opaal

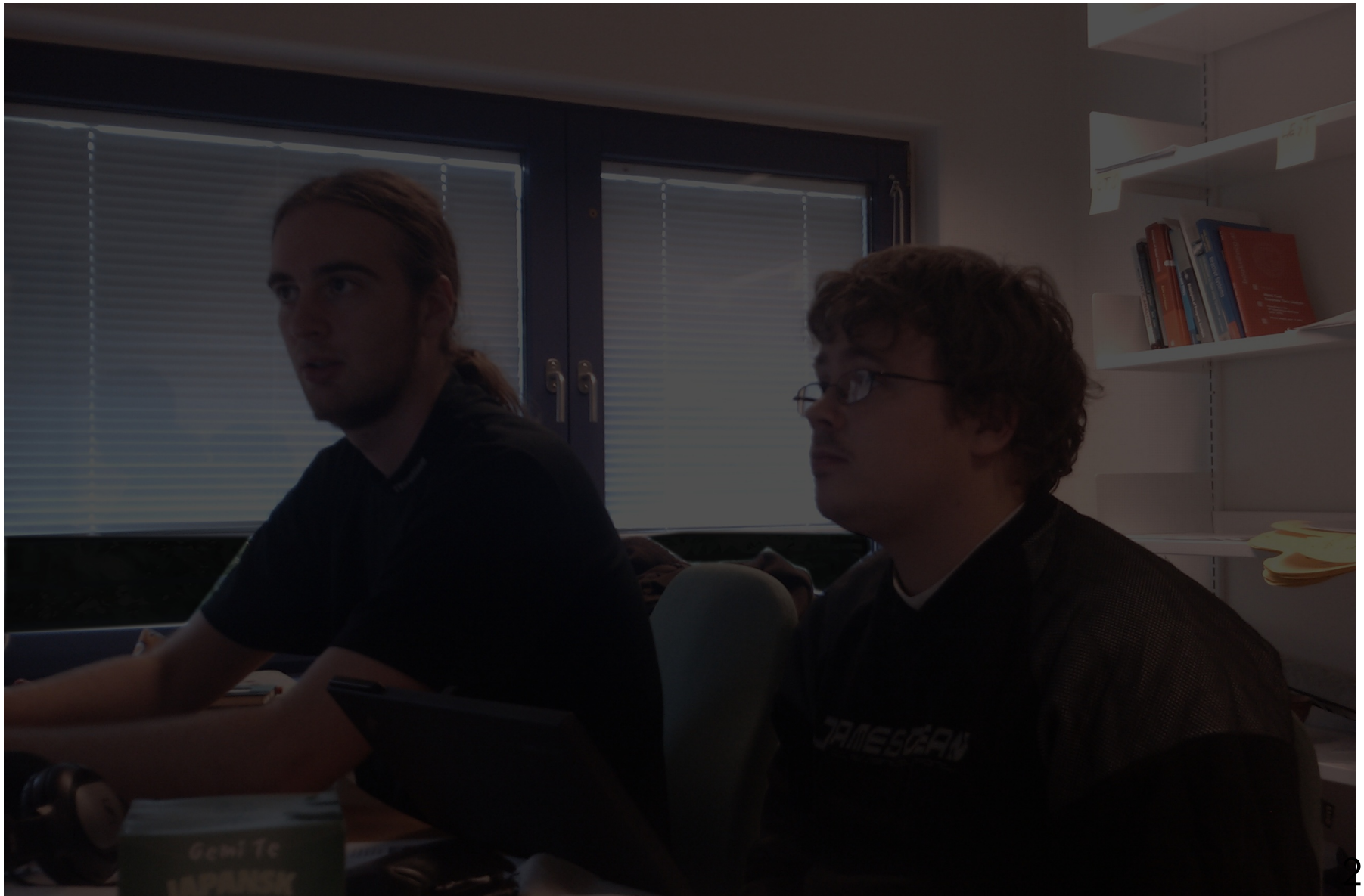
Kenneth Yrke Jørgensen <kyrke@cs.aau.dk>,
Mads Chr. Olesen <mchro@cs.aau.dk>
opaal-modelchecker.com

9th MT-LAB workshop

August 24 2010

How hard can it be to build a model checker?

It was a dark and stormy night. . .



A bit of history

Before 1970:

- Operating systems written in Assembler
- Hard to port, maintain, tedious and error prone to write
- Fast — C was too high-level (= slow)

A bit of history

Before 1970:

- Operating systems written in Assembler
- Hard to port, maintain, tedious and error prone to write
- Fast — C was too high-level (= slow)
- If you did it any other way. . .

A bit of history

Before 1970:

- Operating systems written in Assembler
- Hard to port, maintain, tedious and error prone to write
- Fast — C was too high-level (= slow)
- If you did it any other way...

  ...you were an *idiot*!

A bit of history

Before 1970:

- Operating systems written in Assembler
- Hard to port, maintain, tedious and error prone to write
- Fast — C was too high-level (= slow)
- If you did it any other way...

...you were an *idiot*!

UNIX

A bit of history

Before 1970:

- Operating systems written in Assembler
- Hard to port, maintain, tedious and error prone to write
- Fast — C was too high-level (= slow)
- If you did it any other way...

  ...you were an *idiot*!

UNIX

Written in C

A bit of history

Before 1970:

- Operating systems written in Assembler
- Hard to port, maintain, tedious and error prone to write
- Fast — C was too high-level (= slow)
- If you did it any other way...

  ...you were an *idiot*!

UNIX

Written in C

Today:

- All operating systems written in C

Introducing opaal

opaal is a

- distributed,
- discrete time
- verification tool
- for uppaal Timed Automata
- written in Python,
- to make rapid prototyping.

But why!

Why build opaal?

- Its fun!
- To learn
- Nobody wants to touch uppaal

Why this technology?

- We already knew how to parse uppaal xml files
- Big library of uppaal files
- Python is a good prototyping language

Goals

1. Rapid prototyping
2. Easy to learn
3. Implement the 20% of the optimisations that give 80% of the speedup

Goals

1. Rapid prototyping
   - Try out concepts quickly
   - Before doing more time-consuming, optimised implementation
   - Open Source
2. Easy to learn
3. Implement the 20% of the optimisations that give 80% of the speedup

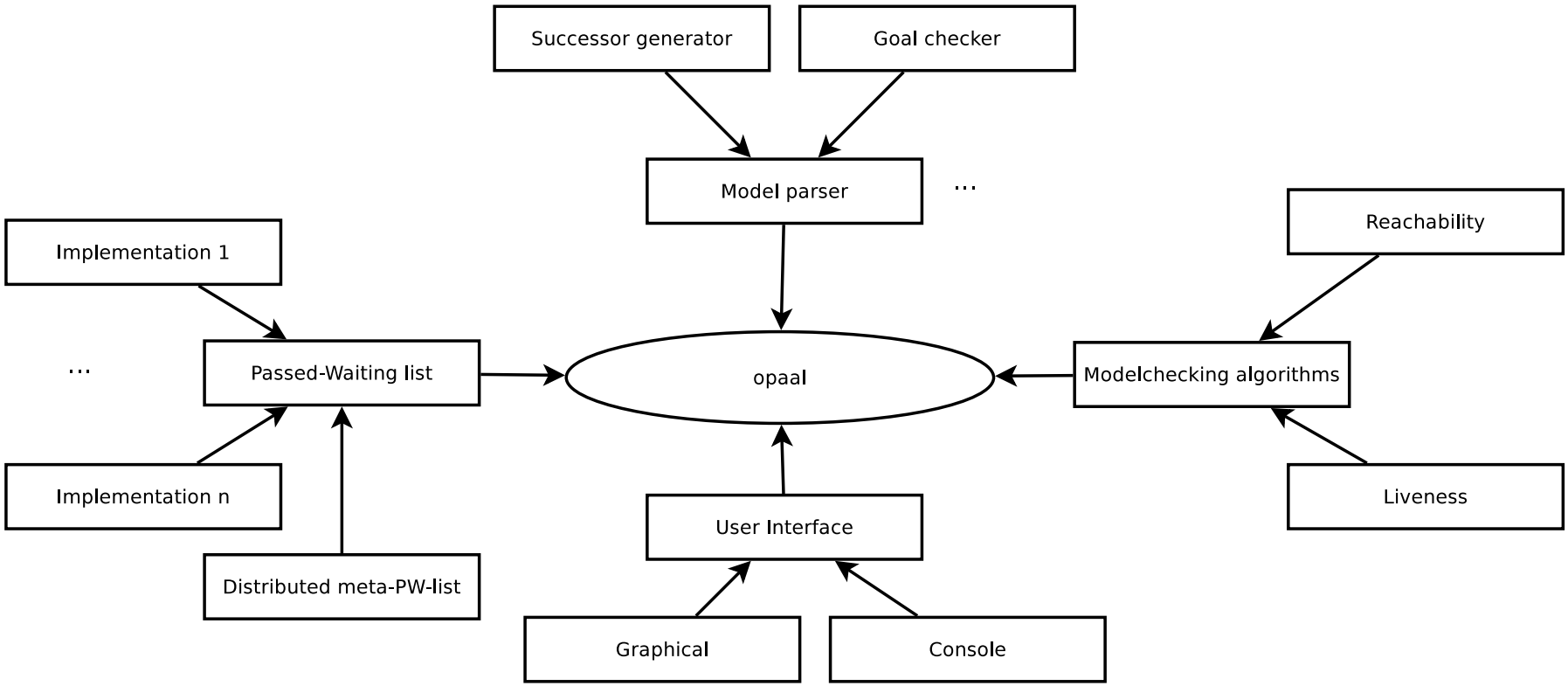
Goals

1. Rapid prototyping
   - Try out concepts quickly
   - Before doing more time-consuming, optimised implementation
   - Open Source
2. Easy to learn
   - A group of 5th semester students should be able to implement *something* in a project
   - Readability, overview, loose coupling
3. Implement the 20% of the optimisations that give 80% of the speedup

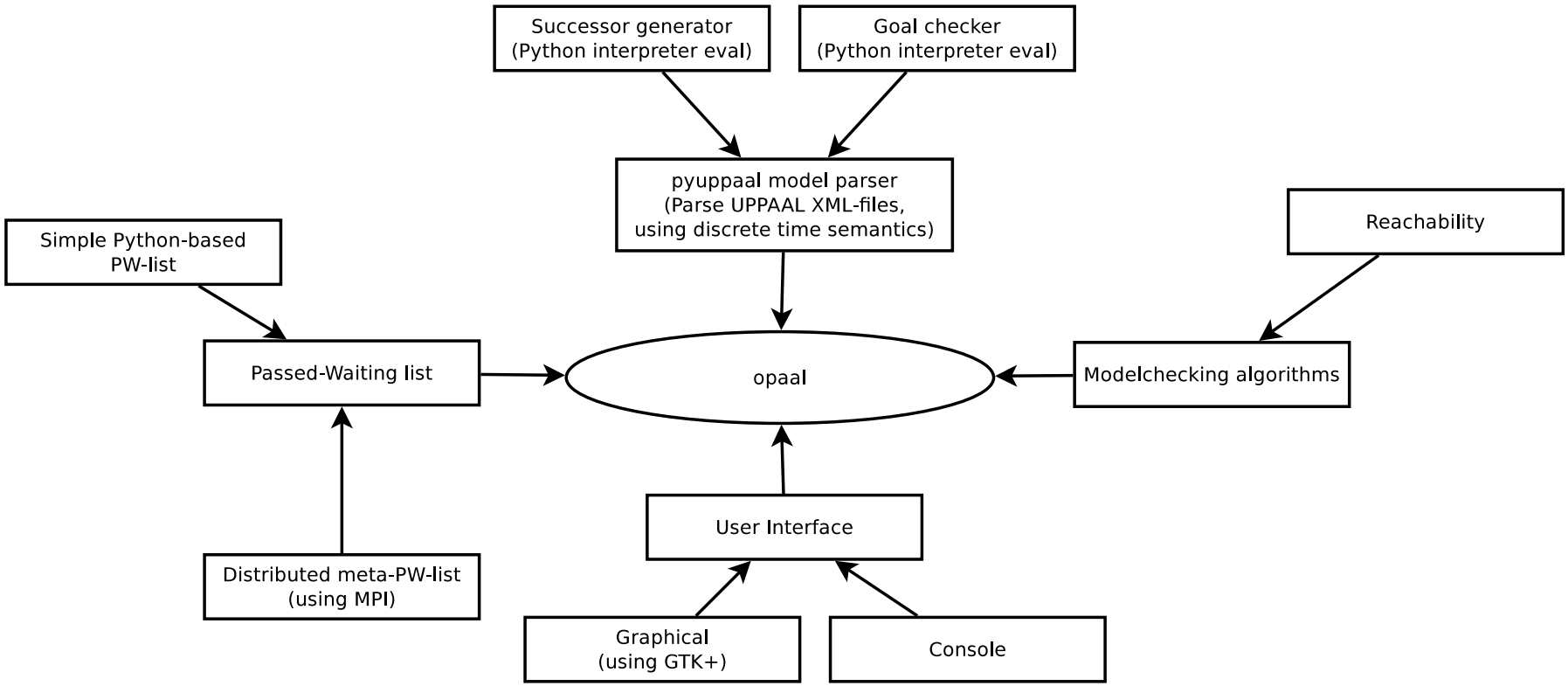
Goals

1. Rapid prototyping
   - Try out concepts quickly
   - Before doing more time-consuming, optimised implementation
   - Open Source
2. Easy to learn
   - A group of 5th semester students should be able to implement *something* in a project
   - Readability, overview, loose coupling
3. Implement the 20% of the optimisations that give 80% of the speedup
   - No gold plating
   - Sufficiently fast
   - We learn as we go

Desired Architecture



Current Architecture



Current Problems

We are two slow (even for prototyping)

- Python not suited for successor generator
- Python PW-list uses very general hash-table

Performance of Python successor generator:

- About 10000 states/sec
- Scaled up to 8 cores
- 8 cores $\approx$ 80000 states/sec $\approx$ UPPAAL single core
- Used up to 32Gb of RAM

Ongoing Work

- Generate pyuppaal LLVM successor generator
  - LLVM = high-level assembler
  - Faster than C? (*-O4*!!!)
- Datastructures optimised for discrete time semantics
- Passed-waiting list using slice memory allocator
  - Memory allocator suited especially for many small allocations of same size
- Lattice Automata

Future Work

- Real-time support
- Counter-Example Guided Abstraction Refinement
- Support for other model (TAPN/TAPAAL)
- Insert your idea here...

About

You can follow the project at

www.opaal-modelchecker.com

or at www.launchpad.net/opaal

Feel free to contact us at {kyrke,mchro}@cs.aau.dk, or #opaal @ irc.efnet.org

1. How hard can it be to build a model checker?
   - It was a dark and stormy night...
   - A bit of history
   - Introducing opaal
   - But why!
   - Goals
   - Desired Architecture
2. Current Status
   - Current Architecture
   - Current Problems
   - Ongoing Work
3. Future Work
   - Future Work
   - About