software verification & validation

securityandtrust.lu

Model-Driven Trace Checking of Temporal Properties

Domenico Bianculli

SnT Centre - University of Luxembourg

UNIVERSITÉ DU LUXEMBOURG

Project Background

joint work with

Lionel
Briand

Wei
Dou

Project Partner

CTIE

LE GOUVERNEMENT
DU GRAND-DUCHÉ DE LUXEMBOURG
Centre des Technologies de l'Information de l'Etat

Centre des Technologies de l'Information de l'Etat

Software development at CTIE

- **Advanced software development process**
- **Unified development process across different systems**
- **Model-driven engineering (MDE) is used as basis for development**

eGovernment Services

developed as

Business Processes

Goals of the Project

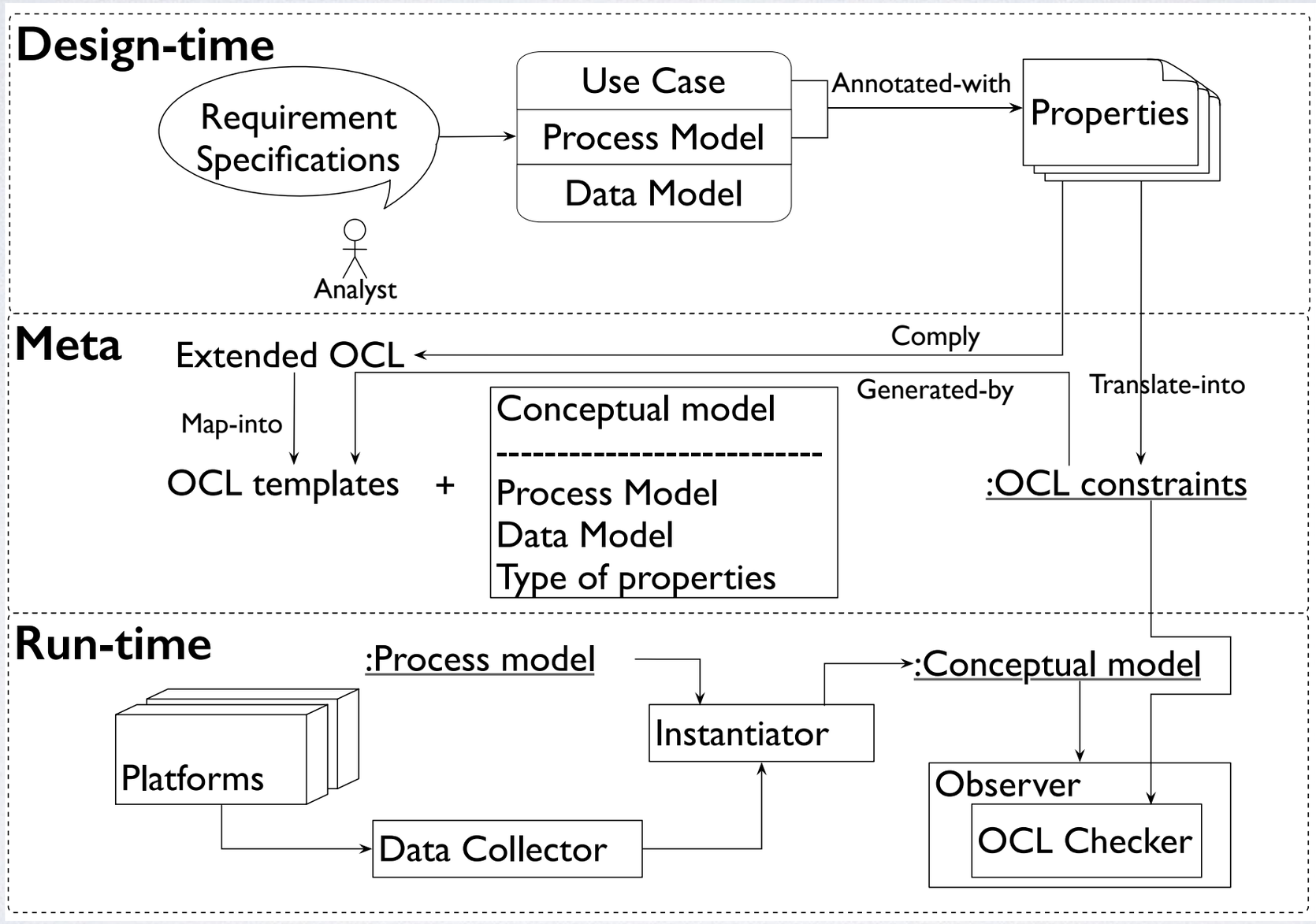
- to extend Prometa to support the specification of requirements to be verified at run time

- to enable run-time verification during the execution of eGovernment applications developed with Prometa

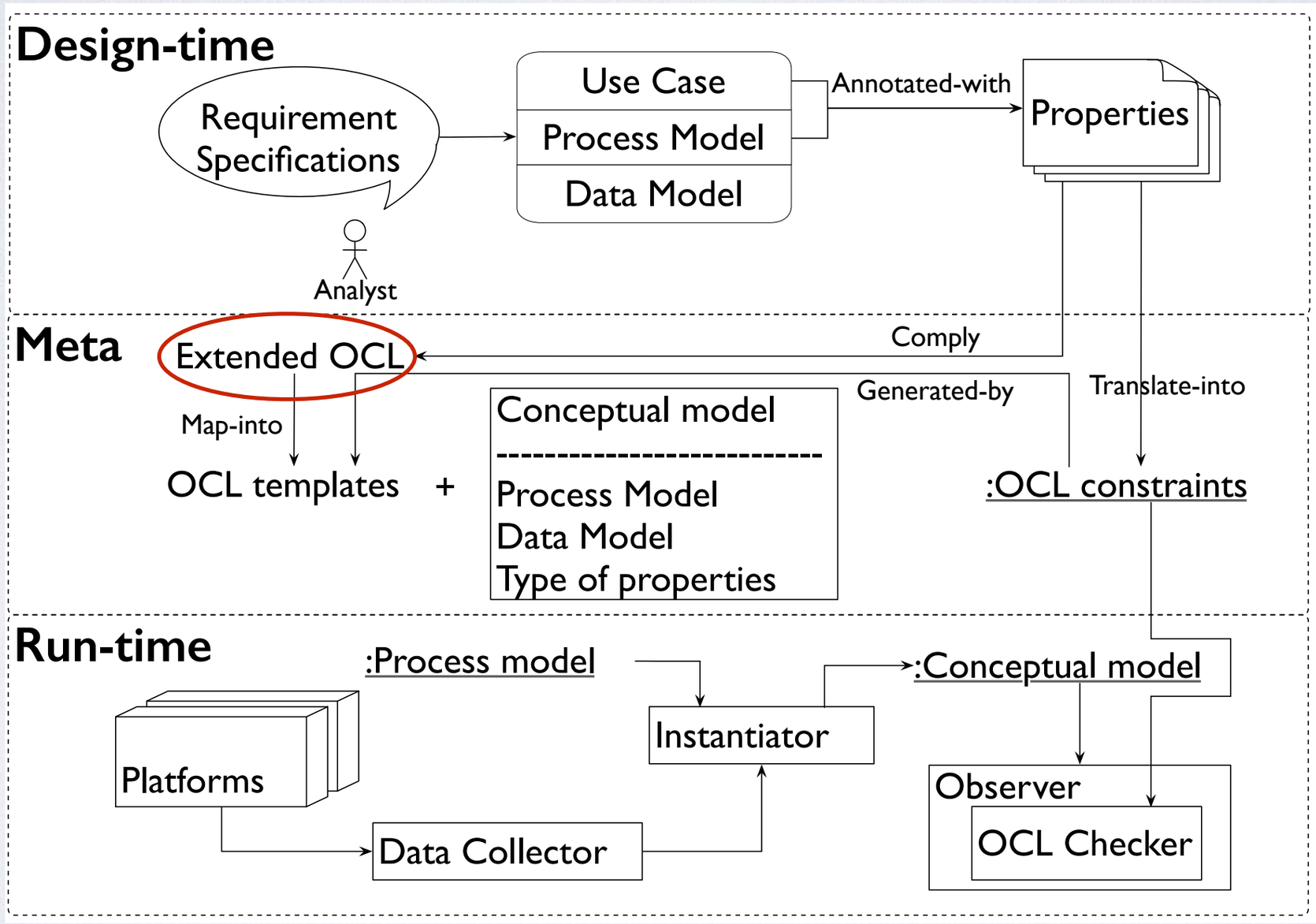
Requirements

- solution adoptable in contexts where MDE is already a development practice
- high-level domain-specific language to allow business analysts to express the properties to be checked
- solution based on OMG standards and stable MDE technology
- performance of the solution comparable to the state of the art

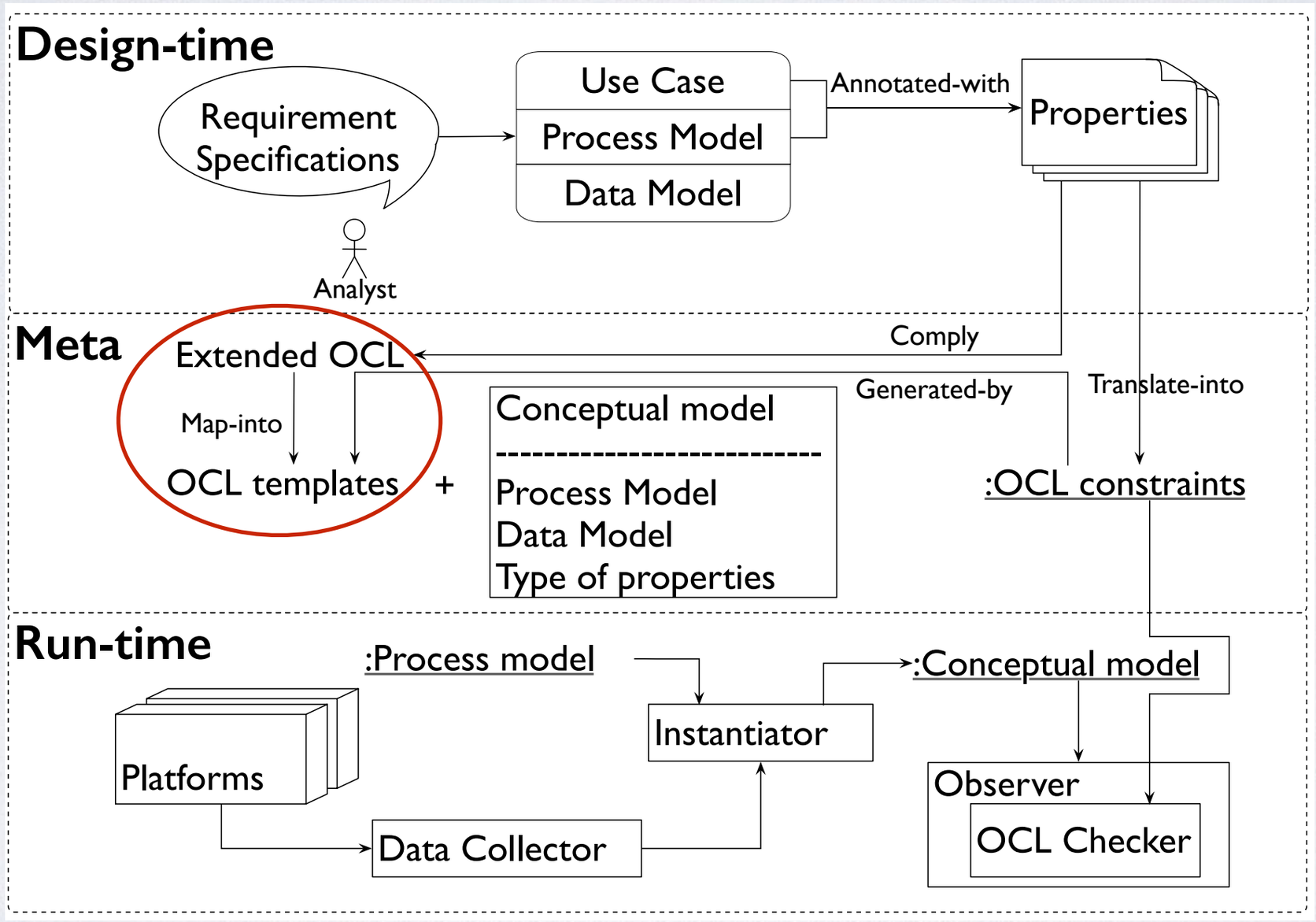
Our Vision


Design-time
Requirement Specifications
Analyst
Use Case
Process Model
Data Model
Annotated-with
Properties
Meta
Extended OCL
Comply
Map-into
OCL templates
+
Conceptual model
Process Model
Data Model
Type of properties
Generated-by
Translate-into
:OCL constraints
Run-time
:Process model
Platforms
Data Collector
Instantiator
:Conceptual model
Observer
OCL Checker

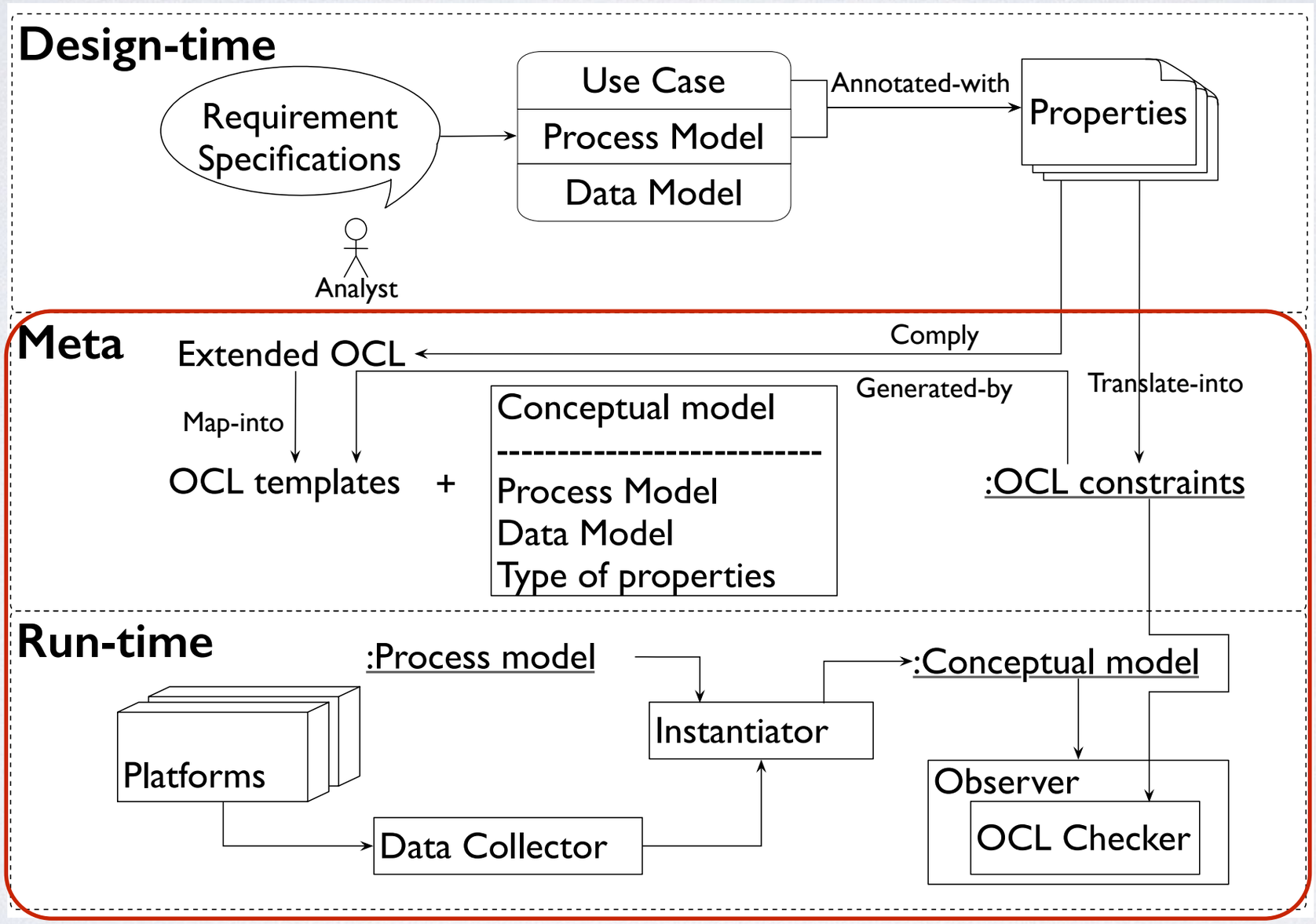
Requirements Specification Language



Property Checking



Integration with Run-time Platforms

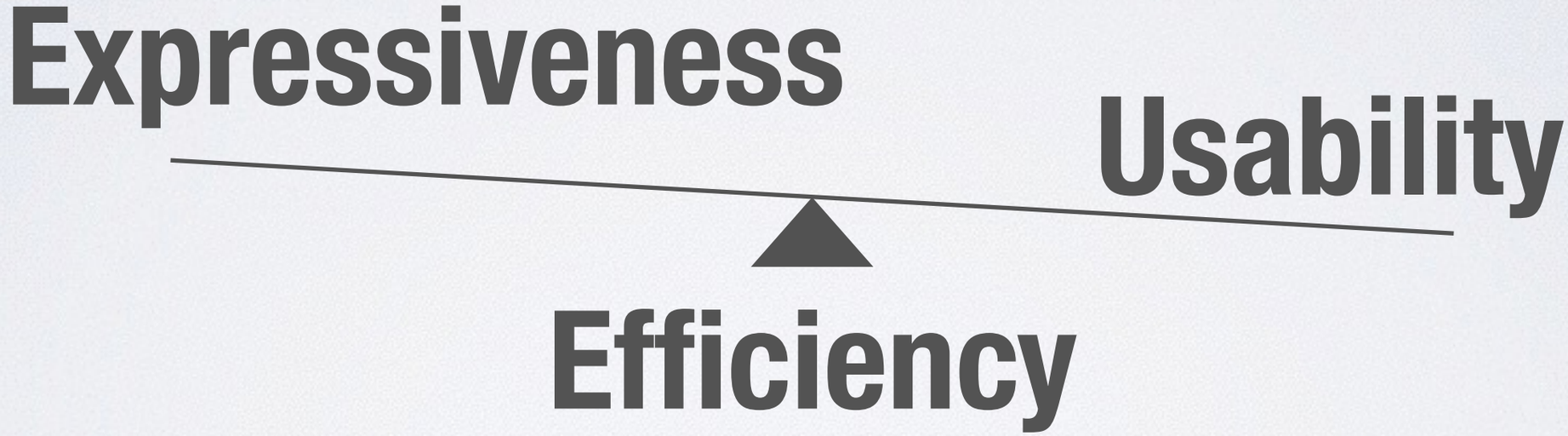


Requirements Specification Language

Requirements Specification Language

Temporal Properties

Requirements Specification Language



Dwyer's Pattern System

Scope

Pattern

Dwyer's Pattern System

Scope

- Globally
- Before
- After
- Between-and
- After-until

Dwyer's Pattern System

Pattern { Universality, Existence, Absence, Precedence, Response

Specific Occurrence of an Event

“A password reset email will be sent to the user after the third incorrect login attempt”

Time Distance from a Boundary

"A speaker should be ready **within 10 minutes after the second** keynote."

OCLR

OCL for Runtime verification

(published at ECMFA 2014)

OCLR Syntax Excerpt

$$
\begin{array}{lll}
\langle \mathit{TemporalExp} \rangle & ::= & \langle \mathit{Scope} \rangle\ \langle \mathit{Pattern} \rangle \\
\langle \mathit{Scope} \rangle & ::= & \text{`}\texttt{globally}\text{'} \\
 & | & \text{`}\texttt{before}\text{'}\ \langle \mathit{Boundary1} \rangle \\
 & | & \text{`}\texttt{after}\text{'}\ \langle \mathit{Boundary1} \rangle \\
 & | & \text{`}\texttt{between}\text{'}\ \langle \mathit{Boundary2} \rangle\ \text{`}\texttt{and}\text{'}\ \langle \mathit{Boundary2} \rangle \\
 & | & \text{`}\texttt{after}\text{'}\ \langle \mathit{Boundary2} \rangle\ \text{`}\texttt{until}\text{'}\ \langle \mathit{Boundary2} \rangle \\
\langle \mathit{Pattern} \rangle & ::= & \text{`}\texttt{always}\text{'}\ \underline{\mathit{OclExpressionCS}} \\
 & | & \text{`}\texttt{eventually}\text{'}\ \langle \mathit{RepeatableEventExp} \rangle \\
 & | & \text{`}\texttt{never}\text{'}\ \langle \mathit{RepeatableEventExp} \rangle \\
 & | & \langle \mathit{EventChainExp} \rangle\ \text{`}\texttt{preceding}\text{'}\ [\langle \mathit{TimeDistanceExp} \rangle] \\
 & & \langle \mathit{EventChainExp} \rangle \\
 & | & \langle \mathit{EventChainExp} \rangle\ \text{`}\texttt{responding}\text{'}\ [\langle \mathit{TimeDistanceExp} \rangle] \\
 & & \langle \mathit{EventChainExp} \rangle \\
\langle \mathit{Boundary1} \rangle & ::= & [\langle \mathit{IntegerLiteratureExpCS} \rangle]\ \langle \mathit{SimpleEvent} \rangle \\
 & & [\langle \mathit{TimeDistanceExp} \rangle] \\
\langle \mathit{Boundary2} \rangle & ::= & [\langle \mathit{IntegerLiteratureExpCS} \rangle]\ \langle \mathit{SimpleEvent} \rangle \\
 & & [\text{`}\texttt{at least}\text{'}\ \underline{\mathit{IntegerLiteratureExpCS}}\ \text{`}\texttt{tu}\text{'}] \\
\langle \mathit{EventChainExp} \rangle & ::= & \langle \mathit{Event} \rangle\ (\text{`}\texttt{,}\text{'}\ [\text{`}\texttt{\#}\text{'}\ \langle \mathit{TimeDistanceExp} \rangle]\ \langle \mathit{Event} \rangle)^* \\
\langle \mathit{TimeDistanceExp} \rangle & ::= & \langle \mathit{ComparingOp} \rangle\ \underline{\mathit{IntegerLiteratureExpCS}}\ \text{`}\texttt{tu}\text{'} \\
\langle \mathit{RepeatableEventExp} \rangle & ::= & [\langle \mathit{ComparingOp} \rangle\ \underline{\mathit{IntegerLiteratureExpCS}}]\ \langle \mathit{Event} \rangle \\
\langle \mathit{ComparingOp} \rangle & ::= & \text{`}\texttt{at least}\text{'} \mid \text{`}\texttt{at most}\text{'} \mid \text{`}\texttt{exactly}\text{'}
\end{array}
$$

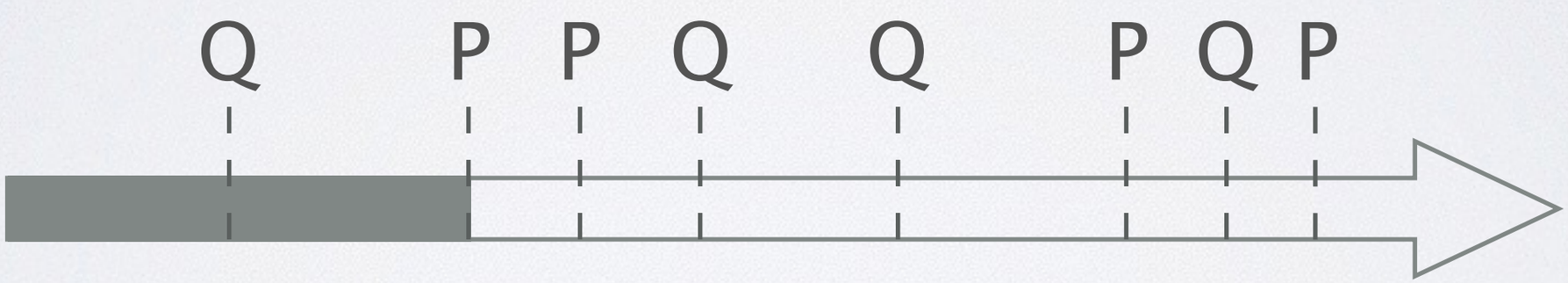
OCLR - event

```
isCalled(P)

isCalled(P, pre: n=0, post: n=1)
```

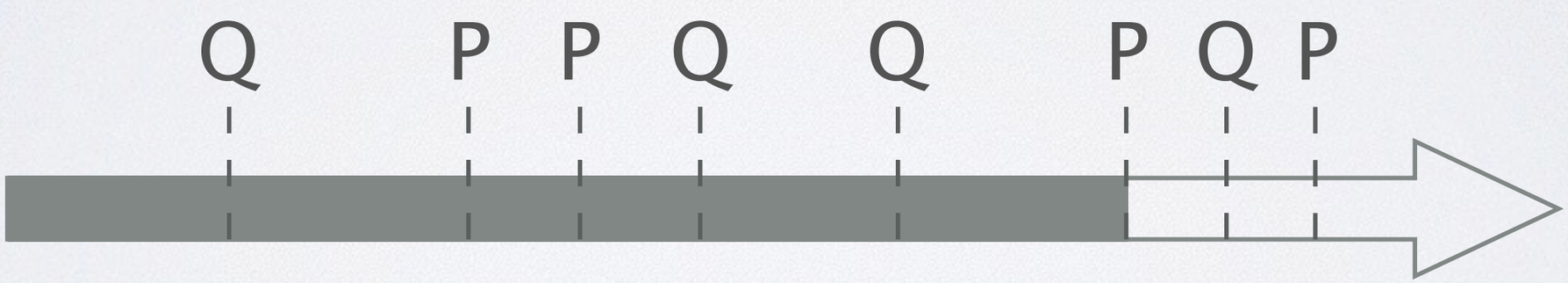
OCLR - Scopes

`before isCalled(P)`



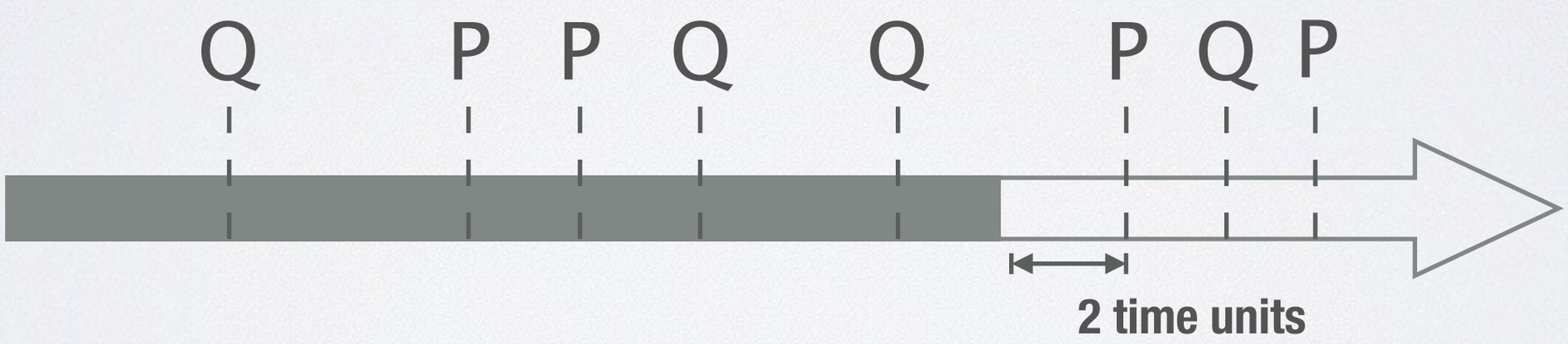
OCLR - Scopes

`before 3 isCalled(P)`



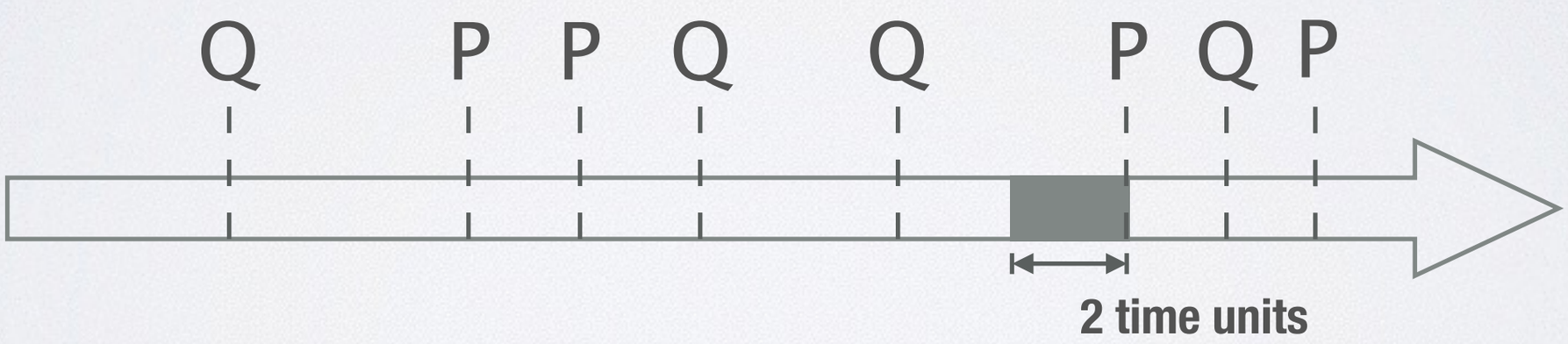
OCLR - Scopes

`before 3 isCalled(P) at least 2tu`



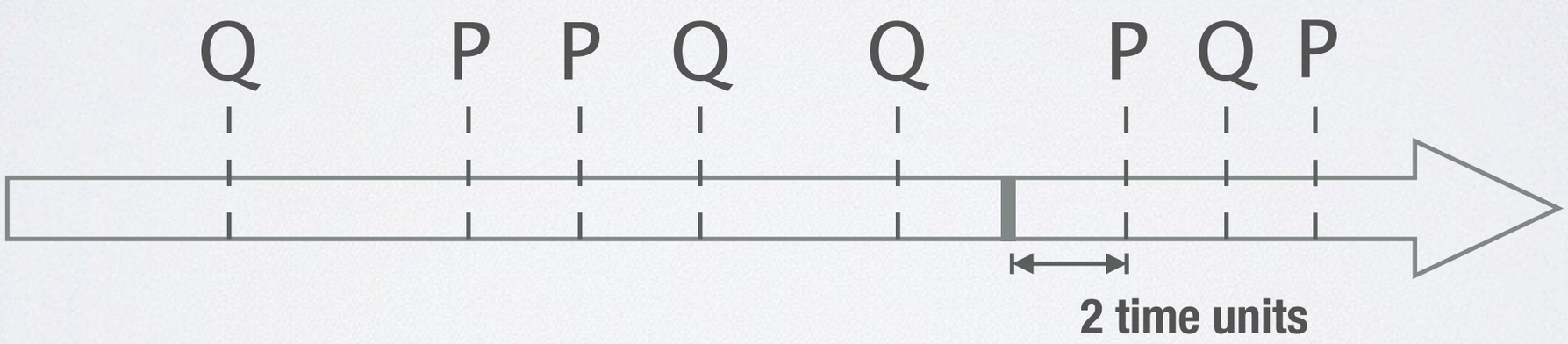
OCLR - Scopes

`before 3 isCalled(P) at most 2tu`



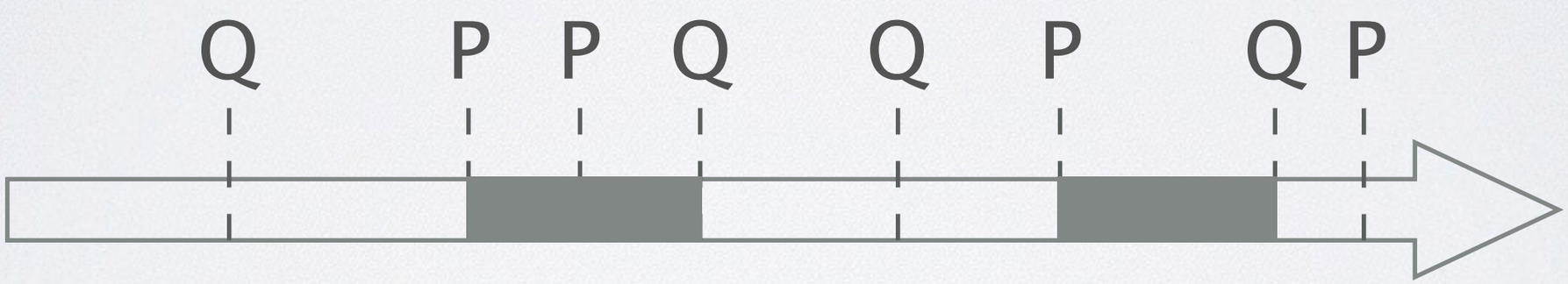
OCLR - Scopes

`before 3 isCalled(P) exactly 2tu`



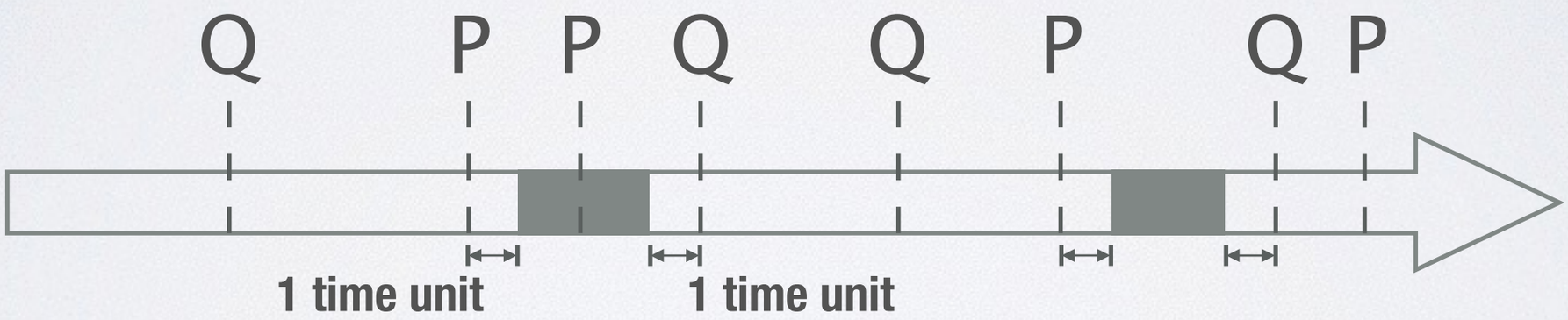
OCLR - Scopes

`between isCalled(P) and isCalled(Q)`



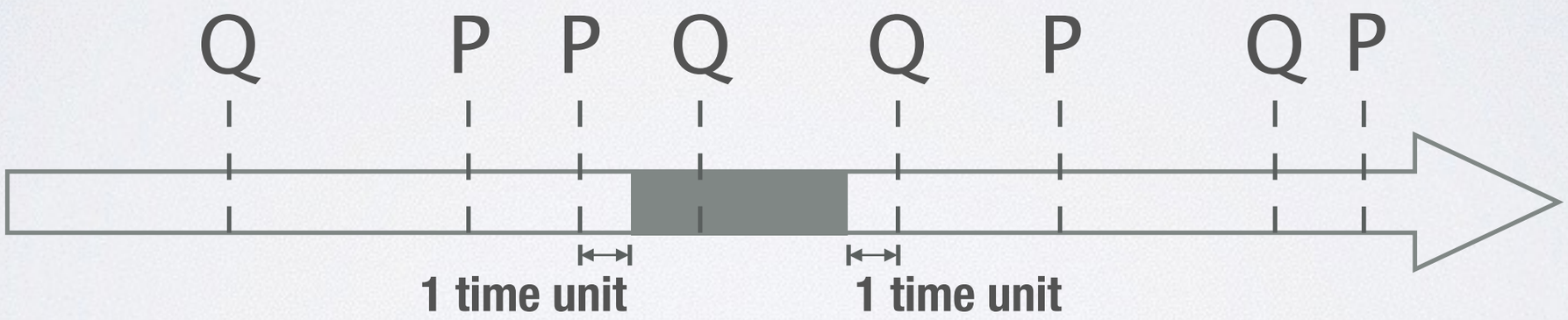
OCLR - Scopes

```
between isCalled(P) at least 1tu
  and isCalled(Q) at least 1tu
```



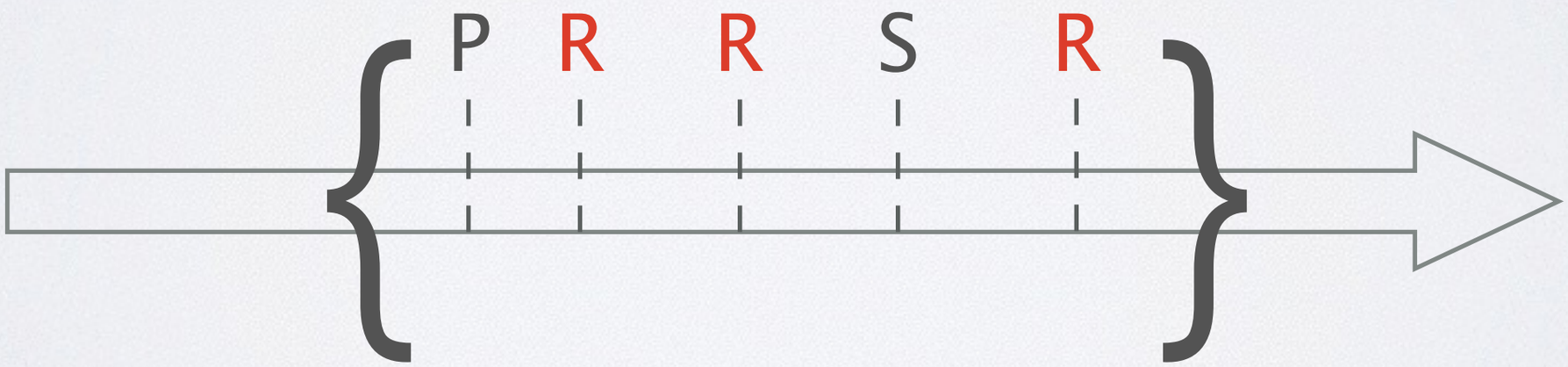
OCLR - Scopes

```
between 2 isCalled(P) at least 1tu
  and 2 isCalled(Q) at least 1tu
```



OCLR - Patterns

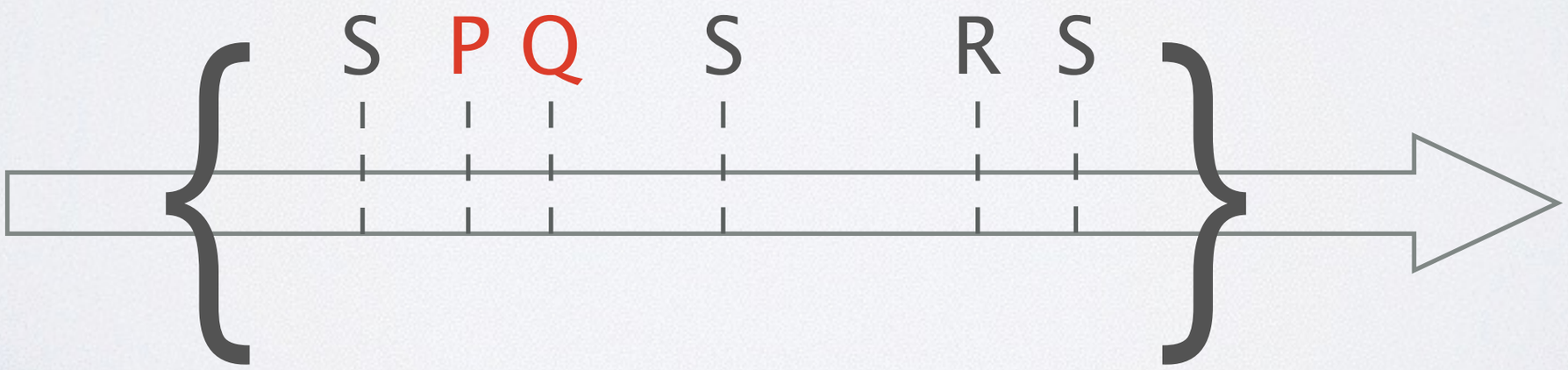
```
eventually exactly 2 isCalled(R)
eventually at most 2 isCalled(R)
eventually at least 2 isCalled(R)
eventually isCalled(R)
```



OCLR - Patterns

“If P happens, then Q will happen”

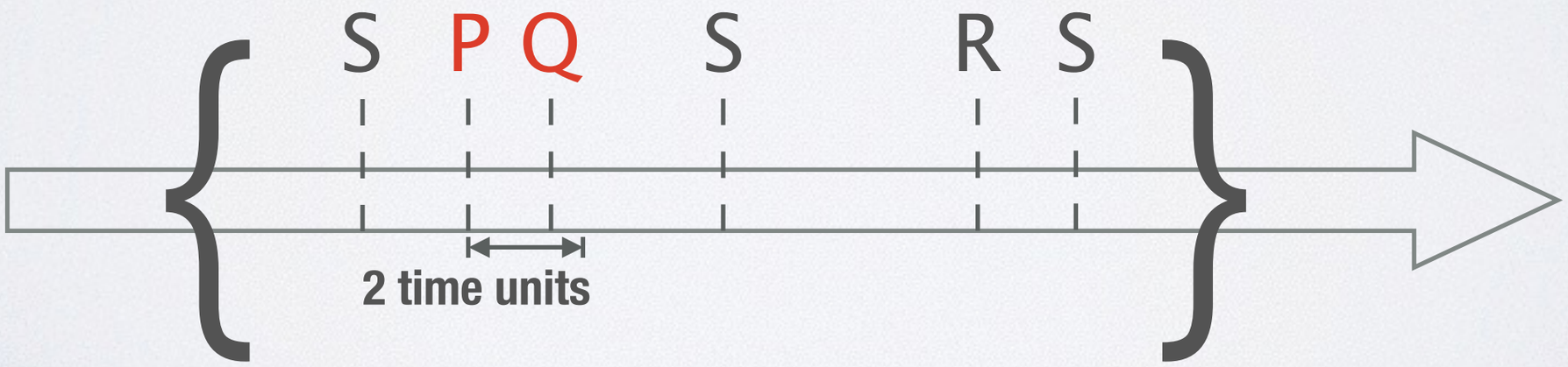
`isCalled(Q) responding isCalled(P)`



OCLR - Patterns

“If P happens, then Q will happen within 2 time units”

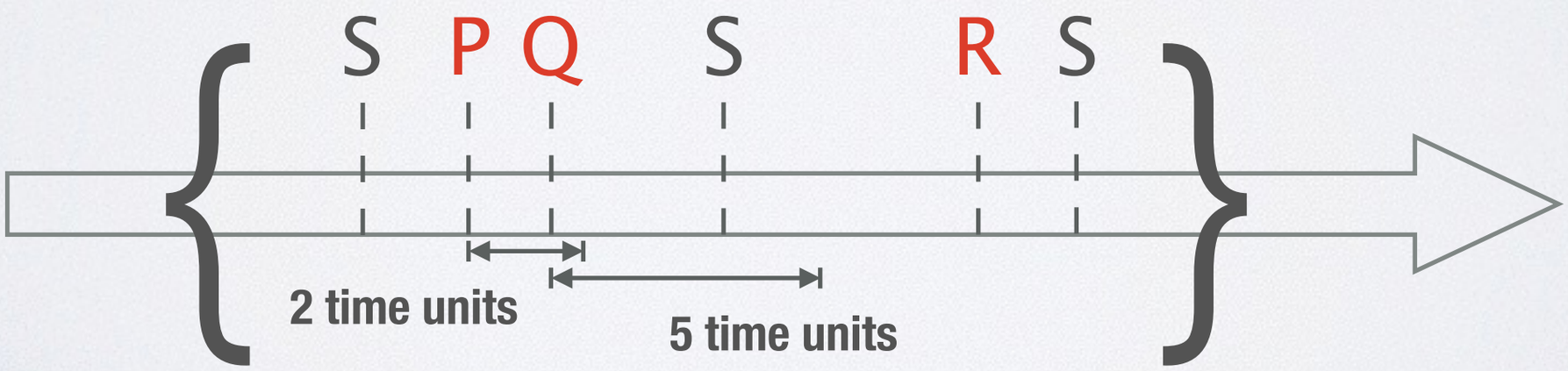
```
isCalled(Q) responding
at most 2tu isCalled(P)
```



OCLR - Patterns

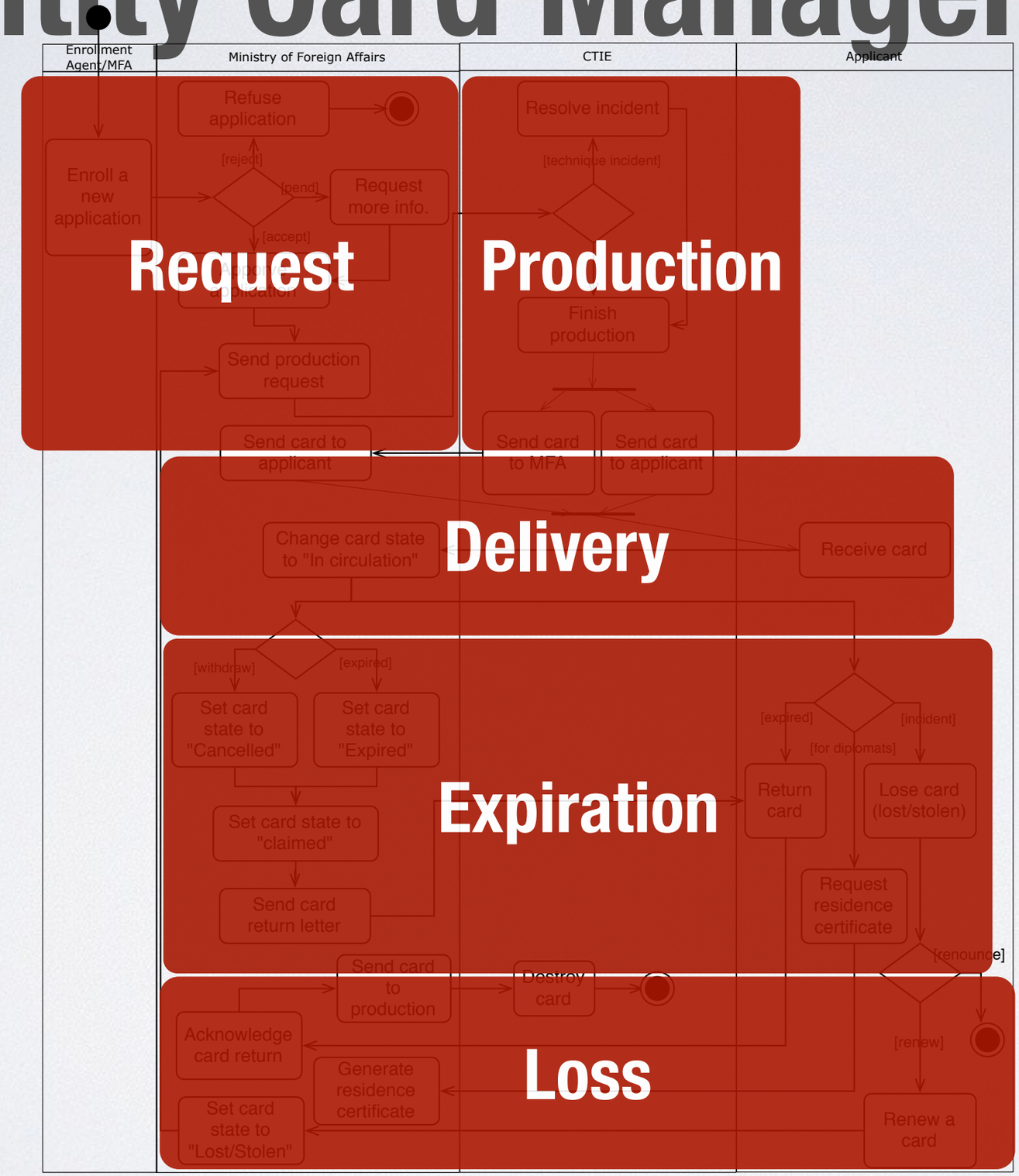
“If P happens, then Q will happen within 2 time units, followed, at least after 5 time units, by R”

```
isCalled(Q) #at least 5 tu
isCalled(R) responding
at most 2tu isCalled(P)
```



Case Study

Identity Card Management



47 temporal properties extracted from documentation analyzed and translated into OCLR

Request

“Once a card request is approved, the applicant is notified within three days; this notification has to occur before the production of the card is started.”

(Before + Response)

Request

```
temporal R1:
    let r : Request in
    before
    becomesTrue(r.card.state = CardState::InProduction)
          isCalled(notifyApproved(r.applicant))
  responding at most 3 “days”
  becomesTrue(r.state = RequestState::Approved)
```

“Once a card request is approved, the applicant is notified within three days; this notification has to occur before the production of the card is started”

Request

```
temporal R1:
    let r : Request in
    before
    becomesTrue(r.card.state = CardState::InProduction)
          isCalled(notifyApproved(r.applicant))
  responding at most 3 “days”
  becomesTrue(r.state = RequestState::Approved)
```

“Once a card request is approved, the applicant is notified within three days; this notification has to occur before the production of the card is started”

Request

```
temporal R1:
    let r : Request in
    before
    becomesTrue(r.card.state = CardState::InProduction)
        isCalled(notifyApproved(r.applicant))
  responding at most 3 “days”
  becomesTrue(r.state = RequestState::Approved)
```

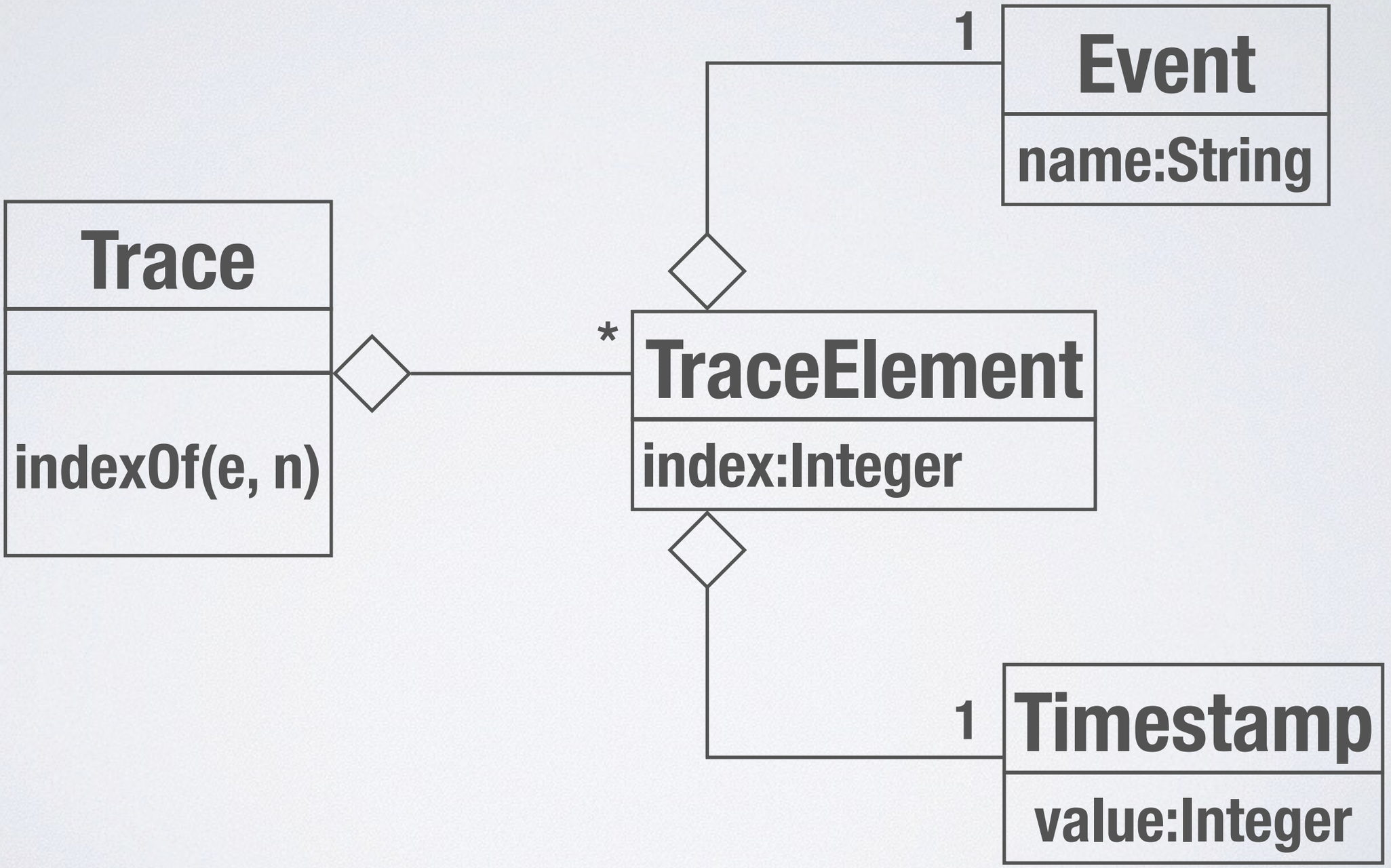
“Once a card request is approved, the applicant is notified within three days; this notification has to occur before the production of the card is started”

Property Checking

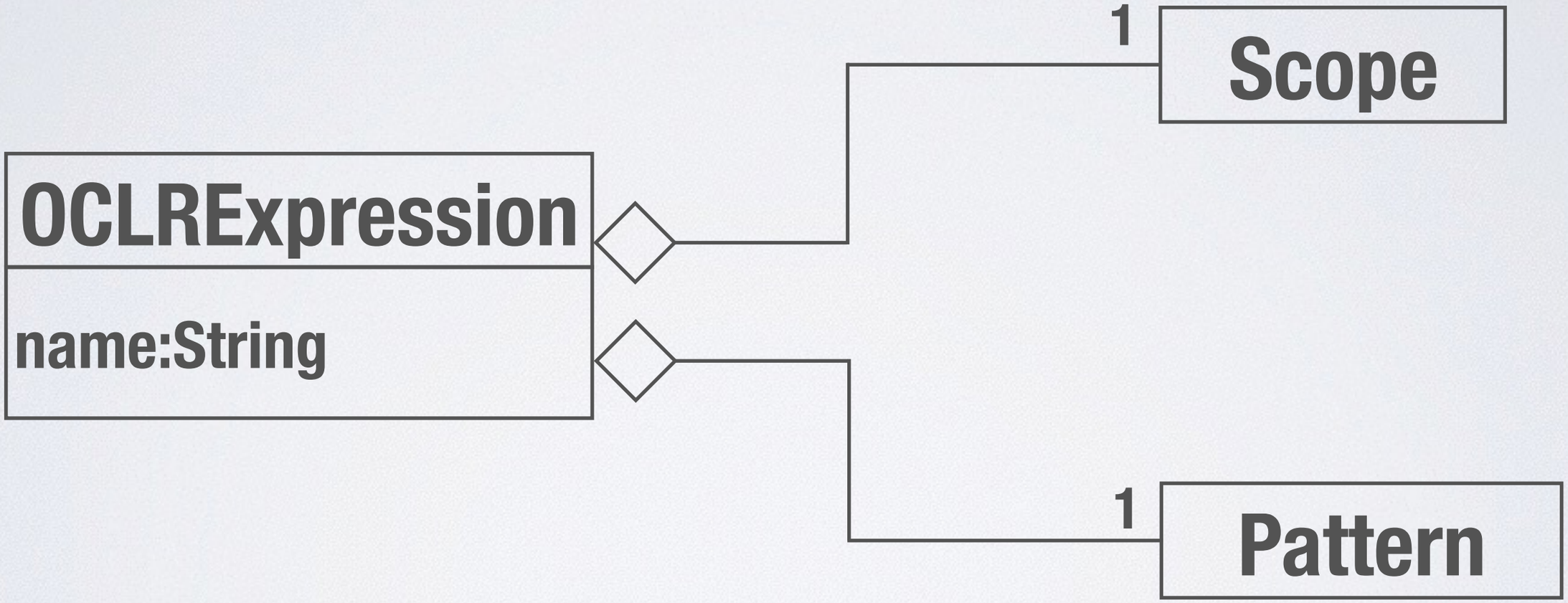
Model-driven Trace Checking

- based on a conceptual model of execution traces
- OCLR properties mapped into OCL constraints on the trace model
- The trace checking problem is reduced to checking OCL constraints on a instance of a trace model, using an OCL checker

Trace Model

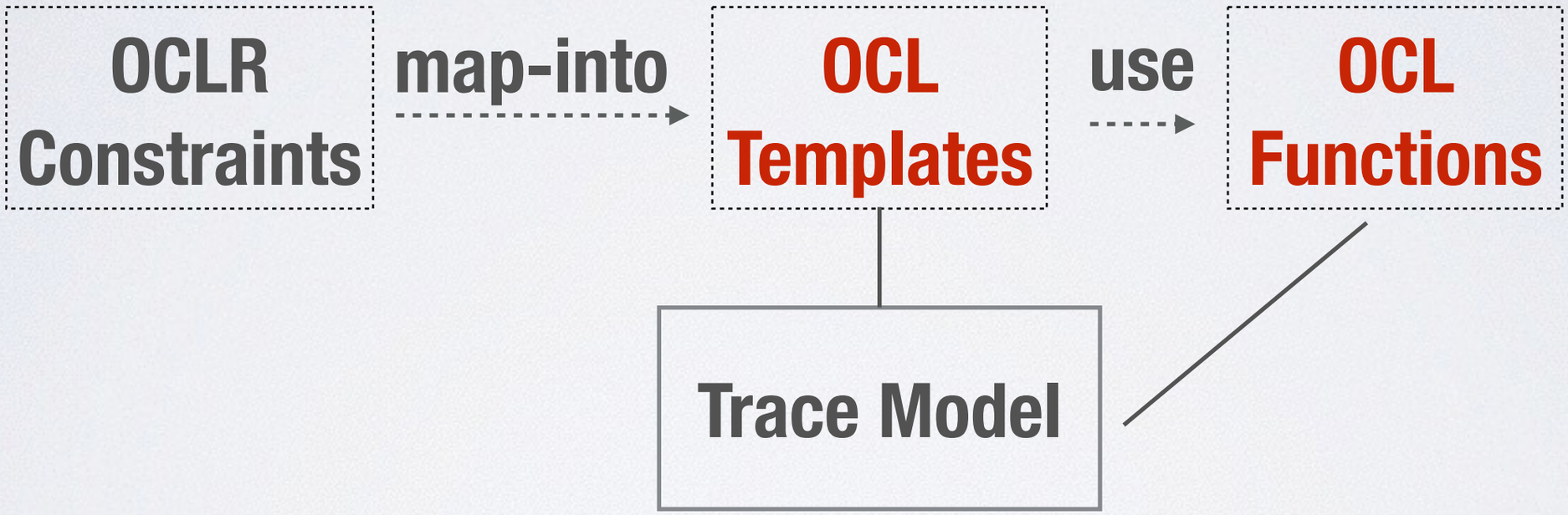

1
Event
name:String
Trace
indexOf(e, n)
*
TraceElement
index:Integer
1
Timestamp
value:Integer

OCLR Model



Offline Trace Checking

Offline Trace Checking



Model-driven Trace Checking in a nutshell

context Trace

inv: let subtraces=applyScope*S*(scope) in

subtraces->forAll(subtrace |

checkPattern*P*(subtrace, pattern))

OCL Templates - Scopes

```
context Trace
inv: let subtraces=applyScope*S*(scope) in
        subtraces->forAll(subtrace |
        checkPattern*P*(subtrace, pattern))
```

Scope

Globally

Before

After

Between-and

After-until

between [n1-th] e1 [time distance 1]
and [n2-th] e2 [time distance 2]

generalBetweenAnd(scope)

Iteration

Find the next occurrence of event A

If event A has occurred, find the next occurrence of event B

Select the trace segment between the occurrences of event A and B

Apply the time distance constraints on the boundaries, if either of them is not empty

Append the trace segment to the output

OCL Templates - Patterns

```
context Trace
inv: let subtraces=applyScope*S*(scope) in

     subtraces->forAll(subtrace |

     checkPattern*P*(subtrace, pattern))
```

Pattern

Universality
Existence
Absence
Precedence
Response

block1

preceding [time distance]

block2

block1

preceding at most T tu

block2

checkPrecedenceAtMost(subtrace, pattern)

Iteration

Find the last occurrence of block1

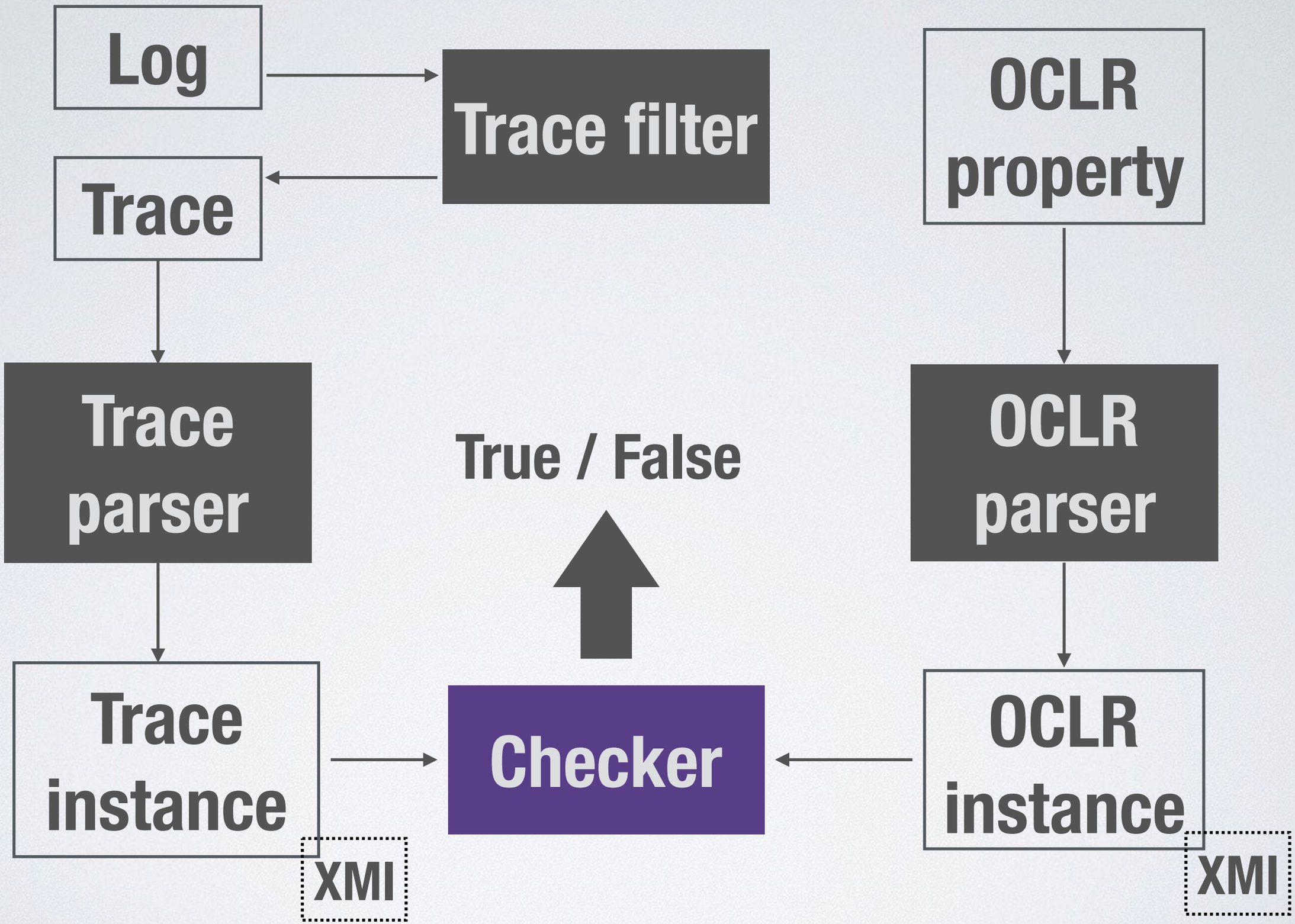
If the first event of block2 occurs:

If t(firstOfBlock2) - t(lastOfBlock1) > T, report a violation if size(block2) = 1 or postpone until the entire block2 is found; otherwise ignore and continue the iteration

Else if the next event of block2 occurs, report a violation if the entire block2 is found

Tool: OCLR-Check

http://weidou.github.io/OCLR-Check/


Log
Trace filter
Trace
OCLR property
Trace parser
True / False
OCLR parser
Trace instance
Checker
OCLR instance
XMI
XMI

Evaluation

Evaluation

- Real properties checked on synthesized traces
- Focus on scalability
- Comparison with MonPoly from ETHZ

OCLR Properties

- P1-P12: globally (12)
- P13-P20: before (8)
- P21-P31: after (11)
- P32-P38: between-and (7)

Why Synthesized Traces?

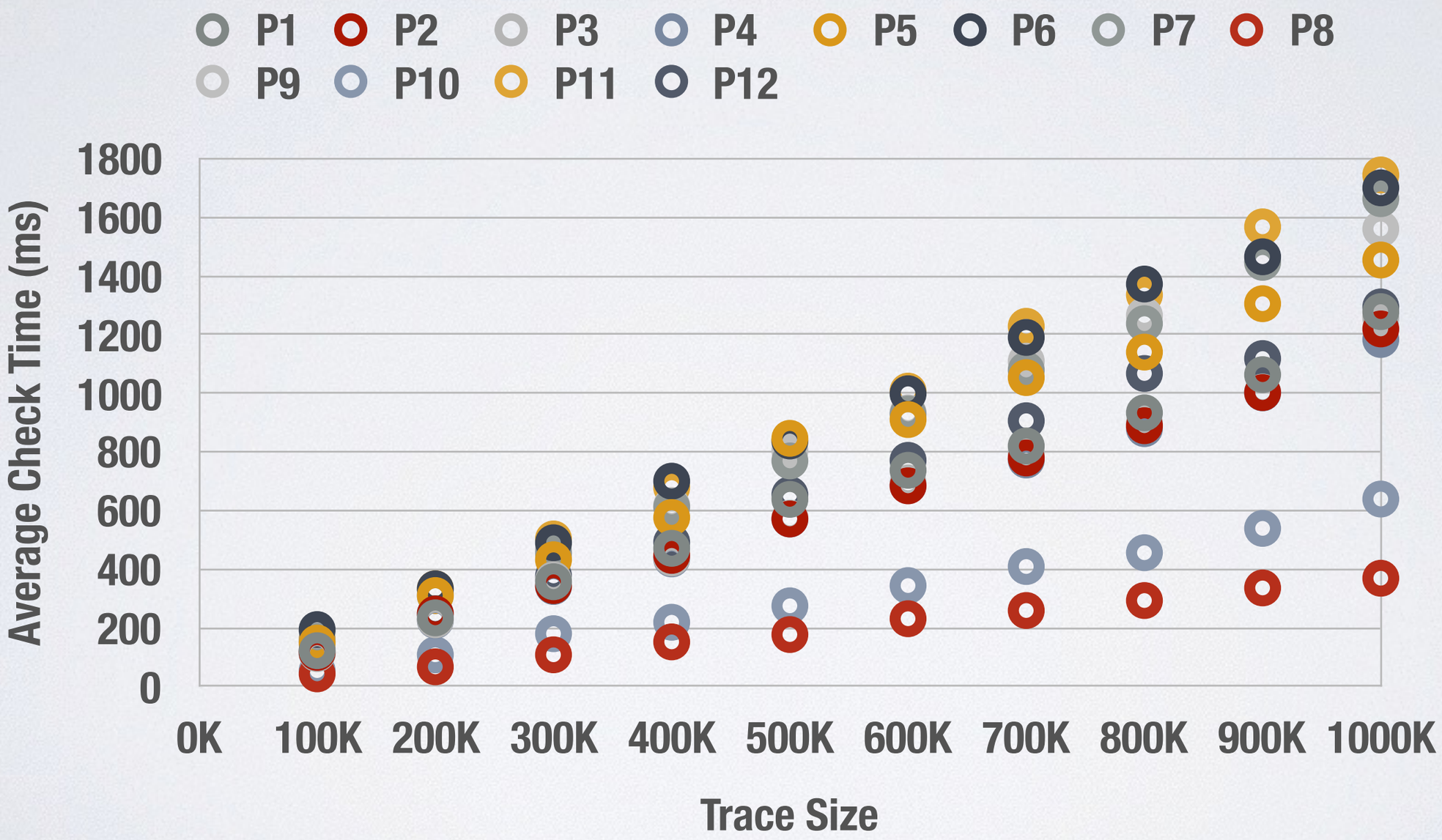
1. Real traces are often inadequate to cover a large range of trace lengths and a variety of properties

2. We want to be able to control the diversity of occurrences of patterns in the traces

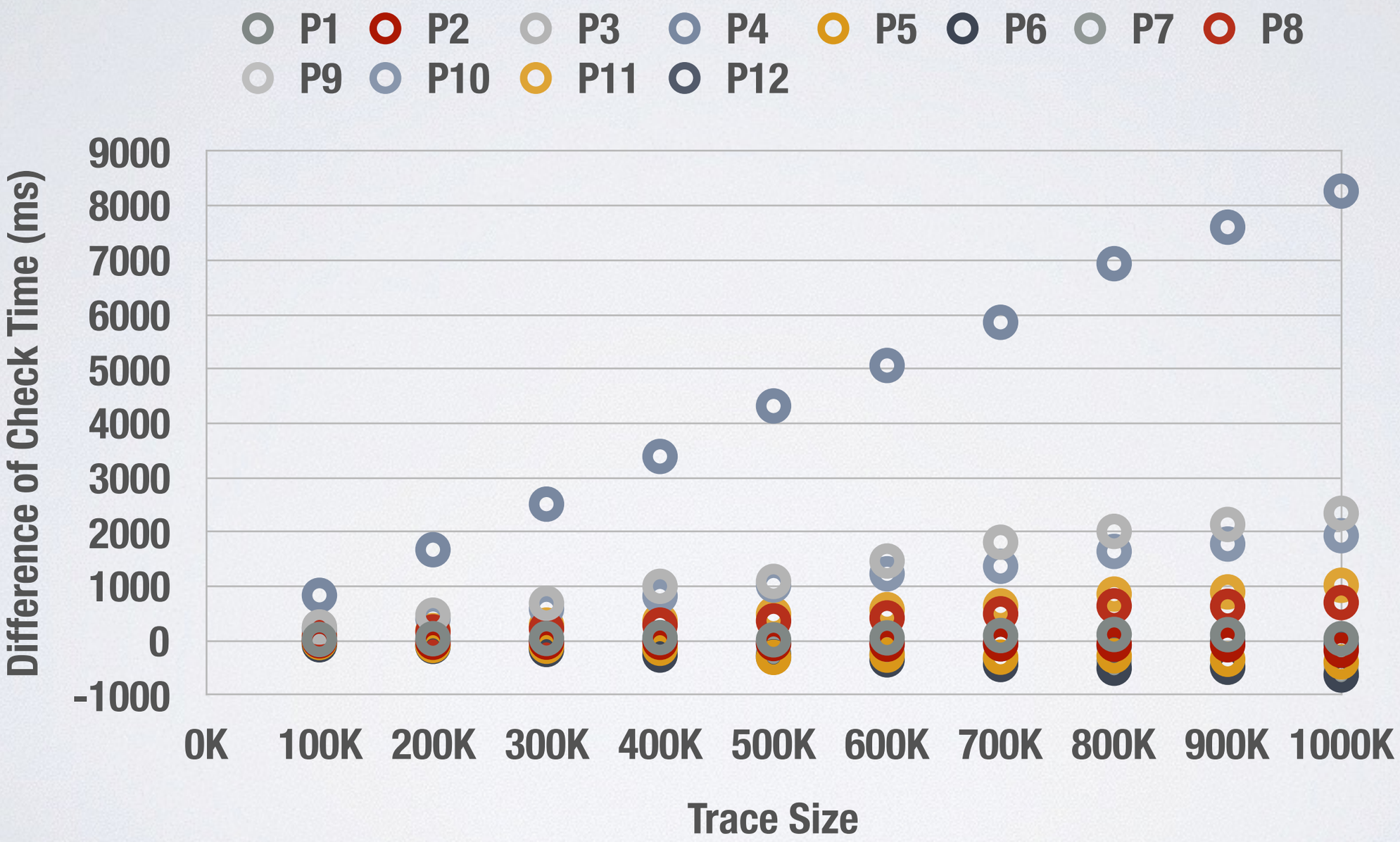
3. Real traces are not suitable for scalability evaluation, since they may contain faults

4. Real traces could not be shared for forming a public benchmark

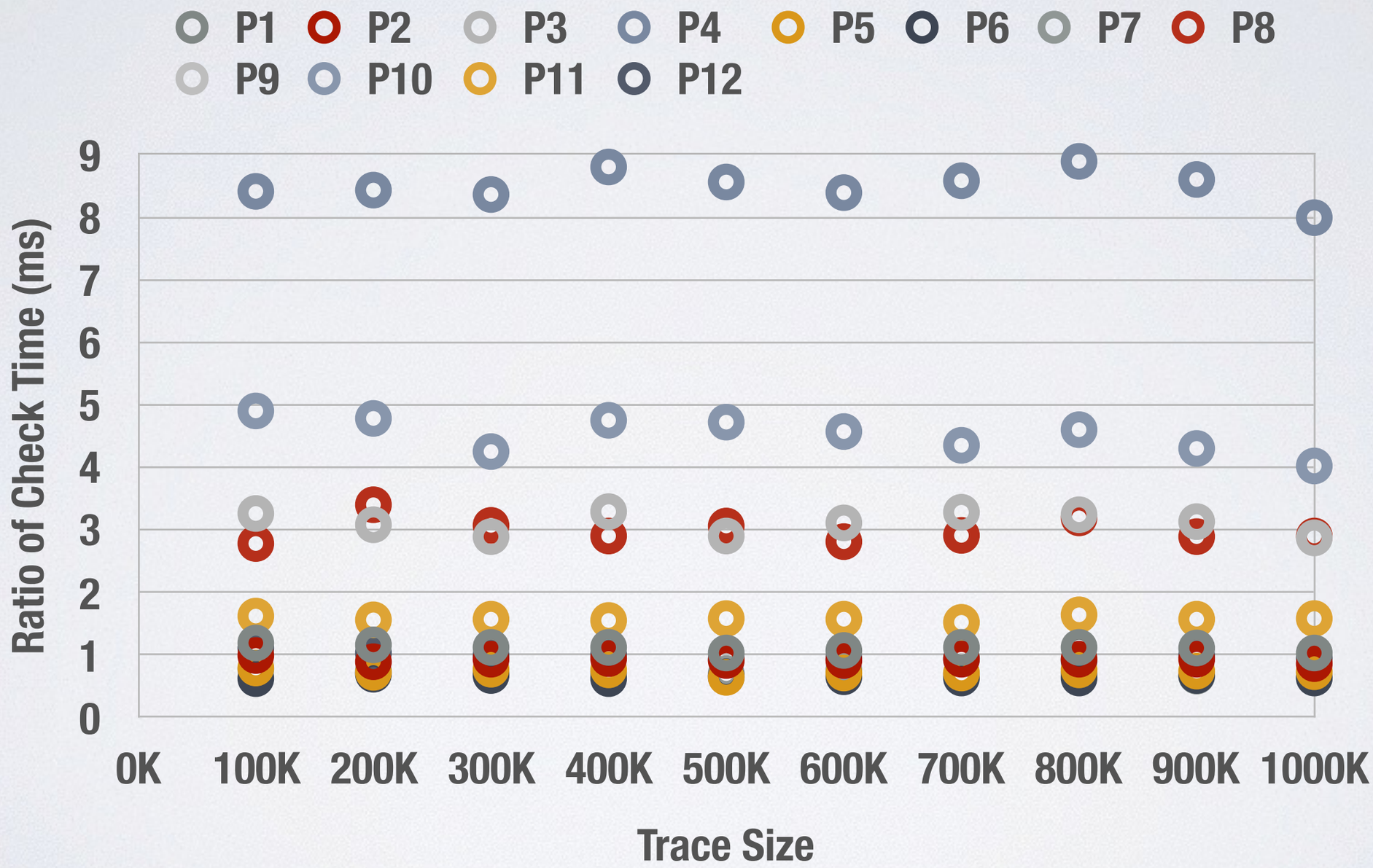
Properties with “globally”



t(MonPoly) - t(OCLR-Check)



t(MonPoly) / t(OCLR-Check)



Properties with “before”

- Existence(2), Absence(1), Precedence(3), Response(2)
- Traces length: 100K
- Varied the position of the boundary event: 10K, 20K, …, 100K
- checking time less than 400ms

Properties with “between-and”

- P32-P35: multiple segments
  - various number of segments (152ms to 390ms, linear)
  - fixed number of segments: 20 (207ms to 451ms, linear)
- P33-P38: single segment
  - various position of the segment (94ms to 158ms, linear)
  - various lengths of the segment (101ms to 357ms, linear)

Insights

- All the properties (trace size <= 1M) can be checked within 2 seconds
- OCLR-Check scales linearly with respect to the size of the trace
- The checking time of OCLR-Check is as good as Monopoly
- OCLR-Check has an overhead due to the loading time of the trace model in the Eclipse OCL

software verification & validation

securityandtrust.lu

Model-Driven Trace Checking of Temporal Properties

Domenico Bianculli

SnT Centre - University of Luxembourg

UNIVERSITÉ DU LUXEMBOURG