Modeling of Cooperative Editors Using CO-OPN/2

Olivier Biberstein
Centre Universitaire d'Informatique
University of Geneva
Switzerland

Plan

- Objectives of CO-OPN/2
- Major features of CO-OPN/2
- Proposed case study
- Cooperative diagram editors
- Modeling structure
- CO-OPN/2 specification
- Conclusion

Objectives of CO-OPN/2

- Propose a concurrent object-oriented formalism
  - ❒ data structures
  - ❒ operational behavior
  - ❒ true concurrency
- Introduce object-orientation into its object-based predecessor
- Provide a formal semantics
- Provide a concrete language and a development environment

Features of CO-OPN/2

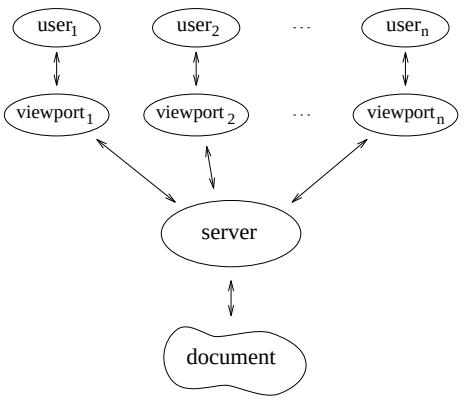
1. Data structures
2. Notion of object and class
3. Object identity
4. Interaction between the objects
5. Concurrency
6. Meaning of inheritance and sub-typing
7. "Objects inside objects"

Proposed case study

- Cooperative editor of hierarchical diagrams
  - simultaneous edition of one document by several users
  - dispatching of the graphical modifications "in real time"
  - concurrent accesses to the document
- Simple coordination protocol
  - Users join/leave with different privileges (edit, view)
  - Graphical elements may be owned by users for ownerships (deletion, encapsulation, modification, inspection)
  - Only encapsulation can persist between the sessions

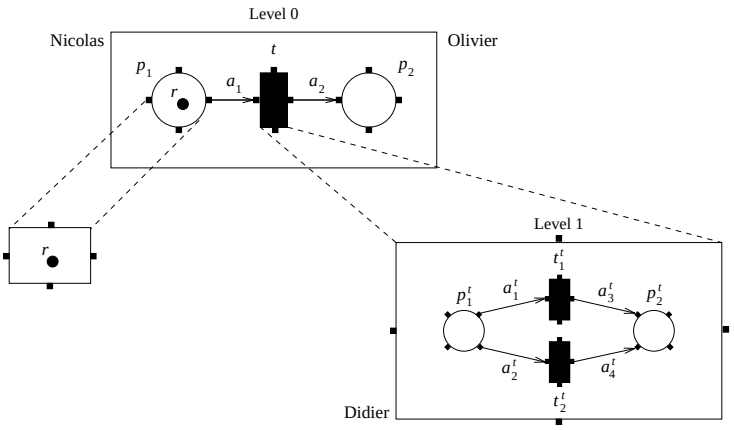
Cooperative Diagram Editors (CDE)

Structure of a CDE with a centralized architecture



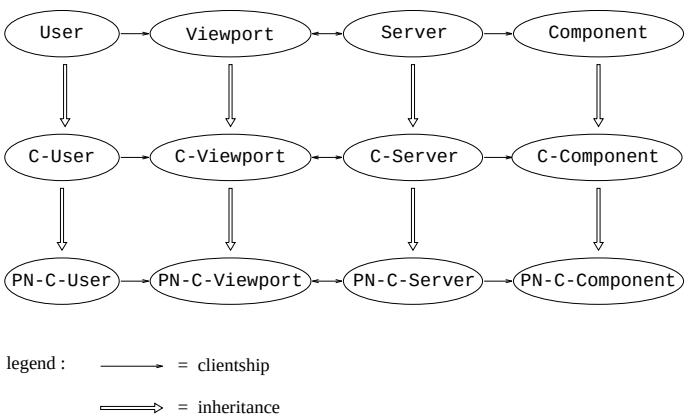
Cooperative Hierarchical Petri Net Editor

An example of an editing session



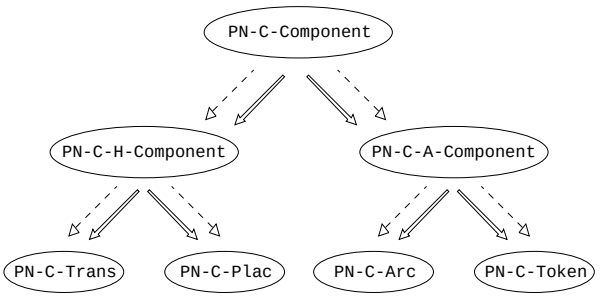
Modeling Structure

Three levels of entities



Modeling Structure (cont'd)

Sub-typing hierarchy between the components



Class of viewports

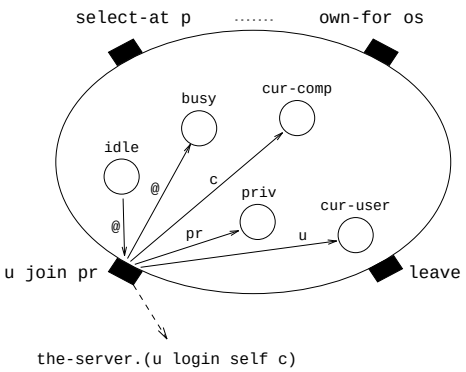
```
Class PN-C-Viewport;
Interface
  Use  Coord, Picture, Privilege,
       Ownership,
       PN-C-User, PN-C-Component;
  Type pn-c-viewport;
  Methods
    _ join _ : pn-c-user privilege;
    leave;
    select-at _    : coord;
    move _         : coord;
    own-for _      : ownership;
    display _      : picture;
    new-trans-at _ : coord;
;; ... and more ...
Body
  Use Unique, Set-of-pn-c-component,
      Server;
  Places
    busy _     : unique;
    idle _     : unique;
    priv _     : privilege;
    cur-user _ : pn-c-user;
    cur-comp _ : pn-c-component;
    selected _ : set-of-pn-c-component;
  Initial
    idle @, selected [];
  Axioms
  u join pr
    with the-server.(u login self c) :
    idle @
    ->
    busy @, priv pr, cur-comp c, cur-user u;

  select-at p
    with the-server.(u select c at p for pr):
    busy @, cur-user u, priv pr, selected compnts
    ->
    busy @, cur-user u, priv pr, selected compnts+c;

  move p
    with the-server.(u move c at p for pr) :
    busy @, cur-user u, priv pr, selected []+c
    ->
    busy @, cur-user u, priv pr, selected [];

  display pic : -> ;
;; ... and more ...
  Where
    p : coord;   pr : privilege;   pic : picture;
    c : pn-c-component;
    u : pn-c-user;
End PN-C-Viewport;
```

Class of viewports (cont'd)

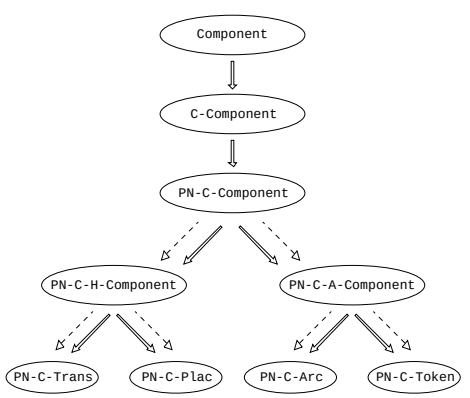


Graphical Components

```
move      _ : coord;
resize    _ : size;
rotate    _ : angle;
put-label _ : string;
get-pos   _ : coord;


_ own-for _         : c-user ownership;
_ authorized-for? _ : c-user ownership;
surrender;


add-child _      : pn-c-component;
del-child _      : pn-c-component;
get-child _ at _ : pn-c-component coord;
get-parent _     : pn-c-component;
delete;
ret-expanded-picture _ : picture;
```

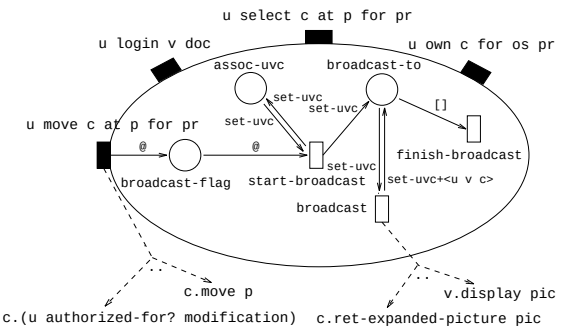


Graphical Components (cont'd)

```
Class PN-C-Trans;
Inherit PN-C-H-Component;
  Rename pn-c-h-component -> pn-c-trans;
Interface
  Subtype pn-c-trans < pn-c-h-component;
  Method
    ret-picture _ : picture;
  Creation
    create-trans-at _ : coord;
Body
  Initial
    dimension <40 100>;
    rotation 0;
    anchors []+<0 50>+<20 100>+<40 50>+<20 0>;
    links [];
  Axiom
    create-trans-at p : -> position p;

    ret-picture pic : -> ;
  Where
    p : coord;
    pic : picture;
End PN-C-Trans;
```


Cooperative Server



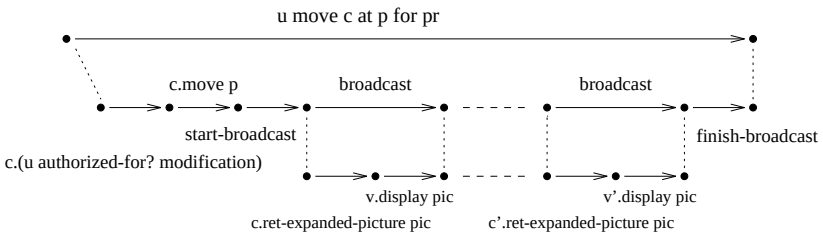

```
u move c at p for pr with c.(u authorized-for? modification) .. c.move p :
  pr = edit => -> broadcast-flag @;

start-broadcast :
  broadcast-flag @, assoc-uvc set-uvc -> assoc-uvc set-uvc, broadcast-to set-uvc;

broadcast with c.ret-expanded-picture pic .. v.display pic :
  broadcast-to (set-uvc + <u v c>) -> broadcast-to set-ucv;

finish-broadcast : broadcast-to [] -> ;
```

Cooperative Server (cont'd)




```
u move c at p for pr with c.(u authorized-for? modification) .. c.move p
start-broadcast
broadcast with c.ret-expanded-picture pic .. v.display pic
finish-broadcast
```


Conclusion

- CO-OPN/2: concurrent object-oriented formalism
- We gave an modeling of a case study on groupware
- We shown how CO-OPN/2 may be used
  - Inheritance ⇒ flexible structure of class
  - Sub-typing ⇒ consistent and regular treatment

Future Work

- Observational sub-typing
- Development environment and tools
- Eurécom, Sophia Antipolis, France
- Test, incremental implementation, coordination