# Trace

| REVISION HISTORY | | | |
|---|---|---|---|
| NUMBER | DATE | DESCRIPTION | NAME |
| | | | |

# Contents

# Chapter 1

# Introduction

The goal of this meta-model is to keep traces of events about timing concerns. Those events come from different possible sources:

- simulators
- analysis providing worst cases
- real network traces
- application logs

More over we must not reinvent the wheel, or at least gain experience from previous projects. That is:

- Tempo metamodel traces
- Trace Compass[1]
- Complex Event Processing (CEP)[2]
- OLAP[3]

Finally, it is mandatory for the meta-model to be self-contained.

[1] http://tracecompass.org/
[2] https://en.wikipedia.org/wiki/Complex_event_processing
[3] https://en.wikipedia.org/wiki/Online_analytical_processing

# Chapter 2

# Rationale

## 2.1 . . . from running code

Let say that your code is creating the events itself. For that you may have a @log@ function that create a timestamp and a message.

```
void foo()
{
  log("Running foo"):
  /* ... */
  log("foo terminated"):
}

void task1Code()
{
  log("Running task1");
  /* ... */
  foo();
  /* ... */
  log("task1 terminated");
}

void main()
{
   task1 = xTaskCreate(task1Code, ...);
}
```

In that case, when generating the event you only have references to functions and tasks' definition.

NB: Most of the time you will have a 1:1 relation between task and definition of task but it may not always be the case.

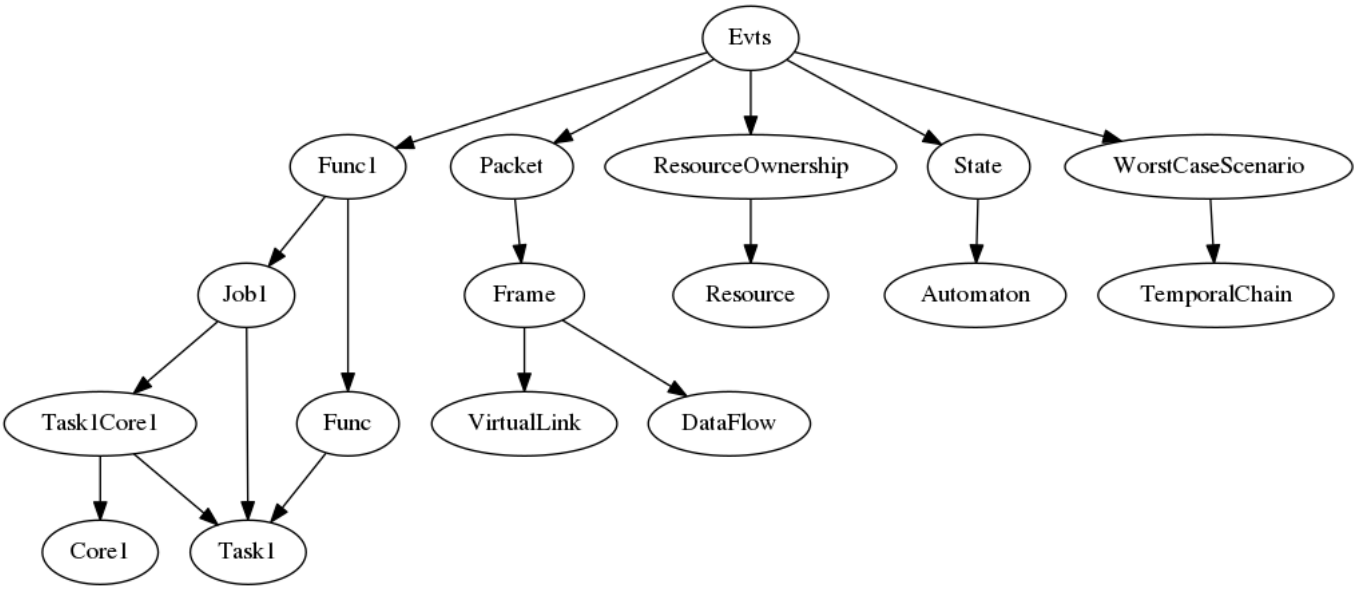
## 2.2 . . . from simulator

In this case, the simulator knows everything from step (aka function) and step's instance, job and task, and even the computing resource used. If it includes a network (sub)simulator or a functional (sub)simulator, you will also know the resource ownership and requests.

# Chapter 3

# Synthesis

The problem is that events can be aggregated in overlapping traces, ie they have several dimensions of interpretation. On the following diagram, the arrow shall be interpreted as "is a subset of":




This forms a lattice and not a strict hierarchy.

Thus our meta-model must separate the list of events, ie the log, and its analysis. Thus the @Trace@ concept is here to **contains** the list of events, while the @Slice@ will *aggregate* some events in a hierarchical way.

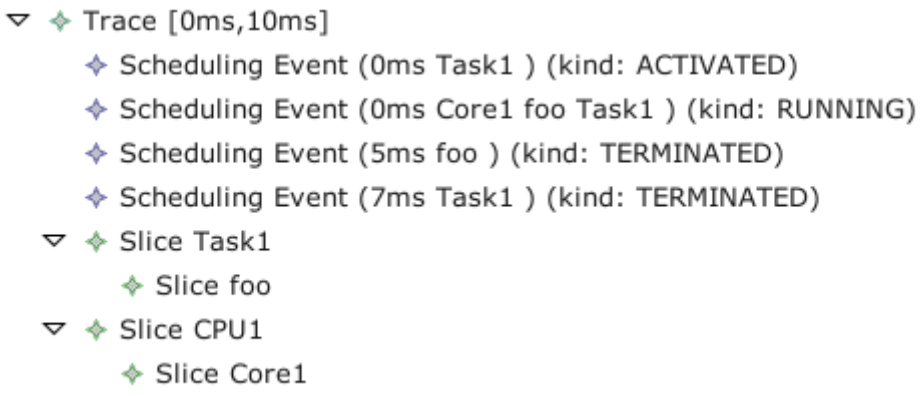
Obviously, on events creation, one partially know about some of those slices (Task, Step/Function, Job, Frame, etc) but not the whole hierarchy. Thus an interpretation pass (aka events analysis) may be necessary to build the hierarchy to fill in the blanks. The field @about@ of event is here to link to known slices. In the running code example, one may translate from string containing the name of the function to the Slice representing that function (really the extension of the function).

But Task/Job/Function is not the whole story. Designers are also interested in end-to-end analysis. Thus this model allow tools to link a chain a events which has cross-cutting concerns.

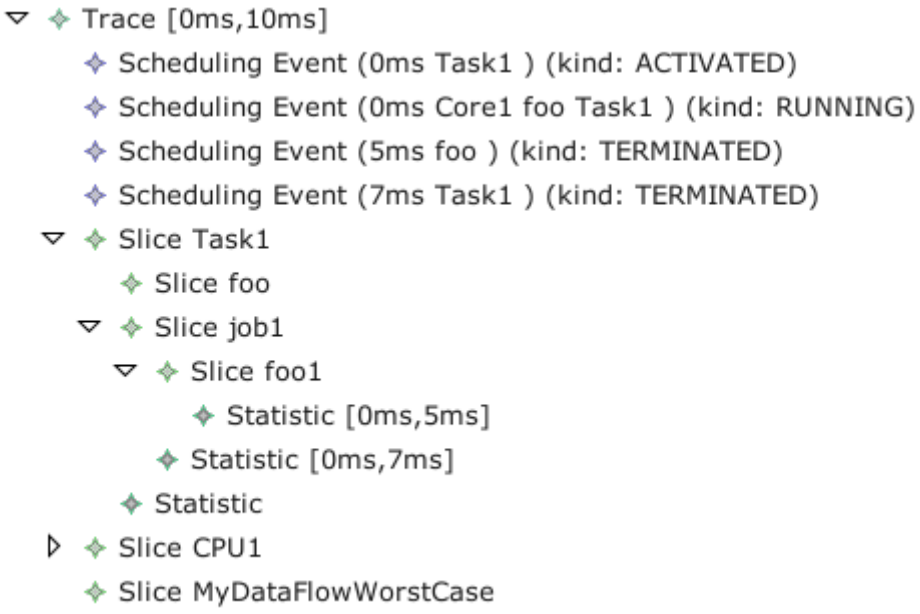
# Chapter 4

# Examples

An implementation may give the following trace:




Then, one may analysed the traces to consolidate it, ie aggregate more dimensions, like the resource, the jobs, the function instance, etc.



Obviously, a simulator may already provides a consolidated traces.

Also note how one can refer to temporal chain of events, like the worst case scenario for a data-flow for instance.

# Chapter 5

# Trace package

## 5.1 Overview

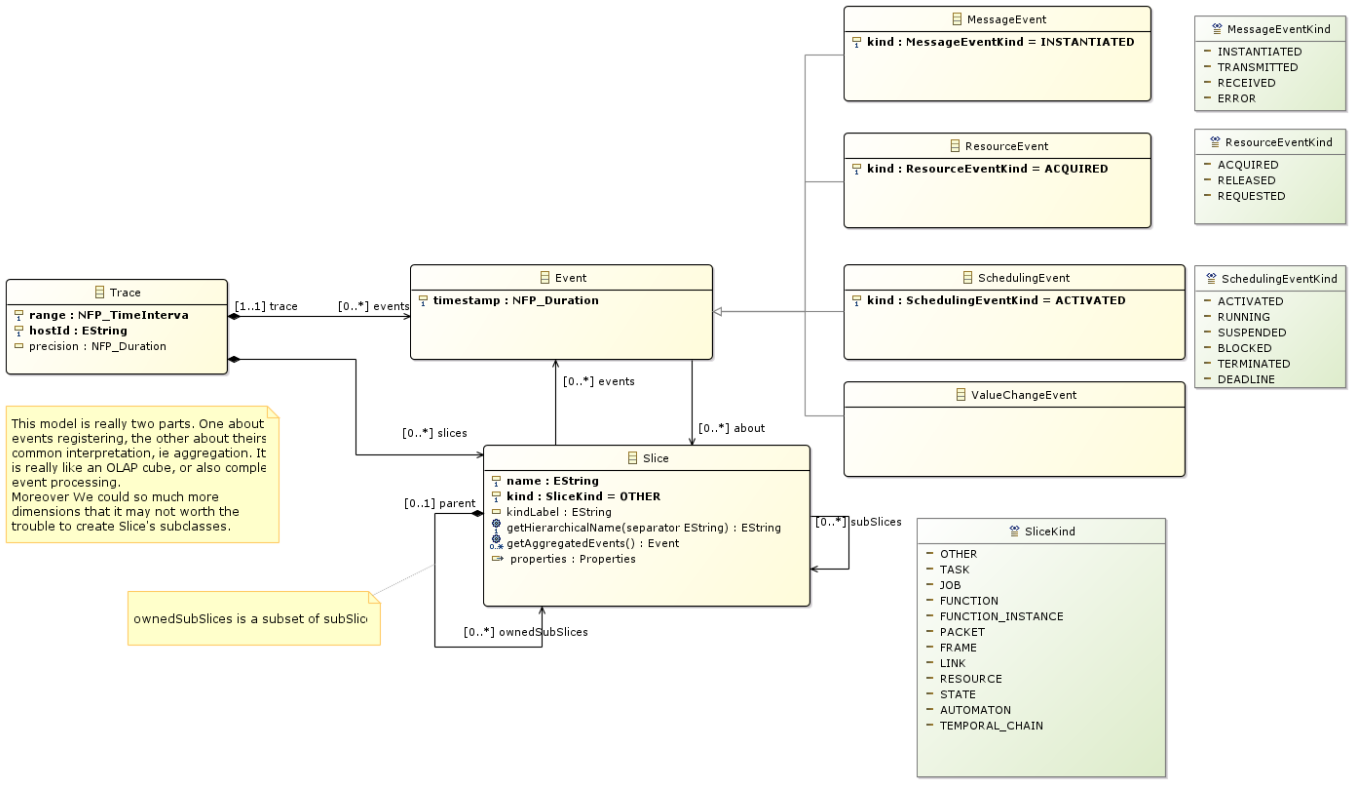



Figure 5.1: trace-class-diagram-overview

## 5.2 Trace classifier

### 5.2.1 Attributes

- range: NFP_TimeInterval [1:1]

- hostId: EString [1:1]See TraceCompass hostId.

```
The host ID is not necessarily the hostname, but should be a unique identifier for ↩
    the machine on which the trace was taken.
It can be used to determine if two traces were taken on the exact same machine ( ↩
   timestamp already synchronized, resources with same id are the same if taken at ↩
    the same time, etc).
```

- precision: NFP_Duration [0:1]

#### 5.2.2 Semantics

A generic container of list of events that has either been observed, simulated, or computed (like a worst-case scenario).

## 5.3 Event classifier

#### 5.3.1 Specializations

- ResourceEvent from trace
- SchedulingEvent from trace
- MessageEvent from trace
- ValueChangeEvent from trace

#### 5.3.2 Attributes

- timestamp: NFP_Duration [1:1]

#### 5.3.3 Semantics

TODO: write a semantic

## 5.4 NFP_Duration classifier

A generic Duration as defined per NFP Package from Time4sys project.

See org.polarsys.time4sys.marte.nfp.Duration.

TODO: write a semantic

## 5.5 NFP_TimeInterval classifier

A generic Time interval as defined per NFP Package from Time4sys project.

See org.polarsys.time4sys.marte.nfp.TimeInterval.

TODO: write a semantic

## 5.6 SchedulingEventKind classifier

This shall be enough to described the state-machine of classical tasks.

### 5.6.1 Values

- ACTIVATED
- RUNNING
- SUSPENDED
- BLOCKED
- TERMINATED
- DEADLINE

### 5.6.2 Semantics

It expected to follow this state-machine:

## 5.7 ResourceEvent classifier

### 5.7.1 Generalizations

- Event from trace

### 5.7.2 Attributes

- kind: ResourceEventKind [1:1]

### 5.7.3 Semantics

TODO: write a semantic

## 5.8 SchedulingEvent classifier

### 5.8.1 Generalizations

- Event from trace

### 5.8.2 Attributes

- kind: SchedulingEventKind [1:1]

#### 5.8.3 Semantics

TODO: write a semantic

### 5.9 ResourceEventKind classifier

#### 5.9.1 Values

- ACQUIRED
- RELEASED
- REQUESTED

#### 5.9.2 Semantics

TODO: write a semantic

### 5.10 MessageEvent classifier

#### 5.10.1 Generalizations

- Event from trace

#### 5.10.2 Attributes

- kind: MessageEventKind [1:1]

#### 5.10.3 Semantics

TODO: write a semantic

### 5.11 MessageEventKind classifier

#### 5.11.1 Values

- INSTANTIATED
- TRANSMITTED
- RECEIVED
- ERROR

#### 5.11.2 Semantics

TODO: write a semantic

## 5.12 Slice classifier

A slice is an aggregation of Events so as to group them together in a logical way. For instance, it could be:

- the Gantt line view of a processor usage,
- an end-to-end dataflow,
- an end-to-end network packet travel,
- etc.

### 5.12.1 Attributes

- name: EString [1:1]
- kind: SliceKind [1:1]
- kindLabel: EString [0:1]

### 5.12.2 Semantics

TODO: write a semantic

## 5.13 SliceKind classifier

### 5.13.1 Values

- OTHER
- TASK
- JOB
- FUNCTION
- FUNCTION_INSTANCE
- PACKET
- FRAME
- LINK
- RESOURCE
- STATE
- AUTOMATON
- TEMPORAL_CHAIN

### 5.13.2 Semantics

TODO: write a semantic

## 5.14 Properties classifier

### 5.14.1 Attributes

- range: NFP_TimeInterval [0:1]
- blockingTime: NFP_Duration [0:1]
- executionTime: NFP_Duration [0:1]
- remainingTime: NFP_Duration [0:1]
- responseTime: NFP_Duration [0:1]
- absoluteDeadline: NFP_Duration [0:1]
- index: ELong [0:1]

### 5.14.2 Semantics

TODO: write a semantic

## 5.15 ValueChangeEvent classifier

### 5.15.1 Generalizations

- Event from trace

### 5.15.2 Semantics

TODO: write a semantic