Modelling Justification Diagrams using jPipe

Sébastien Mosser et. al.
06/06/2024
MDENet Demonstration

My interests in Safety & Requirements

I'm not a safety engineer!

It takes a village...

Corinne Pulgar
2020-…

Deesha Patel
MEng, 2023

Nirmal Chaudhari
2023-…

Mireille Blay-Fornarino

Aaron Loh
BEng intern (2023)

Cass Braun
BEng intern (2024)

Kai Sun
MEng, 2024

Jean-Michel Bruel

... and some funding

- **DevOps for Software-Defined Networks**

  - MITACS/NSERC *industrial partnership*

  - *TL; DR:* **Understanding CI/CD pipelines**

- **RÉSO-T**: *Une approche innovante culturellement sensible pour intégrer le traitement automatique du langage naturel dans les outils de prévention du suicide*
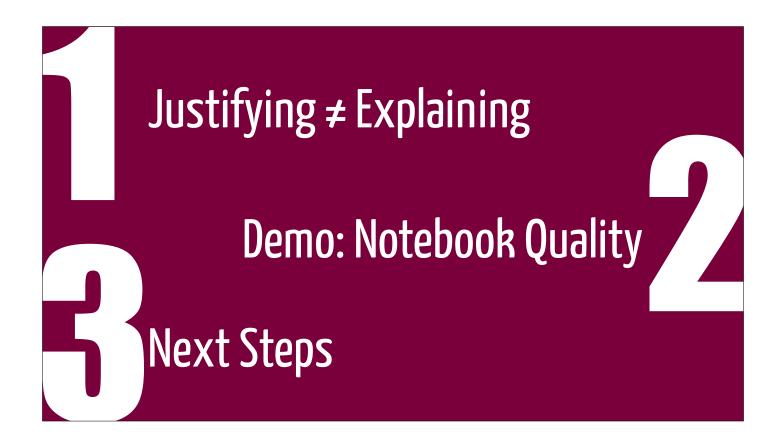
  - *Fonds de Recherche du Québec (**INNO**)*

  - *Domain:* **NLP for Mental health**

1 Justifying ≠ Explaining

2 Demo: Notebook Quality

3 Next Steps

Justifying ≠ Explaining

A common point between Mental Health and SDN?

1

Safety?

- **Definition** from Merriam-Webster:

  - *"The **condition of being safe** from undergoing or causing **hurt**, **injury**, or **loss**"*

- Lots of work on **safety-critical systems**:

  Anatomy of a Robotaxi Crash: Lessons from the Cruise Pedestrian Dragging Mishap

  Philip Koopman

  - Very **sharp** research **expertise**.

- What does it mean for a **Software Defined Network**?

- What does it mean for the **Mental Health Natural & Language Processing?**

Safety?

- **Definition** from Merriam-Webster:

  - *"The **condition of being safe** from undergoing or causing **hurt**, **injury**, or **loss**"*

- Lots of work on **safety-critical systems**:

  Anatomy of a Robotaxi Crash: Lessons from the Cruise Pedestrian Dragging Mishap

  Philip Koopman

  - Very **sharp** research **expertise**.

- What does it mean for a **Software Defined Network**?

  - **24-hour cell network** outage in Toronto, resulting in **911 not being reachable**

- What does it mean for the **Mental Health Natural & Language Processing?**

  - Your **most intimate thoughts are sent to an LLM** and used for training.

So we are not talking about Safety-Critical Systems

(and maybe not even mixed-criticality systems)

But that's not an excuse to ignore these situations!

(We need to a way to express and validate these "requirements")

Explaining is not Justifying

explain verb

Save Word

ex·plain | \ ik-ˈsplān \
explained; explaining; explains

"Documentation"

Essential Meaning of explain

1 : to make (something) clear or easy to understand
// I need a lawyer to *explain* this contract to me.
// The professor *explained* the poem to the class.
See More Examples

2 : to tell, show, or be the reason for or cause of something
// Scientists could not *explain* the strange lights in the sky.
// I don't know how to *explain* the dog's strange behavior.
See More Examples

justify verb

Save Word

jus·ti·fy | \ ˈjə-stə-ˌfī \
justified; justifying

"Actionable requirements validation"

Essential Meaning of justify

1 : to provide or be a good reason for (something) : to prove or show (something) to be just, right, or reasonable
// He tried to *justify* his behavior by saying that he was being pressured unfairly by his boss.
// The fact that we are at war does not *justify* treating innocent people as criminals.
// It's hard to *justify* the cost of a new car right now. = It's hard to *justify* spending money on a new car right now.

2 : to provide a good reason for the actions of (someone)
// Why should I have to **justify myself** [=to provide an explanation for my actions] when it was their fault?

McMaster University

"The experiment shall be reproducible".

- **Context**:
  - **Working with NLP researchers** since 2019
- **Worst** case:
  - **No info**. Fingers crossed, hope for the best.
- **Best** case:
  - Notes in a **meeting's report** (e.g., minutes, doc)
- **Real life**(c):
  - info encoded in **cryptic shell commands**/APIs

```
steps:
  - uses: actions/checkout@v4

  - name: Set up Python
    uses: actions/setup-python@v4
    with:
      python-version: 3.8

  - name: Install python script tool needed -> nbformat
    run: |
      python -m pip install --upgrade pip
      pip install nbformat

  - name: Check notebook file exists
    run: ./helper-scripts/check_file_exist.sh ${{ github.event.inputs.notebook_path }}

  - name: Verify all static paths are relative path
    run: python ./helper-scripts/check_relative_path.py ${{ github.event.inputs.notebook_path }}

  - name: Verify all dynamic paths are relative path
    run: |
      if [ "${{ github.event.inputs.oath }}" = "true" ]; then
        echo "user made an oath that the notebook does not contain absolute path"
      else
        echo "user doubt no absolute path used in the notebook"
        exit 1
      fi
```
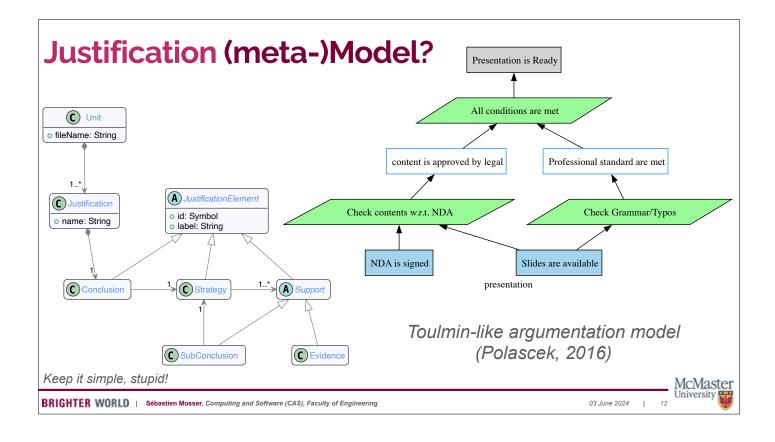
jupyter

We need to defend that the way we're building software is

Just, Right, Reasonable

Justification (meta-)Model?

Unit
- fileName: String

1..*

Justification
- name: String

1

JustificationElement
- id: Symbol
- label: String

Conclusion

Strategy

Support

SubConclusion

Evidence

Keep it simple, stupid!

Presentation is Ready

All conditions are met

content is approved by legal

Professional standard are met

Check contents w.r.t. NDA

Check Grammar/Typos

NDA is signed

Slides are available

presentation

*Toulmin-like argumentation model
(Polascek, 2016)*

McMaster University

Why Justif. Diagrams (JD) and not others?

- **Lots of work** exists in this realm (e.g., **GSN, WF+**)

  - Toulmin's work is from the 50s, after all.

- Extensive work on **Assurance Cases**:

  - "*A **structured set of arguments** and a body of evidence showing that **an information system satisfies specific claims** with respect to a given quality attribute.*" [NIST]

- Let's consider **JD** as "**poor man's language.**"

  - **Poor man** here being "*average soft. developer*"

  - *It's **not as powerful as safety/assurance cases**, but easier to apprehend.*

https://owl.purdue.edu/owl/general_writing/academic_writing/
historical_perspectives_on_argumentation/
toulmin_argument.html

McMaster University

Notebook Quality Demo

Intensively based on the work from Kai & Nirmal
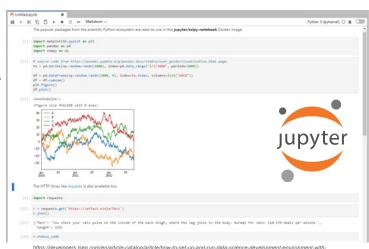
2

Notebook Quality?

- **Jupyter** Notebooks

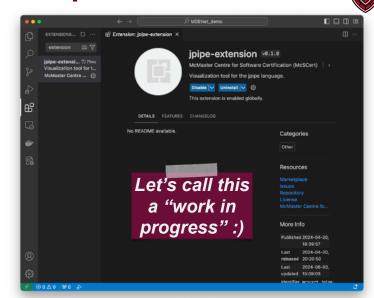 - *Literate Programming*

  - **Massively used** in ML/Data Science

- **Software Engineering support**:

  - **Best Practices**/Guidelines

  - **Linter** (-ish) tooling

- **Problem?**

  - *Can I share a given notebook?*

https://developers.lseg.com/en/article-catalog/article/how-to-set-up-and-run-data-science-development-environment-with-

Getting *jPipe* on your computer

- **Standalone compiler**

  - Java & **ANTLR**

- **VS Code Extension**

  - **Language server** using LEVER

  - **Basic LS support**:

    - *Syntax highlighting*

    - *Code completion*

    - *Fold/unfold*

Let's call this a "work in progress" :)

https://marketplace.visualstudio.com/items?itemName=mcscert.jpipe-extension

---
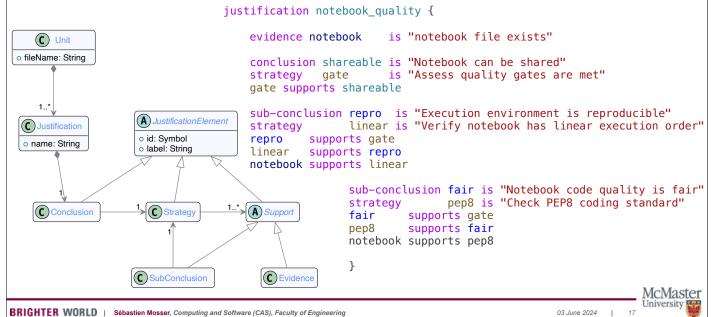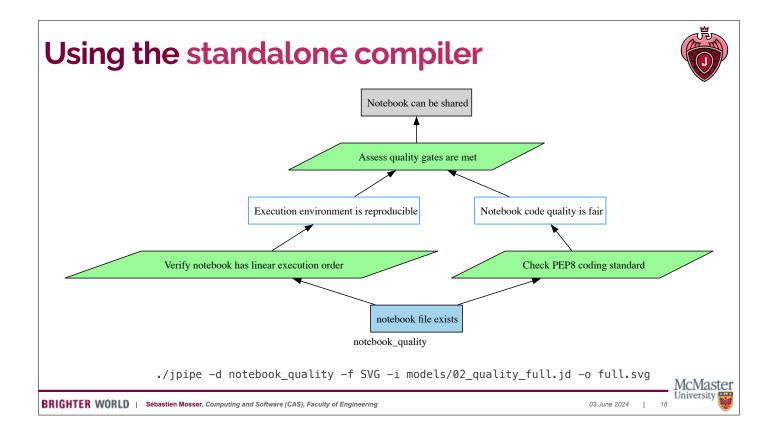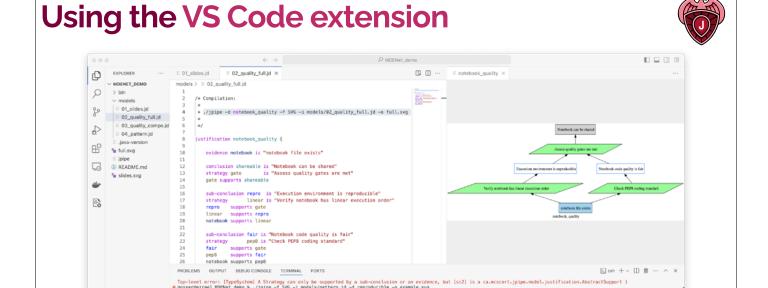
A **Textual DSL** to express **justifications models**


```
justification notebook_quality {

    evidence notebook     is "notebook file exists"

    conclusion shareable is "Notebook can be shared"
    strategy   gate       is "Assess quality gates are met"
    gate supports shareable

    sub-conclusion repro   is "Execution environment is reproducible"
    strategy          linear is "Verify notebook has linear execution order"
    repro     supports gate
    linear    supports repro
    notebook supports linear

        sub-conclusion fair is "Notebook code quality is fair"
        strategy       pep8 is "Check PEP8 coding standard"
        fair      supports gate
        pep8      supports fair
        notebook supports pep8

    }
```

Using the standalone compiler

./jpipe –d notebook_quality –f SVG –i models/02_quality_full.jd –o full.svg

Using the VS Code extension

Composability of models?

Reproducibility

Code Fairness

Notebook can be shared

Assess quality gates are met

Execution environment is reproducible

Notebook code quality is fair

Verify notebook has linear execution order

Check PEP8 coding standard

notebook file exists

notebook_quality

Automated composition makes sense here because of the "additive" nature of these models.

"Shareable" as composition of "reproducible" and "fair"

Notebook can be shared

Assess quality gates are met

Execution environment is reproducible

Verify notebook has linear execution order

notebook file exists

reproducible

⊕

Notebook can be shared

Assess quality gates are met

Notebook code quality is fair

Check PEP8 coding standard

notebook file exists

fair

composition **complete** *{ justification shareable is **reproducible** with fair }*
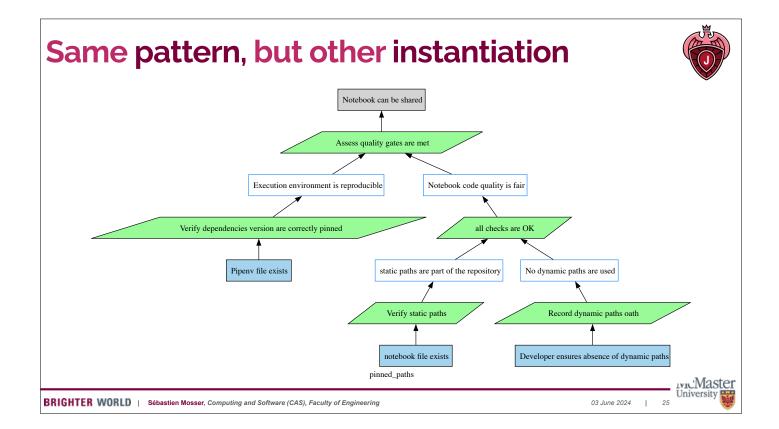
Argumentation Patterns

```
pattern shareable_P {

    conclusion c    is "Notebook can be shared"
    strategy   gate is "Assess quality gates are met"
    gate supports c

    @support repr is "Execution environment is reproducible"
    repr supports gate

    @support fair is "Notebook code quality is fair"
    fair supports gate

}
```

Off-the-shelf patterns to be used by developers

<<pattern>> shareable_P

One might need templates to structure models when purely additive composition does not make sense

Instantiating a pattern

pep8_linear

Instances only "fills in the blanks"

```
pattern shareable_P {

    conclusion c    is "Notebook can be shared"
    strategy  gate is "Assess quality gates are met"
    gate supports c

    @support repr is "Execution environment is reproducible"
    repr supports gate

    @support fair is "Notebook code quality is fair"
    fair supports gate
}
```

The compiler fails if the pattern is not instantiated correctly

```
justification pep8_linear implements shareable_P {

    sub-conclusion repr is "Execution environment is reproducible"
    strategy        s0   is "Verify notebook has linear execution order"
    s0 supports repr

    sub-conclusion fair is "Notebook code quality is fair"
    strategy        s2   is "Check PEP8 coding standard"
    s2 supports fair

    evidence e is "notebook file exists"
    e  supports s0
    e  supports s2
}
```

Assemble your elements in a software product line way!

Same pattern, but other instantiation

What's next?

this demo is one month too early :)
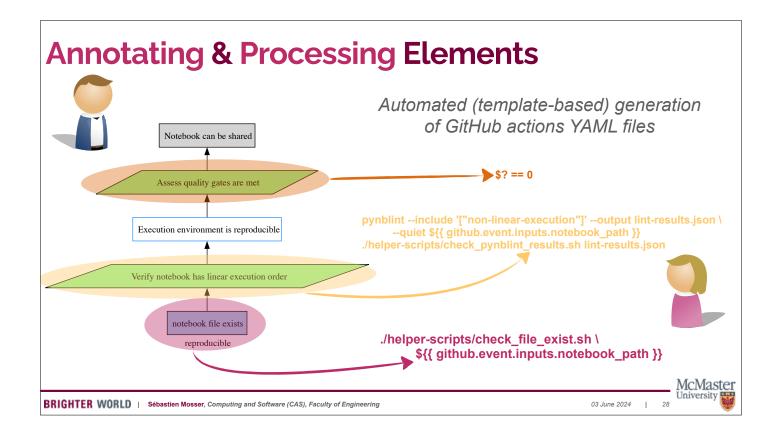
3

A **more extensible** version of jPipe

- The **current version** is "**hard-coded**."

  - *Given a JD file as input, it produces a diagram as output*

  - The **internal model is "hidden."**

- We're **refactoring** the compiler in a **"library"** way.

  - One can **interact with the compilation chain** and invent their **own processing**.

- One can **import jPipe in their own environment** and **integrate it with other tools**

  - *E.g., to support a software product line of best practices for notebooks*
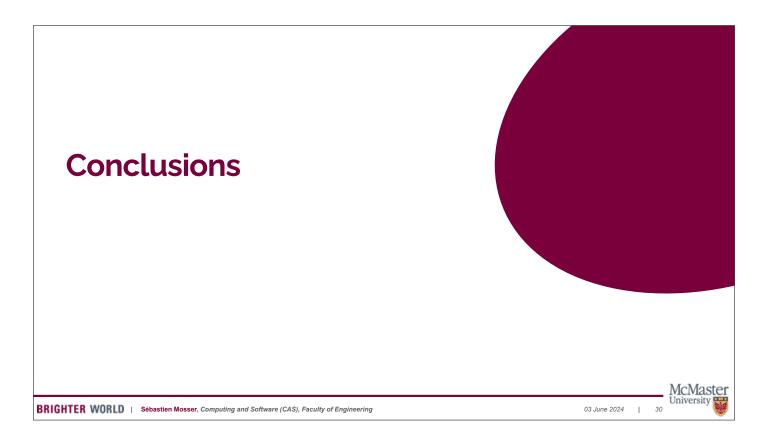
Annotating & Processing Elements

Automated (template-based) generation of GitHub actions YAML files

Notebook can be shared

Assess quality gates are met

$? == 0

Execution environment is reproducible

**pynblint --include '["non-linear-execution"]' --output lint-results.json \
--quiet ${{ github.event.inputs.notebook_path }}
./helper-scripts/check_pynblint_results.sh lint-results.json**

Verify notebook has linear execution order

notebook file exists

reproducible

**./helper-scripts/check_file_exist.sh \
${{ github.event.inputs.notebook_path }}**

An alternative Language Server Extension

- **The current version uses LEVER**

  - **Home-made** Language Server generator for **black-box compilers**

  - *Alexandre Lachance's Master thesis*

- **Question from Steffen**:

  - "*Wouldn't it be easier to retrofit your grammar into **Langium**?*"

- Well, **that's precisely what we're trying** in this alternative version

  - Spoiler: **it's... not that simple!**

Conclusions

Takeaway Message

- **Justification Diagram** as a "**first step towards redemption**" for engineers

  - **Make explicit why things are done**

  - **Integrating "safety" (-ish) practices in non safety-critical systems**

- Example of application domain: **Scientific Notebook development**

  - **Best practices** to ensure the quality of notebooks **from literature**

  - Organize them as **composable models/patterns** in jPipe

- **More exciting stuff to come**!

  - Winter was more hectic than anticipated. But **soon, a new release**!

Interested in jPipe-like challenges?

- **Contact me!**

  - Email: **mossers@mcmaster.ca**

- **Some elements to read:**

  - *Yes, Configuring is Good, But Have You Ever Tried Justifying?* Sébastien Mosser, Corinne Pulgar, Mireille Blay-Fornarino, Deesha Patel, Aaron Loh, Jean-Michel Bruel. CONFLANG 2023 (https://macsphere.mcmaster.ca/handle/11375/28858)

  - *Eat your own DevOps: a model driven approach to justify continuous integration pipelines*. Corinne Pulgar. ACM SRC Competition MODELS 22 *(Bronze medal)*

  - *A Study on Justifying Platform-Independent CI/CD Pipelines*. Deesha Patel, MEng report (McMaster). 2023 https://macsphere.mcmaster.ca/bitstream/11375/29326/1/patel_deesha_meng.pdf

  - *A Study on Justification on Jupyter Notebook quality and Fairness*, Kai Sun, MEng report (McMaster). 2024. https://macsphere.mcmaster.ca/bitstream/11375/29641/2/Meng_Report_KaiSun_2024.03.pdf

---

We are hiring!

- **We're (always) looking for**:

  - *Faculty*

  - *Postdoctoral fellows*

  - *PhD students*

  - *Master students*

  - *Summer interns*

- **Topics**:

  - Lots of **cool stuff** on **model-driven** & **software** engineering

McMaster University

ENGINEERING

Computing & Software

Dr. Sébastien Mosser
Associate Professor

mossers@mcmaster.ca
https://mosser.github.io

ITB 131 (appointment only)