Programming or Flexibility?

Design of Programmable Applications with Biologists

Catherine Letondal

`letondal@pasteur.fr`

In Situ, LRI, Institut Pasteur

In Situ

- New interaction paradigms:
  - multi-scale (or zoomable) interfaces,
  - interactive information visualization,
  - bimanual interaction,
  - video and non-speech audio,
  - augmented or mixed reality.
- Participatory design
  - development of new participatory design methods,
  - make the role of context explicit in the design process.
- Engineering of interactive systems
  - component-based architectures (customizable and dynamic addition or substitution of interaction techniques).

Programming situations

- scripting: search for a sequence pattern, *then* retrieve all the corresponding secondary structures in a database (example)

Programming situations

- scripting: search for a sequence pattern, *then* retrieve all the corresponding secondary structures in a database (example)
- parsing: search for the best match in a database similarity search report *but relative to each subsection*

Programming situations

- scripting: search for a sequence pattern, *then* retrieve all the corresponding secondary structures in a database (example)
- parsing: search for the best match in a database similarity search report *but relative to each subsection*
- formatting: renumber one's sequence positions from -3000 to +500 instead of 0 to 3500

Programming situations

- scripting: search for a sequence pattern, *then* retrieve all the corresponding secondary structures in a database (example)
- parsing: search for the best match in a database similarity search report *but relative to each subsection*
- formatting: renumber one's sequence positions from -3000 to +500 instead of 0 to 3500
- variation: search for patterns in a sequence, *except repeated ones*

Programming situations

- scripting: search for a sequence pattern, *then* retrieve all the corresponding secondary structures in a database (example)
- parsing: search for the best match in a database similarity search report *but relative to each subsection*
- formatting: renumber one's sequence positions from -3000 to +500 instead of 0 to 3500
- variation: search for patterns in a sequence, *except repeated ones*
- finer control on the computation: control in what order multiple sequences are compared and aligned

Programming situations

- scripting: search for a sequence pattern, *then* retrieve all the corresponding secondary structures in a database (example)
- parsing: search for the best match in a database similarity search report *but relative to each subsection*
- formatting: renumber one's sequence positions from -3000 to +500 instead of 0 to 3500
- variation: search for patterns in a sequence, *except repeated ones*
- finer control on the computation: control in what order multiple sequences are compared and aligned
- simple operations: search in a DNA sequence for the characters other than A, C, T and G

Parsing and scripting example

This example shows the report of a program searching for registered well-known patterns in a given protein sequence.

```
Access#      From->To                     Name
_______      ________                     ____
PS00005        39->41            PKC_PHOSPHO_SITE
        Pattern [ST].[RK] matched
        Site    39 TTR 41
PS00005        71->73            PKC_PHOSPHO_SITE
        Pattern [ST].[RK] matched
        Site    71 TSK 73
PS00008        20->25                    MYRISTYL
        Pattern G[^EDRKHPFYW]..[STAGCN][^P] mat
        Site    20 GILAAI 25
```

Problem description

- more and more professional bio-informaticists, still a minority
- the vast majority does not program

Problem description

- more and more professional bio-informaticists, still a minority
- the vast majority does not program
- many biologists *are able to program* and have learned programming
- many of them do not program at all *although they would need it*

Problem description

- more and more professional bio-informaticists, still a minority
- the vast majority does not program
- many biologists *are able to program* and have learned programming
- many of them do not program at all *although they would need it*

Why?

Problem description

- more and more professional bio-informaticists, still a minority
- the vast majority does not program
- many biologists *are able to program* and have learned programming
- many of them do not program at all *although they would need it*

$$\boxed{\boxed{\text{It's too difficult to program } \textit{a little}.}}$$

Approaches

- enable Programming In The User Interface (*programmability*)

Programming In The User Interface

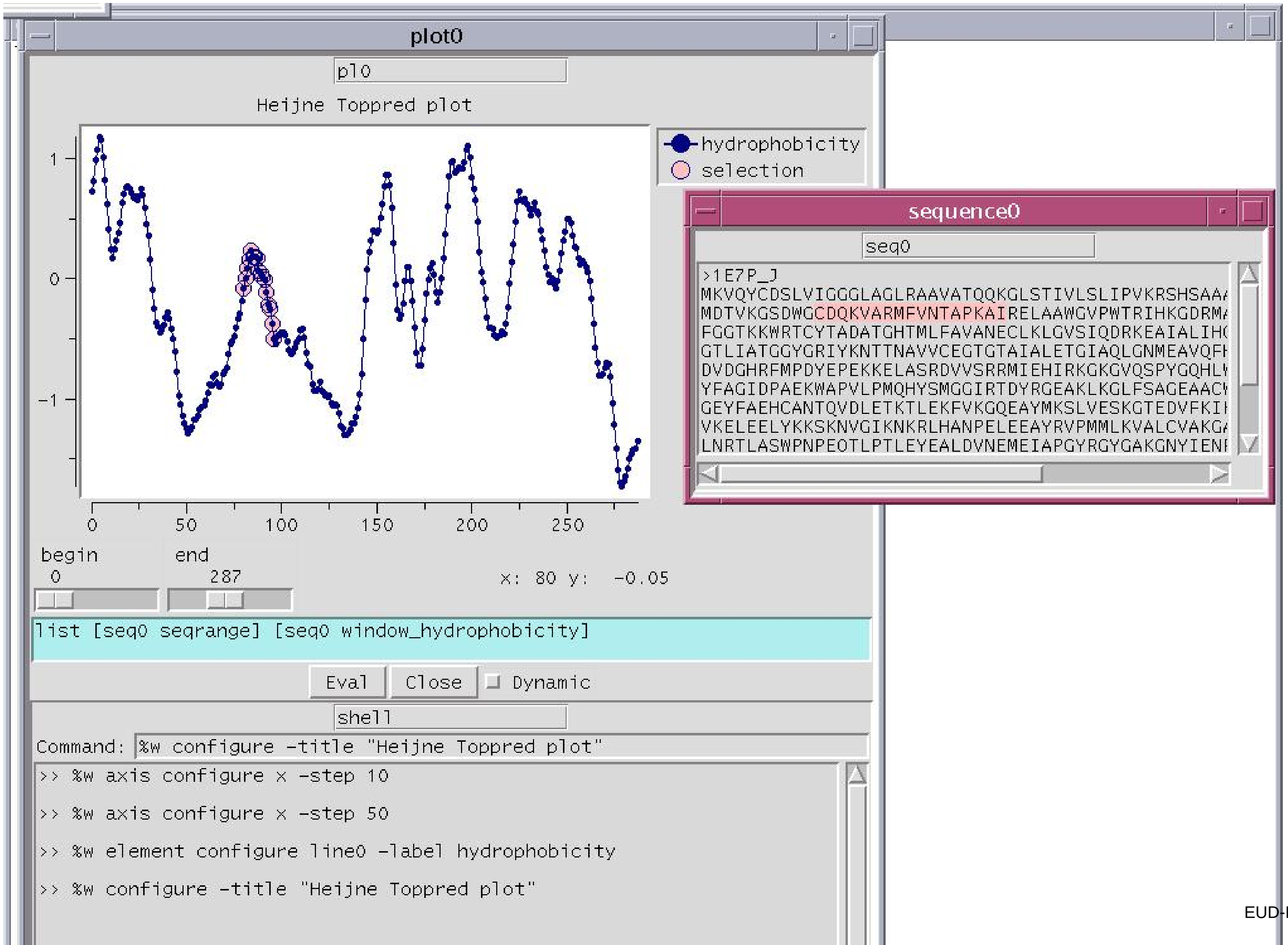
What we want:

working environment

=

programming environment

biok: graphical objects

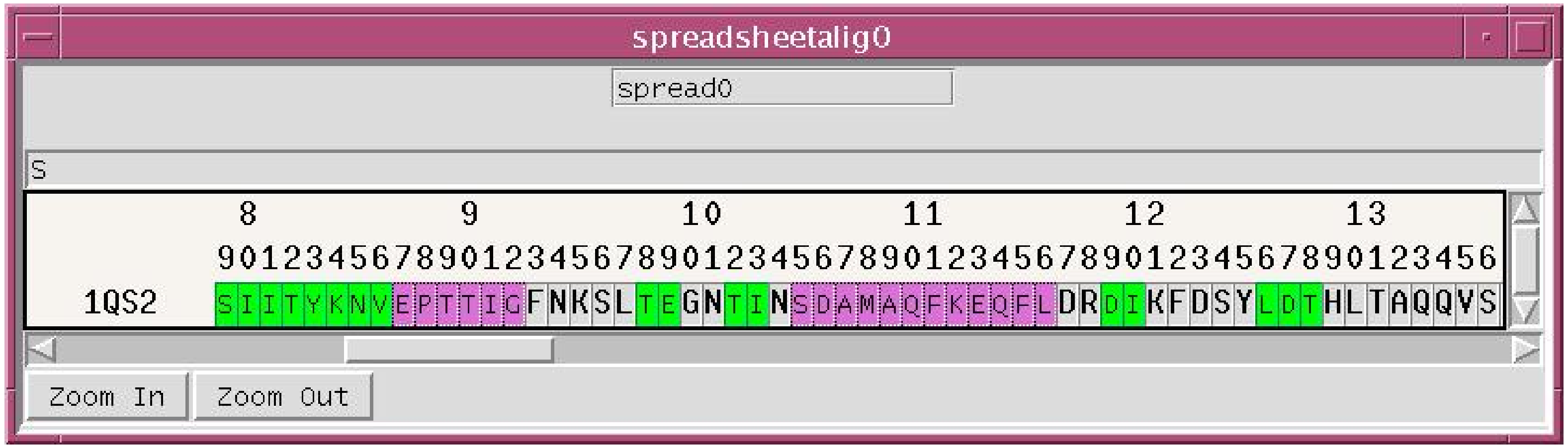


```
>1E7P_J
MKVQYCDSLVIGGGLAGLRAAVATQQKGLSTIVLSLIPVKRSHSAA
MDTVKGSDWGCDQKVARMFVNTAPKAIRELAAWGVPWTRIHKGDRM
FGGTKKWRTCYTADATGHTMLFAVANECLKLGVSIQDRKEAIALIH
GTLIATGGYGRIYKNTTNAVVCEGTGTAIALETGIAQLGNMEAVQF
DVDGHRFMPDYEPEKKELASRDVVSRRMIEHIRKGKGVQSPYGQHL
YFAGIDPAEKWAPVLPMQHYSMGGIRTDYRGEAKLKGLFSAGEAAC
GEYFAEHCANTQVDLETKTLEKFVKGQEAYMKSLVESKGTEDVFKI
VKELEELYKKSKNVGIKNKRLHANPELEEAYRVPMMLKVALCVAKG
LNRTLASWPNPEOTLPTLEYEALDVNEMEIAPGYRGYGAKGNYIEN
```

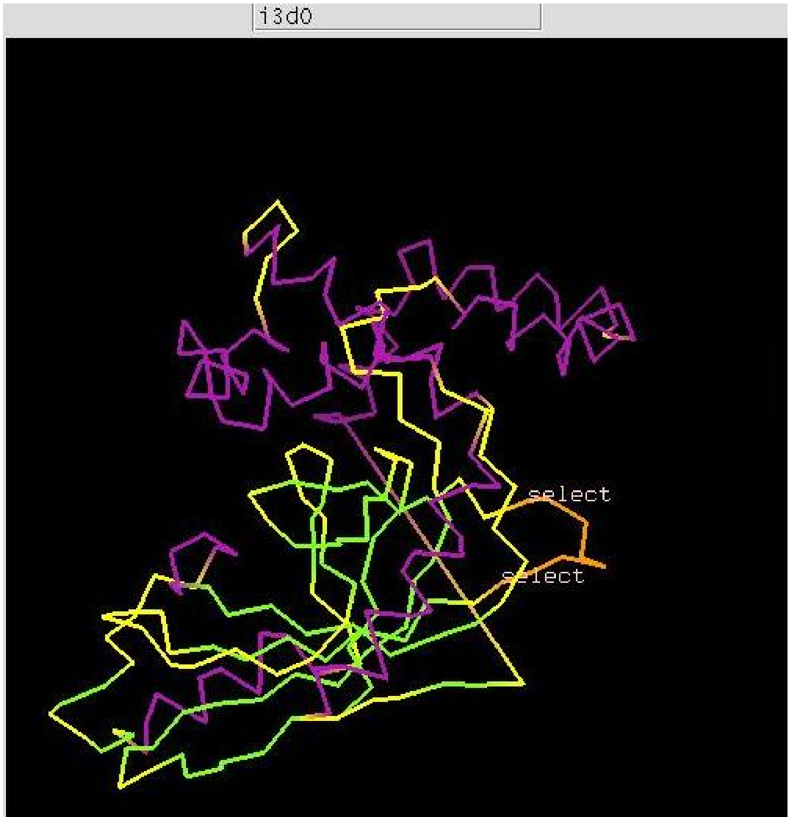
```
list [seq0 seqrange] [seq0 window_hydrophobicity]
```

```
Command: %w configure -title "Heijne Toppred plot"
>> %w axis configure x -step 10
>> %w axis configure x -step 50
>> %w element configure line0 -label hydrophobicity
>> %w configure -title "Heijne Toppred plot"
```

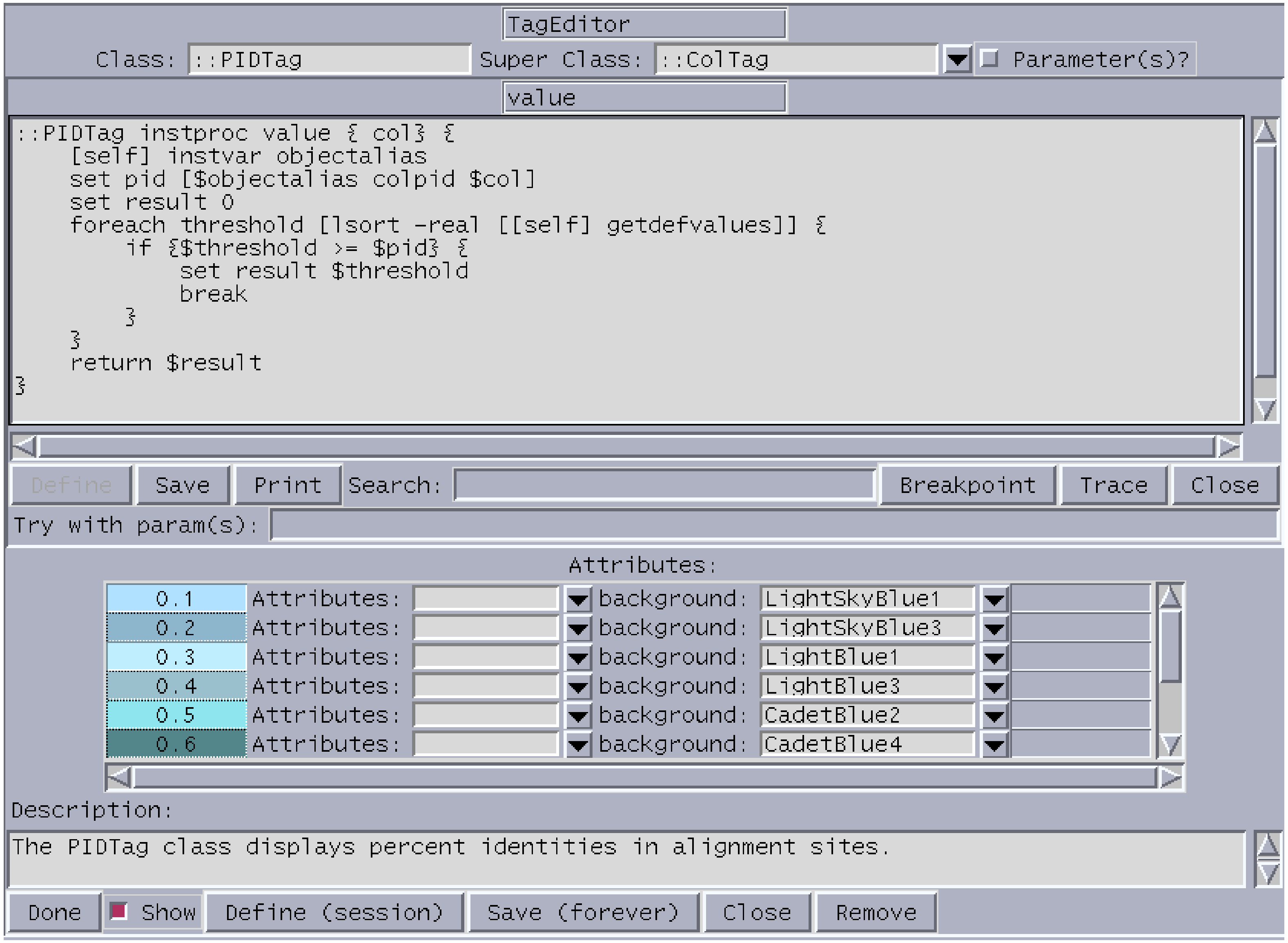
biok: tags



```
              8         9         10        11        12        13
         9012345678901234567890123456789012345678901234567890123456
   1QS2  SIITYKNVEPTTIGFNKSLTEGNTINSDAMAQFKEQFLDRDIKFDSYLDTHLTAQQVS
```



biok: programming tags



Class: ::PIDTag Super Class: ::ColTag ☐ Parameter(s)?

value

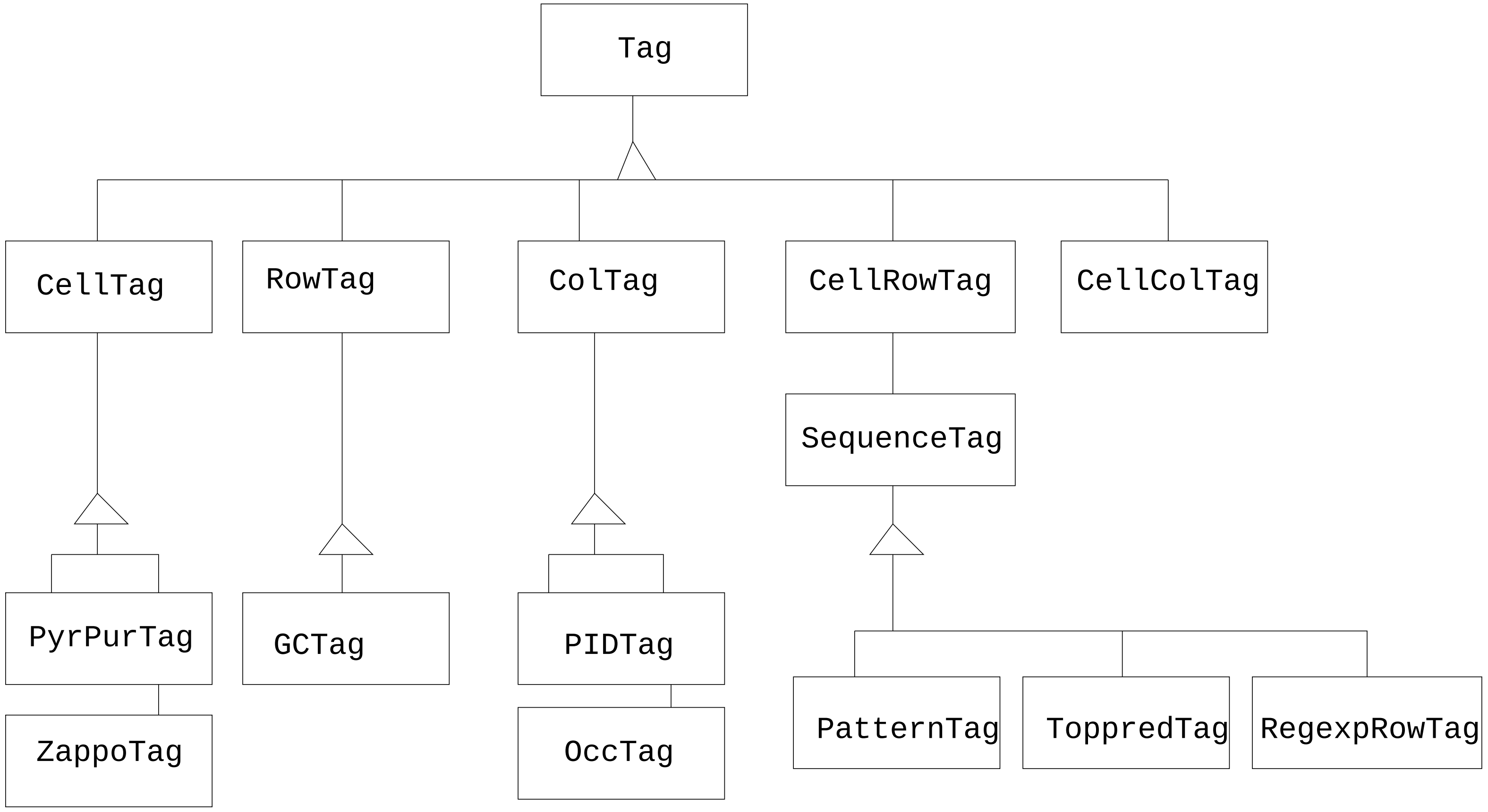
```
::PIDTag instproc value { col} {
    [self] instvar objectalias
    set pid [$objectalias colpid $col]
    set result 0
    foreach threshold [lsort -real [[self] getdefvalues]] {
        if {$threshold >= $pid} {
            set result $threshold
            break
        }
    }
    return $result
}
```

0.1	Attributes:		background:	LightSkyBlue1
0.2	Attributes:		background:	LightSkyBlue3
0.3	Attributes:		background:	LightBlue1
0.4	Attributes:		background:	LightBlue3
0.5	Attributes:		background:	CadetBlue2
0.6	Attributes:		background:	CadetBlue4

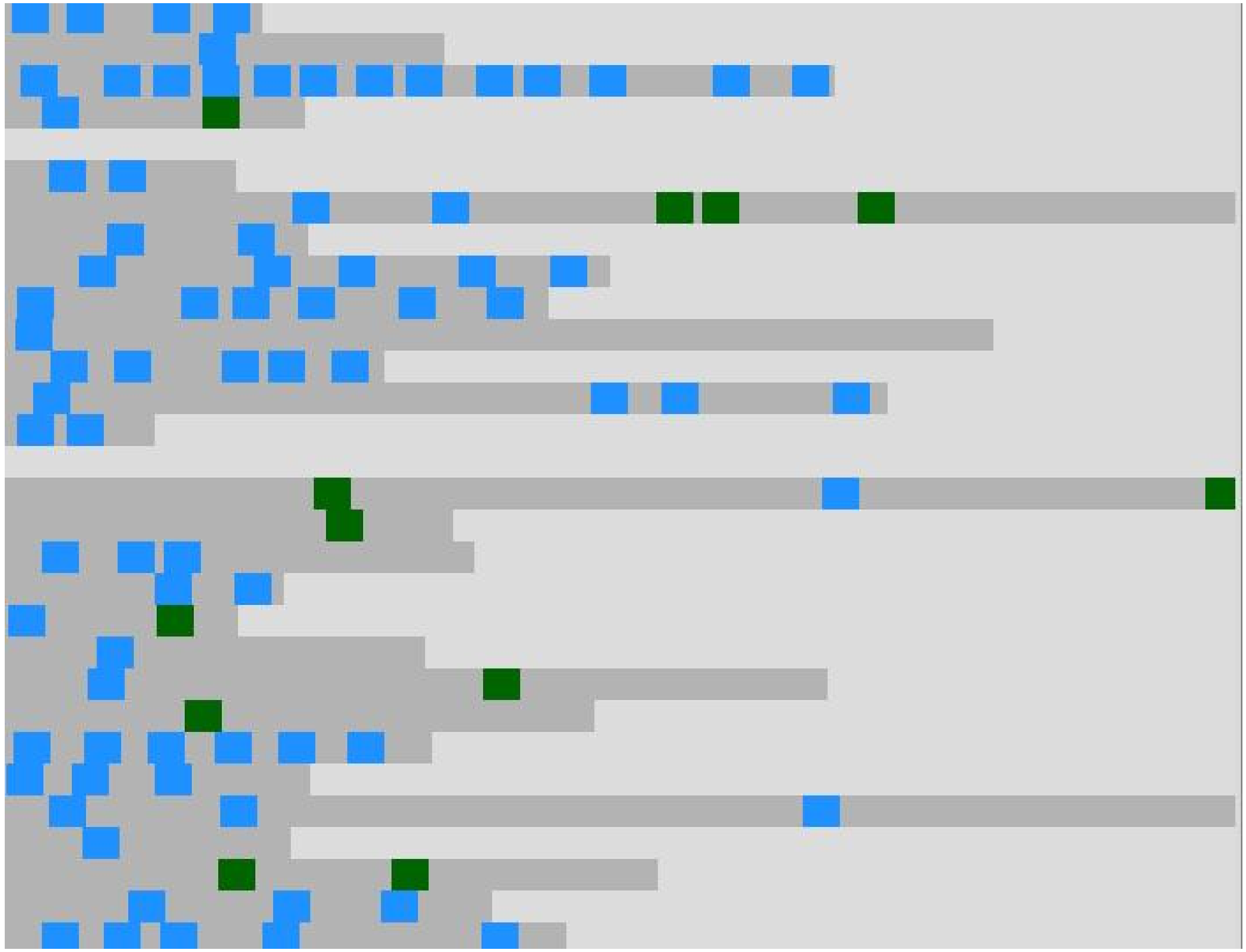
Description:

The PIDTag class displays percent identities in alignment sites.

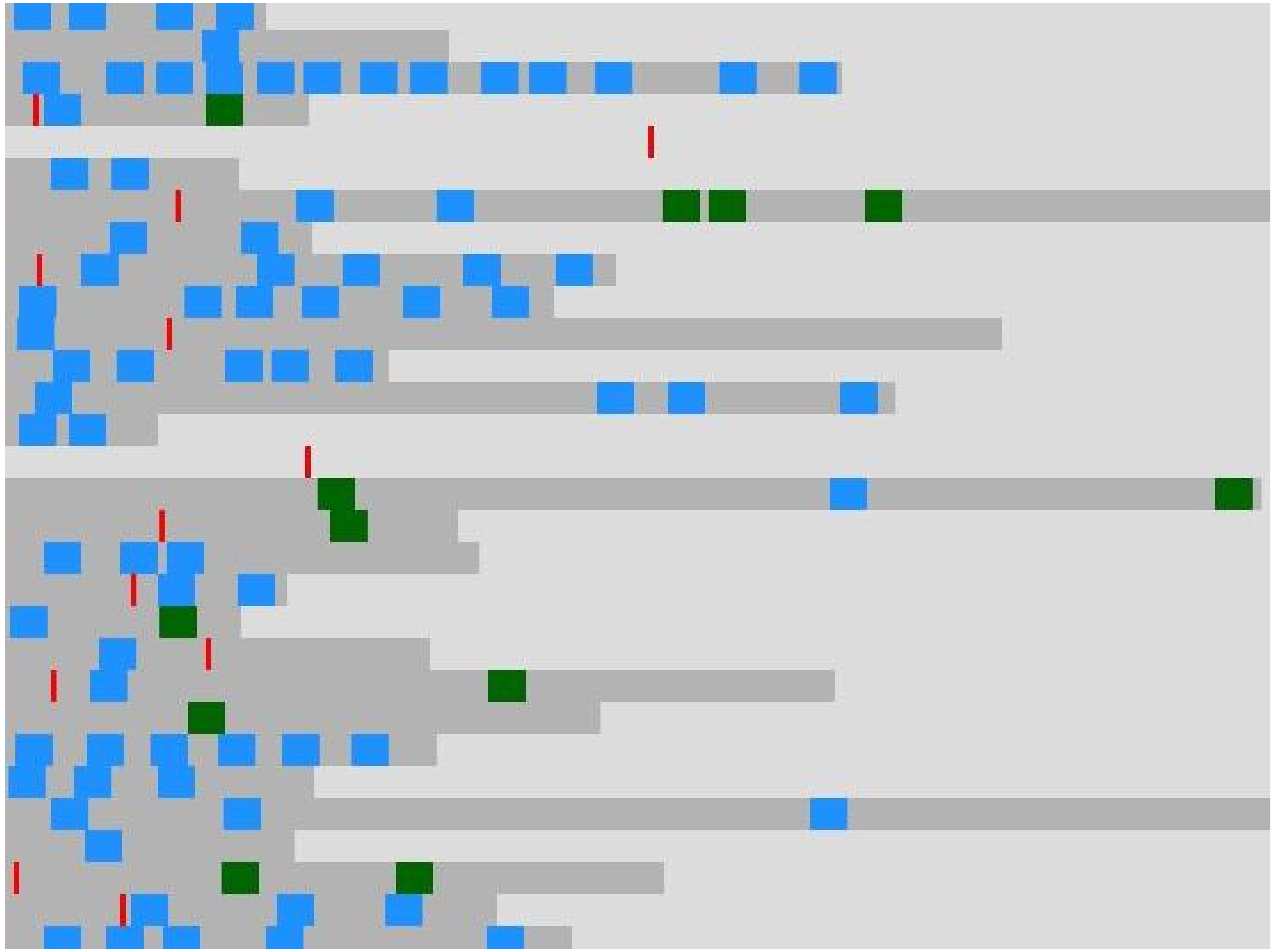
biok: tags classes



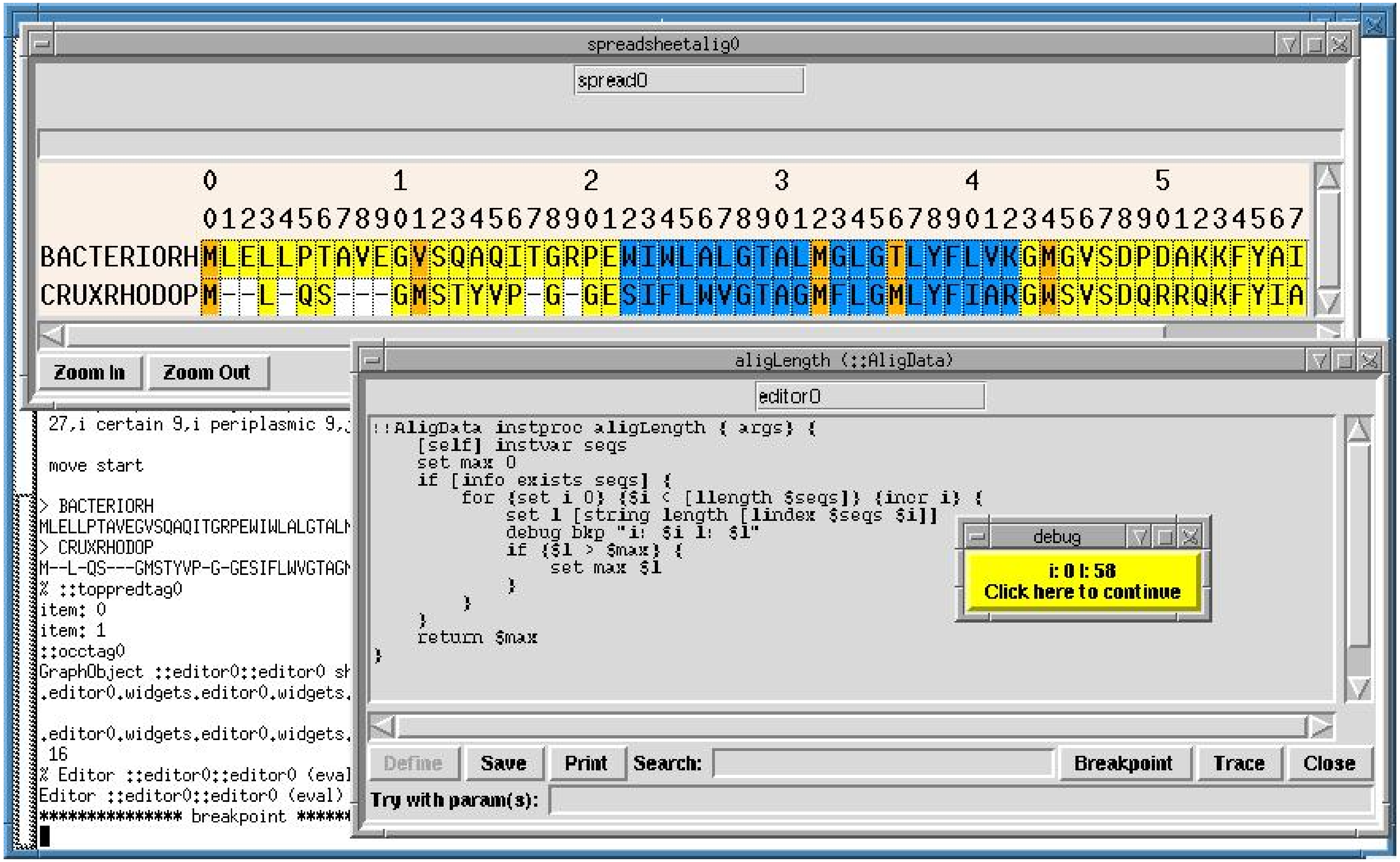
biok: sub-classing tags



biok: sub-classing tags



biok: programming



```
           0         1         2         3         4         5
           012345678901234567890123456789012345678901234567890123456
BACTERIORH MLELLPTAVEGVSQAQITGRPEWIWLALGTALMGLGTLYFLVKGMGVSDPDAKKFYAI
CRUXRHODOP M--L-QS---GMSTYVP-G-GESIFLWVGTAGMFLGMLYFIARGWSVSDQRRQKFYIA
```

```
::AligData instproc aligLength { args} {
    [self] instvar seqs
    set max 0
    if [info exists seqs] {
        for {set i 0} {$i < [llength $seqs]} {incr i} {
            set l [string length [lindex $seqs $i]]
            debug bkp "i: $i l: $l"
            if {$l > $max} {
                set max $l
            }
        }
    }
    return $max
}
```

```
i: 0 l: 58
Click here to continue
```

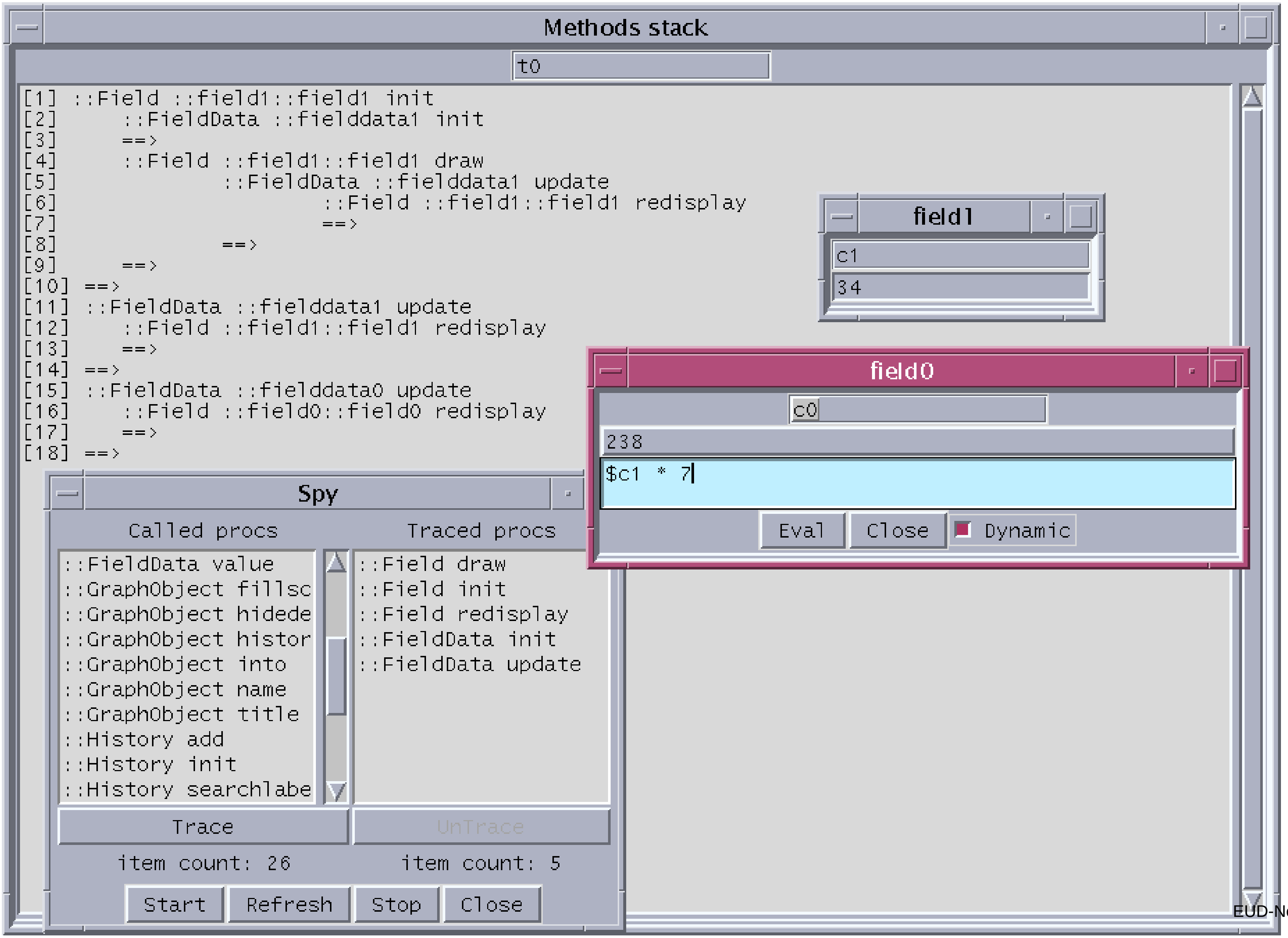
```
 27,i certain 9,i periplasmic 9,i

 move start

> BACTERIORH
MLELLPTAVEGVSQAQITGRPEWIWLALGTALM
> CRUXRHODOP
M--L-QS---GMSTYVP-G-GESIFLWVGTAGM
% ::toppredtag0
item: 0
item: 1
::occtag0
GraphObject ::editor0::editor0 sh
.editor0.widgets.editor0.widgets.

.editor0.widgets.editor0.widgets.
 16
% Editor ::editor0::editor0 (eval
Editor ::editor0::editor0 (eval)
*************** breakpoint ******
```

biok programming: tracing



```
[1] ::Field ::field1::field1 init
[2]     ::FieldData ::fielddata1 init
[3]     ==>
[4]     ::Field ::field1::field1 draw
[5]             ::FieldData ::fielddata1 update
[6]                     ::Field ::field1::field1 redisplay
[7]                     ==>
[8]             ==>
[9]     ==>
[10] ==>
[11] ::FieldData ::fielddata1 update
[12]    ::Field ::field1::field1 redisplay
[13]    ==>
[14] ==>
[15] ::FieldData ::fielddata0 update
[16]    ::Field ::field0::field0 redisplay
[17]    ==>
[18] ==>
```

```
$c1 * 7
```

```
::FieldData value               ::Field draw
::GraphObject fillsc            ::Field init
::GraphObject hidede            ::Field redisplay
::GraphObject histor            ::FieldData init
::GraphObject into              ::FieldData update
::GraphObject name
::GraphObject title
::History add
::History init
::History searchlabe
```

Programming In The User Interface

working environment

=

programming environment

- learning by examples
- incremental programming
- easier switch between 2 modes (using/programming); programming as just a kind of advanced use (with several levels) (but programming should not be required for a standard use)

Approaches

- enable <u>Programming In The User Interface</u> (*programmability*)

Approaches

- enable Programming In The User Interface (*programmability*)
- from programmability to *software flexibity*; for biologists, *lack of flexibility* in current software could provide a better explanation than technical or cognitive problems in programming

Approaches

- enable Programming In The User Interface (*programmability*)
- from programmability to *software flexibity*; for biologists, *lack of flexibility* in current software could provide a better explanation than technical or cognitive problems in programming
- flexibility is anticipation:

Approaches

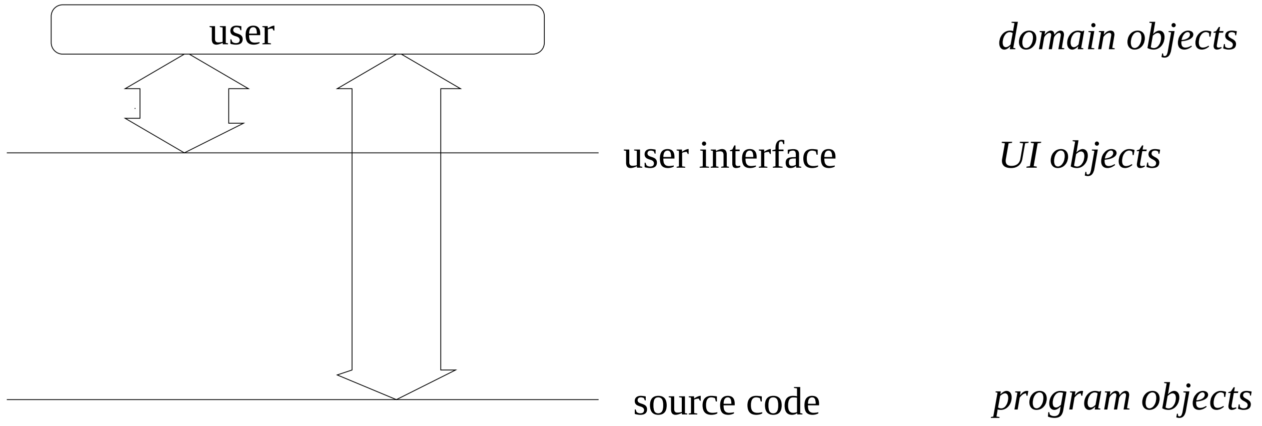
- enable Programming In The User Interface (*programmability*)
- from programmability to *software flexibity*; for biologists, *lack of flexibility* in current software could provide a better explanation than technical or cognitive problems in programming
- flexibility is anticipation:
  - programmability

Approaches

- enable Programming In The User Interface (*programmability*)
- from programmability to *software flexibity*; for biologists, *lack of flexibility* in current software could provide a better explanation than technical or cognitive problems in programming
- flexibility is anticipation:
  - programmability
  - MOP

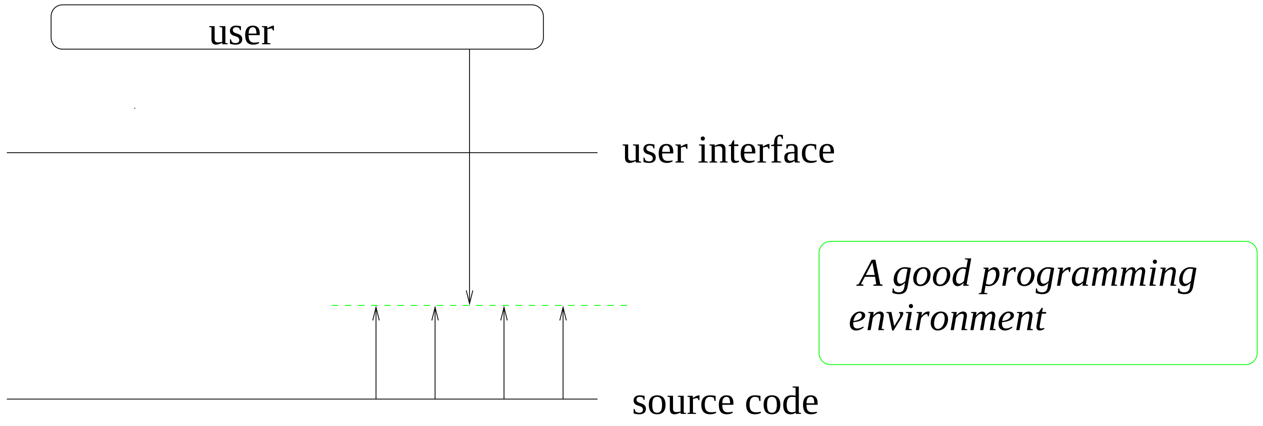
Modifiable systems

Cognitive distance issue:



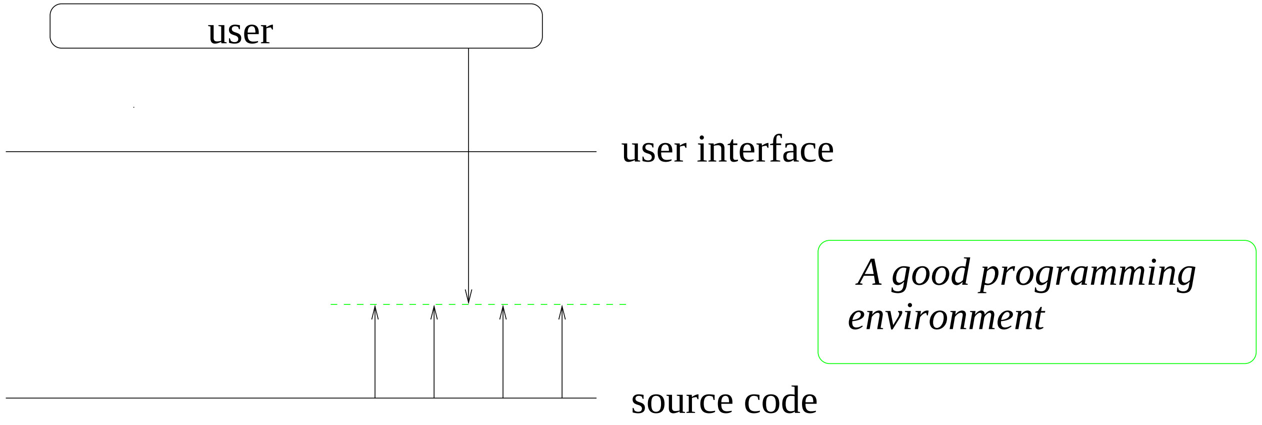
Modifiable systems

Cognitive distance issue:

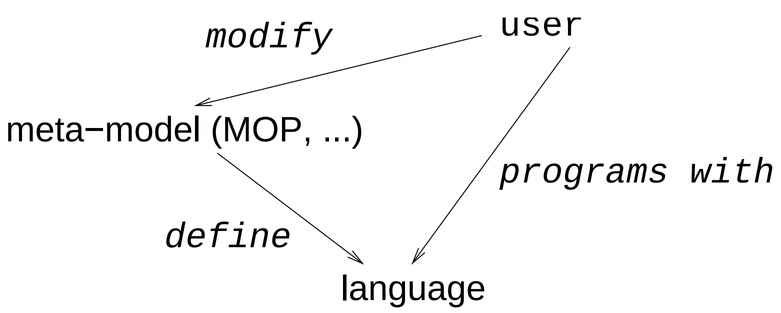


Modifiable systems

Cognitive distance issue:

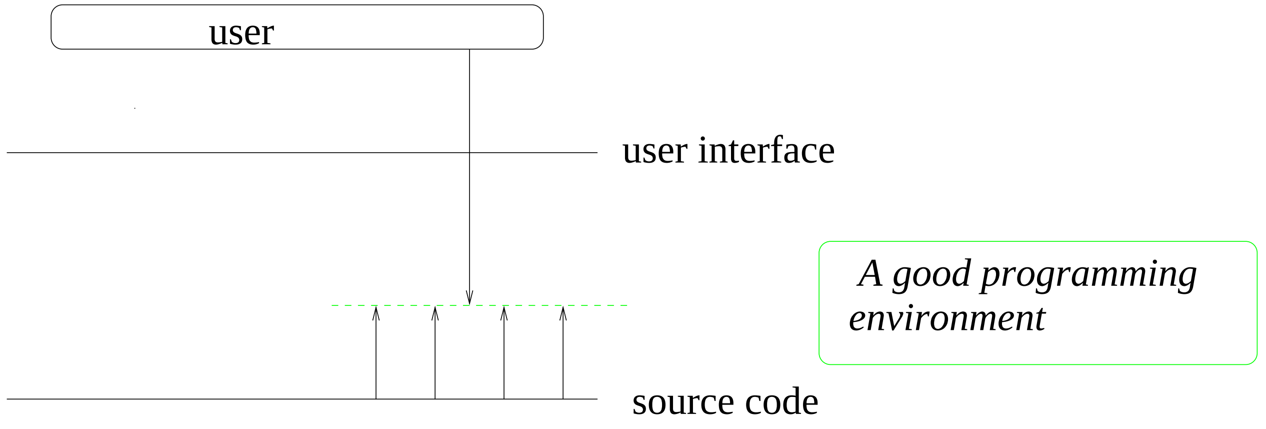


From MOP ...

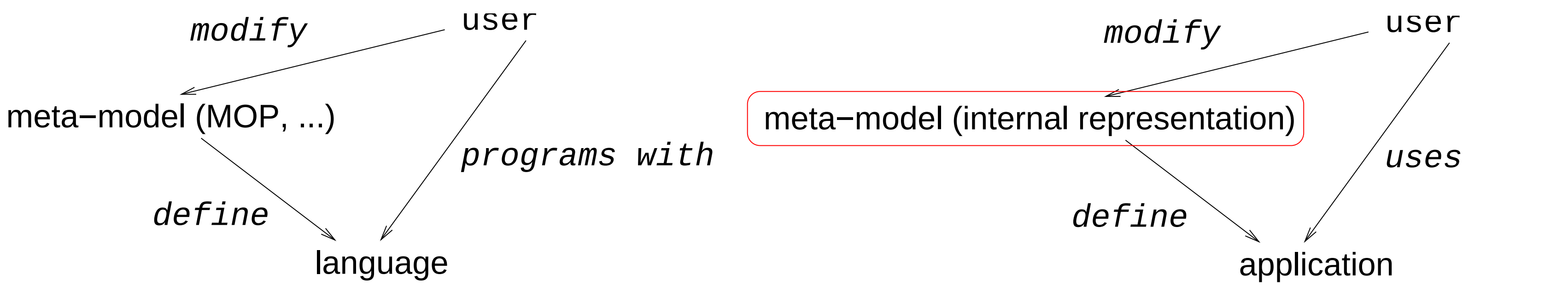


Modifiable systems

Cognitive distance issue:

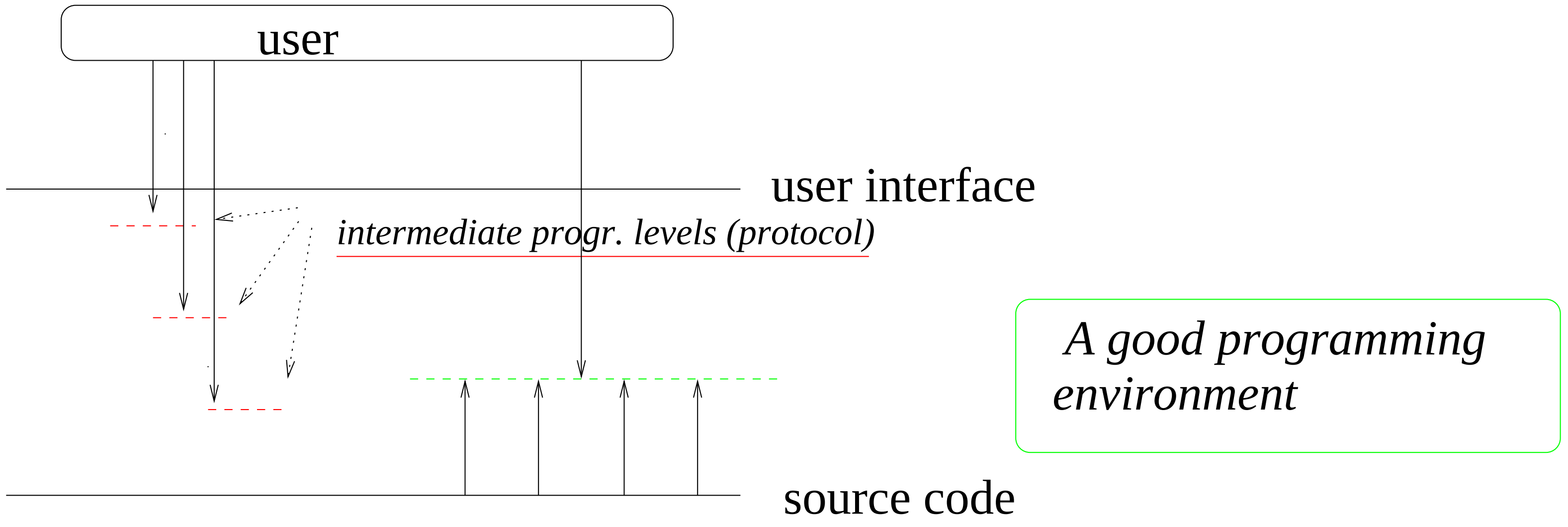


From MOP ... to MAP (Meta-Application Protocol):

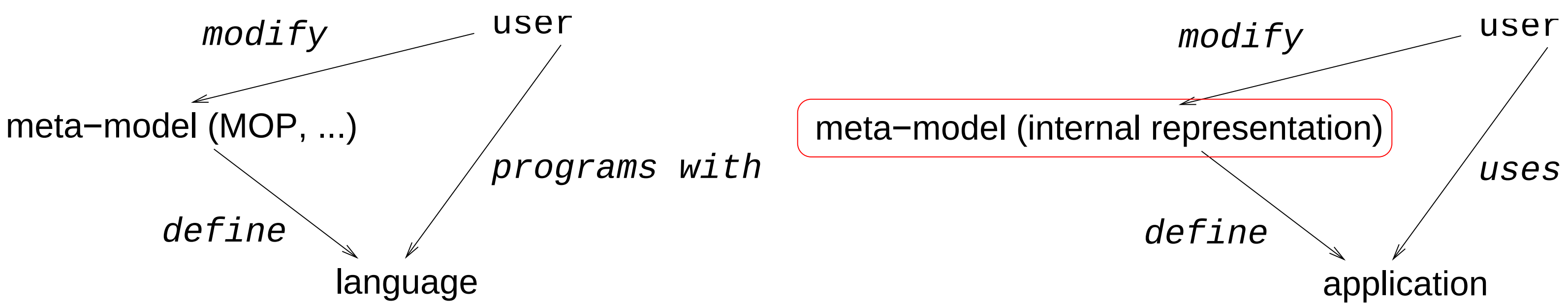


Modifiable systems

Cognitive distance issue:


user
user interface
intermediate progr. levels (protocol)
A good programming environment
source code

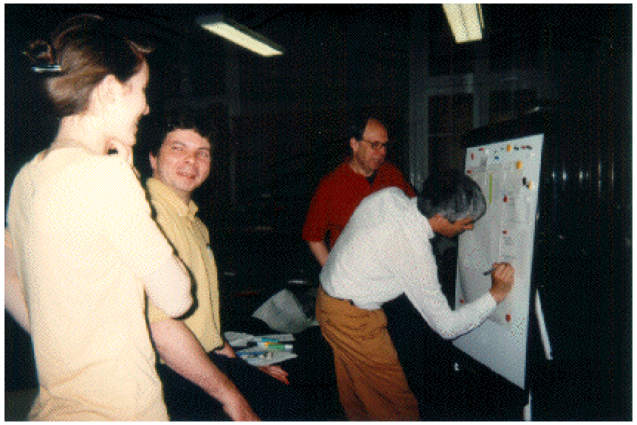
From MOP ... to MAP (Meta-Application Protocol):

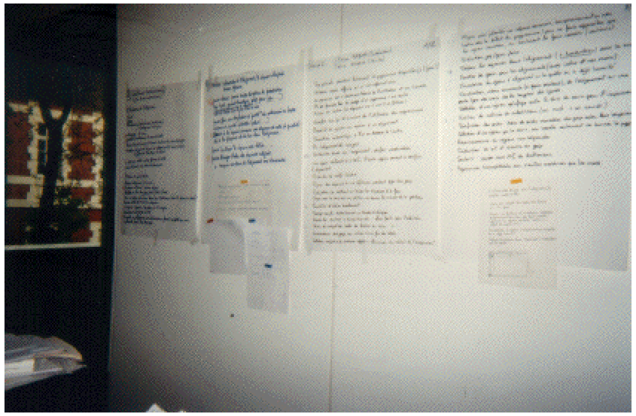


Approaches

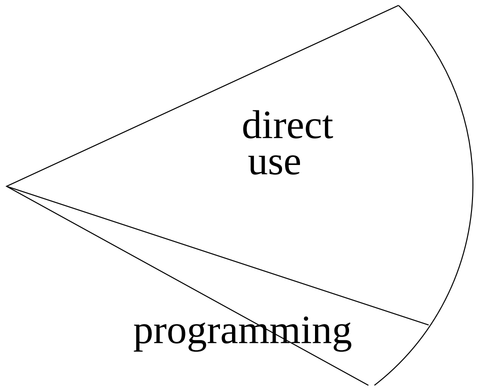
- enable Programming In The User Interface (*programmability*)
- from programmability to *software flexibity*; for biologists, *lack of flexibility* in current software could provide a better explanation than technical or cognitive problems in programming
- flexibility is anticipation:
  - programmability
  - MOP
  - participatory design,

Participatory Design

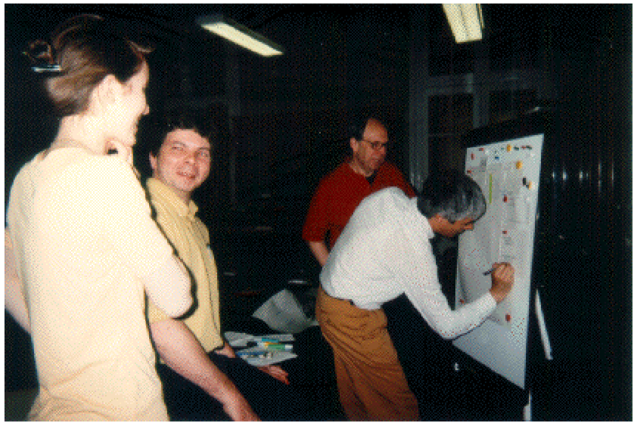


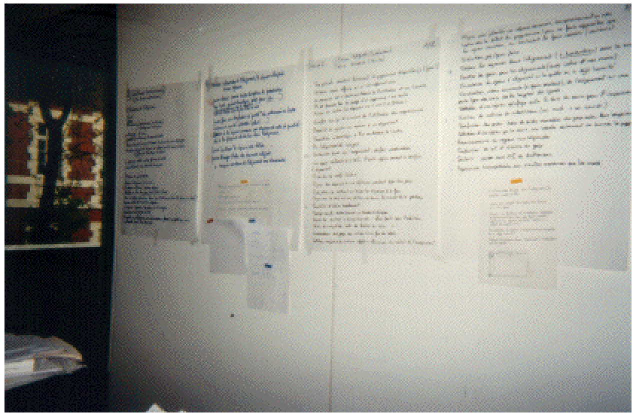


- anticipate flexibility spots (EU-not-P versus EUP)



Participatory Design





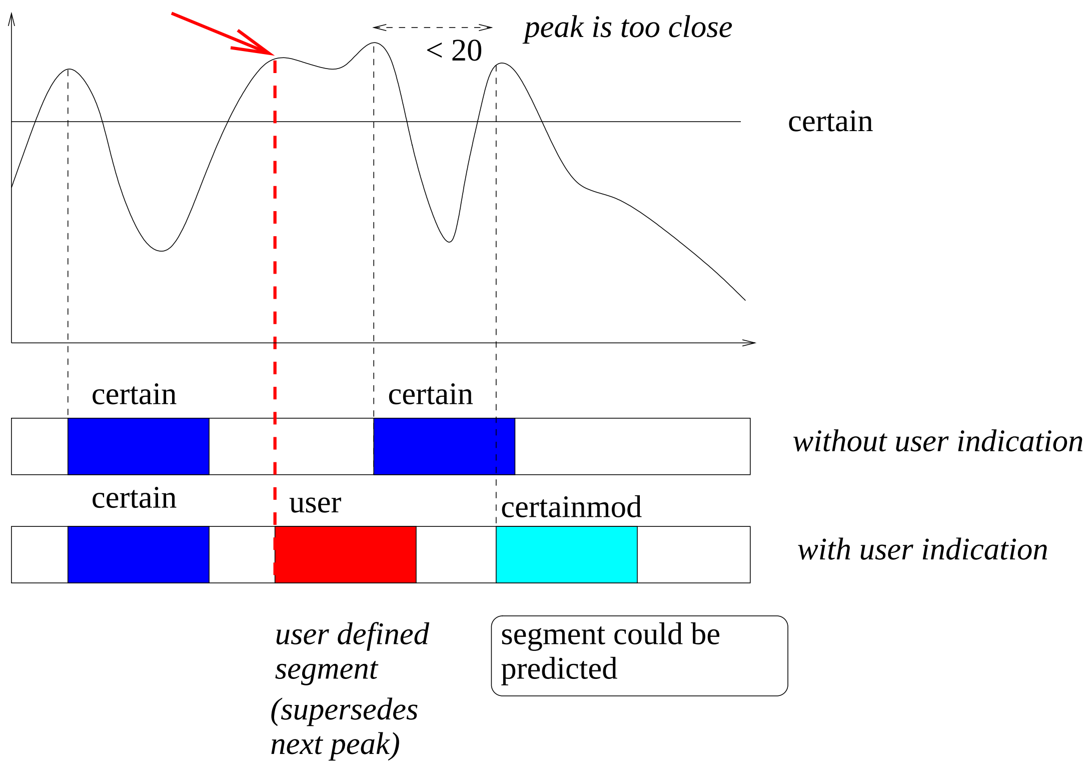
- rather *few* programming issues raised in design workshops

Approaches

- enable Programming In The User Interface (*programmability*)
- from programmability to *software flexibity*; for biologists, *lack of flexibility* in current software could provide a better explanation than technical or cognitive problems in programming
- flexibility is anticipation:
  - programmability
  - MOP
  - participatory design,
- flexibility is provided in informal tools (spreadsheets, ...),
- flexibility is also more control on computation, (*algorithmic flexibility*)

Algorithmic flexibility

A way to integrate user's knowledge in computation neither as parameter nor as programming (using interception techniques (AOP?)).



differences in results are highlighted

Conclusions

- Context:

Conclusions

- Context:
    - user studies: critical need for flexibility

Conclusions

- Context:
  - user studies: critical need for flexibility
  - writing text code is ok for biologists ...

Conclusions

- Context:
  - user studies: critical need for flexibility
  - writing text code is ok for biologists ...
  - ... although programming is not the goal

Conclusions

- Context:
  - user studies: critical need for flexibility
  - writing text code is ok for biologists ...
  - ... although programming is not the goal
- Approach:

Conclusions

- Context:
  - user studies: critical need for flexibility
  - writing text code is ok for biologists ...
  - ... although programming is not the goal
- Approach:
  - user-development also as a software design issue

Conclusions

- Context:
  - user studies: critical need for flexibility
  - writing text code is ok for biologists ...
  - ... although programming is not the goal
- Approach:
  - user-development also as a software design issue
  - *participative programming*

Conclusions

- Context:
  - user studies: critical need for flexibility
  - writing text code is ok for biologists ...
  - ... although programming is not the goal
- Approach:
  - user-development also as a software design issue
  - *participative programming*
  - Who is Mr/Ms End-User...?

Conclusions

- Context:
  - user studies: critical need for flexibility
  - writing text code is ok for biologists ...
  - ... although programming is not the goal
- Approach:
  - user-development also as a software design issue
  - *participative programming*
  - Who is Mr/Ms End-User...?
    - *the user of the software*, e.g:

Conclusions

- Context:
  - user studies: critical need for flexibility
  - writing text code is ok for biologists ...
  - ... although programming is not the goal
- Approach:
  - user-development also as a software design issue
  - *participative programming*
  - Who is Mr/Ms End-User...?
    - *the user of the software*, e.g:
    - the biologist,

Conclusions

- Context:
  - user studies: critical need for flexibility
  - writing text code is ok for biologists ...
  - ... although programming is not the goal
- Approach:
  - user-development also as a software design issue
  - *participative programming*
  - Who is Mr/Ms End-User...?
    - *the user of the software*, e.g:
    - the biologist,
    - ... (have you ever really tried to re-program emacs?)

Conclusions

- Context:
  - user studies: critical need for flexibility
  - writing text code is ok for biologists ...
  - ... although programming is not the goal
- Approach:
  - user-development also as a software design issue
  - *participative programming*
  - Who is Mr/Ms End-User...?
    - *the user of the software*, e.g:
    - the biologist,
    - ... (have you ever really tried to re-program emacs?)
    - not defined by a skill level, rather by his/her role towards the software

Conclusions

- Context:
  - user studies: critical need for flexibility
  - writing text code is ok for biologists ...
  - ... although programming is not the goal
- Approach:
  - user-development also as a software design issue
  - *participative programming*
  - Who is Mr/Ms End-User...?
    - *the user of the software*, e.g:
    - the biologist,
    - ... (have you ever really tried to re-program emacs?)
    - not defined by a skill level, rather by his/her role towards the software