COGNIFYER

From Real-Time Guarantees to AI-Driven Architecture Architecture Design

RTaW · Cognifyer · NEURESA — and the application allocation challenge

Patrick Keller
CTO, Cognifyer

INTRODUCTION

My Journey: From **computer graphics** to **DevOps** to **AI-driven design automation**

2010 – 2014 — **BSc Computer Science — Saarbrücken**
Bachelor thesis: Shadow Mapping for XML3D

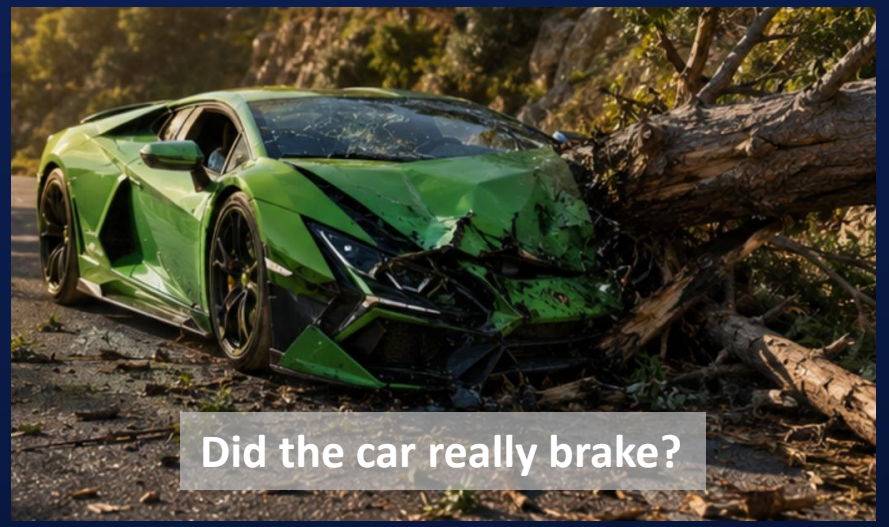
2015 – 2017 — **Software Engineer — Talkwalker, Luxembourg**
Infrastructure team

2017 – 2019 — **MSc Computer Science — University of Luxembourg**
Semantic code-clone detection via code visualization & transfer learning (WySiWiM)

2019 – 2024 — **PhD — University of Luxembourg**
Tightening worst-case timing bounds for real-time networks through simulation

Since Apr 2024 — **CTO — Cognifyer**
Generative, AI-driven design automation and optimization for cyber-physical real-time systems

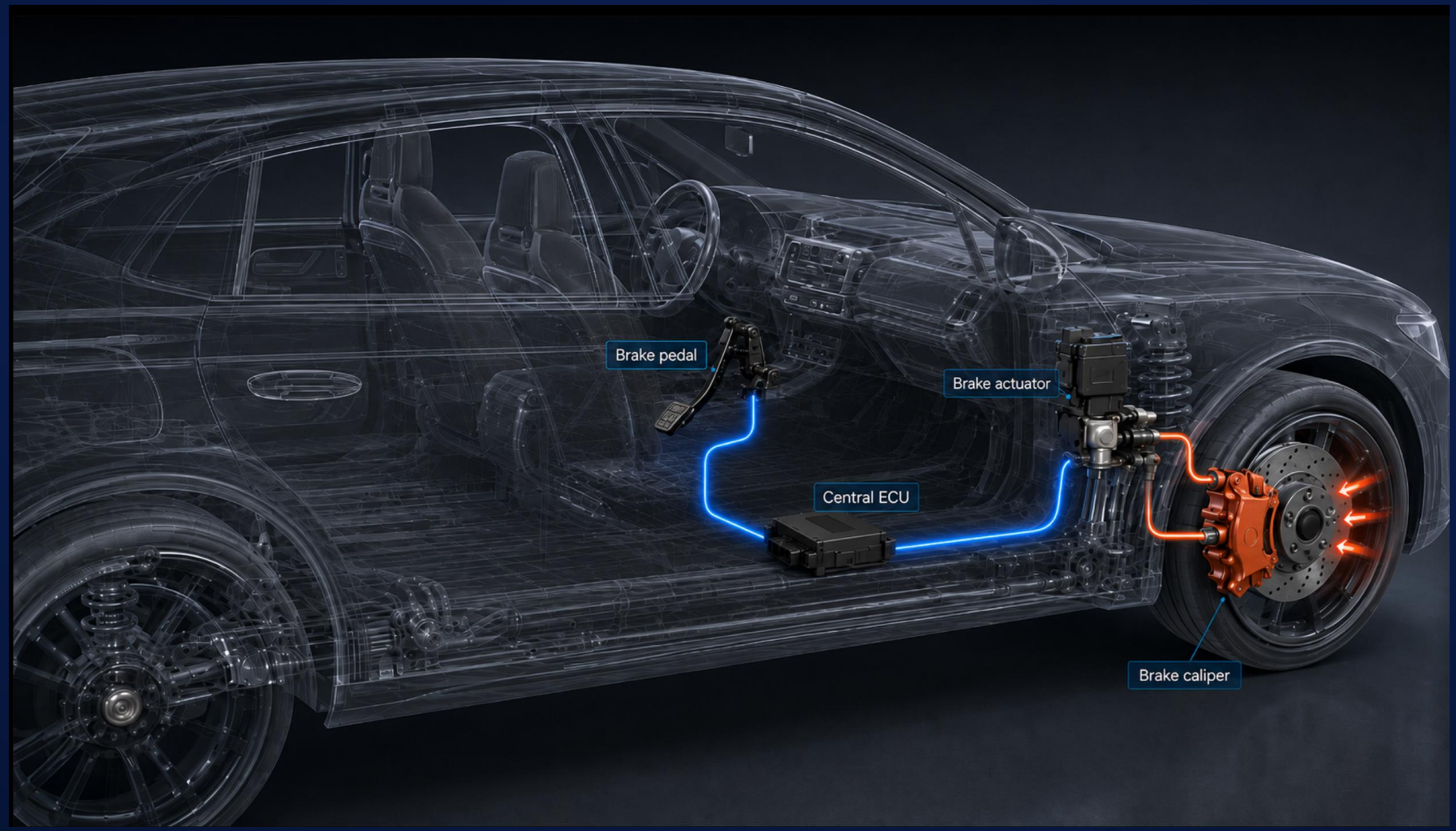
A PRACTICAL EXAMPLE



* Images are generated with ChatGPT, no Lamborghinis were harmed in the creation of this story ;)

A PRACTICAL EXAMPLE

A real Example: A modern brake-by-wire system



A PRACTICAL EXAMPLE

Real-Time Systems: When a late message becomes a catastrophe

- A **brake-by-wire** command must reach the actuator within a hard deadline — every time, not just on average.

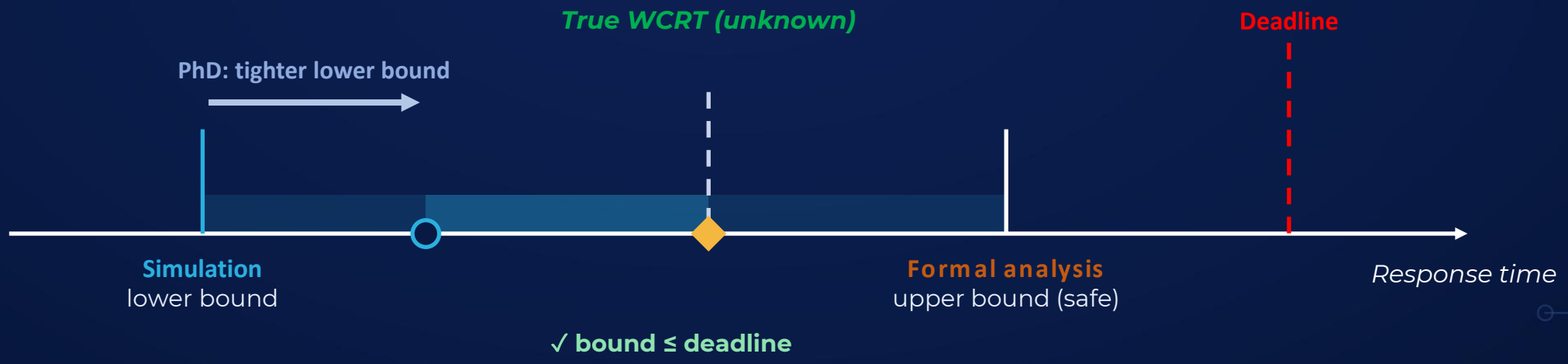
- **Functionally correct but late = wrong.** The deadline must hold under the worst case — not just on average in testing.

- **Surprising Fact:** Real-time does not always mean fast, “immediately” or milliseconds. Some hard real-time systems can have minutes or even hours of hard deadlines. (e.g. emergency nuclear powerplant shutdown)

THE CORE PROBLEM

Bounding the worst case: simulation vs. formal analysis

The true worst-case response time cannot be observed directly — so we bound it from two sides.



Simulation **under-estimates,** formal analysis **over-estimates**.
(Sidenote: My PhD results pushed the simulation bound closer to the truth.)

CORE MESSAGE

Industrial design is becoming too complex for *manual* engineering alone alone

- Embedded systems are becoming increasingly complex (>100 ECU*s in some modern cars) software-defined, networked, and continuously evolving (Over-The-Air updates).
- Correctness is no longer only functional — timing, safety, resources, and evolution all matter.
- **RTaW** mastered model-based modelling and validation of real-time systems.
- **Cognifyer** pushes the next step: AI-assisted** design automation and optimization.
- **NEURESA** makes this practical for complex E/E architecture design.

* ECU = Electronic Computation Unit (a small computer or chip) **AI in the broad sense, not only LLMs/NNs

RTaW · OUR PARENT COMPANY

RealTime-at-Work: design tools for critical embedded systems

RTaW-Pegase is a platform to **model**, **design**, **simulate**, **configure**, and **validate** embedded networks and software-defined systems.

DOMAINS

Automotive | Aerospace | Defense | Industrial Ethernet

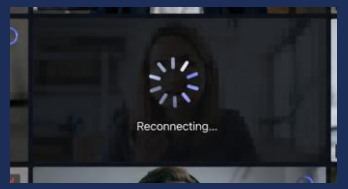
TECHNOLOGIES

TSN | CAN | LIN | ARINC | Wireless

The value proposition: safe, optimized systems with timing behavior validated before integration.

RTAW · HARD REAL-TIME

In hard real-time systems, being late can mean being wrong



Normal IT system

Optimizes **averages** for throughput, cost, and utilization.

Hard real-time system

Must satisfy deadlines under **worst-case** conditions.

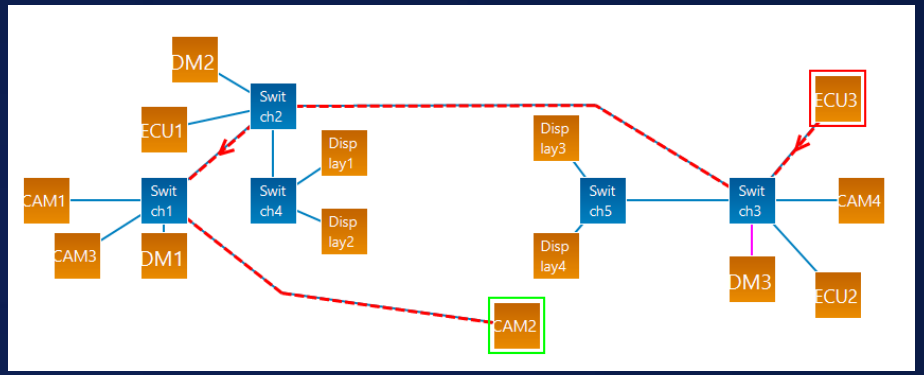
Correct only when the **right computation** happens at the **right time.**

WHERE IT MATTERS

- Braking
- Steering
- Flight control (Avionics, Space)
- Industrial motion control
- Safety-critical communication

Timing problems emerge from interactions

- Many flows share the same switches, links, queues, CPUs, buffers, and scheduling policies.



Small design decisions can create large timing effects:

Priority assignment | Routing | Task placement | TSN schedules | Clock drift | Burstiness | Mixed-criticality traffic

The core question: **can every critical message and task meet its deadline under all relevant conditions?**

RTAW · VALIDATION

Simulation, mathematical analysis, and trace validation

Simulation

Explores realistic behavior and design variants.

Worst-case analysis (Formal methods)

Safe upper bounds for latency, buffers, and schedulability.

Trace validation

Checks that the real system matches the model.

Closing the loop from design to integration — **RTaW-Pegase** unifies simulation, upper-bound computation, configuration, and trace analysis.

From manual trial-and-error to automated configuration

1. Capture requirements and constraints.
2. Model the architecture.
3. Automatically configure network QoS and scheduling parameters.
4. Validate timing, bandwidth, buffers, and resource usage.
5. Iterate & Optimize before hardware integration.

RTaW moves timing evidence earlier in the design process — reducing integration risk by design.

Cognifyer: generative design for cyber-physical systems

- A University of Luxembourg startup building technologies, software IP blocks, and automated design flows (for RTaW at this time).
- Cognifyer develops algorithmic tools for E/E architecture design.

METHODS / SERVICES

Generative design | Optimization | Design-space exploration

Verification | Synthetic data | Deep learning

Cognification

Cognification = turning industrial expertise into executable intelligence

Captured expertise

- Domain knowledge
- Engineering rules
- Constraints
- Design patterns
- Expert heuristics
- Validation logic
- Data from past projects

Executable intelligence

- Algorithms
- AI assistants
- Automated workflows

Not AI bolted onto a tool — the **systematic encoding of industrial reasoning** into the design process.

PROJECTS

From internal AI adoption to industrial design automation

1. **AI in operations at RTaW**
   AI in internal workflows: documentation, support, knowledge access, software engineering (impl. and code reviews).

2. **RTaW assistant**
   Domain-aware interaction with the extensive tool documentation, and in the future with the user architecture.

3. **PowerMode Studio**
   Mode-aware design exploration across power, operating states, and constraints.

4. **TSN Industry Studio**
   Automated configuration for industrial TSN automation networks.

5. **NEURESA**
   Neuro-symbolic reasoning for embedded system architecture design.

NEURESA · THE AMBITION

Neuro-symbolic reasoning for embedded system architecture design

NEURESA aims to develop intelligent, agent-based CAD software for complex automotive electronic systems.

WHAT IT BRINGS TOGETHER

- AI-driven modeling
- Natural-language interfaces
- Dynamic visual interfaces
- GPU-accelerated computation
- Efficient & accessible design
- Neuro-symbolic constraint solving

The aim: **more efficient, reliable, and accessible design** of complex car electronic systems.

Beyond our Industry: Integration of CP into LLMs could **enable complex optimizations** for experts **without experience in CP**.
Potentially benefiting industries reach from Logistics to Healthcare and everything in between.

The design bottleneck is shifting

Yesterday's bottleneck

- Can we simulate this?
- Can we verify this?
- Can we configure this network?

Today's bottleneck

Can we generate, compare, explain, and validate good architecture decisions fast enough?

NEURESA targets exactly that bottleneck & makes implicit requirements/constraints visible.

The main problem: deciding where software should run

Given

- Applications/Functions
- Communication needs between applications
- Computation/Memory needs per application
- Processors with fixes resource supply
- Redundancy requirements
- Location constraints
- …

➢ The problem parameters can become arbitrarily complex and depend on the concrete use-case

Decide (for instance)

- Which application runs on which hardware
- How tasks map to processors
- How communication is routed
- Whether timing and resource constraints hold
- Which trade-off is best

A combinatorial, multi-objective, constraint-heavy design problem.

APPLICATION ALLOCATION

Why it is hard: a small mapping change can break the system

Allocation affects

- CPU load
- Memory
- Network load
- End-to-end latency
- Safety separation
- Energy
- Cost
- Updateability
- Redundancy
- Scalability

Objectives conflict

Fewer ECUs ⇄ lower load
Cheaper hardware ⇄ timing margin
Consolidation ⇄ fault isolation
Flexibility ⇄ certification effort
Performance ⇄ energy

There is no single correct solution & **Manual reasoning does not scale.**

NEURESA · APPROACH

AI extracts constraints, CP solver proposes solutions, solutions, engineers decide & guide

1. Engineer describes goals and constraints in NL.
2. AI assistant helps complete and structure the model.
3. Neuro-symbolic solver generates candidate allocations.
4. DSE ranks trade-offs: cost, timing, resources, safety, scalability.
5. Simulation and formal checks validate candidates.
6. The system explains why a solution works or fails.
7. Engineer refines objectives and constraints interactively.

NEURESA does not replace engineers — it **amplifies expert engineering judgment and makes CP accessible without the friction of acquiring CP expertise first.**

CLOSING

From validation to generative engineering

- **RTaW** brought timing correctness into the design phase.
- **Cognifyer** extends it with cognification, AI, optimization, and design automation.
- My mission: make this industrially robust.
- **NEURESA** focuses the ambition on one of the hardest E/E problems — application allocation.

The goal: a new generation of CAD tools — **intelligent, explainable, constraint-aware, and validation-driven, to accelerate candidate exploration and innovative choices.**