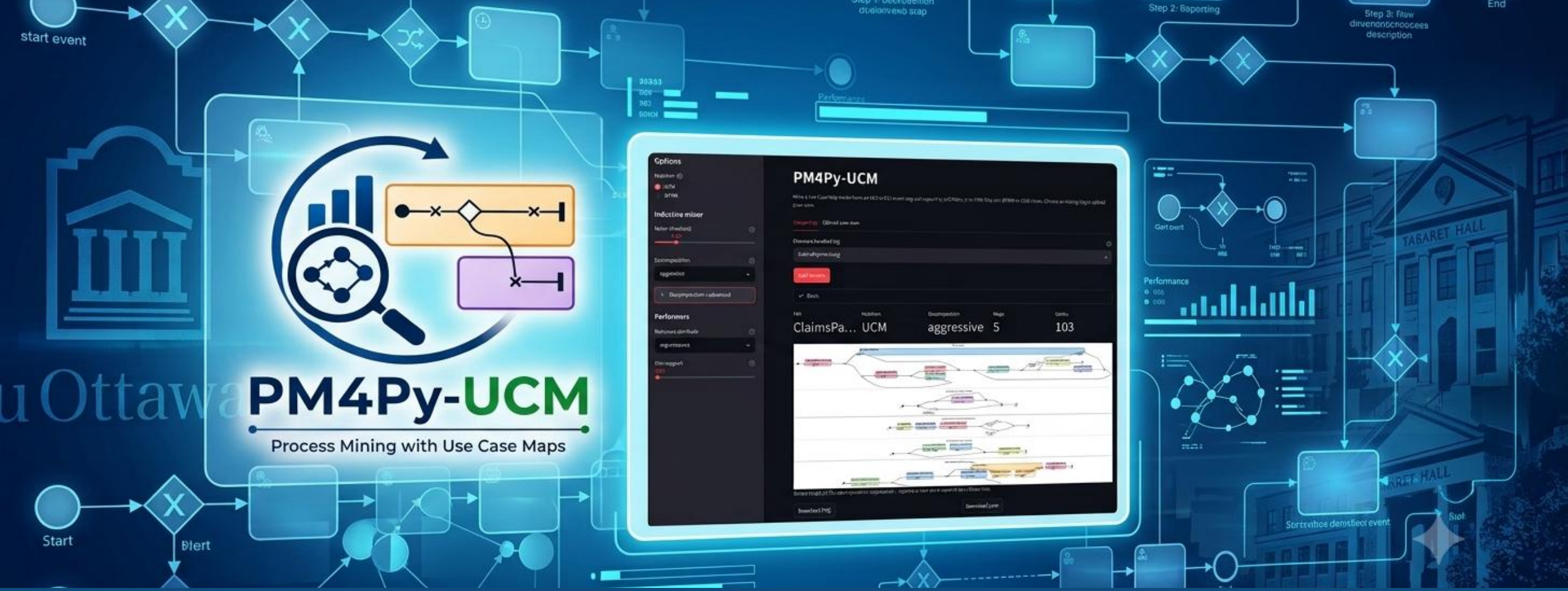
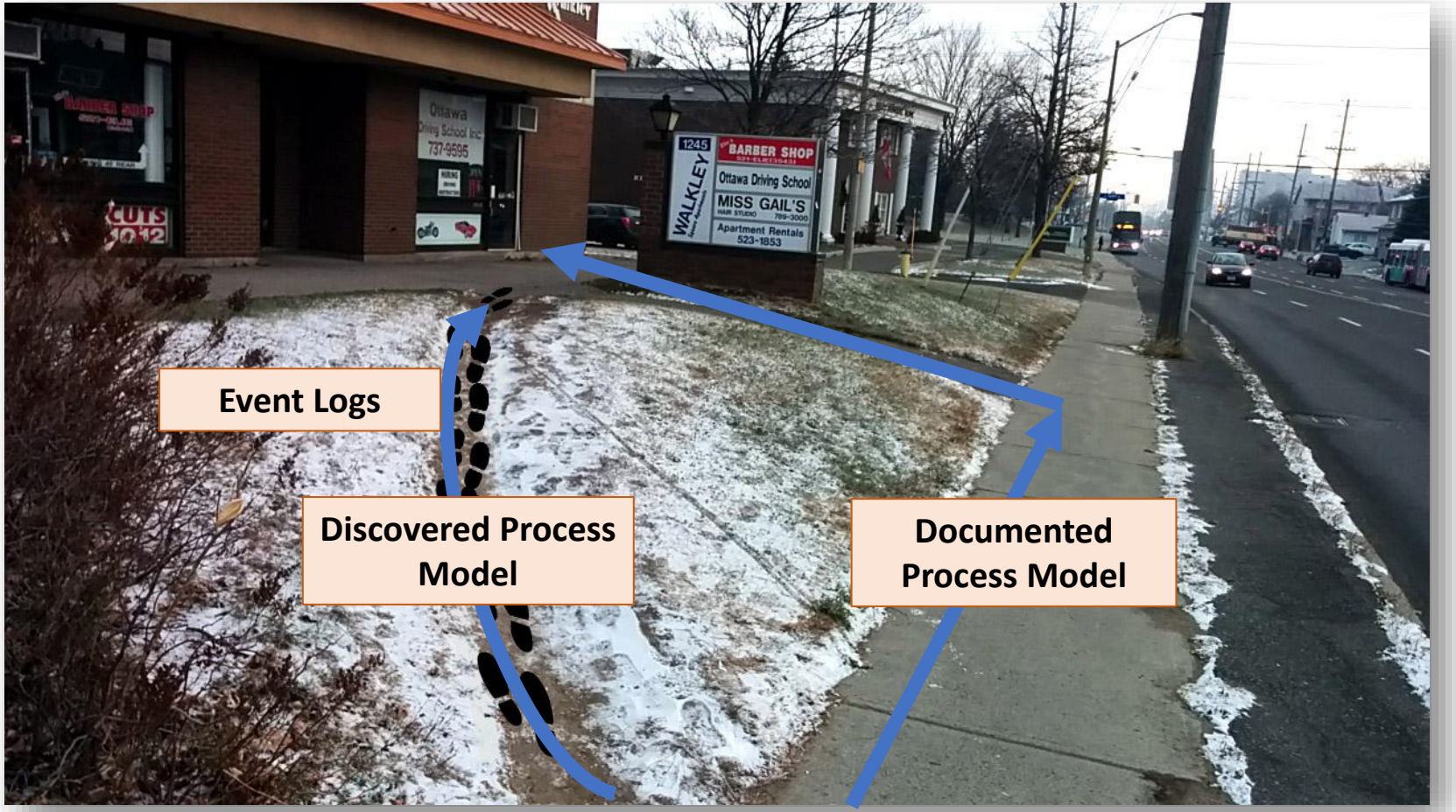


Towards Process Mining Use Case Map Models with PM4Py-UCM

Daniel Amyot · **MoDRE 2026, Montréal, Canada** · damyot@uottawa.ca

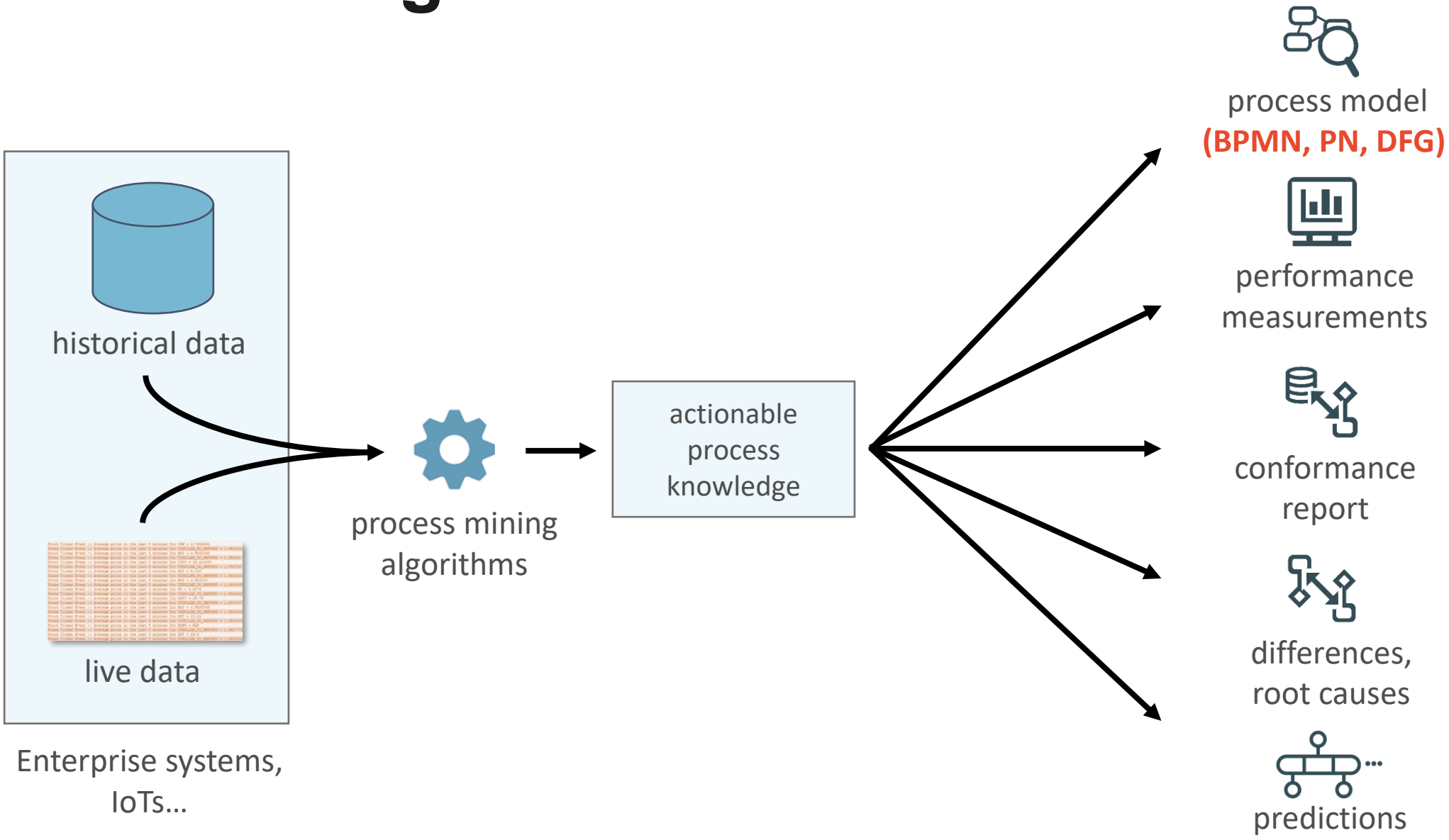
uOttawa

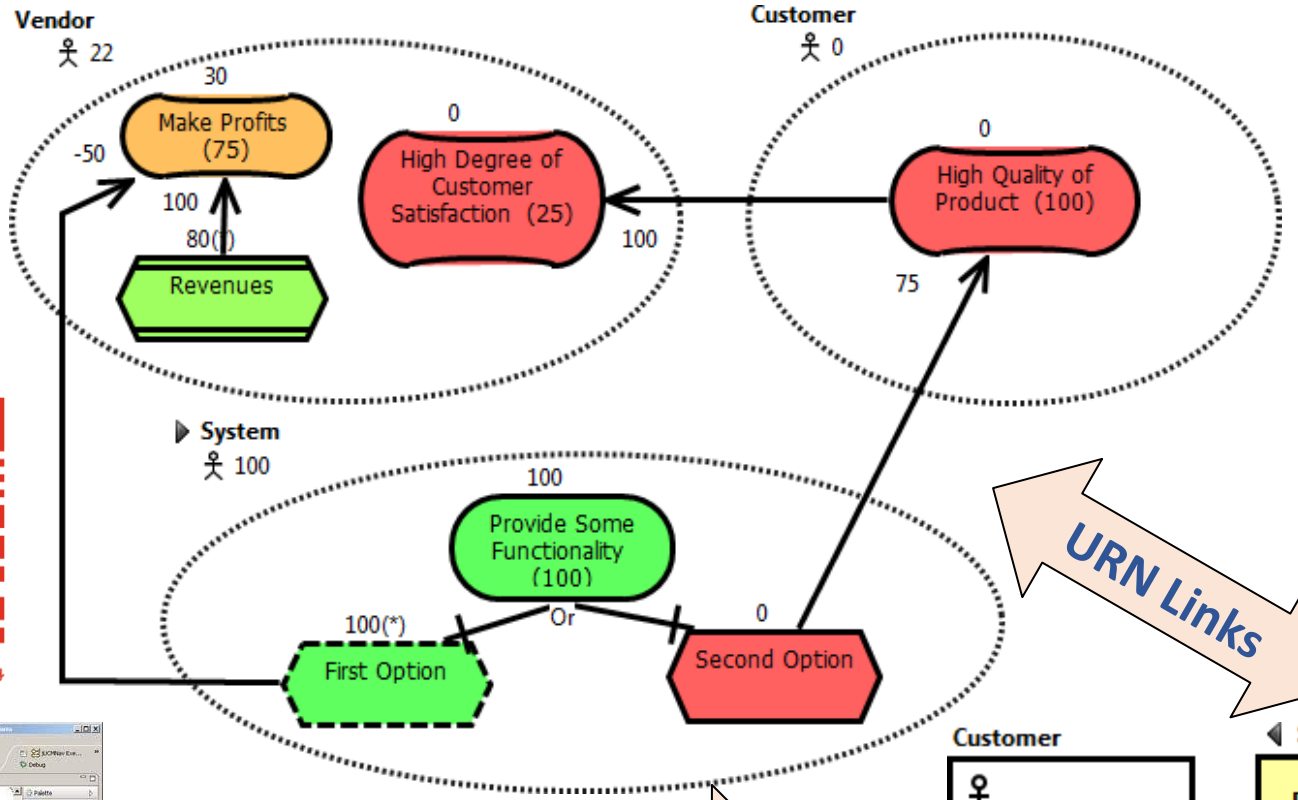
What Process Mining: Data-Driven Modelling!



Credit: Dr. Mahdi Ghasemi

Process Mining Overview



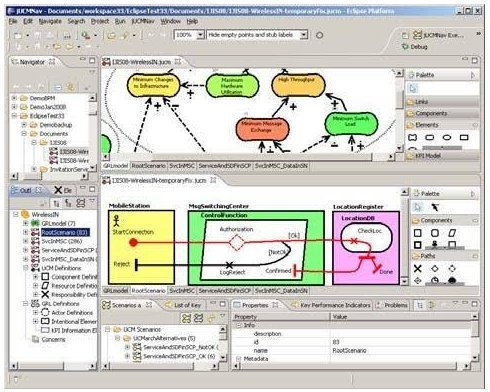


GRL

Goal-oriented Requirement Language (GRL)

intentional elements + actors + links+ indicators + strategies

URN Links



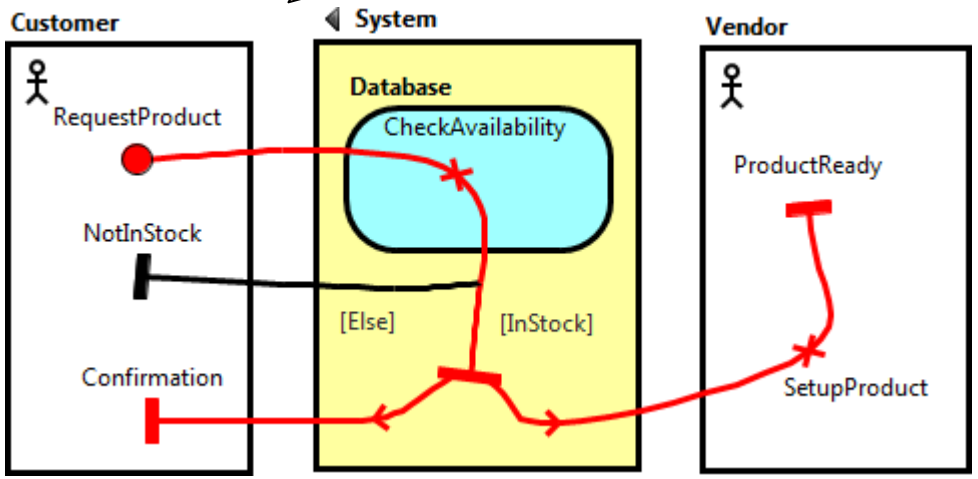
jUCMNav

https://bit.ly/jUCMNavPlus

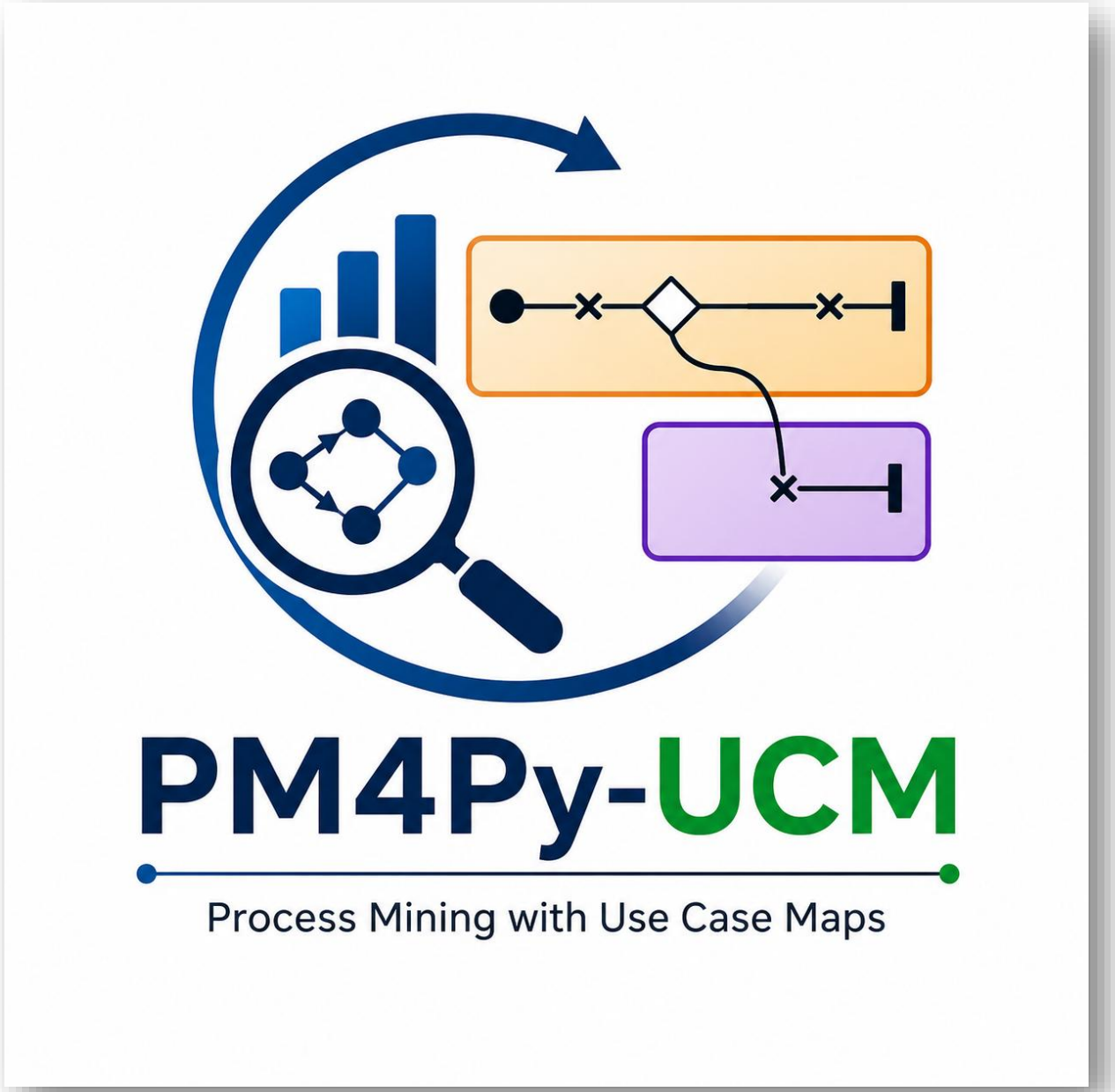
UCM

Use Case Map (UCM)

responsibilities + causality + components + scenarios



ITU-T Rec. Z.151 (2018), **User Requirements Notation**, https://itu.int/rec/T-REC-Z.151/en


PM4Py-UCM
Process Mining with Use Case Maps

PM4Py-UCM

v0.7.10. Daniel Amyot, uOttawa, 2026

Inductive miner

Noise threshold 0.20

Decomposition

Decomposition: auto

- Advanced — boundary rules & sizes
- Performers
- Performance overlay

Notation (Model tab)

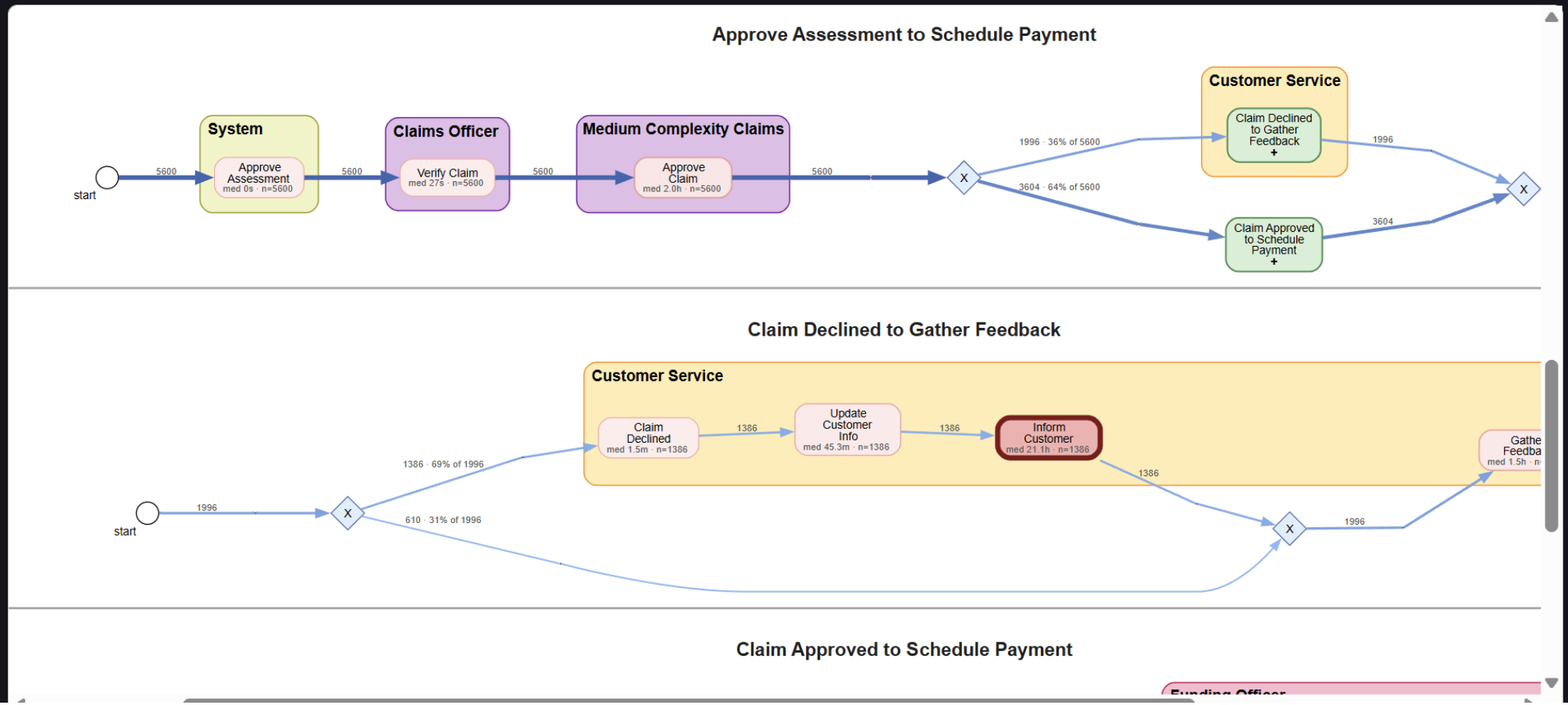
- UCM
- BPMN

Rename activities…

Activities	Cases	Events	Maps	Nodes
25	5,600	78,126	6	107

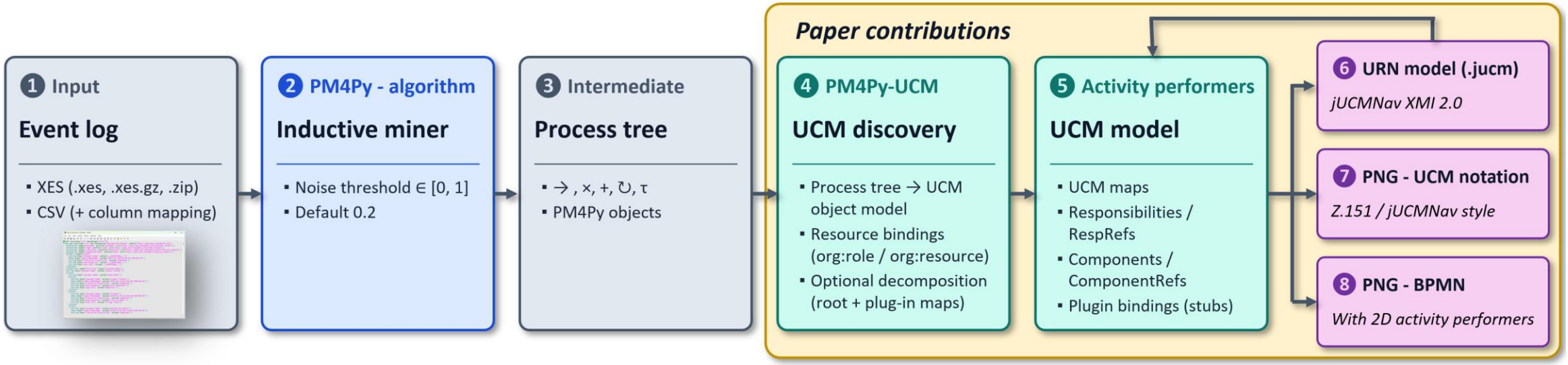
4,990 of 5,600 cases (89%) fit this model exactly. The other 610 are counted on their closest path through the model, so the numbers cover the whole log — for an activity the model treats as mandatory that can read higher than the events actually observed.

Diagram height (px): 620



What PM4Py-UCM does

PM4Py-UCM makes UCM / URN (ITU-T Z.151) a first-class output…
The first log-driven entry point into the URN world!



Components

The “who”, as a 2-D hierarchy

Scenarios

Definitions (variant-based or data-driven) that actually execute

Decomposition

Navigable hierarchy of sub-processes, to manage complexity

Standardized

Round-trips with jUCMNav

Decomposition: stubs and plug-ins

A flat mined map can be huge and messy…
Structural decomposition into a tree of sub-processes can help cope with complexity!

Where it cuts

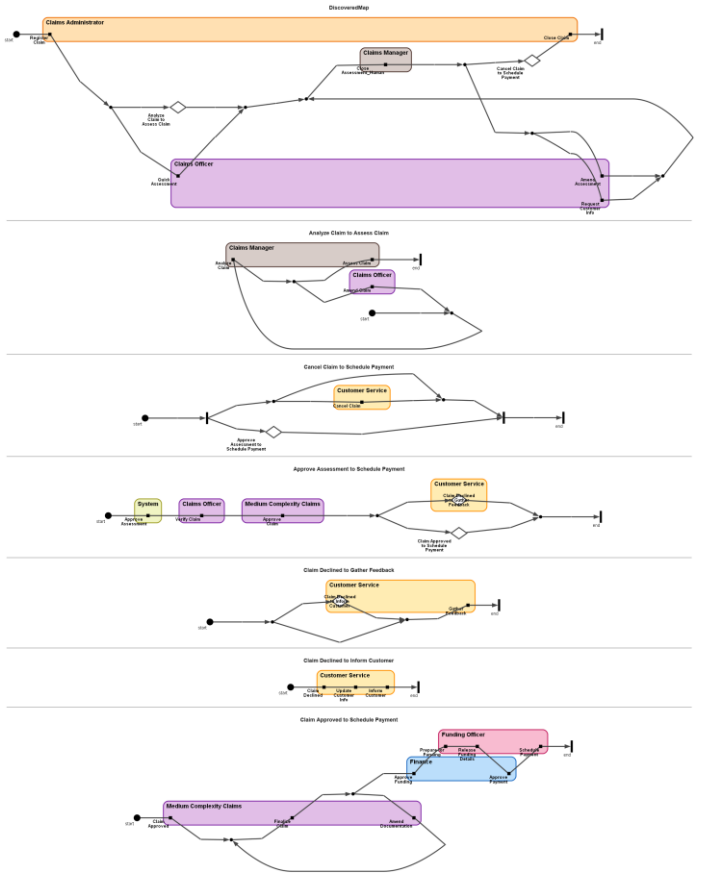
- Root sequence · parallel block · alternative · loop
- Size guards: min / max leaves per map, balance ratio

How you drive it

- Presets: auto · aggressive · custom min/max per view
- Shape-fitted “auto” picks the cut from the tree itself

Why it is safe

- Synthesis is decomposition-agnostic: tree nodes pair with UCM nodes by stable IDs, not position



88

nodes on
one flat map

↓

7

readable maps

Python API

```
import pm4py
import pm4py_ucm

log = pm4py.read_xes("log.xes")
ucm = pm4py_ucm.discover_ucm_inductive(log,
          decomposition="aggressive")

pm4py_ucm.save_vis_ucm(ucm, "diagram_ucm.png")
pm4py_ucm.write_ucm(ucm, "log.jucm")
```

```
# One-shot: mine and bind in a single call.
ucm = pm4py_ucm.discover_ucm_inductive(log,
          parameters={"resource_attribute":
                ["org:role", "org:resource"],
})
pm4py_ucm.write_ucm(ucm, "log.jucm")
```

Concurrency-aware variants

Classic variants split behaviourally identical cases apart for simple interleaving…
PM4Py-UCM clusters variants on sequence and concurrency: Fewer and more meaningful variants!

Process tree: $X \rightarrow (Y \parallel Z) \rightarrow (A \times B) \rightarrow W$

4 sequence variants

- X→Y→Z→A→W (60)
- X→Y→Z→B→W (25)
- X→Z→Y→A→W (10)
- X→Z→Y→B→W (5)

→

2 concurrency-aware variants

- v1 X→(Y‖Z)→A→W (70)
- v2 X→(Y‖Z)→B→W (30)

interleavings merged · 100% fitness · choices preserved

13.6×
fewer variants — Road Traffic (17 from 231)

9.1×
fewer variants — Claims Payment (18 from 164)

2183×
BPI 2012 — too good: coarsening collapses a deep loop nest

Two ways to explain a branch's conditions

Every OR-fork needs a condition the traversal engine can resolve…
Two interchangeable encodings, with genuinely different strengths

Variant-driven

- A synthetic variant_id enum; each branch is guarded by the variants that took it
- Faithful by construction — replays every variant exactly
- Needs no attributes at all
- But: says nothing about the domain, and guards grow with the variant count

Data-driven

- A depth-3 decision tree per fork over case attributes
- Emits business-readable Boolean rules in jUCMNav grammar
- A 5-rule Boolean minimizer keeps them short
- But: not guaranteed to reproduce the variant's own branch

A real mined condition — expense claims

`Amount > 615` → `Manual review`

*…and the minimizer collapsed a 5-clause Road Traffic condition to **true**, hence reporting that the tree found no signal on that branch.*

Executable scenarios

One scenario definition per variant, with typed variables and initializations…
Plus the loop machinery that makes a mined model runnable at all.

Termination

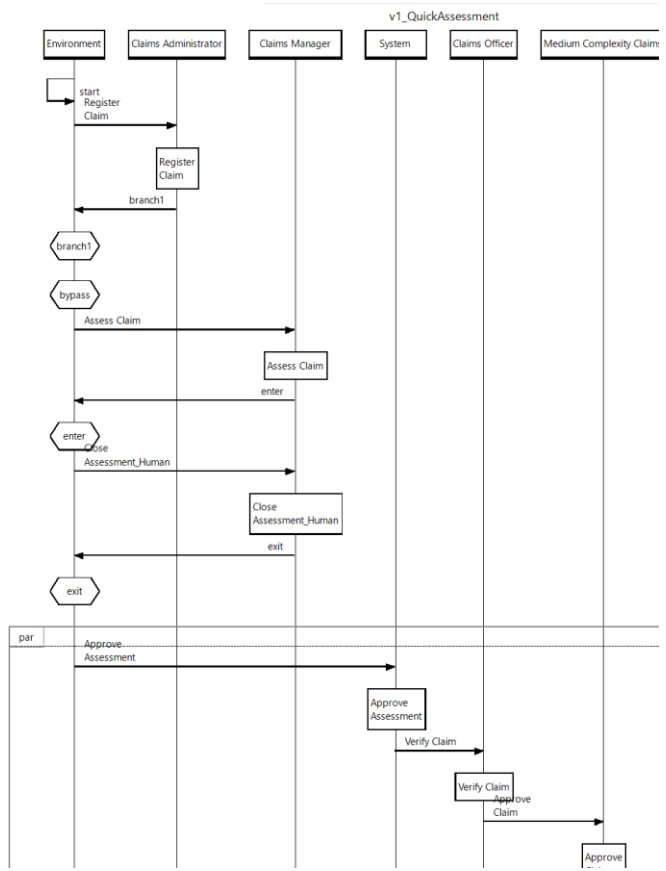
- Per-loop integer counter; redo guarded by v > 0, exit by v ≤ 0
- Decrement sits at the end of the body; no path skips it

Zero iterations

- A LoopEntryGuard bypasses the loop when the counter starts at 0
- Counter = exact number of body executions

Covering choices inside loops

- Each branch gets a contiguous slice of iterations, sized from the tree so none is undemonstrable

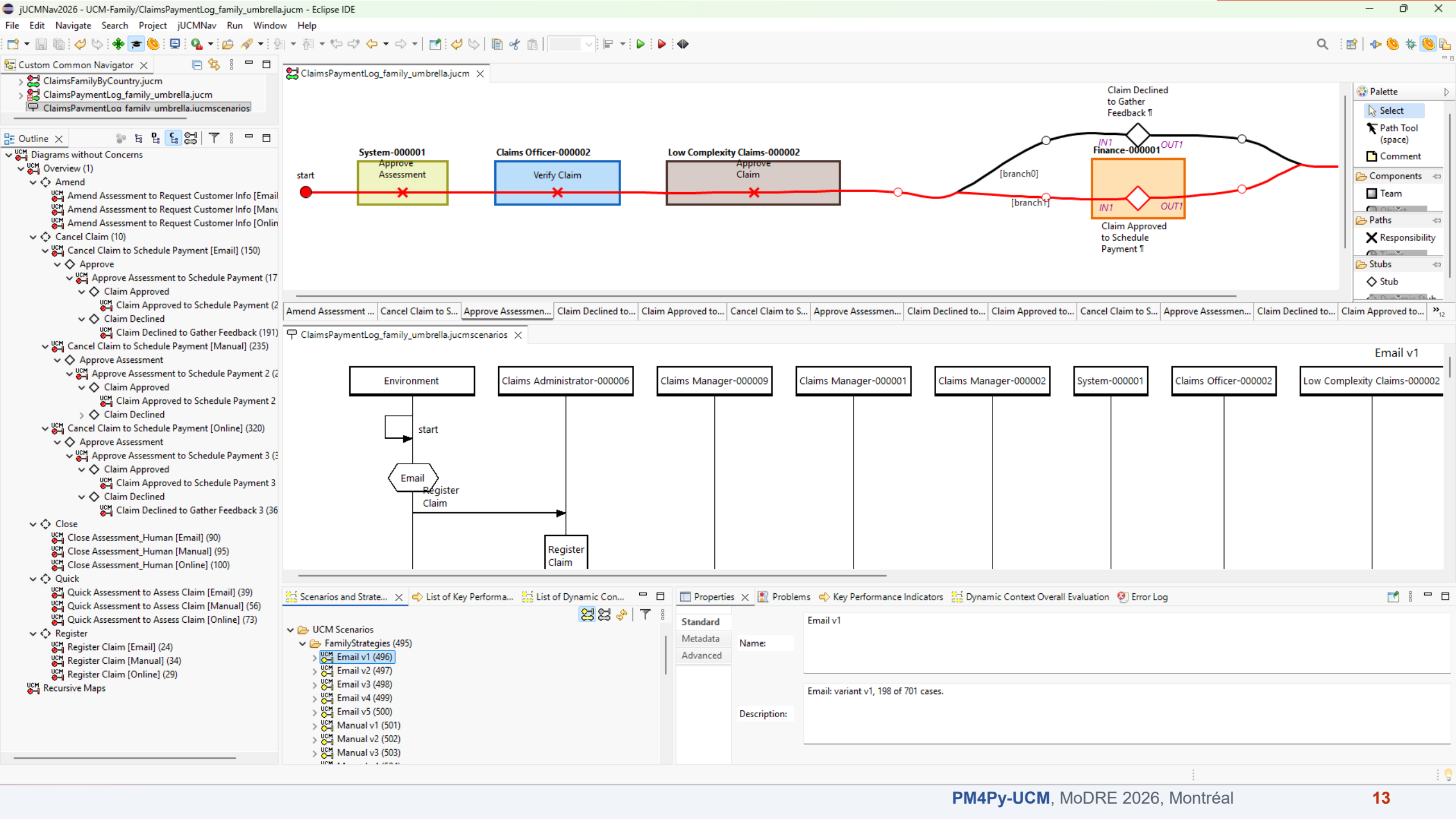


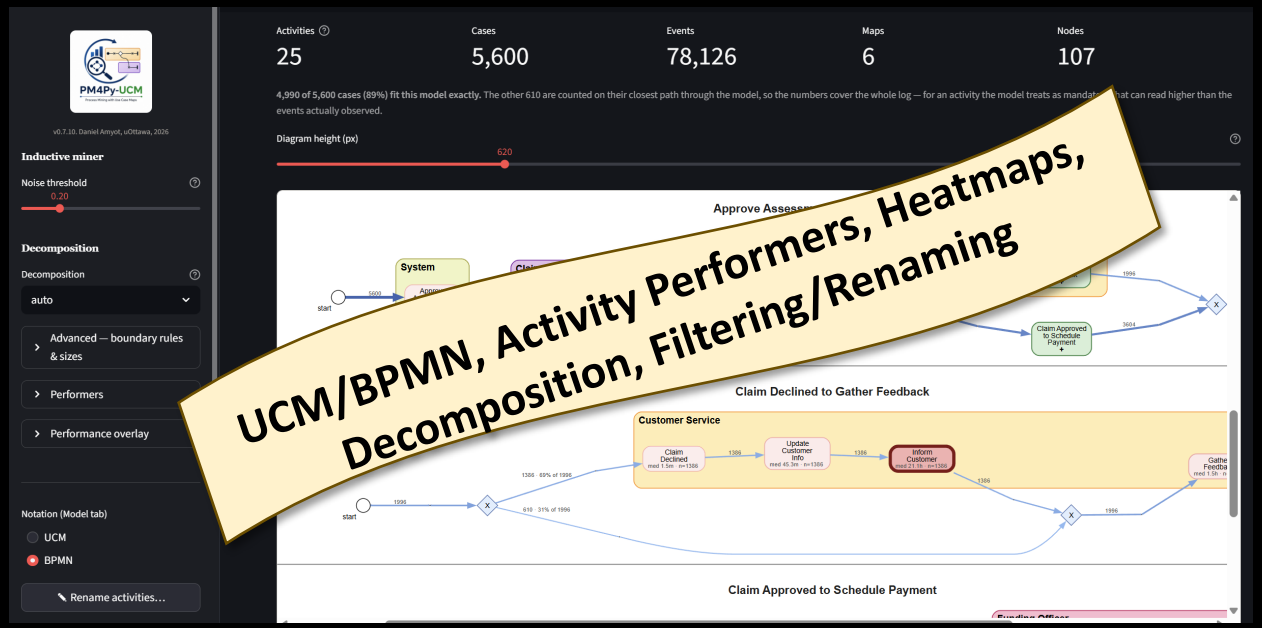
Import to jUCMNav

execute (red paths)

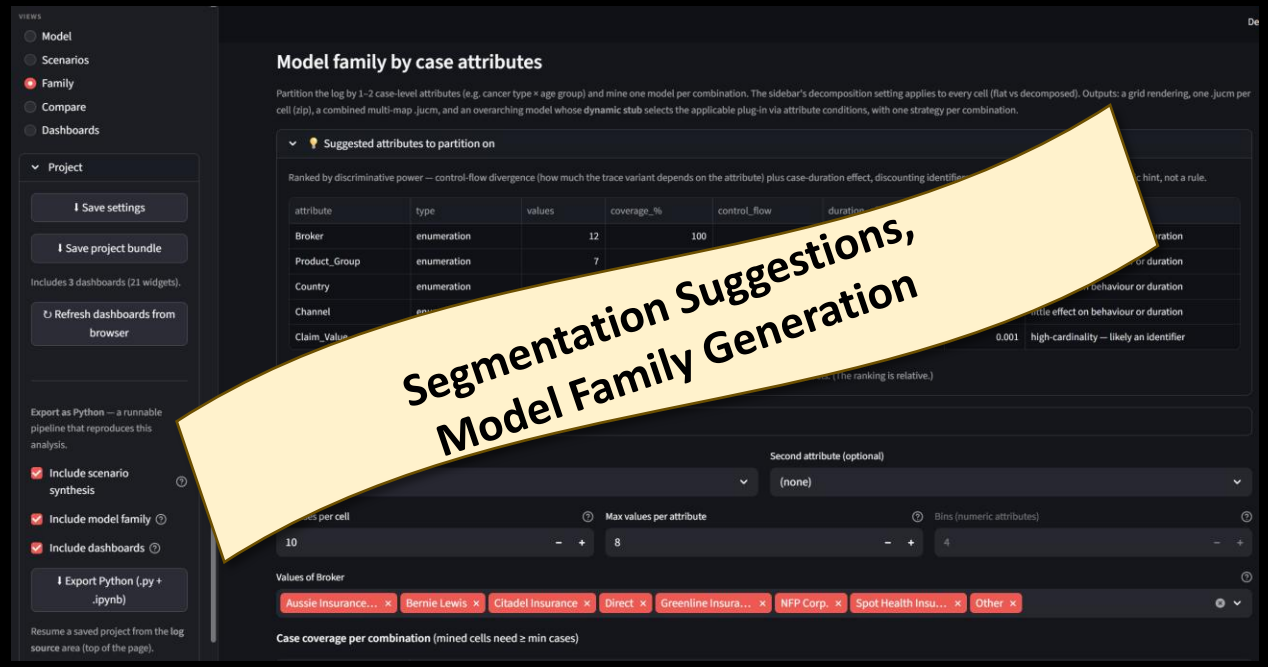
export MSCs

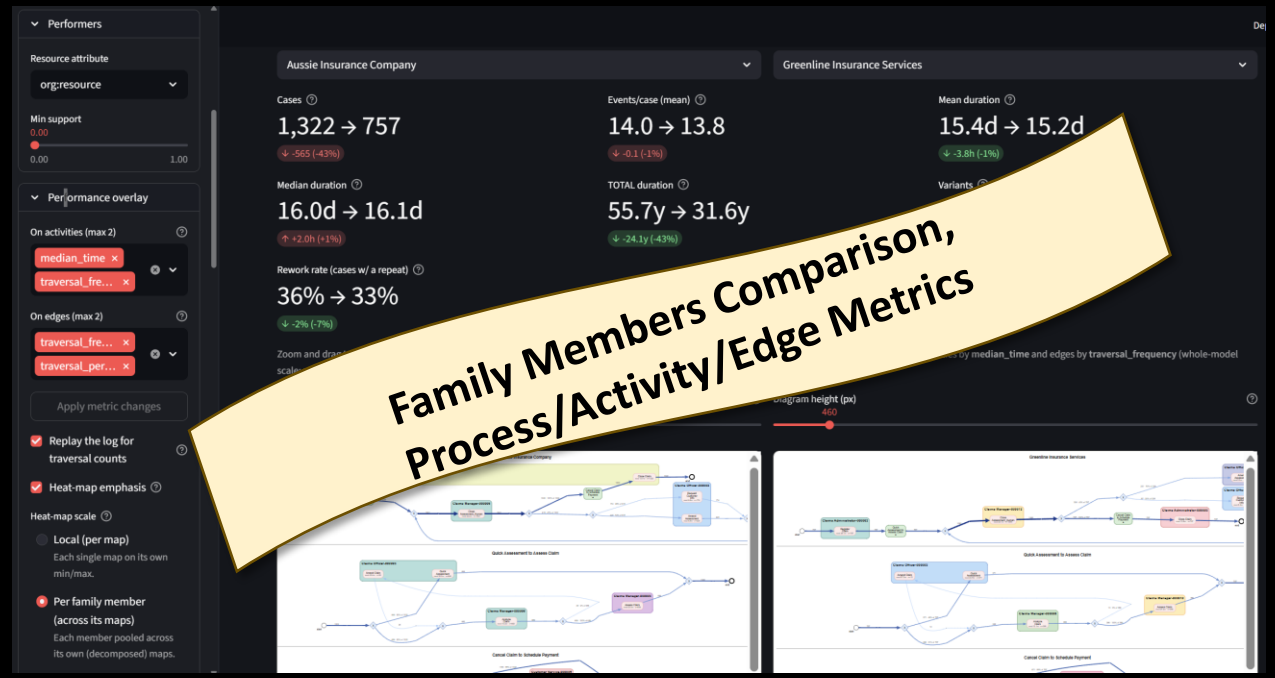
generate tests


jUCMNav2026 - UCM-Family/ClaimsPaymentLog_family_umbrella.jucm - Eclipse IDE
File Edit Navigate Search Project jUCMNav Run Window Help
Custom Common Navigator
ClaimsFamilyByCountry.jucm
ClaimsPaymentLog_family_umbrella.jucm
ClaimsPaymentLog_family_umbrella.jucmscenarios
Outline
Diagrams without Concerns
Overview (1)
Amend
Amend Assessment to Request Customer Info [Email]
Amend Assessment to Request Customer Info [Manu
Amend Assessment to Request Customer Info [Onlin
Cancel Claim (10)
Cancel Claim to Schedule Payment [Email] (150)
Approve
Approve Assessment to Schedule Payment (17
Claim Approved
Claim Approved to Schedule Payment (2
Claim Declined
Claim Declined to Gather Feedback (191)
Cancel Claim to Schedule Payment [Manual] (235)
Approve Assessment
Approve Assessment to Schedule Payment 2 (2
Claim Approved
Claim Approved to Schedule Payment 2
Claim Declined
Cancel Claim to Schedule Payment [Online] (320)
Approve Assessment
Approve Assessment to Schedule Payment 3 (3
Claim Approved
Claim Approved to Schedule Payment 3
Claim Declined
Claim Declined to Gather Feedback 3 (36
Close
Close Assessment_Human [Email] (90)
Close Assessment_Human [Manual] (95)
Close Assessment_Human [Online] (100)
Quick
Quick Assessment to Assess Claim [Email] (39)
Quick Assessment to Assess Claim [Manual] (56)
Quick Assessment to Assess Claim [Online] (73)
Register
Register Claim [Email] (24)
Register Claim [Manual] (34)
Register Claim [Online] (29)
Recursive Maps
ClaimsPaymentLog_family_umbrella.jucm
start
System-000001
Approve Assessment
Claims Officer-000002
Verify Claim
Low Complexity Claims-000002
Approve Claim
[branch0]
[branch1]
Claim Declined to Gather Feedback ¶
IN1
OUT1
Finance-000001
Claim Approved to Schedule Payment ¶
Palette
Select
Path Tool (space)
Comment
Components
Team
Paths
Responsibility
Stubs
Stub
Amend Assessment ...
Cancel Claim to S...
Approve Assessmen...
Claim Declined to...
Claim Approved to...
ClaimsPaymentLog_family_umbrella.jucmscenarios
Email v1
Environment
Claims Administrator-000006
Claims Manager-000009
Claims Manager-000001
Claims Manager-000002
System-000001
Claims Officer-000002
Low Complexity Claims-000002
start
Email
Register Claim
Register Claim
Scenarios and Strate...
List of Key Performa...
List of Dynamic Con...
UCM Scenarios
FamilyStrategies (495)
Email v1 (496)
Email v2 (497)
Email v3 (498)
Email v4 (499)
Email v5 (500)
Manual v1 (501)
Manual v2 (502)
Manual v3 (503)
Properties
Problems
Key Performance Indicators
Dynamic Context Overall Evaluation
Error Log
Standard
Metadata
Advanced
Name:
Email v1
Description:
Email: variant v1, 198 of 701 cases.

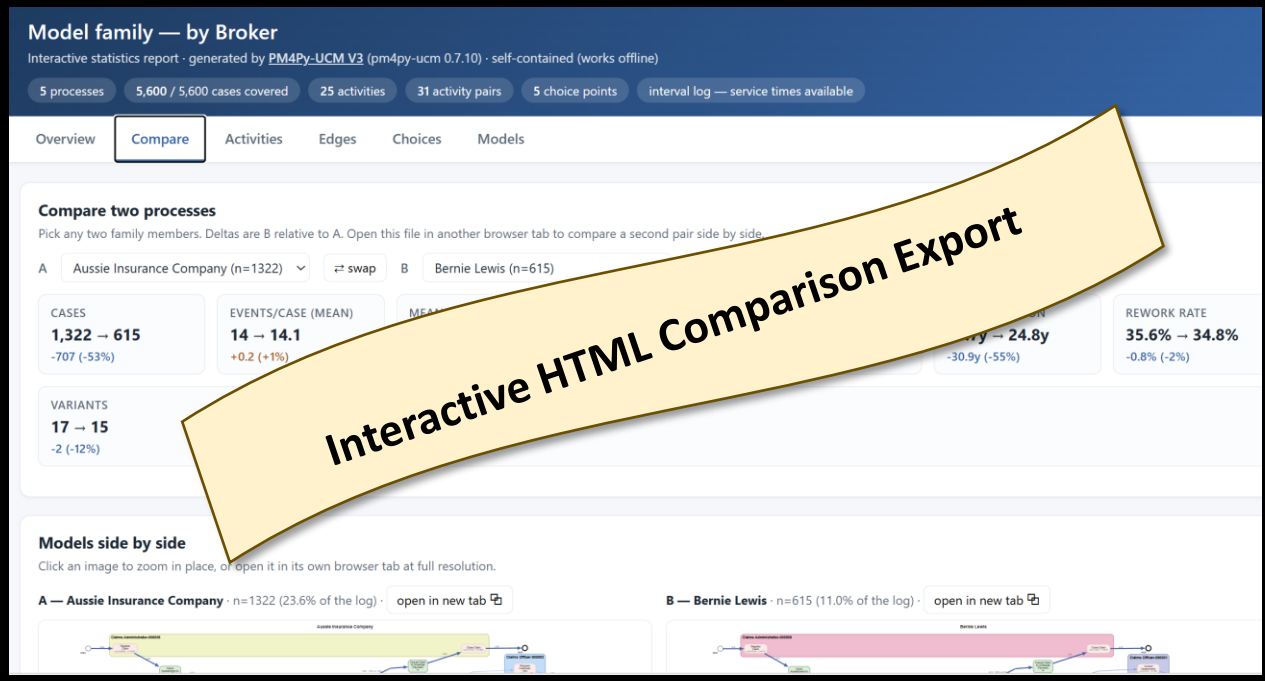

PM4Py-UCM
Inductive miner
Noise threshold
Decomposition
Decomposition
auto
Advanced — boundary rules & sizes
Performers
Performance overlay
Notation (Model tab)
UCM
BPMN
Rename activities…
Activities
25
Cases
5,600
Events
78,126
Maps
6
Nodes
107
Diagram height (px)
Claim Declined to Gather Feedback
Customer Service
Claim Approved to Schedule Payment
UCM/BPMN, Activity Performers, Heatmaps,
Decomposition, Filtering/Renaming

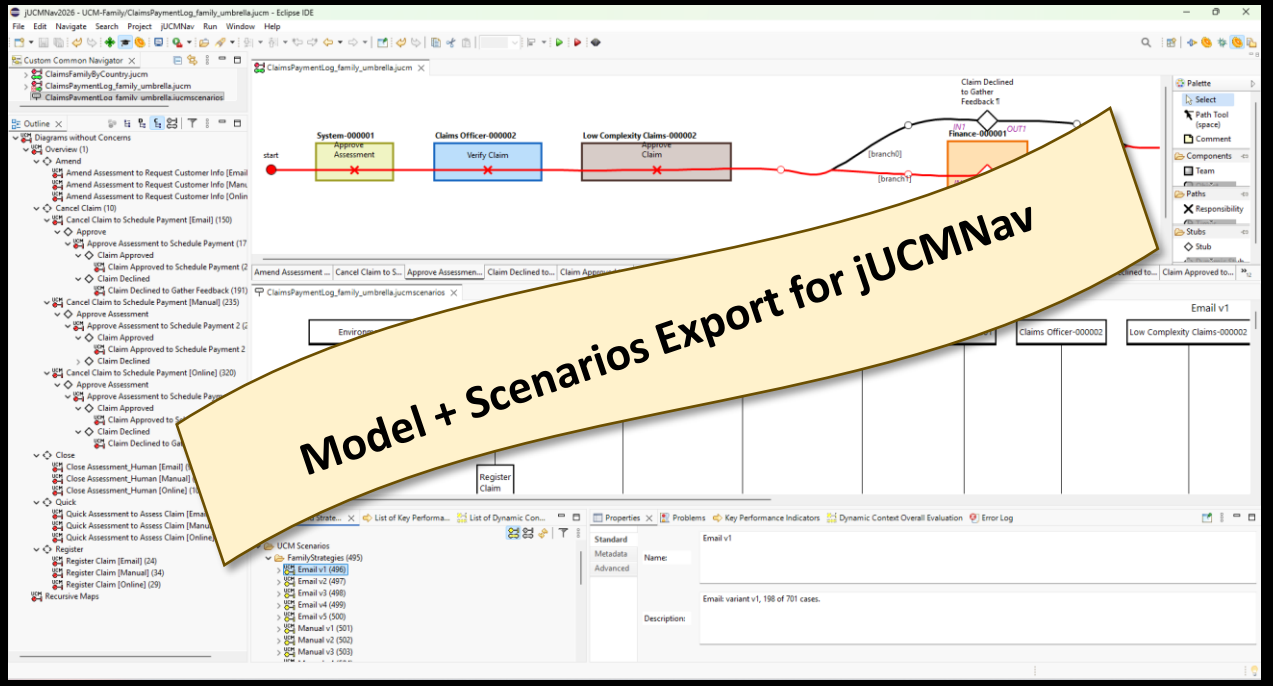

Date range
Cycle-time percentile (case duration)
Variants by frequency rank
…or top variants by case coverage (%)
ClaimsPaymentLog.zip
Model
Scenarios
Family
Compare
Dashboards
Project
Save settings
Save project bundle
Synthesize executable scenarios
Condition strategy
variant
data-driven
max_loop_iterations
Scenario group name
MinedScenarios
Synthesize scenarios
Variants
18
18
Concurrency-Aware Variants,
Scenario Generation, jUCMNav Export


Model
Scenarios
Family
Compare
Dashboards
Project
Save settings
Save project bundle
Refresh dashboards from browser
Export as Python — a runnable pipeline that reproduces this analysis.
Include scenario synthesis
Include model family
Include dashboards
Export Python (.py + .ipynb)
Model family by case attributes
Suggested attributes to partition on
Second attribute (optional)
(none)
Values of Broker
Case coverage per combination (mined cells need ≥ min cases)
Segmentation Suggestions,
Model Family Generation


Performers
Resource attribute
org:resource
Min support
Performance overlay
On activities (max 2)
median_time
On edges (max 2)
Apply metric changes
Replay the log for traversal counts
Heat-map emphasis
Heat-map scale
Local (per map)
Per family member (across its maps)
Aussie Insurance Company
Greenline Insurance Services
Cases
1,322 → 757
Events/case (mean)
14.0 → 13.8
Mean duration
15.4d → 15.2d
Median duration
16.0d → 16.1d
TOTAL duration
55.7y → 31.6y
Variants
Rework rate (cases w/ a repeat)
36% → 33%
Diagram height (px)
Family Members Comparison,
Process/Activity/Edge Metrics


Dashboard Creation & HTML Export

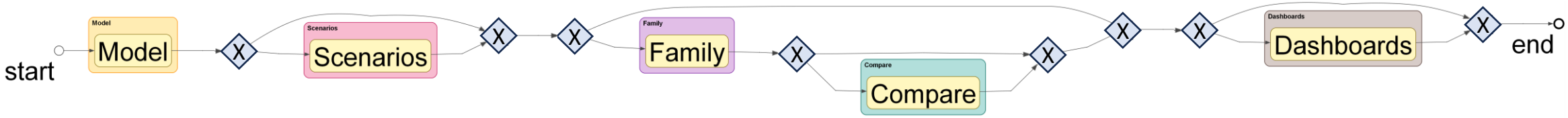

Model family — by Broker
Interactive HTML Comparison Export


Model + Scenarios Export for jUCMNav


Python Script & JupyterLab Notebook Export

The tool’s five views, mined as a process

The model below was produced by PM4Py-UCM from a log of 192 analyst sessions in its own workspace; each view is a component, each session a path through them.



192	21	12	5
analyst sessions	things you can do	distinct view journeys	views, as components

Every combination is a use case

- Model alone, or Model → Scenarios
- Model → Family → Compare, and every mix

12 of 16 — Compare without Family never happens

Mined, not drawn

- Same miner, converter,.jucm export
- Views become components via org:role

79 nodes, 21 responsibilities, 5 components

Everything else the tool does

Capabilities a mature process-mining tool is expected to have, and many distinctive to PM4Py-UCM

Expected

- Log **filtering**: activities, attributes, frequency, variant frequency, date ranges, duration percentile, top variants by coverage
- Activity **renaming**, with **import / export** of the map
- Project **saving**, sharing, and resuming
- Performance overlays and metrics **visual heat-maps** on the models
- **Dashboard** creation with KPIs

Distinctive

- Log-driven **UCM discovery** — the first of its kind
- Performer-aware **component binding** (the “who”)
- **Hierarchical decomposition** into stubs and plug-ins
- Executable **scenario synthesis**, variant- or data-driven
- **Model families** with dynamic-stub variation points
- Self-contained **interactive HTML reports** with navigable SVG
- A fully scriptable **API**, and export of a session as runnable Python script + **JupyterLab Notebook**

Conclusions

- ✓ **Mined UCM models now arrive executable**
  one scenario per concurrency-aware variant, every branch guarded, every loop bounded
- ✓ **Two encodings, deliberately different**
  variant-driven is faithful and needs no attributes; data-driven proposes rules a domain expert can argue with
- ✓ **Beyond performers, decomposition, variants, and executable scenarios**
  family generation and comparison, dashboard, interactive HTML files, and Python script generation
- ✓ **An agent can build this, with scaffolding**
  executable oracles at every transformation boundary, and a domain expert as the verifier of record
- ✓ **What's next?**
  UCM + goals + indicators (goal mining), and (of course) AI and non-AI-based recommendations

16th International Model-Driven Requirements Engineering Workshop (MoDRE 2026)

https://arxiv.org/abs/2606.04350

Towards Process Mining Use Case Map Models with PM4Py-UCM

Daniel Amyot
School of Electrical Engineering and Computer Science
University of Ottawa
Ottawa, Ontario, Canada
damyot@uottawa.ca

On Process Mining Executable Use Case Maps: Concurrency-Aware Scenario Synthesis with Variant- and Data-Driven Conditions

Daniel Amyot
damyot@uottawa.ca
University of Ottawa
Ottawa, ON, Canada

18th System Analysis and Modelling Conference (SAM2026)

To appear

Submitted

https://arxiv.org/abs/2607.28825v1

Building a Process-Modeling Tool using Agentic AI: An Experience Report on PM4Py-UCM

Daniel Amyot
University of Ottawa, Ottawa, Canada
damyot@uottawa.ca

Try PM4Py-UCM!

GitHub repo, with code, documentation, and tutorials (JupyterLab notebooks)

https://github.com/ProcessMining-uOttawa/pm4py-ucm/

pip install pm4py-ucm

SCAN ME

Streamlit Web deployment, with sample logs

https://pm4py-ucm.streamlit.app/