Learning Formal Methods through Project-Based Modeling of Concurrent Systems with Anemone

Manel Barkallah Jean-Marie Jacquet

Faculty of Computer Science, Namur Digital Institute
University of Namur, Belgium

{*manel.barkallah, jean-marie.jacquet*}*@unamur.be*

27th International Symposium on Formal Methods
Formal Methods Teaching Workshop (FMTea 2026)
Tokyo, Japan — May 19, 2026

UNIVERSITÉ DE NAMUR

Outline

Motivation

- Formal Methods (FM) are increasingly important in industry.
- Teaching FM remains challenging:
  - abstraction barriers,
  - mathematical rigor,
  - low student engagement.
- Project-based learning can improve:
  - motivation,
  - practical understanding,
  - collaboration.

Goal

Explore how executable concurrent-system modeling with **Anemone** supports learning Formal Methods through: **coordination modeling**, **interference reasoning**, and **interleavings analysis**.

Educational Context

Course

- Design of Reactive Applications
- Master level
- University of Namur

Students

- Software Engineering
- Data Science
- heterogeneous profiles

Structure

- 30h lectures
- 15h project sessions

Topics

- Reactive systems
- Coordination via shared spaces
- Concurrent system modeling
- Executable formal models

Outline

Multi-Bach Coordination Language

Core Concept

Multi-Bach separates computation from coordination via a shared executable tuple space.

Main Operations:

- `tell(t)`: add data `t` to the store
- `ask(t)`: check for the presence of `t`
- `nask(t)`: check for the absence of `t`
- `get(t)`: remove `t` from the store

Anemone Workbench

Pedagogical Idea

Students see concurrency as execution, not description.

Main Objectives:

1. Facilitate learning of coordination primitives via stepwise execution
2. Support modeling of dynamic systems via interactive visualization
3. Observe executions through animation and trace inspection

Example:

$$Reach(\#finish = 1)$$

This means: the system eventually reaches a state where `finish` is present

Outline

Five Modeling Principles

1. Autonomous agents
2. Shared coordination space
3. Data-driven interaction
4. Non-deterministic execution
5. Observable interleavings

Key Shift

From sequential reasoning $\rightarrow$ coordination reasoning.

Concurrency Patterns

- Reactive agents
- Sensor–Actuator pattern
- Environmental agents
- Observer agents
- Coordination without central controller

Observation

Global behavior emerges from local interactions.

Outline

Case Study: Concurrent Systems

Example Topics

Braceleds

- interactive concert
- bracelets, lights, sensors

Smart Mirror

- reactive smart home
- user + autonomous agents

Hydroponics

- resource management
- temperature + water + nutrients

Student Tasks

- Model concurrent behavior
- Explore interleavings
- Analyze coordination issues

Key focus

- coordination modelling
- interference reasoning
- interleavings analysis

Concurrency Patterns in Practice: Braceleds

Reactive Agent Pattern

Example: Diffuser reacts to `music(love)` and produces `scent(chocolate)`.

```
Diffuser = ask(music (love));
           att(state, diffuser, concert, on);
           tell(scent (chocolate));
           NoDiffusion.


NoDiffusion = nask(music (love));
              get(scent (chocolate));
              att(state, diffuser, concert, off);
              Diffuser.
```

Concurrency Patterns in Practice: Braceleds

Sensor–Actuator Pattern

Sensors publish vibration levels, actuators react to them.

```
VibrationSensorOn = ( att(vib, vib_sensor, concert, low); tell(vib(low)) )
                  +
                  ( att(vib, vib_sensor, concert, high); tell(vib(high)) ).


VibrationSensorOff = ( get(vib(low)) + get(vib(high)) );
                      att(vib, vib_sensor, concert, off).
```

Concurrency Patterns in Practice: Braceleds

Environmental Agent Pattern

Example: participant movement, music evolution, and vibration updates evolve continuously and independently.
Insight: concurrency also comes from background system evolution, not only from user actions.

Observer Pattern

Example: traces and animations observe the shared state without modifying it, showing music changes, vibration levels, and lighting reactions.
Insight: students realise that observations depend on timing and may capture only partial system states.

Concurrency Patterns in Practice: Braceleds

Orchestration without Sequential Control

Global behavior emerges from concurrent agents.

```
PlayMusic = SpeakerOn;
            VibrationSensorOn;
            LightFloodlights(floodlight1, floodlight2);
            LightBracelets;
            (NextMusic + NextMusic + NextMusic + EndMusic).

NextMusic = (get(music(sad)) + get(music(happy)) + get(music(love)));
            (get(vib(low)) + get(vib(high)));
            PlayMusic.

EndMusic = SpeakerOff;
           VibrationSensorOff;
           LightsOff.
```

Outline

Evaluation

- 3 academic years
- 68 Master students
- qualitative evaluation

Initial difficulties

- sequential reasoning
- sequential execution bias
- misunderstanding non-determinism

Progression

- explicit concurrency reasoning
- better coordination policies
- improved abstraction choices

Observed Outcomes

Positive Effects

- Better understanding of concurrency phenomena
- Stronger collaboration

Challenges

- Initial complexity
- Tool learning curve
- Time management
- Balancing theory/practice

Takeaway

Hands-on projects make Formal Methods more accessible and meaningful.

Outline

Lessons Learned

- Incremental guidance is essential.
- Realistic examples improve motivation.
- Visualization and tooling reduce abstraction barriers.
- Peer collaboration supports learning.

Conclusion

- Formal Methods become more accessible through projects.
- Students better understand:
  - concurrency,
  - coordination,
  - non-determinism.
- Anemone makes concurrency observable through executable models and trace inspection.

Future Work

- Time primitives, other coordination frameworks.
- Larger-scale evaluations
- Integration with automated feedback
- AI-assisted tutoring and verification support

Concurrency is not made theoretic.
It is **executed**, **observed**, and **analyzed**.

Thank You!

Questions?

manel.barkallah@unamur.be