Extending UML with time: a concrete framework

Susanne Graf
Ileana Ober, Iulian Ober

VERIMAG, Grenoble
http://www-verimag.imag.fr

Motivation: analysis of the constraint

Time constraint:

Between

the **moment** *an Engine initiates a show on its screen*

and

the **moment** *the same Engine updates the information (calls updateInfo) on its screen*

less than 10 time units pass

if *i+k has not changed.*

Recurrent time related elements:

« moment » (instant = event)

« duration » (time elapsed between instants)

Extending UML with time

Untimed UML model (with state machines):

- A set of behaviors, restricting the possible ***orders*** of occurrences of events

  (events = interactions between objects)

Time extended UML model:

- A set of timed behaviors, restricting the possible ***order*** and ***occurrence times*** of occurrences of events

  (events = identifiable with an instant in time)

  1. Define constraints on occurrence times of events
  2. Define constraints on durations between occurrence times of events

➔ **How to express time constraints ?**

Extending UML with time

Proposal 1: extend state machines with clocks (measuring durations) which can be started at certain points and tested later

➔ convenient for the operational specification of *time dependent* behaviors

Problems:

- allows to define the time at which events ***can*** occur, not when they ***must*** occur
- not all relevant events are accessible in state machines, due to implicit events and objects
- what if there are objects without state machines ?

➔ **use this kind of time extension, but not alone**

Extending UML with time

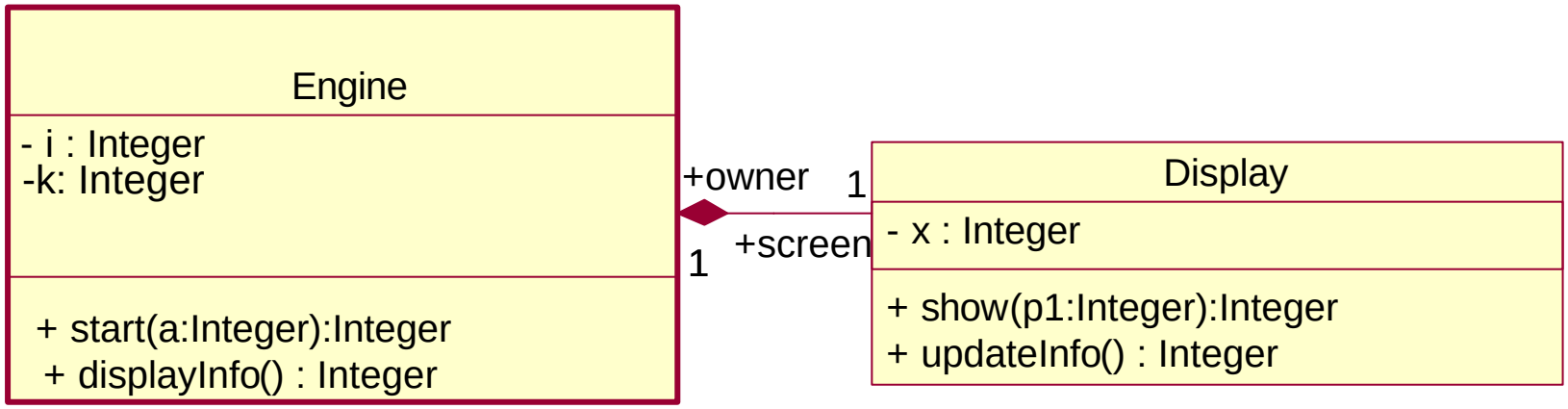
Proposal 2: real-time characteristics orthogonal to the functional behavior

- provide a syntax allowing to identify all state changes in the underlying semantic model (events)
- express timing by constraints on *durations between such events*, representing invariants of the system
- define patterns for certain event pairs, frequently under some time constraint (execution time of actions, response time to requests, transmission delays of channels, …)

Time profile

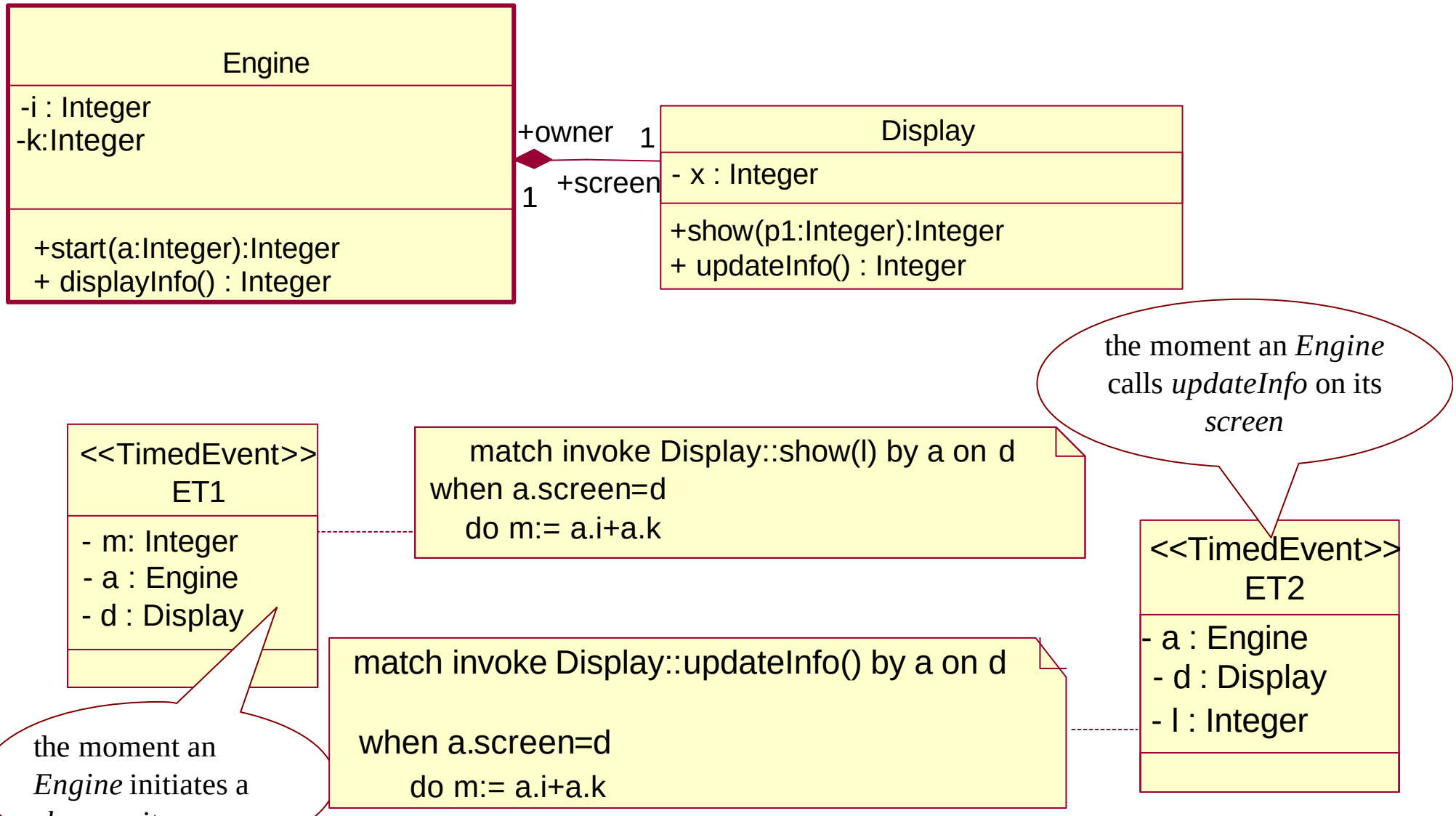
- **Basics**
  - A notion of *global time, external* to the system
  - Time primitive types: Time, Duration with operations
  - Events: history of occurrence times of identified state changes
- **operational time access: *time dependent behaviour***
  - Expression *now* for accessing global time
  - Guards on durations
  - Mechanisms for measuring durations: timers, clocks
- **Time constraints: *orthogonal to functional aspects***
  - Constraints on durations
    - Assumptions (taken as given)
    - Requirements (to be verified)

Motivation: model + constraint



Time constraint:
Between the *moment an Engine initiates a show on its screen* and the *moment the same Engine updates the information (calls updateInfo) on its screen* less than 10 time units pass, if the sum i+k has not changed.

1. Timed events (example)



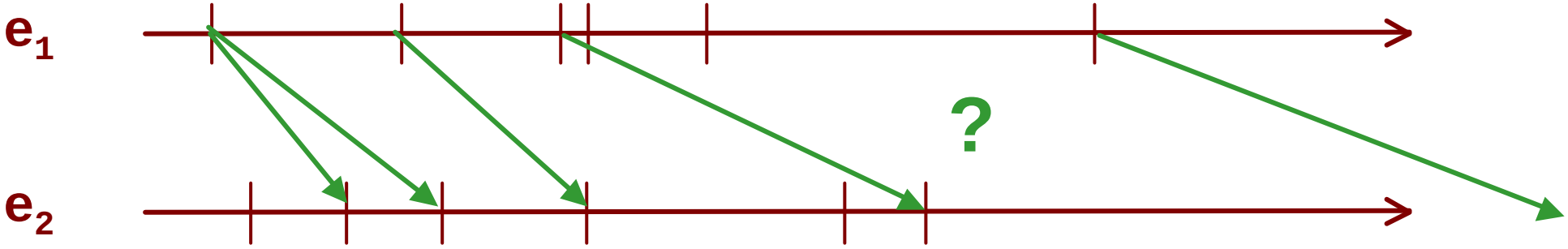
Event type, event, event occurrence

We distinguish between:

- Event type: pattern of event
  - says how it is identified
    - event kinds
    - matching conditions
  - has local memory
- Event type instance (object attributes): history of event occurrences
- Event occurrence: the actual run-time occurrence of some event

2. Durations

Duration: time elapsed between instants of two event occurrences (particular case: time elapsed since some event)



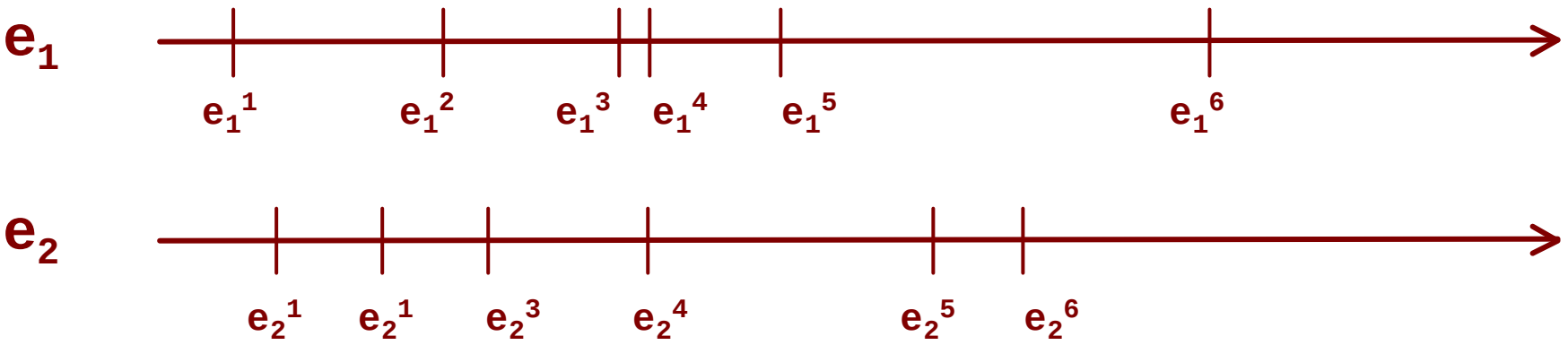
Define a syntactic expression: *duration*(e_1, e_2)

- *Which occurrences to identify?*
- *The causally related ones ?*
- *Causal relationship needs to be specified explicitly*

Problem: find a mechanism to identify

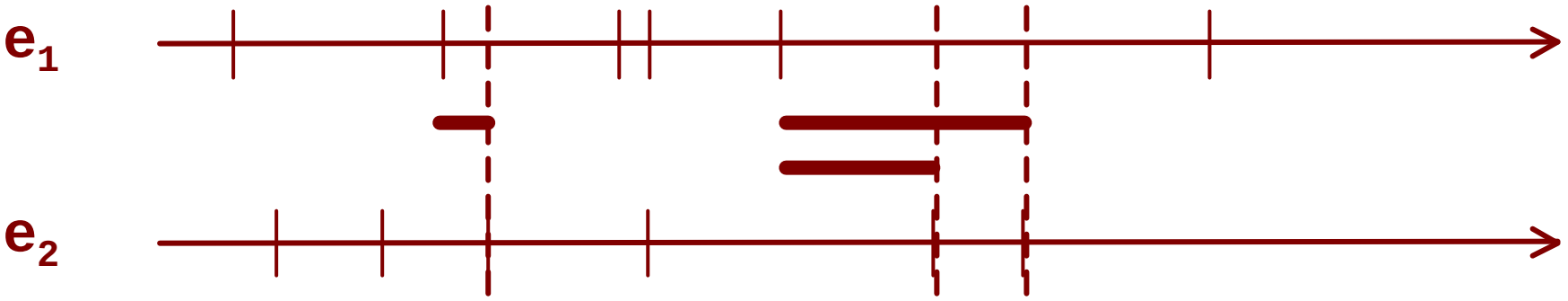
matching event occurrence pairs

Matching event occurrences



Solution 1: use index of event occurrences to identify matching pairs

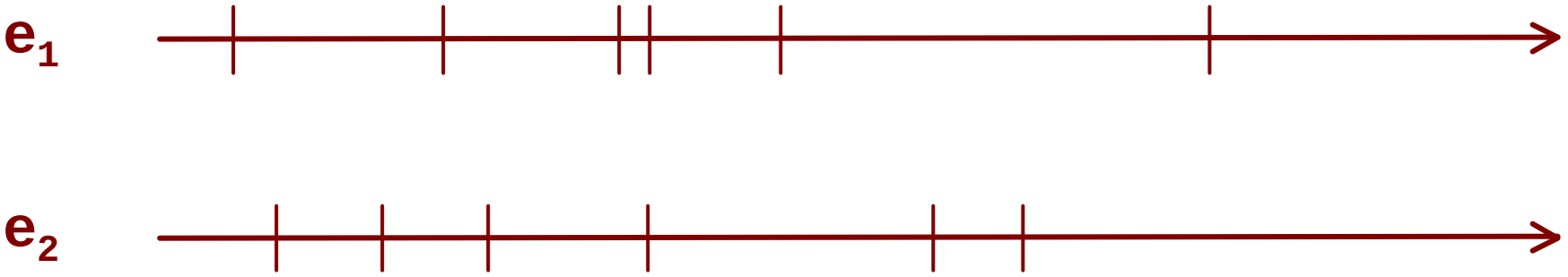
Matching event occurrences



Solution 1: use index of event occurrences to identify matching pairs

Solution 2: match the most recent occurrences looking backward from e2 (use filter conditions to identify the right occurrences)

Matching event occurrences



Solution 1: use index of event occurrences to identify matching pairs

Solution 2: match the most recent occurrences looking backward from e2 (use filter conditions to identify the right occurrences)

Solution 3: define useful patterns

3. Constraints

1. OCL allows to express constraints in the form of *invariants*

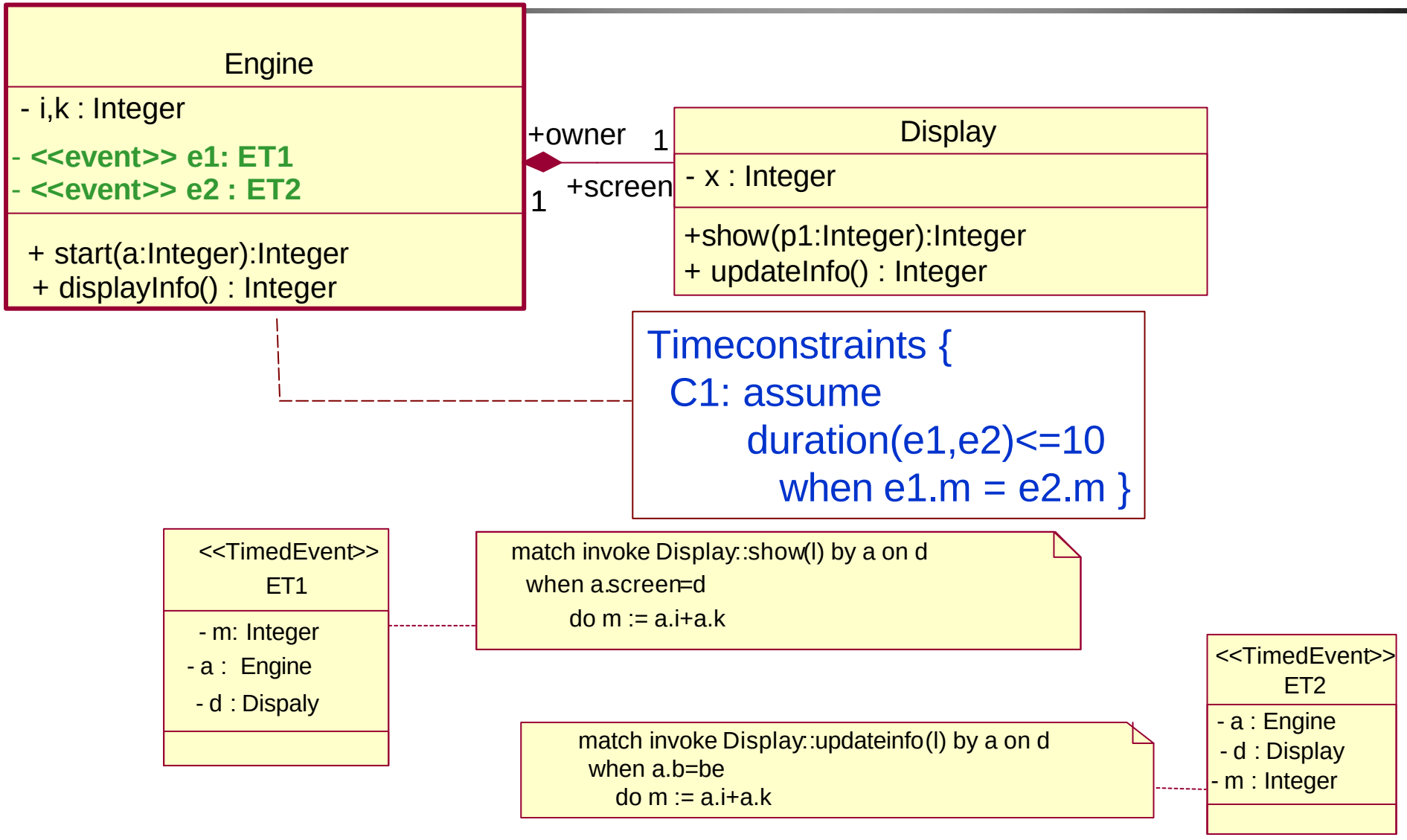
- Constraint = requirement to be checked: any invariant on durations
- Constraint = assumption: each constraint must constrain the occurrence of a well identified event (executable model)

duration(e1,e2) £ 10 when cond(attr(e1),attr(e2))

2. General history dependent time constraints

- Increase the expressive power of OCL with event histories
- Use state machines triggered by “timed events” to define history dependent constraints

3. Constraints (an example)



Duration patterns

- **Define patterns associated with syntactic features, identifying particular pairs of occurrences between particular events**
  - ResponseTime (caller / callee)
  - TransmissionDelay (channel)
  - ExecutionTIme (action)
  - InterarrivalTime (signal /call)
  - TimeInState (state machine state)
  - ReactionTime ….

➔ More useful patterns have to be identified

Resources and Scheduling

What is needed?

- **A notion of «resource» with an attribute defining its «preemptibility»**
- **Distinction of execution time and duration**
- **A means to identify «tasks» which might be atomic or not and need resources:**
  1. Use methods to identify tasks and associate with them used resources, atomicity, execution time and deadline
  2. Define tasks and their properties dynamically in state machines by scheduling related method calls, such as «needs(resource)», «startatomic», …
- **A means to define scheduling policies: priority rules**
  1. Priorities attached to methods or objects
  2. Dynamic priorities

Summary

- **Time profile integrated in UML syntax**
  - Basics
    - A notion of *global external time*
    - Time primitive types: Time, Duration with operations
    - Events: history of occurrence times of identified state changes
  - Imperative time access: *time dependent behavior*
    - Expression *now* for accessing global time
    - Guards on durations
    - Mechanisms for measuring durations: timers, clocks ??
  - Time constraints: *orthogonal to functional aspects*
    - Constraints on durations
      - Assumptions (taken as given)
      - Requirements (to be verified)
- **Well-defined semantics in terms of timed automata**
- **(Partially) implemented in a tool**