Towards a Dynamic Strategy for Computer-Aided Visual Placement

François Bodart, Anne-Marie Hennebert, Jean-Marie Leheureux, and Jean Vanderdonckt
Facultés Universitaires Notre-Dame de la Paix, Institut d'Informatique,
rue Grandgagnage, 21, B-5000 NAMUR (Belgium)
Tel: +32 (0)81-72.49.75 - Fax: +32 (0)81-72.49.67 - Telex: 59.222 FacNamB
Email: {fbodart, amhennebert, jmleheureux, jvanderdonckt}@info.fundp.ac.be

ABSTRACT
This study is devoted to the layout problem in the TRIDENT project (Tools foR an Interactive Development ENvironmenT), which is dedicated to highly interactive business-oriented applications. In this project, the placement problem consists of a computer-aided visual placement of interaction objects (IO) included in a more composite IO called Presentation Unit (PU). Two strategies for placing IO within a PU are characterised and investigated : a static two-column based strategy and a dynamic right/bottom strategy. Each strategy decomposes the placement into three partially overlapping dimensions : localisation, dimensioning, and arrangement. A set of simple mathematical relationships is introduced to rate the quality of visual principles gained with the result of each strategy within the three dimensions. This rating shows that the lack of flexibility should lead us more towards a dynamic strategy for computer-aided visual placement.

KEYWORDS: Dynamic Strategy, Grid, Interaction Objects, Layout, Presentation Unit, Static Strategy, Visual Interface Design and Management, Visual Interaction, Visual Placement, Visual Techniques.

INTRODUCTION
When the time arises to decide how to visually place interaction objects (IO) of a user interface into a more composite IO (e.g. a window, a panel or a dialog box), it is important to draw and apply a visual layout. This layout should not only be consistent and attractive but, moreover, should be compatible with the user's task. Therefore, choosing a good layout configuration relies on the involvement of the user's task [2] that a placement strategy is supposed to reflect. In general, each placement strategy decomposes the placement into three partially overlapping dimensions :

1. the localisation, which answers to the question “where to place information on a screen?” (or “where to place IO in a PU?”);
2. the dimensioning, which answers to the question “how to place information on a screen?” (or “with which dimensions to size IO in a PU?”);
3. the arrangement, which answers to the question “according to which order to place information on a screen?” (or “how should IO be arranged in a PU?” by logical order, by frequency or by predefined format?).

General and specific guidelines can be equally stated for these three dimensions [1], but their application remains without real effect unless they are translated into visual principles expressed with the help of mathematical relationships. These mathematical relationships, because of their workability, may be applied to IO to achieve effective placement.

PLACEMENT IN TRIDENT
In TRIDENT project, the layout problem consists of a computer-aided visual placement of IO composing a Presentation Unit (PU). Two strategies for placing interaction objects are characterised and investigated : a static *two-column based strategy* and a dynamic *right/bottom strategy*. Each strategy decomposes the placement into three subparts for which extensive guidelines can be found in the literature [1,5,9] :

1. the *localisation* is interested by logically positioning IO in a PU. It covers position, alignment, justification. Localisation can be achieved through
   - consistency, e.g. the position of IO should be compatible with the users' conventions, consistent in format;
   - sequentiality, e.g. most frequent used IO should be located first;
   - screen image, e.g. IO should be located in all four quadrants of the container.
2. the *dimensioning* goes in for the uniformity and standardisation of IO dimensions. It deals for instance with the length of abbreviations, the maximum number of characters per label, the length of an edit box, the item number in a list box, the harmony between length and height for a dialog box.
3. the *arrangement* takes into account the IO orientation and constraints related to the logical ordering of IO. Arrangement should be logical as much as possible (e.g.

by preference, by consensus, by physical property, by data flow), should emphasize visual cues, should care about aesthetics and should reduce ocular movements and screen density.

About 300 guidelines for these dimensions have been gathered [1]. Both strategies have common properties:

- they apply some mathematical relationships and visual principles [12];
- they are based on the information ordering for promoting visual continuity;
- they rely on the information structure and definition for reflecting structure, grouping and proximity.

But these strategies differ in the sense they strive for different visual criteria by using different techniques :

- the two-column based strategy employs a predefined two-column layout grid for preserving consistency;
- the right/bottom strategy places the next IO either on the right or at the bottom of the previous IO according to different heuristics minimising unused screen space.

While the first strategy is static and rigid, the second strategy is shown to be dynamic and flexible because of its ability to progressively place IO step by step with dynamic intervention of heuristics on a state-space representation. Dynamic strategies are finally believed as more flexible strategies for facilitating the designer assessment of different steps before obtaining the final placement. To establish this conclusion, the judgement will be based on mathematical relationships and visual principles.

MATHEMATICAL RELATIONSHIPS

Before to see relevant visual principles, it is sound to define mathematical relationships to improve the practicability, the workability and the applicability of these principles into a systematic strategy. The co-ordinates of upper left and bottom right corners characterize each IO (fig. 1).

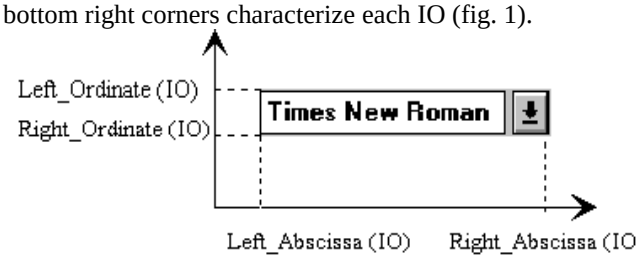


Fig. 1. Co-ordinates of an Interaction Object.

Mathematical relationships are first described with the natural language and then defined using the first-order predicate calculus (table 1). "iff" stands for "if and only if":

- *Horizontal sequencing* : two IO are horizontally sequenced iff the right abscissa of the first IO is less than the left abscissa of the second IO (fig. 2a).
- *Vertical sequencing* : two IO are vertically sequenced iff the right ordinate of the first IO is greater than the left ordinate of the second IO (fig. 2b).

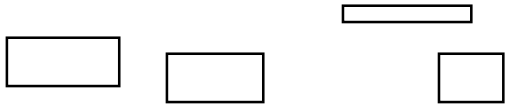


Fig. 2. Horizontal and vertical sequencings.

- *Left justification* : two IO are left justified iff their left abscissa are identical (fig. 3a).
- *Right justification* : two IO are right justified iff their right abscissa are identical (fig. 3b).
- *Upper justification* : two IO are upper justified iff their left ordinates are identical (fig. 3c).
- *Bottom justification* : two IO are bottom justified iff their right ordinates are identical (fig. 3d).

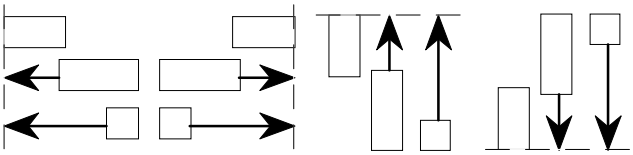


Fig. 3. Left, right, upper and bottom justifications.

- *Horizontal centering* : two IO are horizontally centered iff the ordinates of their centres are identical (fig. 4a).
- *Vertical centering* : two IO are vertically centered iff the abscissa of their centres are identical (fig. 4b).

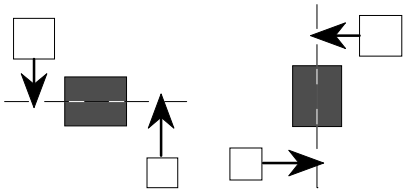


Fig. 4. Horizontal and Vertical centerings.

- *Horizontal equilibrium* : three IO are horizontally equilibrated iff the horizontal inter-object distances are identical (fig. 5a).
- *Vertical equilibrium* : three IO are vertically equilibrated iff the vertical inter-object distances are identical (fig. 5b).
- *Diagonal equilibrium* : three IO are diagonally equilibrated iff their centres are equally distributed (fig. 5c).

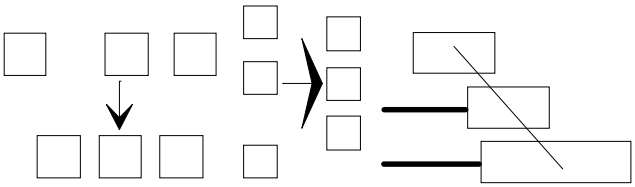


Fig. 5. Horizontal, vertical and diagonal equilibrium.

- *Horizontal uniformity* : two IO are horizontally uniform iff their lengths are identical.
- *Vertical uniformity* : two IO are vertically uniform iff their heights are identical.

Let ℑ be the set of all possible IO. ∀ x,y,z ∈ ℑ:	
HOR_SEQ (x,y) ⇔	Right_Abscissa (x) < Left_Abscissa (y)
VER_SEQ (x,y) ⇔	Right_Ordinate (x) > Left_Ordinate (y)
LEFT_JUST (x,y) ⇔	Left_Abscissa (x) = Left_Abscissa (y)
RIGHT_JUST (x,y) ⇔	Right_Abscissa (x) = Right_Abscissa (y)
UP_JUST (x,y) ⇔	Left_Ordinate (x) = Left_Ordinate (y)
BOT_JUST (x,y)⇔	Right_Ordinate (x) = Right_Ordinate (y)
HOR_CENT (x,y) ⇔	Left_Abscissa (x) + [Right_Abscissa (x) - Left_Abscissa (x)]/2 = Left_Abscissa (y) + [Right_Abscissa (y)-Left_Abscissa (y)]/2
VER_CENT (x,y) ⇔	Right_Ordinate (x) +[Left_Ordinate (x) - Right_Ordinate (x)]/2 = Right_Ordinate (y) + [Left_Ordinate (y) - Right_Ordinate (y)]/2
HOR_EQUI (x,y) ⇔	Right_Abscissa (x) - Left_Abscissa (x) = Right_Abscissa (y) - Left_Abscissa (y)
VER_EQUI (x,y) ⇔	Left_Ordinate (x) - Right_Ordinate (x) = Left_Ordinate (y) - Right_Ordinate (y)
DIA_EQUI (x,y,z) ⇔	HOR_EQUI (x,y) Λ VER_EQUI (x,y)
HOR_UNIF (x,y) ⇔	Left_Abscissa (z) - Right_Abscissa (y) = Left_Abscissa (y) - Right_Abscissa (x)
VER_UNIF (x,y) ⇔	Right_Ordinate (z) - Left_Ordinate (y) = Right_Ordinate (y) - Left_Ordinate(x)
PROP_EQUI (x,y,z) ⇔	[VER_UNIF (x,y,z) Λ VER_EQUI (x,y,z)] V [HOR_UNIF (x,y,z) Λ HOR_EQUI (x,y,z)]
TOT_EQUI (x,y,z) ⇔	[VER_EQUI (x,y,z) V HOR_EQUI (x,y,z)] Λ HOR_UNIF (x,y,z) Λ VER_UNIF (x,y,z)

Table 1. Logical formulas of mathematical relations.

- *Proportional equilibrium* : three IO are proportionally equilibrated iff they are uniform and equilibrated either vertically or horizontally (fig. 6a).
- *Total equilibrium* : three IO are totally equilibrated iff they are equilibrated either vertically or horizontally, and uniform vertically and horizontally (fig. 6b).

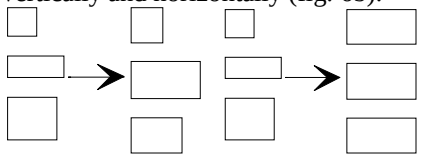


Fig. 6. Proportional and total equilibrium.

VISUAL PRINCIPLES

Effective visual design of layouts definitely contributes to quality and usability [9]. The following question arises : what visual principles should guide design decisions of layouts? First things first : the users' task and the information to be displayed are to be defined before applying visual principles. Numerous inspiring visual principles come from graphical arts, graphical techniques, screen design guidelines, visual and verbal communication [1,5,6,9,12]. These visual principles are useful for clarifying how layouts should be formatted, structured and organised, but do not say anything about the content itself. The principles have been arranged in opposite poles on a same scale not only to demonstrate and accentuate the wide range of possible applications but, also, to underline the need of harmony versus contrast.

On one side, visual design may seek harmony, where all visual elements and stimuli are placed in a state of ease, reducing tension, stress, and increasing pleasance. On the other hand, contrast is a counter face to this human desire by unbalancing all elements, shocking, disturbing, stimulating, arresting attention. Contrast can be expressed in tone, color, shape, and scale. Visual principles listed in [12] include balance, symmetry, regularity, alignment, proportion, horizontality,... These principles are sorted by similarity and not by rank of importance because all visual principles cannot be applied with the same representativeness. Some principles are very straightforward to apply, some other are more difficult to compel with, and some other become very hard to translate. Moreover, applying this or that principle mostly depends of the involved IO and the visual aims that the designer has in mind. These visual principles will be now exploited in the definition of two placement strategies.

A TWO-COLUMN BASED STATIC STRATEGY

Description Of The Strategy

The layout grid in our first strategy has the general form illustrated in fig. 7. Its purpose is to divide the sorted sequence of IO into two columns (as in a book). The two columns are to be equal in length and proportion as much as possible by applying mathematical relationships :

- vertical centering for the title and two columns;
- the four justifications, the two uniformities and vertical or horizontal equilibrium for column contents;
- proportional or total equilibrium for push buttons,icons.

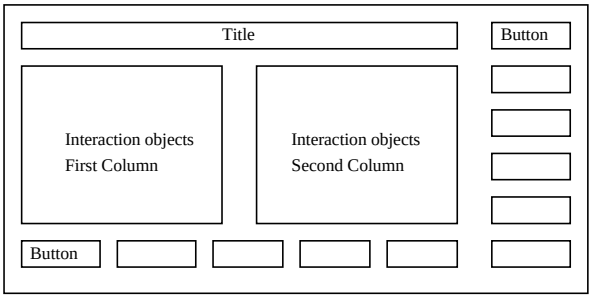


Fig. 7. The layout grid in the two-column strategy.

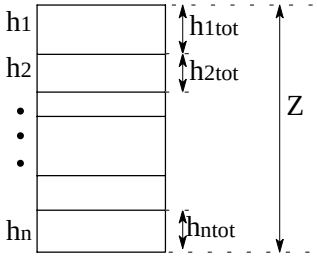


Fig. 8. Dimensions of stacked IO.

Actually, proportional and total equilibrium are the most complete and aesthetic spatial relationships. Therefore, these are also the most difficult to apply for the two columns. The summarised steps are the following :

1. the characterisation of the IO sequence: the contents of all IO are identified (identification and descriptive labels, prompt and field);

2. the fixation of visual parameters: standard space between two lines or rows (I), between label and prompt, prompt and field (L) in terms of a measure unit (U) (in characters, pixels or points), font height (H);

3. the calculation of standard dimensions of the IO. All IO are pushed in a vertical stack (fig. 8) whose dimensions are calculated recursively as the sum of the heights of individual IO (fig. 9) : $Z = \sum_{i=1}^{n} h_{itot}$

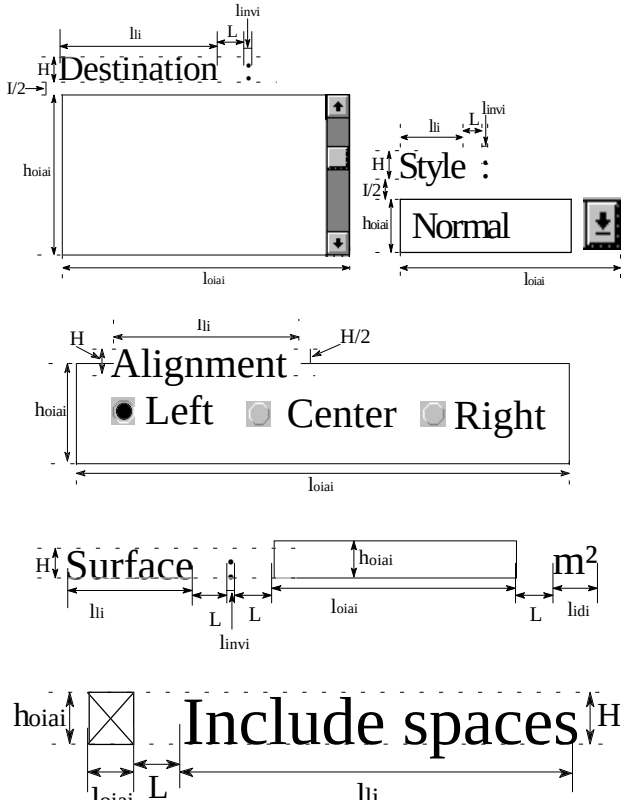


Fig. 9. Individual dimensions of IO.

4. the computation of the dimensions of the two columns: the above stack is bipartitioned by finding regular proportions between the columns. If Z/2 falls between two IO, the two columns are directly defined. Otherwise, this is achieved by minimising the deviations between the frontier of IO : if the deviation before the cut IO is smaller (respectively, greater) than the one beneath, the splitting will be decided before (respectively, after) this IO :

   Compute

   $\varepsilon_1 = \left| Z/2 - \sum_{i=1}^{k-1} h_{itot} \right|$, $\varepsilon_2 = \left| \sum_{i=1}^{k} h_{itot} - Z/2 \right|$,

   $\varepsilon_{min} = \min(\varepsilon_1, \varepsilon_2)$.

   If $\varepsilon_1 \leq \varepsilon_2$, then $n_1 = k - 1$ else $n_1 = k$;

   calculate $n_2 := n - n_1$.

   In each column, different relations will be progressively satisfied: vertical sequencing, justification, uniformity and equilibrium. For instance, if there exists a sub-sequence of IO of the same type, their labels and fields are subject to equilibrium. Final dimensions h_{tot} and l_{tot} are derived from heights and lengths of the two columns with n_1, n_2 IO respectively:

   $h_1 = \sum_{i=1}^{n1} h_{itot} + (n_1 - 1)I$, $l_1 = \max_{i=1,...,n1}(l_{itot})$,

   $h_2 = \sum_{i=n1+1}^{n} h_{itot} + (n_2 - 1)I$, $l_2 = \max_{i=n1+1,...,n}(l_{itot})$.

   $h_{tot} = H + I/2 + 1 + I/2 + \max(h_1,h_2) + I + m(H + I/2)$,

   $l_{tot} = l_1 + 2L + l_1$.

5. the calculation of the internal proportion: after adding title, message area and separators if necessary, the actual proportion is calculated;

6. the adding of standard and custom push buttons, drawn buttons, and icons (if any). The goal is to choose whether these buttons will be arranged in column on the right (fig. 12a) or in a row at the bottom of the composite IO (fig. 12b). We already known that, if the internal proportion $p_{int} = l_{tot}/h_{tot} \geq 1$, then horizontality is achieved ; otherwise, verticality results. First, respective dimensions are computed in the two cases according to the formulas reproduced in fig. 10, 11. The orientation is then decided with the organisation heuristic detailed in fig. 13. Proportional and total equilibrium are tried with vertical or horizontal equilibrium according to the internal proportion induced by the chosen organisation. Let *g* be the total amount of icons.

if g = 0,
then

$h_{vert} = c*\overline{H} + (c - 1)*I/2$, $l_{vert} = \max_{i=1,...,c}(l_{itot}) + 2L$

else

$h_{vert} = c*\overline{H} + (c - 1)*I/2 + I + (g - 1)*I/2 + \sum_{i=1}^{q} h_{itot}$

$l_{vert} = \max [\max_{i=1,...,c}(l_{itot}) + 2L, \max_{i=1,...,g}(l_{itot})]$

Fig. 10. Dimensions for vertical disposition.

if g = 0, then

$h_{horiz} = \overline{H}$, $l_{horiz} = c*\max_{i=1,...,c}(l_{itot}) + (c - 1)*L$

else

$h_{horiz} = \max [\overline{H}, \max_{i=1,...,g}(h_{itot})]$, $l_{horiz} = c*\max_{i=1,...,c}(l_{itot}) + (c - 1)*L + 2L + \sum_{i=1}^{q} l_{itot} + (g - 1)*L$

Fig. 11. Dimensions for horizontal disposition.

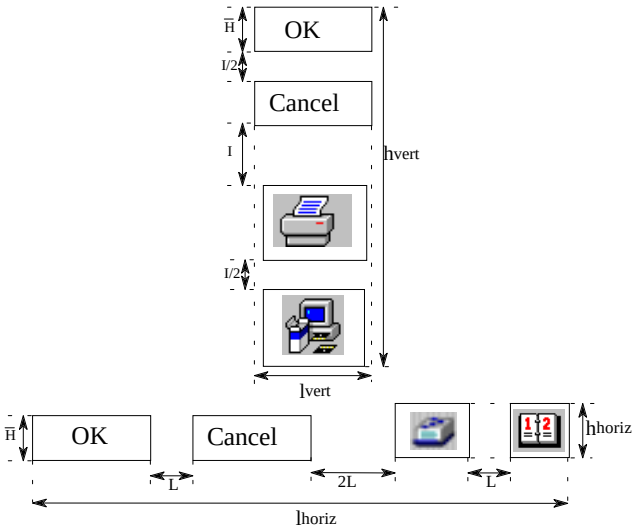


Fig. 12. Vertical and horizontal organizations of push buttons, drawn buttons, and icons.

if $p_{int} \geq 1$
then
  if $h_{vert} \leq h_{tot}$
  then choose vertical right organization
  else
    if $l_{horiz} \leq l_{tot}$
      then choose horizontal bottom organization
      else
        if $g \neq 0$
          then choose vertical right organization in two columns
          else choose vertical right organization in one column
else choose vertical right organization

Fig. 13. Placement heuristic.

7. the selection of appropriate margins: margins result from an optimisation problem based on the internal proportion (fig. 14). This proportion is forced to converge to one of the proportion recommended by Marcus [9] such as : 1:√2, 1:1+√5/2, 1:2, 1:1.29, 1:1.5, 1:4/3, 1:1.6.

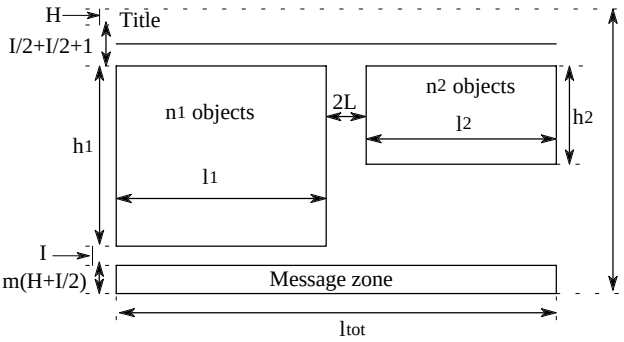


Fig. 14. Computing the internal proportion.

Examples and Discussion

This strategy is now applied to an example taken in the field of a hospital admission application. The hierarchy of predefined IO is depicted in fig. 15 with the appropriate information ordering to be respected. DBX denotes a dialog box, GBX denotes a group box, EBX denotes an edit box, RBX denotes a radio box, SLB denotes a scrollable list box, MSG denotes a message area and PBT denotes a push button.

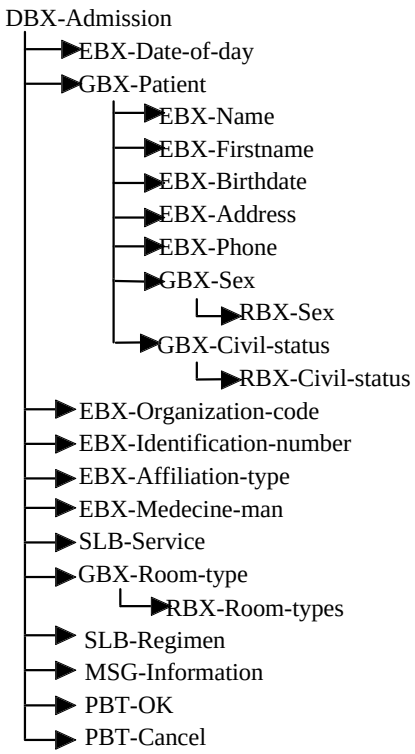


Fig. 15. Hierarchy of IO.

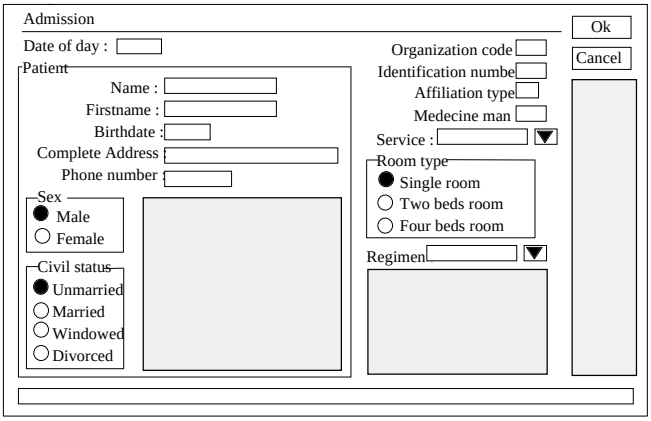


Fig. 16. Example with two-column strategy

This two-column based strategy produced the graphical dialog box of fig. 16. We now analyse this strategy :

- *flexibility* : the strategy provides a tree with physically arranged IO and supports the information ordering since the IO ordering reflects the information ordering. This strategy is rigid since it proposes a single version of the future dialog box according to a constant layout;
- *button placement* : the strategy always places buttons either horizontally at the bottom of the dialog box or vertically on the right of the dialog box. Though always consistent and visually isolated, these placements require a lot of space;
- *column placement* : the strategy attempts to share out IO equally amongst the two columns to preserve balance and repartition. Thus, if the number of IO is high, a visually appealing dialog box will result; if the number of IO is very low, a flat unappealing dialog box may be viewed and horizontality become irrelevant;
- *proportion* : the strategy uses a convergence to a recommended proportion. This could resize global dimensions if the current dialog box has bad dimensions, but this can potentially lead to an increase of the global dimensions. But the dimensions are big regardless the unused greyed regions showed in fig. 16.
- *alignment* : the strategy automatically establishes all possible alignments increasing the feeling of unity;
- *grouping* : the strategy lay out IO in each group recursively, thus preserving the grouping principle.

The two-column strategy lacks both generality and reusability on other contexts. This strategy have been so defined to conform to a predefined grid so that changing the number of columns, modifying the tile or push button location is not supported. The last goal for this strategy was to build a narrow, specific tool for business-oriented applications, and therefore more sophisticated than a large general tool where precision and consistency can be endangered.

A RIGHT/BOTTOM DYNAMIC STRATEGY

Strategy Description

After discussing these advantages and inconveniences, static placement strategies (such as the two-column based strategy) are unable to reach the goal of effective and ergonomic placement because they are too rigid, not flexible enough for that purpose. A strategy which dynamically places IO can offer more alternatives among which we can choose, refine iteratively. Let S_i denotes the IO placed at time i. S_{i+1} is the next IO to be placed. The idea of the *right/bottom strategy* consists of following the visual continuity principle by either placing S_{i+1} on the right of S_i or beneath S_i. The placement strategy is defined as the following :

if the total length does not exceed the limit
then
  place S_{i+1} with horizontal sequencing
  three cases are to be considered
  1. height (S_i) = height (S_{i+1})
     apply proportional uniformity
  2. height (S_i) > height (S_{i+1})
     if S_{i+1} = edit box
       then
         if S_i = list box or edit box
           then apply bottom justification
           else apply upper justification
  3. height (S_i) < height (S_{i+1})
     if available space is sufficient
     then apply bottom justification (fig. 17a)
     else maximize upper justification (fig. 17b)
else
  place S_{i+1} with vertical sequencing.

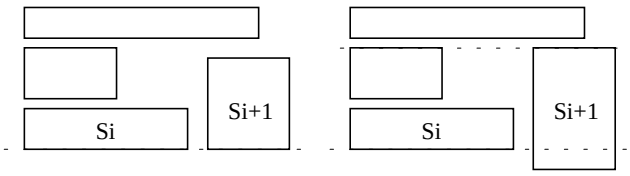


Fig. 17. Two situations in the Right/Bottom Strategy.

Because each step consists of two alternatives, the space of all placement alternatives becomes a complete state-space which is a binary tree. A *state* is a set of placement conditions that describe the system at a specified point during the processing. At each state, dynamic heuristics are applied depending on the sequence and type of IO. These heuristics are based on the mathematical relationships. For example, apply horizontal equilibrium among horizontally placed IO, apply vertical equilibrium among vertical group, apply proportional or total equilibrium for any group of radio boxes. Finding a good placement then consists of

- browsing the tree,
- examining right/bottom alternative placements at each state,
- applying placement heuristics,
- cutting branches which lead to unpleasant placements.

Some Placement Sub problems

In the placement problem, two particular problems can be readily solved due to their well-known nature :

1. the placement of labels : identification labels of IO can either be placed on the left of the IO or on the top of it. In the first case, horizontal sequencing and bottom justification are applied ; in the second case, vertical sequencing and left justification are applied.

```
if ni ≥ 5,
    then
        if ni is even
            then
                compute s = the greatest squared number
                which is smaller or equal than ni
                if s = ni
                    then
                        place radio items in a √s x √s squared
                        grid
                    else
                        compute Δ = ni - c
                        place radio items in √s x √s squared
                        grid with a row of Δ items
            else
                compute ni = ni - 1
                go to the "even" case and place radio items in
                a square with a row of one item
    else
        if ni = 4
            then
                if l ≤ lmax,
                    then
                        place radio items in a 2x2 grid
                    else
                        place items in a 4x1 vertical grid
            else
                if ni = 3,
                    then
                        if l ≤ lmax
                            then
                                place radio items in a 1x3
                                horizontal grid
                            else
                                place radio items in a 3x1
                                vertical grid
                    else
                        if l ≤ lmax
                            then
                                place radio items in a 1x2
                                horizontal grid
                            else
                                place radio items in a 2x1
                                vertical grid
```

Fig. 18. A Placement technique for radio buttons.

2. the placement of radio-buttons : a small algorithm technique (fig. 18) is suggested to preserve balance, unity, and repartition. Let *ni* be the total number of radio items and *l* the maximal length of the items.

The principle of the placement technique for radio buttons (fig. 18) is to "square" as much as possible the arrangement of radio buttons to reach equal balance and repartition. Grouping is automatically done by surrounding the radio items by a group box. Horizontality is only preferred for short radio items.

Dynamic Heuristics

1. If a right or a bottom placement involves a horizontal or vertical dimension that exceeds a maximum value (or the screen space dimension), cut this branch;

2. If the remaining space between S_{i-2} is large, the S_i should be placed at the bottom of S_{i-2}, with left justification (fig. 19);

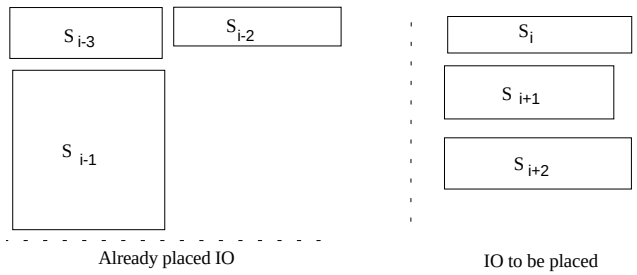


Fig. 19. Space heuristic.

3. If a branch leads where a large IO (S_{i-1}) is placed on the right of a former IO (S_{i-1}), cut this branch to avoid placing S_i and S_{i+1} on the bottom of S_{i-2} (fig. 20);

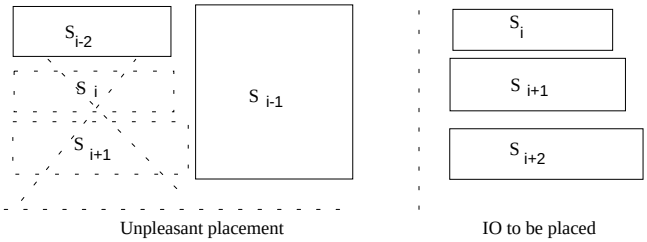


Fig. 20. Placement heuristic for large IO.

4. If there exists a sub-sequence of similar IO, their labels and/or their prompts and/or their controls are subject to left/right justification (fig. 21, 22):

```
if ∃ j, k  tq  1 ≤ j < k ≤ n  et  oia_j = oia_{j+1} = ... = oia_k
then
    compute l_min = min_{i=j,...,k}(l_li), l_max = max_{i=j,...,k}(l_li)
    if l_min ∈ [l_max - 5, l_max ]
        (* if the length deviation does not exceed 5 units *)
        then
          ∀ i ∈ [j, k] :  l_li  = l_max
          (* then apply left justification of labels *)
        else
          ∀ i ∈ [j, k] :  l_li  =  (l_max - l_li)spaces + l_li
          (* else apply right justification of labels *)
```

Fig. 21. Justification heuristic.

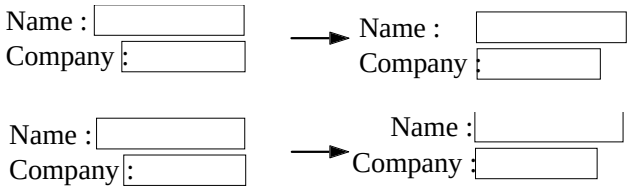


Fig. 22. Left and right justifications of labels.

5. Labels of IO could be left justified with labels of group boxes (fig. 23);

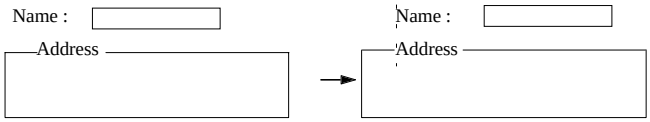


Fig. 23. Label placement heuristic.

6. Sequential group boxes could be proportionally equilibrated (fig. 24);

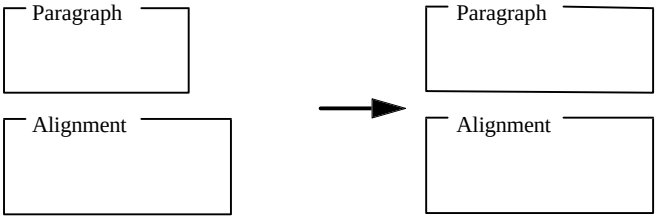


Fig. 24. Sequential group boxes heuristic.

Examples and Discussion

In order to illustrate how the right/bottom strategy works, let us show a hypothetical example with six IO and three push buttons to be laid out (fig. 25). Dimensions of these are : IO1= (8 ; 0,5), IO2=(12 ; 0,5), IO3=(6 ; 1,5), IO4=(10 ; 1), IO5 = (13 ; 0,5), IO6=(8 ; 2,5) ; Button : B1=(3 ; 1), B2=(3 ; 1), B3=(4 ; 1)

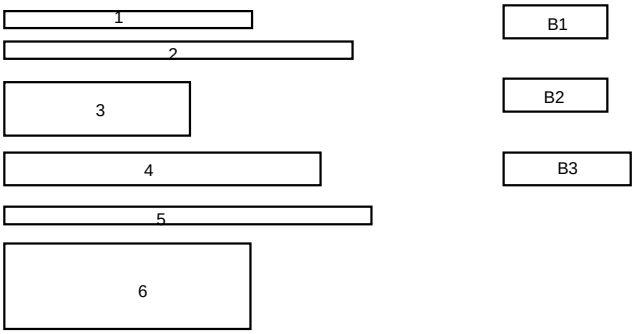
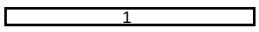


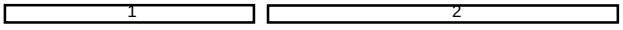
Fig. 25. Pool of IO in the example

We now build the state-space where 1 denotes a right placement, 2 denotes a bottom placement. (1.2.2) could be read as a right placement for IO1, a bottom placement for IO2, and a bottom placement again for IO3.

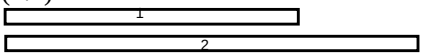
(1)



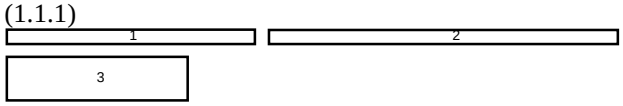
(1.1)



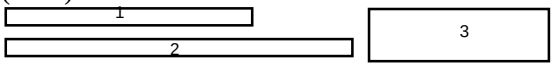
(1.2)



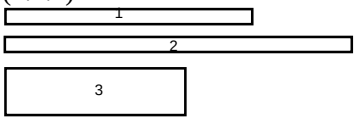
(1.1.1)



(1.1.2)

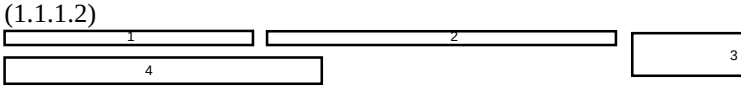


(1.2.2)

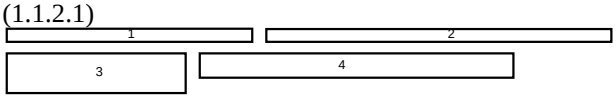


(1.1.1.1) $L_T > 33$. (* branch cut due to heuristic 1 *)

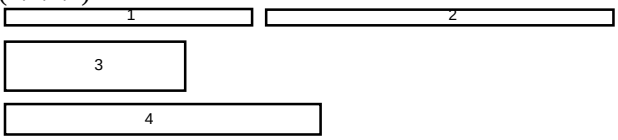
(1.1.1.2)



(1.1.2.1)

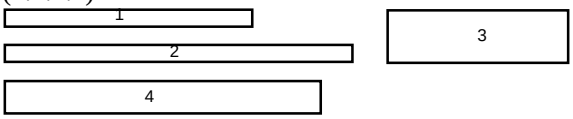


(1.1.2.2)



(1.2.1.1) $L_T > 33$. (* branch cut due to heuristic 1 *)

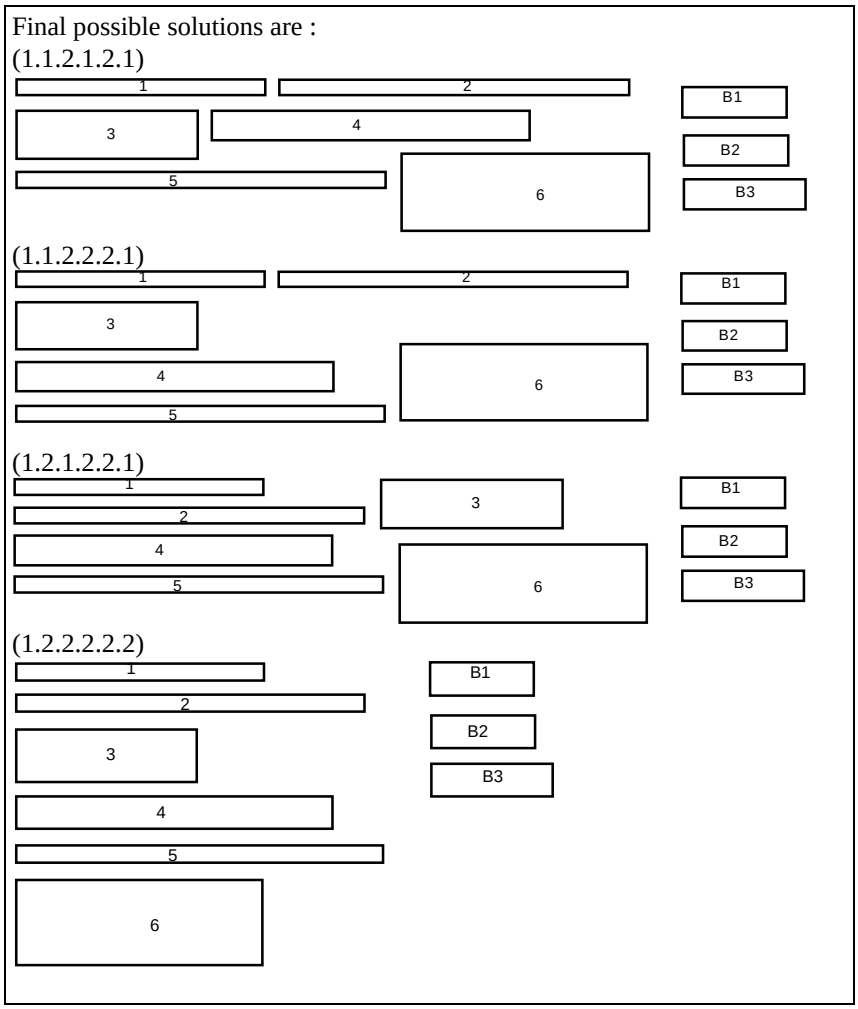
(1.2.1.2)



...

As we can see, the right/bottom strategy allows the designer to dynamically follow the layout of a PU. A computer-aided tool for displaying the different steps for this is welcome. Moreover, multiple solutions are offered to the designer, from which a final layout is kept. The last solution above (1.2.2.2.2.2) is the "vertical and left-pushed" layout that may be obtained with UIDE [3] and GENIUS [7].

Therefore, their strategy can be considered as a particular case of the right/bottom strategy. On the other hand, the right/bottom minimises unused blank spaces by condensing the placement of IO. This is a good point if the screen density does not become huge. In order to control this density, additional dynamic heuristics should be provided. Of course, new heuristics might always be supplemented, but this requires an important job resulting from intensive testing with examples. The cost for obtaining this expertise is significant and not negligible.



The final result of the example is depicted in fig. 25.
The right/bottom strategy is quite the opposite of the two-column strategy : the first is only driven by user-definable heuristics, whereas the former is unmodifiable. In each case, it is already been proved [3,4,11] that a direct-manipulation graphical editor should provide complete facilities for tailoring the final layout of the PU. The output of the generation may be considered as a fair starting input for manual refinement. This task can also be computer-aided supported to avoid any catastrophe of the designer. For instance, the designer should be not allowed to destroy some parts that are already well placed.

CONCLUSION AND FUTURE WORKS

Experiments with Static and Dynamic Strategies

What could be interesting is to automatically evaluate each result of the two strategies according to different metrics. The two-column strategy may receive a LA-score (defined by Sears [10]) which is worse than the right/bottom strategy. It is more likely that the right/bottom strategy provides a layout which should be quasi-LA-optimal since it tries to follow the most frequent sequence among IO and to minimize blank space and gaps between groups.

But this conclusion seems not to be obvious. Having an optimal layout for one metric does not necessarily lead to an optimal solution for another metric. For instance, a layout generated by the two-column strategy shall benefit of a good score in measuring its symmetry, balance, and alignments. But it will certainly not receive the best score in measuring the Layout Appropriateness. Our opinion is that having a LA-optimal layout does not guarantee an aesthetic-optimal layout for increasing the subjective satisfaction of the user. The two-column strategy should gain a good score for the predictability too since its consistency is checked each time, whereas it regrets not to have a good score for the economy metric since remaining spaces are left.

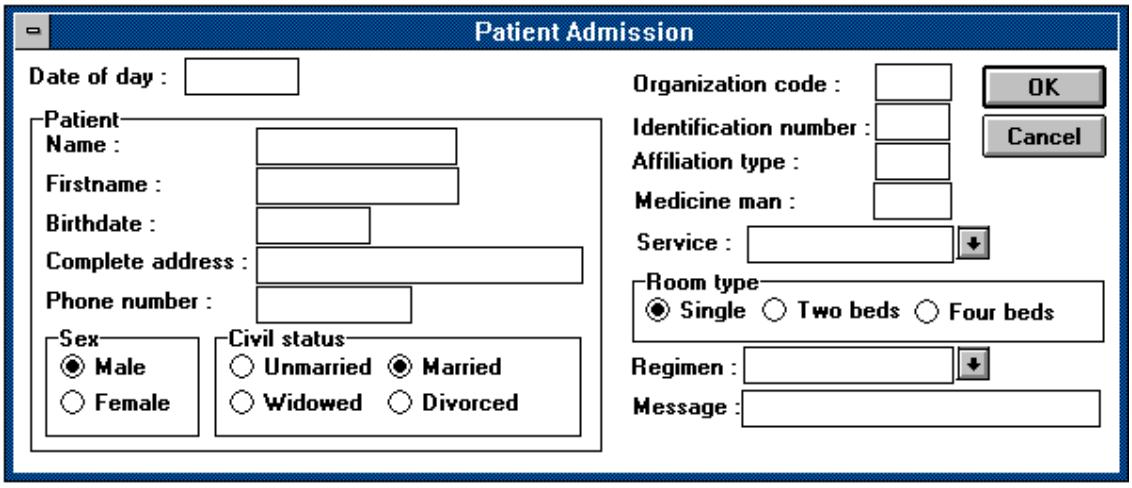


Figure 25. The example resulting from the Right/Bottom strategy.

Our final conclusion is that we have to look more at dynamic strategies than at static ones because they can be more suitable and adaptable to one particular environment by adjunction of heuristics. We also believe that future tools for computer-aided layout should encompass different strategies among which the designer may choose in order to satisfy one particular evaluation metric. For example, if the designer wants the best layout for minimising interaction time, the tool should generate a LA-optimal layout ; if the designer strives for consistency, the tool should generate a clean two-column layout ; if the designer prefers a compact layout, the tool should allow a right-bottom optimal layout ; if the designer specifies no necessary information ordering, the tool should propose a shape-based strategy (as in DON [8]). Each strategy has a different metric in mind.

We also can regret some limitations of layout evaluation metrics :

- they are restricted to the presentation part : if they were able to deal with the conversation, it would be possible to evaluate the appropriateness of some user actions on the layout (e.g. double clicking, IO access by accelerator, contextual menus, control keys, shortcuts);
- they usually cover only one composite IO (e.g. a screen, a window, a dialog box). Layouts of multiple composite IO are difficult to evaluate together. For instance, this situation occurs when the user manages IO belonging to different partially overlapping windows. LA metric may be a good starting point for extension.

Placement as an Optimisation Problem

The placement problem can also be considered as an optimisation mathematical problem. These algorithms typically define the problem of minimising unused space in a layout by optimal placement of rectangular pieces. Solving the equations potentially lead to a solution which maximises the layout space, but without taking care of the information ordering. Obviously, all IO are considered to have the same probability to be used. Therefore, IO are placed according to their dimensions, not according to the task sequence.

Generalisation to Non-rectangular Objects

Both demonstrated strategies are not miraculous, they deal with traditional layout. If one desires to generalise the layout problem for non-rectangular shapes (e.g. circle, ellipse, isosceles triangle, rectangular triangle, equilateral triangle), both two-column and right-bottom strategies will fail. This problem particularly occurs when dealing with multimedia applications where objects are no longer rectangular. Of course, the right-bottom strategy can be adapted by replacing each non-rectangular object by its smallest convex envelope or rectangle. The strategy should then work as well, but will certainly generates extra unwanted space. Using kerning techniques (as in typography) for better object placement and imbrication will not solve basically the layout problem. In fact, the layout problem in this case no longer consists of a layout grid, but a layout frame with oblique, non linear, curvilinear lines rather than only vertical and horizontal lines.

ACKNOWLEDGMENTS

The authors would like to thank Isabelle Provot, Xavier Gillo, and Pascal Beaujeant for their participation and support. This work was partially supported by the FIRST research program, Ref. RASE/SCHL319/Conv. 1487 and by the "Informatique du Futur" project under contract N°IT/IF/1. Any opinions, findings, conclusions or recommendations expressed in this paper are those of the authors, and do not necessarily reflect the view of the Belgian Government.

REFERENCES

1. Bodart, F. and Vanderdonckt, J. (1994). Guide ergonomique de la présentation des applications hautement interactives. Namur: Presses Universitaires de Namur.
2. Card, S., Moran, T., and Newell, A. (1983). The Psychology of Human-Computer Interaction, Hillsdale: Lawrence Erlbaum.
3. de Baar, D., Foley, J.D., and Mullet, K.E. (1992). Coupling Application Design and User Interface Design. In *Proceedings of CHI'92*, New York: ACM Press, pp. 259-266.
4. Feiner, S. (1988). A Grid-Based approach to Automating Display Layout. In *Proceedings of Graphics Interface'88*, pp. 192-197.
5. Horton, W.K. (1993). Illustrating Computer Documentation—The Art of Presenting Information Graphically on Paper and Online, New York: John Wiley & Sons.
6. Hurlburt, A. (1978). Layout: The Design of the Printed Page, New York: Watson-Guptill Publishing.
7. Janssen, C., Weisbecker, A., and Ziegler, J. (1993). Generating User Interfaces from Data Models and Dialogue Net Specifications. In *Proceedings of InterCHI'93*, New York: ACM Press, pp. 418-423.
8. Kim, W.C. and Foley, J.D. (1993). Providing High-level Control and Expert Assistance in the User Interface Presentation Design. In *Proceedings of InterCHI'93*, New York: ACM Press, pp. 430-437.
9. Marcus, A. (1992). Graphic Design for Electronic Documents and User Interfaces, New York: ACM Press.
10. Sears, A. (1993). Layout Appropriateness: A metric for evaluating user interface widget layout, *IEEE Transactions on Software Engineering*, Vol. 19, No. 7, pp. 707-719.
11. Vanderdonckt, J. and Bodart, F. (1993). Encapsulating Knowledge for Intelligent Automatic Interaction Objects Selection. In *Proceedings of InterCHI'93*, New York: ACM Press, pp. 424-429.
12. Vanderdonckt, J. and Gillo, X. (1994). Visual Techniques for Traditional and Multimedia Layouts, in this AVI'94 Proceedings.