المدرسة الدولية الخاصة للتقنيات
POLYTECH INTL

Laboratoire des Sciences du Numérique de Nantes

SIMULATION OF CONTROL IN A DIOID USING A PYTHON LIBRARY

Prepared by:
Abir BEN BOUZAIENE

Supervised by:
Pascal ANDRÉ - VELO
Oliver BOUTIN - CPS3

Academic Year: 2024-2025

Agenda

1 Project Overview

2 System & Modelling Rules

3 Tools

4 Modelling Across Tools

5 Conclusion & Next Steps

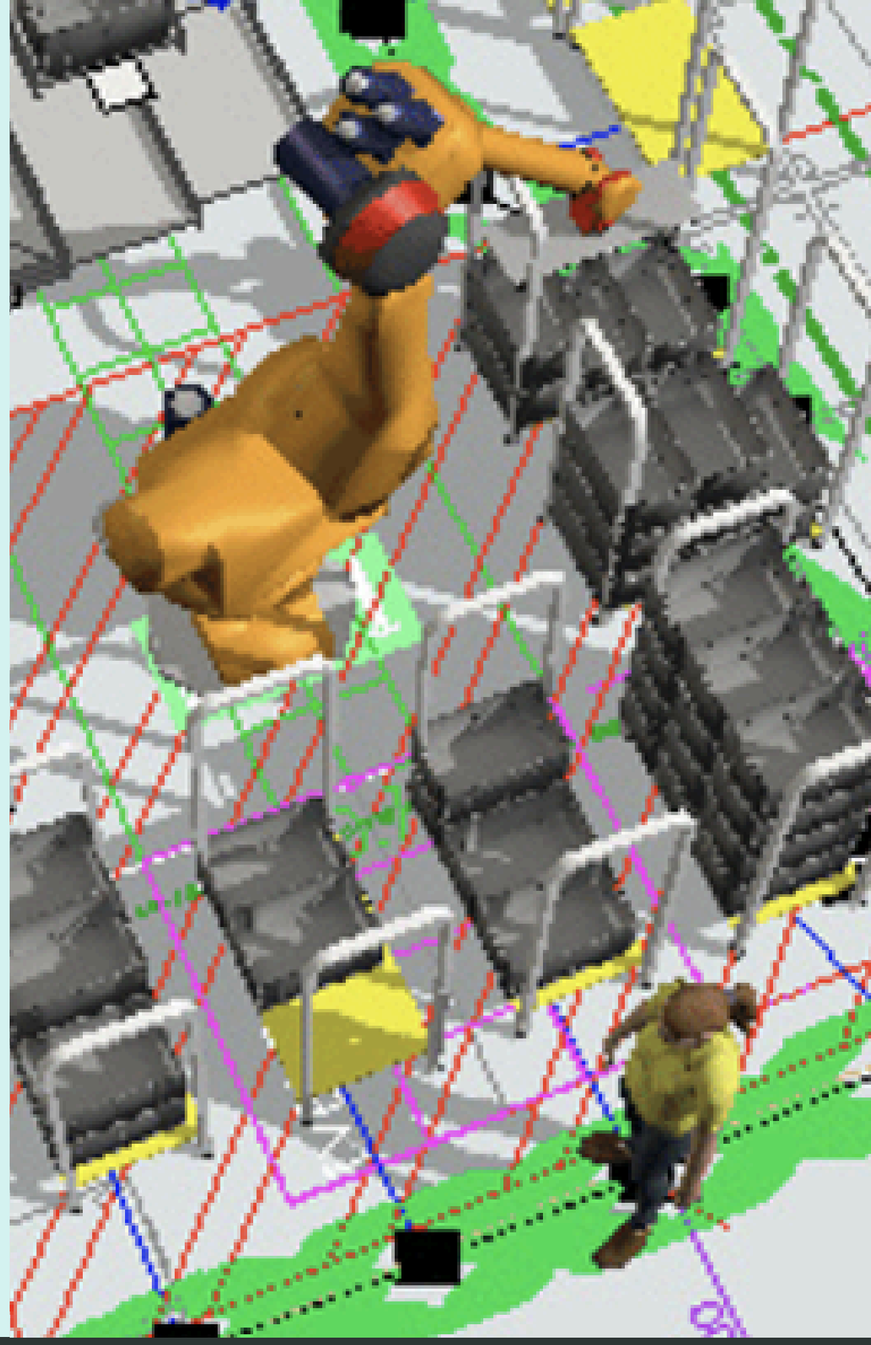
Project Setting

Modelling and simulation tools enable us to:

- Predict system behavior before physical implementation
- Optimize performance (throughput, efficiency, resource allocation)
- Test control strategies risk-free in digital twins

However, real-world systems present unique challenges:

- Behavior depends on modeling choices
- No tool captures all complexities



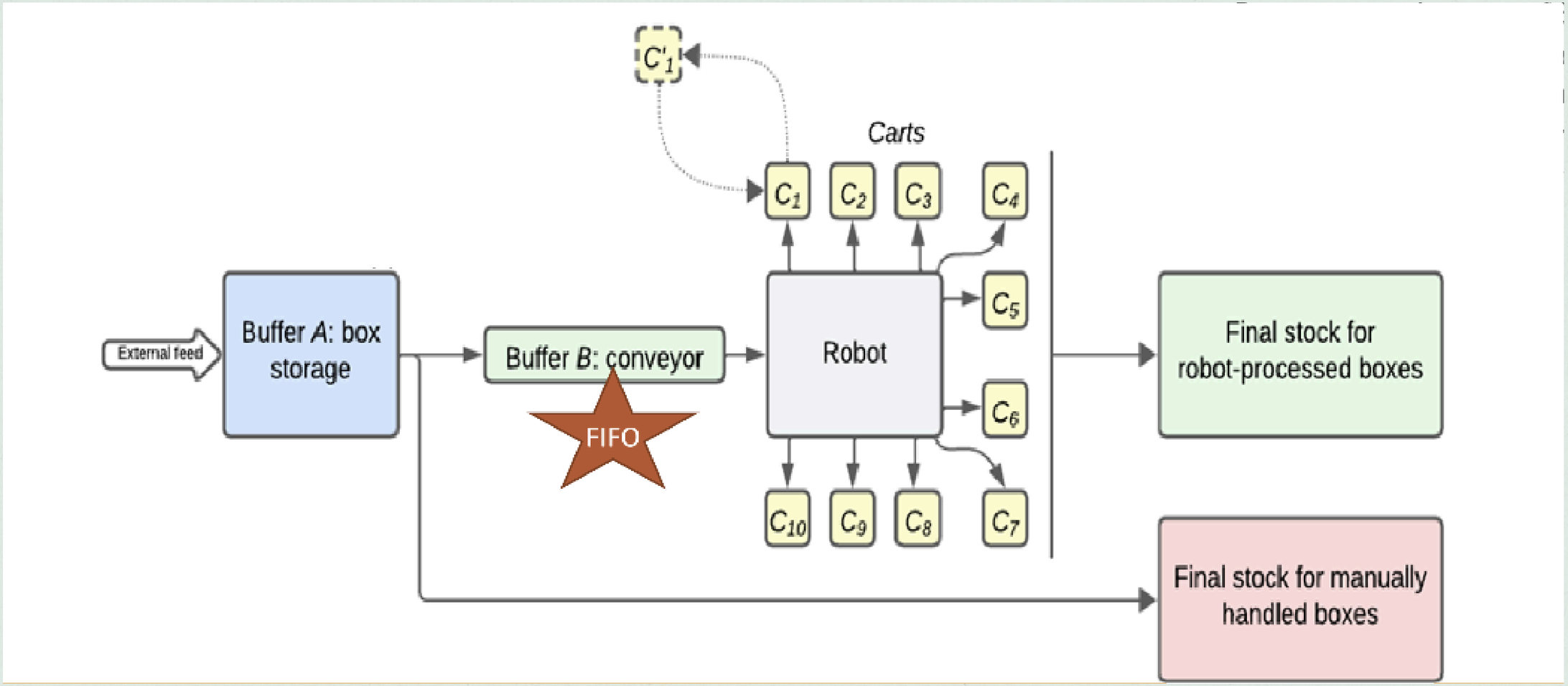
Project Objectives

Model the Flexibac system under different assumptions.

Test different modelling/simulation tools.

Explore Dioid algebra via PyMinMaxGD (Python library) for discrete event control.

The Flexibac System



Entry to Buffer A

Rule Type	Rule Specification	Rule Explanation	Rule Representation	Chosen for Modelling?
Fixed Time Interval	Boxes enter the buffer at regular intervals	Simulates a predictable, controlled input flow.	$t_i = t_0 + i \times \Delta t$ where Δt is the fixed interval.	Yes, it's useful for simplifying simulation and analysis.
Random Arrival Time	Boxes arrive at random intervals	Reflects real-world variability (Poisson or exponential distributions).	$P(T \leq t) = 1 - e^{-\lambda t}$ Exponential interarrival times.	Yes, but it requires probabilistic modeling
Scheduled Entries	Box entry follows a predefined schedule or time window.	Reflects timed batch arrivals (postal delivery truck drop-offs).	$t_i \in \{t_{sched}\}$ where $\{t_{sched}\}$ are scheduled times.	No, it may require a calendar-based input model which is not available for all tools.
Capacity Constraint	A new box can only enter if the buffer has not reached its capacity limit.	A physical constraint necessary to avoid overflows.	$n(t) < N_{\max}$ where $n(t)$ is current number of boxes.	No, this is a rule of the system and is applied automatically in all models.

Entry to Buffer B

Rule Type	Rule Specification	Rule Explanation	Rule Representation	Chosen for Modelling?
FIFO	Boxes are moved in the order they entered Buffer A.	Ensures fairness and simplicity. This rule is commonly used in queuing systems.	next_box = bufferA[0]	Yes, easy to implement in most modeling tools.
LIFO	Last box to enter is the first to be moved.	Less commonly used in industries because it can lead to older items being delayed or forgotten,	next_box = bufferA[-1]	No, this rule is less realistic for this context.
Random Selection	Boxes are moved randomly.	Used to test robustness or simulate noisy environments,	next_box = random.choice(bufferA)	Yes, This rule is included to model variability and randomness in box movement
Heuristic Rule	Move box b if its destination is already present in the current cart and there is space in Buffer B.	A specific rule that mimics operational preferences and real-time availability in Buffer B.	if b.destination in cart.destinations and bufferB.has_space(): move(b)	Yes, This rule reflects real-world decision-making by considering both destination and available space in Buffer B.

Carts Changing

Rule Type	Rule Specification	Rule Explanation	Rule Representation	Chosen for Modelling?
Full Cart	Cart is changed only when fully loaded with boxes of a single destination.	Requires each cart to be fully loaded before moving, which can create bottlenecks due to a high number of destinations (250) and limited carts (10).	if cart.load == Qmax: change_cart(cart)	Yes, it ensures efficient cart usage and is easy to model.
Time-based Change	Carts are changed at fixed time intervals, regardless of their load.	Enables regular rotation of carts, preventing delays caused by waiting for carts to fill completely.	if current_time - cart.last_change_time >= T: change_cart(cart)	Yes, it ensures regular cart movement even if a cart never reaches full capacity,
Conditional Change	A new cart is assigned when a box's destination does not have an available cart in the buffer.	Ensures all destinations have carts available, preventing blockages caused by missing carts for certain destinations.	if box.destination not in cart.dest: assign_new_cart(box.destination)	Yes, this rule prevents process blockages and avoids bottlenecks caused by destination mismatch.

Tools

PyMinMaxGD

- Dioid algebra
- Ideal for systems with delays & sync

Roméo

- Timed Event Graphs
- Formal verification & performance checks

FlexSim

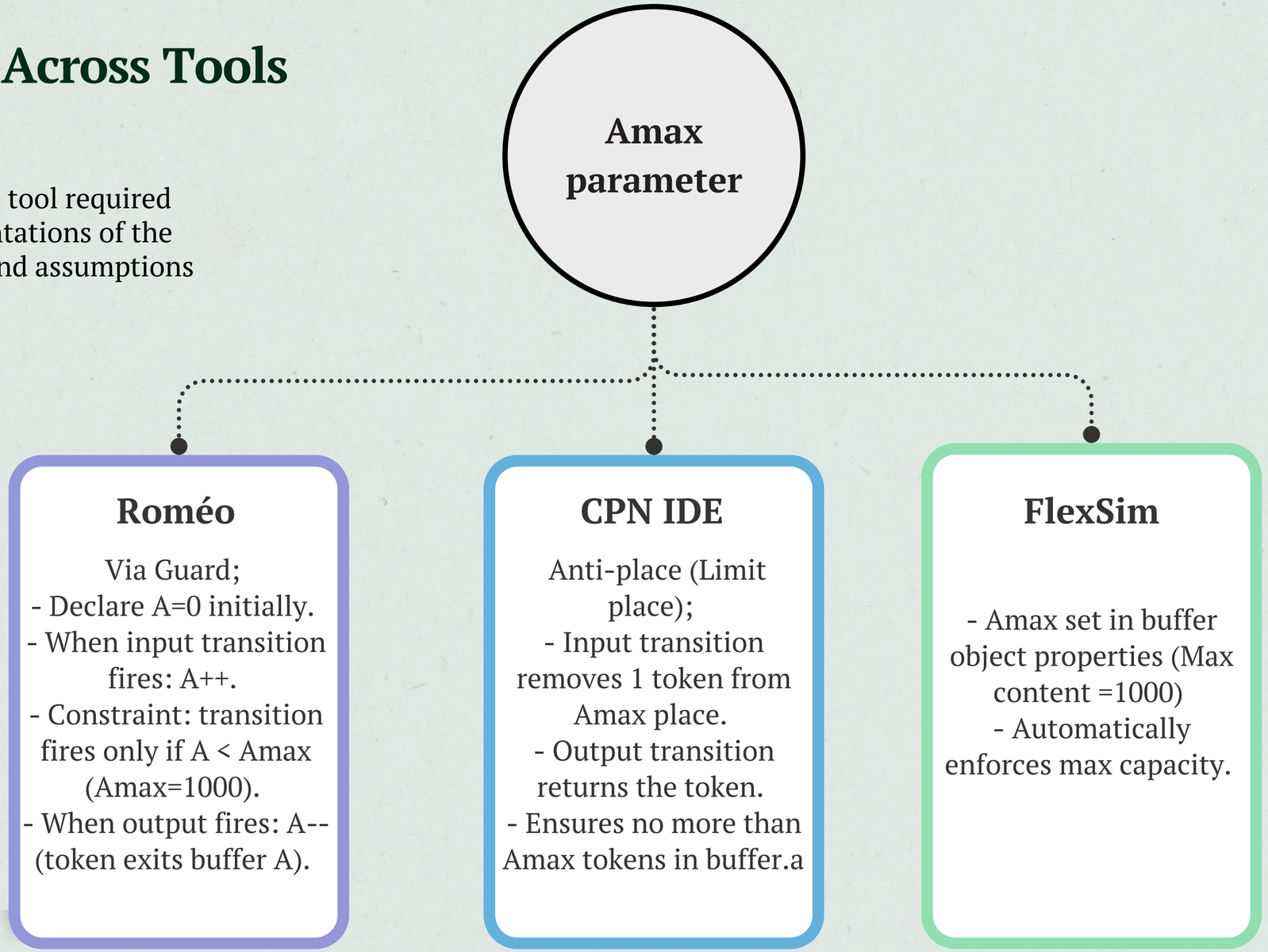
- Discrete-event simulation
- Widely used in industrial systems

CPN Tools

- Colored Petri nets
- Good for modeling complex logic

Modelling Across Tools

Record how each tool required unique implementations of the same parameters and assumptions



Conclusion: *Choosing the Right Tool and Assumptions*

- **No single tool captures all system complexities, each has limits.**
- **Select, combine and integrate tools based on problem context.**

“What is the problem I'm trying to solve? which tool best fits its nature?
“What are the key system behaviors to capture?”

Next Steps

- **Focus on modeling using Dioid algebra.**

→ Use PyMinMaxGd for Dioid computations
→ Use FlexSim for simulation and visualization.

Thank you!