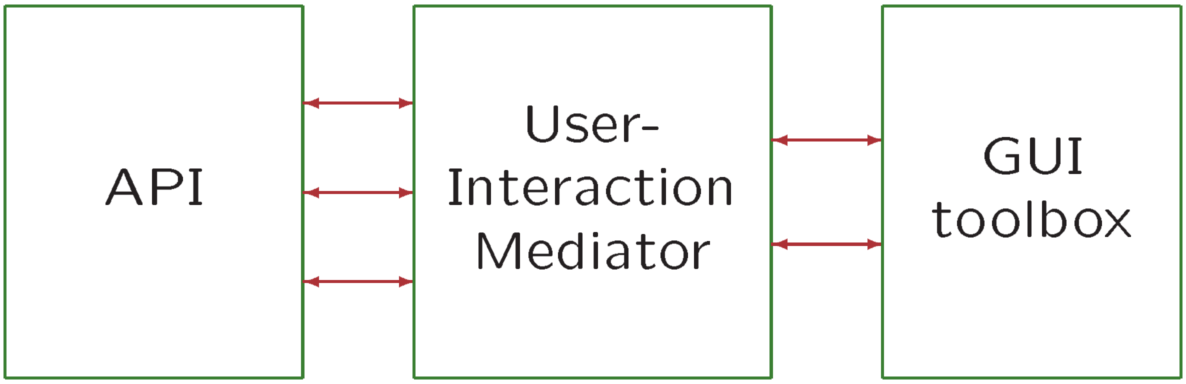
UserInteractionWare

Lambert Meertens



The conceptual position of
the *User-Interaction Mediator*
(UIM)

How to specify the UIM

Derive the mapping between GUI and application from:

- a generic *theory* of user interaction
- a *description* of the application functionality in terms of that theory
- a *refinement* of the *interactor subtheory* to GUI capabilities

Current practice

The design is dominated by focus on low-level GUI capabilities (menus, buttons, dialog boxes, colors).

The user-interaction paradigm is not a conscious part of the design.

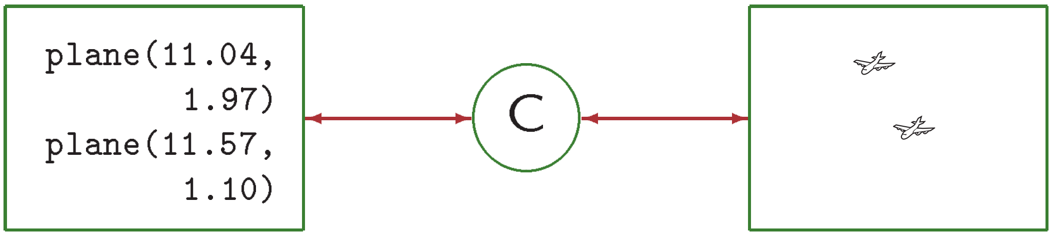
The result is hard to change.

Model-View-Controller (MVC)

The *model* is formed by “abstract” data.

The *view* is a visual presentation of the data.

The *controller* is a process maintaining the correspondence between model and view.



The MVC Loop

The user “edits” (interacts with) the view.

The controller adjusts the model.

The *application* reacts and further changes the model.

The controller adjusts the view.

States and events

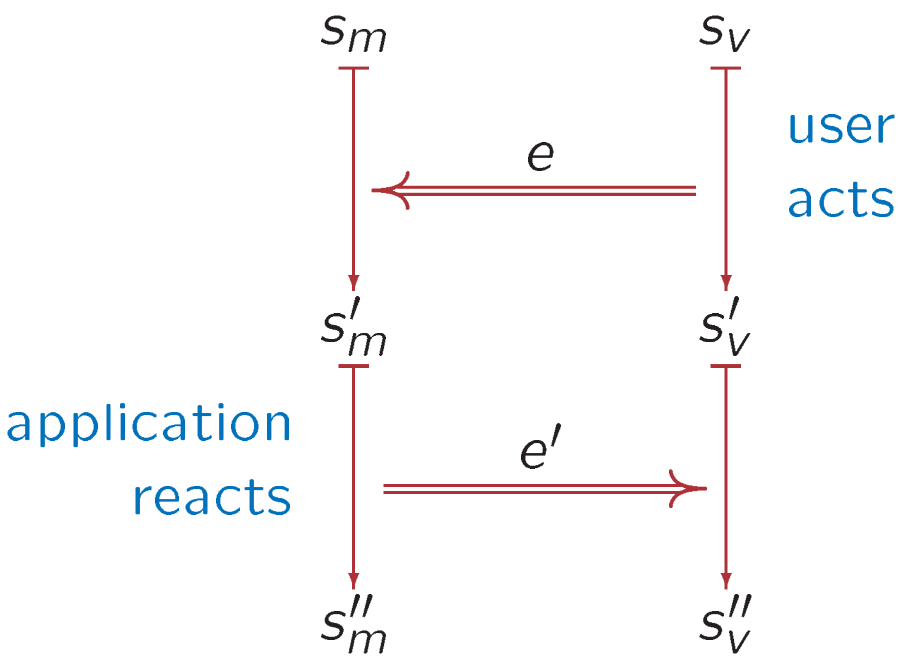
Assume a *state space* Σ, and an *event space* (or *edit-action space*) E.

The *semantics* of events is a mapping

$$\mathcal{M} : E \rightarrow (\Sigma \rightarrow \Sigma)$$

We must define this *twice*: once for the models and once for the views. The event space, however, is *shared* between the two sides.

Event passing



At the model side

Specify functions

$$esem : \ \Sigma \times E \ \rightarrow \ \Sigma$$

$$react : \ \Sigma \ \rightarrow \ E$$

where $esem(s, e) \ = \ \mathcal{M}(e)(s)$

Reacting to a stream

$$sreact : \Sigma \rightarrow E^{\omega} \rightarrow E^{\omega}$$

$$\begin{aligned} sreact\ s\ (e : es) \ &= \ e' : sreact\ s''\ es \\ \text{where} \quad s' &= esem\ s\ e \\ e' &= react\ s' \\ s'' &= esem\ s'\ e' \end{aligned}$$

Approach

Specify the application side in SLANG.

Write the view side directly in some conventional programming language until we have enough experience to know what we need.

Start with simple problems; gradually expand to more ambitious problems.

Some tasks

- experiment with simple *textual* reactive system
- test various ways of modelling reactivity
- design a GUI library at a good level of abstract
- perform trials with GUI reactivity
- redo generic parts of view side
  as much as possible using Specware