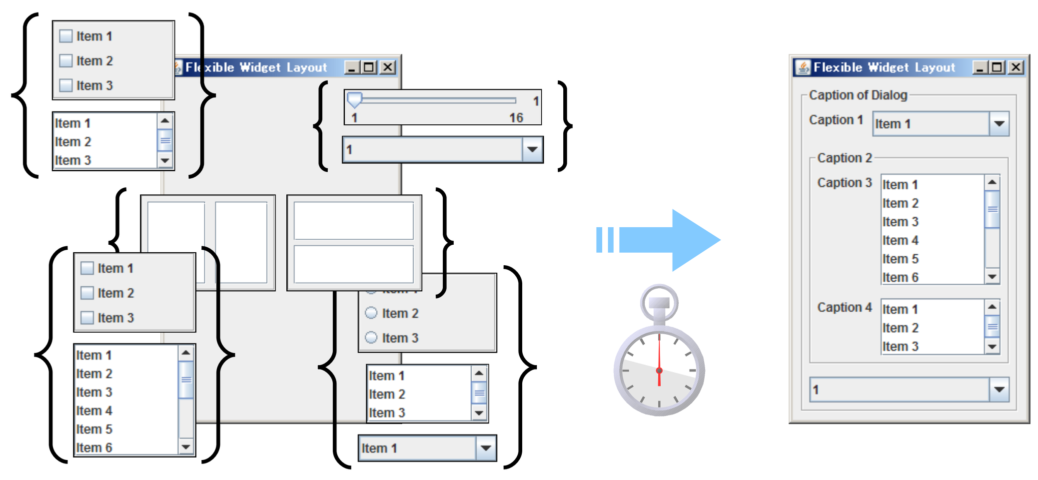
Improved Formulation of Flexible Widget Layout

Takuto Yanagida and Hidetoshi Nonaka

Graduate School of Information Science and Technology
Hokkaido University, Sapporo 060-0814, Japan
Tel: +81-11-706-6861, Fax: +81-11-706-6861
E-mail: {takty, nonaka}@main.ist.hokudai.ac.jp

http://aiwwww.main.ist.hokudai.ac.jp/~takty/index.en.html

- Automatic widget layout is one of the challenges for dynamic generation of graphical user interfaces (GUIs).
- In the field of model-based user interface (UI) design, systems generate GUIs from logical specification descriptions, which do not specify widgets.
- The flexible widget layout (FWL) is the automatic GUI generation requires both
  1. deciding which widget are used,
  2. completing the layout immediately especially when the system does this at run time.



- In our previous work, we have formulated FWL problem as a fuzzy constraint satisfaction problem (FCSP) a method for solving the problem.
  - However, its domains are not statically decided, and dynamically change when searching solutions.
- In this presentation, we improve our previous work so that the formulation coincides more strictly with FCSP using the binarization of n-ary constraints.

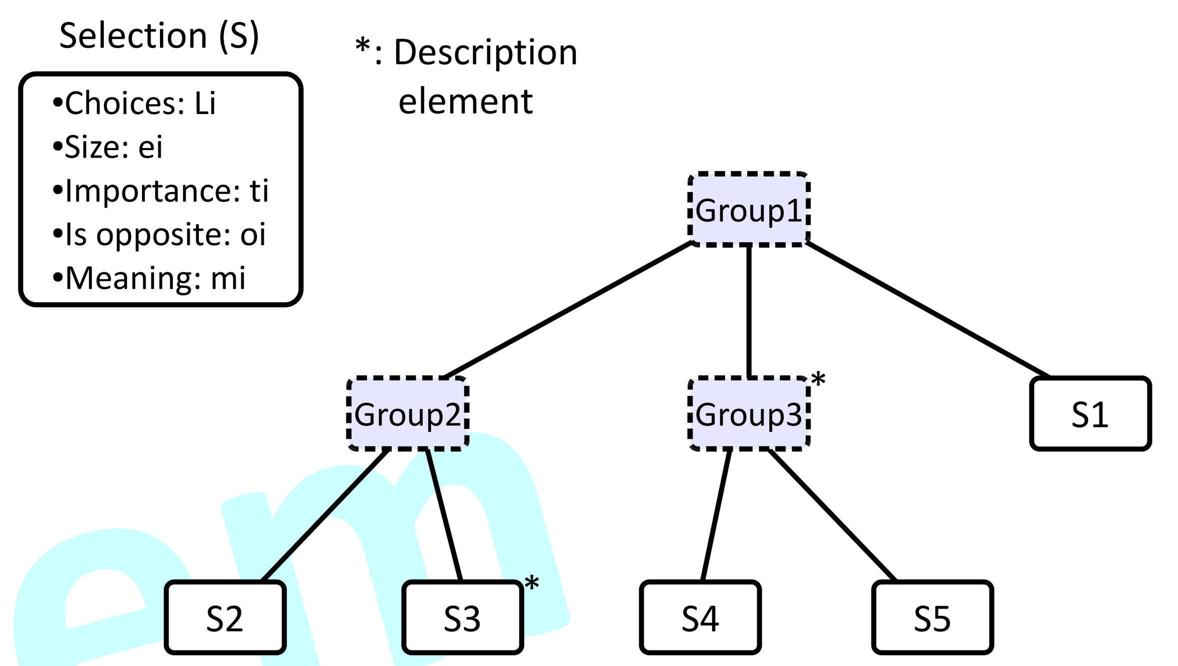
GUI Layout Problem Solved As Constraint Satisfaction

FWL Problem

- The FWL problem is a solution search problem for finding better combinations of widgets.
- Each widget is selected from a widget candidate set, which contains widgets representing the same UI function, but having different size and desirability.
- The complexity of FWL is caused by that widgets with the trade-off between their desirability α and the ease of layout involving their dimensions.

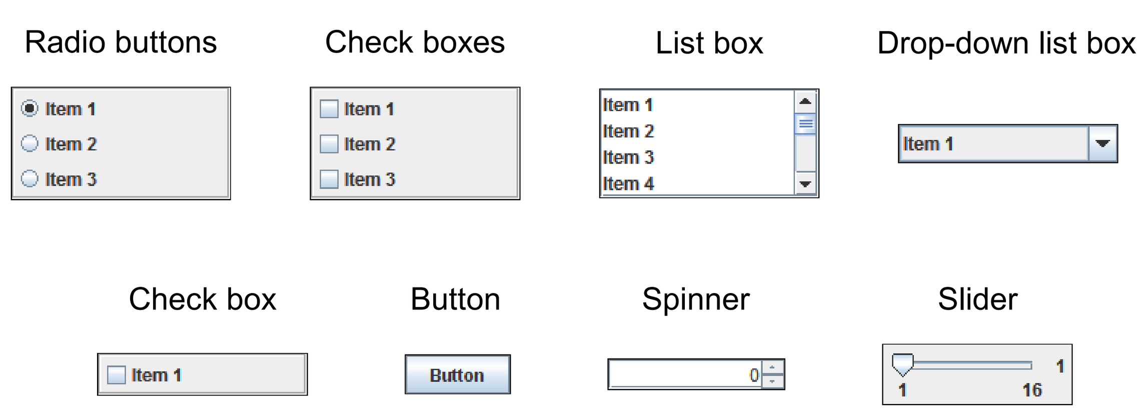
User Interface Model

- As a UI model generally expressed in logical descriptions, we adopt selection act model.
- A set of UI elements is expressed as $U = U_S \cup U_G \cup U_D$, where U_S, U_G, and U_D denotes respectively selection, group, and description elements.
- In this model, selection elements are represented as selection acts with some parameters, and they are grouped to make a tree graph.

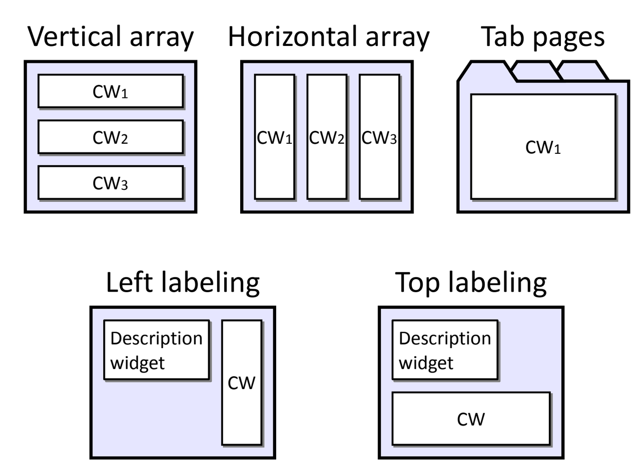


Model and Widgets

- The UI elements are represented as widgets $W = W_N \cup W_C$, normal widgets and container widgets.
- The UI elements are mapped to corresponding widget candidate sets.
  - Selection elements and description elements are mapped to a set of normal widget candidates $W_i \subset W_N$
  - Group elements and positioning of description elements are mapped to a set of container widget candidates $W_i \subset W_C$.
- As normal widgets, we use eight widely-used widgets for representing selection elements; and caption label and abbr. label for description elements.
  - The desirability (usability) $\alpha \in [0, 1]$ is defined.



- As container widgets, we use the three widgets for representing group elements; and the other two widgets for the positioning of description elements.
  - The desirability (usability) $\alpha \in [0, 1]$ is defined.



Formulation

Variables

- Variable $x_i \in X$ corresponds to widget candidate set W_i and the value assigned in it expresses a selected candidate from the set.
- X is divided into X_N and X_C, which express the variable sets for the normal and container widget candidates respectively.
- Widget candidate sets of selections, groups, descriptions, and the positioning of the descriptions, are expressed with the variables.

Domains

- The values of domains are tuples calculated from the bottom to the top of the tree structure of the variables.
  - The domain of $x_i \subset X_N$ is a set of the tuples:

$$D_i(\in D_N) = \{\langle w, ms_w \rangle | w \in W_i \subset W_N\},$$

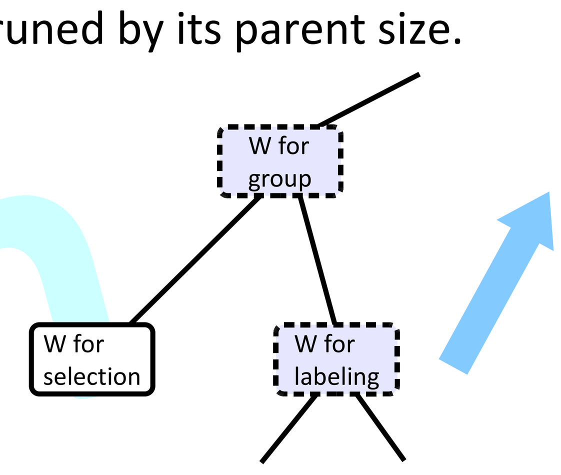
$ms_w = \langle ms.width_w, ms.height_w \rangle$ is the minimum size of w

  - The domain of $x_i \subset X_C$ is a set of the tuples:

$$D_i(\in D_C) = \{\langle w, M, ms_{w,M} \rangle \ \ w \in W_i \subset W_C, \ M \in D_{\text{child }(i,1)} \times \cdots \times D_{\text{child }(i,\text{cn}_i)}, \text{checksize}(W_i, ms_{w,M}) \},$$

M is a combination of values of child widget candidates, child(i, j) is the function for obtaining the index of jth child of W_i, cn_i is the number of children of W_i, and checksize(W_i, ms) is the function which checks whether the combination of its parameters is available or not.

- For the domain of a container,
  - The minimum sizes of all combinations of its child elements ($ms_{w,M}$) are calculated from bottom to top.
  - To avoid the combinatorial explosion, the minimum sizes are pruned by its parent size.



- The minimum sizes of containers are calculated by the minimum sizes of their child elements.

Widgets for group

$$\begin{cases} ms_{\text{VA} \in W_i} = \langle \max_j \left(ms.width_{w_{i,j}}\right), \sum_j ms.height_{w_{i,j}} \rangle, \\ ms_{\text{HA} \in W_i} = \langle \sum_j ms.width_{w_{i,j}}, \max_j \left(ms.height_{w_{i,j}}\right) \rangle, \\ ms_{\text{TP} \in W_i} = \langle \max_j \left(ms.width_{w_{i,j}}\right), \max_j \left(ms.height_{w_{i,j}}\right) \rangle. \end{cases}$$

Widgets for Labeling

$$\begin{cases} ms_{\text{LL} \in W_i} = \langle ms.width_{w_{i,\text{D}}} + ms.width_{w_{i,\text{C}}}, \max\left(ms.height_{w_{i,\text{D}}}, ms.height_{w_{i,\text{C}}}\right) \rangle, \\ ms_{\text{TL} \in W_i} = \langle \max\left(ms.width_{w_{i,\text{D}}}, ms.width_{w_{i,\text{C}}}\right), ms.height_{w_{i,\text{D}}} + ms.height_{w_{i,\text{C}}} \rangle. \end{cases}$$

Constraints

- Unary constraint $c_k \in C_D$ denotes the desirability of the value of its scope x_{k1} as their satisfaction degrees.

If the scope of c_k is $S_k = \{x_{k1}\}$ and the value of x_{k1} is v ($\in D_{k1}$) = $\langle w, \ldots \rangle$, $w \in W_{k1}$, the satisfaction degree of c_k is calculated as follows:

$$c_k(v)(\in C_D) = des(w)$$

(des is the projection from widgets to their desirability)

- Binary constraint $c_k \in C_P$ denotes whether the values of the variables of its scope correspond with each other.

If the scope of c_k is $S_k = \{x_{k1}, x_{k2}\}$, the value of x_{k1} is v_p ($\in D_{k1}$) = $\langle w, M, ms_w \rangle$, and the value of x_{k2} is $v_c \in D_{k2}$, the satisfaction degree of c_k (v_p, v_c) is calculated as follows:

$$c_k(v_p, v_c)(\in C_P) = \begin{cases} 1 & \text{if } v_c = M\left[\text{childindex}(x_{k1}, x_{k2})\right] \\ 0 & \text{otherwise} \end{cases}.$$

(childindex(x_1, x_2) is the projection from variable pairs to the index of the widget candidates as a child)