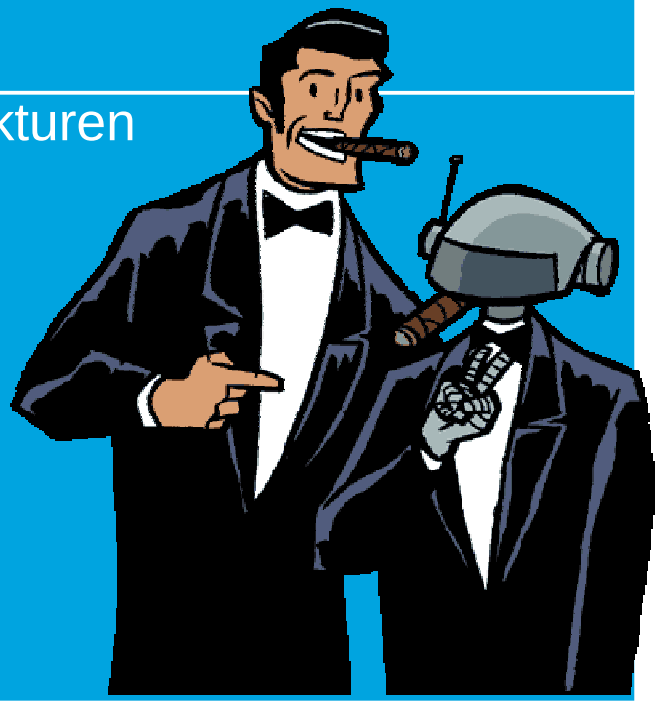
Embedded Software Engineering Kongress

Datum:
01.12.2015
08:50 – 09:35 h

Betriebssystem: mit oder ohne?

Vergleich und Auswahl von Software-Laufzeitarchitekturen

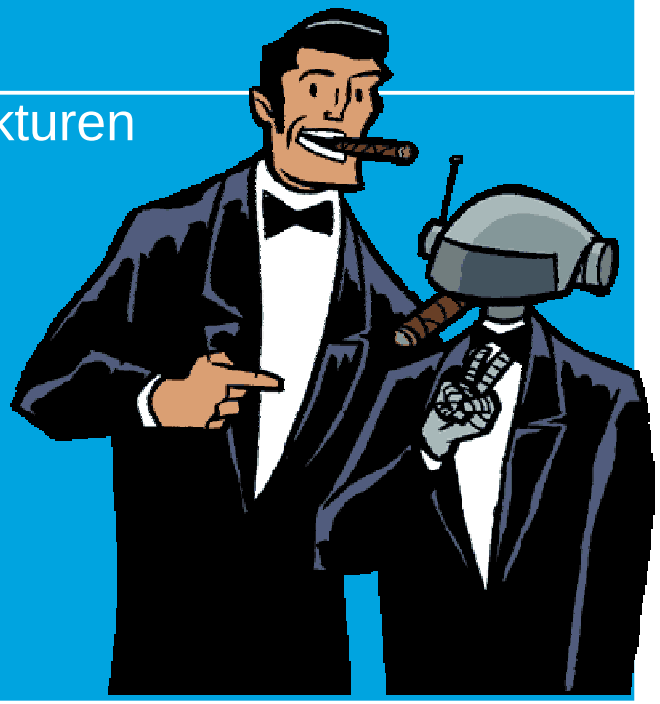


MicroConsult GmbH
Dipl.-Ing. (FH) **Thomas Batt**
Manager Management-Training und -Coaching
Trainer & Coach für Embedded- und Echtzeitsysteme
Charles-de-Gaulle-Str. 6 • 81737 München • Germany
Tel.: +49 (0)89 450617-35
E-Mail: t.batt@microconsult.com
Internet: www.microconsult.de

Embedded Software Engineering Kongress

Betriebssystem: mit oder ohne?

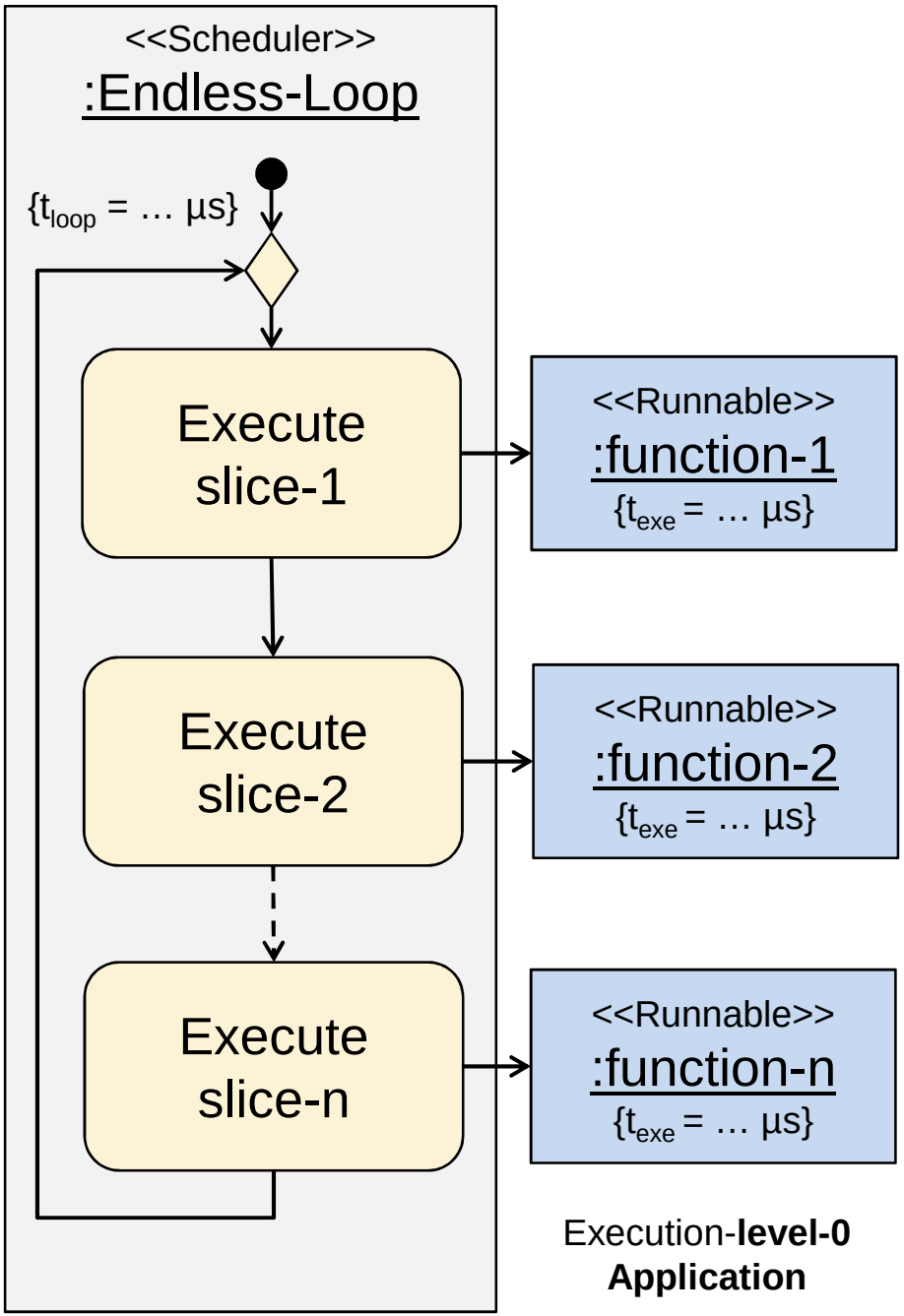
Vergleich und Auswahl von Software-Laufzeitarchitekturen



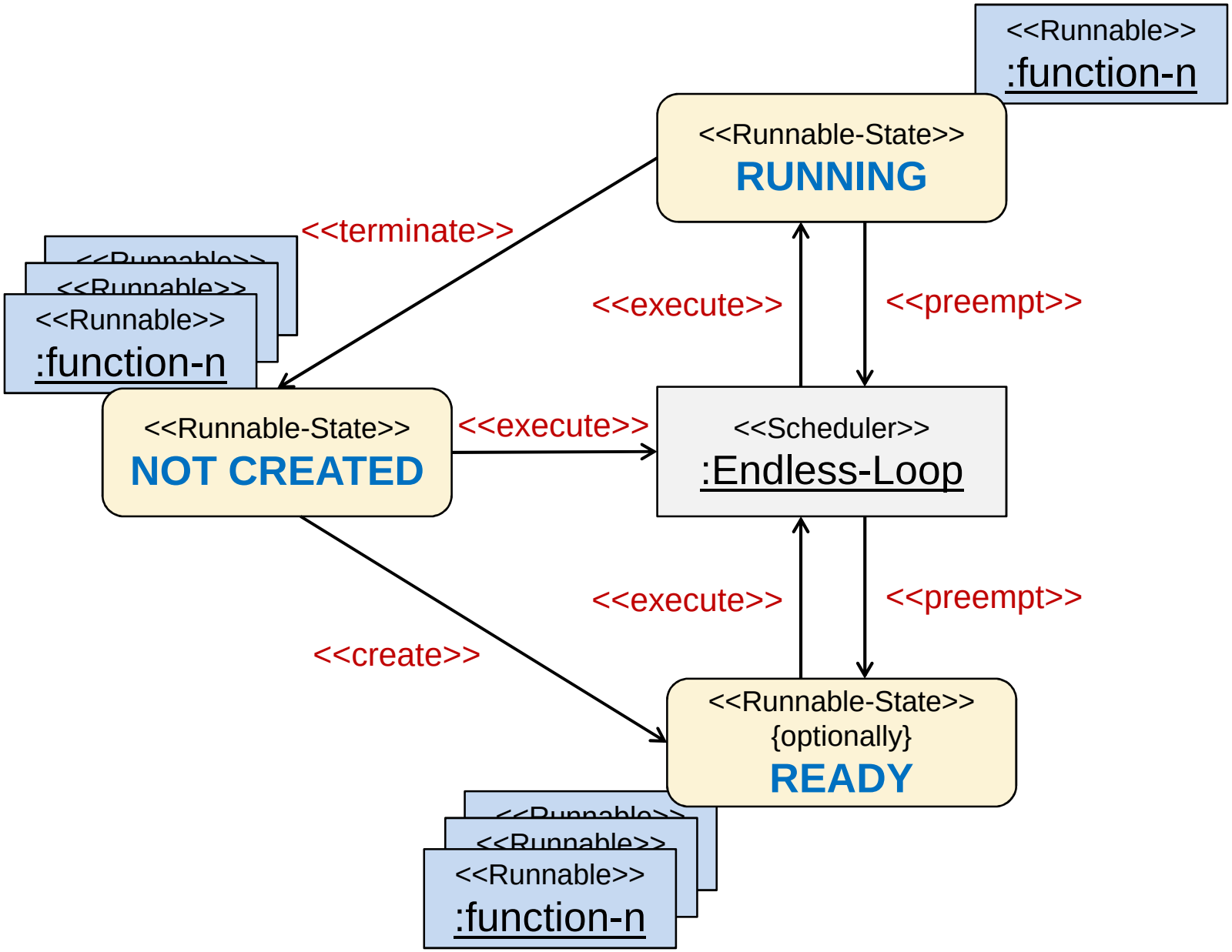
MicroConsult GmbH
Dipl.-Ing. (FH) **Thomas Batt**
Manager Management-Training und -Coaching
Trainer & Coach für Embedded- und Echtzeitsysteme
Charles-de-Gaulle-Str. 6 • 81737 München • Germany
Tel.: +49 (0)89 450617-35
E-Mail: t.batt@microconsult.com
Internet: www.microconsult.de

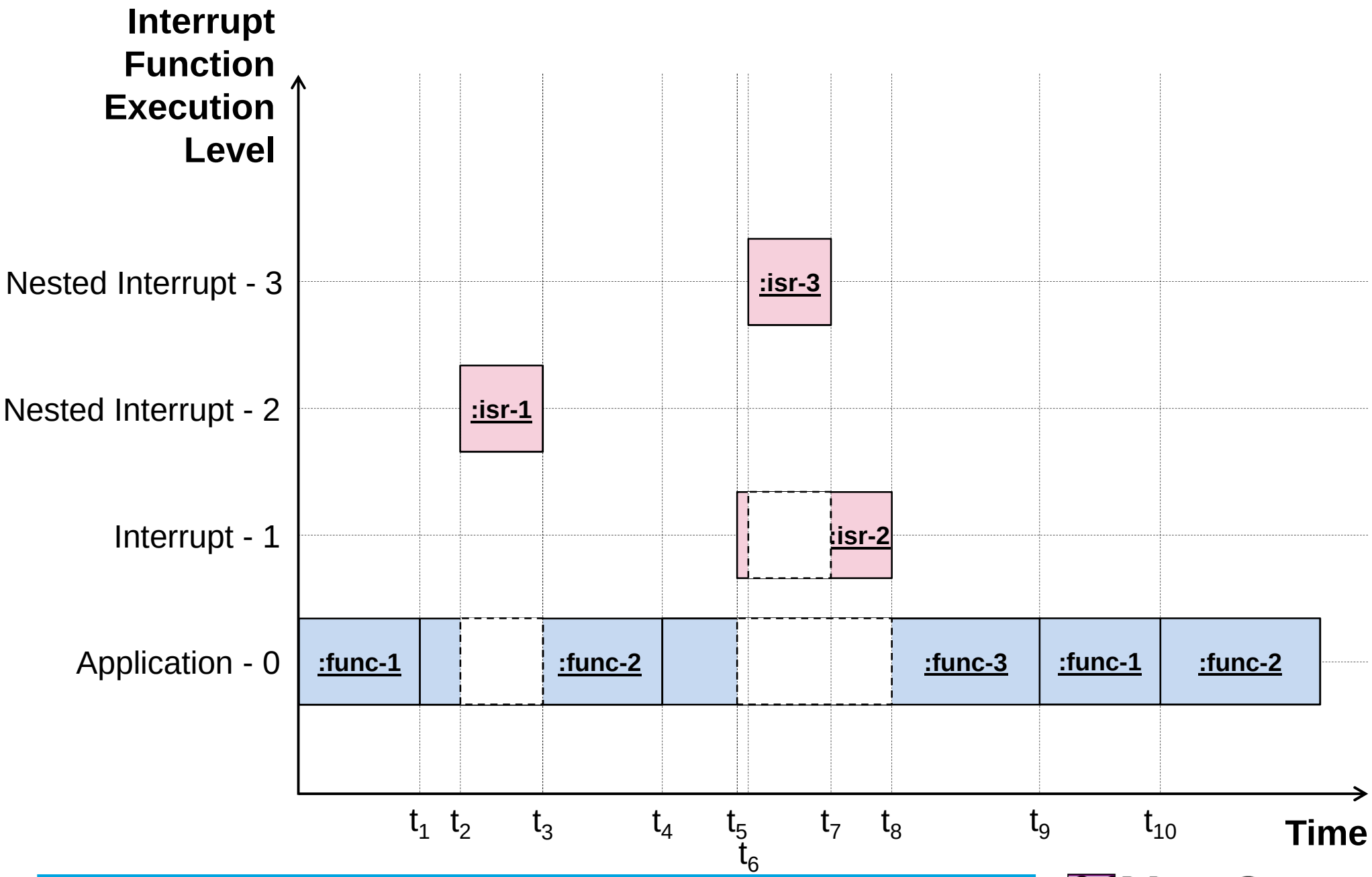
- Vorstellung und Auswahlhilfen für verschiedene Laufzeitarchitekturen:
  - Sequential Endless-Loop Scheduling
  - Time-Triggered Scheduling
  - Priority Scheduling
  - Priority and Time-Slice Scheduling
  - Earliest-Deadline-First Scheduling
- Embedded-Softwarearchitektur
- Resümee

Download-Link für diese Präsentation:
http://download.microconsult.net/ese2015/laufzeitarchitekturen.zip



Execution-**level-0**
Application


<<Runnable>>
:function-n
<<Runnable-State>>
RUNNING
<<terminate>>
<<Runnable>>
:function-n
<<execute>>
<<preempt>>
<<Runnable-State>>
NOT CREATED
<<execute>>
<<Scheduler>>
:Endless-Loop
<<execute>>
<<preempt>>
<<create>>
<<Runnable-State>>
{optionally}
READY
<<Runnable>>
:function-n


Interrupt Function Execution Level
Nested Interrupt - 3
Nested Interrupt - 2
Interrupt - 1
Application - 0
:isr-3
:isr-1
:isr-2
:func-1
:func-2
:func-3
:func-1
:func-2
t1
t2
t3
t4
t5
t6
t7
t8
t9
t10
Time

Simple

Continuous function execution

Functions with fewer dependencies

…

(Quasi) parallel execution of functions of no benefit

Application Criterion

Sequential Endless-Loop Scheduling

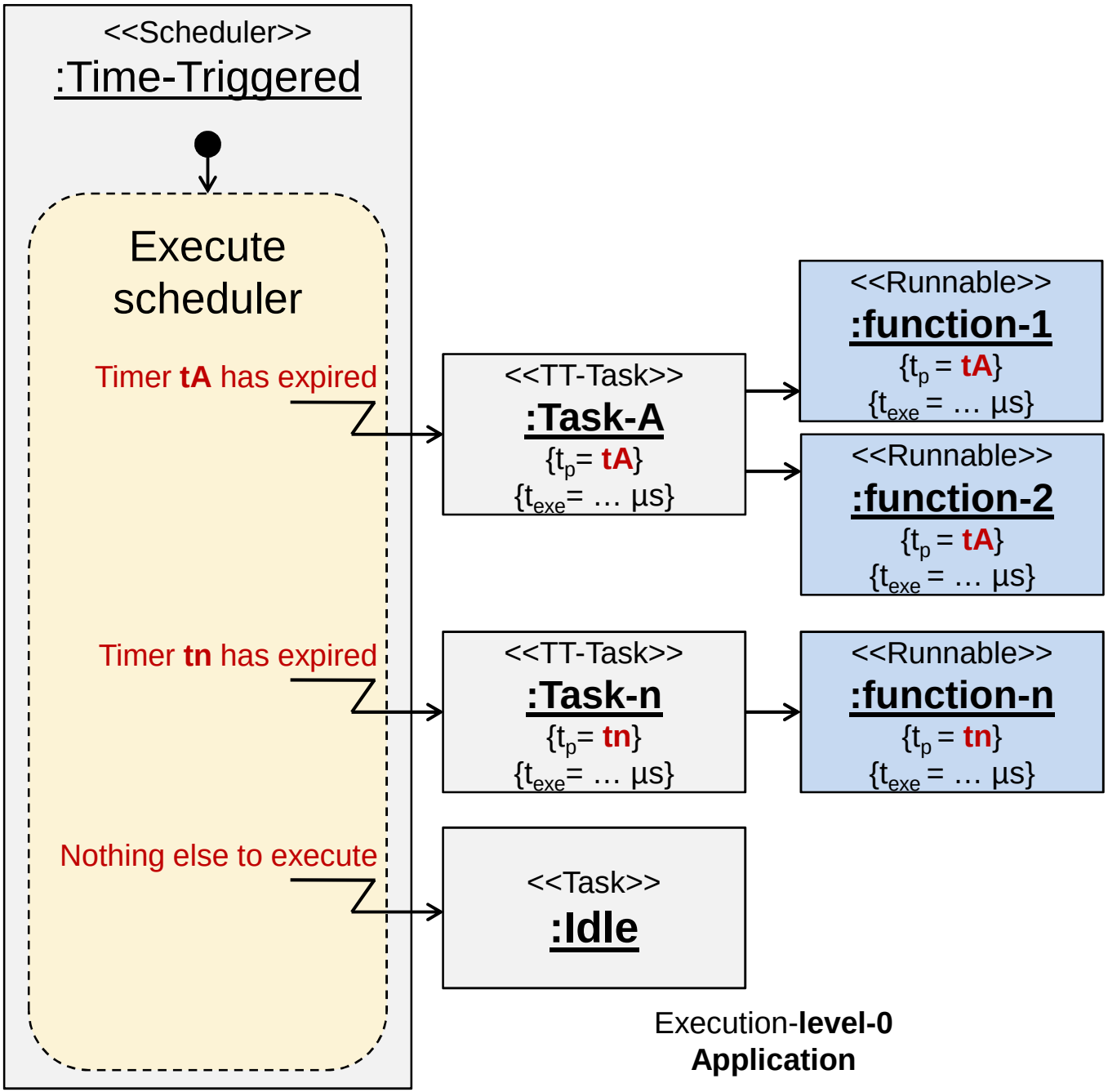
…

Function execution frequency is uncritical

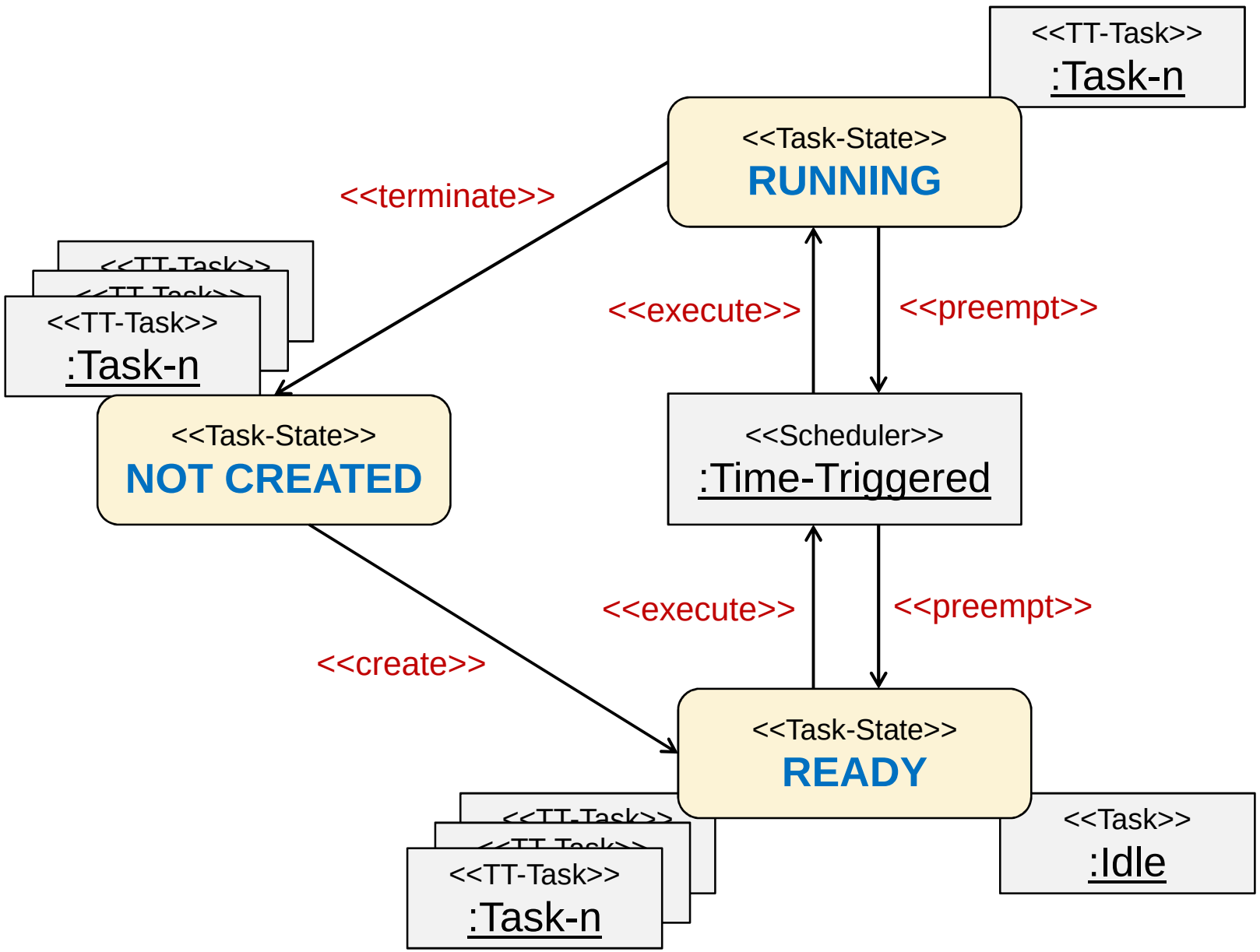
…

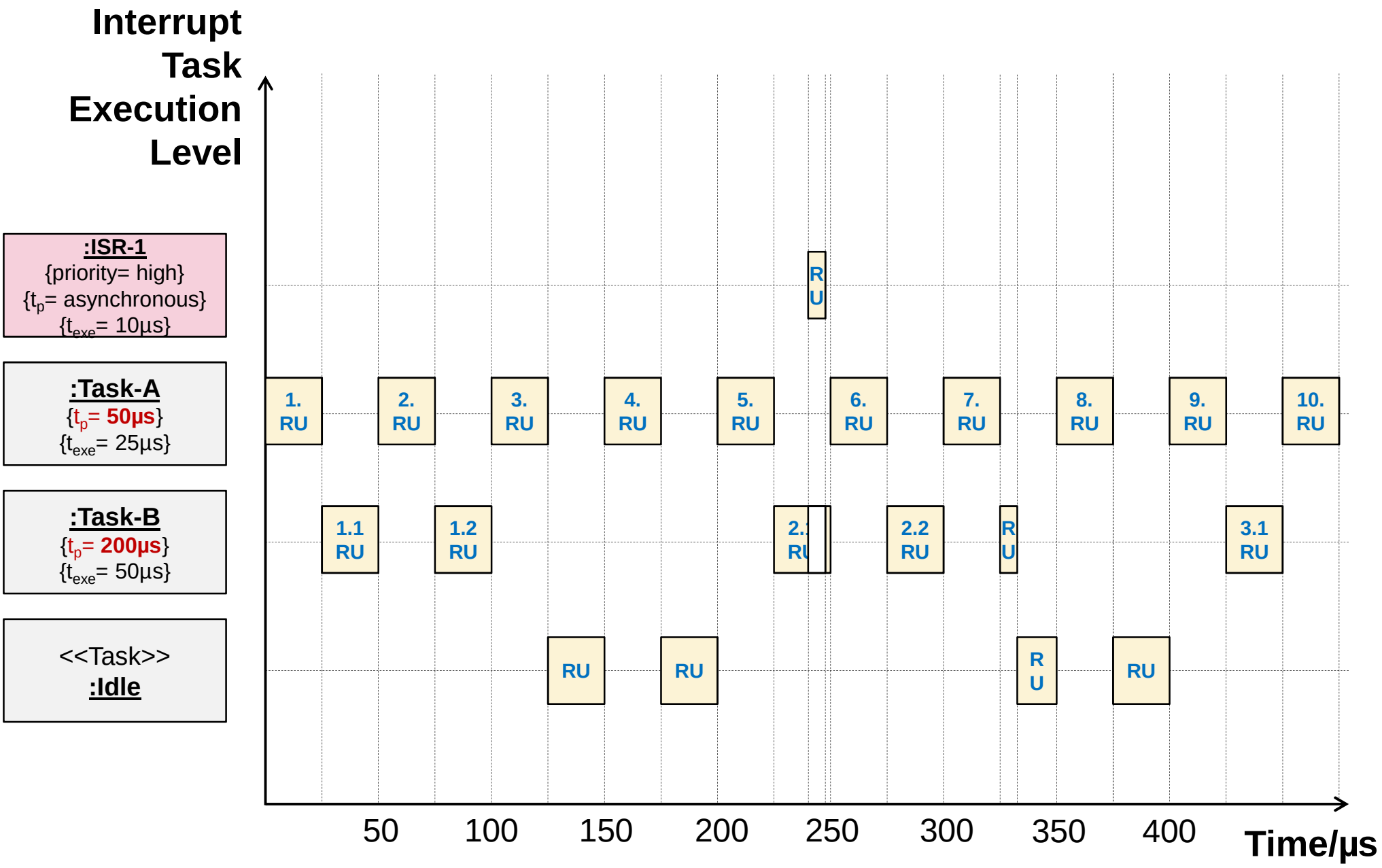
No future requirements for expandability

Only synchronous functions



Execution-**level-0**
Application


<<TT-Task>>
:Task-n
<<Task-State>>
RUNNING
<<terminate>>
<<execute>>
<<preempt>>
<<TT-Task>>
:Task-n
<<Task-State>>
NOT CREATED
<<Scheduler>>
:Time-Triggered
<<execute>>
<<preempt>>
<<create>>
<<Task-State>>
READY
<<TT-Task>>
:Task-n
<<Task>>
:Idle

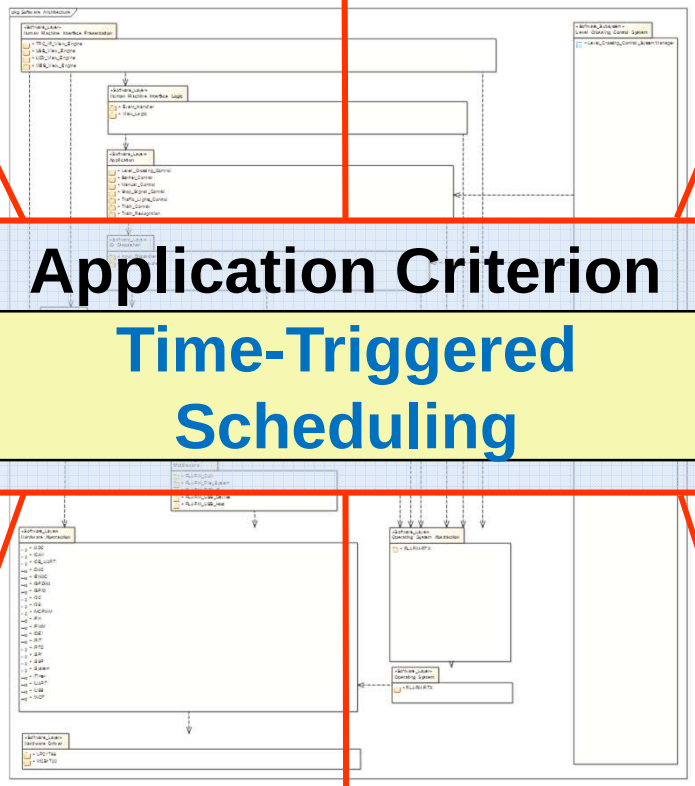

Interrupt
Task
Execution
Level
:ISR-1
{priority= high}
{tp= asynchronous}
{texe= 10µs}
:Task-A
{tp= 50µs}
{texe= 25µs}
:Task-B
{tp= 200µs}
{texe= 50µs}
<<Task>>
:Idle
RU
1. RU
2. RU
3. RU
4. RU
5. RU
6. RU
7. RU
8. RU
9. RU
10. RU
1.1 RU
1.2 RU
2.2 RU
3.1 RU
50
100
150
200
250
300
350
400
Time/µs

(Different) tasks have to be executed with the same periodicity

Different execution periodicities are required

Continuous task execution

…



Tasks with fewer dependencies

Application Criterion

Time-Triggered Scheduling

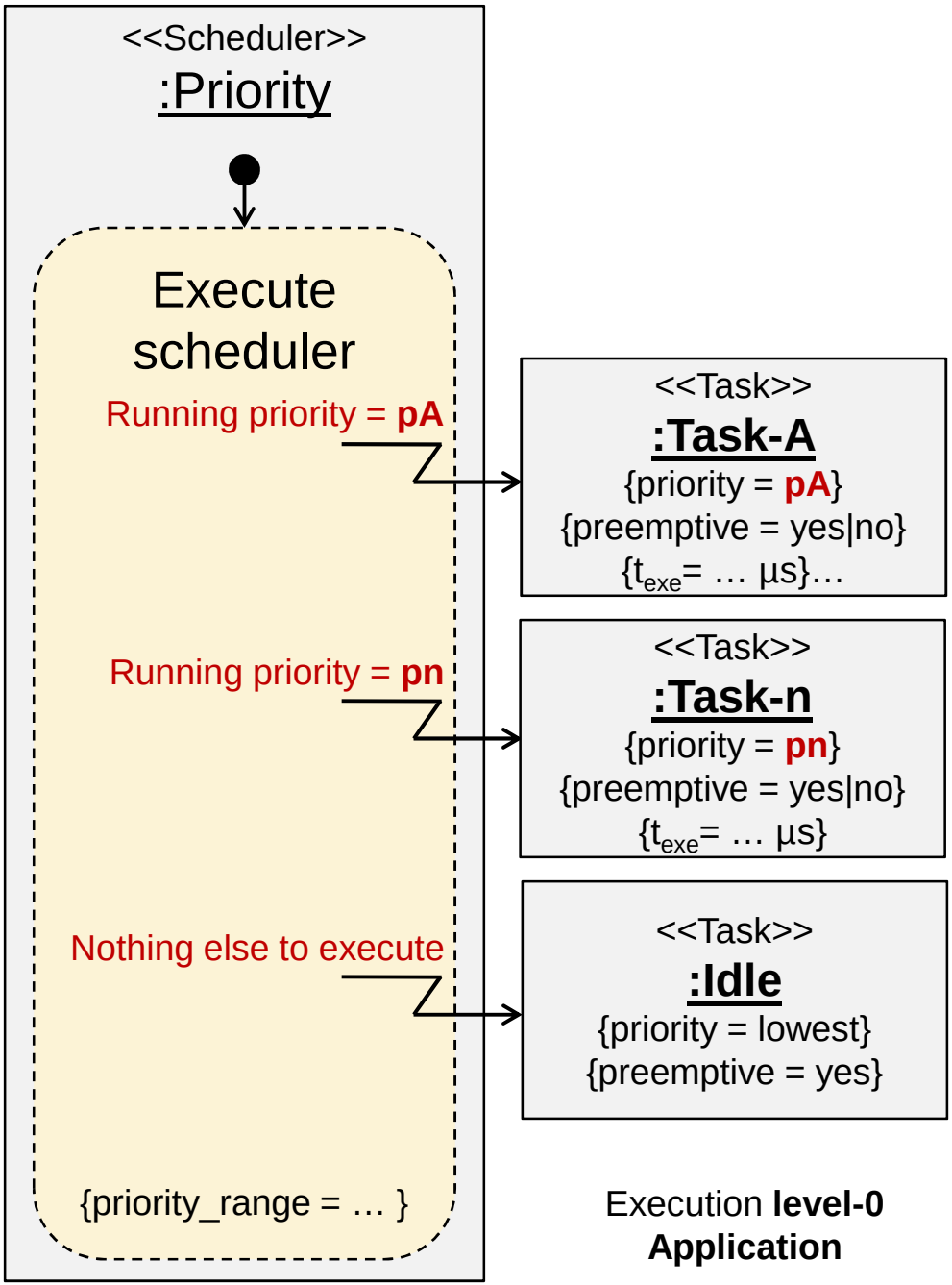
…

Only synchronous tasks

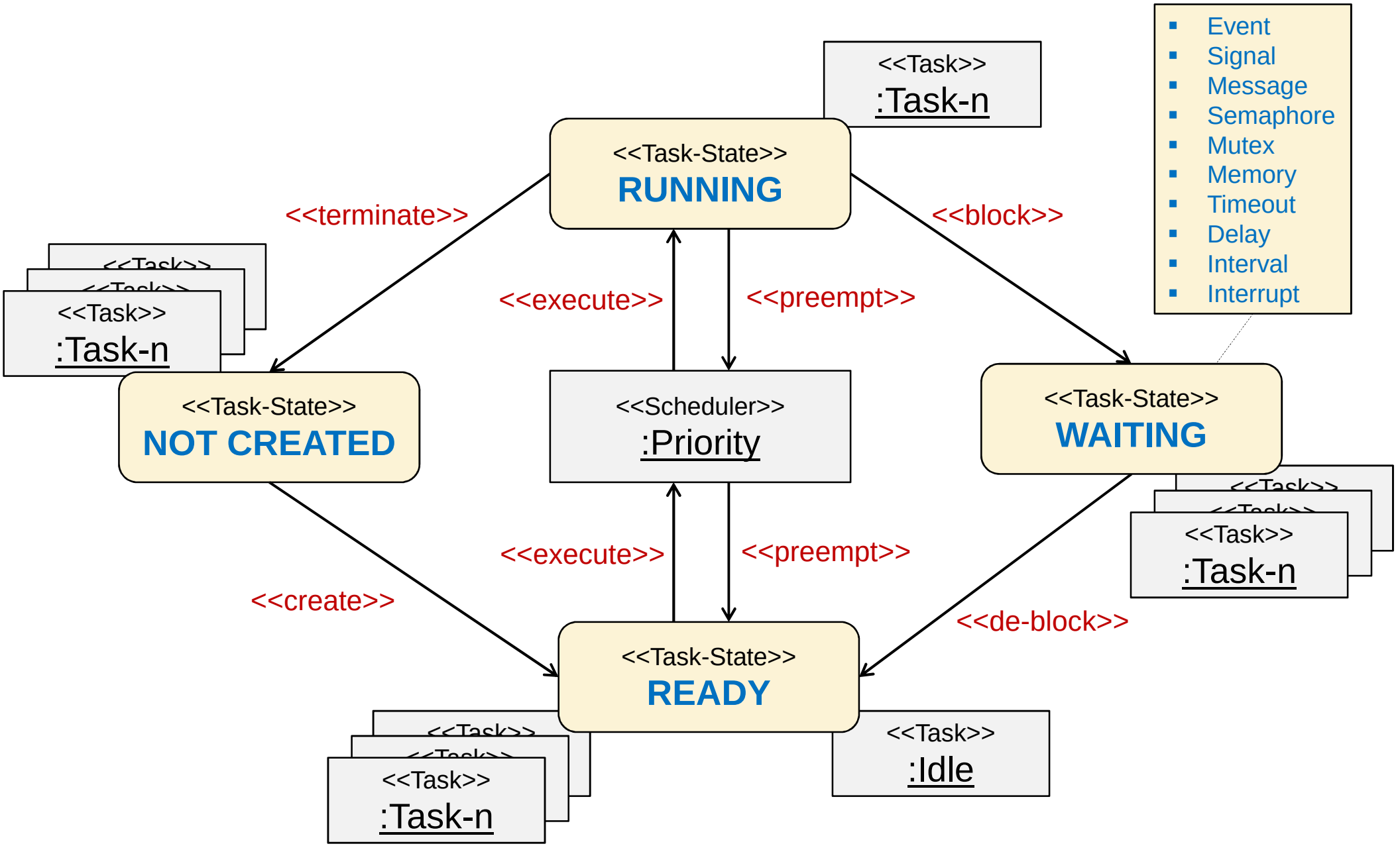
High degree of determinism
➔ calculability

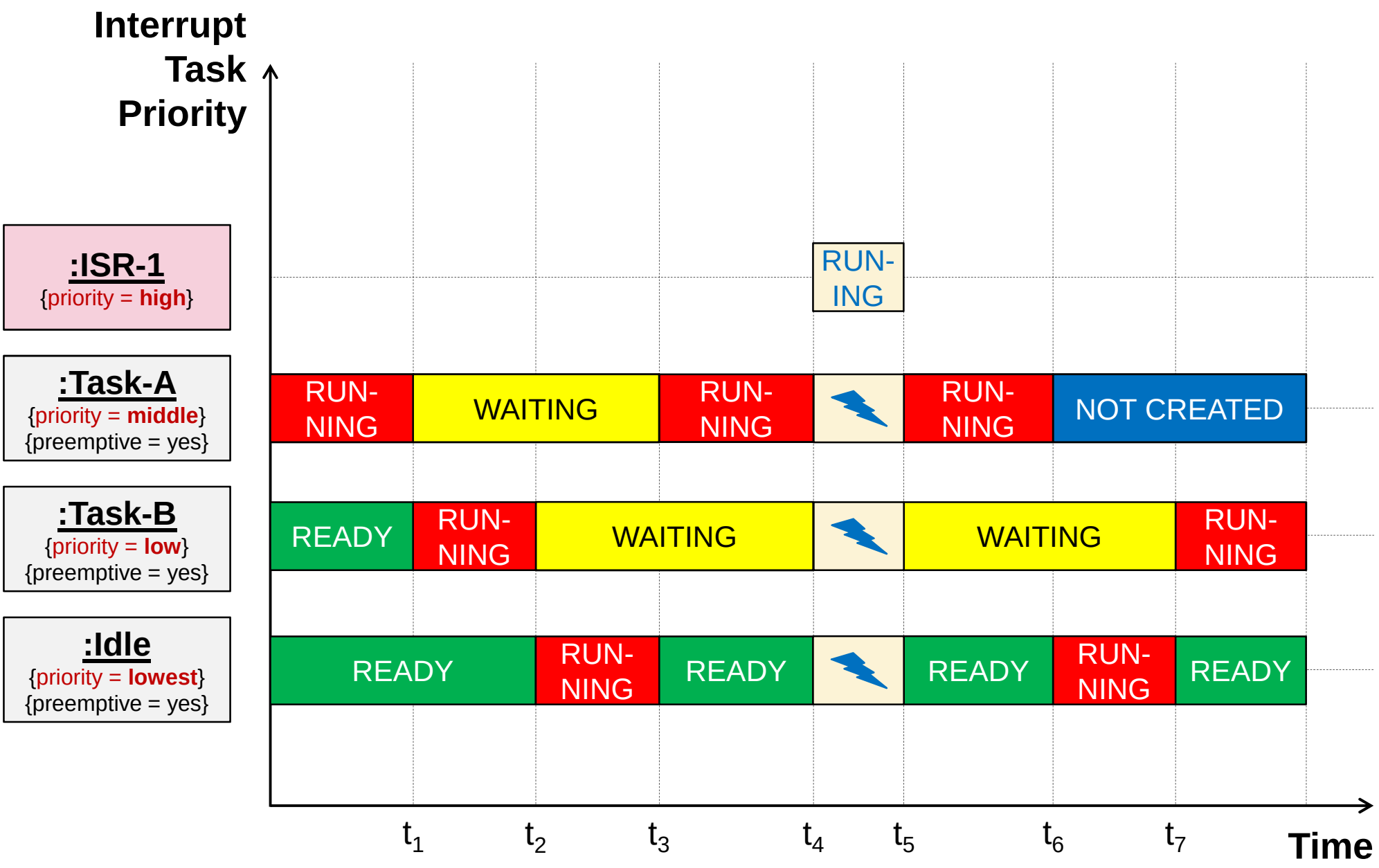
High performance requirements

Accurate timing requirements



Execution **level-0 Application**


Event
Signal
Message
Semaphore
Mutex
Memory
Timeout
Delay
Interval
Interrupt
<<Task>>
:Task-n
<<Task-State>>
RUNNING
<<terminate>>
<<block>>
<<execute>>
<<preempt>>
<<Task>>
:Task-n
<<Task-State>>
NOT CREATED
<<Scheduler>>
:Priority
<<Task-State>>
WAITING
<<Task>>
:Task-n
<<execute>>
<<preempt>>
<<create>>
<<de-block>>
<<Task-State>>
READY
<<Task>>
:Task-n
<<Task>>
:Idle

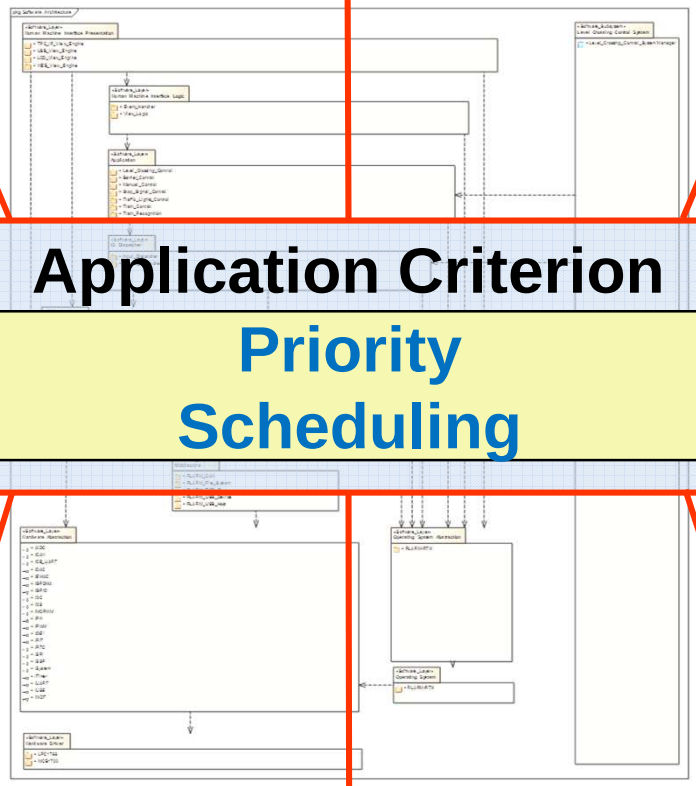

Interrupt
Task
Priority
:ISR-1
{priority = high}
:Task-A
{priority = middle}
{preemptive = yes}
:Task-B
{priority = low}
{preemptive = yes}
:Idle
{priority = lowest}
{preemptive = yes}
RUN-
ING
RUN-
NING
WAITING
RUN-
NING
RUN-
NING
NOT CREATED
READY
RUN-
NING
WAITING
WAITING
RUN-
NING
READY
RUN-
NING
READY
READY
RUN-
NING
READY
t1
t2
t3
t4
t5
t6
t7
Time

Complex tasks

High degree of task interaction

Asynchronous tasks

…



(Quasi) parallel task execution of benefit

Application Criterion

Priority Scheduling

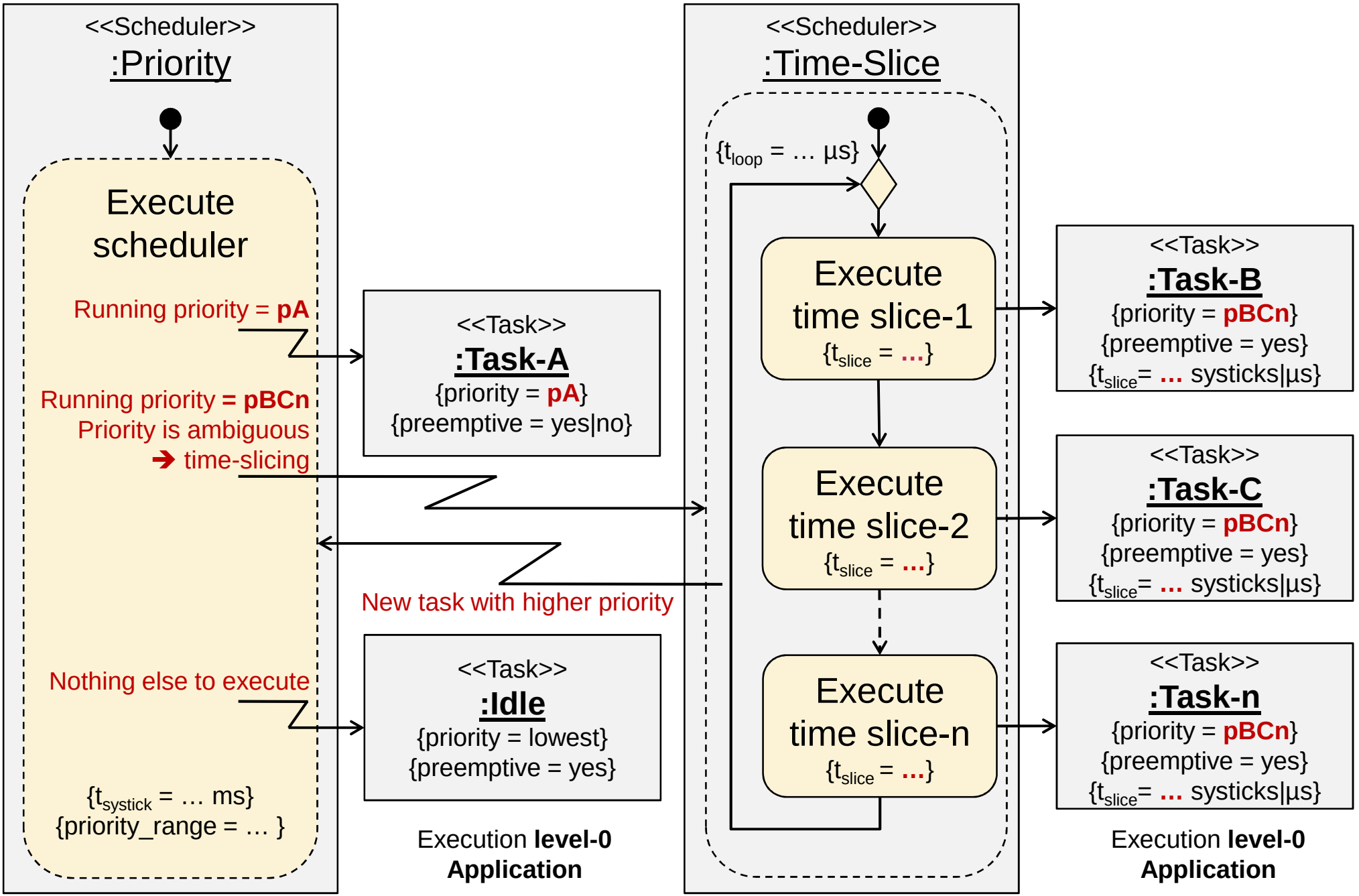
…

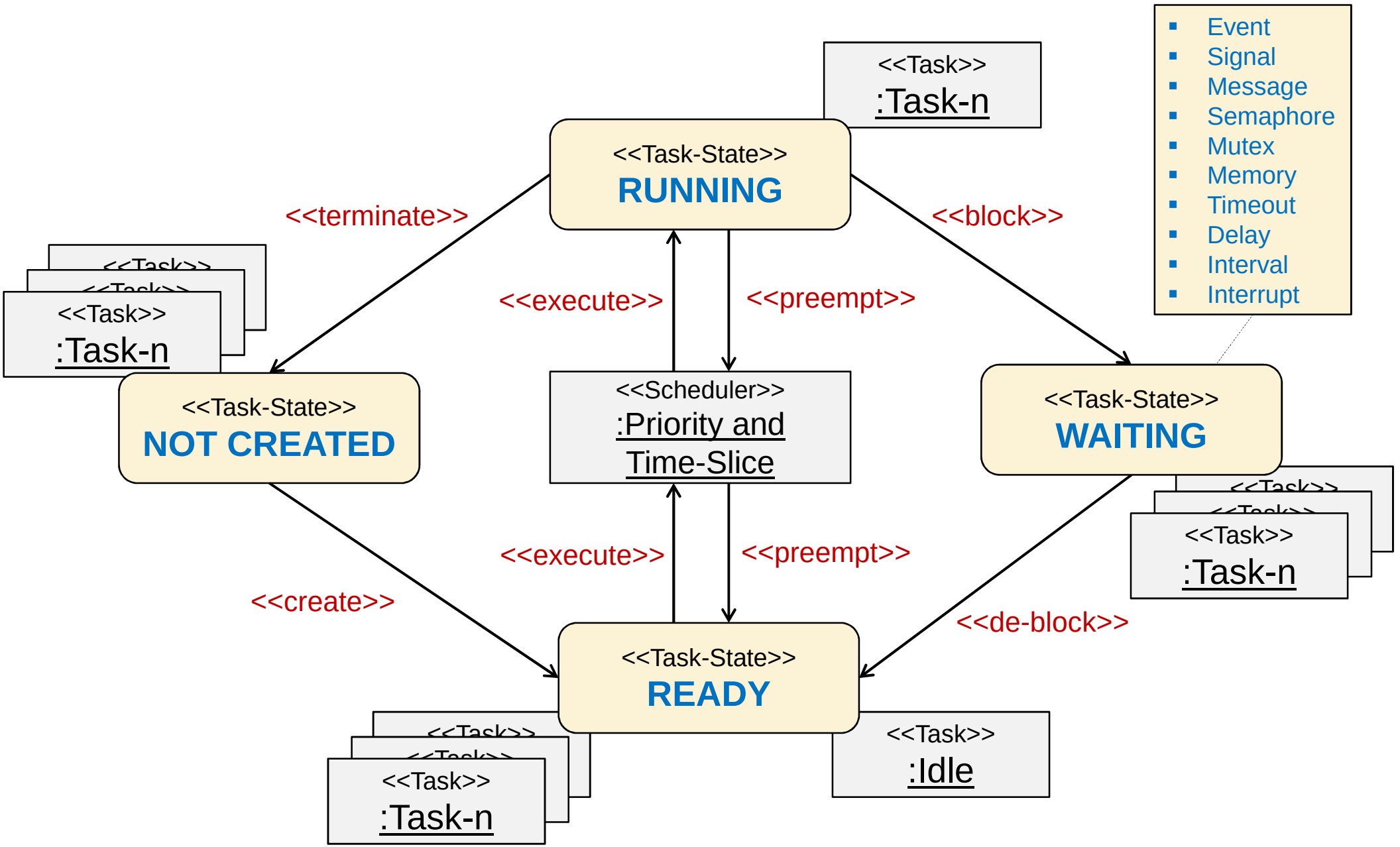
Fewer timing requirements

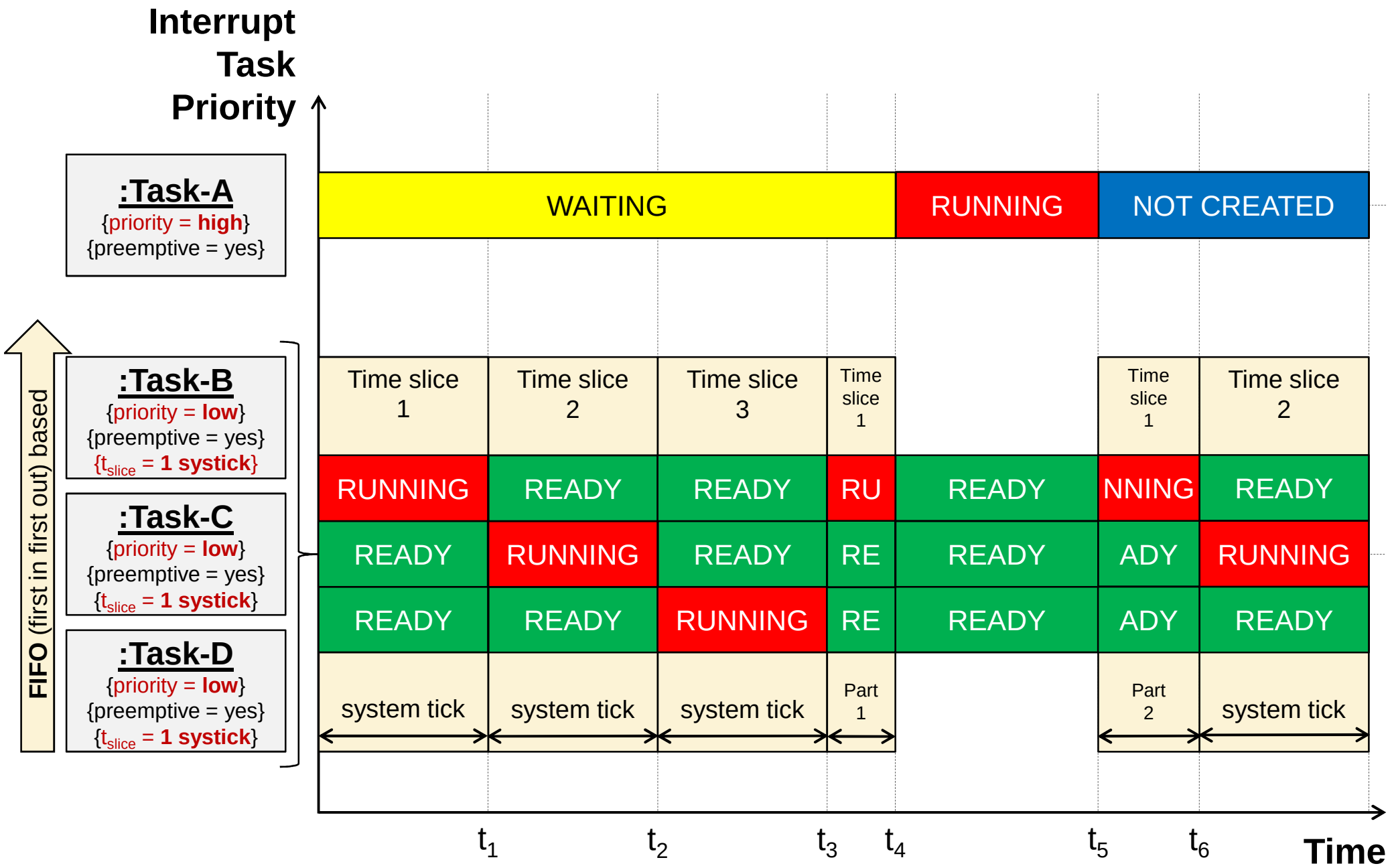
…

Integration of other/ different software components

Future expandability


<<Scheduler>>
:Priority
Execute scheduler
Running priority = pA
Running priority = pBCn
Priority is ambiguous
➔ time-slicing
Nothing else to execute
{t_systick = … ms}
{priority_range = … }
<<Task>>
:Task-A
{priority = pA}
{preemptive = yes|no}
New task with higher priority
<<Task>>
:Idle
{priority = lowest}
{preemptive = yes}
Execution level-0 Application
<<Scheduler>>
:Time-Slice
{t_loop = … µs}
Execute time slice-1
{t_slice = …}
Execute time slice-2
{t_slice = …}
Execute time slice-n
{t_slice = …}
<<Task>>
:Task-B
{priority = pBCn}
{preemptive = yes}
{t_slice= … systicks|µs}
<<Task>>
:Task-C
{priority = pBCn}
{preemptive = yes}
{t_slice= … systicks|µs}
<<Task>>
:Task-n
{priority = pBCn}
{preemptive = yes}
{t_slice= … systicks|µs}
Execution level-0 Application


Event
Signal
Message
Semaphore
Mutex
Memory
Timeout
Delay
Interval
Interrupt
<<Task>>
:Task-n
<<Task-State>>
RUNNING
<<terminate>>
<<block>>
<<execute>>
<<preempt>>
<<Task>>
:Task-n
<<Task-State>>
NOT CREATED
<<Scheduler>>
:Priority and
Time-Slice
<<Task-State>>
WAITING
<<Task>>
:Task-n
<<execute>>
<<preempt>>
<<create>>
<<de-block>>
<<Task-State>>
READY
<<Task>>
:Task-n
<<Task>>
:Idle

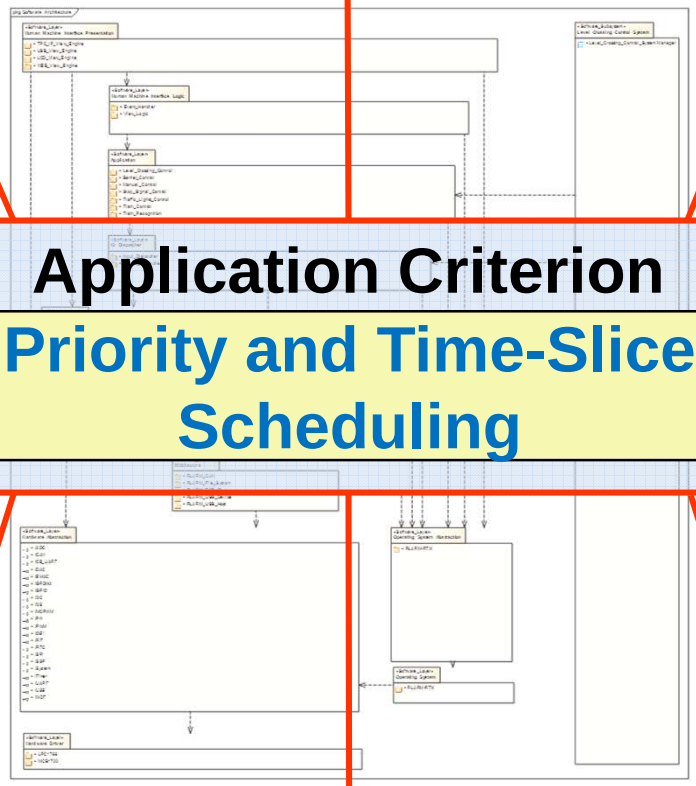

Interrupt
Task
Priority
:Task-A
{priority = high}
{preemptive = yes}
:Task-B
{priority = low}
{preemptive = yes}
{tslice = 1 systick}
:Task-C
{priority = low}
{preemptive = yes}
{tslice = 1 systick}
:Task-D
{priority = low}
{preemptive = yes}
{tslice = 1 systick}
FIFO (first in first out) based
WAITING
RUNNING
NOT CREATED
Time slice 1
Time slice 2
Time slice 3
Time slice 1
Time slice 1
Time slice 2
RUNNING
READY
READY
RU
READY
NNING
READY
READY
RUNNING
READY
RE
READY
ADY
RUNNING
READY
READY
RUNNING
RE
READY
ADY
READY
system tick
system tick
system tick
Part 1
Part 2
system tick
t1
t2
t3
t4
t5
t6
Time

Complex tasks

High degree of task interaction

Asynchronous tasks

…



(Quasi) parallel task execution of benefit

Application Criterion

Priority and Time-Slice Scheduling

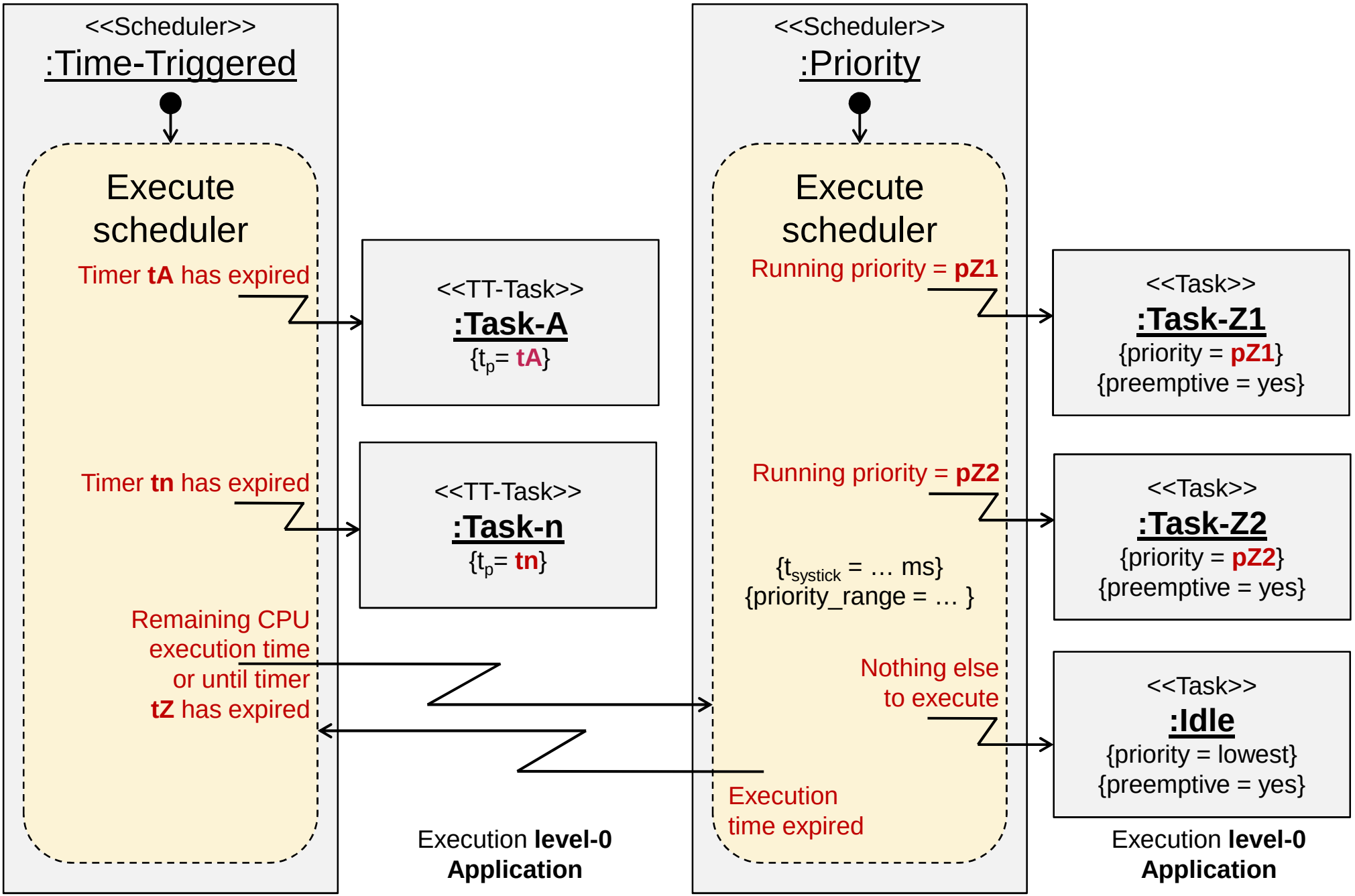
…

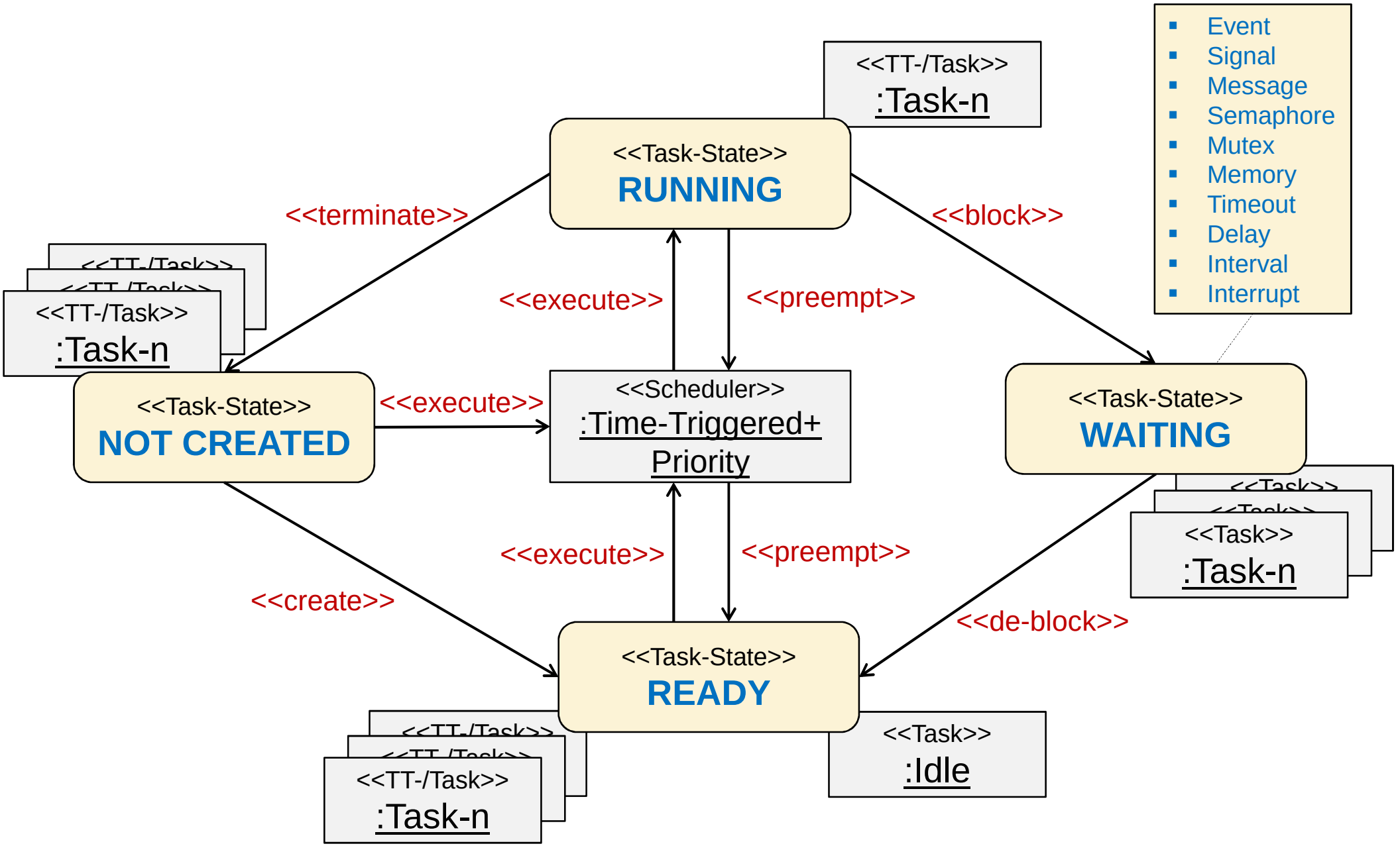
Fewer timing requirements

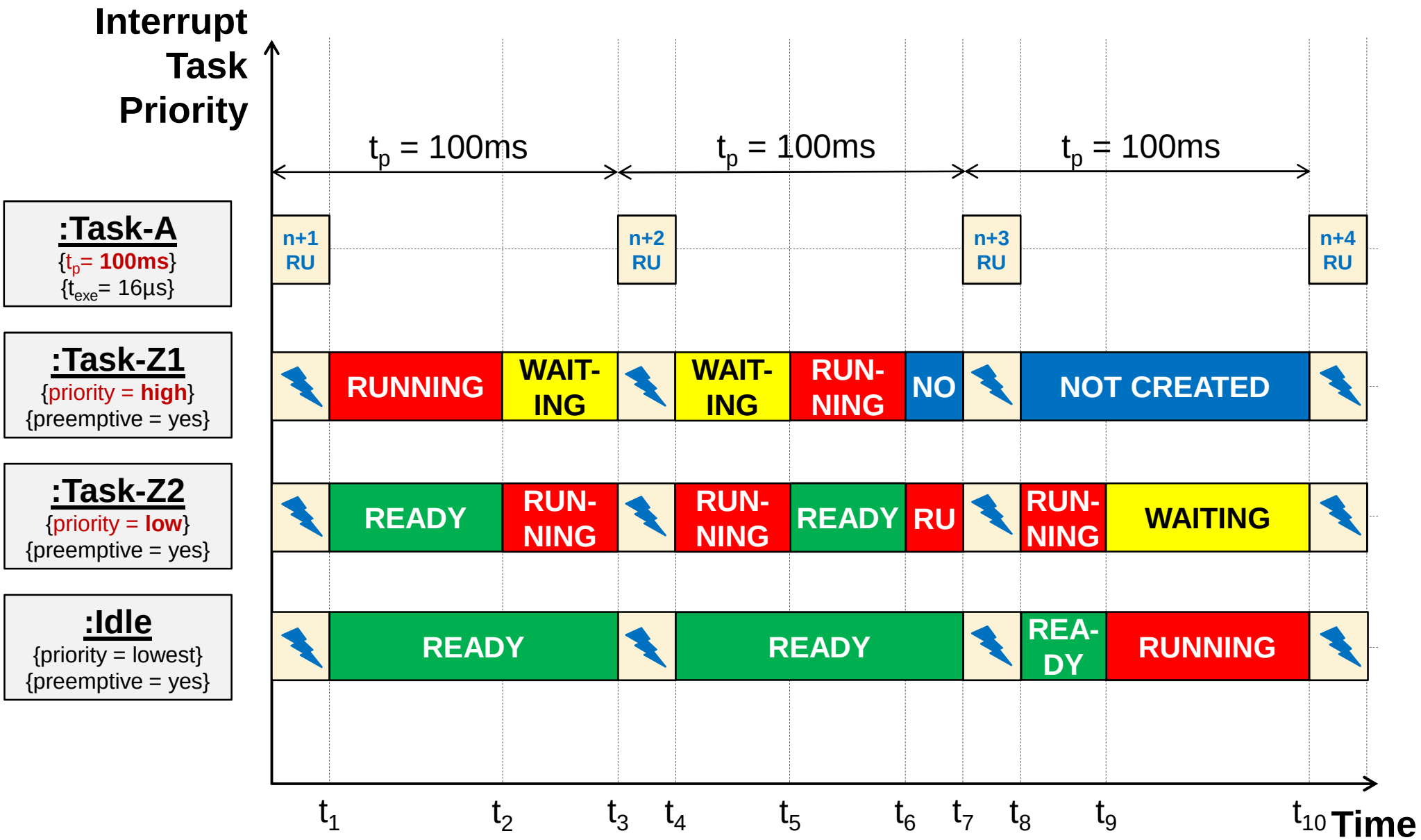
Time controlled task execution

Integration of other/ different software components

Future expandability


<<Scheduler>>
:Time-Triggered
Execute scheduler
Timer tA has expired
Timer tn has expired
Remaining CPU execution time or until timer tZ has expired
<<TT-Task>>
:Task-A
{tp= tA}
<<TT-Task>>
:Task-n
{tp= tn}
Execution level-0 Application
<<Scheduler>>
:Priority
Execute scheduler
Running priority = pZ1
Running priority = pZ2
{tsystick = … ms}
{priority_range = … }
Nothing else to execute
Execution time expired
<<Task>>
:Task-Z1
{priority = pZ1}
{preemptive = yes}
<<Task>>
:Task-Z2
{priority = pZ2}
{preemptive = yes}
<<Task>>
:Idle
{priority = lowest}
{preemptive = yes}
Execution level-0 Application


Event
Signal
Message
Semaphore
Mutex
Memory
Timeout
Delay
Interval
Interrupt
<<TT-/Task>>
:Task-n
<<Task-State>>
RUNNING
<<terminate>>
<<block>>
<<execute>>
<<preempt>>
<<TT-/Task>>
:Task-n
<<Task-State>>
NOT CREATED
<<execute>>
<<Scheduler>>
:Time-Triggered+
Priority
<<Task-State>>
WAITING
<<Task>>
:Task-n
<<execute>>
<<preempt>>
<<create>>
<<de-block>>
<<Task-State>>
READY
<<TT-/Task>>
:Task-n
<<Task>>
:Idle


Interrupt
Task
Priority
:Task-A
{tp= 100ms}
{texe= 16µs}
:Task-Z1
{priority = high}
{preemptive = yes}
:Task-Z2
{priority = low}
{preemptive = yes}
:Idle
{priority = lowest}
{preemptive = yes}
tp = 100ms
tp = 100ms
tp = 100ms
n+1 RU
n+2 RU
n+3 RU
n+4 RU
RUNNING
WAIT-ING
WAIT-ING
RUN-NING
NO
NOT CREATED
READY
RUN-NING
RUN-NING
READY
RU
RUN-NING
WAITING
READY
READY
REA-DY
RUNNING
t1
t2
t3
t4
t5
t6
t7
t8
t9
t10
Time

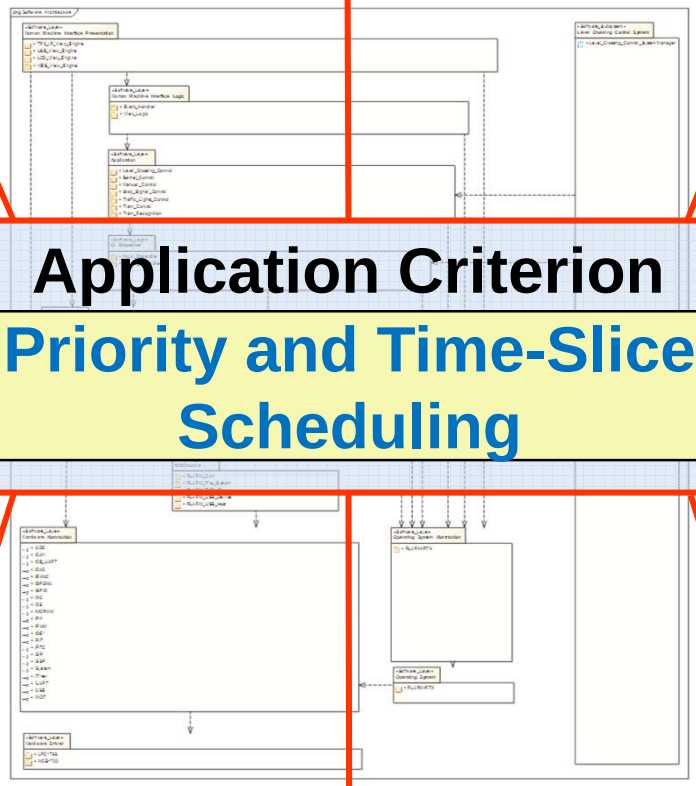
Hard real-time part
Time-triggered scheduling

Application can be
← split into →

Soft real-time part
Priority scheduling

…

…



Application Criterion

Priority and Time-Slice Scheduling

…

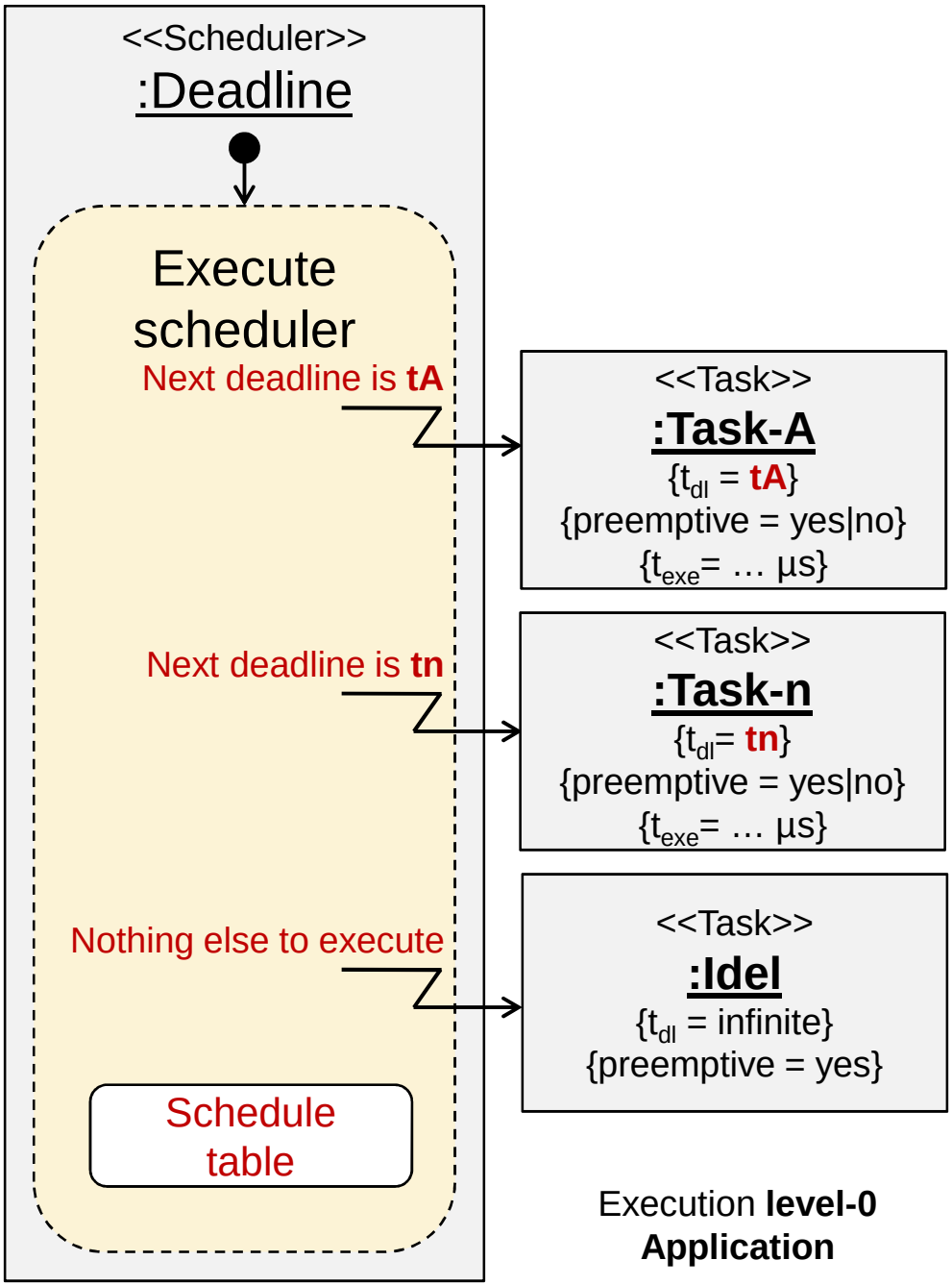
…

…

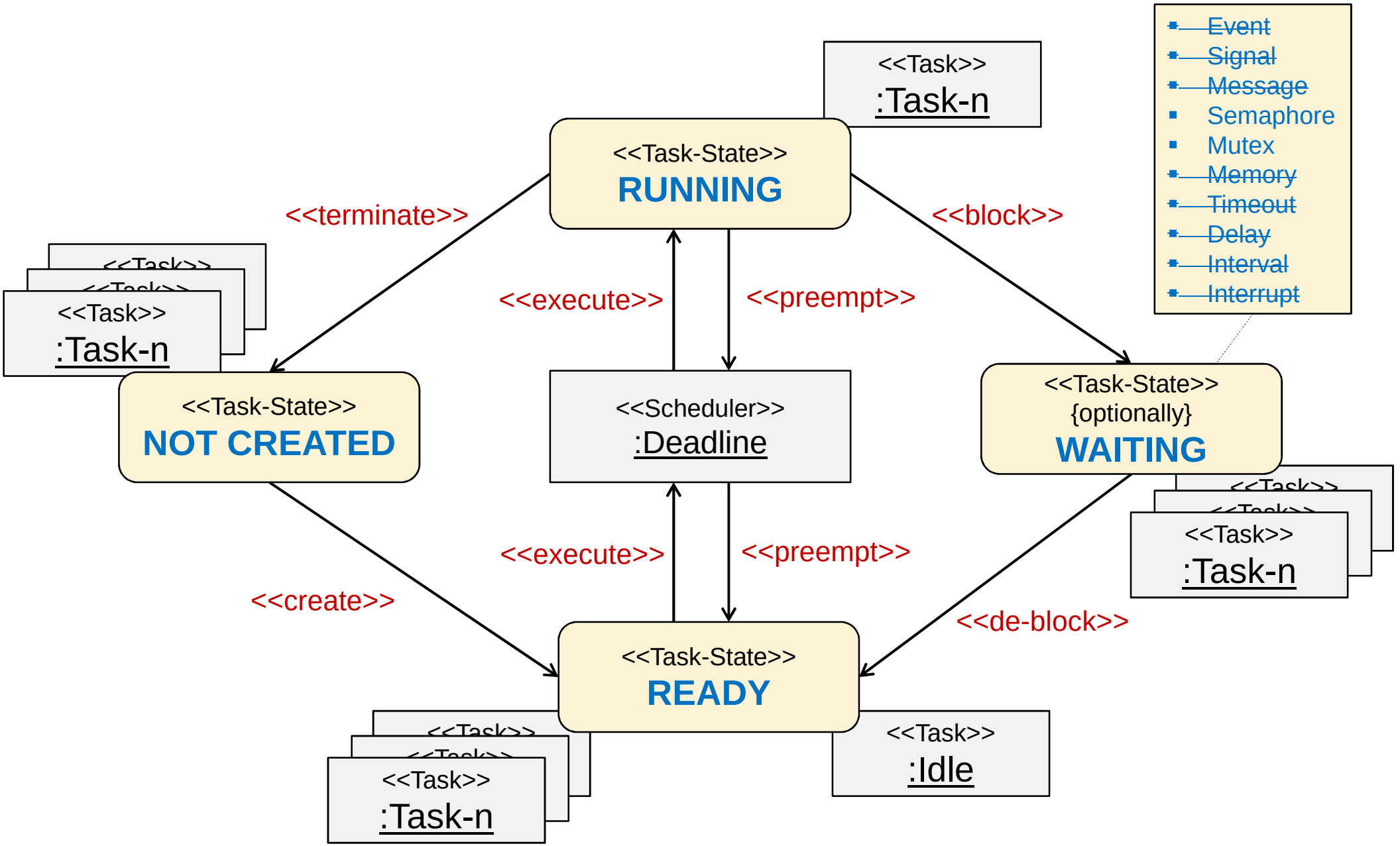
…

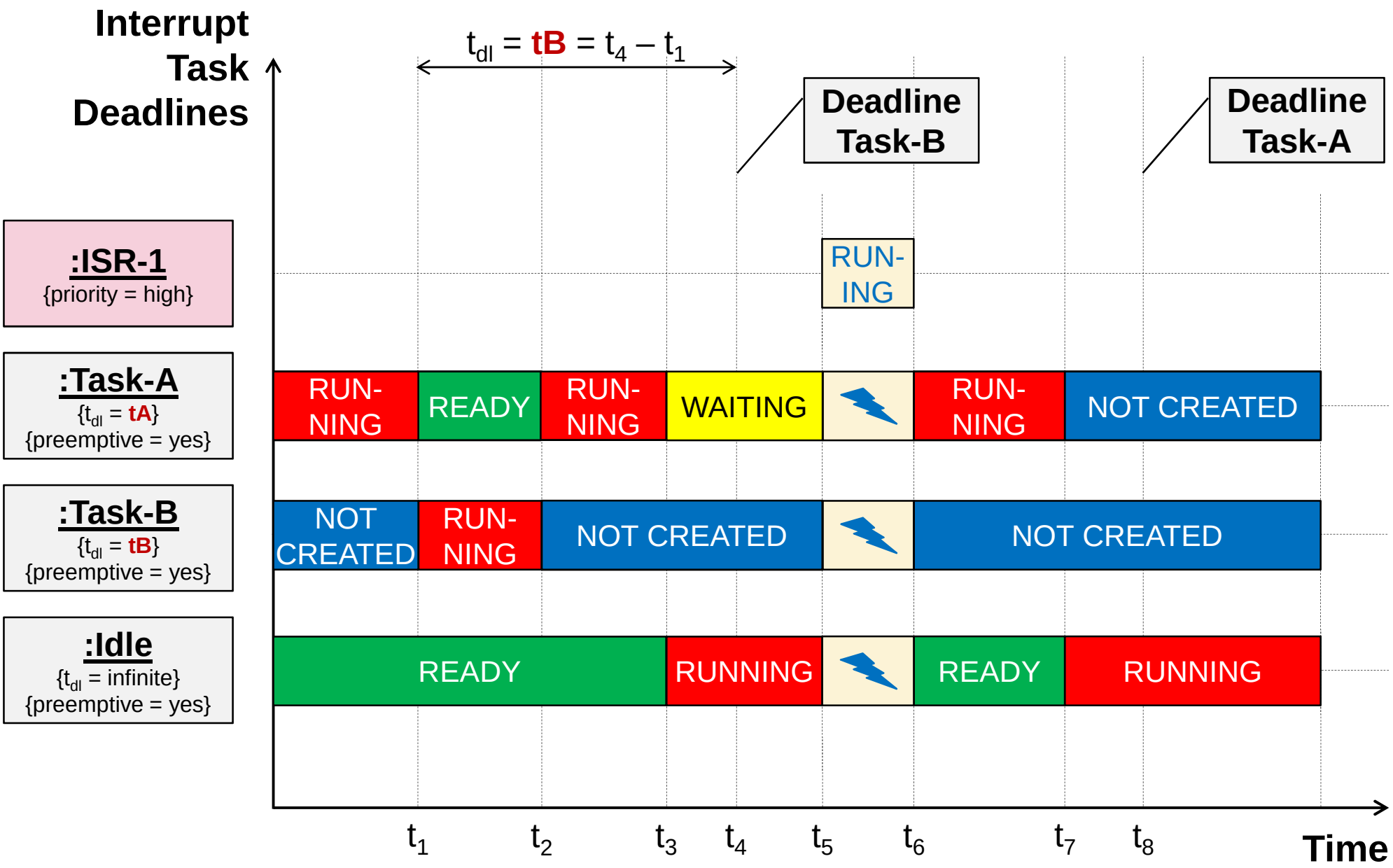
…

…



Execution **level-0 Application**


<<Task>>
:Task-n
<<Task-State>>
RUNNING
<<terminate>>
<<block>>
<<execute>>
<<preempt>>
Event
Signal
Message
Semaphore
Mutex
Memory
Timeout
Delay
Interval
Interrupt
<<Task>>
:Task-n
<<Task-State>>
NOT CREATED
<<Scheduler>>
:Deadline
<<Task-State>>
{optionally}
WAITING
<<Task>>
:Task-n
<<execute>>
<<preempt>>
<<create>>
<<de-block>>
<<Task-State>>
READY
<<Task>>
:Task-n
<<Task>>
:Idle

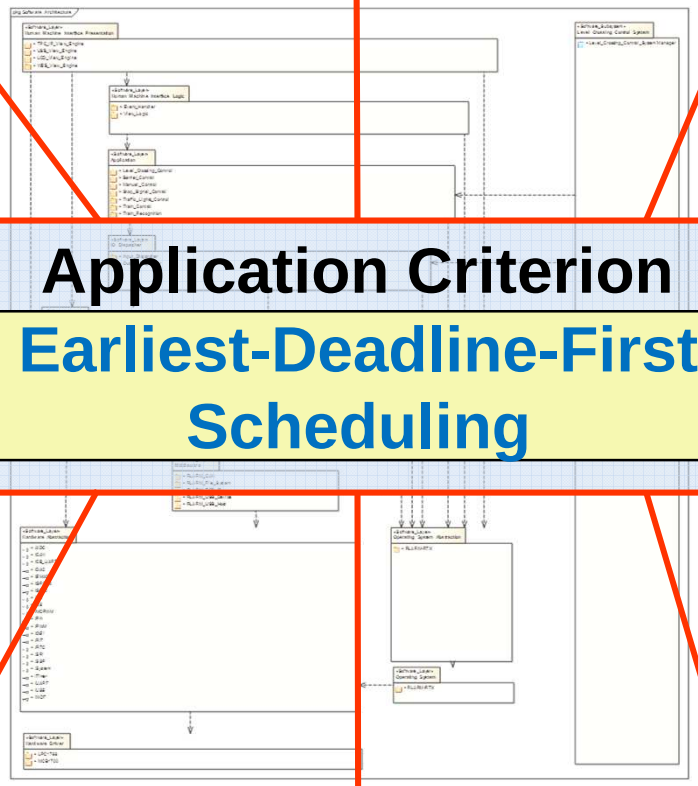

Interrupt Task Deadlines
t_{dl} = tB = t_4 – t_1
Deadline Task-B
Deadline Task-A
:ISR-1
{priority = high}
:Task-A
{t_{dl} = tA}
{preemptive = yes}
:Task-B
{t_{dl} = tB}
{preemptive = yes}
:Idle
{t_{dl} = infinite}
{preemptive = yes}
RUN-NING
RUN-ING
READY
WAITING
NOT CREATED
RUNNING
t_1
t_2
t_3
t_4
t_5
t_6
t_7
t_8
Time

Small
➔ fewer tasks

Deadline for each task is definable

Task deadlines have to be guaranteed

…



Tasks have fewer interactions

Application Criterion

Earliest-Deadline-First Scheduling

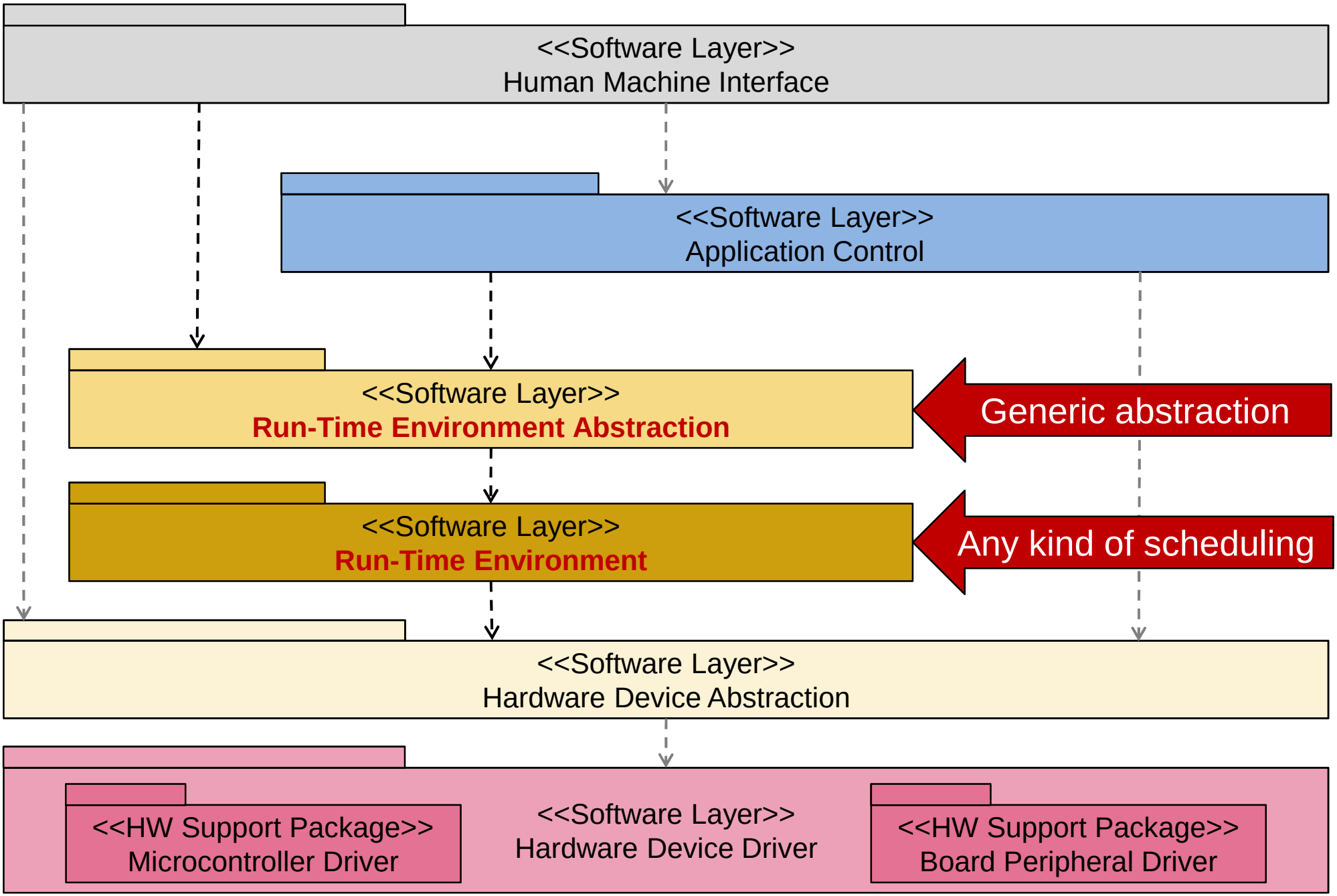
…

A scheduling method can be found to fulfill all task deadlines

…

…

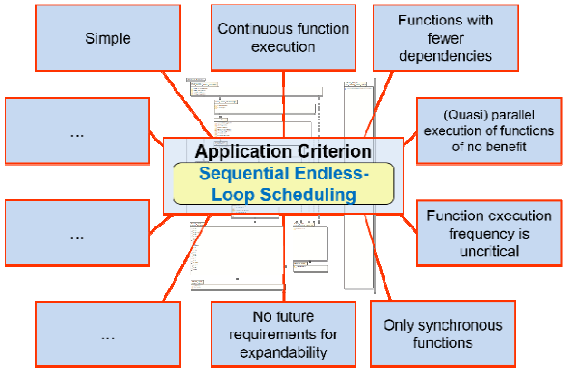
…

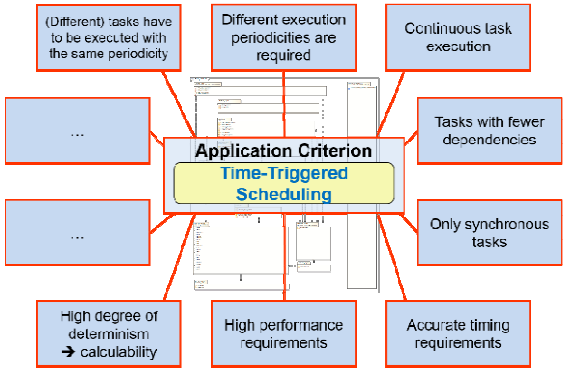

<<Software Layer>>
Human Machine Interface
<<Software Layer>>
Application Control
<<Software Layer>>
Run-Time Environment Abstraction
Generic abstraction
<<Software Layer>>
Run-Time Environment
Any kind of scheduling
<<Software Layer>>
Hardware Device Abstraction
<<HW Support Package>>
Microcontroller Driver
<<Software Layer>>
Hardware Device Driver
<<HW Support Package>>
Board Peripheral Driver

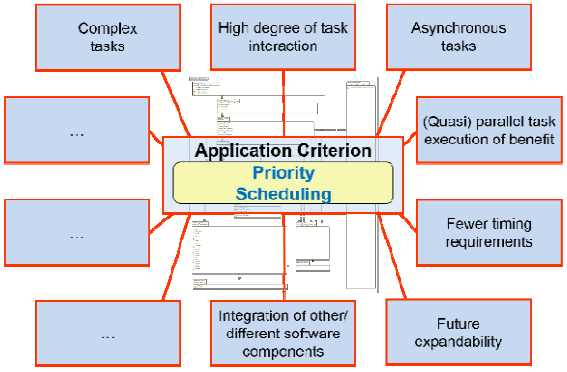
Betriebssystem: mit oder ohne?

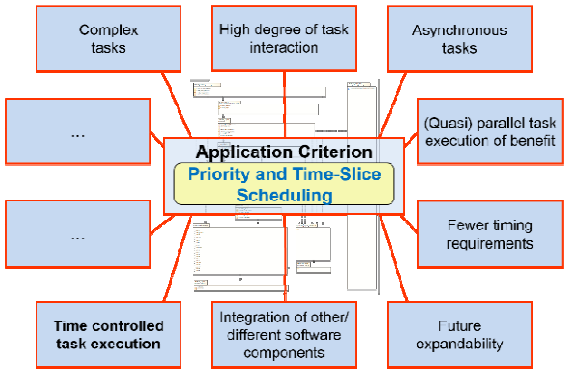
- Jede Embedded-Software benötigt eine Laufzeitarchitektur!
- Jede Laufzeitarchitektur enthält eine Art von „Betriebssystem“!
- Die zentrale und genauere Frage lautet:

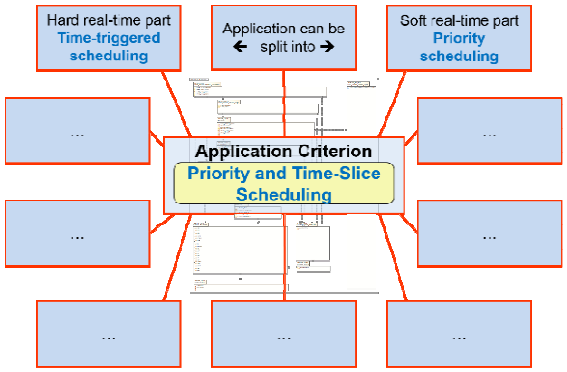
Welche Art von Laufzeitarchitektur ist für meine Applikation die geeignetste?

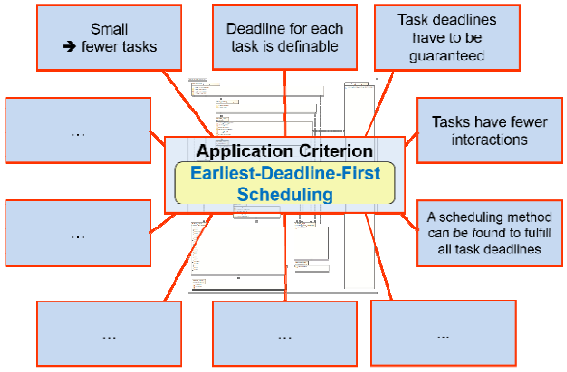












MICRO CONSULT