

# Justifications

**Describing Explicitly & Continuously Verifying Justification Reasonings**

SIG LLODIA – 2026-06-25

`jean-michel.brue1@irit.fr`

# Available material

<https://www.jpipe.org/>

<https://bit.ly/jmb-talks>

Get the slides (pdf)

HOW TO CITE:

Jean-Michel Bruel et al, “Justification Diagrams presentation at the SIG LLODIA”.  
Webinar, 2026.



*If you have any content that I did not reference well or that should be removed, please do not hesitate to contact me so that I can correct this presentation.*

# Outline

- Context
- Jpipe
- Ongoing research

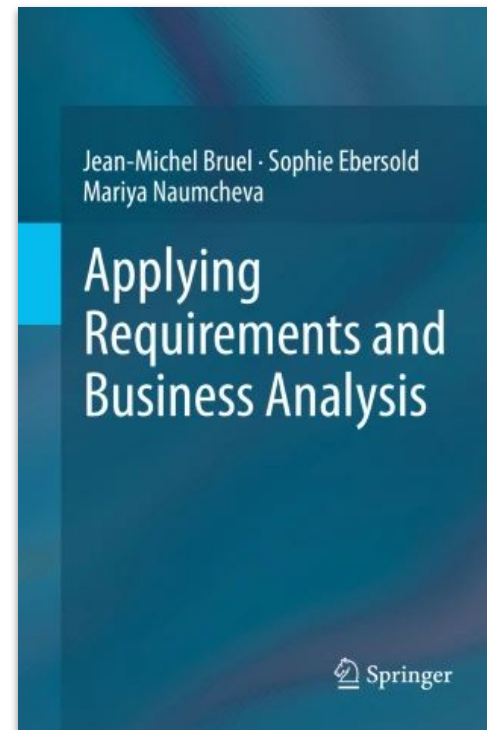
# Context



## Why me?

- Professor at Toulouse University
  - Teaching **modeling** and **DevOps**
- Member of the CNRS-IRIT Laboratory
  - Model-Based **Systems Engineering**
- **Airbus** MBSE Chair of Toulouse
- Leader of the **companion book** (published in 2025)

<https://bit.ly/jmbruel>



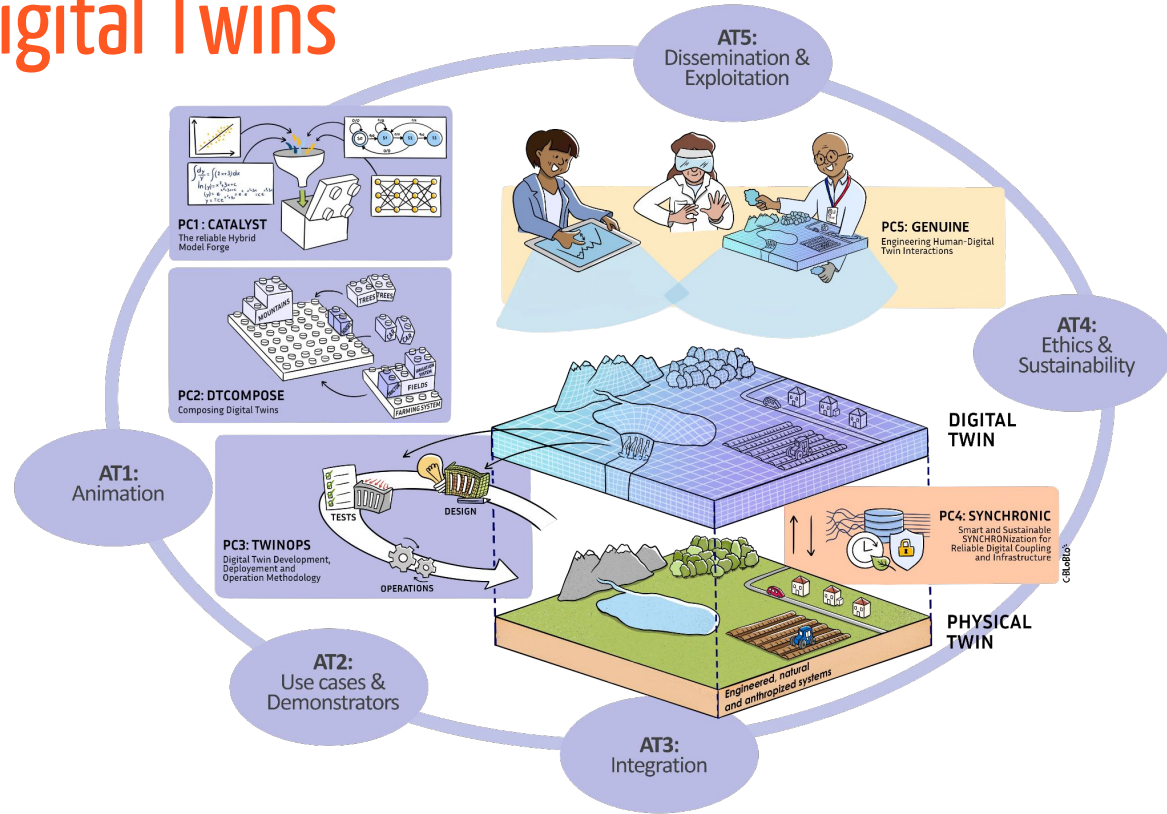
<https://requirements.university>



# Engineering Digital Twins

- French national project
  - 40 PhDs
  - 6 years
- Leading PC3 on methodology

<https://edtlab.fr/en/>



# Collaboration with Steffen Zschaler

## V&V of of Digital Twins

=> Justification Diagrams + **Ontologies**

=> Interest for SIG LLODIA?

# FATES-MLOps

Incorporating FATES Principles in Continuous Development of ML-Integrated Systems: A MLOps Perspective

ANR Project Funding this research

**F**airness

**A**ccountability

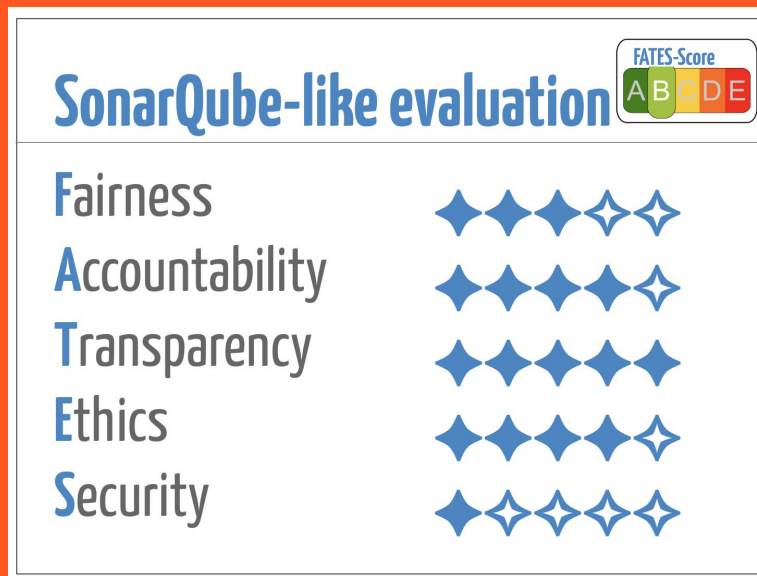
**T**ransparency

**E**thics

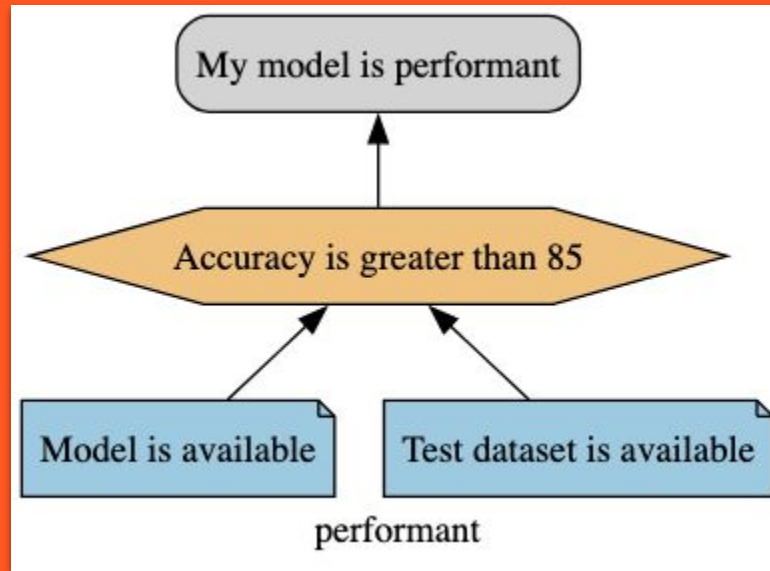
**S**ecurity (and/or Safety and/or Sustainability)



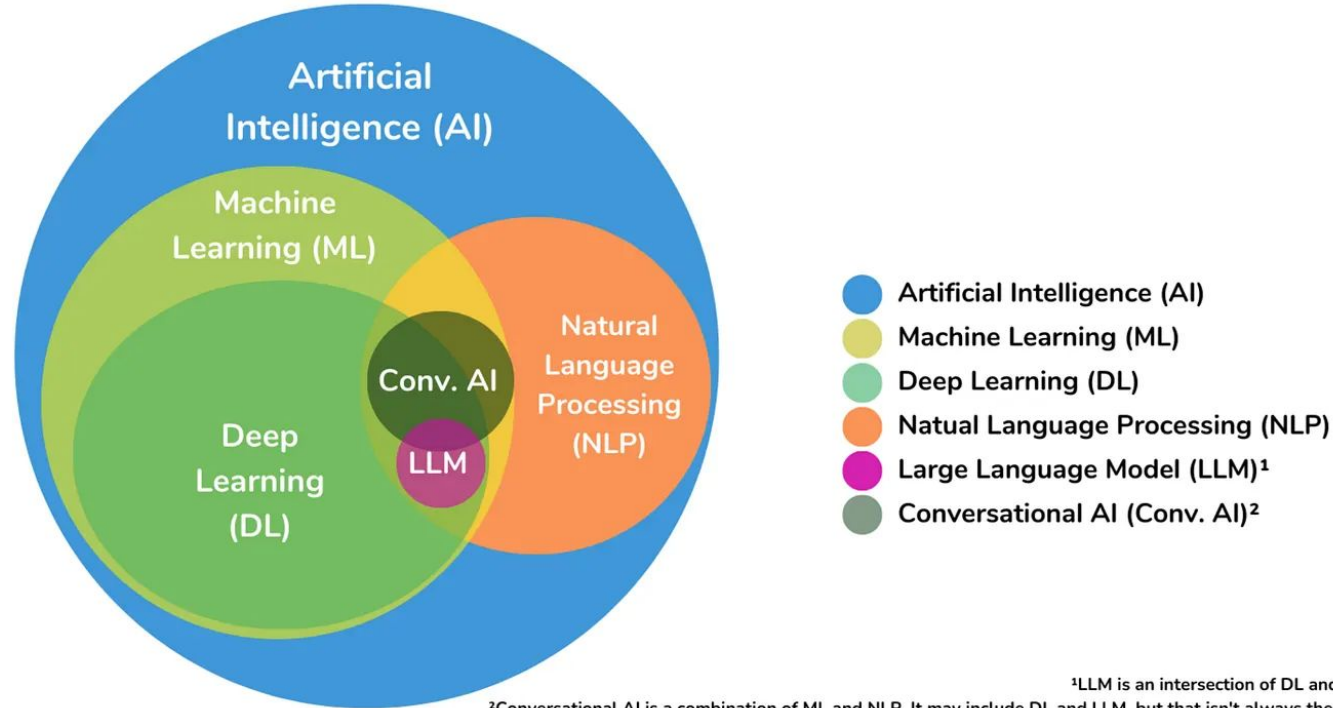
# Claim #1: AI needs Q&A (Quality Assessment)



# Claim #2: AI needs Requirements Engineering



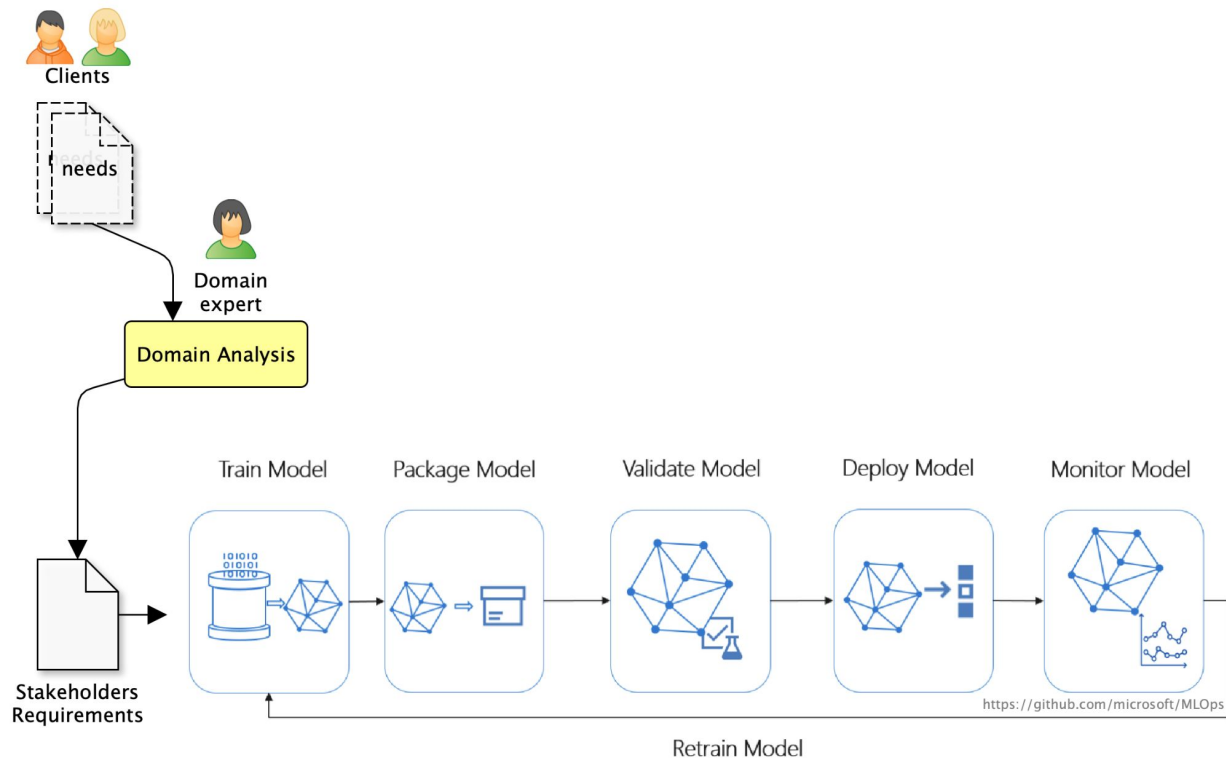
# AI / GenAI / LLM / ML / AIA => ML(Ops)



<sup>1</sup>LLM is an intersection of DL and NLP  
<sup>2</sup>Conversational AI is a combination of ML and NLP. It may include DL and LLM, but that isn't always the case.

# Big picture

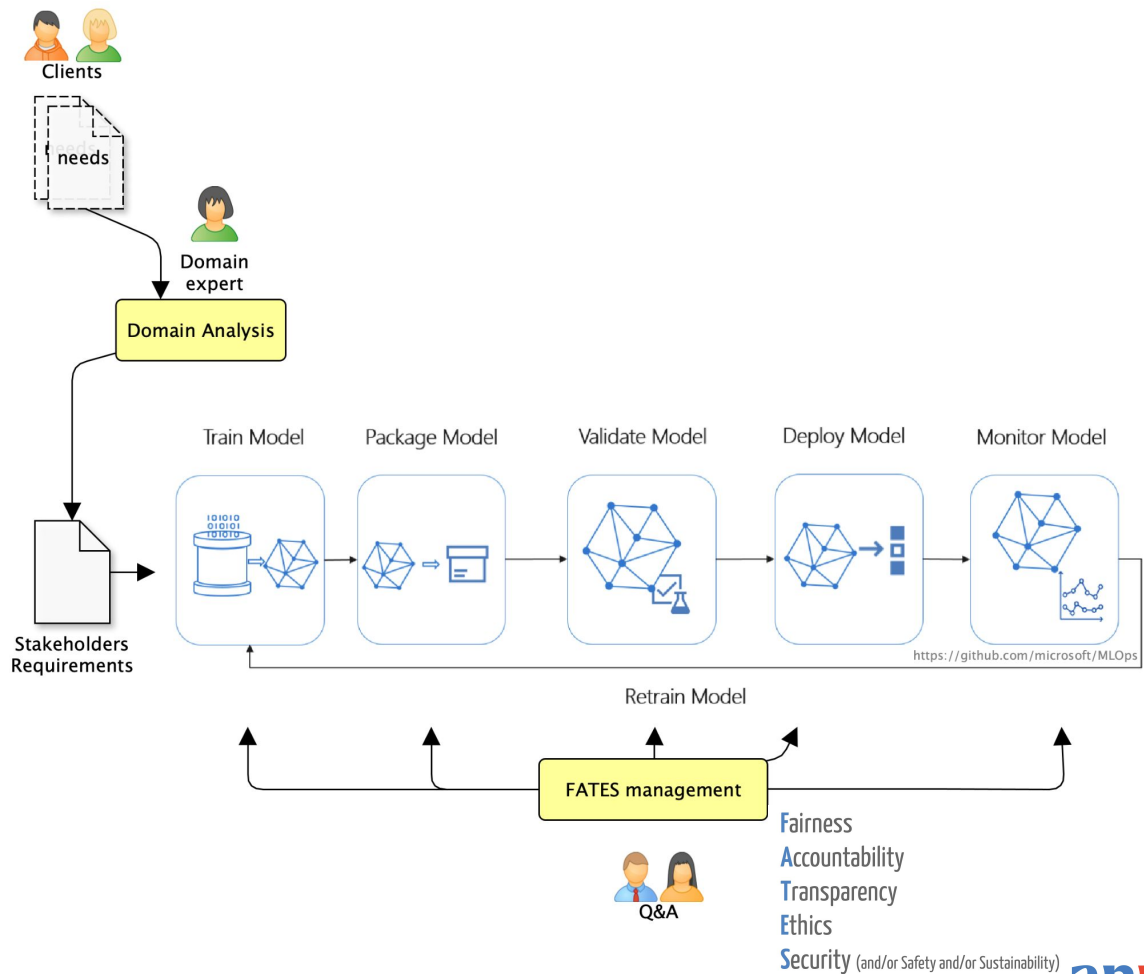
## MLOps context



# Big picture

FATES consideration

Continuous effort



<https://github.com/microsoft/MLOps>

# FATES properties

**F**airness

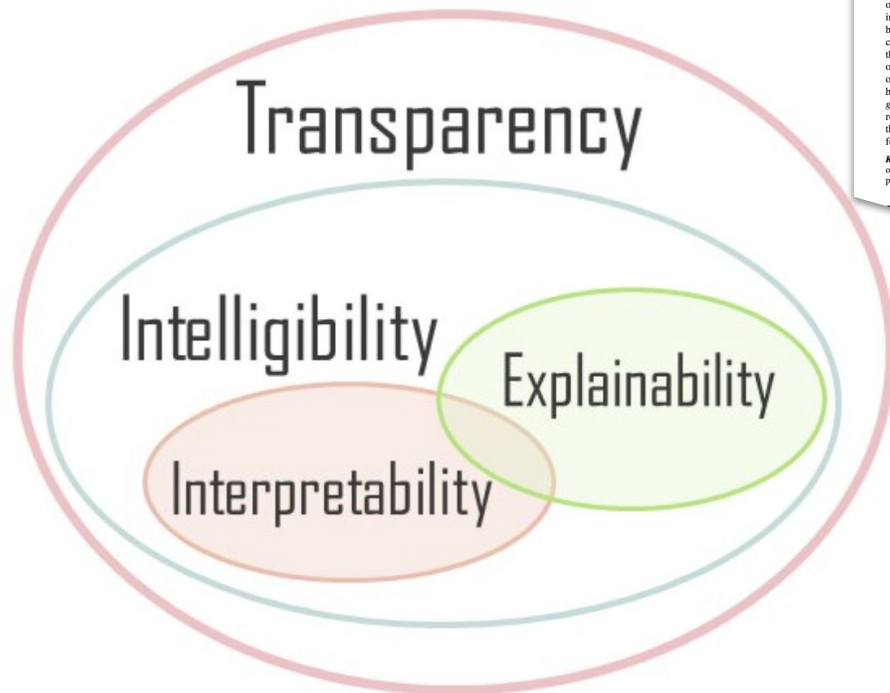
**A**ccountability

**T**ransparency

**E**thics

**S**ecurity (and/or **S**afety and/or **S**ustainability)

# Definitions issues



## A Survey of Explainable AI Terminology

Miruna A. Clinciu and Helen F. Hastie  
Edinburgh Centre for Robotics  
Heriot-Watt University, Edinburgh, EH14 4AS, UK  
{mc191, H.Hastie}@hw.ac.uk

### Abstract

The field of Explainable Artificial Intelligence attempts to solve the problem of algorithmic opacity. Many terms and notions have been introduced recently to define Explainable AI, however, these terms seem to be used interchangeably, which is leading to confusion in this rapidly expanding field. As a solution to overcome this problem, we present an analysis of the existing research literature and examine how key terms, such as *transparency*, *intelligibility*, *interpretability*, and *explainability* are referred to and in what context. This paper, thus, moves towards a standard terminology for Explainable AI.

**Keywords**— Explainable AI, Black-box, NLG, Theoretical Issues, Transparency, Intelligibility, Interpretability, Explainability

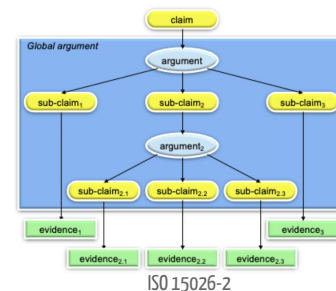
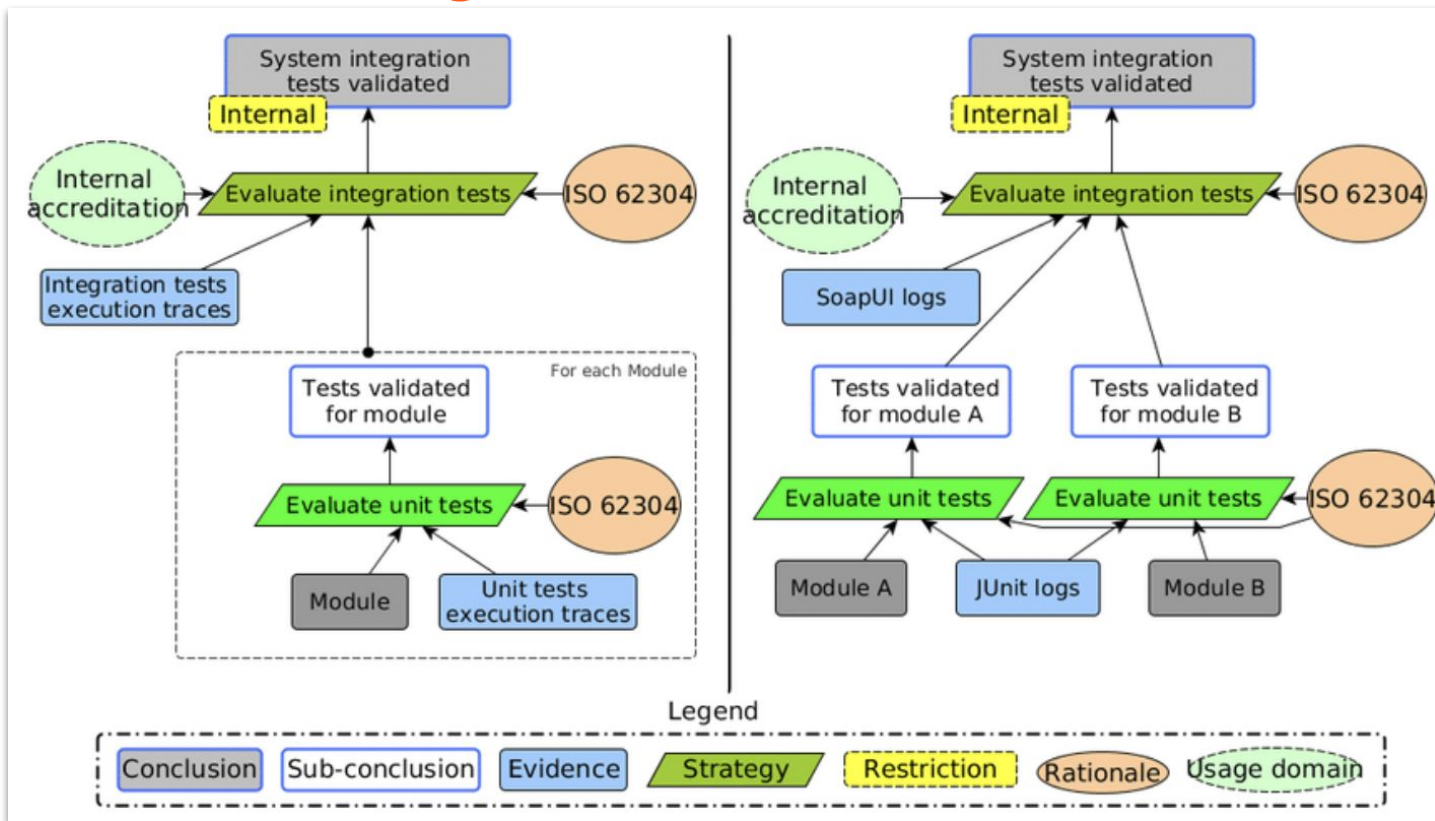
### Introduction

• “Explainable AI can present the user with an easily understood chain of reasoning from the user’s order, through the AI’s knowledge and inference, to the resulting behaviour” (van Lent et al., 2004).

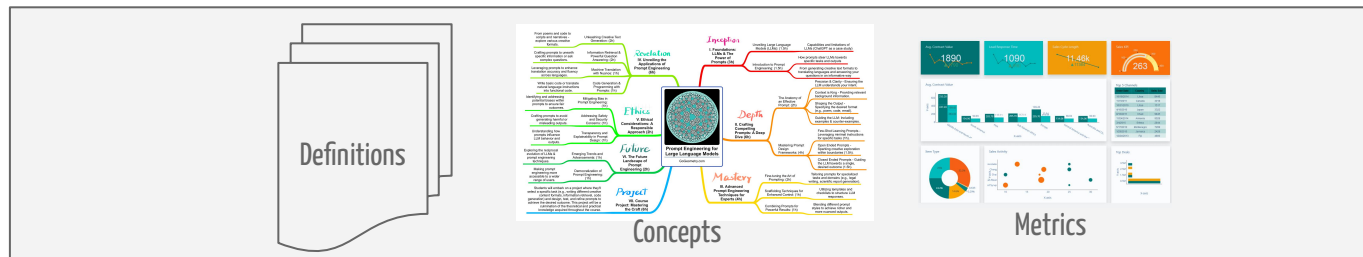
• “XAI is a research field that aims to make AI systems results more understandable to humans” (Adadi and Berrada, 2018).

Thus, we conclude that XAI is a research field that focuses on giving AI decision-making models the ability to be easily understood by humans. Natural language is an intuitive way to provide such Explainable AI systems. Furthermore, XAI will be key for both expert and non-expert users to enable them to have a deeper understanding of the appropriate level of explanation required to increase their confidence in the system’s output.

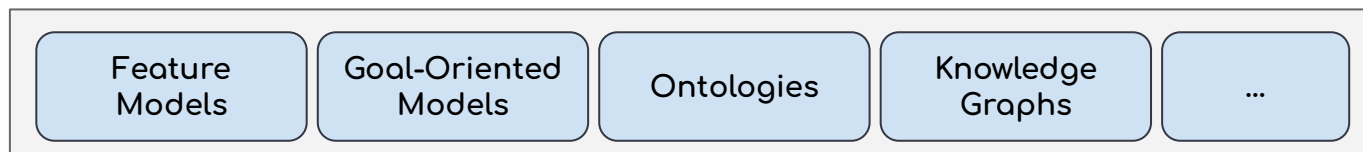
# Justification diagrams (ISO-IEC-IEEE 15026-2)



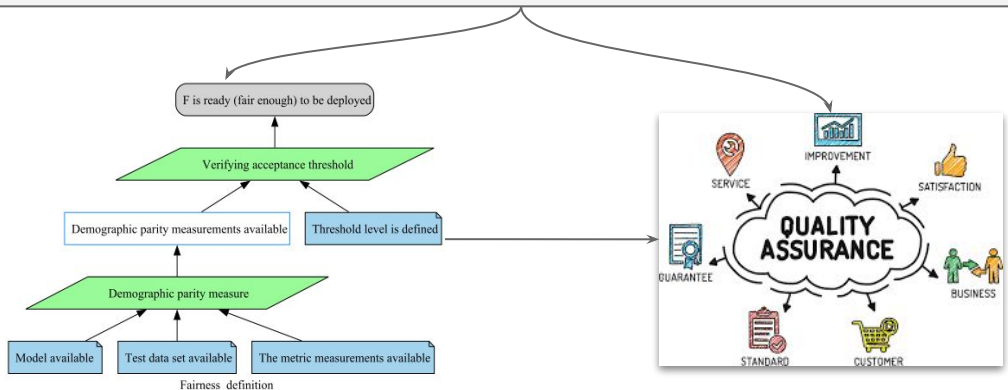
# Requirements and Traceability Issues



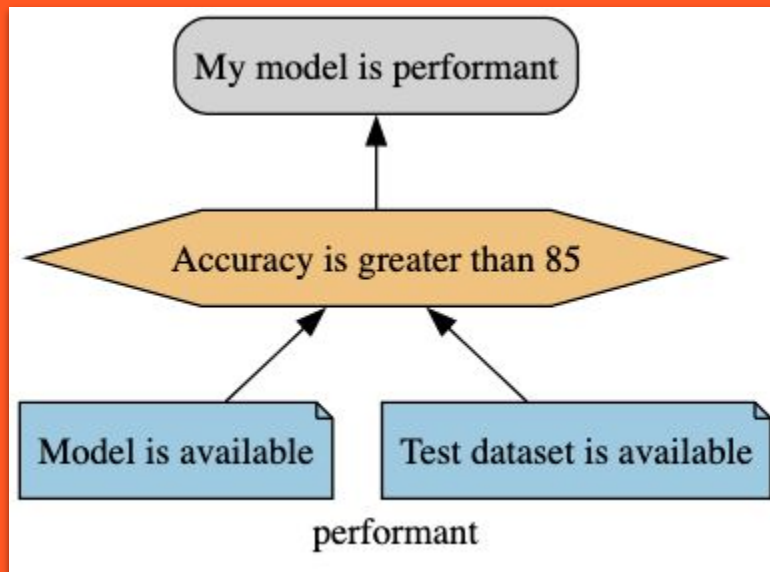
THIS EVOLVES FAST!



WHAT'S BEST?



# Jpipe





fairness-income

Update

fairClassification.jd ×

JPipe Preview ×

JPIPE

```
1 justification Fairness_definition {
2
3   conclusion c is "F is ready (fair enough) to be deployed"
4
5   strategy verify is "Verifying acceptance threshold"
6   verify supports c
7
8   sub-conclusion metric is "Demographic parity measurements"
9   metric supports verify
10
11   //evidence definition is "Fairness is defined"
12   //definition supports verify
13
14   evidence level is "Threshold level is defined"
15   level supports verify
16
17   strategy fmetric is "Demographic parity measure"
18   fmetric supports metric
19
20   evidence dataset is "Test data set available"
21   dataset supports fmetric
22
23   evidence model is "Model available"
24   model supports fmetric
25
26   evidence measurement is "The metric measurements available"
27   measurement supports fmetric
28 }
```

```
graph BT
    G1(F is ready (fair enough) to be deployed)
    G2{Verifying acceptance threshold}
    G3{Demographic parity measure}
    E1[Demographic parity measurements available]
    E2[Threshold level is defined]
    E3[Test data set available]
    E4[Model available]
    E5[The metric measurements available]

    G2 --> G1
    E1 --> G2
    E2 --> G2
    G3 --> E1
    E3 --> G3
    E4 --> G3
    E5 --> G3
```

main\* 3↓ 1↑ Launchpad 0 △ 0 Mode: Folder Git Graph Failed to initiate Application Insights extension. Check the console for mor SonarQu



## jPipe Language (McSCert)

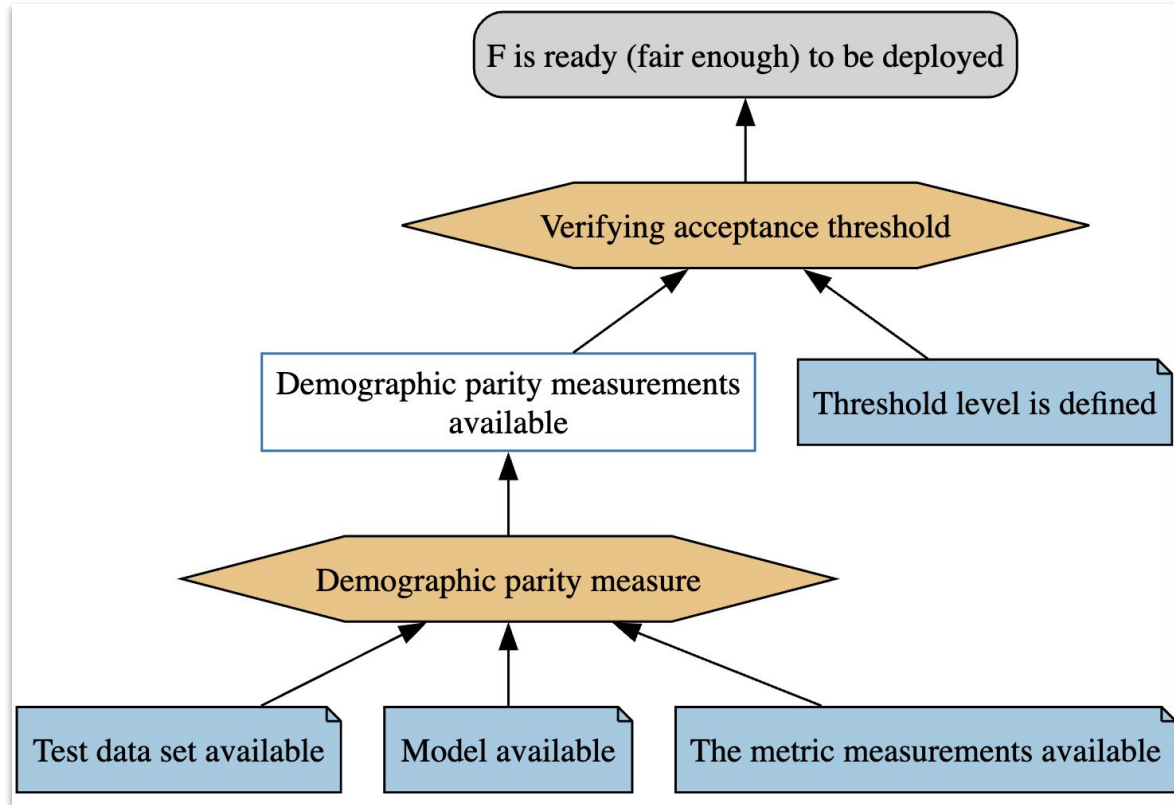
McMaster Centre for Software Certification (McSCert) [mcs-cert.ca](https://mcs-cert.ca)

Language Support for the jPipe language

Uninstall ✓ Auto Update ⚙

<https://jpipe.org>

# To express requirements we need measurable properties



# Jpipe: an editor for justification diagrams (vscode plugin)

The screenshot displays the Jpipe VS Code plugin interface. The left pane shows the source code for a justification diagram, and the right pane shows its graphical representation.

```
1 justification Fairness_definition {
2
3   conclusion c is "F is ready (fair enough) to be deployed"
4
5   strategy verify is "Verifying acceptance threshold"
6   verify supports c
7
8   sub-conclusion metric is "Demographic parity measurements"
9   metric supports verify
10
11   //evidence definition is "Fairness is defined"
12   //definition supports verify
13
14   evidence level is "Threshold level is defined"
15   level supports verify
16
17   strategy fmetric is "Demographic parity measure"
18   fmetric supports metric
19
20   evidence dataset is "Test data set available"
21   dataset supports fmetric
22
23   evidence model is "Model available"
24   model supports fmetric
25
26   evidence measurement is "The metric measurements available"
27   measurement supports fmetric
28 }
```

The diagram in the preview pane is a hierarchical justification graph. At the top is the goal node: "F is ready (fair enough) to be deployed". This goal is supported by the strategy "Verifying acceptance threshold". This strategy is supported by two sub-conclusions: "Demographic parity measurements available" and "Threshold level is defined". The "Demographic parity measurements available" sub-conclusion is supported by the strategy "Demographic parity measure". This strategy is supported by three evidence nodes: "Test data set available", "Model available", and "The metric measurements available".

# Jpipe: a code generator to support evaluations

```
@jpipe_link("performant:e1")
@jpipe(produce=["model"])
def model_is_available(produce: JpipeProduce) -> bool:
    """[evidence] Model is available"""
    if (found := 'model_file' in mock):
        produce('model', mock['model_file'])
    return found

@jpipe_link("performant:e2")
@jpipe(produce=["tests"])
def test_dataset_is_available(produce: JpipeProduce) -> bool:
    """[evidence] Test dataset is available"""
    if (found := 'test_dataset' in mock):
        produce('tests', mock['test_dataset'])
    return found
```

# Jpipe: a runner to execute...

```
step_02 $ jpipe-runner --library steps/performant.py \  
--diagram performant \  
--format svg performant.json
```



```
=====  
jPipe Files  
=====
```

```
jPipe Files.Justification :: performant  
=====
```

```
evidence<performant:e1> :: Model is available | PASS |  
-----
```

```
evidence<performant:e2> :: Test dataset is available | PASS |  
-----
```

```
strategy<performant:s> :: Accuracy is greater than 85 | PASS |  
-----
```

```
conclusion<performant:c> :: My model is performant | PASS |  
-----
```

```
jPipe Files
```

```
1 justification, 4 passed, 0 failed, 0 skipped  
=====
```

```
performant diagram saved to: performant.svg
```

# Jpipe: a runner to execute & support CI/CD

The screenshot shows a GitHub pull request interface. At the top, it says "Step 4 | Pushing new code #9" and "mouzer wants to merge 1 commit into main from step\_04". A comment from "github-actions" (Bot) states "Justification process failed!". Below this is a flowchart diagram illustrating the justification process. The flowchart starts with three blue boxes at the bottom: "Counterfactual dataset is available", "Model is available", and "Test dataset is available". Arrows from "Counterfactual dataset is available" and "Model is available" point to a yellow diamond decision box labeled "Flip-rate is below 10%". Arrows from "Model is available" and "Test dataset is available" point to a red diamond decision box labeled "Accuracy is greater than 85". Both decision boxes have arrows pointing to a white rectangular box labeled "My model is fair" and "My model is performance" respectively. These two boxes then have arrows pointing to a white diamond decision box labeled "All conditions are met". Finally, an arrow from "All conditions are met" points to a white rectangular box labeled "Model is deployable". Below the flowchart, there is a "Download Diagram Artifact" link and a "Runner Output" section. At the bottom of the screenshot, there is a summary of checks: "All checks have failed" (1 failing check), "jPipe Runner Workflow -- Step 04 / jpipe-runner (pull\_request) Falling after 40s", and "No conflicts with base branch" (Merging can be performed automatically).

<https://github.com/jpipe-mscert/bellairs-cse>

# Ongoing Research

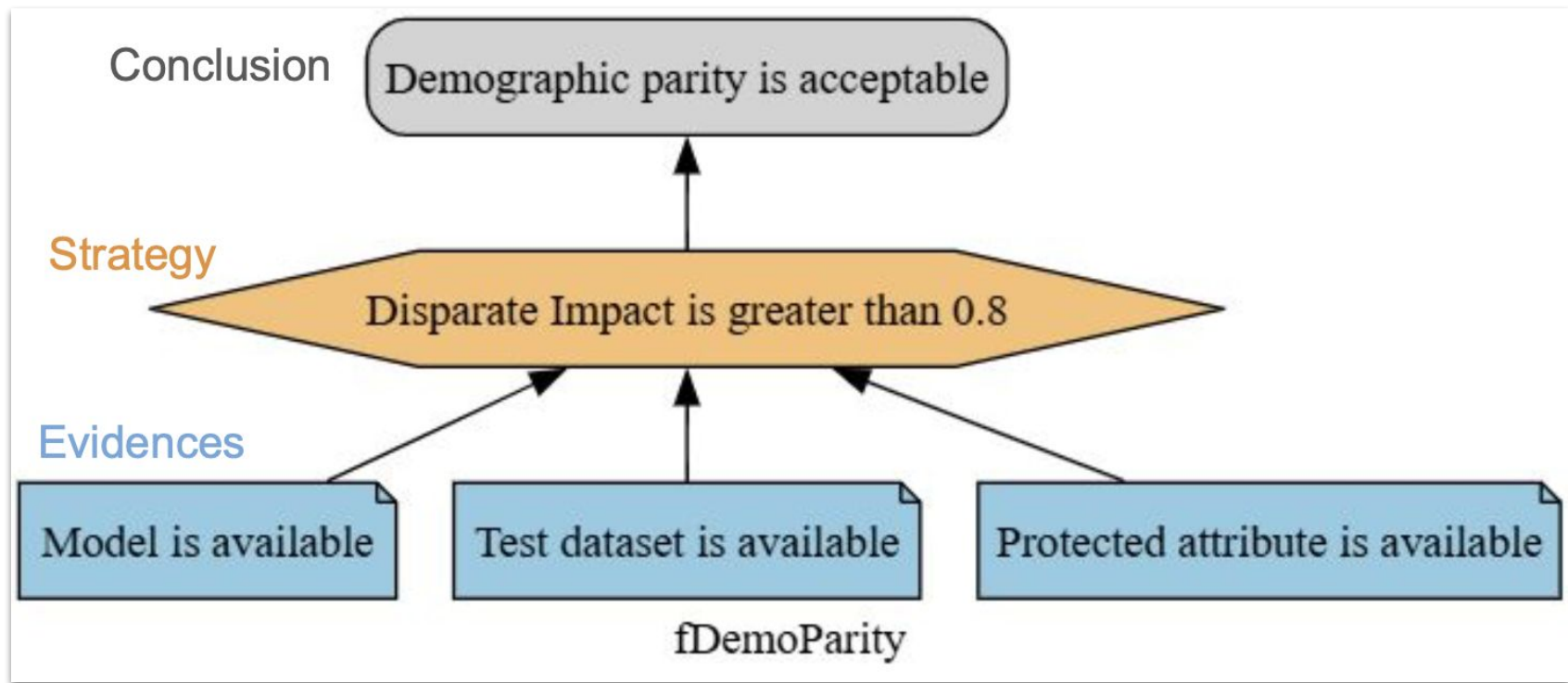


IRIT

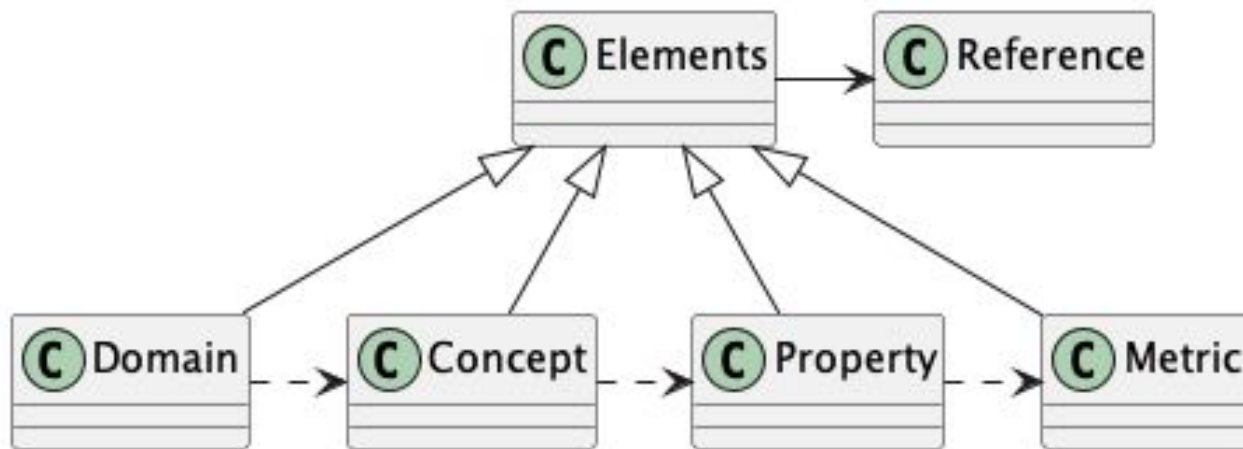
Institut de Recherche  
en Informatique de Toulouse  
CNRS - Université - MIT - Orange



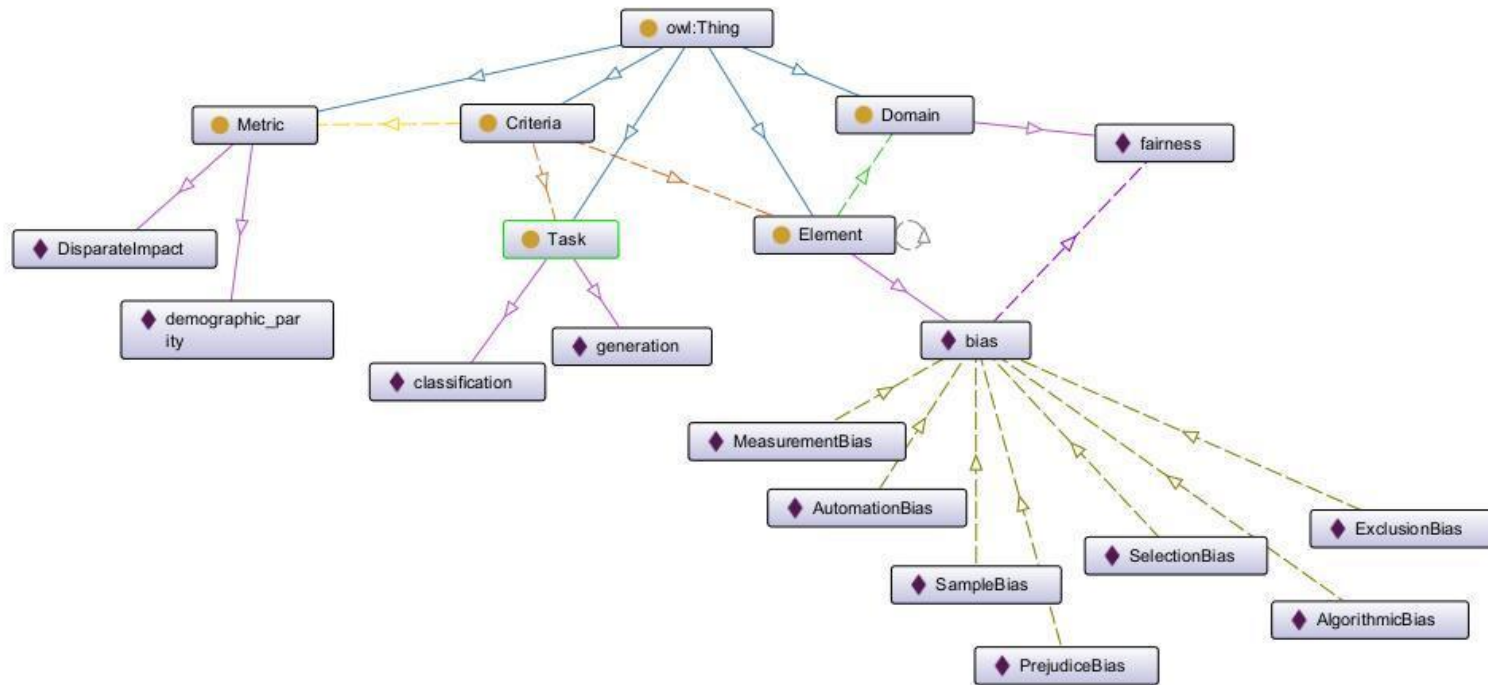
# Semantically correct justifications



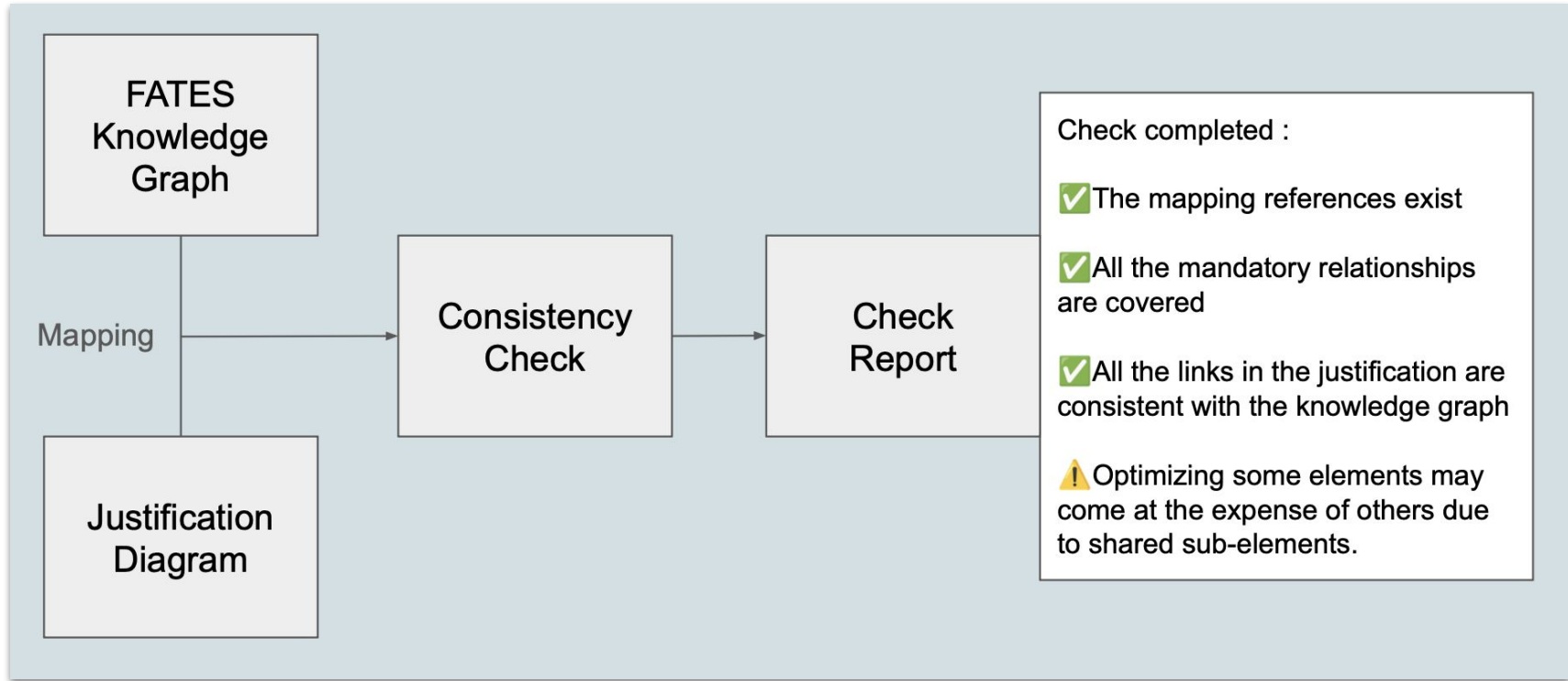
# Domain / Concepts / Properties / Metrics



# Domain / Concepts / Properties / Metrics

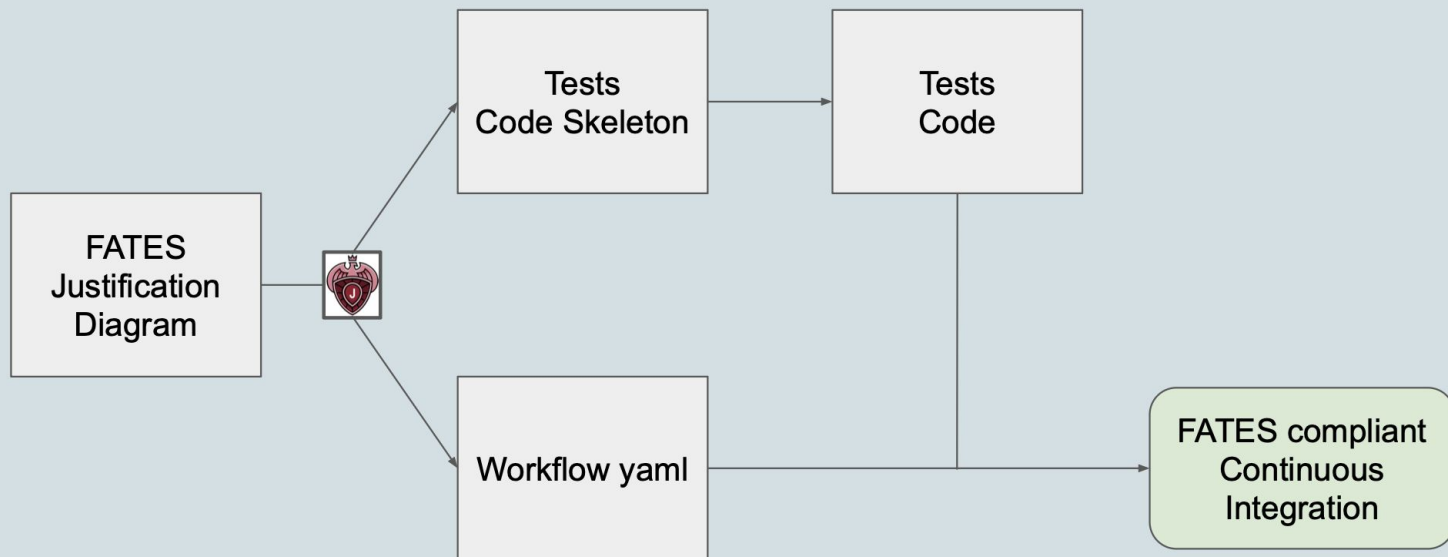


# Semantically correct justifications

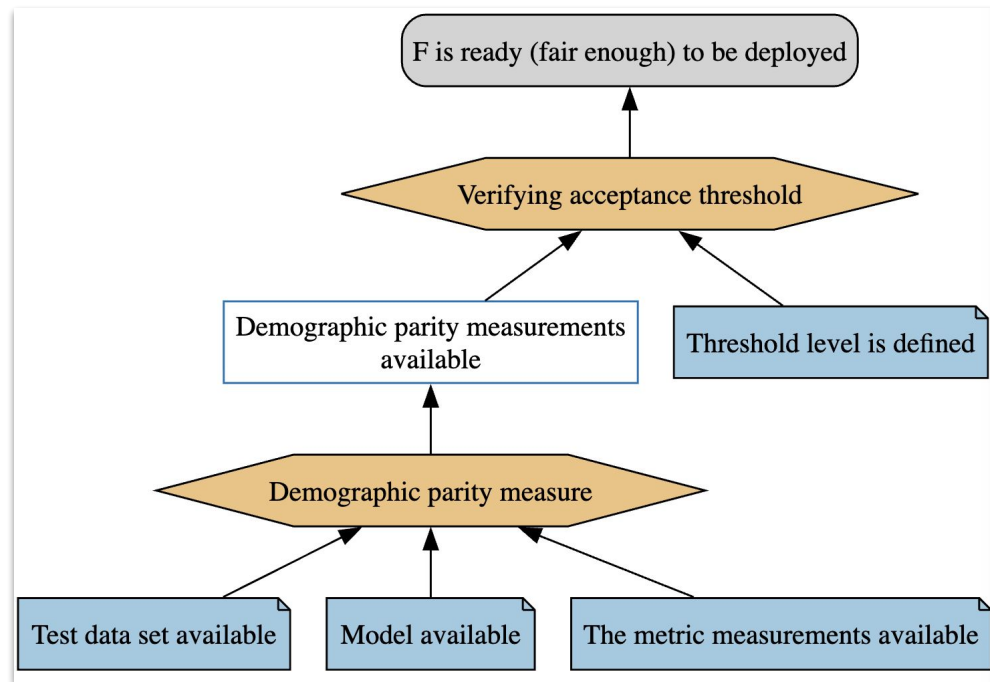
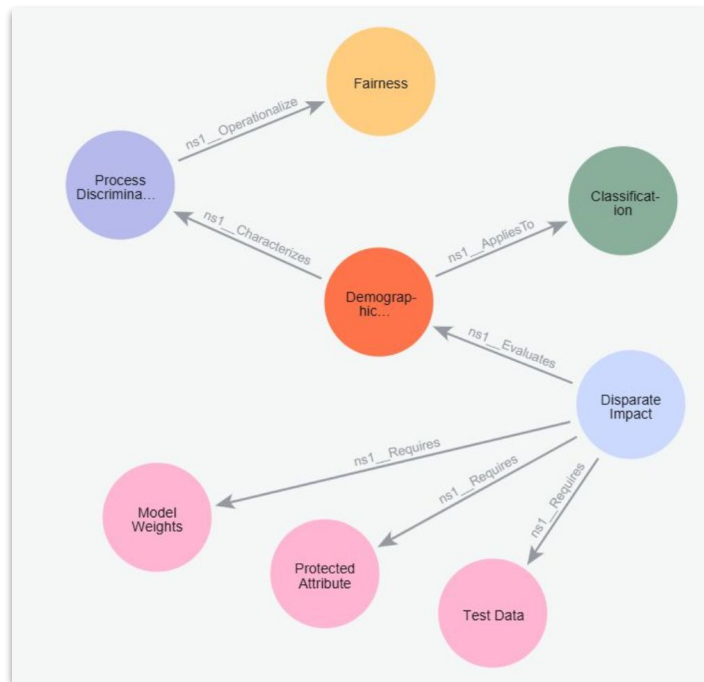


# Semantically correct justifications

From expression  
to integration



# Semantically correct justifications





RESEARCH  
PROGRAM

---

*Engineering  
Digital Twins*

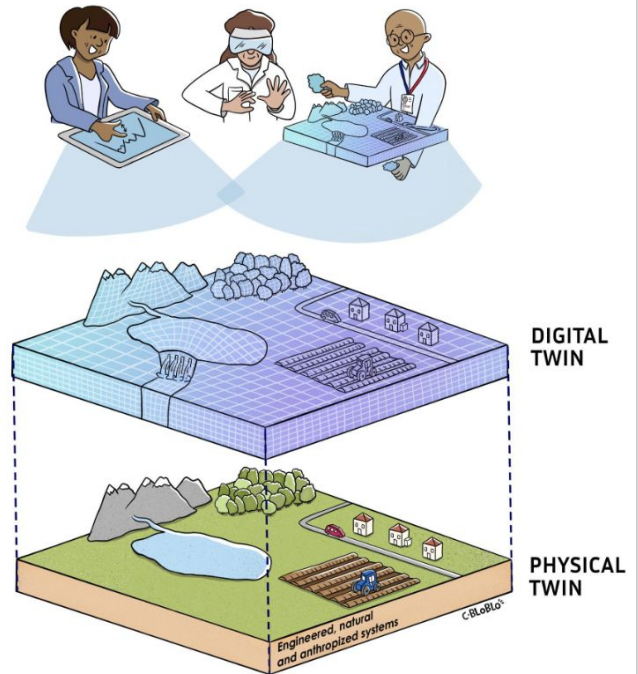
# Ongoing Research

# Digital Twins, Yet Another Definition...

*“A digital twin is a software system that purposefully represents an original, accurately reflects its changes, can act upon its original, and provides added value to users or other systems.”*

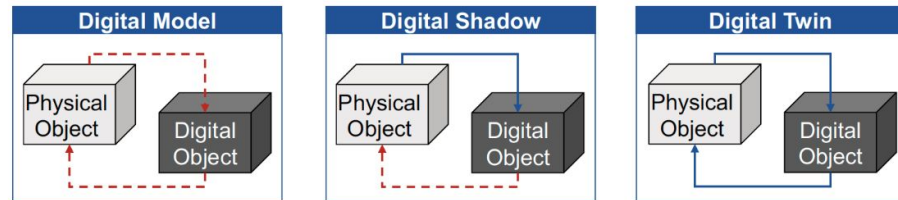
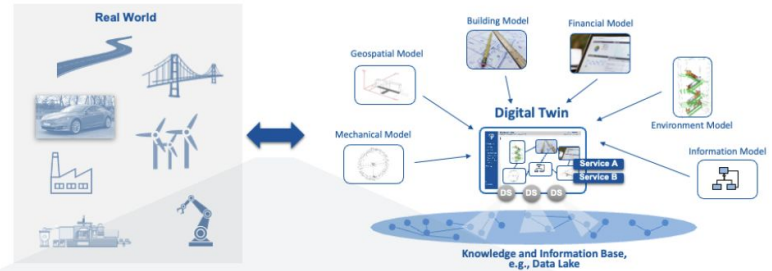
-- Michael, Rumpe, Wortmann, Combemale, Jézéquel

*In upcoming book...*



# Creating Digital Twins for...

... elements in the "real" world that can be sensed, monitored, actuated and controlled



W. Kritzinger, M. Karner, G. Traar, J. Henjes, W. Sihn: *Digital Twin in manufacturing: A categorical literature review and classification*. IFAC-PapersOnLine, 2018.

# Digital Twins: Engineering the V&V

Once DT validated:

- How to formalize the justification?
- How to ensure fidelity (continuous validation)?
- What is the diff equivalent in that situation
  - Original evolutions?
  - Twin evolutions?
  - Drift?

<http://fates-mlops.org/>  
<https://www.jpipe.org>  
<https://edtlab.fr/en/>

# Discussions time!



 <https://bit.ly/jmbruel>  
 @jmbruel

<https://bit.ly/jmb-talks>

Get the slides

