UNIVERSITY OF TWENTE.

LUNA: Hard Real-Time, Multi-Threaded, CSP-Capable Execution Framework

M. M. Bezemer

R. J. W. Wilterdink

J. F. Broenink

Control Engineering, University of Twente, The Netherlands

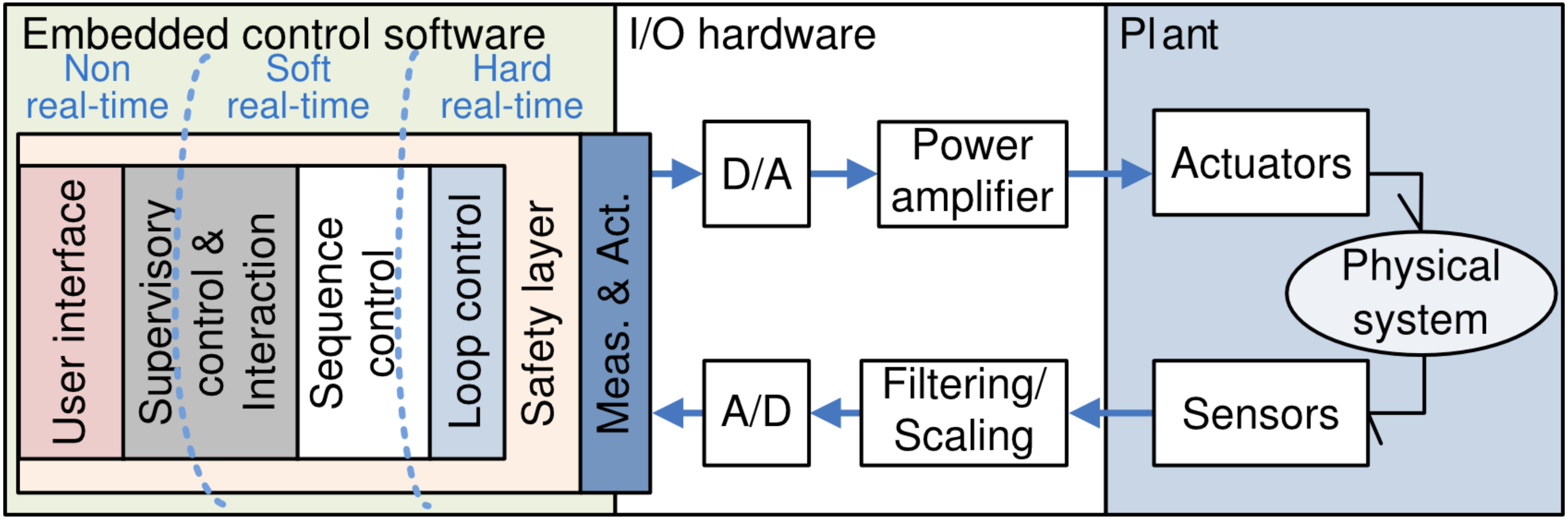
Outline

- Context and Introduction
- Framework architecture
  - Threading
  - Channels
  - CSP processes
  - Alternative
- Results
  - Measurements
  - Comparison
- Conclusions

Context

- Controlling embedded set ups / robots
  - Low resources
  - Custom build (Linux) Operating System
  - Guaranteed deadlines for updates for calculated motor signals
- Frameworks help with generic implementations / behaviour
- Multi core and/or distributed systems
  - Requires extra support from framework
  - CSP helps with organizing the execution flow
- Support multiple targets
  - Also requires extra support from framework

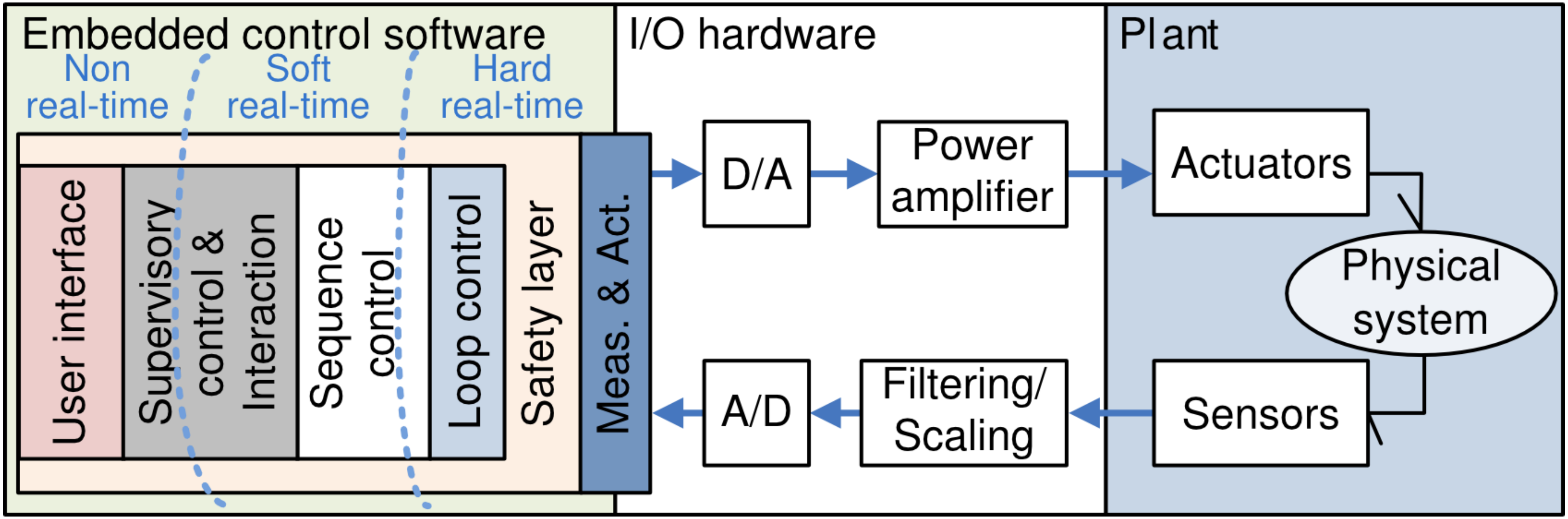
Embedded Control SW



- Controlling actual set ups requires different layers
  - Loop control - Control the physical system
  - Sequence control - Provide 'setpoints'
  - Supervisory control - Complex tasks: planning, mapping, …
  - User Interface - Connection with user

Embedded Control SW

UNIVERSITY OF TWENTE.



- Controlling actual set ups requires real-time levels
  - Hard real-time - must meet deadlines
  - Soft real-time - should meet deadlines
  - Non real-time - everything else

Introduction

- Requirements for an embedded control software framework
  - Hard real-time
  - Multi-platform
  - Thread support
  - Scalability

- Other 'handy features'
  - CSP execution engine
  - Low development time for framework user
  - Debugging and Tracing

Introduction

- Existing solutions do *not* meet all requirements
  - C++CSP2 not hard real-time
  - CTC++ not multi-threaded

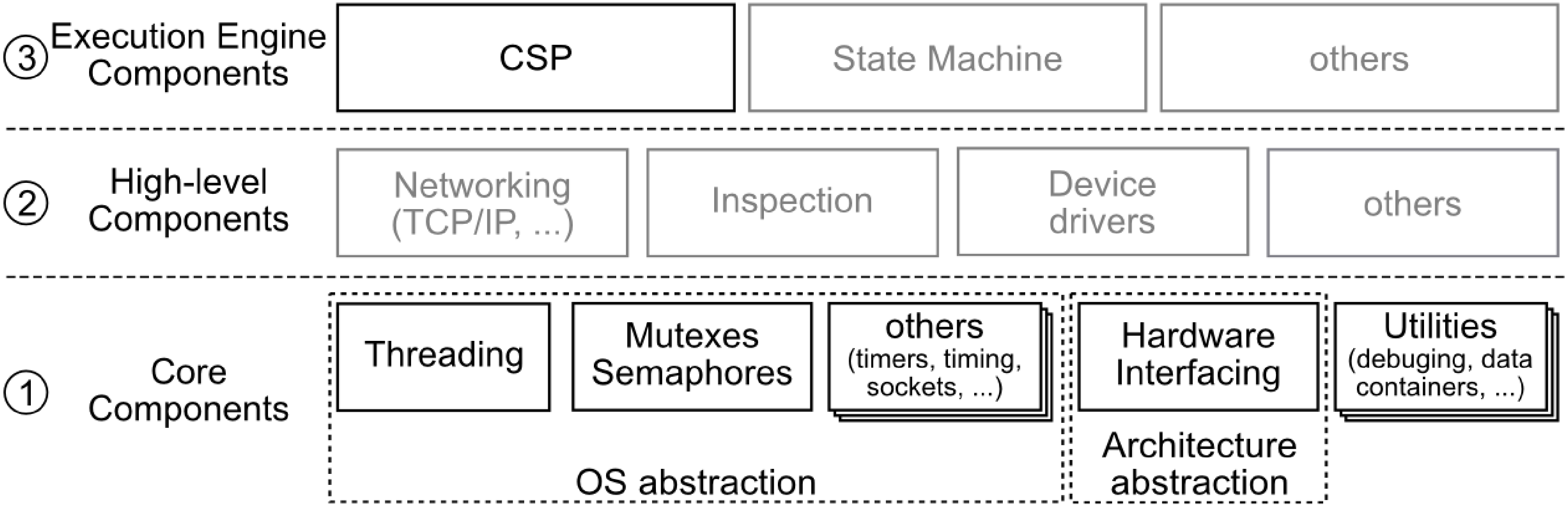
- Develop a new framework to meet all the requirements

LUNA

LUNA is a Universal Networking Architecture

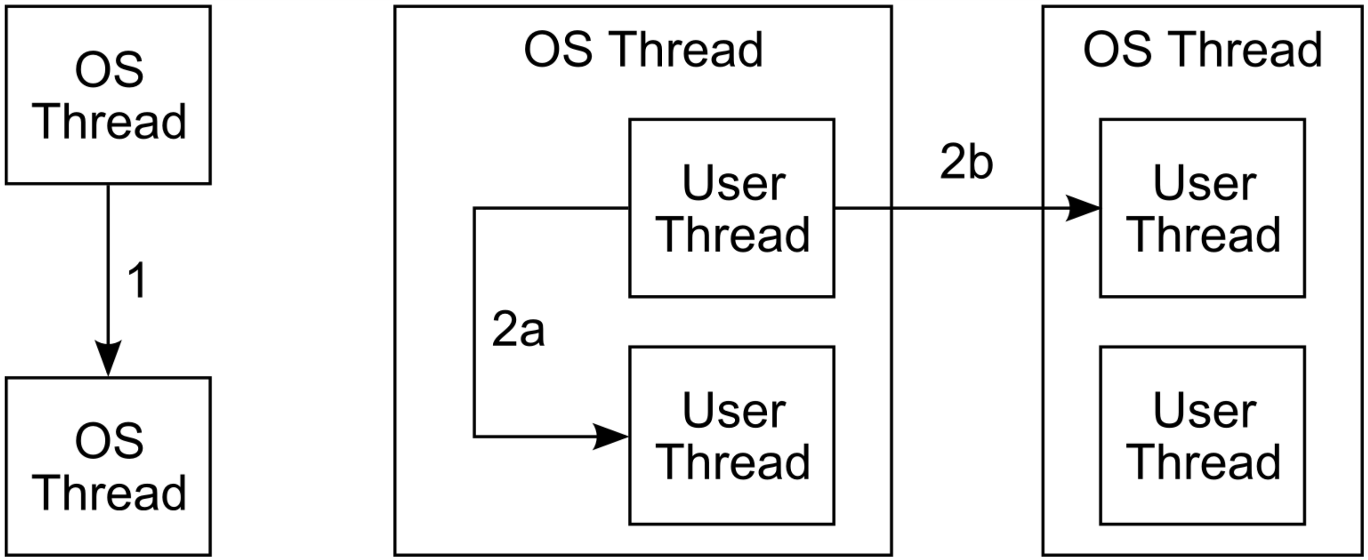
Architecture

- 1) Core Components
  - Platform support components + utility components
- 2) High-level Components
  - Platform independent components
- 3) Execution Engine Components
  - Components to determine the order of execution



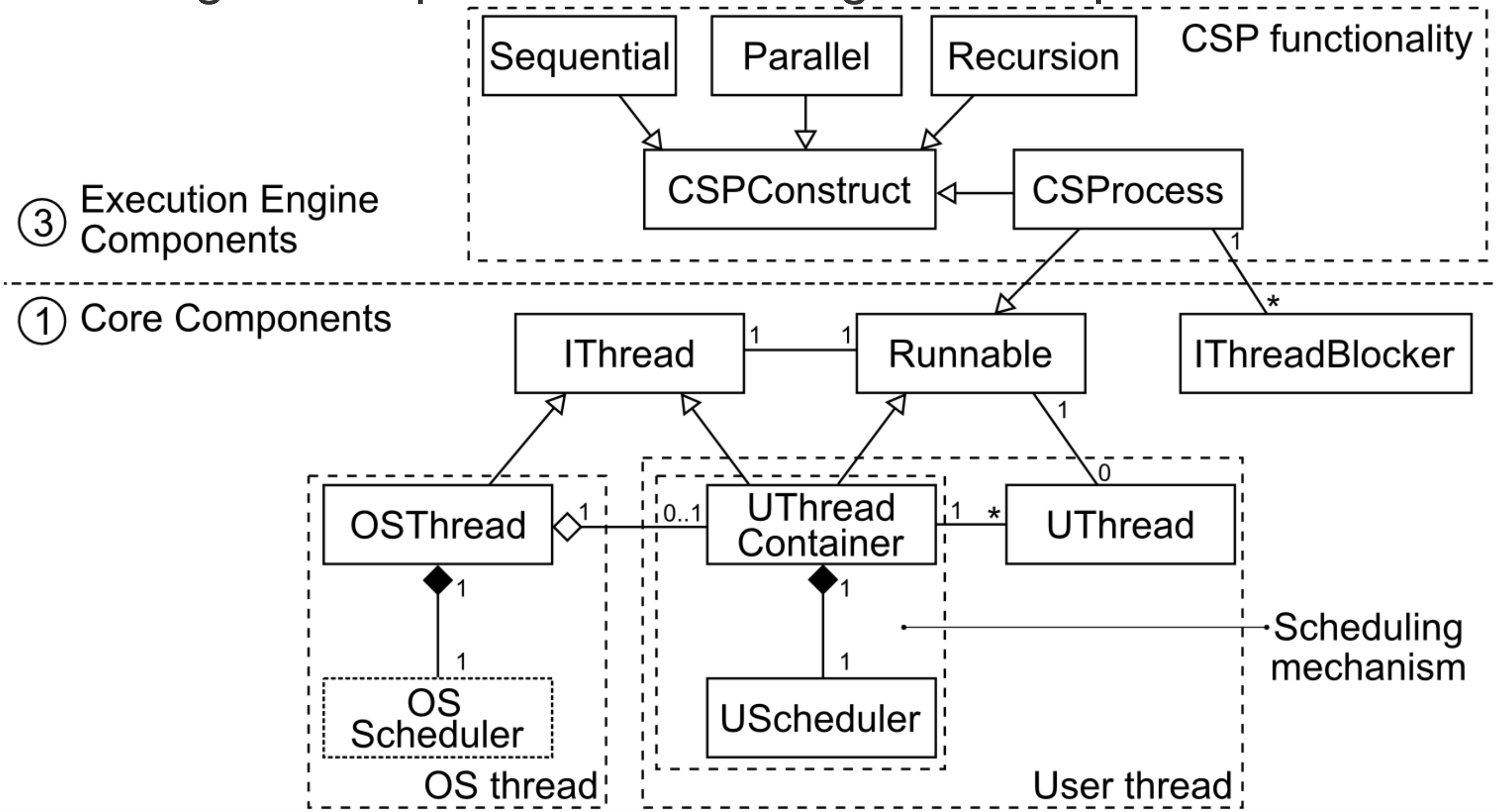
Threading

- Hybrid threading support
  - OS Threads – required for multi-core support
  - User Threads – fast(er) switching between threads



Threading

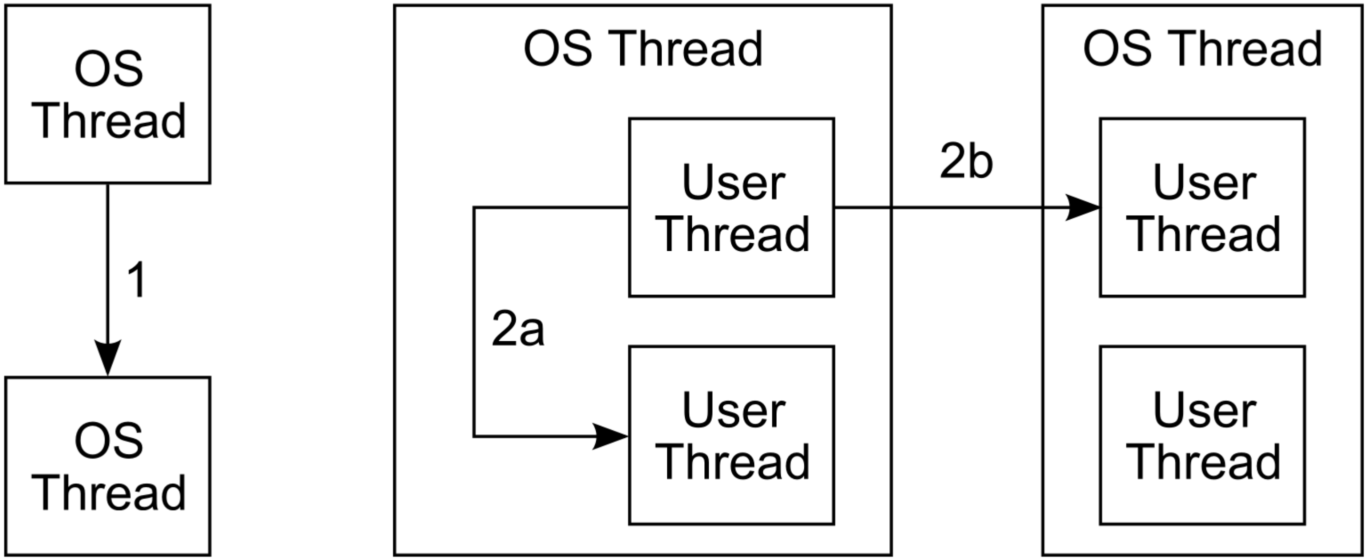
- CSP implementation with separation of concerns
  - Core components for platform-dependent threading components
  - Execution engine component for CSP algorithm implementation



Channels

- Two types of channels
  1) Rendez-vous communication between 2 OS threads
     Blocks the complete OS thread, used for multi-core communication

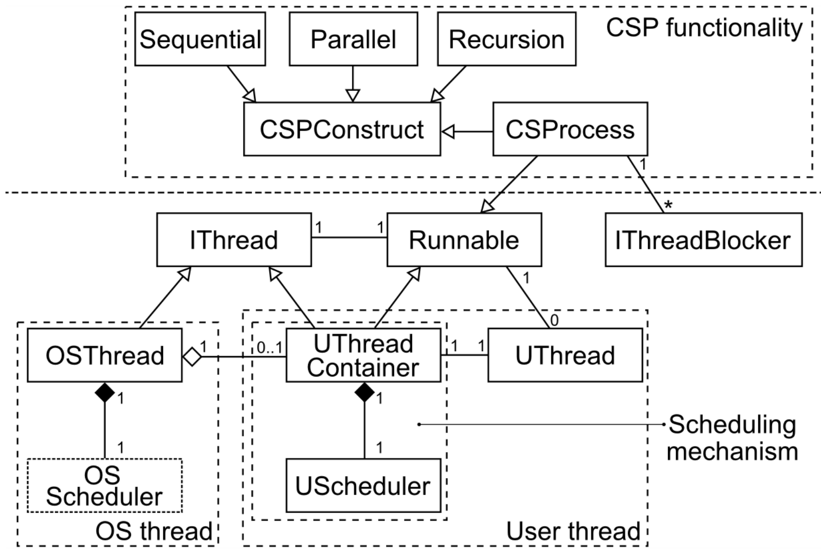
  2) Rendez-vous communication between User Threads
     Faster and without blocking complete OS thread
     Complete CSP functionality: buffered, guarded



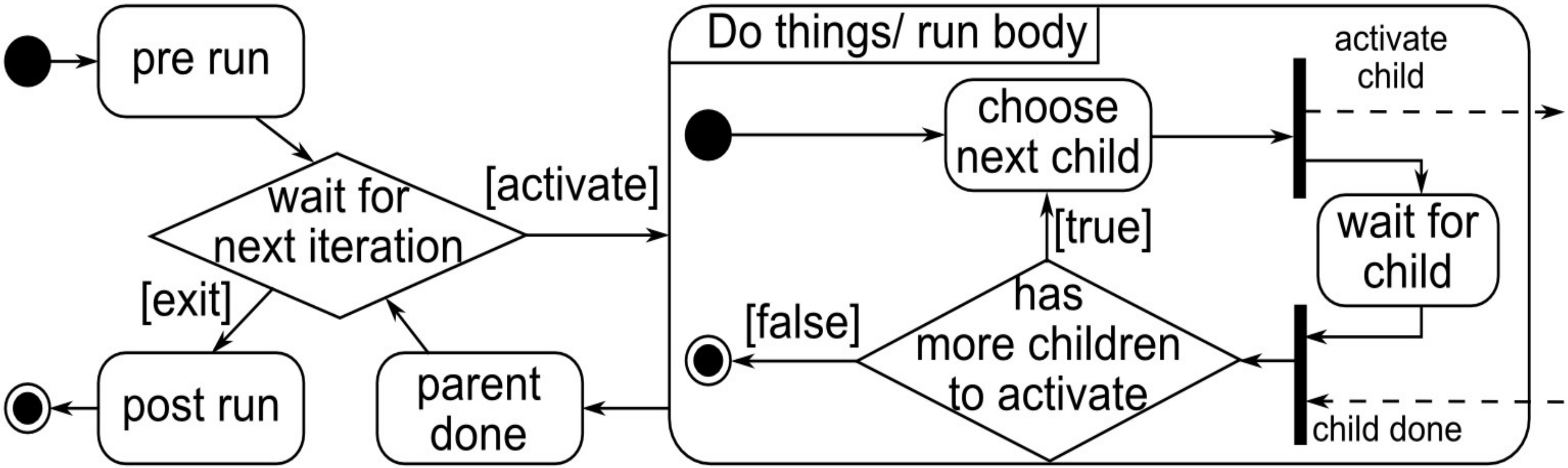
CSP Execution Engine

UNIVERSITY OF TWENTE.

- CSP Process
  - Initialise process (pre run)
  - Perform main operations
  - Finalise (post run)



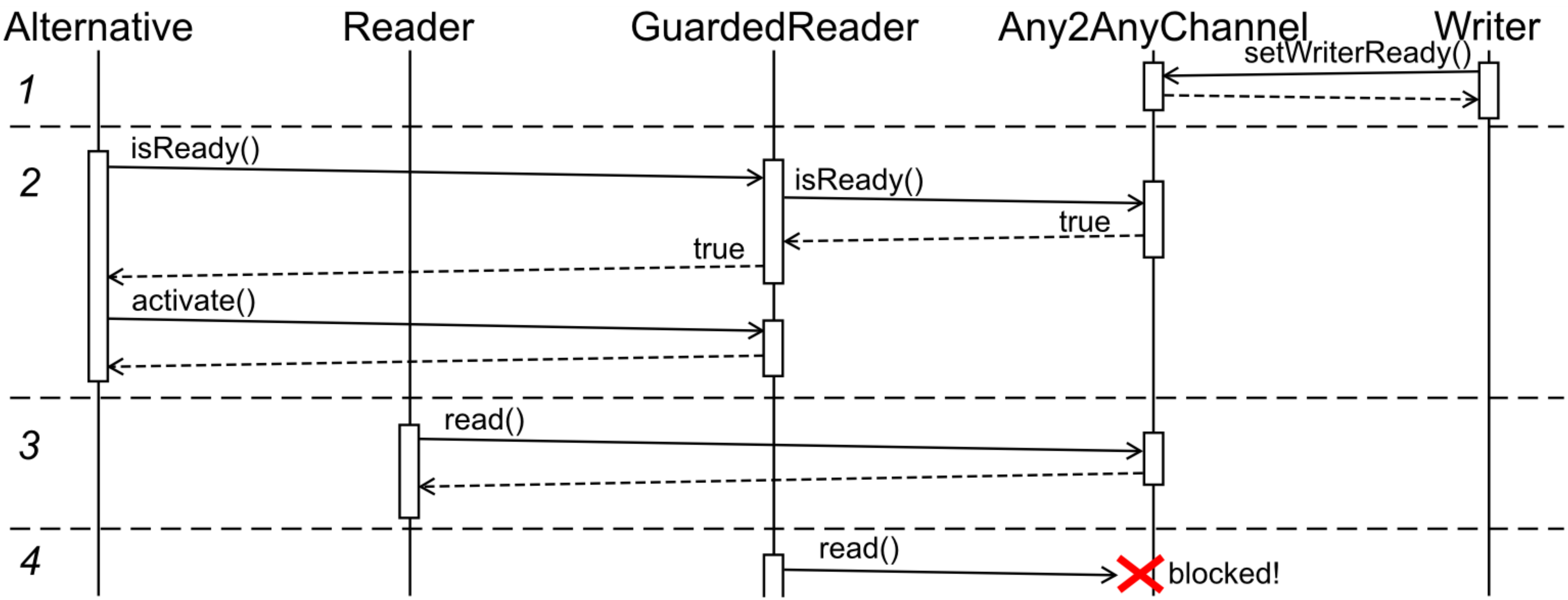
- Example of a sequential process



Alternative

- Naive Alternate implementation
  - Possibility of 'high-jacking' the channel, blocks GuardedReader

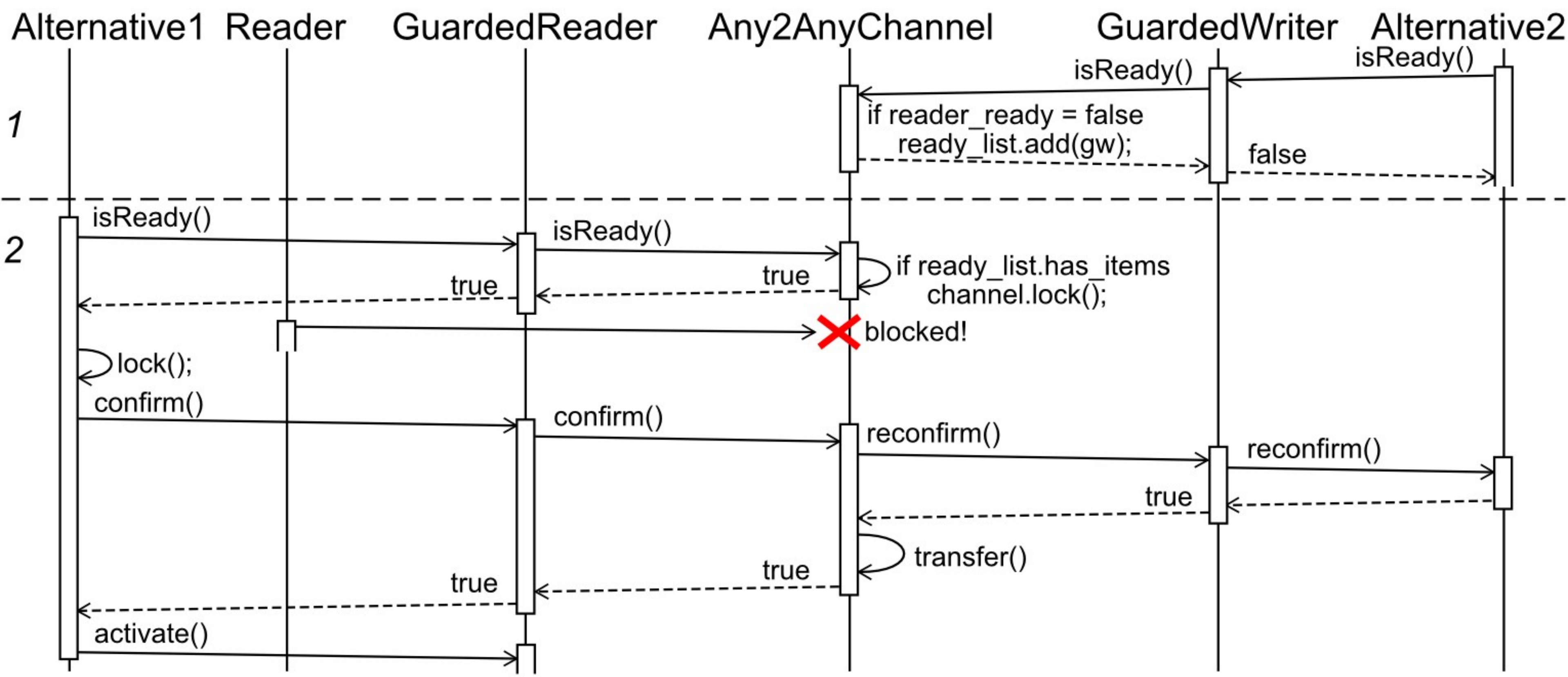
- Example: 1 GuardedReader, 1 'regular' Reader



Alternative

UNIVERSITY OF TWENTE.

- Solution for the high-jacking problem
  - Added lock to channel, now Reader blocks



Results

- Context-switch speed
  - Switch as fast as possible between two threads
- Commstime
  - Determine CSP efficiency
- Real robotic set up
  - Performance in real life situations

Results

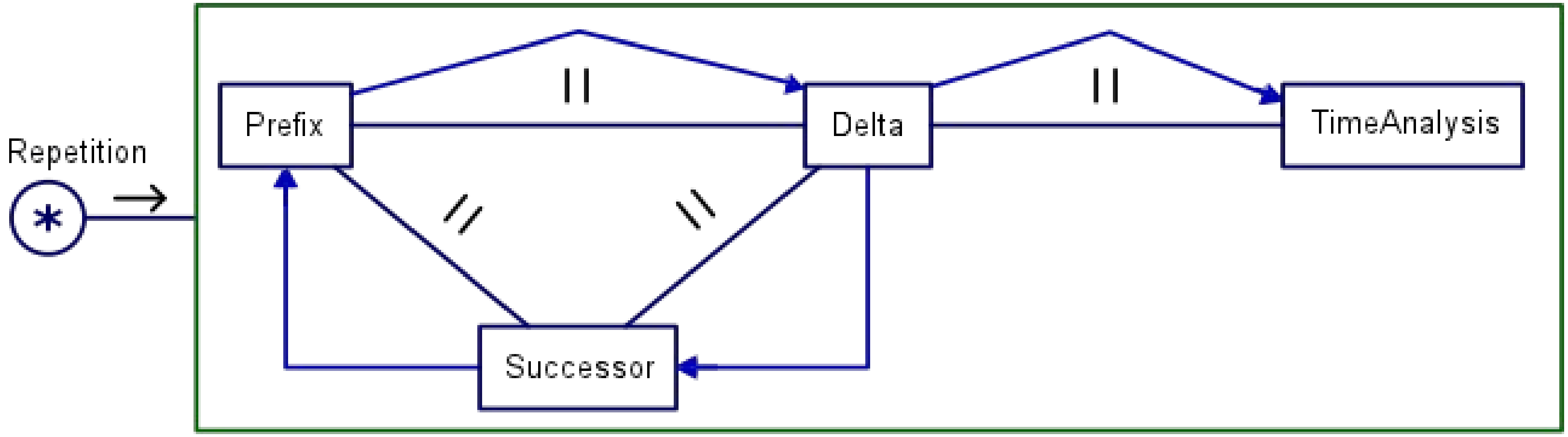
- Context-switch speed
  - Switch as fast as possible between two threads
  - 10,000 switches, average time

Framework	OS thread (µs)	User thread (µs)
CTC++ 'original'	-	4.275
C++CSP2	3.224	3.960
CTC++ QNX	3.213	-
LUNA QNX	3.226	1.569

- OS thread switch speed is comparable
- User thread switch speed is fast!
  - LUNA has virtually no management overhead
  - (high speeds only do not determine the framework efficiency)

Results

- Commstime Benchmark
  - Measure the efficiency of the CSP execution
  - 10,000 cycles, average time



Results

UNIVERSITY OF TWENTE.

- Commstime Benchmark

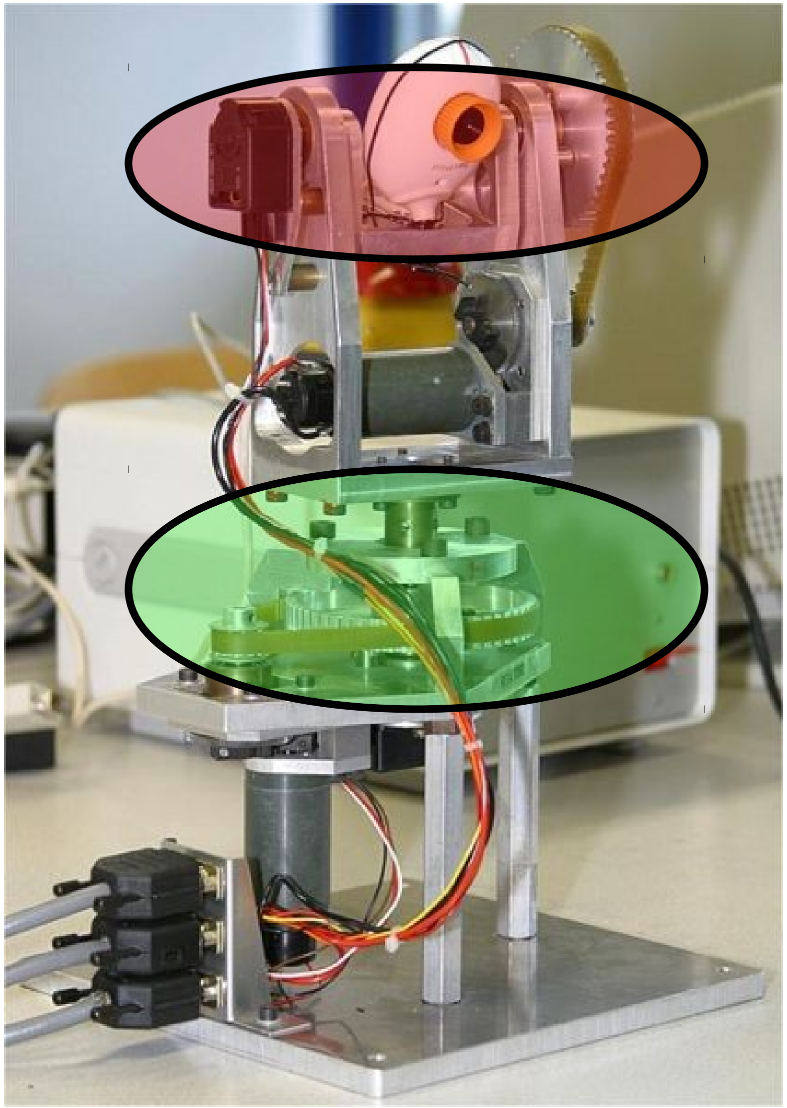
Framework	Thread type	Cycle time (µs)	# Context-switches
CTC++ 'original'	User	40.76	5
C++CSP2	OS	44.59	-
	User	18.60	4
CTC++ QNX	OS	57.06	-
LUNA QNX	OS	34.03	-
	User	9.34	4

- OS thread cycle time somewhat faster
  - Efficient way to block a OS thread (low management)
- User thread cycle time fast!
  - Mainly due to efficient context-switching
- Naive code generation results in bad performance
  - Design point of view versus execution point of view

Results

- Simple 2 DOF pan-tilt robotic set up

- Used for educational purposes
  - Practical assignments
    - Easy platform for experimenting
    - Vision-in-the-loop
    - Spot tracking
  - Courses
    - Real-time software development
    - Hardware/Software trade-offs



Results

- Real Robotic Set up
  - Performance in real life situations
  - Measurement runs of ~60 seconds

Framework	Frequency (Hz)	Cycle time (ms)			Standard deviation (μs)	Processing time (μs)
		Mean	Min	Max		
CTC++ 'original'	100	11.00	10.90	11.11	14.8	199.0
	1000	1.18	0.91	2.10	386.5	174.5
	1000.15	1.00	0.91	1.10	20.7	172.5
LUNA QNX	100	10.00	9.93	11.00	39.6	111.6
(user threads)	1000	1.00	0.80	2.01	35.8	89.3
	1000.15	1.00	0.79	1.21	33.2	87.3
LUNA QNX	100	10.00	9.97	11.00	39.1	214.3
(OS threads)	1000	1.00	0.96	2.00	14.4	185.6
	1000.15	1.00	0.95	1.05	8.3	190.8

Results

UNIVERSITY OF TWENTE.

- Real Robotic Set up

Framework	Frequency (Hz)	Cycle time (ms)			Standard deviation (µs)	Processing time (µs)
		Mean	Min	Max		
CTC++ 'original'	100	11.00	10.90	11.11	14.8	199.0
	1000.15	1.00	0.91	1.10	20.7	172.5
LUNA QNX	100	10.00	9.93	11.00	39.6	111.6
(user threads)	1000.15	1.00	0.79	1.21	33.2	87.3
LUNA QNX	100	10.00	9.97	11.00	39.1	214.3
(OS threads)	1000.15	1.00	0.95	1.05	8.3	190.8

- LUNA user threads are faster than CTC++
- LUNA OS threads are slightly slower than CTC++ (user threads!)

Conclusions

UNIVERSITY OF TWENTE.

- LUNA meets all requirements
  - Hard real-time
  - Multi-platform
  - Multi-threaded
  - Scalable

- Fast and efficient compared to related frameworks

- Usable for controlling real robotic set ups

- Need model optimisation for code generation

Future work

- Develop controller for Production Cell with LUNA
  - To show that complex set ups can also controlled using LUNA
- Support Linux, RTAI and/or Xenomai
  - More drivers available to use webcams, joysticks, …
- Support for Windows
  - Well known by (starting) developers
  - Good (graphical) debugging facilities
- Graphical CSP modelling tool with code generation capabilities
  - Replacement for gCSP
  - Model optimisation algorithms included