Logical Time @ work: the \textbf{RT-Simex} project

\textbf{Julien DeAntoni, Frédéric Mallet, Charles André}
\textbf{I3S/UNS/INRIA}

\textbf{Frédéric Thomas}
\textbf{Obeo}

\textbf{SAFA 2010}
Logical Time...

- focuses on causal relations between events
- Is independent of the abstraction level
- Is multi-clock (polychronous)
- Provides a partial order between events

After 23 starts of the computer, a disk check is done

The computation duration is 156 processor ticks
Logical Time...

- focuses on causal relations between events
- Is independent of the abstraction level
- Is multi-clock (polychronous)
- Provides a partial order between events

After 23 starts of the computer, a disk check is done

The computation duration is 156 processor ticks

In modern laptop, tick is logical since the processor speed depend on the battery level
Logical Time...

- focuses on causal relations between events
- Is independent of the abstraction level
- Is multi-clock (polychronous)
- Provides a partial order between events

LOGICAL TIME
Friedemann Mattern
Darmstadt University of Technology
Physical time Time...

• Can be seen as a special case of logical time

\[\text{After 5 seconds, it stops...} \approx \text{After 5 events of the “second” clock, it stops}\]
Logical Time @ work: the **RT-Simex** project

Julien DeAntoni, Frédéric Mallet, I3S/UNS/INRIA

Frédéric Thomas
Obeo

SAFA 2010
Retro-ingénierie de Traces d'analyse de SIMulation et d'EXécution de systèmes temps-réal

RT-Simex
Context: help the designer

Analysis of the execution
Context: execution traces

Designer → Code → Real time Multi-task execution

Execution analysis and reports

Temporal execution traces

JumpShot, Vampir, Linux Tool Eclipse Project

OTF, VCD
observations

Designer -> Model -> Code

SDL, AADL ...

Real time Multi-task execution

Temporal execution traces

Execution analysis and report
RT-Simex objectives

Execution analysis and report AT THE MODEL LEVEL

Real time Multi-task execution

Temporal execution traces
Processus RT-Simex

Modèle de conception (contraintes temporelles)

Génération de code instrumenté

Code source

Rétro-ingénierie du code

Exécution

Traces au format OTF

Réconciliation et analyse des différences (spécifié vs observé)

Modèle des traces temporelles observés

Rétro-ingénierie des traces
Logical Time @ work: the RT-Simex project

Julien DeAntoni, Frédéric Mallet, I3S/UNS/INRIA

Frédéric Thomas
Obeo

SAFA 2010
MARTE TIME MODEL

• SubProfile of the MARTE UML profile standardized by the OMG (Object Management Group)
  – Reviewed and accepted by the community
  – Implemented in Papyrus, magic draw, etc
  – Under Implementation in other UML tools

• A Domain Model integrated with eclipse and usable with Domain Specific Language
MARTE TIME MODEL

- The main concept is the **Clock**.
  - It is a way to specify a, possibly infinite, ordered set of instants
  - It can be logical or chronometric, discrete or dense
  - Its type is a ClockType

```plaintext
«clockType»
MyClockType

itsResolution: Integer [1]

«Clock»
itsClock: MyClockType [1]
itsResolution = 1

«Clock»
standard = TAI

- type = MyClockType
- unit = tick
```
Simplified view of the MARTE Time meta-Model
MARTE TIME MODEL

• Sketchy example of its use

User model

Producer

Consumer

c1
MARTE TIME MODEL

• Sketchy example of its use

User model

Producer

sendEvent

c1

Consumer

receiveEvent
MARTE TIME MODEL

• Sketchy example of its use

User model

Producer

sendEvent

c1

Consumer

receiveEvent

MARTE model

LogicalClock : ClockType
IsLogical : True
Nature : discrete

The ordered set of sendEvent is bijective with the ordered set of instants of c1.sendEvent
CCSL

- **Clock Constraint Specification Language**
  - Firstly introduced in the MARTE TIME profile
  - Declarative model-based language integrated with Eclipse
  - Formal semantics (both denotational and operational)
  - Tooled (TimeSquare)

→ Explicitly represents / specifies relations between clocks
CCSL (Clock Constraint Specification Language)

– Relations: dependencies between clocks
  • Coincidence → =
  • Exclusion → #
  • Precedence → <
  • Alternance → ~

– Expressions: a mean to create new clocks from others
  • Delay → delayedFor X on aClock
  • Filtering → aClock filteredBy aBinaryWord
  • Union → aClock union anotherClock
  • Intersection → aClock inter anotherClock
  • Periodicity → periodicOn aClock period X offset Y
  • ...

…
CCSL (Clock Constraint Specification Language)

- Relations: dependencies between clocks
  - Coincidence       \(\rightarrow\) =
  - Exclusion         \(\rightarrow\) #
  - Precedence        \(\rightarrow\) <
  - Alternance        \(\rightarrow\) ~

- Expressions: a mean to create new clocks from others
  - Delay             \(\rightarrow\) delayedFor \(X\) on \(aClock\)
  - Filtering         \(\rightarrow\) \(aClock\) filteredBy \(aBinaryWord\)
  - Union             \(\rightarrow\) \(aClock\) union \(anotherClock\)
  - Intersection      \(\rightarrow\) \(aClock\) inter \(anotherClock\)
  - Periodicity       \(\rightarrow\) periodicOn \(aClock\) period \(X\) offset \(Y\)

- Libraries: user-defined relations and expressions
CCSL
- Sketchy example of its use

User model

MARTE model

The ordered set of sendEvent is bijective with the ordered set of instants of c1.sendEvent
CCSL

- Sketchy example of its use

User model

MARTE model

```
LogicalClock : ClockType
IsLogical : True
Nature : discrete
```

```
Producer

sendEvent

Consumer

receiveEvent
```

```
c1.sendEvent: LogicalClock
```

```
c1.receiveEvent: LogicalClock
```

```
Producer

sendEvent

Consumer

receiveEvent
```

```
c1.sendEvent: LogicalClock
```

```
c1.receiveEvent: LogicalClock
```

```
Producer

sendEvent

Consumer

receiveEvent
```

```
c1.sendEvent: LogicalClock
```

```
c1.receiveEvent: LogicalClock
```

```
Producer

sendEvent

Consumer

receiveEvent
```

```
c1.sendEvent: LogicalClock
```

```
c1.receiveEvent: LogicalClock
```

```
Producer

sendEvent

Consumer

receiveEvent
```

```
c1.sendEvent: LogicalClock
```

```
c1.receiveEvent: LogicalClock
```

```
Producer

sendEvent

Consumer

receiveEvent
```

```
c1.sendEvent: LogicalClock
```

```
c1.receiveEvent: LogicalClock
```

```
Producer

sendEvent

Consumer

receiveEvent
```

```
c1.sendEvent: LogicalClock
```

```
c1.receiveEvent: LogicalClock
```

```
Producer

sendEvent

Consumer

receiveEvent
```

```
c1.sendEvent: LogicalClock
```

```
c1.receiveEvent: LogicalClock
```

```
Producer

sendEvent

Consumer

receiveEvent
```

```
c1.sendEvent: LogicalClock
```

```
c1.receiveEvent: LogicalClock
```

```
Producer

sendEvent

Consumer

receiveEvent
```

```
c1.sendEvent: LogicalClock
```

```
c1.receiveEvent: LogicalClock
```

```
Producer

sendEvent

Consumer

receiveEvent
```

```
c1.sendEvent: LogicalClock
```

```
c1.receiveEvent: LogicalClock
```

```
Producer

sendEvent

Consumer

receiveEvent
```

```
c1.sendEvent: LogicalClock
```

```
c1.receiveEvent: LogicalClock
```

```
Producer

sendEvent

Consumer

receiveEvent
```

```
c1.sendEvent: LogicalClock
```

```
c1.receiveEvent: LogicalClock
```

```
Producer

sendEvent

Consumer

receiveEvent
```

```
c1.sendEvent: LogicalClock
```

```
c1.receiveEvent: LogicalClock
```

```
Producer

sendEvent

Consumer

receiveEvent
```

```
c1.sendEvent: LogicalClock
```

```
c1.receiveEvent: LogicalClock
```

```
Producer

sendEvent

Consumer

receiveEvent
```

```
c1.sendEvent: LogicalClock
```

```
c1.receiveEvent: LogicalClock
```

```
Producer

sendEvent

Consumer

receiveEvent
```

```
c1.sendEvent: LogicalClock
```

```
c1.receiveEvent: LogicalClock
```

```
Producer

sendEvent

Consumer

receiveEvent
```

```
c1.sendEvent: LogicalClock
```

```
c1.receiveEvent: LogicalClock
```

```
Producer

sendEvent

Consumer

receiveEvent
```

```
c1.sendEvent: LogicalClock
```

```
c1.receiveEvent: LogicalClock
```

```
Producer

sendEvent

Consumer

receiveEvent
```
Graphical formal annotation over a UML model for RT-Simex...
Simulate and animate the UML/MARTE model in TimeSquare
(http://www-sop.inria.fr/aoste/dev/time_square/)
Simulate and animate the UML/MARTE model in TimeSquare
(http://www-sop.inria.fr/aoste/dev/time_square/)
Processus RT-Simex

1. Modèle de conception (contraintes temporelles)
2. Génération de code instrumenté
3. Code source
4. Exécution
5. Réconciliation et analyse des différences (spécifié vs observé)
6. Modèle des traces temporelles observés
7. Rétro-ingénierie des traces
8. Traces au format OTF
Processus RT-Simex

CCSL specification

Génération de code instrumenté

Code source

Rétro-ingénierie du code

Exécution

Modèle des traces temporelles observés

Rétro-ingénierie des traces

Traces au format OTF
Processus RT-Simex

CCSL specification

For a specific platform, this is a total order...
Processus RT-Simex

**CCSL specification**

For a specific platform, this is a total order...
For two communicating platforms, the traces are multi-clocks...
Processus RT-Simex

CCSL specification

If simulated it replays the execution...

CCSL specification
Processus RT-Simex

If simulated it replays the execution...
So you can have a timing diagram, animate the model and so on...

CCSL specification

Génération de code instrumenté

Code source

Rétro-ingénierie du code

Exécution

Traces au format OTF

CCSL specification
A specification and the actual execution
Processus RT-Simex

CCSL specification

Génération de code instrumenté

Code source

Rétro-ingénierie du code

Exécution

Traces au format OTF

Réconciliation et analyse des différences (spécifié vs observé)

Modèle des traces temporelles observés
Processus RT-Simex

CCSL assertions

Réconciliation et analyse des différences (spécifié vs observé)

Modèle des traces temporelles observés

CCSL specification

Génération de code instrumenté

Code source

Rétro-ingénierie du code

Exécution

Traces au format OTF
Processus RT-Simex

You can simulate both together to see if a violation occurs
Processus RT-Simex

You can simulate both together to see if a violation occurs

Violation of "Clk3b isSubClockOf clk3"

CCSL assertions
CCSL specification
Conclusion

Modèle de conception (contraintes temporelles)

Réconciliation et analyse des différences (spécifié vs observé)

Modèle des traces temporelles observés

Génération de code instrumenté

Code source

Rétro-ingénierie du code

Exécution

Rétro-ingénierie des traces

Traces au format OTF
Conclusion

- Logical Time, via Marte and CCSL is used for
  - The specification the expected behaviour
  - The simulation at the model level of this behaviour
  - The simulation at the model level of the actual behaviour (as monitored during the execution)
  - The comparison between the specified and the actual behaviour