CCSL:
The Clock Constraint Specification Language

Frédéric **MALLET**
AOSTE Team-Project
Université **N**ice **S**ophia Antipolis
I3S Laboratory – UMR 7271 CNRS
INRIA Sophia Antipolis Méditerranée

Outline

- **Logical Time** as/at design time
  - → The **C**lock **C**onstraint **S**pecification **L**anguage

- **CCSL** usages
  - A syntax to specify time semantics explicitly and formally
  - A language to express timed requirements

Logical Time is

- an explicit design artifact
- **functional**
  - Should be made a first-class citizen
  - Not just a mere annotation introduced only for performance/time analysis
- **partially ordered** and progressively **refined**
  - The relative rates are **often** more important than the actual durations
  - possibly **approximate**
- **abstract** and **multiform**

→ The physical (real) time is just a special case => chronometric

A model to capture (logical) time properties

Design time

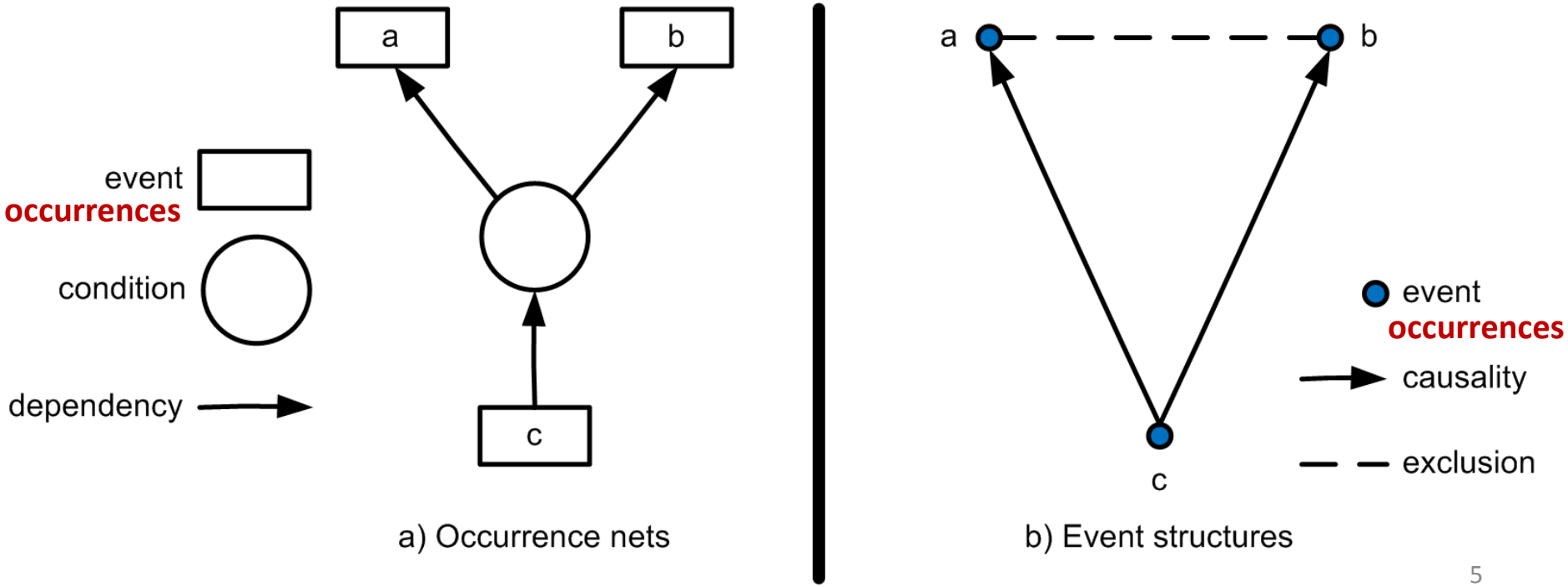
- Untimed relationships
  - Causality (control and data flows)
  - Exclusion
- Timed relationships
  - Before
  - After
  - At the same time (simultaneously)

Untimed relationships

- From occurrence nets / event structures
  - Causality / dependency
  - Exclusion / Inhibit

{}, {c}, {c,a}, {c,b}

~~{a}~~, ~~{b}~~, ~~{a,b}~~, ~~{a,b,c}~~



Causality

Causality (untimed)

- a **causes** b => b cannot occur unless a occurs

becomes

❑ Temporal **sequentiality**

- a **precedes** b => a always occurs before b

or

❑ **Simultaneity**

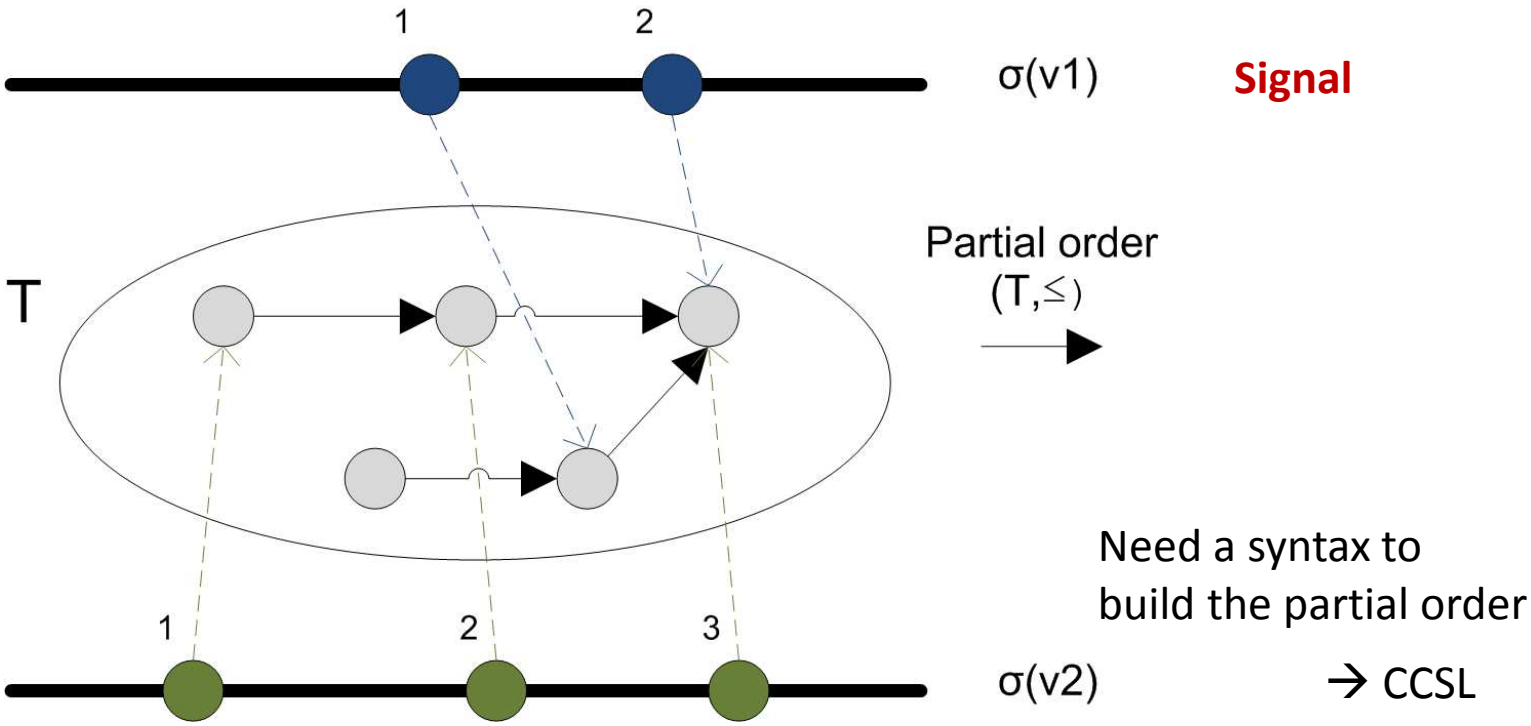
- a **causes** b **and**
- a is **coincident** with b

Exclusion

- Untimed **exclusion**
  - a **exclusiveWith** b => either a or b can occur
- Timed exclusion
  - a **#** b => a and b cannot occur simultaneously
- Asymmetry => **priority**

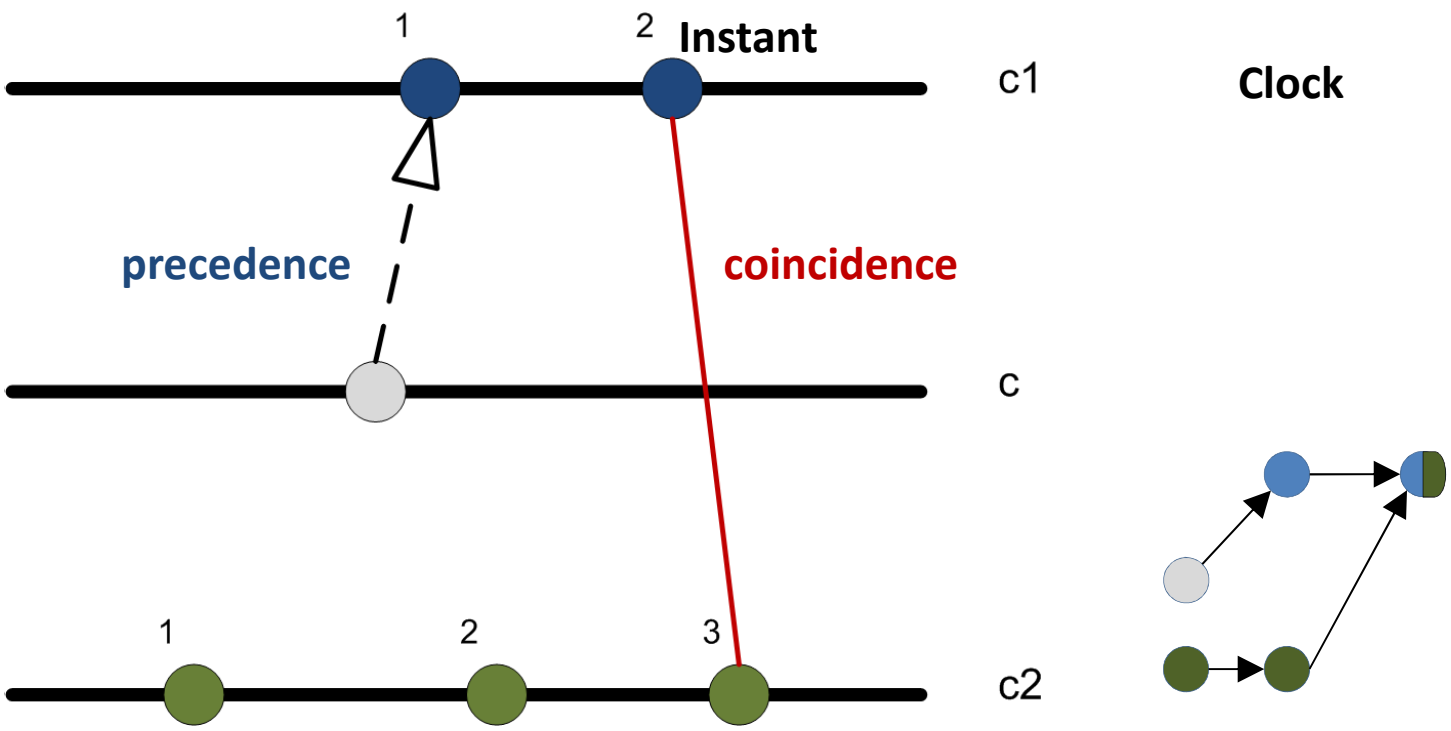
Inspiration: Tag structures

- Support for encoding several (timed or untimed) Models of Computation



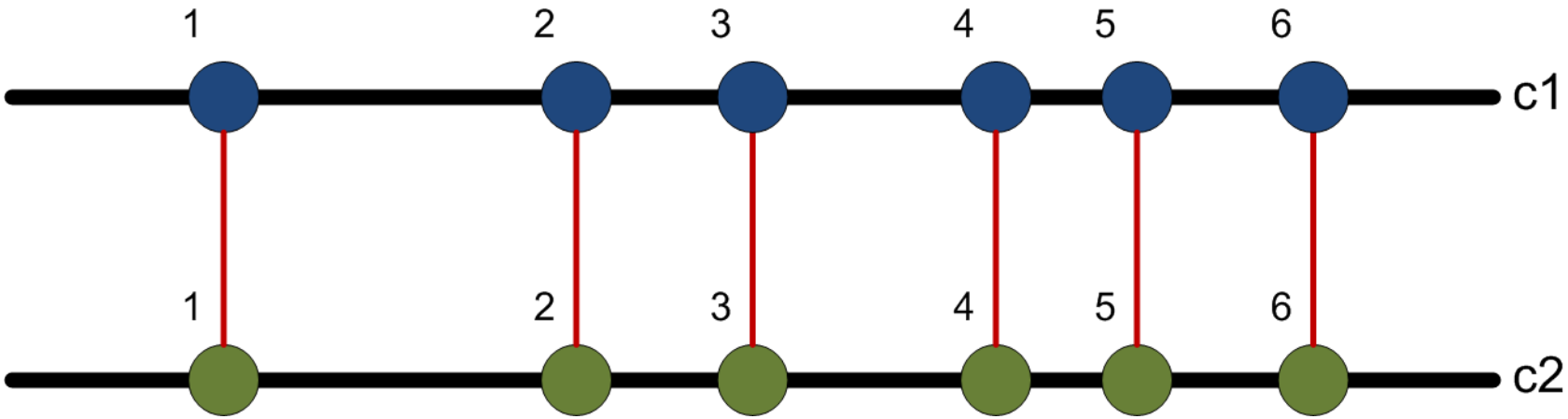
Logical clocks – instant relations

- Clocks are *a priori* independent



Clock relations – Coincidence-based

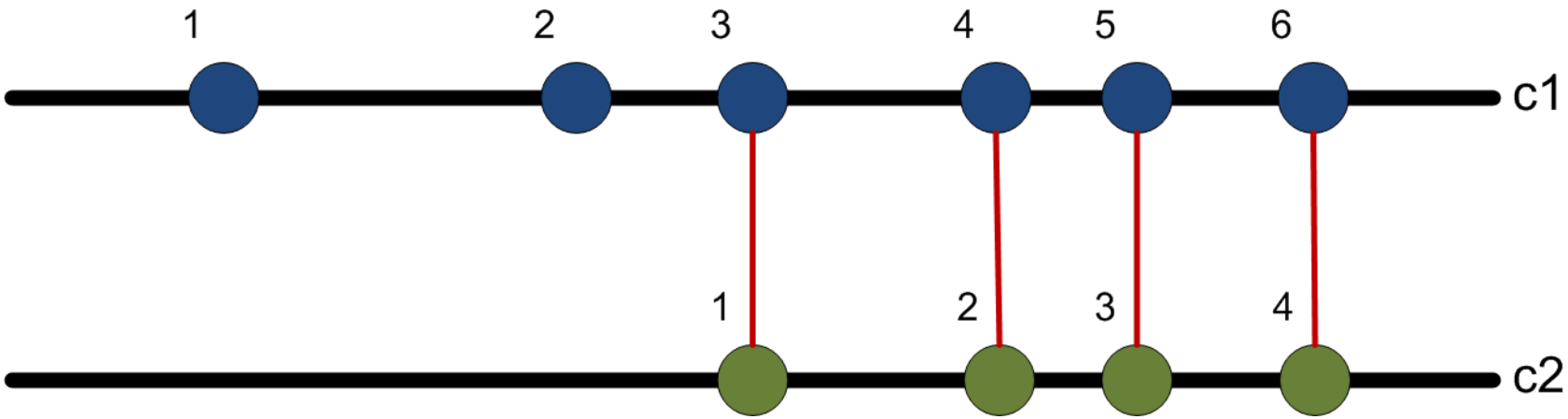
- Infinitely many coincidence relations



c2 = c1

Clock relations – Coincidence-based

- Infinitely many coincidence relations

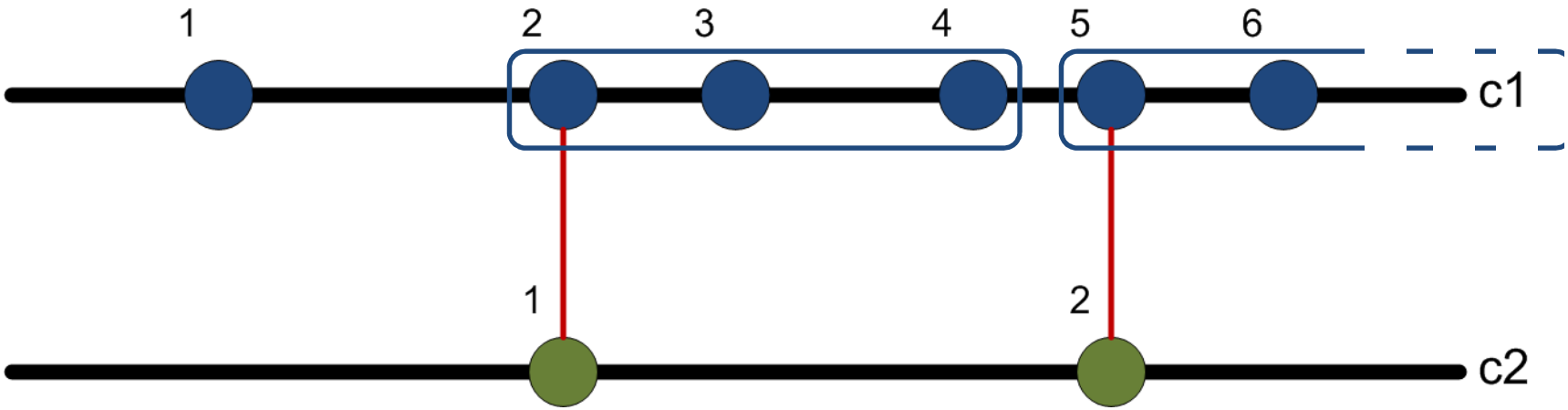


c2 ⍇ c1 $ 2

c2 is a **subclock** of c1

Clock relations – Coincidence-based

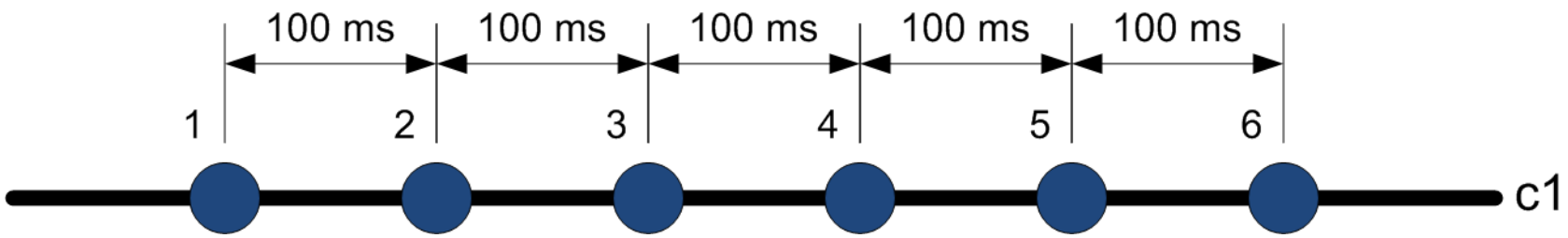
- Infinitely many coincidence relations



c2 isPeriodicOn c1 period=3 offset=1

Clock relations – Coincidence-based

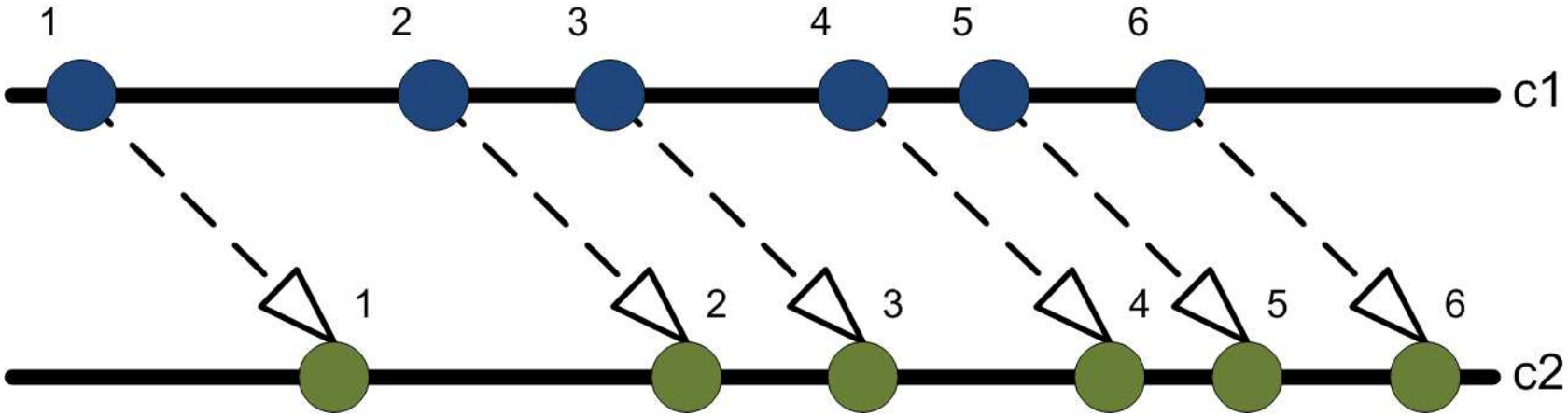
- Infinitely many coincidence relations



c2 ⩵ idealClk discretizedBy 0.1

Clock relations – Precedence-based

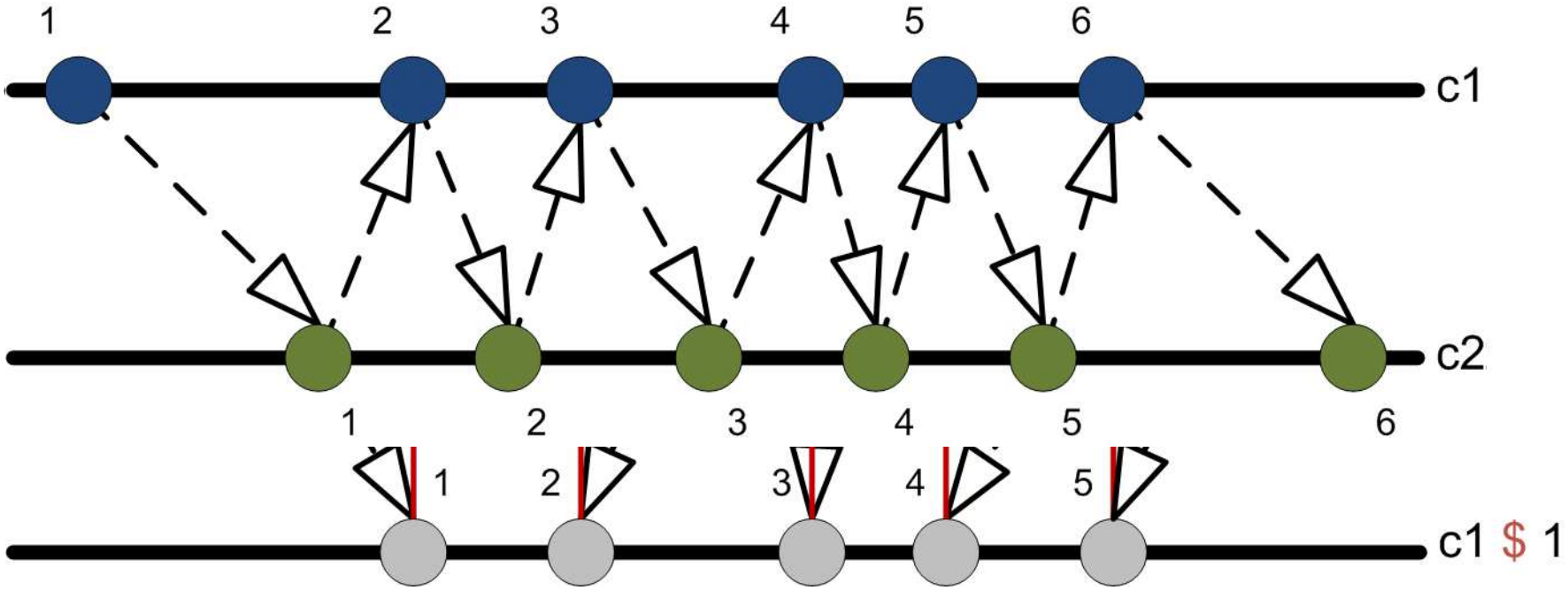
- Infinitely many precedence relations



c1 ⊴ c2

Clock relations – Precedence-based

- Can be bounded if required (for analysis)

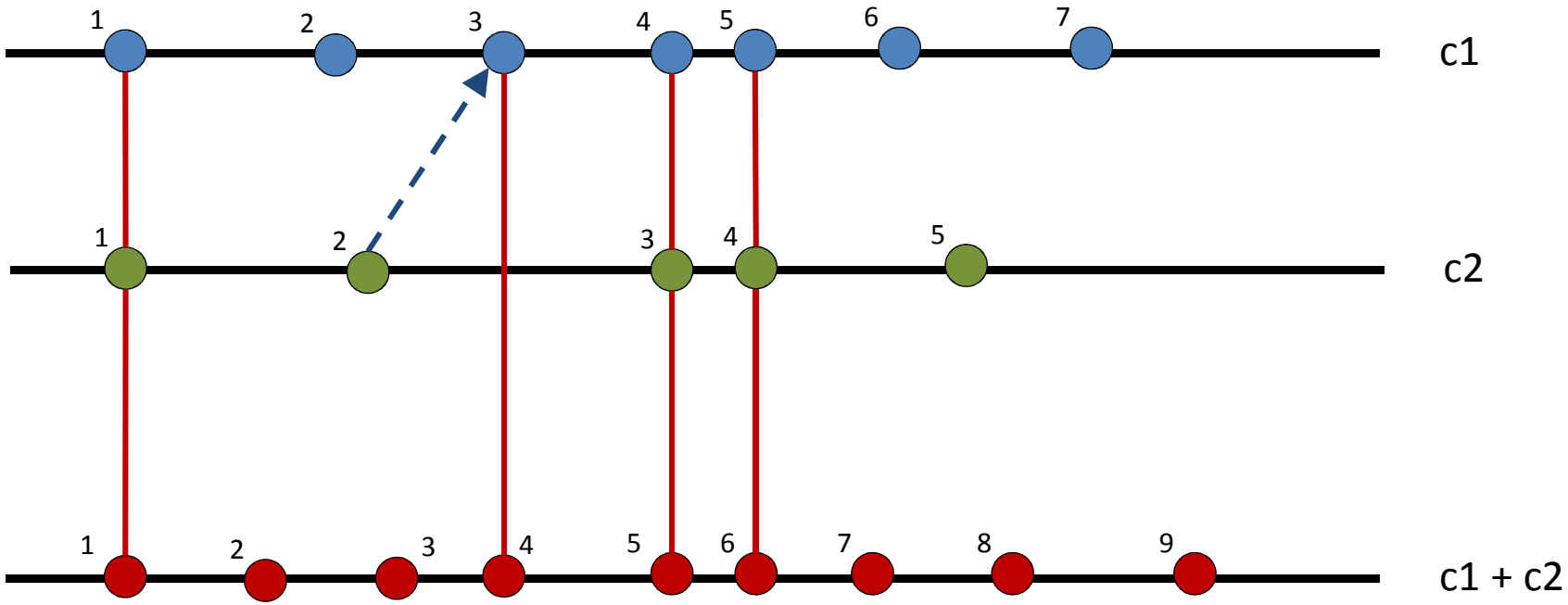


c1 ⋋ c2 c1 $ 3 → c1 ~ c2

Kernel + Libraries

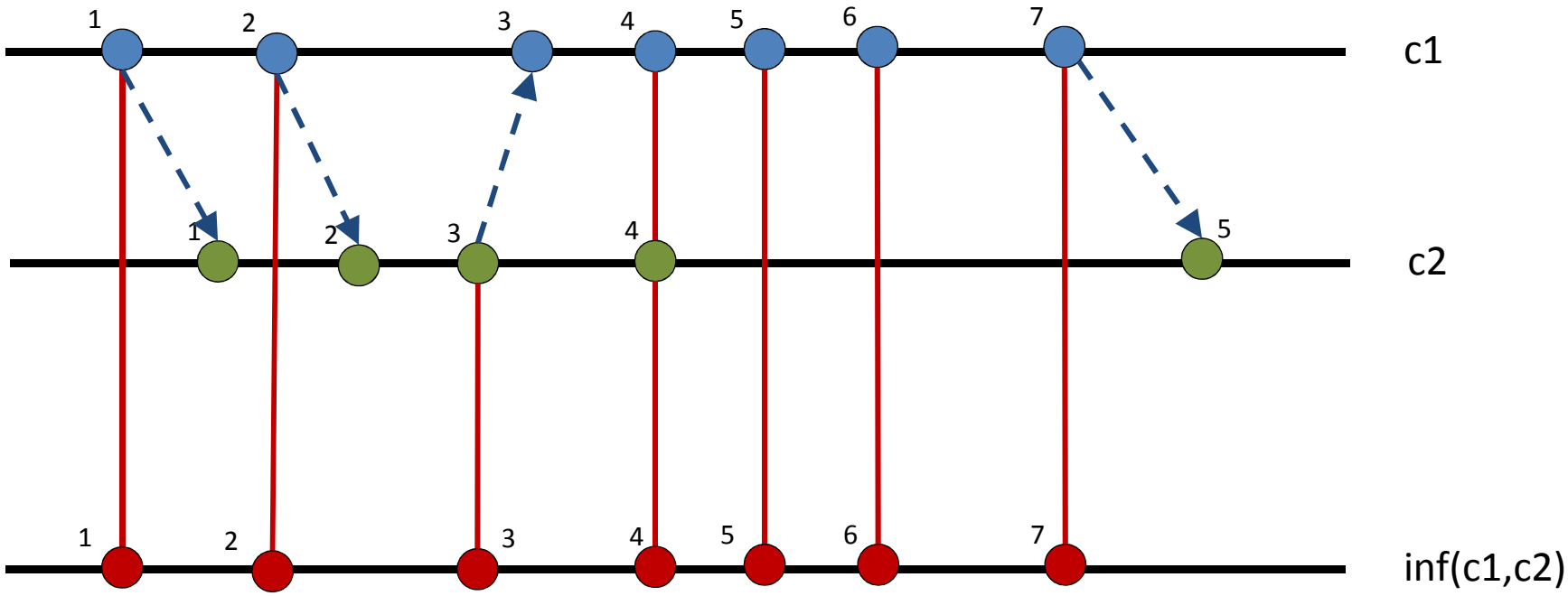
Clock expressions - Union

- Several solutions
  - Also intersection



Clock expressions - Inf

- Slowest clock faster than both c1 and c2 (GLB)
  - Also Sup (Least Upper Bound)



Clock relations - summary

- Elementary relations
  - Coincidence, precedence
  - Mixed relations (sampling, delay)
- Combined to build common time patterns
  - Periodicity |a[i+1]-a[i]| = period
  - Sporadicity |a[i+1]-a[i]| > interArrival
  - Deadline |end[i]-start[i]| < deadline
  - Jitter, skew, …

→The Clock Constraint Specification Language

Outline

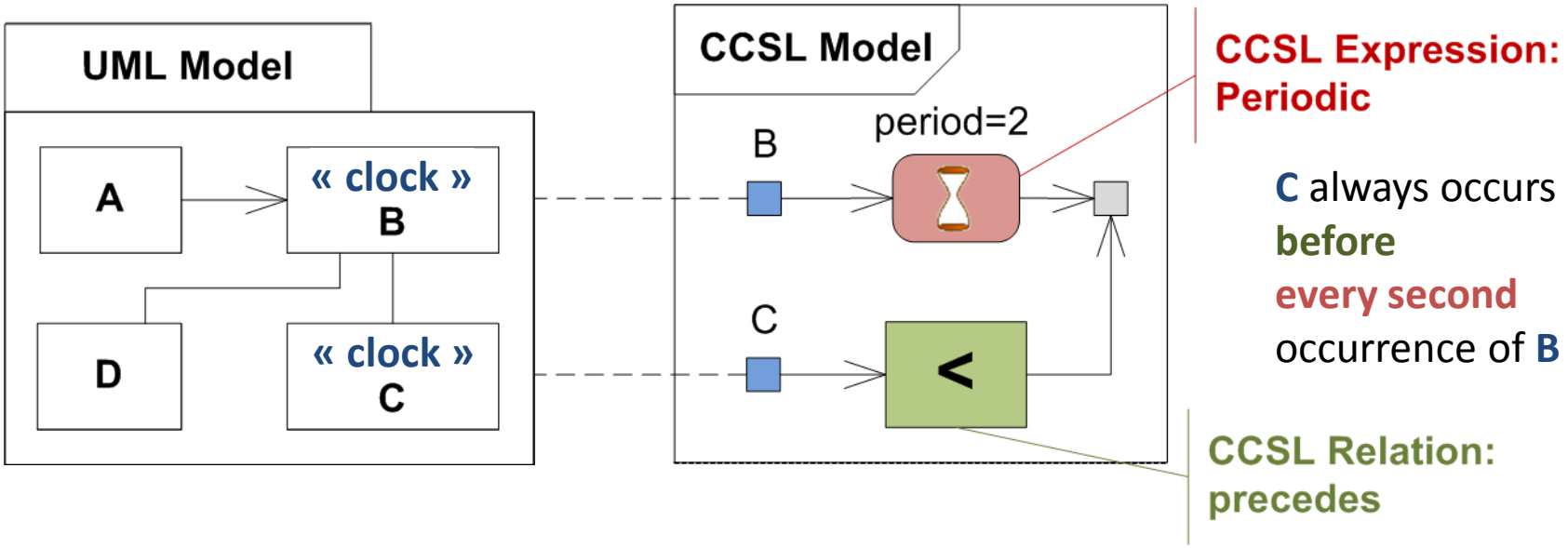
- **Logical Time** as/at design time
  - → The Clock Constraint Specification Language
- **Model-Driven Engineering**: OMG UML MARTE
  - → The Time Model
- **CCSL** usages
  - A syntax to specify timed semantics explicitly and formally
  - A language to express timed requirements

MARTE – Time Model

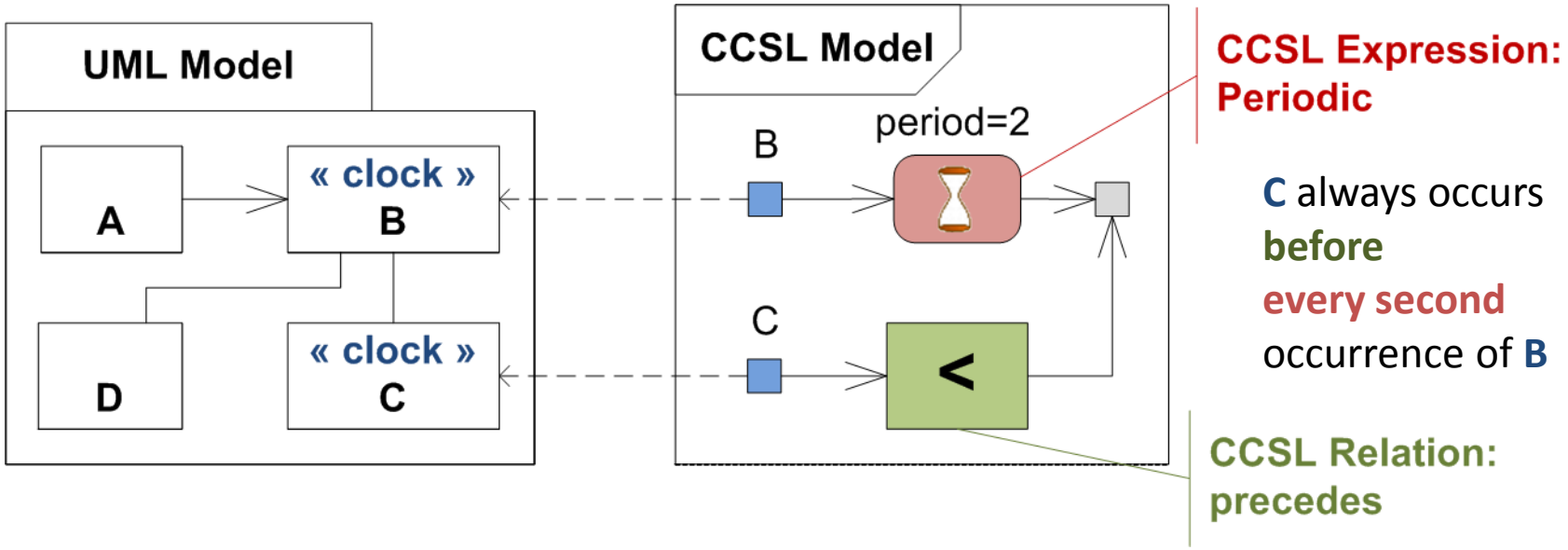
- Annotate UML models to identify logical clocks
  - Events
  - Behavior (start, finish)
  - Messages (send, receive), Signals
- Specify constraints with CCSL
  - To specify the causal / timed constraints
- Rely on logical clocks
  - Reuse existing diagrams
  - Orthogonal notion of time

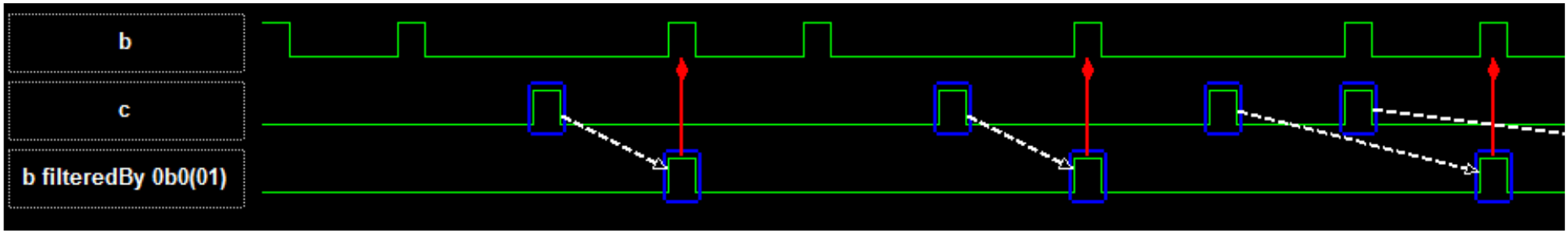
Annotate UML models

Identify and constrain clocks



Annotate UML models
for simulation and animation





http://timesquare.inria.fr/

TimeSquare

Outline

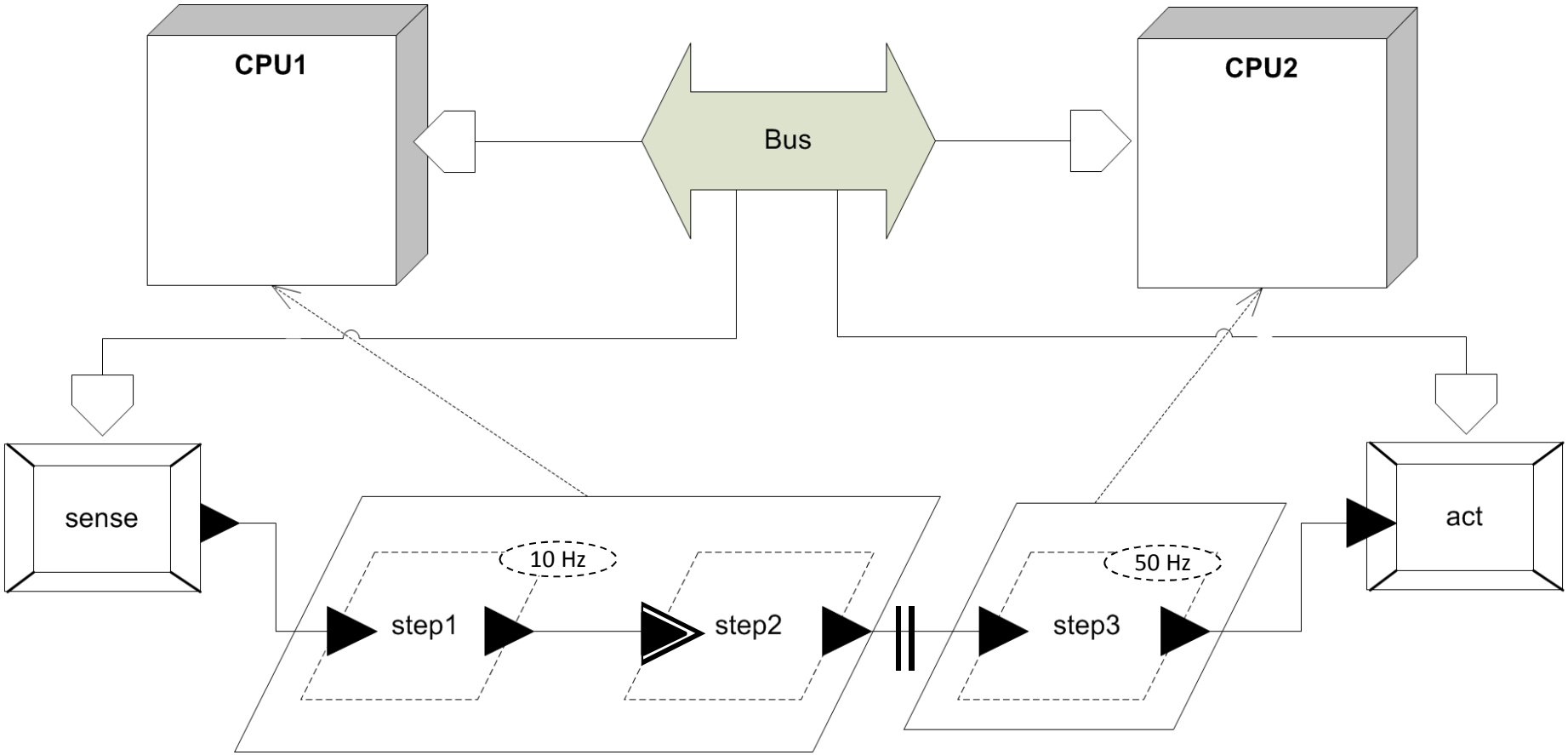
- **Logical Time** as/at design time
  - → The **C**lock **C**onstraint **S**pecification **L**anguage
- **M**odel-**D**riven **E**ngineering: OMG UML MARTE
  - → The Time Model
- **CCSL** usages
  - A syntax to specify timed semantics explicitly and formally
  - A language to express timed requirements

Examples

- Synchronous and polychronous languages
- Process networks
  - Self-timed => fully timed by scheduling/optimization

- Avionics: AADL – SAE Standard
- Automotive: East-ADL – dedicated to AutoSAR
- EDA/SoC Design: IP-Xact – Accelera consortium

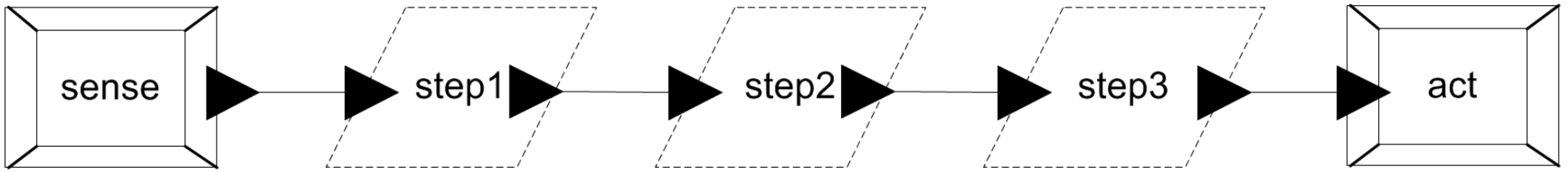
An example pattern in AADL (with binding representation)



CCSL: Time Refinement

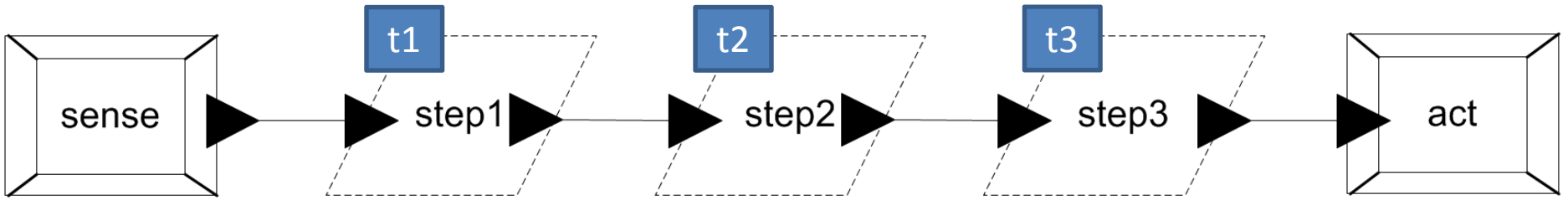
- Application
  - `sense` **`causes`** `step1`
  - `step1` **`causes`** `step2`
  - `step2` **`causes`** `step3`
  - `step3` **`causes`** `act`

[asynchronous]



CCSL: Time Refinement

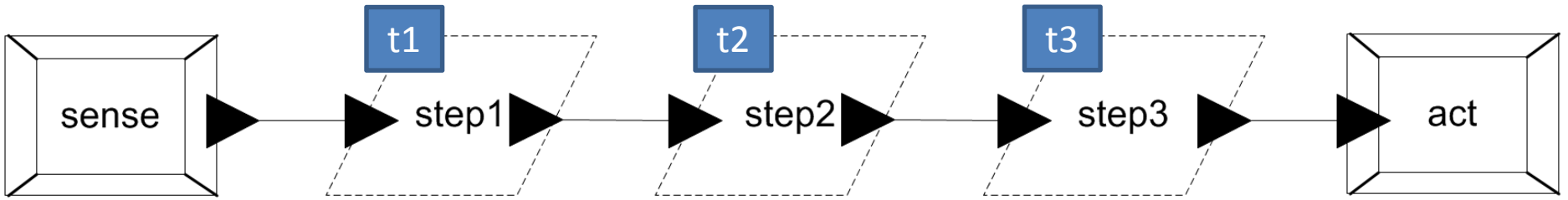
- Execution Platform: 3 threads
  - t3 is twice slower than t1 **[synchronous]**
    - `t3 isPeriodicOn t1 period=2`
    - `t1 causes step1`
    - `t3 causes step3`
  - Communications
    - `t1 precedes t2` **[temporality]**



CCSL: Time Refinement

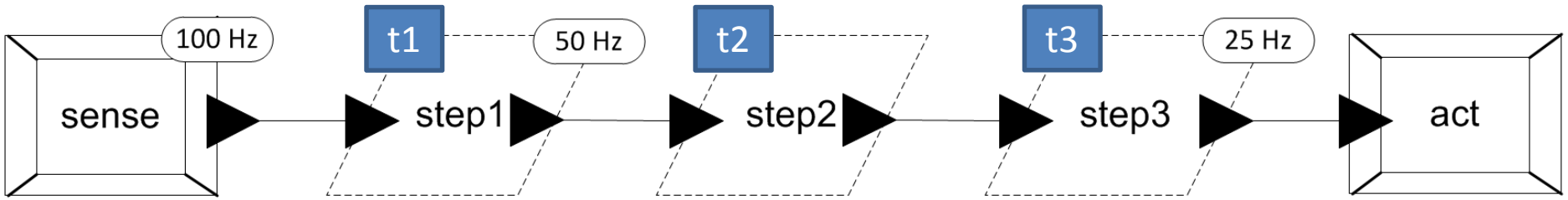
- Execution Platform: 3 threads
  - t3 is twice slower than t1 **[synchronous]**
    - `t3 isPeriodicOn t1 period=2`
    - `t1 causes step1`
    - `t3 causes step3`
  - Communications
    - `t1 5-precedes t2` **[temporality]**

[bounded]

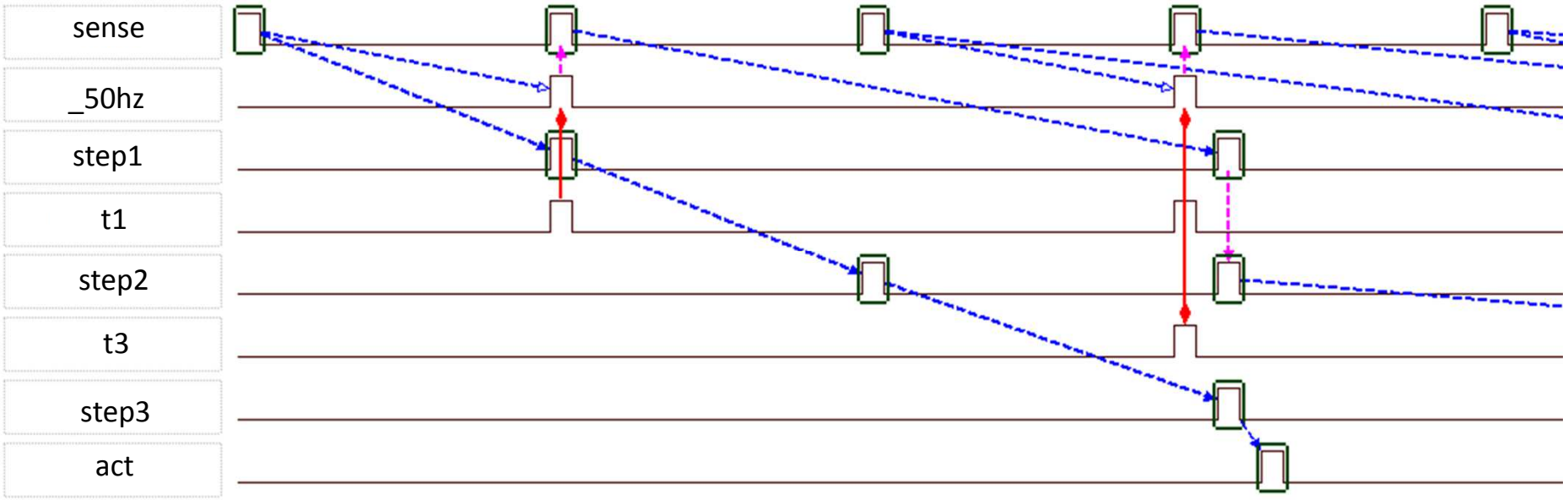


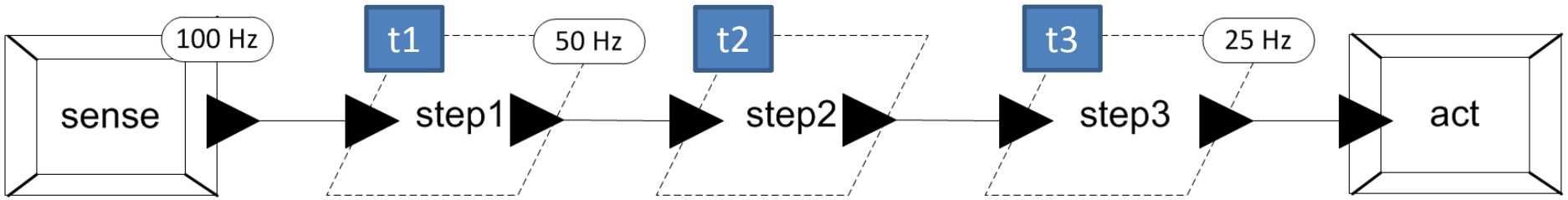
CCSL: Time Refinement

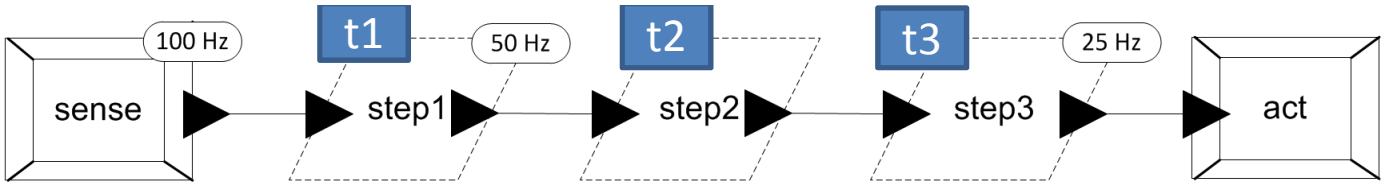
- Physical time
  - Periodic sensor **[synchronous]**
    - `sense is _100Hz`
  - Periodic task
    - `t1 is sense SampledOn _50Hz`



CCSL: Time Refinement





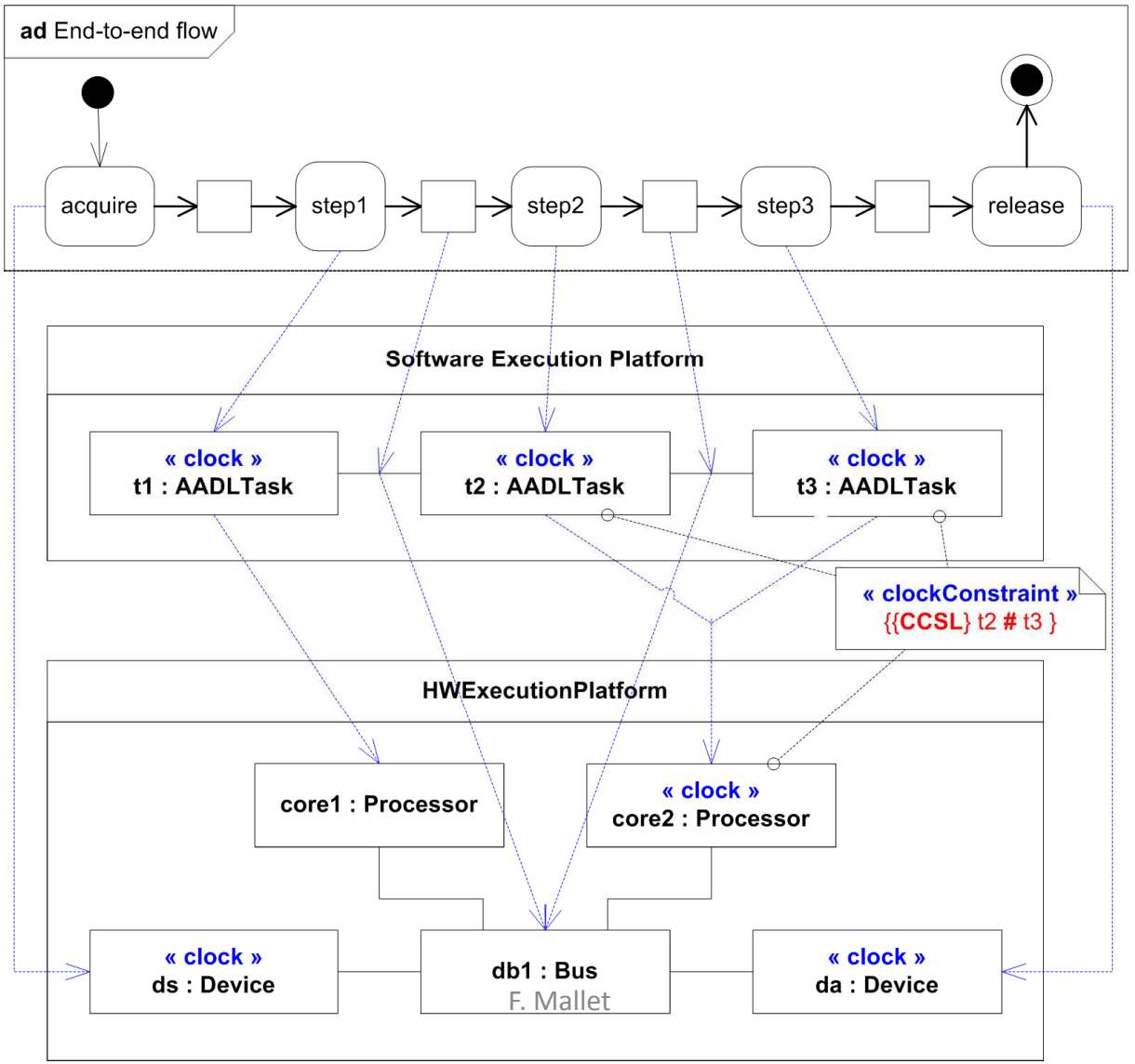


Example

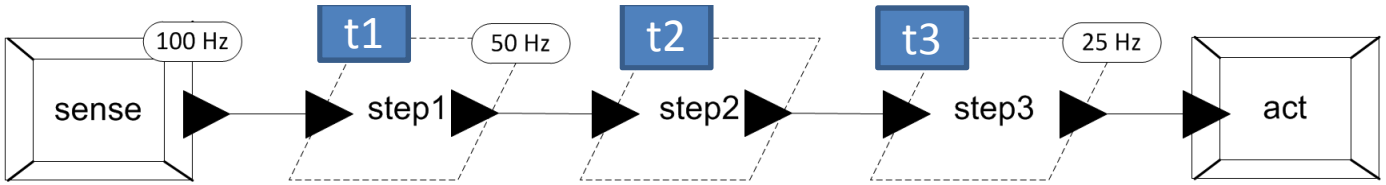
Application

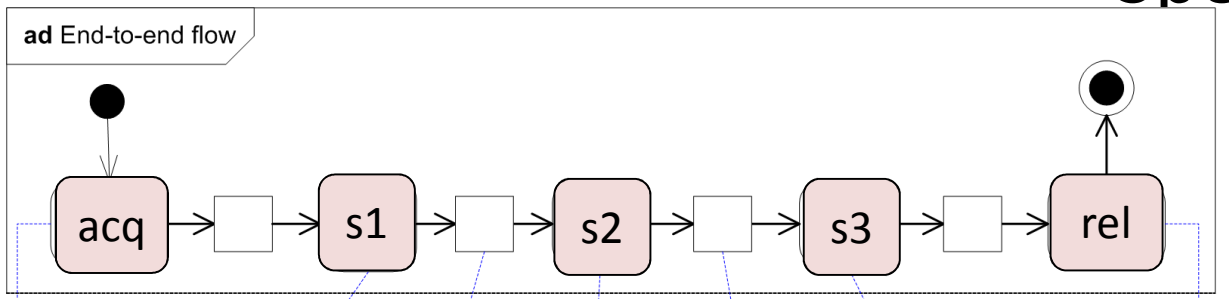
Software execution platform

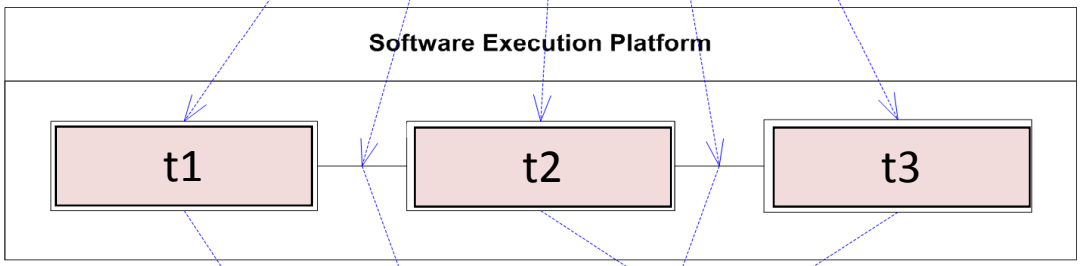
Hardware execution platform

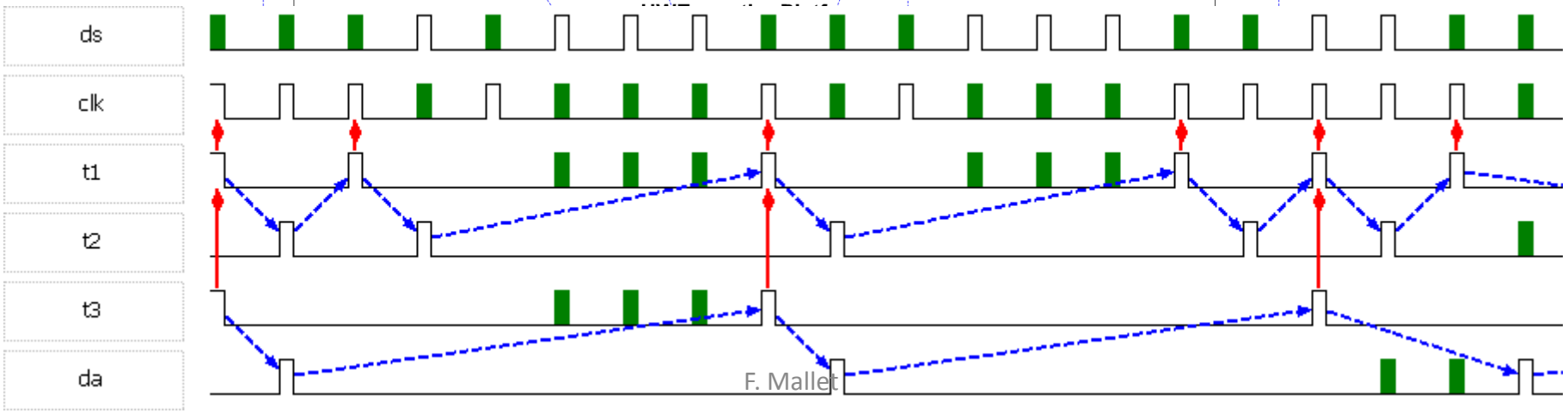


Executable Specification





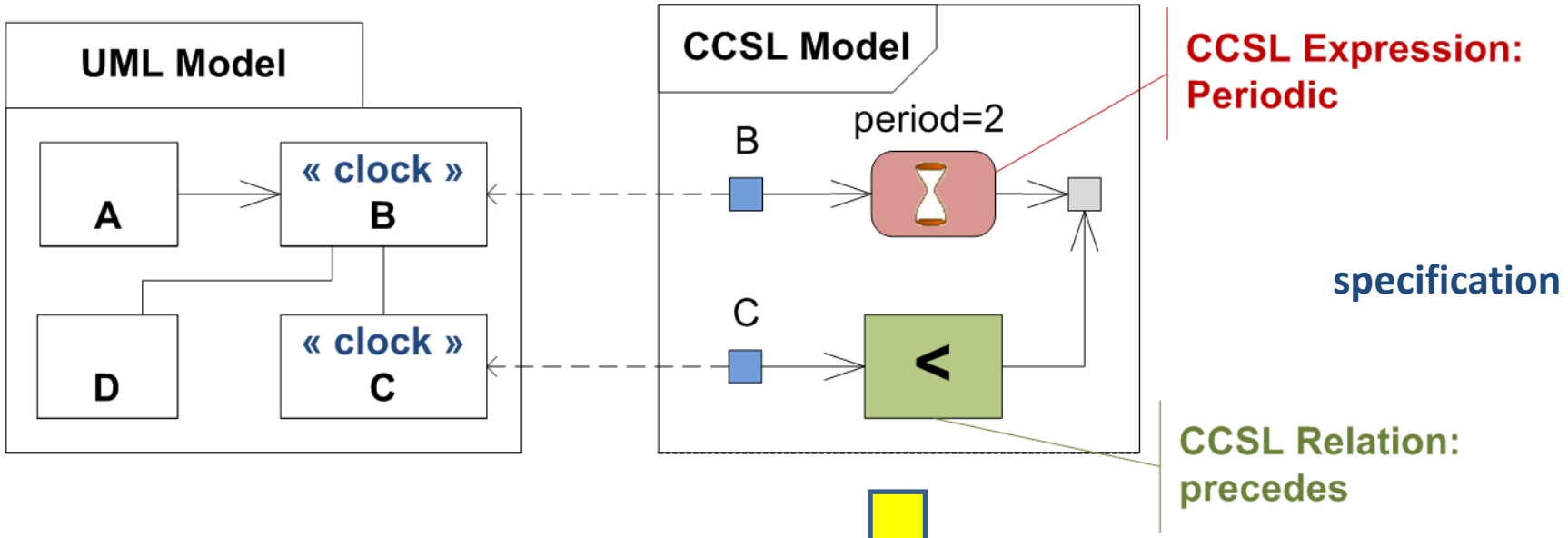


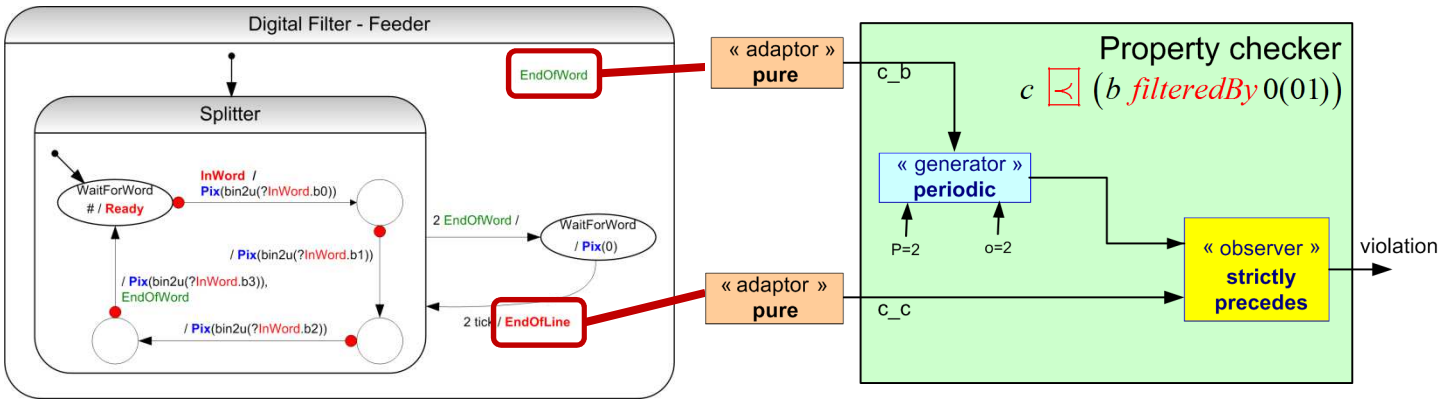


Outline

- **Logical Time** as/at design time
  - → The **C**lock **C**onstraint **S**pecification **L**anguage
- **M**odel-**D**riven **E**ngineering: OMG UML MARTE
  - → The Time Model
- **CCSL** usages
  - A syntax to specify time semantics explicitly and formally
  - A language to express timed requirements

Adorns UML models
for requirements

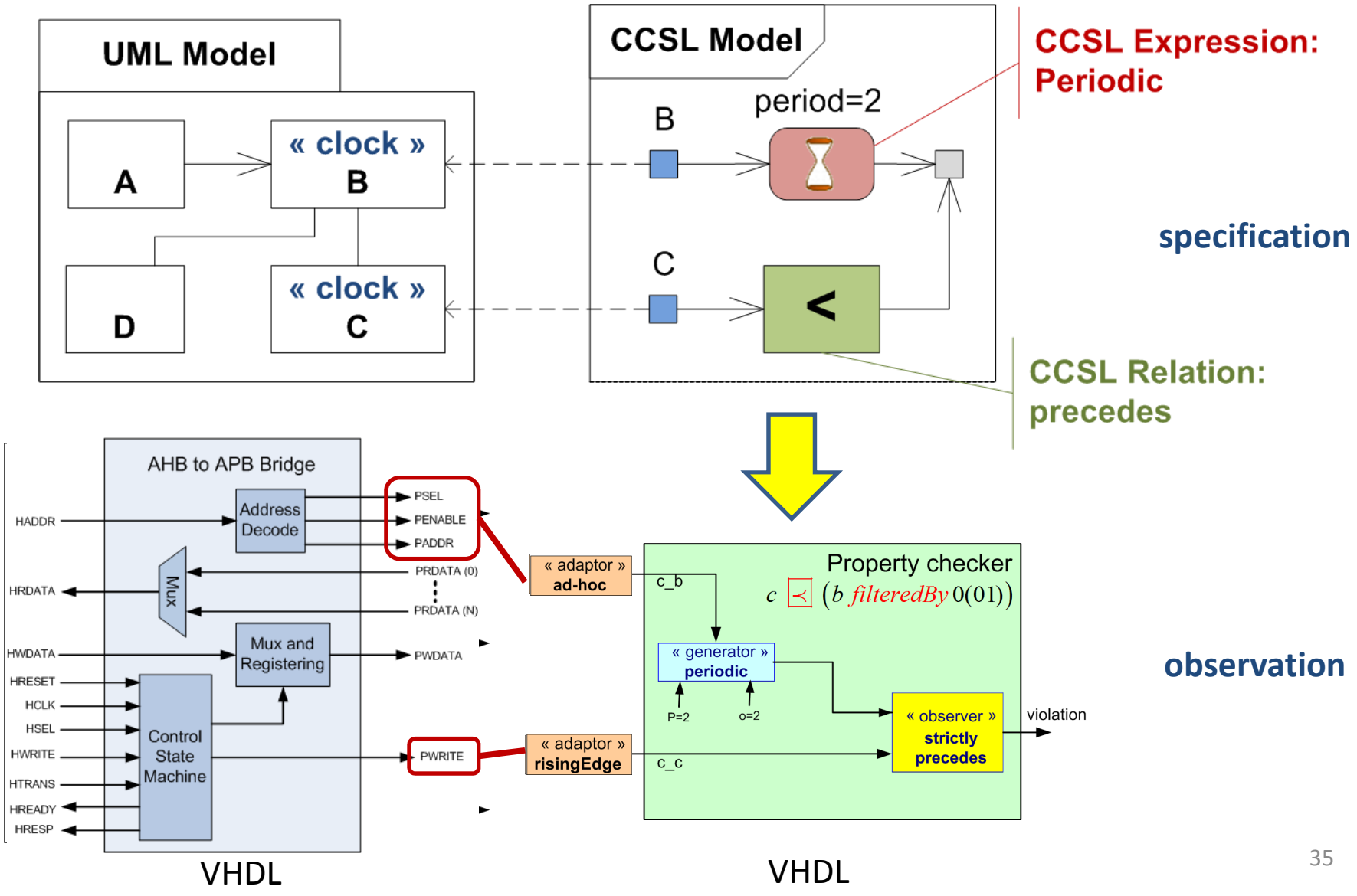




Esterel/SyncCharts

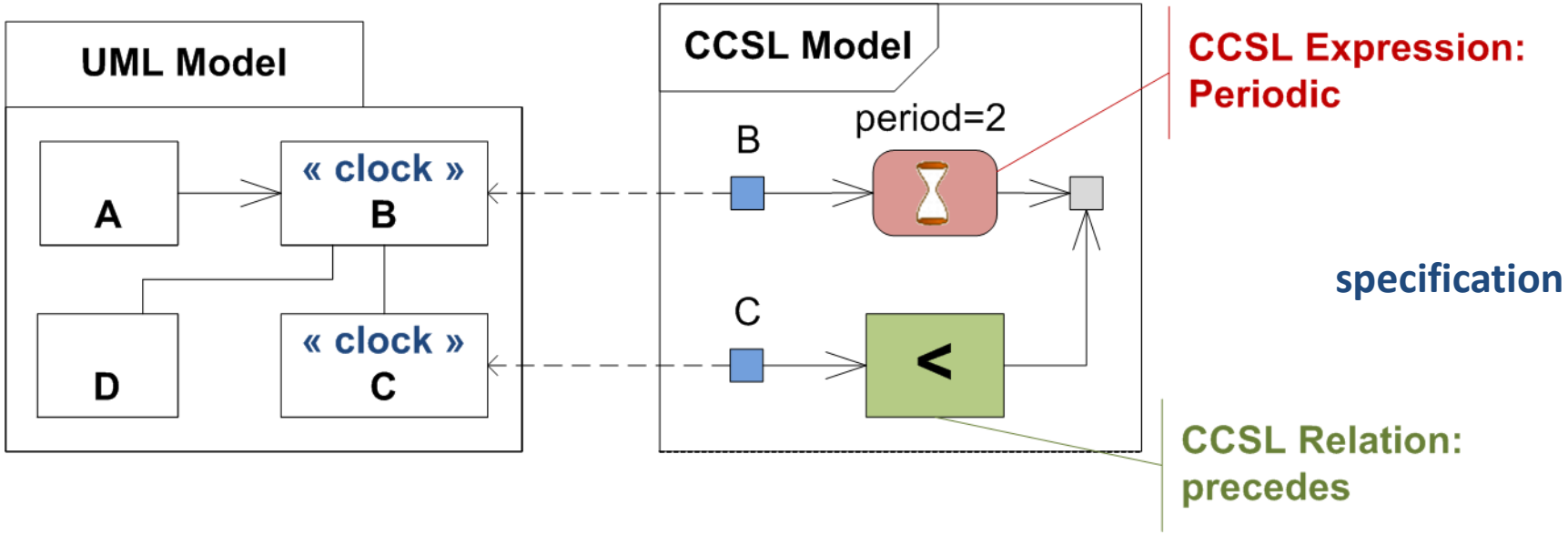
Esterel/SyncCharts

Adorns UML models
for requirements



Safety properties

Verification of CCSL specifications



- Check Properties on CCSL specifications
  - Linear Temporal Logics (LTL)
  - First Results (Yin Ling's PhD, Sep. 2010)
    - Modular Transformation of CCSL into Promela
  - ***P***roperty ***S***pecification **L**anguage (Régis Gascon)
    - Transformation of CCSL operators into Büchi automata

THANK YOU !