argosim
SYSTEM VALIDATION

# Innovation and efficiency for Safety Requirements Validation

Continuous and Automatic **Safety Requirements Validation**
all along the Development Process

## ❑ Presentation objectives:

- Be informative on the recent progress of the **state-of-the-art** in safety verification techniques

- Present an innovative approach enabling to **prevent the safety requirements violations all along the development process**

- As the approach is supported by an easy-to-use industrial tool, propose a method **immediately applicable** to any embedded system project

# Agenda

- Argosim Company
- Classic safety validation approaches and their limits
  - Reviews
  - Formal Verification Techniques
- An innovative approach: continuous safety requirements validation all along the development process
- Conclusion

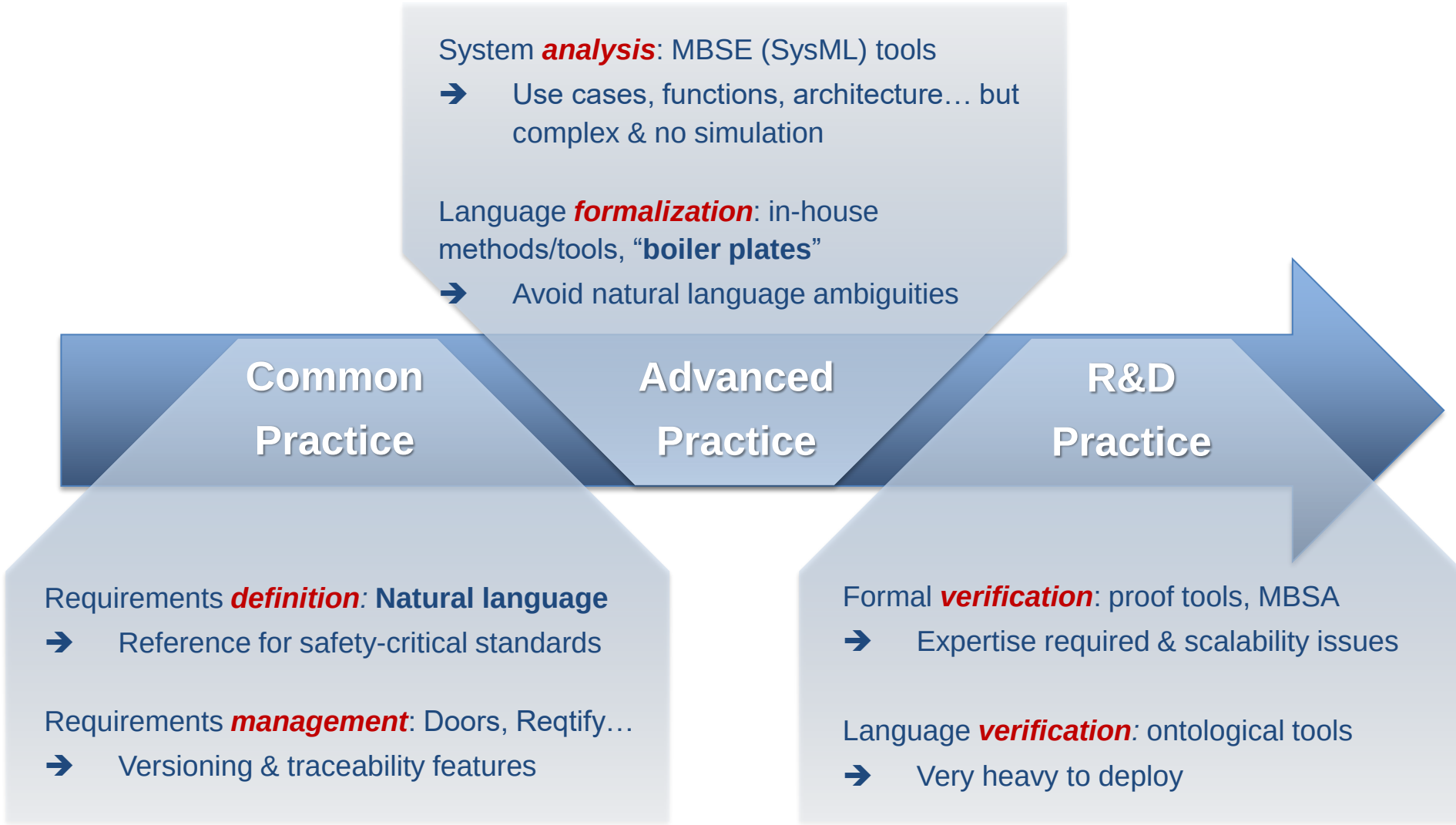
# Argosim Company Profile

- Company created in 2013 in Grenoble (France). STIMULUS released in early 2015.
- STIMULUS users in avionics, automotive, transportation, energy.
- International presence: USA, UK, Germany, Spain, Israel, Japan, China, Korea, India

# Agenda

- Argosim Company
- **Classic safety validation approaches and their limits**
  - **Reviews**
  - **Formal Proof**
- An innovative approach: continuous safety requirements validation all along the development process
- Conclusion

# Safety Requirement Engineering: State of the Art



# Classic Safety Requirements Validation: Reviews

- Objectives:
  - Verify that an artefact meets always a set of properties defined as "**safety properties**"
  - The artefact can be a specification, a design model or the actual implementation
- Reviewers should not be the artefact producers (independent review)
- Limits:
  - Manual work: tedious, costly, review errors can be made
  - Expert reviewers often good at detecting problems but can only achieve a limited level of confidence
- Conclusion: can find safety violations but can't produce a high level of confidence that the safety properties are met

# Advanced Safety Requirements Validation: Formal Proof

- Objectives:
  - Verify that an artefact meets always a set of properties defined as "**safety properties**"
  - The artefact can be a formal specification or a design model (Simulink, SCADE)

- When a proof is produced, a very high level of confidence is achieved

- Limits:
  - A lot of manual work to define the properties to be verified
  - Language to be used requires formal methods experts so the approach can't be widely deployed
  - Difficult scalability issues often **prevent getting any result**

- Conclusion: even though formal proof seems very attractive, it can rarely be used as the key method for safety properties verification

# Agenda

- Argosim Company
- Classic safety validation approaches and their limits
  - Reviews
  - Formal Proof
- **An innovative approach: continuous safety requirements validation all along the development process**
- Conclusion

# Innovative Safety Validation: pre-requisites

- Deploying an effective verification method requires to:
  - Respect the industry practices as much as possible. For instance, safety properties are extracted from the requirements which are expressed in natural language. So, the formulation of the properties should be **as close as possible to natural language**
  - Use a **tool** that automates the verification
  - The tool should be user-friendly to be **accepted by practitioners**
  - The safety verification should be applied as early as the requirements engineering phase and be applicable then easily to all the following phases (design, coding, testing)

# Innovative Safety Requirements Validation: Basic Concepts

## ❑ Requirements:

- All functional Requirements can be written using a set of templates
- Example:
  - Natural language requirement: “When the switch is AUTO and the light intensity is less than 60% for more than one second, headlight shall be set to ON”
  - Equivalent template-based requirement:

condition

*When* lightIntensity *is less then* 60 % *for more than* 1 second ,
*then* headlight *shall be* ‘ON

action

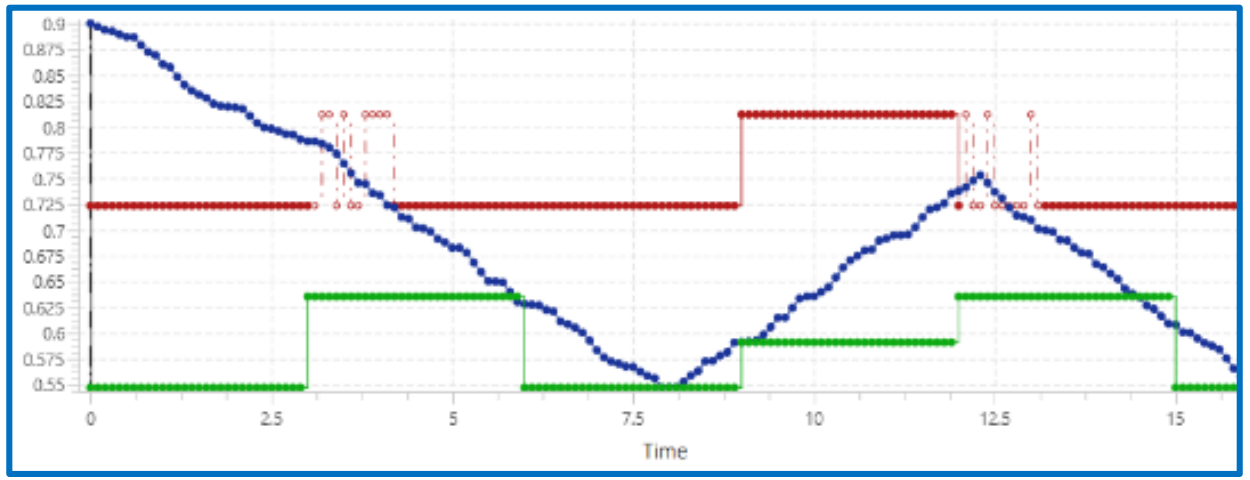
**Or, in Korean:**

lightIntensity 가 60 % 미만 인 상태가 1 [second] 이상 유지될 때,
headLight 는 'ON 이어야 한다

# Innovative Safety Requirements Validation: Basic Concepts

- Each template has an executable semantics

> *When* lightIntensity *is less then* 60 % *for more than* 1 second ,
> *then* headlight *shall be* 'ON

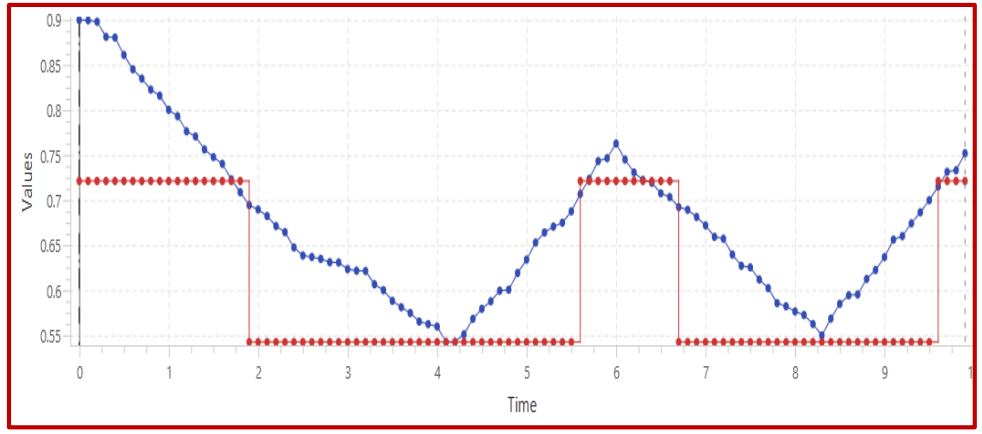



**Possible Execution of the System**

**Benefits:** Debug the Requirements as soon as your write them

# Innovative Safety Requirements Validation: Basic Concepts

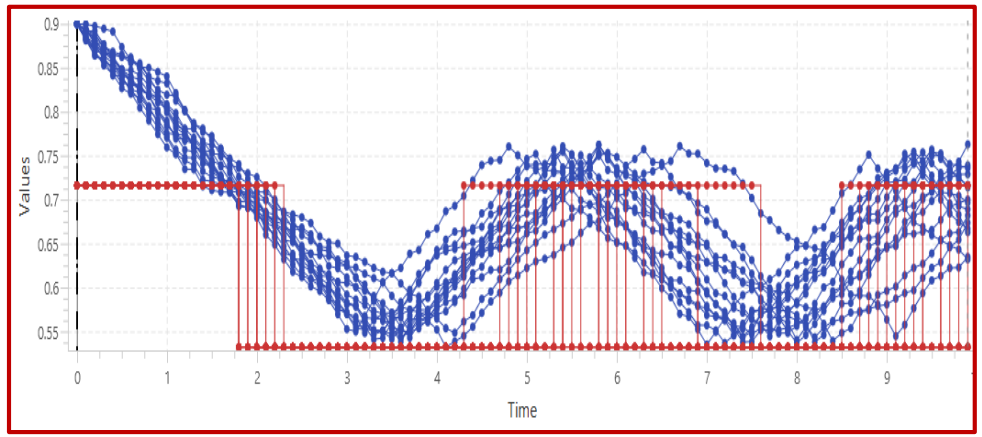
- **Use Case:** a set of constraints on the inputs and between the inputs
  - Example:
    - Vhcl_Acceleration ∈ [ 0 ; 5 ]
    - When LaneChange_Conditions = NOK,
      Stirring_Mode = Manual




One possible Test Vector

One Use Case can be turned into **as many Test Vectors as desired**

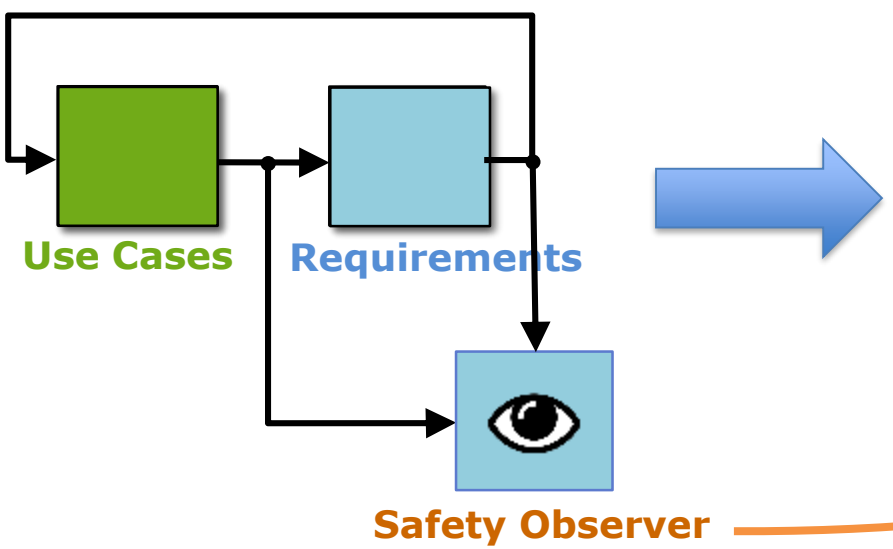
**(Automatic Test Vectors Generation)**

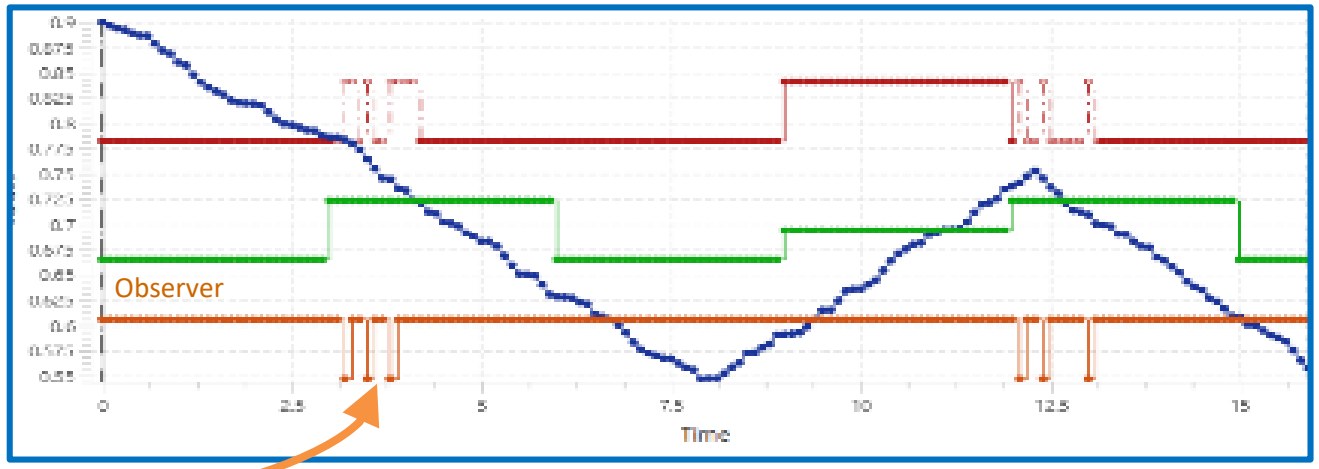



15 possible Test Vectors

# Validate Specification vs Safety Requirements

- Principle:
  - Define the safety properties as **Observer**
  - Apply the test cases generated from the Use Cases to the Requirements
  - **The Observer will detect any safety properties violation**

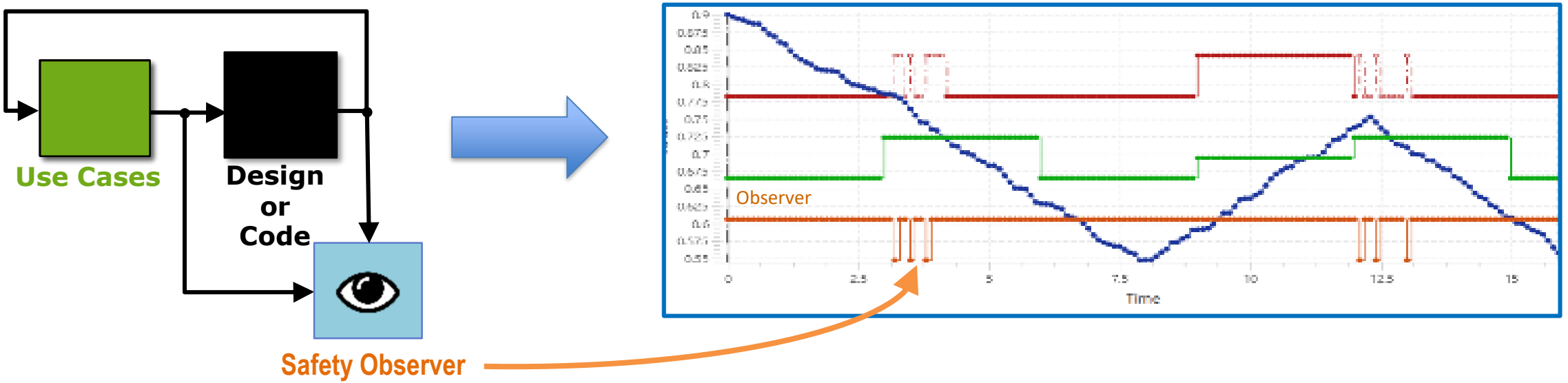



**Benefits:** Validate the Specification vs the safety properties automatically

# Validate Implementation vs Safety Requirements

- Principle:
  - Reuse the Observer containing the safety properties
  - Generate code from the Design and compile it into a DLL
  - Apply the test cases generated from the Use Cases to the DLL
  - The Observer will detect any safety properties violation

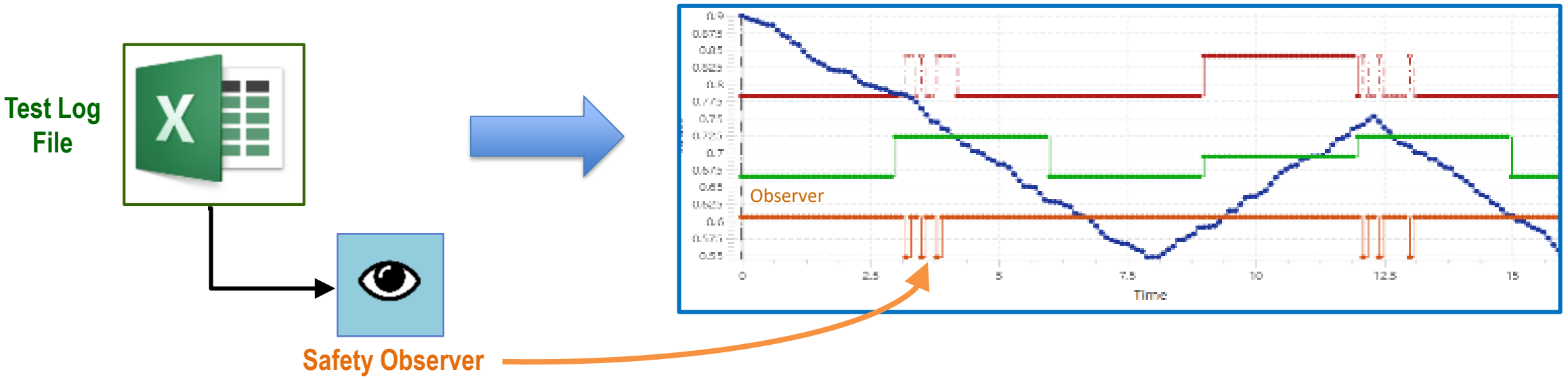



## Benefits:

1. Validate the Implementation vs the safety properties automatically
2. Validate the Implementation vs the System Requirements (put the requirements into the Observer)

# Validate System vs Safety Requirements

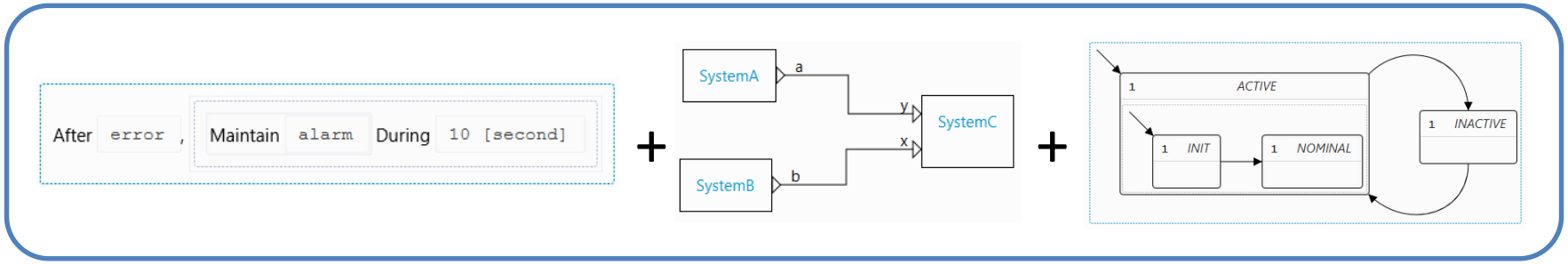
- Principle:
  - Apply the test cases generated from the Use Cases to the system in its testing environment
  - Record all I/Os into a log file
  - Load the log file (formatted in CSV) and the Observer will detect any safety properties violation




## Benefits:

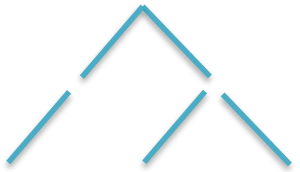
1. Validate the System vs the safety properties automatically
2. Validate the System vs the System Requirements (put the requirements into the Observer)

# Theoretical and Technical Backgrounds




## Compiler

Constraints over variables



Data constraints:
- Logico-numerical solver processing the relationships among data

- Control graph
- Backtrack mechanism

## Simulator

BDD (Binary Decision Diagrams) + convex polyhedra

## Where is this approach used?

- Countries: Europe, Japan, USA
- Domains:
  - Transportation: signalling systems, embedded systems on trains and metros, ground systems
  - Automotive: Engine control, ADAS, Body control
  - Aerospace/Defence: automatic pilots, airborne radars, on-board fuel management systems…
  - Energy: Nuclear Plants control and command systems
- Any application with safety / high integrity constraints

# Agenda

- Argosim Company
- Classic safety validation approaches and their limits
  - Reviews
  - Formal Proof
- An innovative approach: continuous safety requirements validation all along the development process
- **Conclusion**

# Safety Requirement Engineering: **New** State of the Art

**+ Simulation = Effective safety requirements verification**

System ***analysis***: MBSE tools

➔ Use cases, functions, architecture… but complex & no simulation

Language ***formalization***: in-house methods/tools, **"boiler plates"**

➔ Avoid natural language ambiguities

**Common Practice** | **Advanced Practice** | **R&D Practice**

Requirements ***definition***: **Natural language**

➔ Reference for safety-critical standards

Requirements ***management***: Doors, Reqtify…

➔ Versioning & traceability features

Formal ***verification***: proof tools, MBSA

➔ Expertise required & scalability issues

Language ***verification***: ontological tools

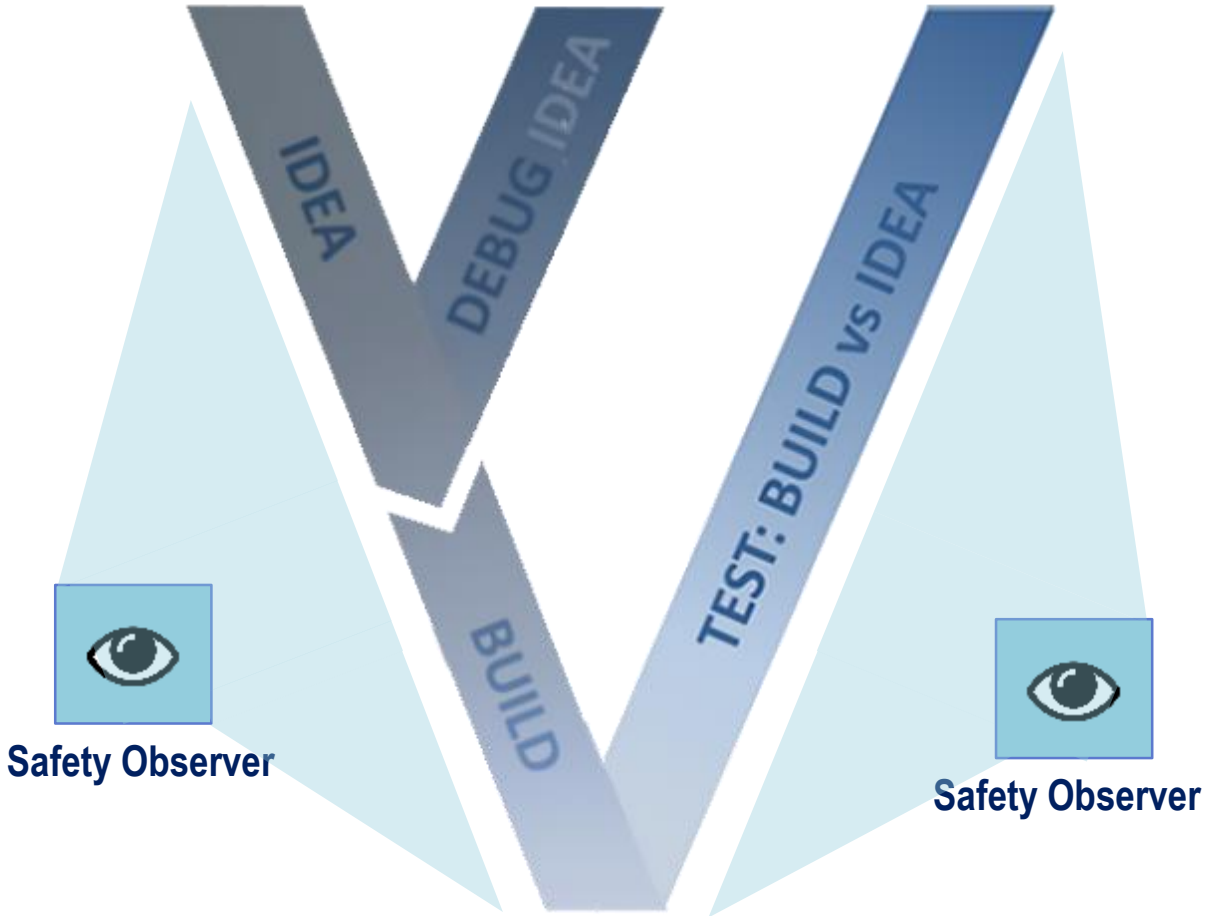
➔ Focus on the form than the semantics

# Conclusion


- Safety Activities Start from the Specification phase but consist essentially in manual reviews
- **The State-of the-Art has changed**: you can now actually debug the requirements…

  …and assess automatically the specification compliance to the safety requirements

# Conclusion




- Then, during all the development process the safety properties can be reused as "Virtual Guides" (Observers) that will **detect automatically violations of the Safety**


- Safety Engineers remain the safety process pilots but they can **make the Observers available to any actor of the development process**: System Architects, Designers, SW Engineers, Validation Teams…

# Thank you ! 감사합니다

# Questions?

yves.genevaux@argosim.com