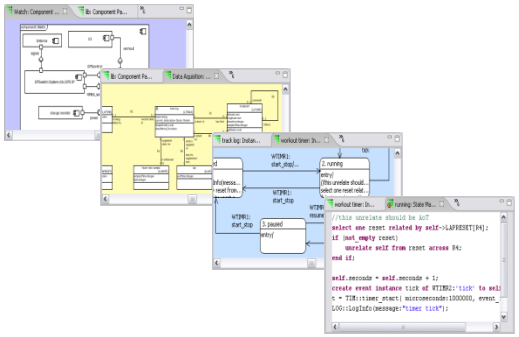
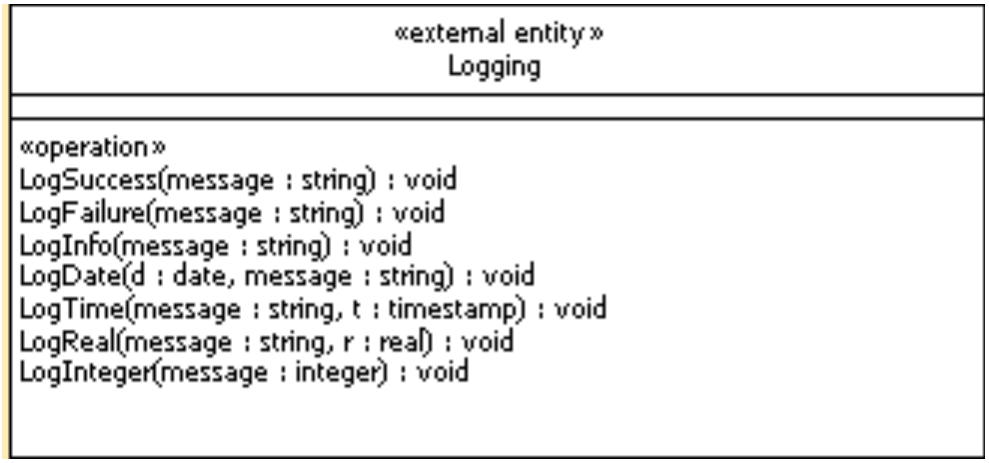
The xtUML method – Specifying Activities

- **Analysis** – questioning, thinking, sketching...
  - Descriptive UML diagrams
    - use case, sequence, ...
- **Executable Modeling** – formalizing the analysis:
  - Component Diagrams (partitioning/interfaces)
  - Class Diagrams (data)
  - State Machines (control)
  - **Activities (processing)**
- **Verification**
  - Interpretive Model Execution
- **Code generation**
  - Template and Rule-Based Translation



External Entities

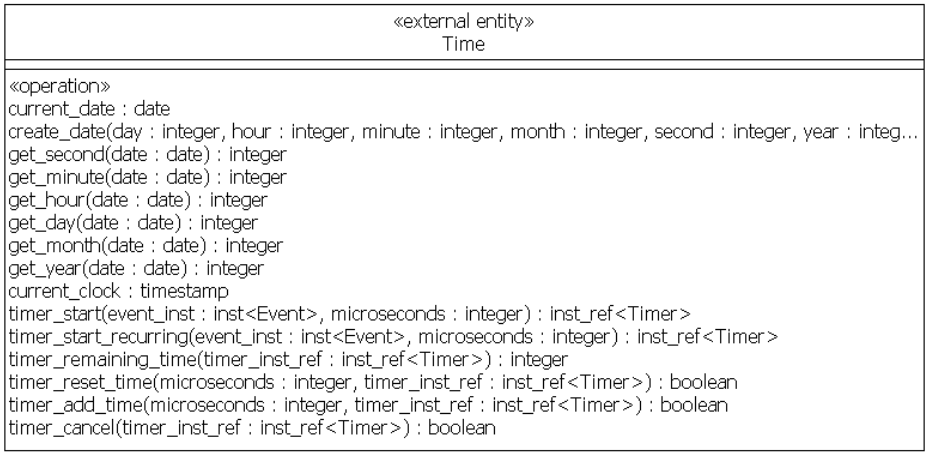
- EE's are used for accessing functions external to the model
  - 'C' routines in legacy code for translation
  - Java methods for use in model verification
- Common logging and timing functions included by default in xtUML projects in LOG:: and TIM:: packages



Time Package

- **Date and Time Access**

```
now = TIM::current_date();
year  = TIM::get_year(date:now);
month = TIM::get_month(date:now);
day   = TIM::get_day(date:now);
hour   = TIM::get_hour(date:now);
minute = TIM::get_minute(date:now);
second = TIM::get_second(date:now);
```



- **Timers**

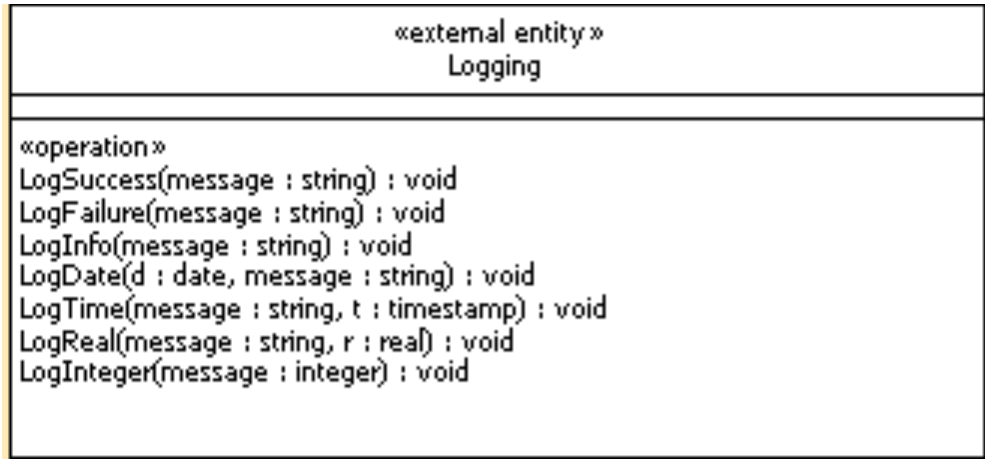
```
st = row.sampling_time;
create event instance move_on of SP1:finished_sampling() to self;
mo_timer = TIM::timer_start(microseconds:st, event_inst:move_on);
```

```
create event instance evt of WorkoutTimer3:tick() to self;
self.timer = TIM::timer_start_recurring(event_inst: evt, microseconds: 1000000);
```

Logging Package

- Send test messages to the console.

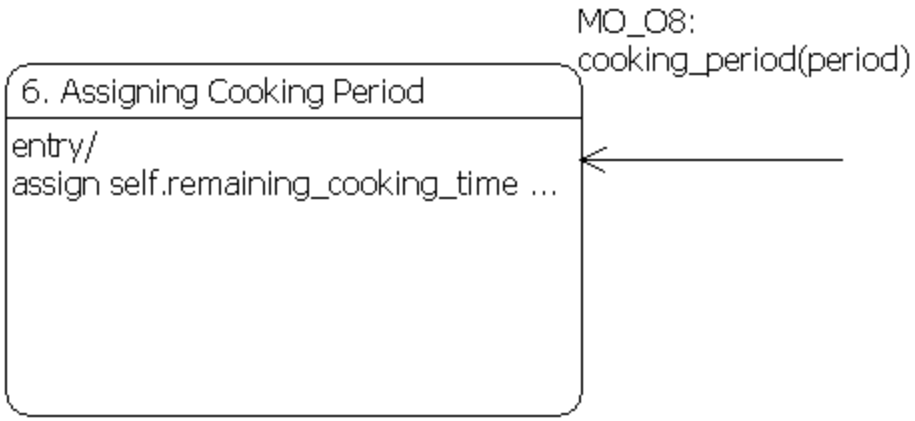
```
now = TIM::current_date();
LOG::LogDate(d:now,message:"Current Date:");
LOG::LogInfo(message:"Hello World!!");
```



Events

```
generate MO_O8:cooking_period(period: 60) to oven;
```

Event name Event parameters Destination



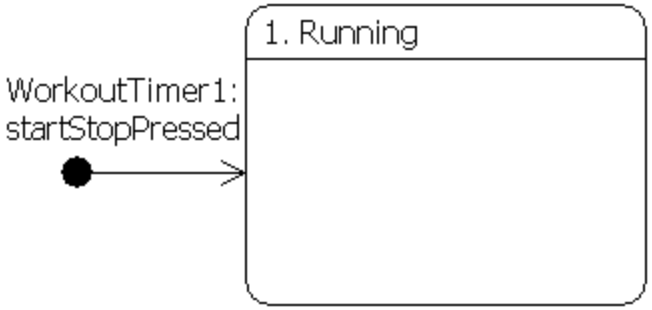

```
// Receiving Event - rcvd_evt is a keyword.

assign self.remaining_cooking_time = rcvd_evt.period;
```

Creation events

- Events can be used to create an instance of a class
  - Must generate an event assigned to a Creation Transition
  - This new instance has no handle name to be addressed by

```
// asynchronous instance creation
generate WorkoutTimer1() to WorkoutTimer creator;
```

Class keyletters



Event Pre-creation

- **An event may be created without sending it by using the *create event* statement.**

```
Create object instance class_inst of WorkoutTimer;

Create event instance event_inst of
WorkoutTimer1:startStopPressed to class_inst ;
```

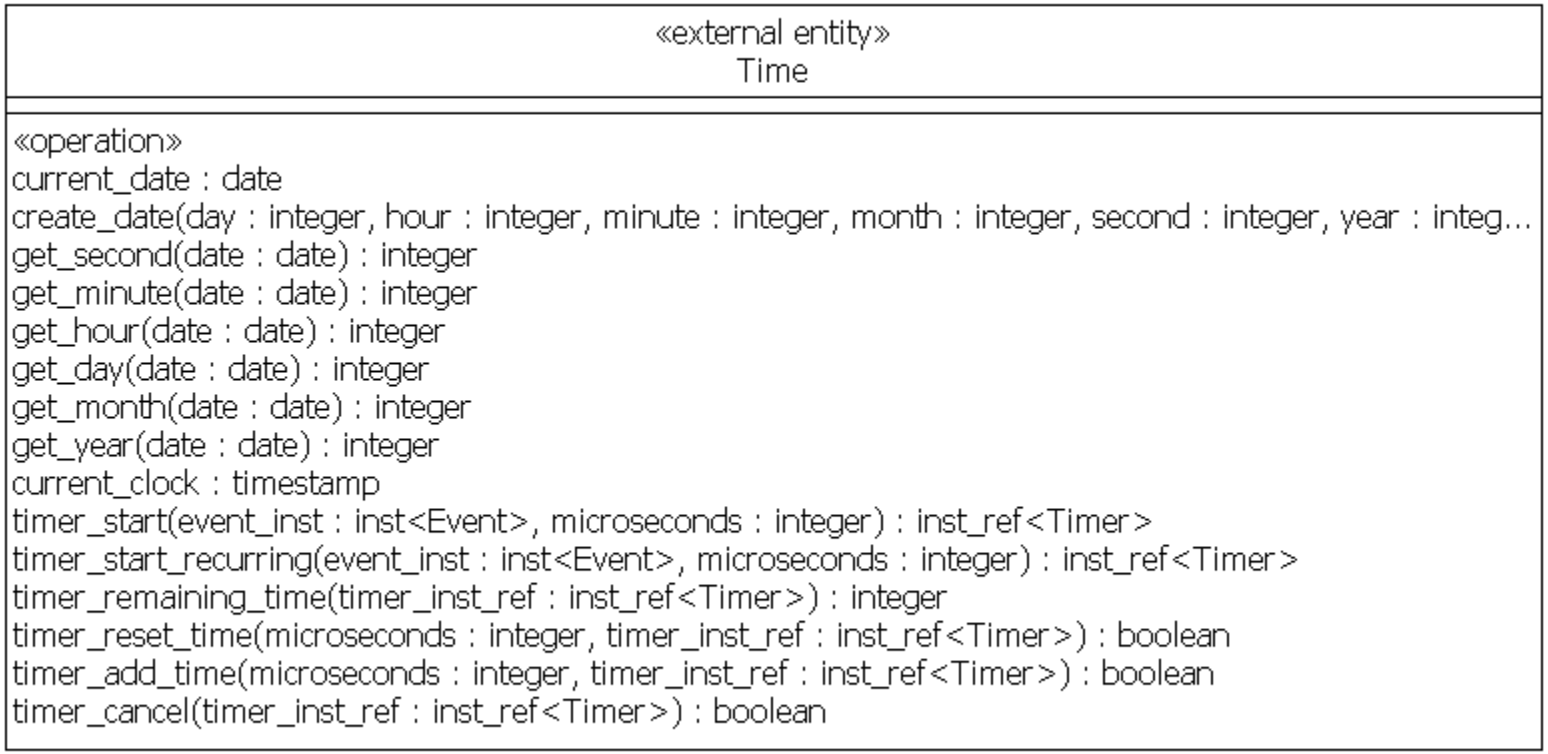
Event Label

- **Sending a Pre-Created Event.**

```
generate event_inst ;
```

Delayed event

```
create event instance event of MO_O8(period: 60) to oven;

handle = TIM::timer_start(event_inst: event, microseconds: 1000000);
```



Interface messages

```
// asynchronous interface message - Interface signal
send LocationProvider::registerListener(interval: 1000);
```

Interface or port name

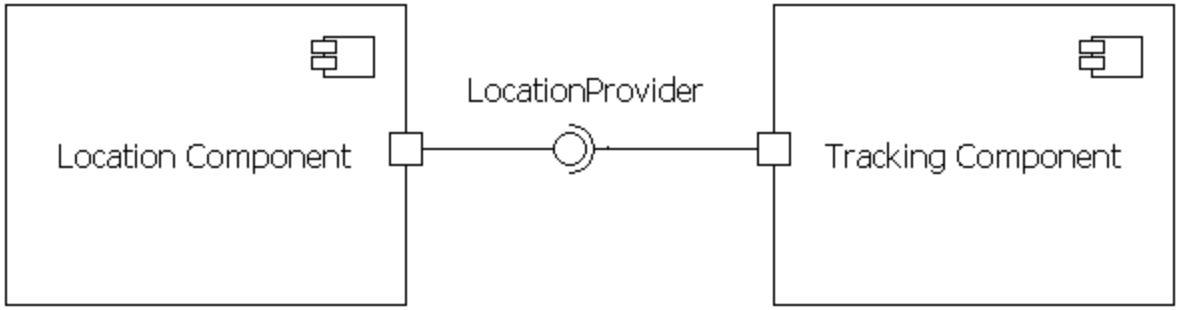
Signal name

Signal parameters

```
// synchronous interface message - Interface operation
send dist = LocationProvider::distance(from: loc1, to:
loc2);
```

Local variable

Operation name



Mathematically Derived Attributes

- Attributes can be computed using the full range of Action Language statements
- No return statement needed
- Read-only in all other places