :Requirement

+ **titre:** 'The UI must display an error message in case of an invalid login.'
+ **description:** '…'
+ **source:** example.CAPELLA

```
1  loginButtonAction
2
3     | simulationResult |
4  simulationResult := simulation
5     canAuthenticateUsername: self username
6     password: self password.
7  simulationResult
8     ifTrue: [ self validateLogin ]
9     ifFalse: [ self refuseLogin ]
10 |
```

Verification of the login information

The login information is valid
The login information is not valid

Link the requirements to the prototype (3/3)
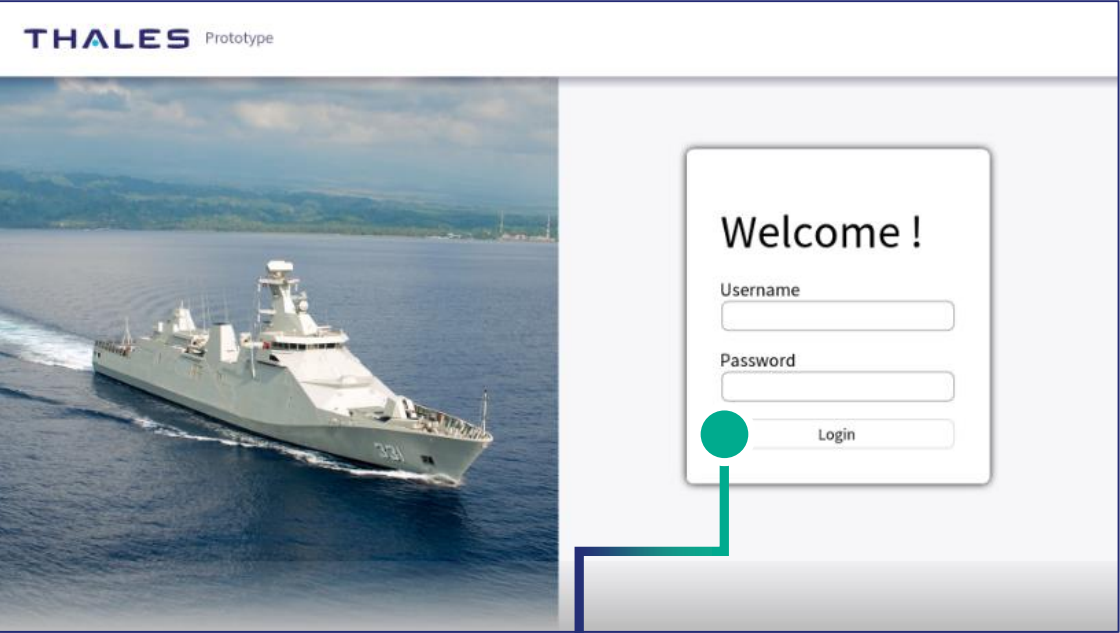
Exigence 1 :Exigence

+ **titre:** 'The UI must display an error message in case of an invalid login.'
+ **description:** '…'
+ **source:** example.CAPELLA

_ :Verification

_ :Step

+ **precondition:** 'The error is not visible'
+ **postcondition:** 'The error is visible'

+ node

```
1  loginButtonAction
2
3     | simulationResult |
4     simulationResult := simulation
5         canAuthenticateUsername: self username
6         password: self password.
7     simulationResult
8         ifTrue: [ self validateLogin ]
9         ifFalse: [ self refuseLogin ]
10 |
```

The login information is not valid

Create, Install and Visualize TracingPoints

Installation of TracingPoint

List of requirements

Description

List of verifications and steps

Status of Step Reports

Targeted node

Precondition

Postcondition

Demo (1/2)

Demo (2/2)

The DebugPoint System

Node

[1]

node

DebugPoint

+ tag: { **#before** . **#after** }

TracingPoint

+ verifyTheConditions ()

‣ **Verify the precondition (with the tag #before)**

‣ **Verify the postcondition (with the tag #after)**

‣ **Produce a step report**

Handling non-local return and unwinding exceptions (1/2)

> Nominal Case

‣ The precondition is executed

‣ The node refuseLogin is executed

‣ The postcondition is executed

```
loginButtonAction

    | simulationResult |
    simulationResult := simulation
        canAuthenticateUsername: self username
        password: self password.
    simulationResult
        ifTrue: [ self validateLogin ]
        ifFalse: [ self refuseLogin ]
```

> Handling of non local returns

‣ The precondition is executed

‣ The node refuseLogin is executed

‣ **The method return**

‣ The postcondition is **not** executed

```
loginButtonAction

    | simulationResult |
    simulationResult := simulation
        canAuthenticateUsername: self username
        password: self password.
    simulationResult
        ifTrue: [ self validateLogin ]
        ifFalse: [
            [
                self refuseLogin.
                ^ self ] value ]
```

> Handling of exceptions

‣ The precondition is executed

‣ The node refuseLogin is executed

‣ **The Error is signaled**

‣ The postcondition is **not** executed

```
loginButtonAction

    | simulationResult |
    simulationResult := simulation
        canAuthenticateUsername: self username
        password: self password.
    simulationResult
        ifTrue: [ self validateLogin ]
        ifFalse: [ self refuseLogin ]
```
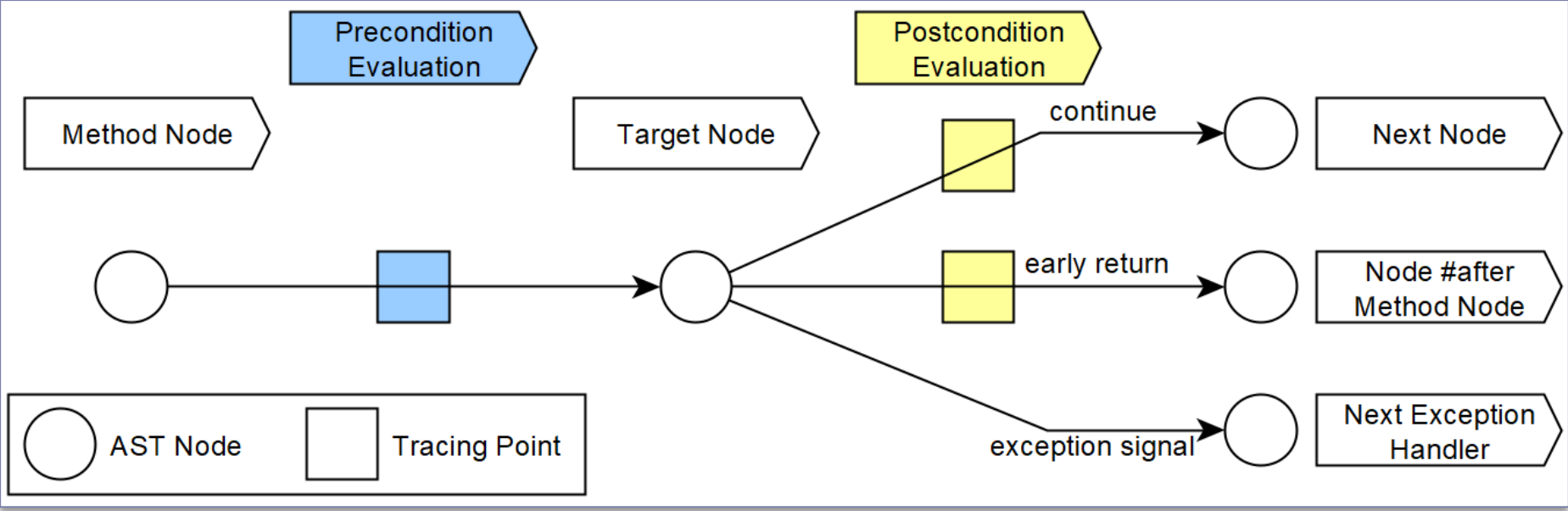
```
refuseLogin

    Error signal: 'Wrong login information'
```

THALES
Building a future we can all trust

UBO
Université de Bretagne Occidentale

Handling non-local return and unwinding exceptions (2/2)

> The exceptions are currently not handled

Table of Contents

Conclusion

> **Integrating Executable Requirements in Prototyping**

> **Tool to install the TracingPoint in the code**

> **Futur Works:**

‣ Testing the solution on industrial prototypes

‣ Generalize to other models beyond System models

‣ Track the evolution of requirements over time