Interreg North Sea

Co-funded by the European Union

STORM_SAFE

STORM_SAFE FORMAL METHODS

Matthias Volk (Eindhoven University of Technology)
Martijn Goorden (Eindhoven University of Technology)
Maarten Goegebeur (Flemish Environment Agency)
Theodora Popescu (French National inland Waterways)
Marion Degache (French National inland Waterways)
Wendy Clerx (Rijkswaterstaat)
Marjolein Weergang (Rijkswaterstaat)
Yigal Levin (Rijkswaterstaat)

Agenda

1. Welcome and opening
2. Introduction to the STORM_SAFE Project
3. Why it matters
4. What are formal methods?
5. Pilot Case study
   Presentation of a real pilot project demonstrating how formal methods are applied in practice.
6. Join the **Software reliability ECOSYSTEM**
   Opportunities for collaboration, knowledge sharing, and involvement in the growing STORM_SAFE community.
7. Discussion and Q&A

Interreg North Sea

Co-funded by the European Union

STORM_SAFE

Quick Poll

What comes to mind when you hear “Formal Methods”?

Interreg North Sea

Co-funded by the European Union

STORM_SAFE

Quick Poll

Has anyone here personally experienced a **software failure** in their professional work that caused a **major disruption**?

Show of hands

STORM_SAFE

Collaboration in motion – Impact in progress

Co-funded by the European Union

Project Duration: 2024 - 2027

Partners:

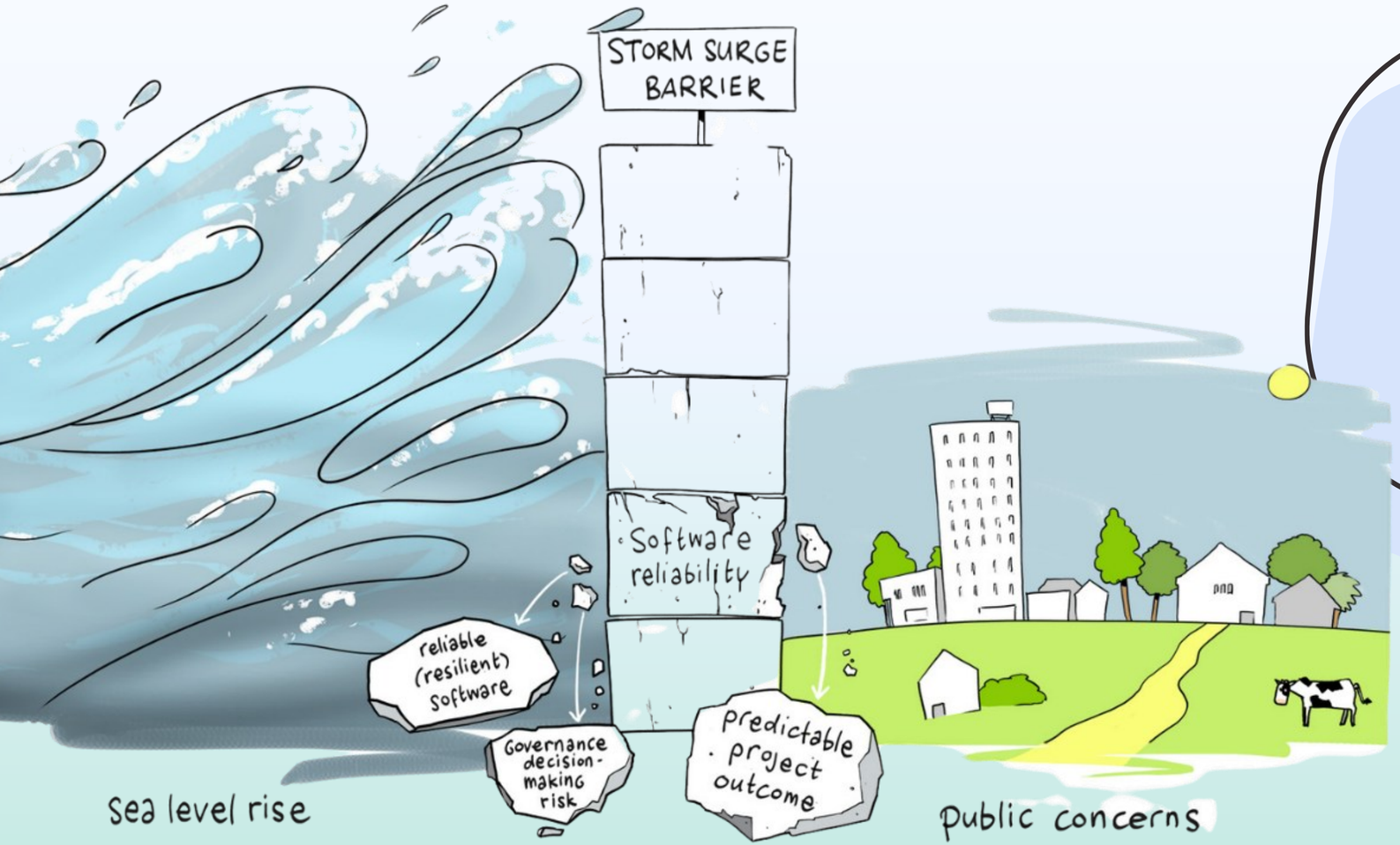
- Rijkswaterstaat – Ministerie van Infrastructuur en Waterstaat
- Göteborgs Stad
- Ministry of Environment of Denmark – Environmental Protection Agency
- Aalborg University
- VNF – Voies navigables de France
- NLWKN – Niedersächsischer Landesbetrieb für Wasserwirtschaft, Küsten- und Naturschutz
- University of Twente.
- Universität des Saarlandes
- Flanders – State of the Art
- Lunds Universitet
- TU/e – Eindhoven University of Technology

Associate Partners:

- US Army Corps of Engineers
- Environment Agency
- Fjordgruppen

Time to act

Interreg North Sea
Co-funded by the European Union
STORM_SAFE



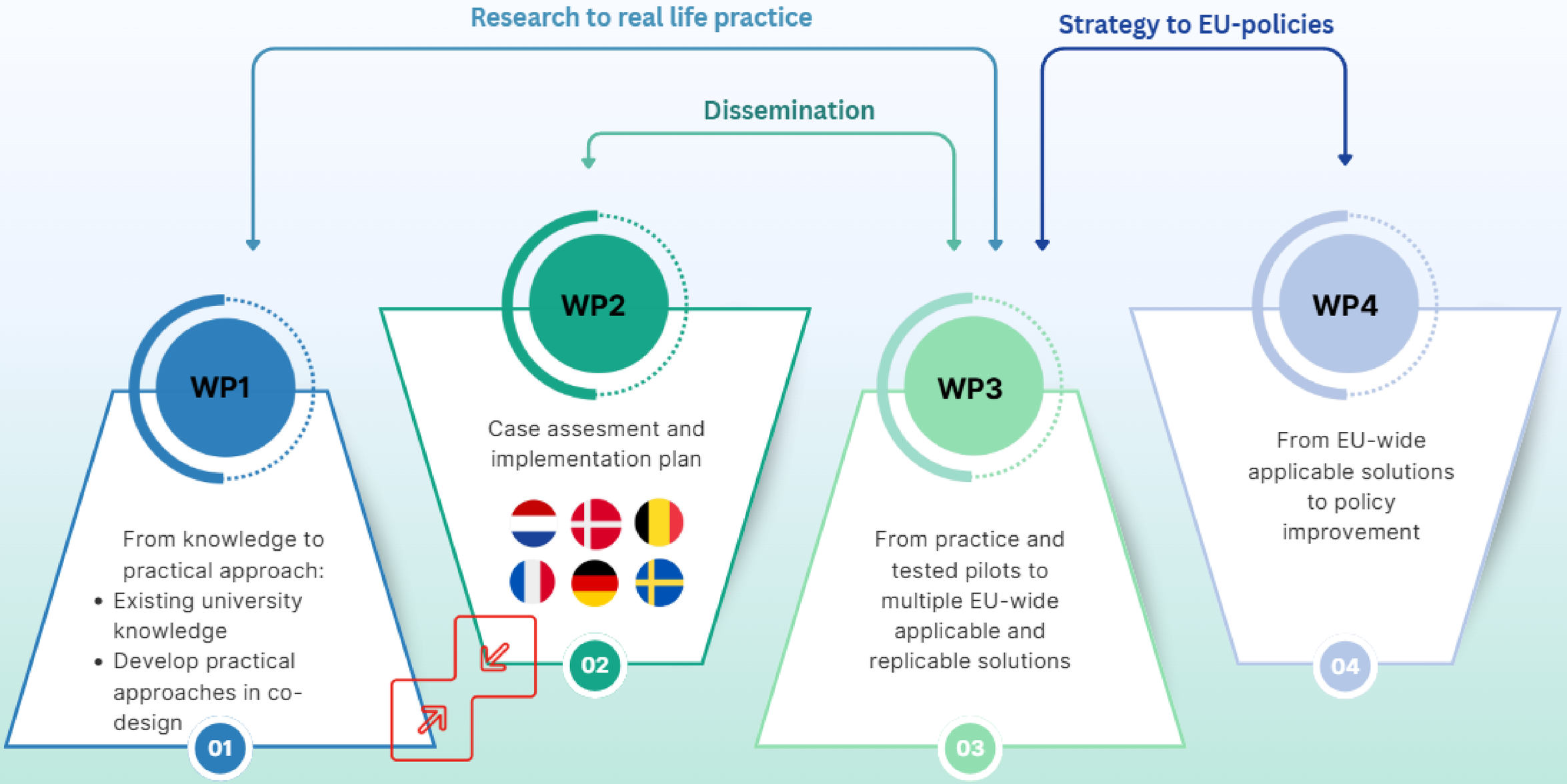
Digital Resilience in Storms:
How Cross-Border Cooperation Protects Europe from Flooding.

Introduction to the STORM_SAFE Project

How to maintain Ageing Critical Infrastructure in the fast-changing arena of Software Reliability

Interreg North Sea | Co-funded by the European Union

STORM_SAFE



Why it matters..

Interreg North Sea
Co-funded by the European Union
STORM_SAFE

Making reliable software is essential to safeguard critical infrastructures

CLIMATE CHANGE

Climate change fuels more frequent flood

PRESSURE

While intensified operations use and accelerates wear, raises costs, and increases the risk of failure

HUMAN FACTOR

The ageing workforce is causing a loss of critical knowledge in universities, contractors and infrastructure organizations

Model-based engineering

Interreg North Sea
Co-funded by the European Union
STORM_SAFE

How Other Engineering Disciplines Solve Problems

1.Create a model

2. Analyze to understand the system

3. Models guide the construction

What Often Happens in Software Engineering

1. Design and plan

2. Code and Implement

3. Test and Validate

modelling is not common in software engineering

If we model before building in engineering, why not in software?

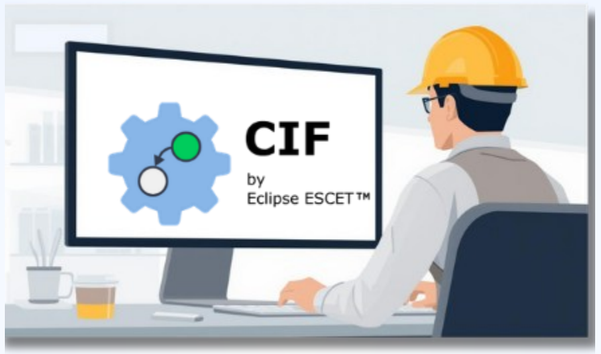
Formal methods explored in Storm_Safe

Interreg North Sea
Co-funded by the European Union
STORM_SAFE

Formal methods comprise model-based methods for software engineering founded in rigorous mathematics

Formal verification

Aim: given a model of the controller and a model of a property, check whether the model satisfies the property



Synthesis-Based Engineering

Aim: given a model of the plant and a model of the requirements, construct a model of a controller such that the controlled system satisfies all requirements

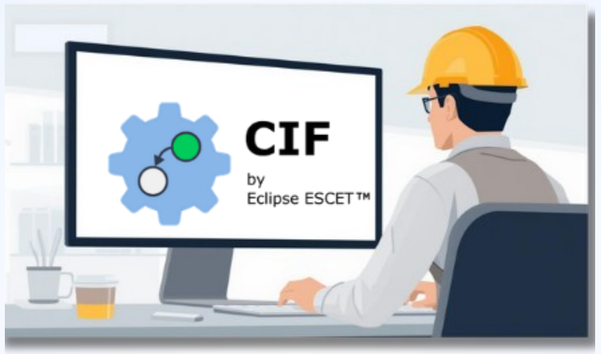
Formal methods explored in Storm_Safe

Interreg North Sea
Co-funded by the European Union
STORM_SAFE

Formal methods comprise model-based methods for software engineering founded in rigorous mathematics

Formal verification

Aim: given a model of the controller and a model of a property, check whether the model satisfies the property



Synthesis-Based Engineering

Aim: given a model of the plant and a model of the requirements, construct a model of a controller such that the controlled system satisfies all requirements

Steps of synthesis-based engineering

(Iterative process — especially between steps 2–5)

Interreg North Sea
Co-funded by the European Union
STORM_SAFE

Understand the system

01

Identify physical components

02

Model component functionality

What could the components do?

Specify & Design

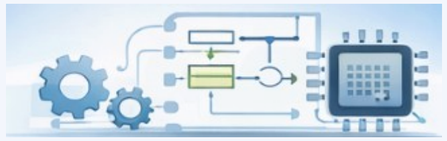
03

Model requirements

What should the components do?

04

Synthesize model of the supervisory controller



Verify & Implement

05

Validate the supervisory controller model

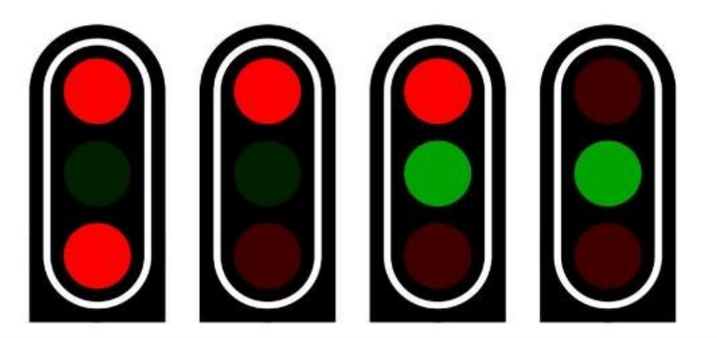
06

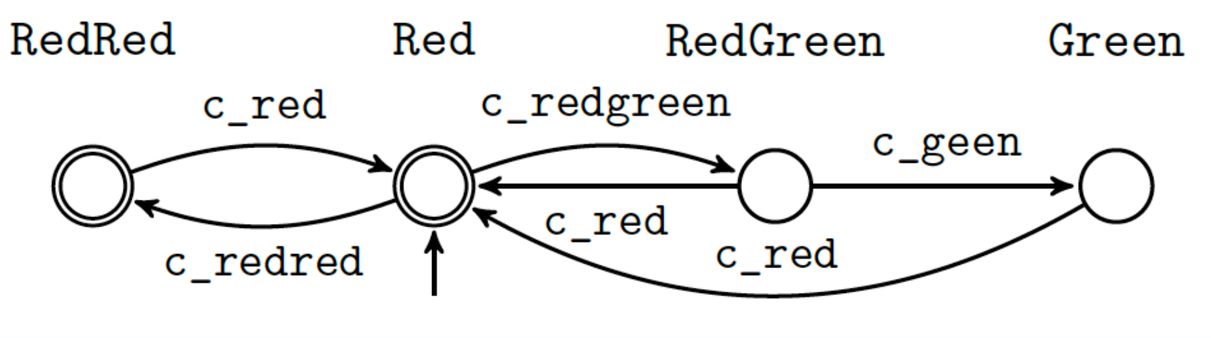
Generate code from the supervisory controller mode

Typical models in synthesis-based engineering

Models of the plants

Example of a ship traffic sign





Models of the requirements

”If an outgoing traffic light on the same lock head shows the ‘green’ aspect, the incoming traffic light may not show the ‘green’ aspect.”

$$\texttt{IncomingTL.Actuator.c\_green} \Longrightarrow \texttt{not OutgoingTL.SensorGreen.On}$$

Verification and validation of synthesized supervisor

Interreg North Sea
Co-funded by the European Union
STORM_SAFE

Verification

"Did I make the system right?"

Check if the system satisfies all **requirements**

Can be done using simulation, testing, or formal verification (for example using mCRL2)

Validation

"Did I make the right system?"

Check if the system satisfies all **needs**

Can be done using simulation, testing, or reviews

Typical situations in practice

1. Verification of requirements that are not modeled for SBE, using formal verification
2. Validation of wrongly or missing requirements, using simulation

Include fault behavior in models

Interreg North Sea
Co-funded by the European Union
STORM_SAFE

Murphy's law

"Anything that can go wrong, will go wrong."

Engineering systems are bound to break down

Fault-tolerant control

Faults can be modeled!

1

Identify which faults can occur

2

Automatically detect when the fault has occurred

Using diagnoser models

3

Specify the (additional) requirements for when a fault has occurred

PLC code generation

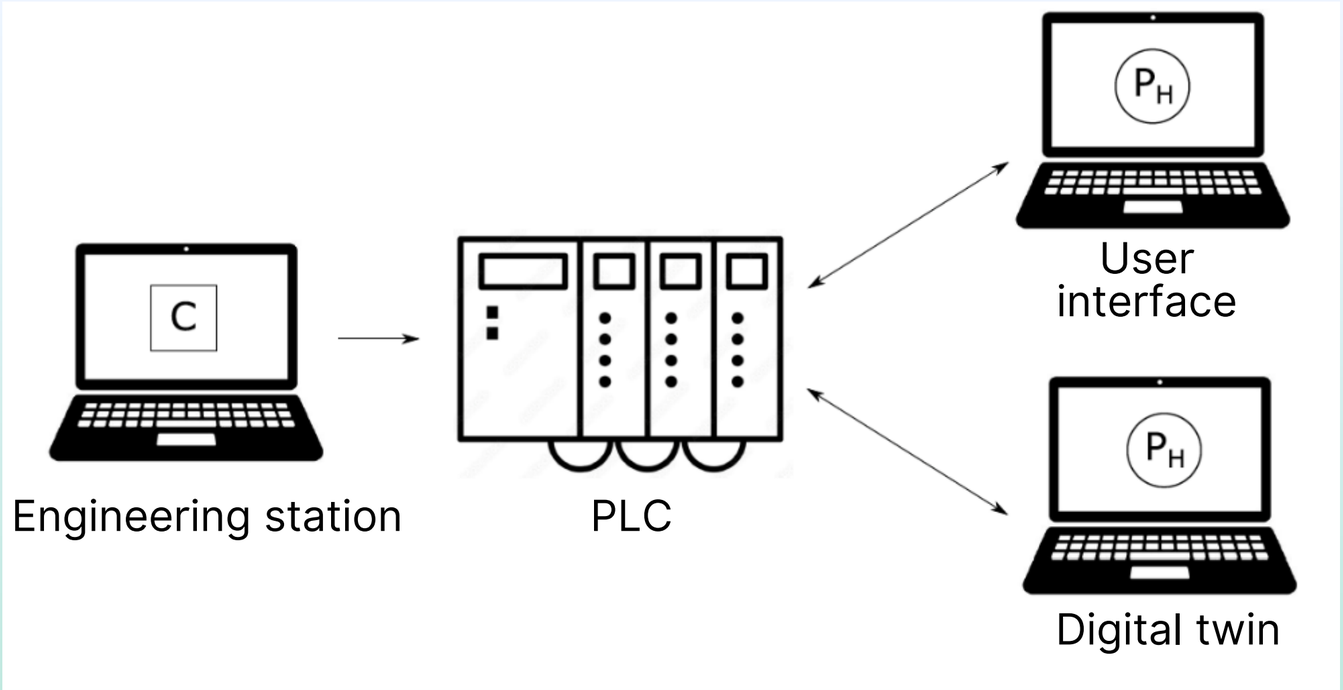
Code generation

PLC code can be generated from the synthesized supervisory controller model

- Works for the IEC 61131-3 standard, as well as several PLC brands (Siemens, ABB, Beckhoff)
- Focus on both correctness and traceability

Hardware-in-the-loop testing

Validate the PLC controller on the actual hardware



Synthesis-based engineering in summary

Interreg North Sea
Co-funded by the European Union
STORM_SAFE

Guarantee that the synthesized supervisory controller satisfies all requirements

Input:
1. Formal model of the plant (system to be controlled)
2. Formal model of the requirements

Specifying formal requirements requires detailed understanding
→ leads to clear consensus on requirements, no ambiguities

Can combine synthesis-based engineering and model-checking

Synthesis-based engineering helps identify and fix issues in the design phase

Formal methods explored in Storm_Safe

Interreg North Sea
Co-funded by the European Union
STORM_SAFE

Formal methods comprise model-based methods for software engineering founded in rigorous mathematics

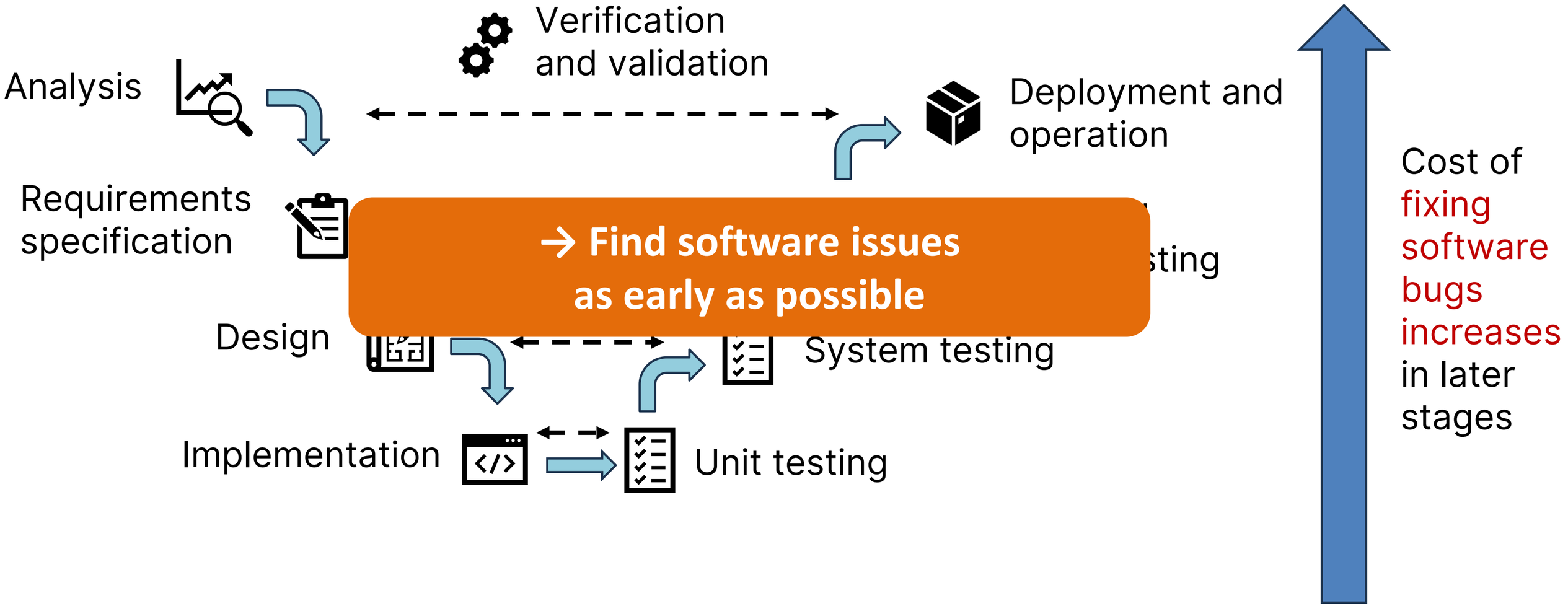
Formal verification

Aim: given a model of the controller and a model of a property, check whether the model satisfies the property

Synthesis-Based Engineering

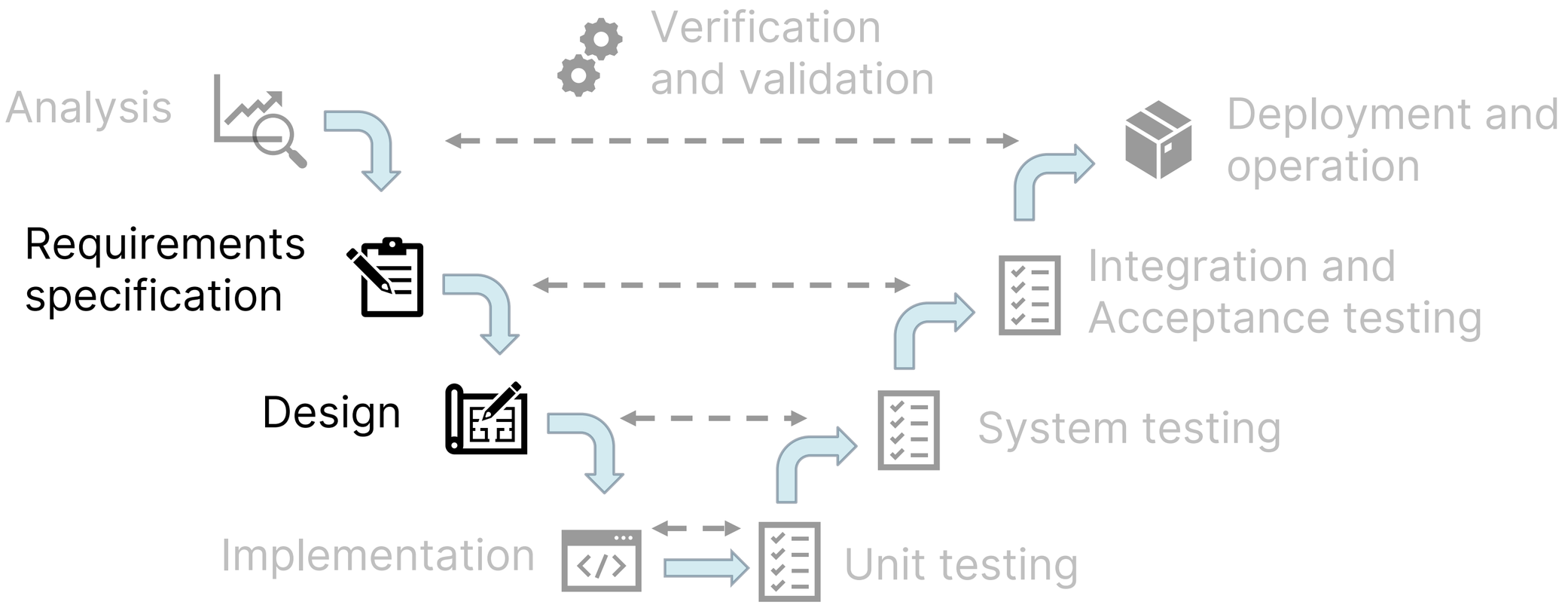
Aim: given a model of the plant and a model of the requirements, construct a model of a controller such that the controlled system satisfies all requirements

Software design process

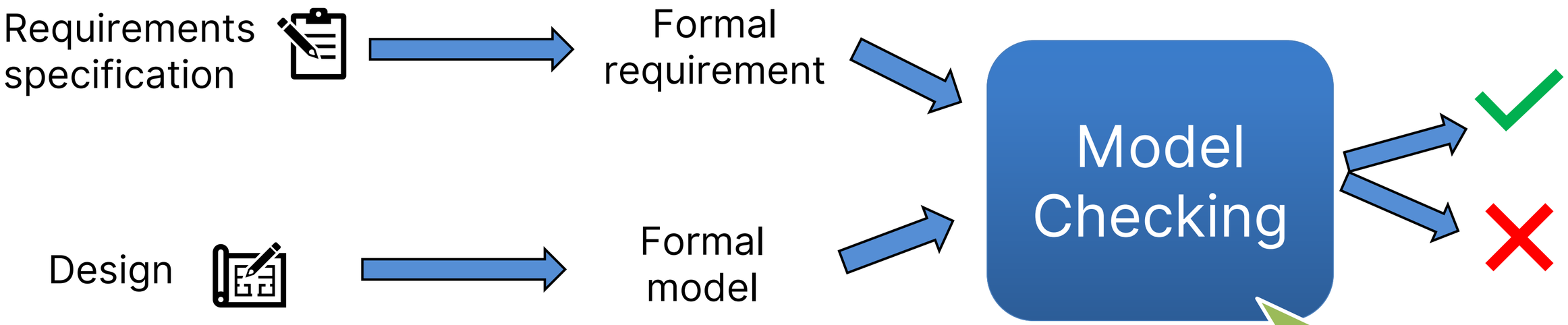


Software design process

Interreg North Sea
Co-funded by the European Union
STORM_SAFE



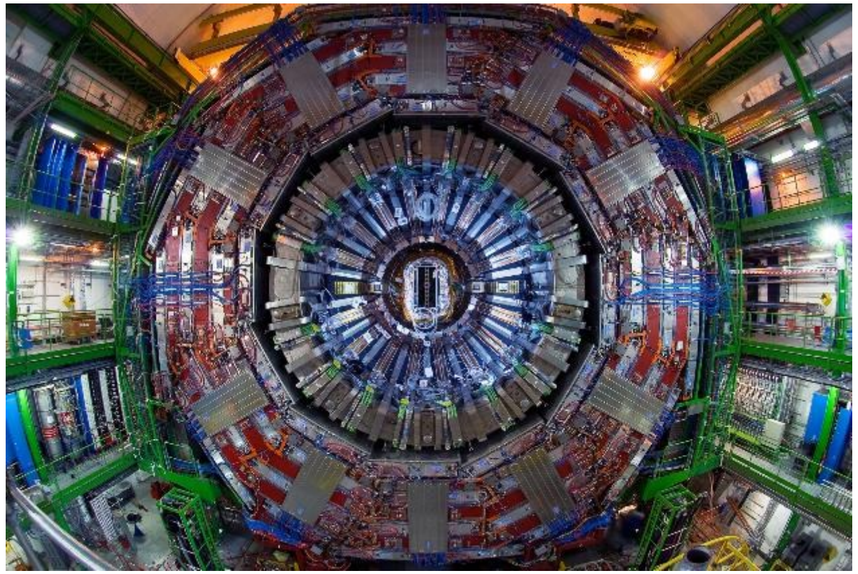
Model checking



- Proves/disproves the correctness of the model
- Counterexample shows reason for violation
- Fully automated

Model checking: case studies

Interreg North Sea
Co-funded by the European Union
STORM_SAFE



https://doi.org/10.1016/j.scico.2012.11.009

Found unresponsiveness in control software of CERN's CMS detector

Architecture and Hardware Contributed articles

How Amazon Web Services Uses Formal Methods

Engineers use TLA+ to prevent serious but subtle bugs from reaching production.

By Chris Newcombe, Tim Rath, Fan Zhang, Bogdan Munteanu, Marc Brooker, and Michael Deardeuff

Posted Apr 1 2015

Share Download PDF Print Join the Discussion View in the ACM Digital Library

https://cacm.acm.org/research/how-amazon-web-services-uses-formal-methods/

Found subtle bugs in DynamoDB of AWS

Verified major control system of Maeslant storm surge barrier

https://doi.org/10.1007/978-3-032-00942-5_12

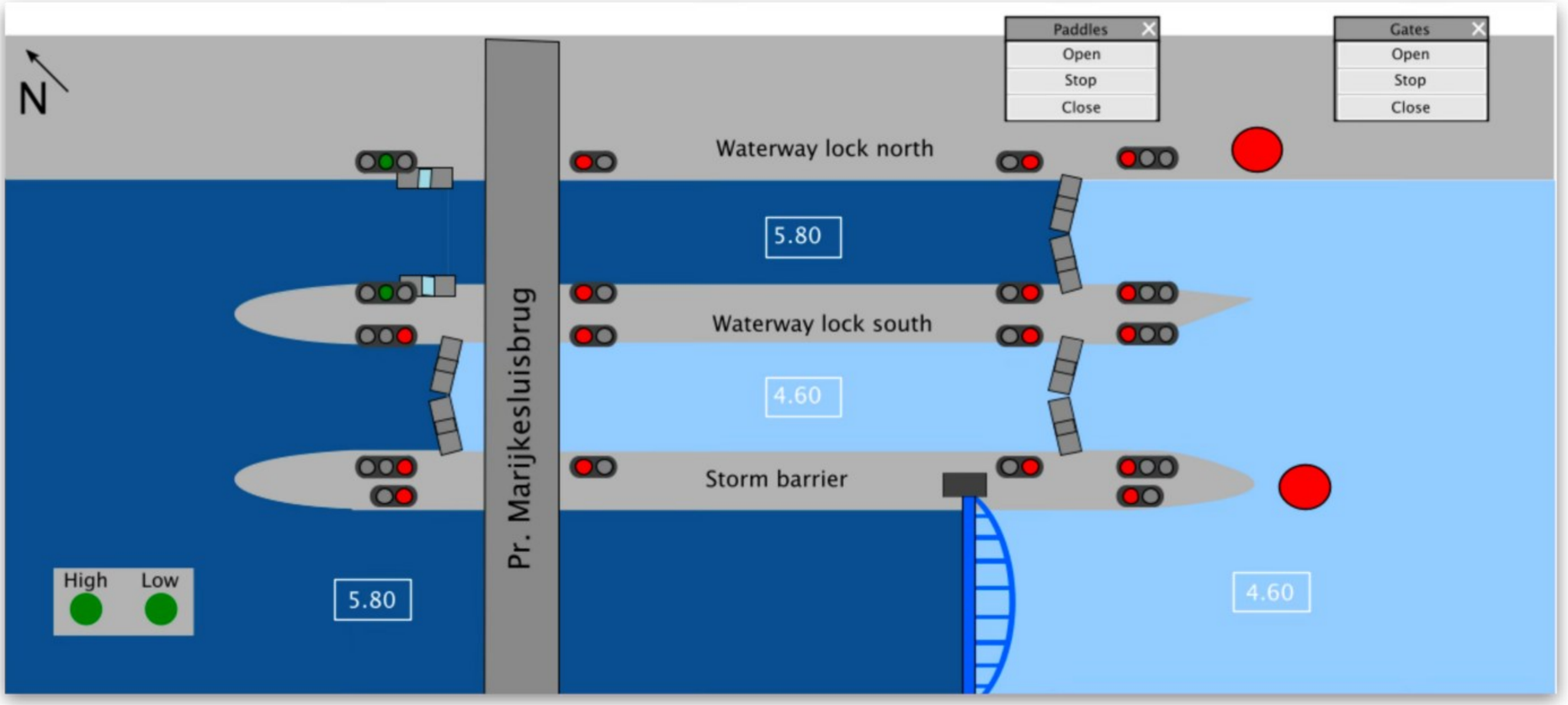
Example: Princess Marijke lock complex

- Lock complex in NL on Amsterdam-Rhine canal
- 2 water locks for ships
- 1 barrier to prevent flooding

Example: Princess Marijke lock complex

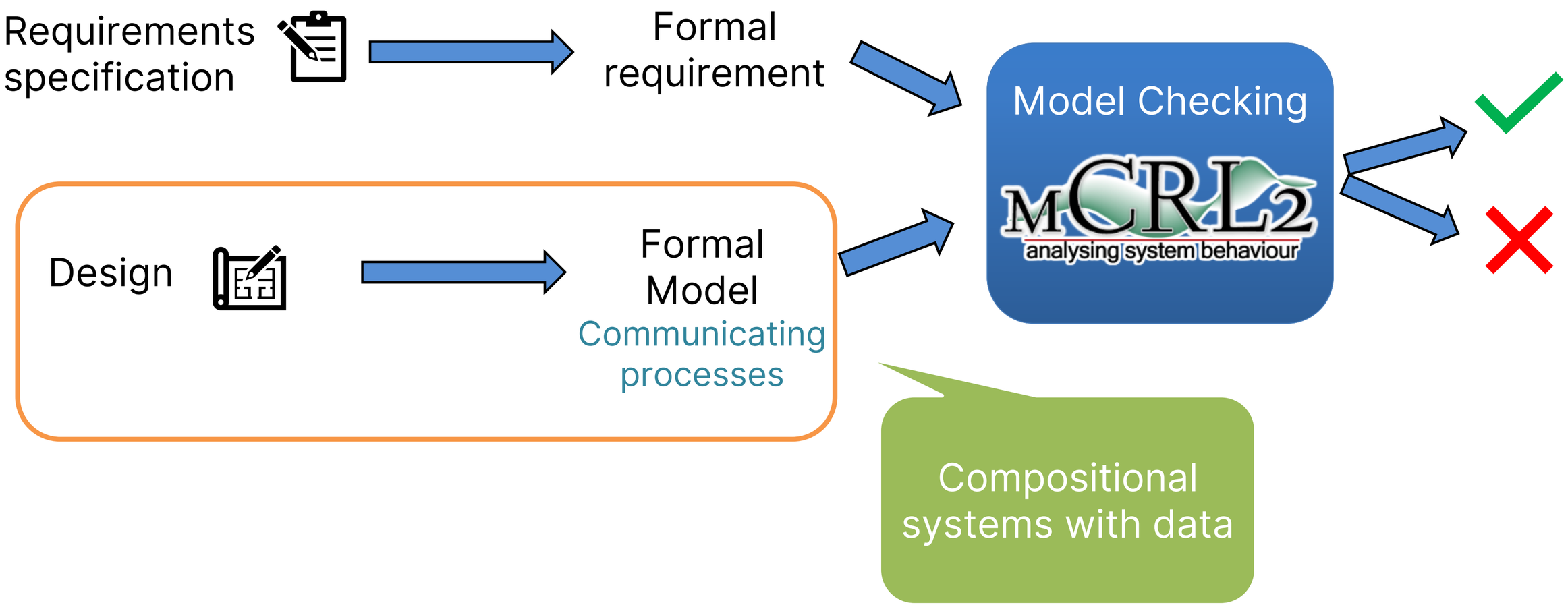
Components:

- Gates with paddles
- Sensors:
  - Gate
  - Paddles
  - water level
- Traffic lights:
  - Entering
  - leaving
- Barrier
- Controller



[Reijnen, Verbakel, van de Mortel-Fronczak, Rooda, CCTA 2019]

mCRL2


Requirements specification
Formal requirement
Model Checking
mCRL2
analysing system behaviour
Design
Formal Model
Communicating processes
Compositional systems with data

mCRL2 model

- Single process: the controller
- Interactions with components through sensor/actuator actions
  - No explicit processes for the components (traffic lights, gates, etc)
  → no assumptions how components behave, could be faulty
- mCRL2 model: 400 lines

```
% Paddle control.
sum l:Lock, s:StreamSide. PaddleCommand(l, s, command_open).
    (GateStatus(triple(l,opposite(s), east)) == closed &&
     GateStatus(triple(l,opposite(s), west)) == closed &&
     PaddleStatus(triple(l,opposite(s), east)) == closed &&
     PaddleStatus(triple(l,opposite(s), west)) == closed &&
     !(l in LocksInEmergency))
    -> PaddleActuator(l, s, east, do_open).
       PaddleActuator(l, s, west, do_open).
       Controller(PaddleStatus=PaddleStatus[triple(l,s,east)->opening]
```

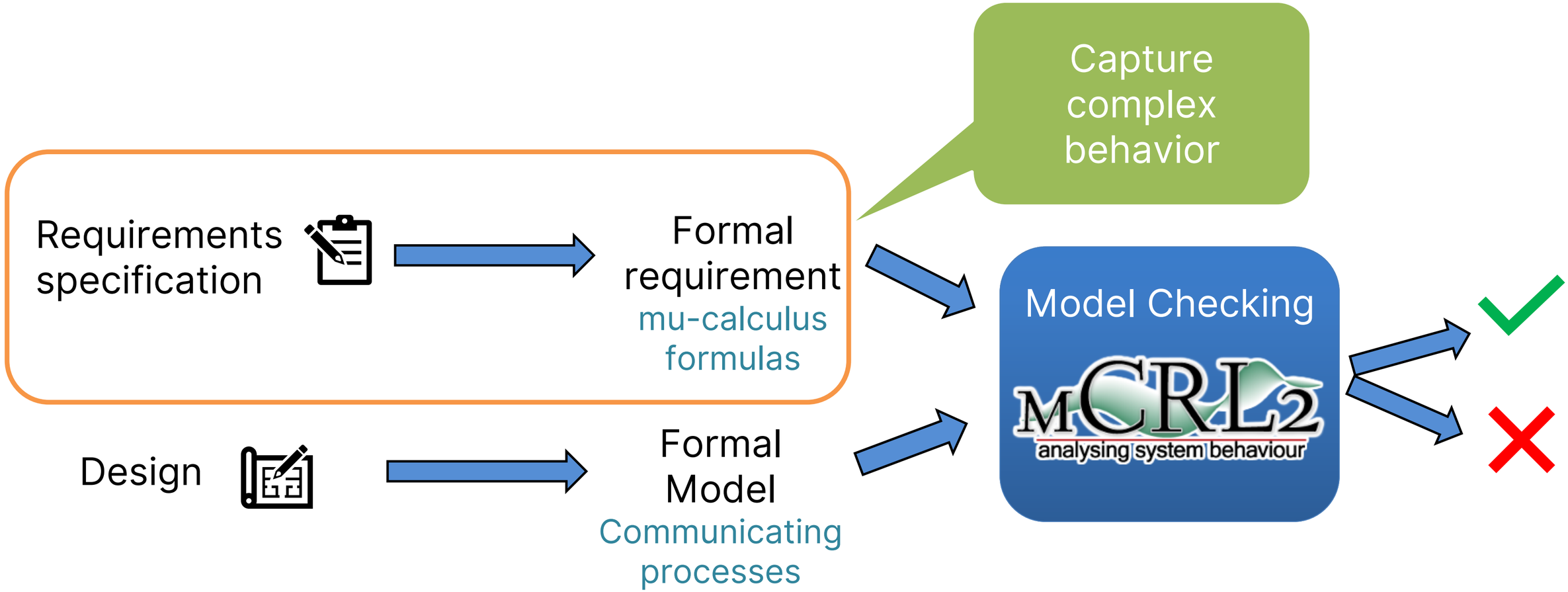
Received command to open paddle

Check conditions

Command to actuators

Update internal state

mCRL2


Capture complex behavior
Requirements specification
Formal requirement
mu-calculus formulas
Design
Formal Model
Communicating processes
Model Checking
MCRL2
analysing system behaviour

Requirements as mu-calculus formulas

Interreg North Sea
Co-funded by the European Union
STORM_SAFE

Liveness requirements:

"If instruction to open the gate is given to controller, both gates will open"

Safety requirements:

"If the paddles are not closed at one stream side of a sluice, the gates at the other stream side cannot be opened."

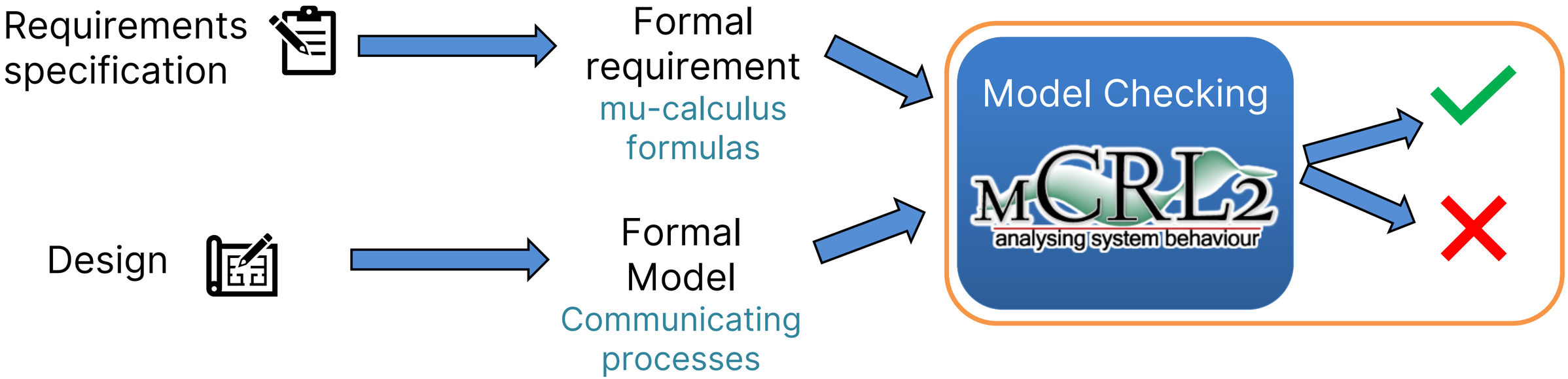
```
forall l:Lock, s:StreamSide, o:Orientation.
[true*.
  (PaddleSensor(l,s,o,sense_open)||PaddleSensor(l,s,o,sense_intermediate)||
        PaddleSensor(l,s,o,fail_position)||PaddleActuator(l,s,o,do_open)).

  !(PaddleSensor(l,s,o,sense_closed))*.

  exists o':Orientation.GateActuator(l,opposite(s),o',do_open)]false
…
```

Paddle was open

Paddle sensor did not indicate closed

Opening the gate is not possible

mCRL2


Requirements specification
Formal requirement
mu-calculus formulas
Design
Formal Model
Communicating processes
Model Checking
mCRL2
analysing system behaviour

Verification

Interreg North Sea
Co-funded by the European Union
STORM_SAFE

- Formalized 53 requirements
- Successfully verified all requirements
- Time: between 4 minutes and 31 hours
- State space: 1.6 $\cdot 10^{18}$ states

Findings

- Missing safety requirement for emergency stop

  "In emergency mode the barrier cannot be instructed to open or close."

- Unclear requirements

  Can lights be switched back to green color in emergency mode?

- Failures of traffic lights or water level sensor could prevent closing of barrier

  → Explicitly incorporate failures:

  "At any time, the barrier can be instructed to close
  and then closes successfully with a probability of 0.9999."

Model checking

- Guarantee that model satisfies requirements
- Input:
  1. Formal model of system
  2. Formalization of requirements
- Specifying formal requirements requires detailed understanding
  → leads to clear consensus on requirements, no ambiguities
- Can combine model checking and synthesis-based engineering

→ Model checking helps identify and fix issues in the design phase

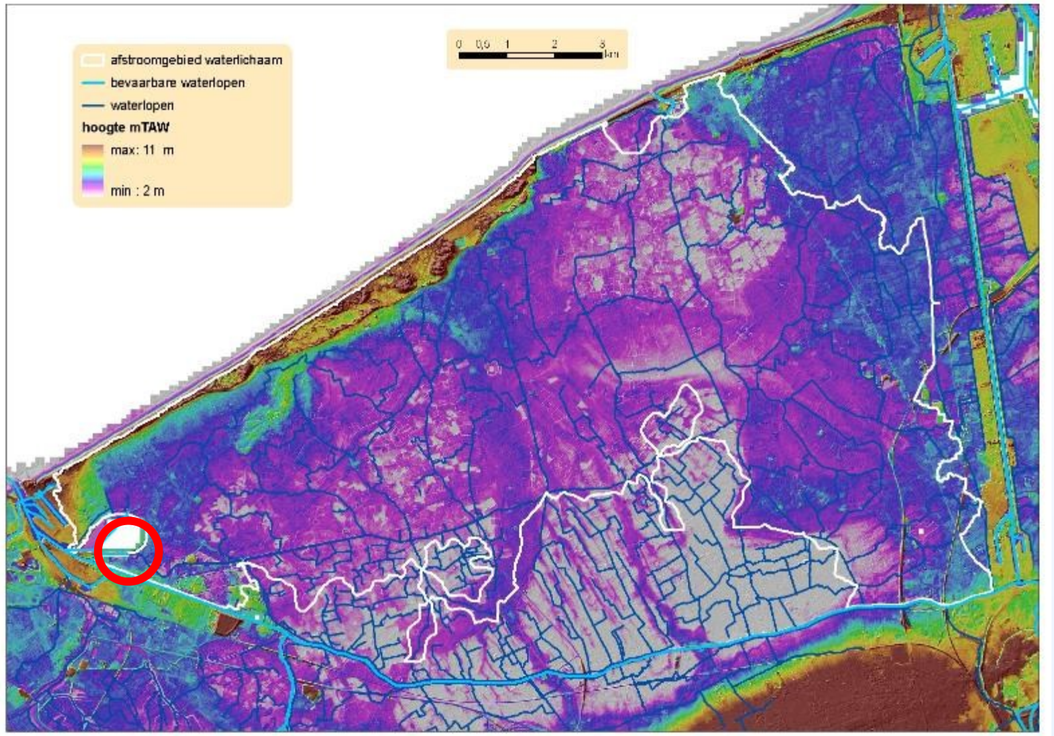
- Rich ecosystem and expertise within STORM_SAFE

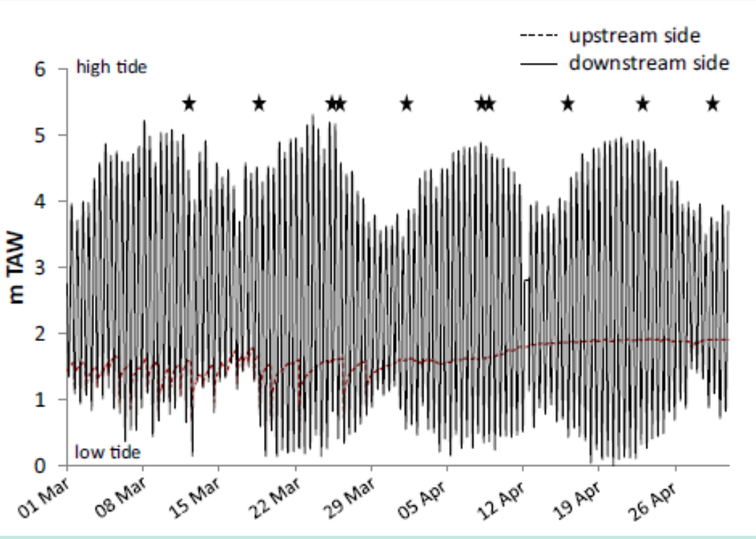
mCRL2 analysing system behaviour Storm UPPAAL Modest

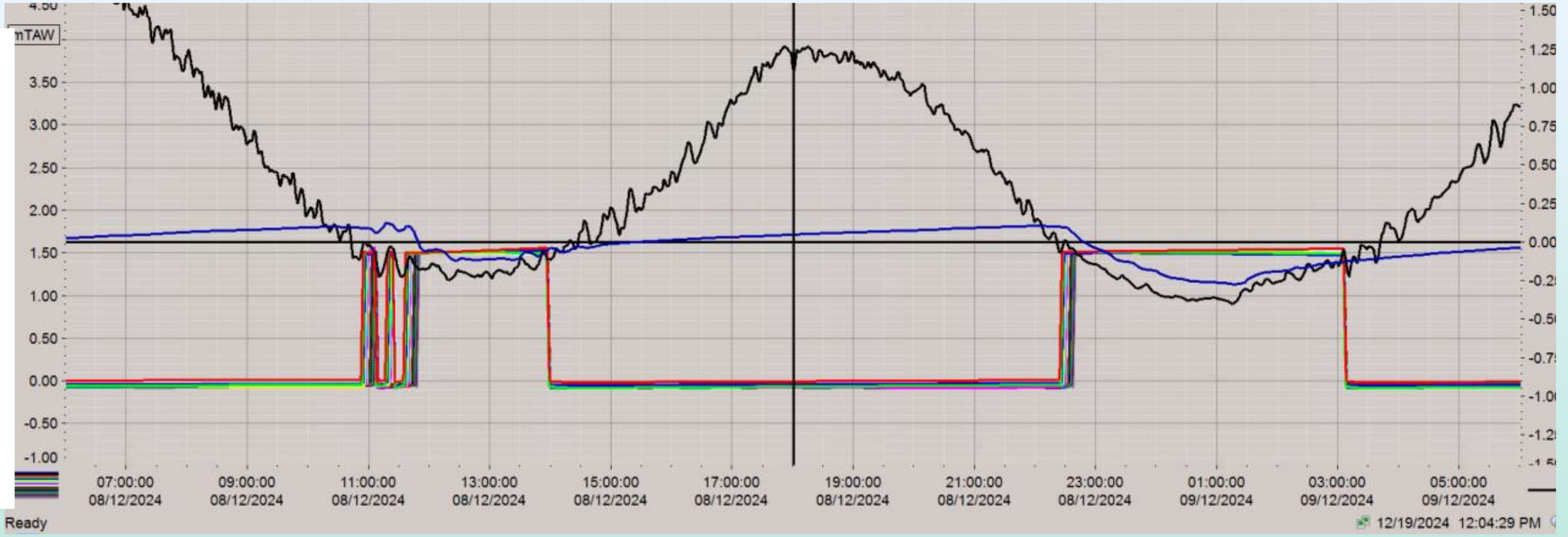
Focus session: Maertenssas pilot

Maarten Goegebeur
Martijn Goorden

Interreg North Sea
Co-funded by the European Union
STORM_SAFE
VLAAMSE MILIEUMAATSCHAPPIJ
TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

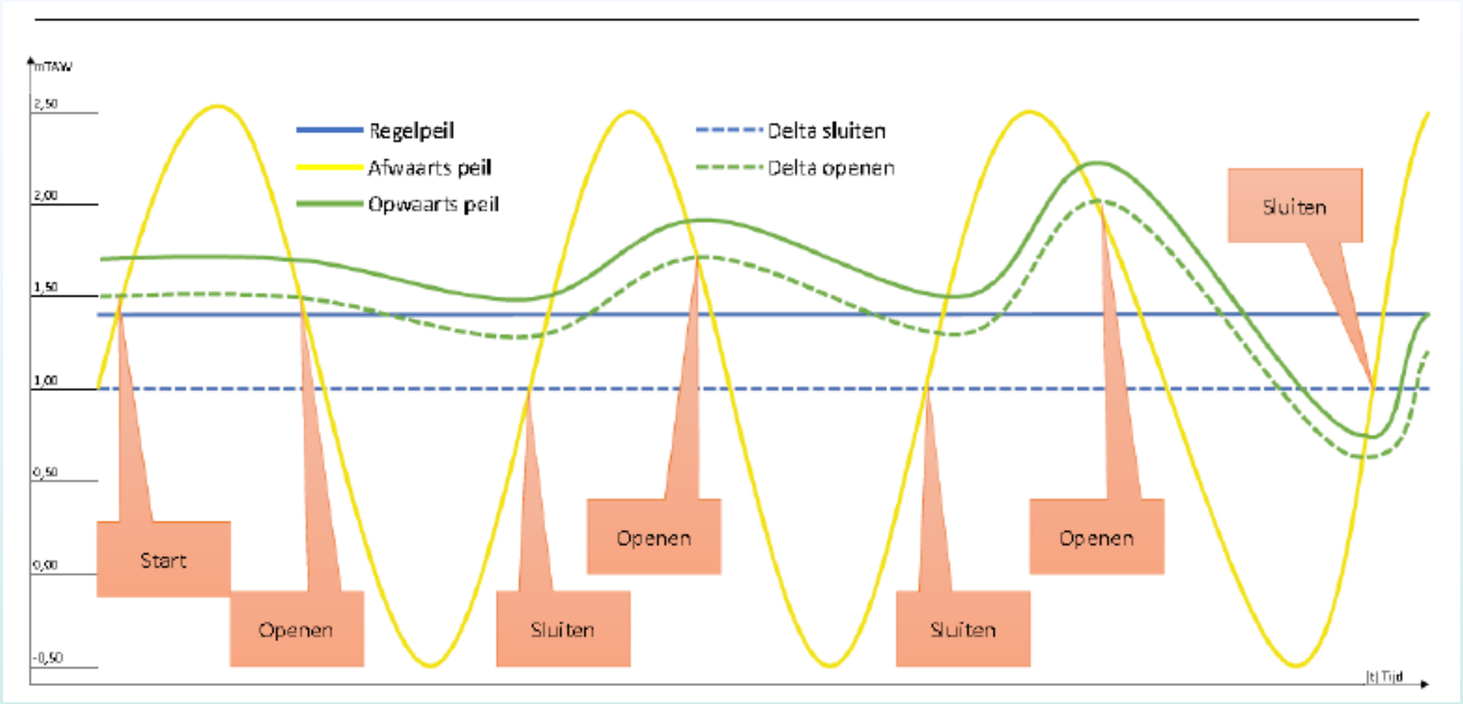

afstroomgebied waterlichaam
bevaarbare waterlopen
waterlopen
hoogte mTAW
max: 11 m
min : 2 m


upstream side
downstream side
high tide
low tide
m TAW
01 Mar
08 Mar
15 Mar
22 Mar
29 Mar
05 Apr
12 Apr
19 Apr
26 Apr

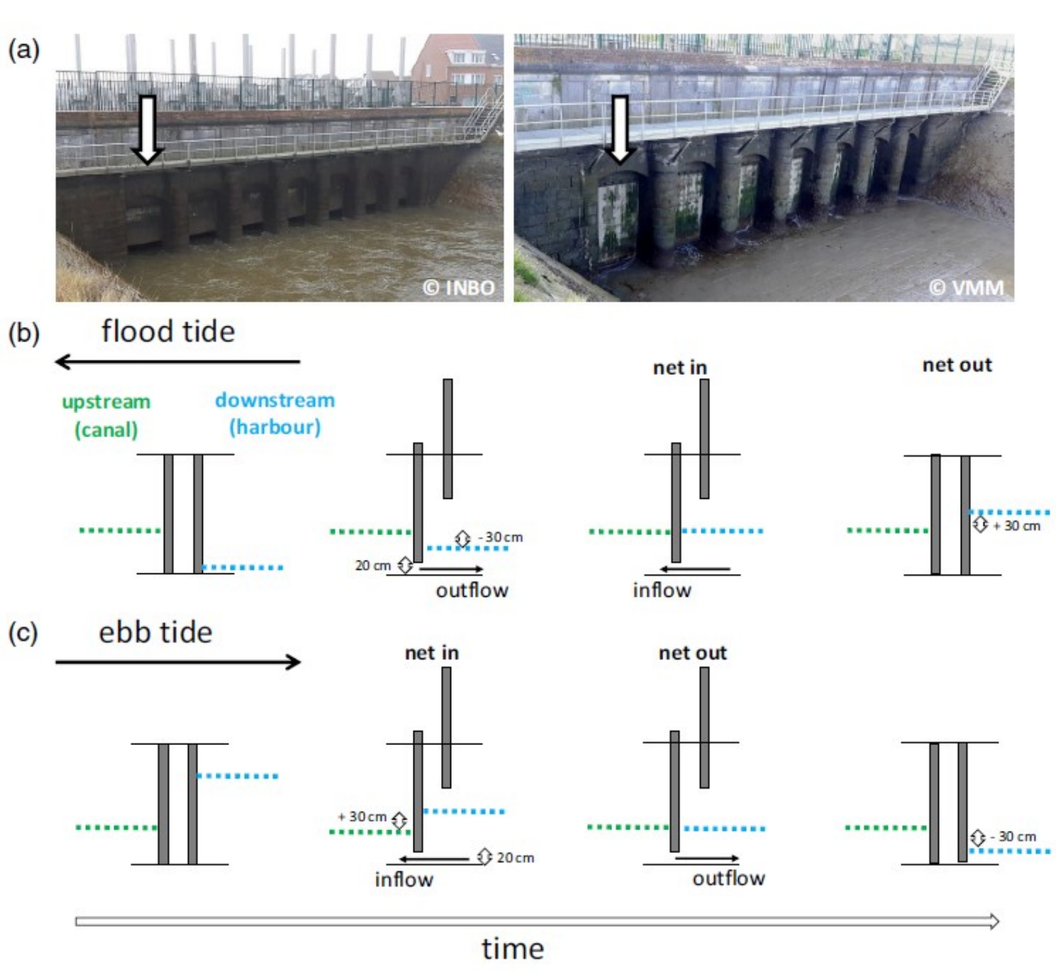

mTAW
07:00:00 08/12/2024
09:00:00 08/12/2024
11:00:00 08/12/2024
13:00:00 08/12/2024
15:00:00 08/12/2024
17:00:00 08/12/2024
19:00:00 08/12/2024
21:00:00 08/12/2024
23:00:00 08/12/2024
01:00:00 09/12/2024
03:00:00 09/12/2024
05:00:00 09/12/2024
Ready
12/19/2024 12:04:29 PM

Desired system control

- Local – Remote, and Manual – Automatic
- Double function tube 7 (gates 13 & 14): besides gravity discharge, also fish passage
- Variable parameters for automatic control
- Externe data via internet
- Priority: form a barrier against high water levels



Glass eel migration



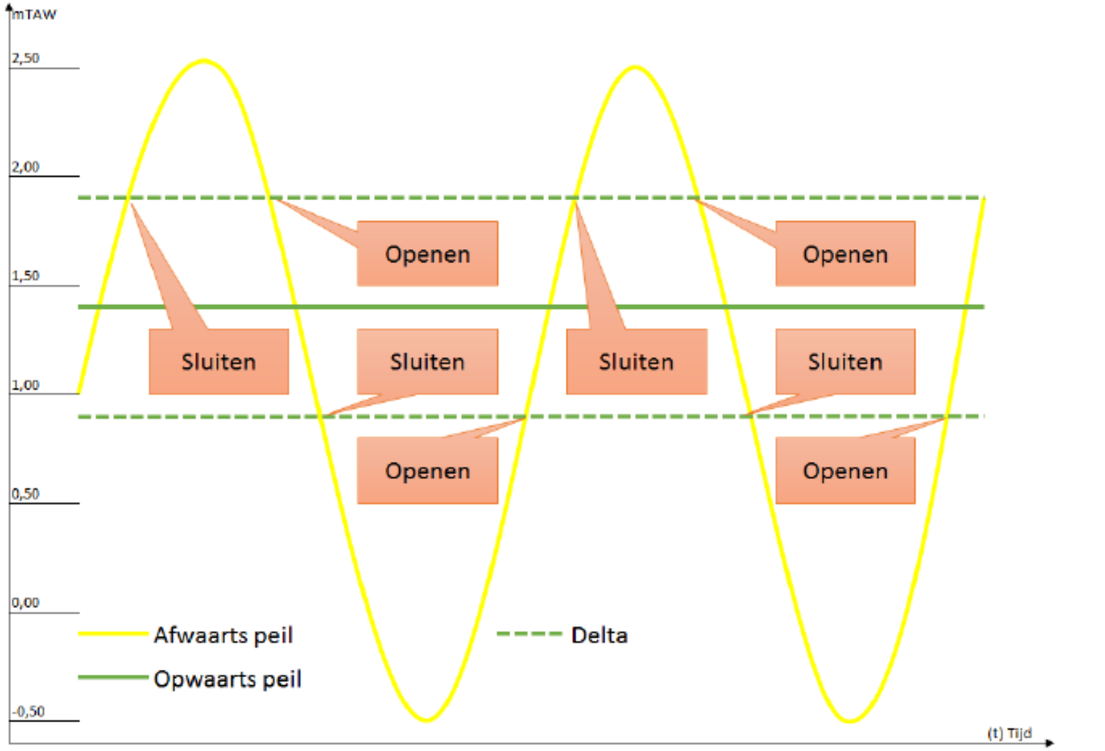


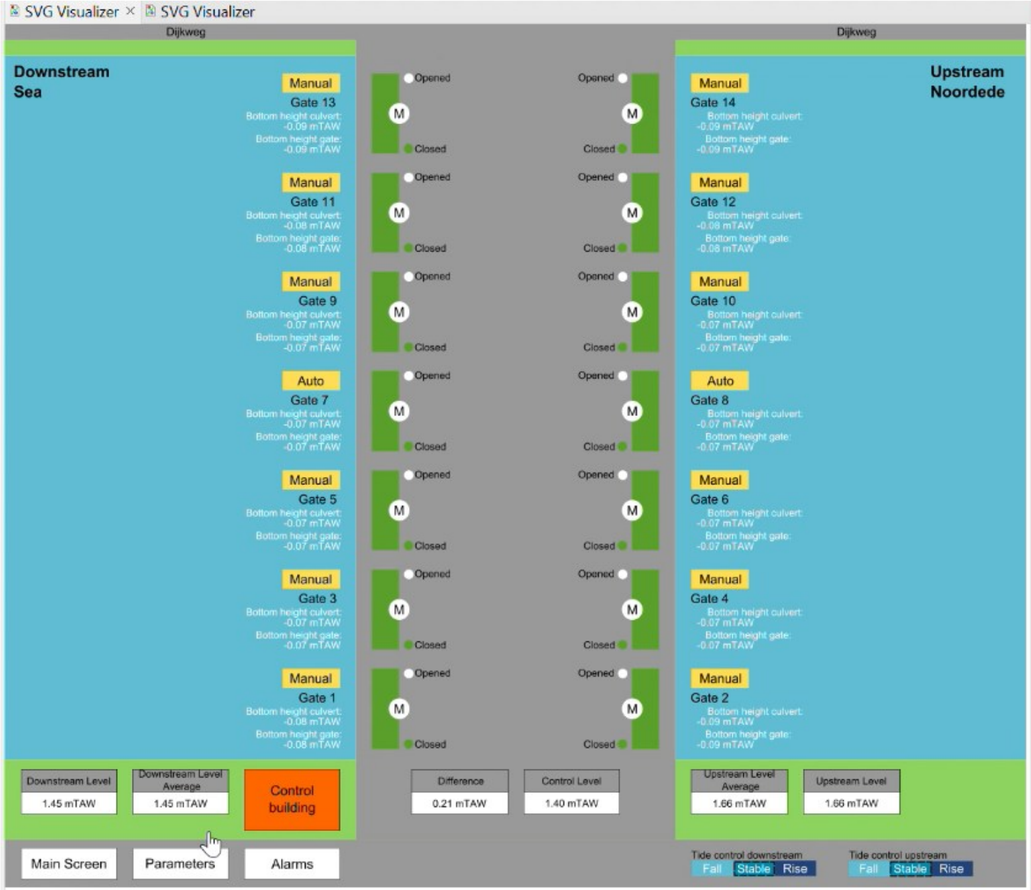
TABLE 1 Specifics and results of glass eel catches during 20 inflow events at the Maertensas tidal barrier (Ostend, Belgium)

Date	Tide	Prior discharge event	Day/night	Inflow duration (min)	N glass eels	N glass eels min inflow^{-1}
1/03/2019[a]	Ebb	Y	D	17	8	0.5
11/03/2019	Ebb	Y	D	20	56	3
11/03/2019	Flood		D	27	31	1
18/03/2019	Ebb	Y	D	4	1	0.3
18/03/2019	Flood		N	23	2,167	94
25/03/2019	Ebb	N	D	10	9	1
25/03/2019	Flood		D	20	26	1
25/03/2019	Ebb	N	N	8	309	39
26/03/2019	Flood		N	21	3,750	179
1/04/2019	Ebb	N	D	16	17	1
1/04/2019	Flood		N	22	3,827	174
8/04/2019	Ebb	N	D	12	21	2
8/04/2019	Flood		D	22	69	3

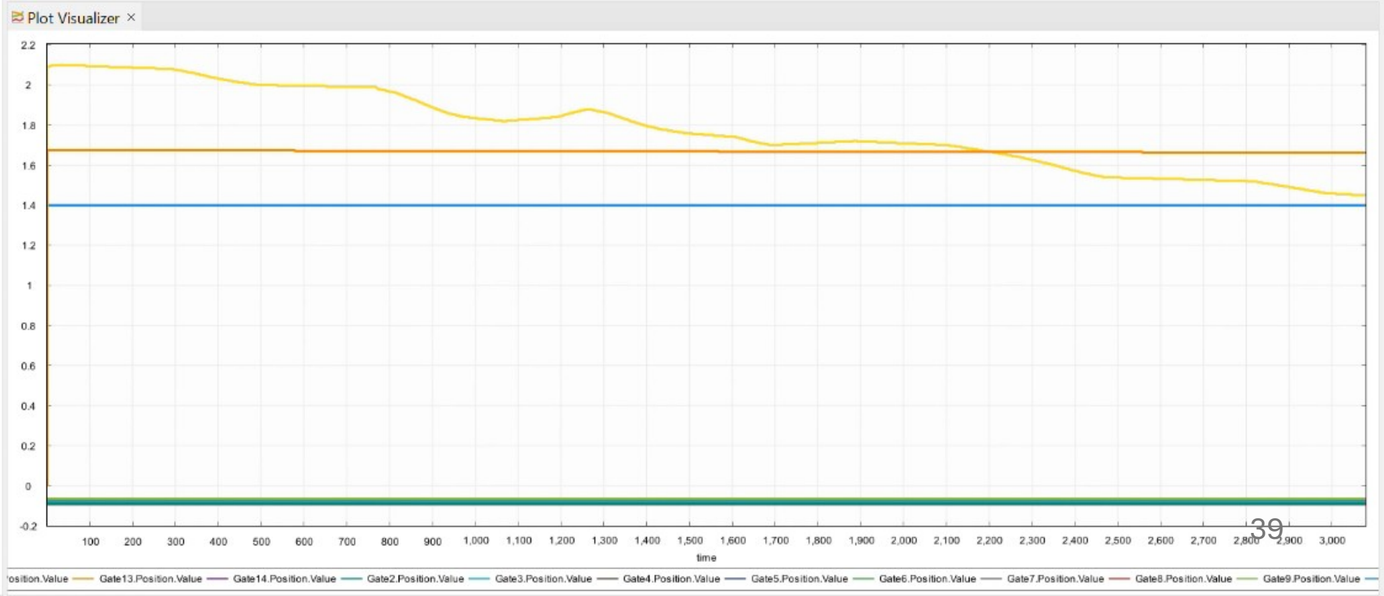
https://onlinelibrary.wiley.com/doi/10.1002/rra.4088

Problem statement of the Maertenssas sluice

- Complicated control logic and further optimization makes it complex
- PLC hardware upgrade after failure -> software changes
- Climate changes -> software changes
- Standardization needs


SVG Visualizer
SVG Visualizer
Dijkweg
Dijkweg
Downstream
Sea
Upstream
Noordede
Manual
Gate 13
Bottom height culvert: -0.09 mTAW
Bottom height gate: -0.09 mTAW
Manual
Gate 11
Bottom height culvert: -0.08 mTAW
Bottom height gate: -0.08 mTAW
Manual
Gate 9
Bottom height culvert: -0.07 mTAW
Bottom height gate: -0.07 mTAW
Auto
Gate 7
Bottom height culvert: -0.07 mTAW
Bottom height gate: -0.07 mTAW
Manual
Gate 5
Bottom height culvert: -0.07 mTAW
Bottom height gate: -0.07 mTAW
Manual
Gate 3
Bottom height culvert: -0.07 mTAW
Bottom height gate: -0.07 mTAW
Manual
Gate 1
Bottom height culvert: -0.08 mTAW
Bottom height gate: -0.08 mTAW
Opened
Closed
M
Manual
Gate 14
Bottom height culvert: -0.09 mTAW
Bottom height gate: -0.09 mTAW
Manual
Gate 12
Bottom height culvert: -0.08 mTAW
Bottom height gate: -0.08 mTAW
Manual
Gate 10
Bottom height culvert: -0.07 mTAW
Bottom height gate: -0.07 mTAW
Auto
Gate 8
Bottom height culvert: -0.07 mTAW
Bottom height gate: -0.07 mTAW
Manual
Gate 6
Bottom height culvert: -0.07 mTAW
Bottom height gate: -0.07 mTAW
Manual
Gate 4
Bottom height culvert: -0.07 mTAW
Bottom height gate: -0.07 mTAW
Manual
Gate 2
Bottom height culvert: -0.09 mTAW
Bottom height gate: -0.09 mTAW
Downstream Level
1.45 mTAW
Downstream Level Average
1.45 mTAW
Control building
Difference
0.21 mTAW
Control Level
1.40 mTAW
Upstream Level Average
1.66 mTAW
Upstream Level
1.66 mTAW
Main Screen
Parameters
Alarms
Tide control downstream
Fall
Stable
Rise
Tide control upstream
Fall
Stable
Rise


SVG Visualizer
Downstream water level
1.45 mTAW
Upstream water level
1.66 mTAW
Emergency button
Reset
Downstream
Culvert 1
Culvert 2
Culvert 3
Culvert 4
Culvert 5
Culvert 6
Culvert 7
Bottom height gate 1
-0.08 mTAW
Bottom height gate 3
-0.07 mTAW
Bottom height gate 5
-0.07 mTAW
Bottom height gate 7
-0.07 mTAW
Bottom height gate 9
-0.07 mTAW
Bottom height gate 11
-0.08 mTAW
Bottom height gate 13
-0.09 mTAW
Manual
Auto
Open
Opened
Stop
Alarm
Close
Closed
Bottom height gate 2
-0.09 mTAW
Bottom height gate 4
-0.07 mTAW
Bottom height gate 6
-0.07 mTAW
Bottom height gate 8
-0.07 mTAW
Bottom height gate 10
-0.07 mTAW
Bottom height gate 12
-0.08 mTAW
Bottom height gate 14
-0.09 mTAW
Gravitational discharge
Fish passage
Upstream
Maintenance switch gate 1
Maintenance switch gate 3
Maintenance switch gate 5
Maintenance switch gate 7
Maintenance switch gate 9
Maintenance switch gate 11
Maintenance switch gate 13
Maintenance switch gate 2
Maintenance switch gate 4
Maintenance switch gate 6
Maintenance switch gate 8
Maintenance switch gate 10
Maintenance switch gate 12
Maintenance switch gate 14
On
Off


Plot Visualizer
2.2
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
100
200
300
400
500
600
700
800
900
1,000
1,100
1,200
1,300
1,400
1,500
1,600
1,700
1,800
1,900
2,000
2,100
2,200
2,300
2,400
2,500
2,600
2,700
2,800
2,900
3,000
time
osition.Value
Gate13.Position.Value
Gate14.Position.Value
Gate2.Position.Value
Gate3.Position.Value
Gate4.Position.Value
Gate5.Position.Value
Gate6.Position.Value
Gate7.Position.Value
Gate8.Position.Value
Gate9.Position.Value

Steps of synthesis-based engineering

(Iterative process — especially between steps 2–5)

Interreg North Sea
Co-funded by the European Union
STORM_SAFE

Understand the system

01 Identify physical components

02 Model component functionality

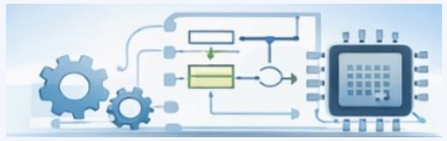
What could the components do?

Specify & Design

03 Model requirements

What should the components do?

04 Synthesize model of the supervisory controller



Verify & Implement

05 Validate the supervisory controller model

06 Generate code from the supervisory controller mode

Plant modelling: importance of decomposition

Advantages of decomposing the model

- Alignment between model and documented decomposition
- Better understandability
- Reusability of model templates
- Advanced synthesis techniques can benefit from good decompositions

Decompose

Extracting the control requirements

Different sources of control requirements

- VMM documentation
- PLC code
- Stakeholder and system expert interviews
- Experience of the modeler

Example from the PLC code

“To switch to fish passage mode or gravitational discharge mode, gate 13 and gate 14 must be closed.”

This translates to the requirement model

```
requirement {ModeCulvert7.c_to_fish_passage, ModeCulvert7.c_to_grav_discharge}
    needs Gate13.LowerLimitSwitch.On and Gate14.LowerLimitSwitch.On;
```

Extracting the control requirements

Schuifvolgorde openen van de schuiven: 1 – 2 – 3 – 4 – 5 – 6 – 7 - 8 – 9 – 10 – 11 – 12 – [13] – [14].
Schuifvolgorde sluiten van de schuiven : 2 – 4 – 6 – 8 – 10 – 12 – [14] – 1 - 3 – 5 – 7 – 9 – 11 – [13].

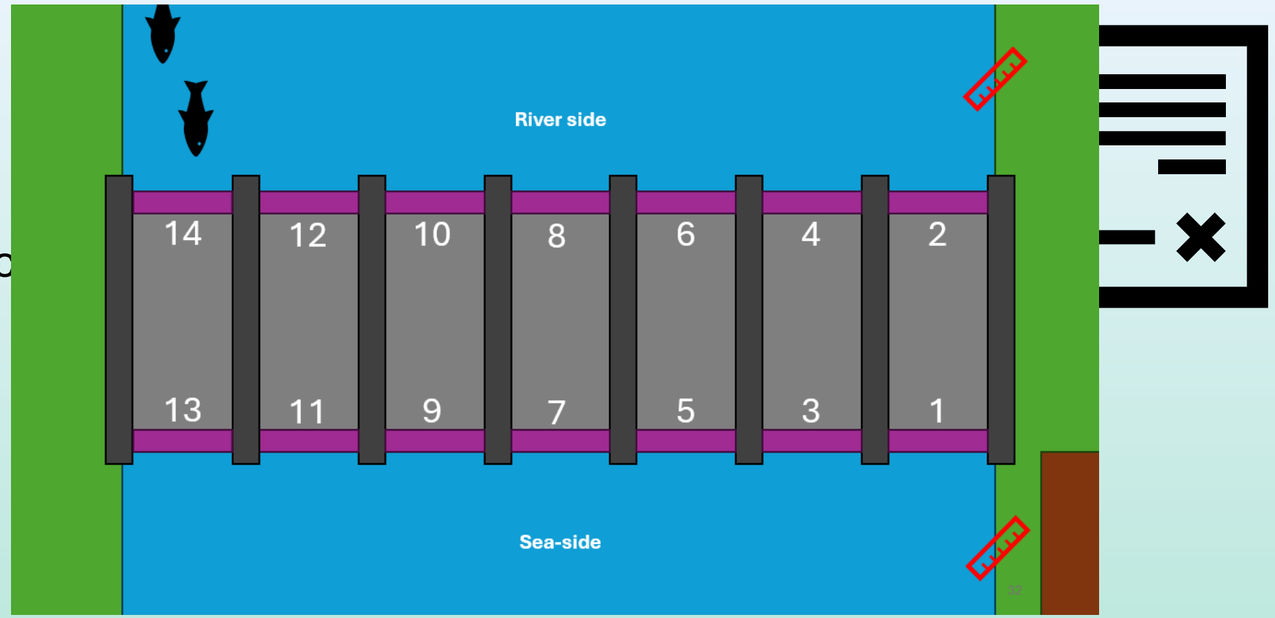
Deze schuifvolgorde van aansturen van schuiven na elkaar met een onderlinge vertraging die instelbaar is via het parameterscherm heeft als doel om piekstromen die aan het net worden onttrokken te vermijden.

HOW

WHAT/WHY

“Gates

“Each c



Synthesizing the supervisor

Interreg North Sea
Co-funded by the European Union
STORM_SAFE

Synthesis input

In total, the Maertenssas sluice model consists of

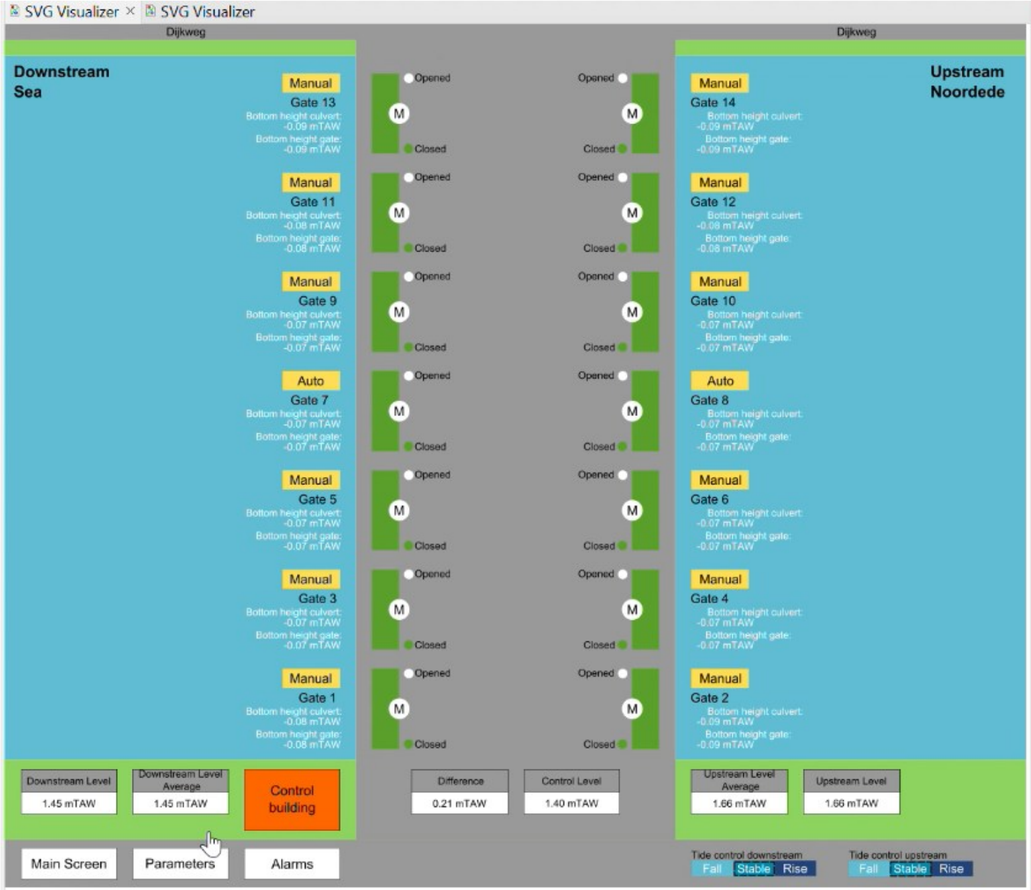
- 370 plant models
- 481 requirement models

Synthesis results

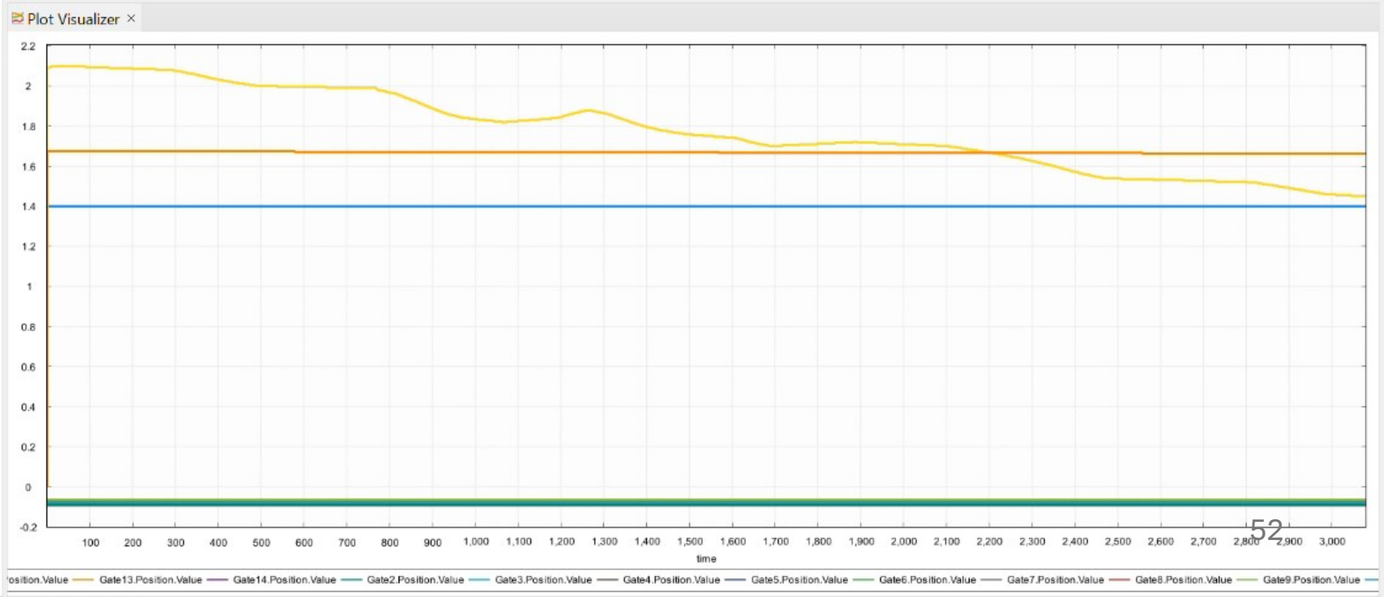
Successfully synthesized a supervisor for the system

The total state-space is at most $8.24 \cdot 10^{113}$ states

The supervisor adheres to all requirements by construction and is nonblocking

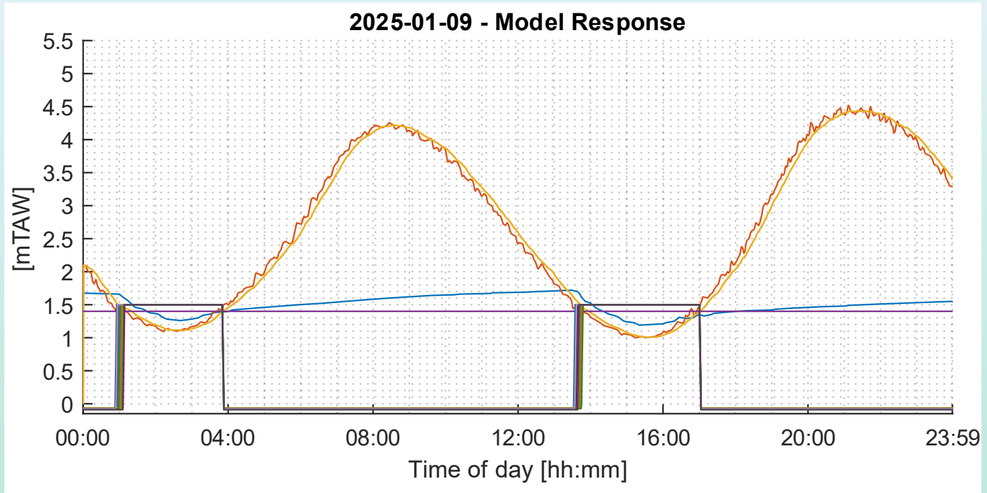

SVG Visualizer
SVG Visualizer
Dijkweg
Dijkweg
Downstream
Sea
Upstream
Noordede
Manual
Gate 13
Bottom height culvert: -0.09 mTAW
Bottom height gate: -0.09 mTAW
Manual
Gate 11
Bottom height culvert: -0.08 mTAW
Bottom height gate: -0.08 mTAW
Manual
Gate 9
Bottom height culvert: -0.07 mTAW
Bottom height gate: -0.07 mTAW
Auto
Gate 7
Bottom height culvert: -0.07 mTAW
Bottom height gate: -0.07 mTAW
Manual
Gate 5
Bottom height culvert: -0.07 mTAW
Bottom height gate: -0.07 mTAW
Manual
Gate 3
Bottom height culvert: -0.07 mTAW
Bottom height gate: -0.07 mTAW
Manual
Gate 1
Bottom height culvert: -0.08 mTAW
Bottom height gate: -0.08 mTAW
Opened
Closed
M
Manual
Gate 14
Bottom height culvert: -0.09 mTAW
Bottom height gate: -0.09 mTAW
Manual
Gate 12
Bottom height culvert: -0.08 mTAW
Bottom height gate: -0.08 mTAW
Manual
Gate 10
Bottom height culvert: -0.07 mTAW
Bottom height gate: -0.07 mTAW
Auto
Gate 8
Bottom height culvert: -0.07 mTAW
Bottom height gate: -0.07 mTAW
Manual
Gate 6
Bottom height culvert: -0.07 mTAW
Bottom height gate: -0.07 mTAW
Manual
Gate 4
Bottom height culvert: -0.07 mTAW
Bottom height gate: -0.07 mTAW
Manual
Gate 2
Bottom height culvert: -0.09 mTAW
Bottom height gate: -0.09 mTAW
Downstream Level
1.45 mTAW
Downstream Level Average
1.45 mTAW
Control building
Difference
0.21 mTAW
Control Level
1.40 mTAW
Upstream Level Average
1.66 mTAW
Upstream Level
1.66 mTAW
Main Screen
Parameters
Alarms
Tide control downstream
Fall
Stable
Rise
Tide control upstream
Fall
Stable
Rise

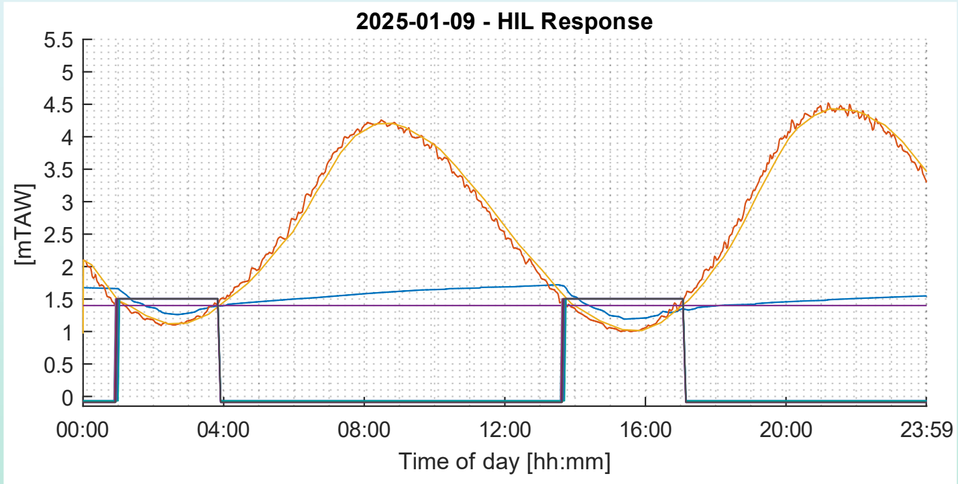

SVG Visualizer
Downstream water level
1.45 mTAW
Upstream water level
1.66 mTAW
Emergency button
Reset
Downstream
Culvert 1
Culvert 2
Culvert 3
Culvert 4
Culvert 5
Culvert 6
Culvert 7
Bottom height gate 1
-0.08 mTAW
Bottom height gate 3
-0.07 mTAW
Bottom height gate 5
-0.07 mTAW
Bottom height gate 7
-0.07 mTAW
Bottom height gate 9
-0.07 mTAW
Bottom height gate 11
-0.08 mTAW
Bottom height gate 13
-0.09 mTAW
Manual
Auto
Open
Opened
Stop
Alarm
Close
Closed
Bottom height gate 2
-0.09 mTAW
Bottom height gate 4
-0.07 mTAW
Bottom height gate 6
-0.07 mTAW
Bottom height gate 8
-0.07 mTAW
Bottom height gate 10
-0.07 mTAW
Bottom height gate 12
-0.08 mTAW
Bottom height gate 14
-0.09 mTAW
Gravitational discharge
Fish passage
Upstream
Maintenance switch gate 1
Maintenance switch gate 3
Maintenance switch gate 5
Maintenance switch gate 7
Maintenance switch gate 9
Maintenance switch gate 11
Maintenance switch gate 13
Maintenance switch gate 2
Maintenance switch gate 4
Maintenance switch gate 6
Maintenance switch gate 8
Maintenance switch gate 10
Maintenance switch gate 12
Maintenance switch gate 14
On
Off


Plot Visualizer
2.2
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
100
200
300
400
500
600
700
800
900
1,000
1,100
1,200
1,300
1,400
1,500
1,600
1,700
1,800
1,900
2,000
2,100
2,200
2,300
2,400
2,500
2,600
2,700
2,800
2,900
3,000
time
osition.Value
Gate13.Position.Value
Gate14.Position.Value
Gate2.Position.Value
Gate3.Position.Value
Gate4.Position.Value
Gate5.Position.Value
Gate6.Position.Value
Gate7.Position.Value
Gate8.Position.Value
Gate9.Position.Value

Hardware-in-the-loop testing

- PLC code generated from the model (structured text)
- Siemens S7-300 model used
- Only 8 gates due to CPU and memory limitations
- Ignition interface to simulate and visualize the system
- Performed several tests using historical data





Lessons learned

Interreg North Sea
Co-funded by the European Union
STORM_SAFE

Using synthesis-based engineering, a supervisory controller can be obtained that satisfies all modeled requirements

The supervisor model is a hardware-independent representation of the desired control logic, and can be translated to Siemens, Beckhoff, ABB, PLCopen XML, C-code, Matlab Simulink

Ambiguity in documentation needed to be resolved

Since the model follows the as-is situation, some subroutines look complicated in the model (downside of SBE, other formal methods might perform better in this case)

Partial hardware-in-the-loop testing due to limited-capacity PLC hardware

Readability of generated code not on-par with human produced code

Next steps

Future projects

- Add and test several fault situations (ongoing project)
- Hardware-in-the-loop testing using Pheonix Contact series
- Optimalization of parameter values to maximize gravitational discharge, especially during high tide and storm surges
- Implementing the formal method on a pumping station of the Zwim area
- Implementation and standardization through government procurement specifications

Shaping the future of software reliability

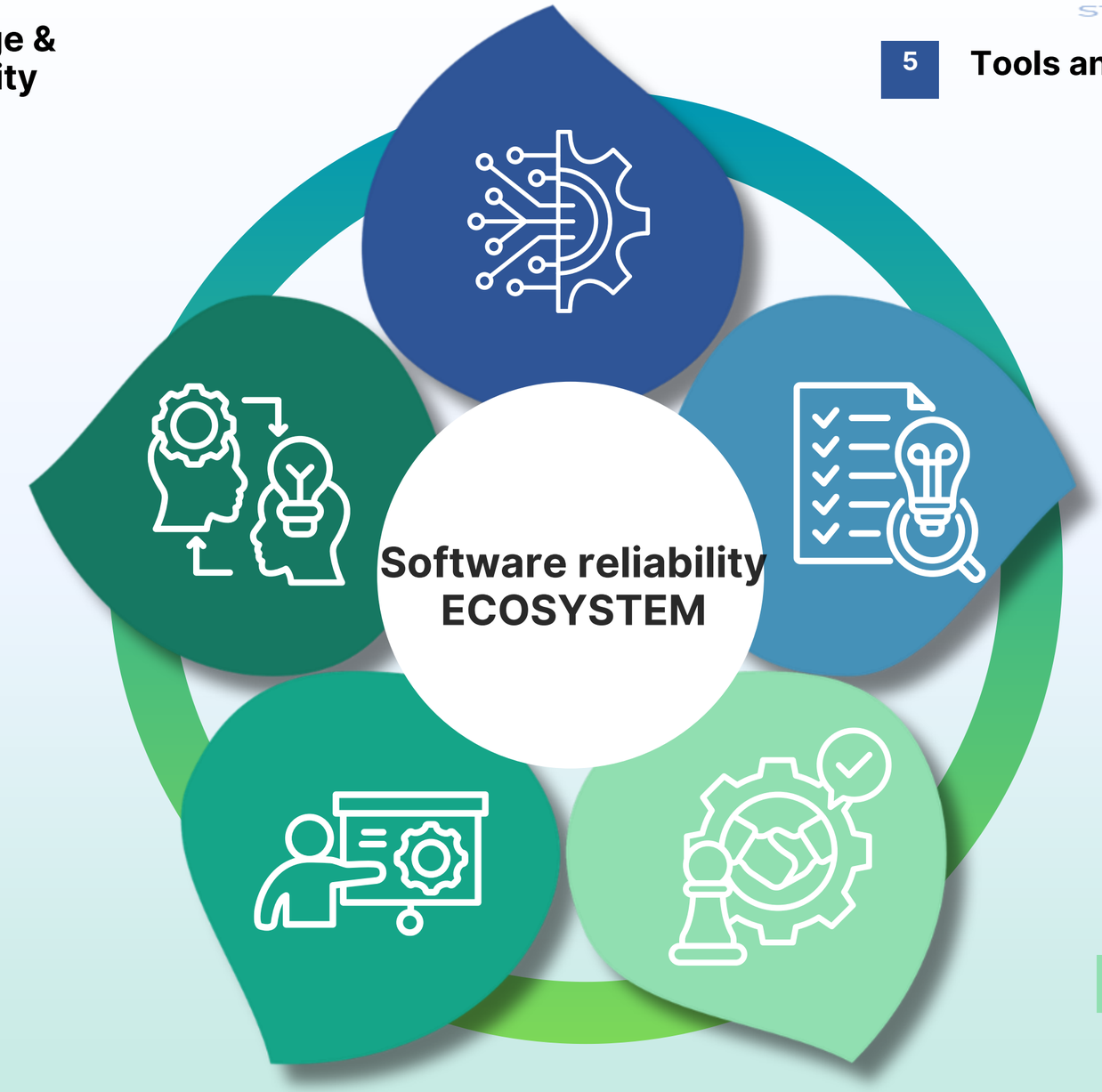
Interreg North Sea
Co-funded by the European Union
STORM_SAFE

Connecting research excellence with real life solution

Developing resilient mission-critical systems

Strengthening digital resilience in the North Sea Region

A community of Knowledge & Practice Software Reliability



1 Community of Practice & Knowledge sharing

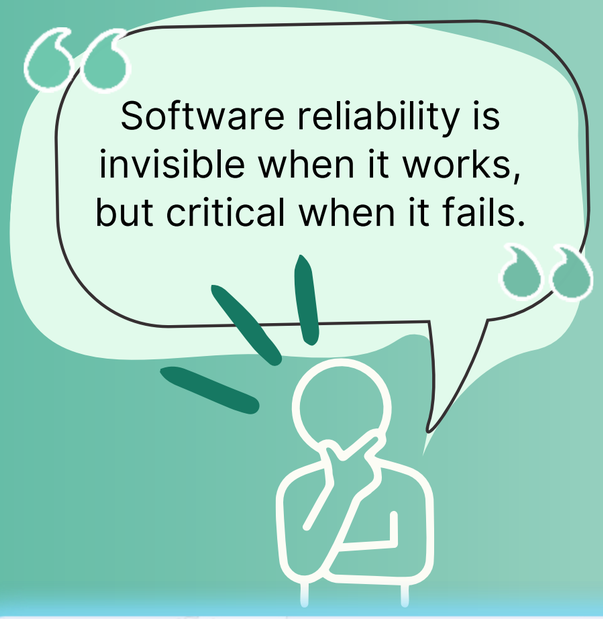
2 Training and Skills

3 Strategic Collaboration

4 Pilot applications

5 Tools and Methods

CONCLUSION



Shape the Future of
Resilient Water Systems

Interreg North Sea

Co-funded by the European Union

STORM_SAFE

CALL TO ACTION

- Collaborative work
- Shared learning
- Innovative solutions in action

JOIN US