Personalizing Graphical User Interfaces on Flexible Widget Layout

Takuto Yanagida, Hidetoshi Nonaka, and Masahito Kurihara

Hokkaido University, Japan

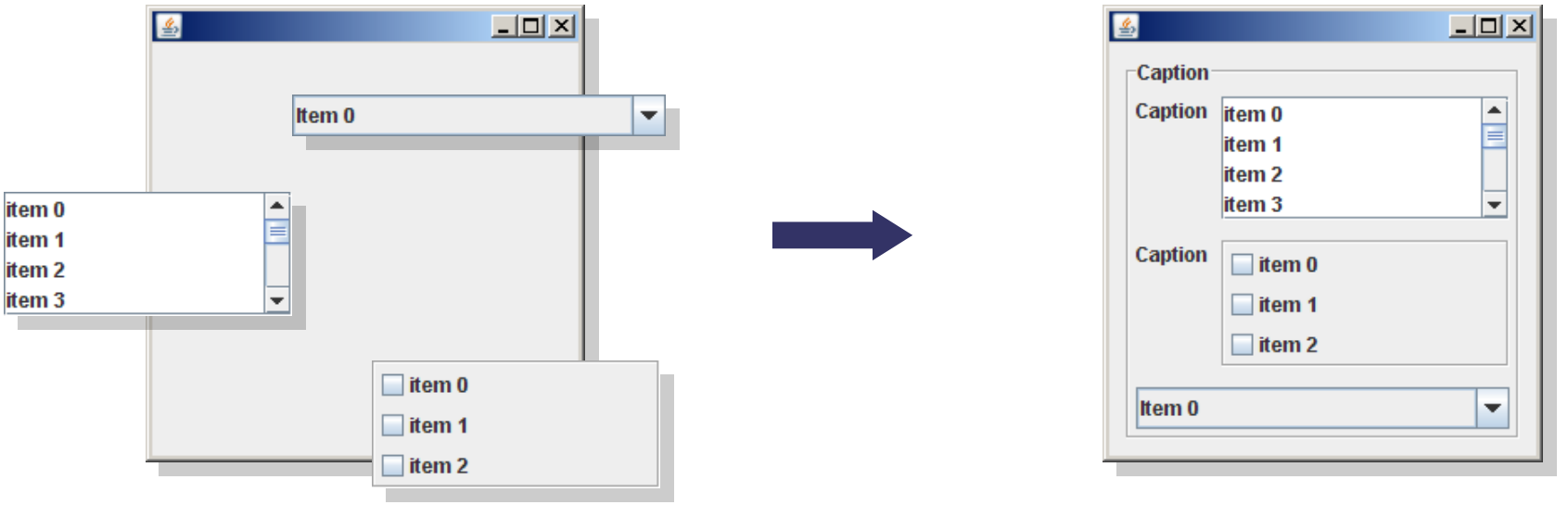
Contents

1. Demonstration
   - The flexible widget layout system [27]
     (An implementation in Java)
2. Flexible widget layout [a]
3. Personalization

[27] Demo of flexible widget layout
http://kussharo.complex.eng.hokudai.ac.jp/~takty/demo/fwl.en.html
[a] T. Yanagida and H. Nonaka. Flexible Widget Layout Formulated as Fuzzy Constraint Satisfaction Problem. In Proc. of KES IDT 2009, 2009.

Widget layouts

- The process of deciding positions and sizes of widgets (list boxes, radio buttons, and panels)



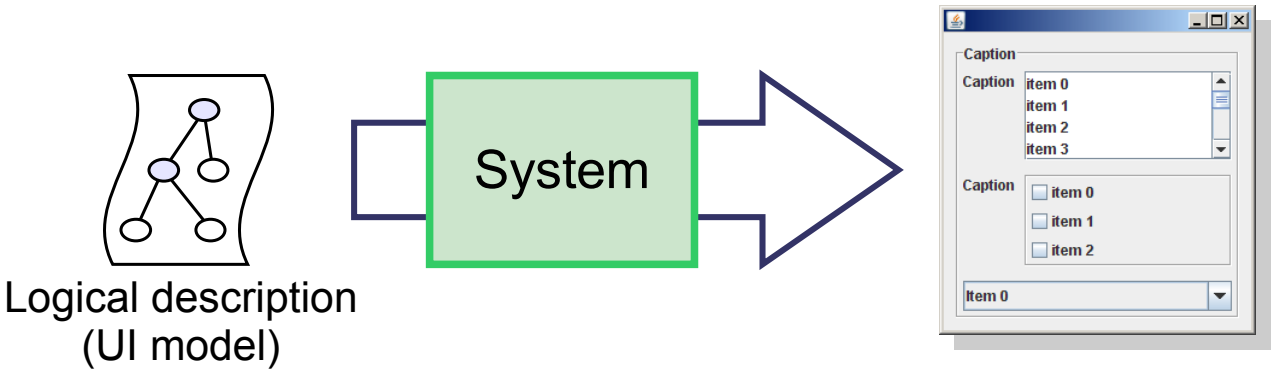
The layout has a significant impact on the usability of tasks which can be accomplished with GUIs.

Model-based UI design

- In the field of model-based UI design
  - Systems generate UIs from logical descriptions.

> **Logical descriptions** (UI models)
>
> - specifying **UI functions** independently of platforms, instead of specifying widgets.

  - It is useful for realizing the diversity of UIs.



Widget layouts + model-based UI

- In the field of model-based UI design
  - A layout system needs **to select widgets** before creating a layout.
  - In addition, widgets are sometimes not uniquely determined.

A system could select small widgets of inferior usability for small screens, or large ones of sufficient usability for large screens.

→ Related studies on how to generate GUIs

Related work (1/2)

- Design time layout systems
  - An adaptive algorithm for automated UI design [5]
  - An approach using mathematical relationships [2]
- Dynamic layout
  - GADGET [9]
  - SUPPLE [11]

[5] J. Eisenstein, A. Puerta, and R. Software. Adaption in automated user-interface design. In Proc. of IUI 2000, 2000.
[2] F. Bodart, A.-M. Hennebert, J.-M. Leheureux, and J. Vanderdonckt. Towards a dynamic strategy for computer-aided visual placement. In Proc. of AVI ’94, pp. 78–87, Italy, 1994.
[9] J. Fogarty and S. E. Hudson. Gadget: a toolkit for optimization-based approaches to interface and display generation. In Proc. of UIST ’03, pp. 125–134, Canada, 2003.
[11] K. Gajos and D. S. Weld. SUPPLE: automatically generating user interfaces. In Proc. of IUI ’04, pp. 93–100, Portugal, 2004.

Related work (2/2)

- Plasticity of widgets
  - Handling widget selections as plasticity [3]
  - The graceful degradation [8]
  - An intelligent editor for GUIs [4]
- Other studies
  - A lot of studies for the LSI or VLSI layout problem
  - Existing *layout managers* offered by GUI toolkits

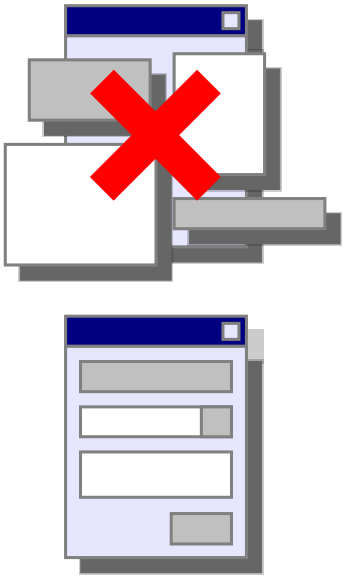
[3] G. Calvary, J. Coutaz, D. Thevenin, Q. Limbourg, L. Bouillon, and J. Vanderdonckt. A unifying reference framework for multi-target user interfaces. Interacting with Computers, 15:289–308, 2003.
[8] M. Florins and J. Vanderdonckt. Graceful degradation of user interfaces as a design method for multiplatform systems. In Proc. of IUI 2004, pp. 140–147, Portugal, 2004.
[4] B. Collignon, J. Vanderdonckt, and G. Calvary. An intelligent editor for multi-presentation user interfaces. In Proc. of SAC 2008, pp. 1634–1641, Brazil, 2008.

Consideration (1/2)

- How to **select widgets** for UI functions?
  - General usability guidelines
  - Adaptation to **users** and **environments**

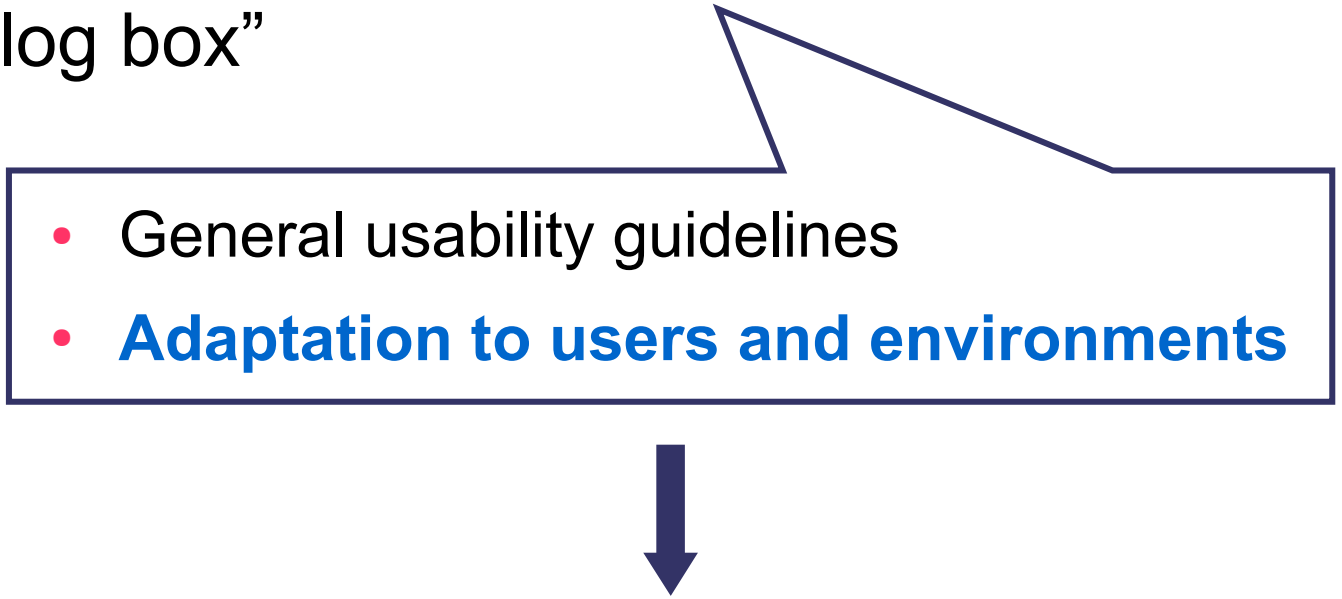
  } Viewpoint of **desirability**

- Tactics of widget selections
  - To select properly desirable ones
    - **Trade-off** between usability and ease of layout
    - All widget can be put inside a dialog box



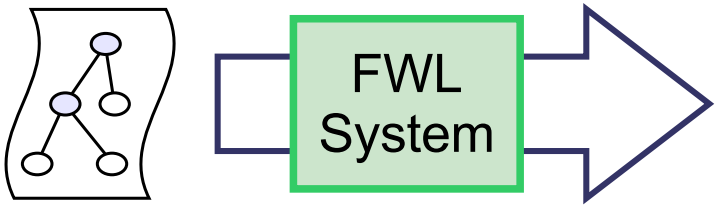
Consideration (2/2)

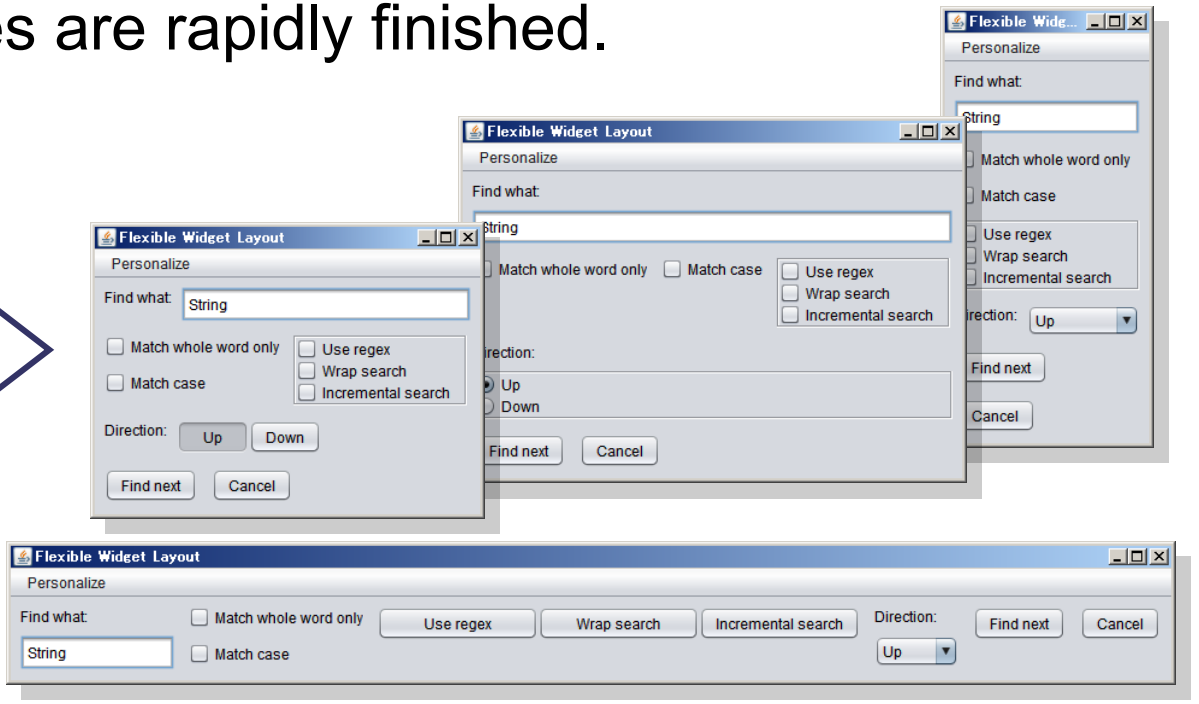
- GUI generations in model-based UI designs
  - “To select properly **desirable** widgets to be put in a dialog box”



Our method

- Flexible widget layout (FWL)
  - Automated GUI generation based on UI models
    - Widgets to be used are dynamically selected.
    - Layout processes are rapidly finished.

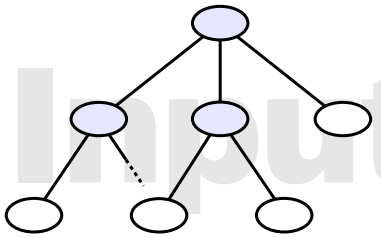




GUIs corresponding to the same UI model

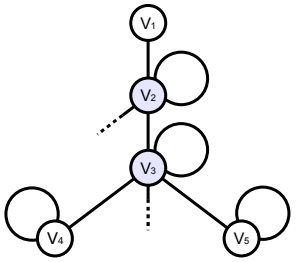
Phases of FWL

1. Generate an FCSP* from the given **model**



UI model

Phase 1



FCSP

2. Solve the FCSP to get combinations of widgets

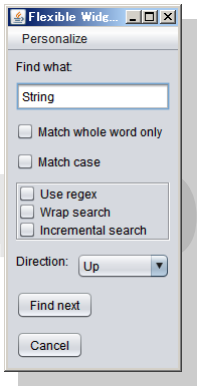
Phase 2

$V_1 = 3$
$V_2 = 1$
$V_3 = 0$
$V_4 = 3$
$V_5 = 2$
⋮

Assignments

3. Make a layout

Phase 3



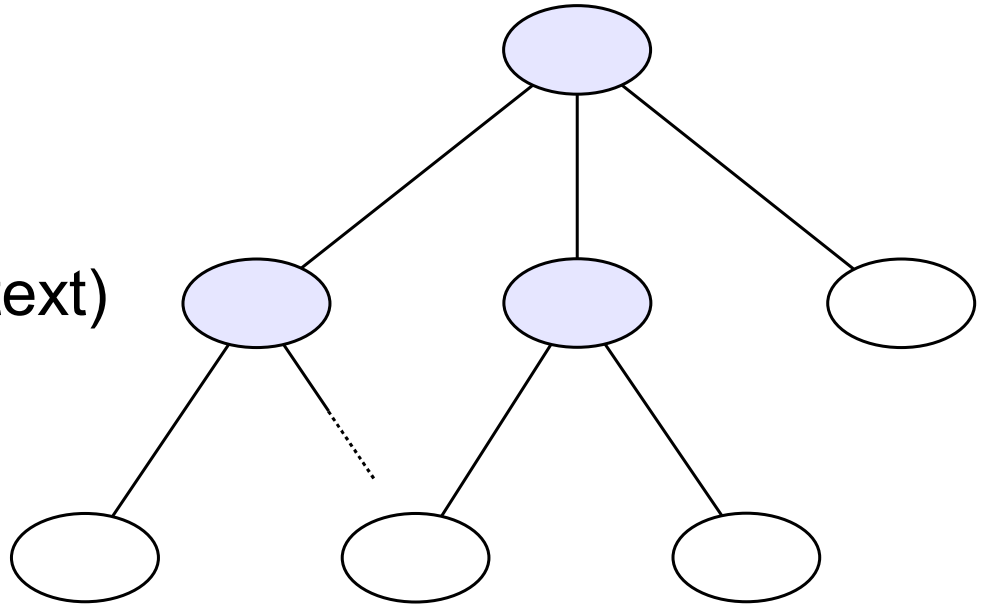
Output

Layout

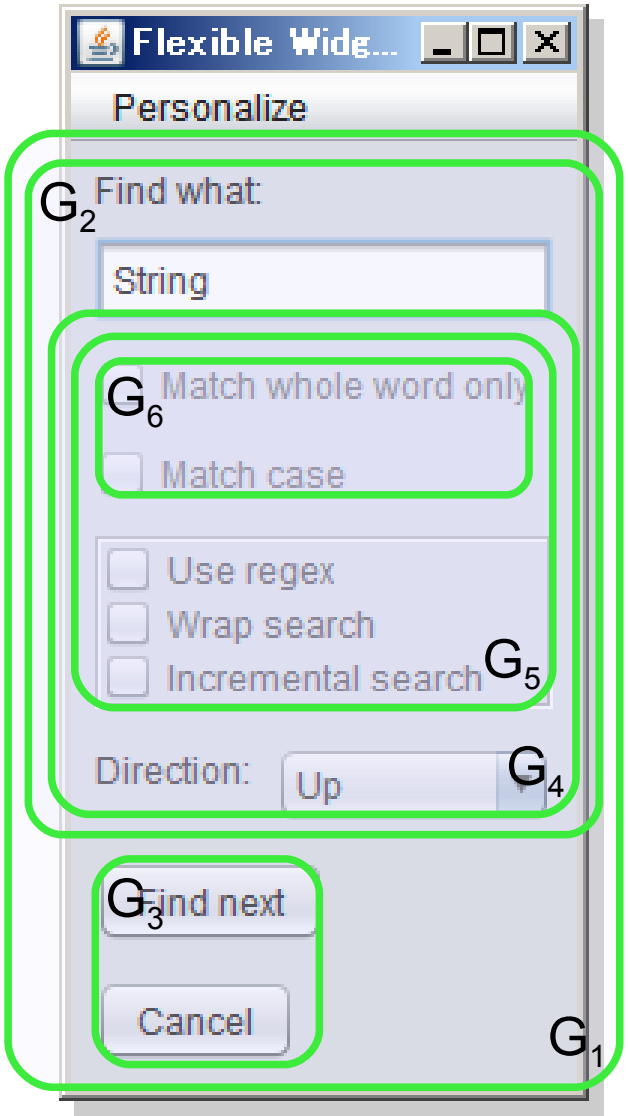
* Fuzzy constraint satisfaction problem: a method for modeling problem, which is offered in the field of artificial intelligence.

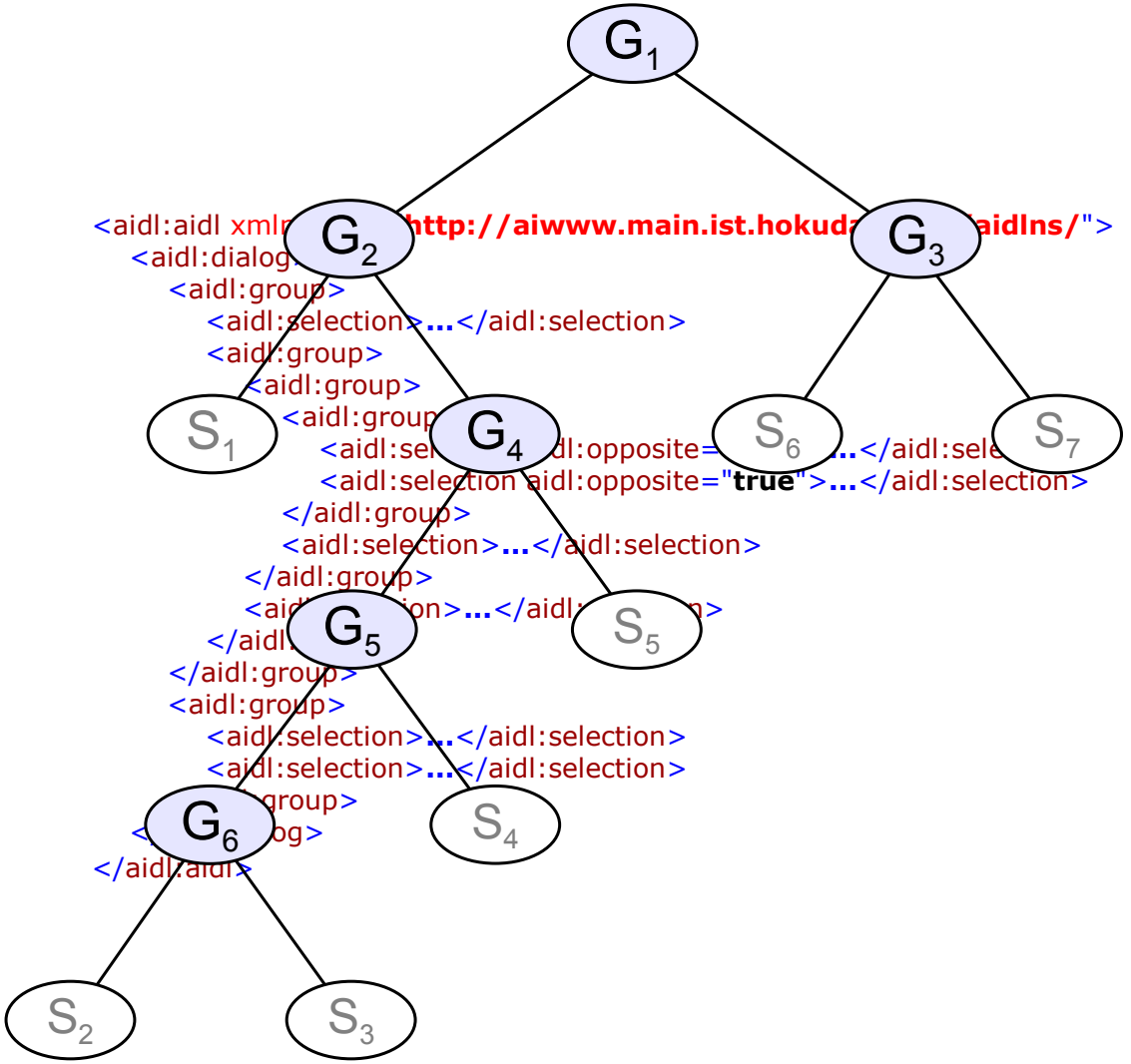
Logical description (UI model)

- Abstract interaction description language (AIDL)
  - UI function description based on selection act model
    - Selection elements (acts)
      - Choices
        - A set of choices
        - A type
    - Group elements
    - Description elements (text)

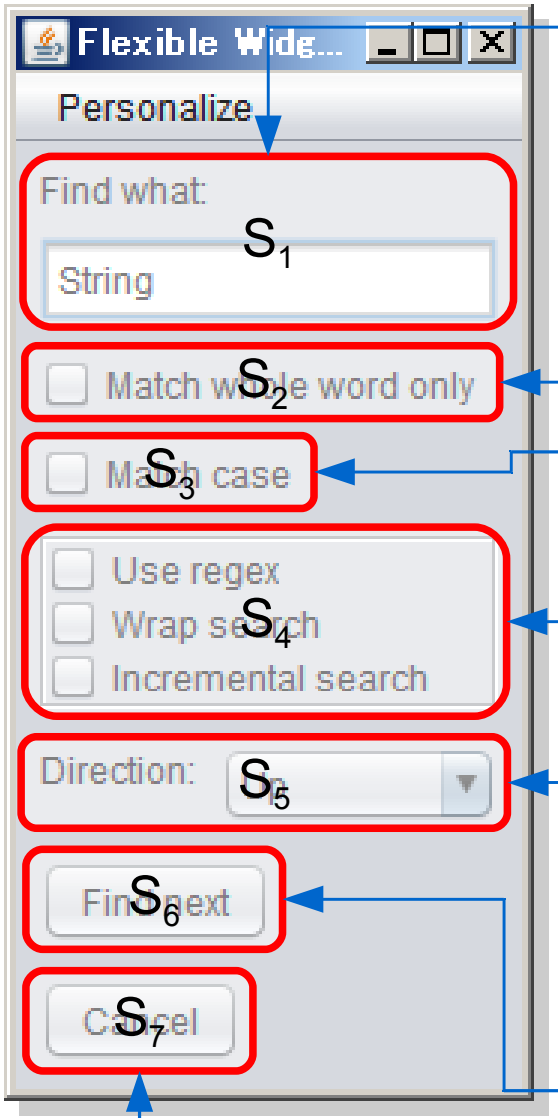


Tree structure of model





Each element




```
<aidl:selection>
   <aidl:description aidl:abbr="FW" aidl:caption="Find what" />
   <aidl:state>String</aidl:state>
   <aidl:strings />
</aidl:selection>
```

```
<aidl:selection aidl:opposite="true">
   <aidl:strings>
      <aidl:choice aidl:str="Match whole word only" />
      <aidl:choice aidl:str="Match partial word" />
   </aidl:strings>
</aidl:selection>
```

```
<aidl:selection aidl:opposite="true">
   <aidl:strings>
      <aidl:choice aidl:str="Match case" />
      <aidl:choice aidl:str="Match no case" />
   </aidl:strings>
</aidl:selection>
```

```
<aidl:selection>
   <aidl:strings>
      <aidl:choice aidl:str="Use regex" />
      <aidl:choice aidl:str="Wrap search" />
      <aidl:choice aidl:str="Incremental search" />
   </aidl:strings>
</aidl:selection>
```

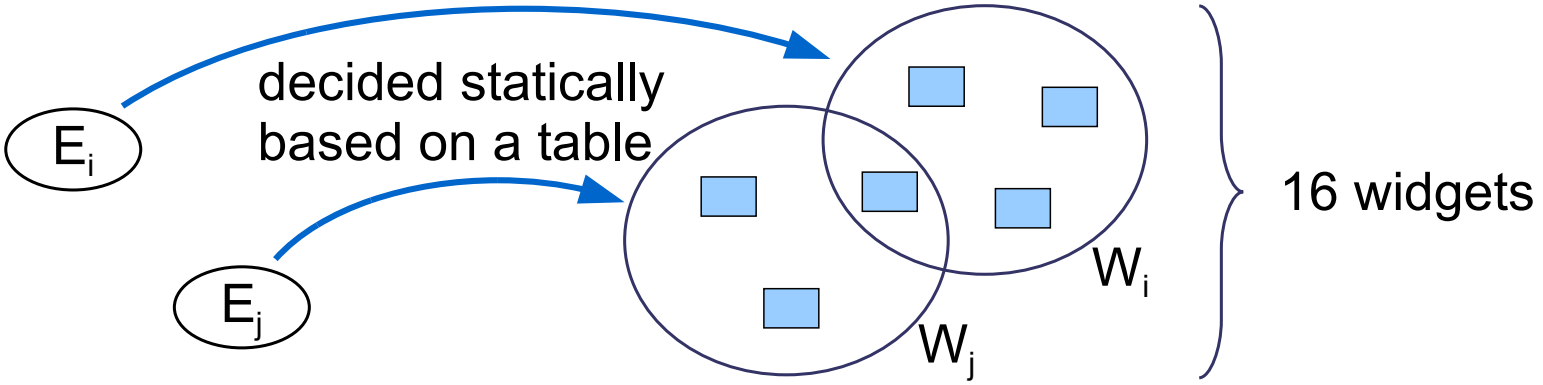
```
<aidl:selection>
   <aidl:description aidl:abbr="DIR" aidl:caption="Direction" />
   <aidl:strings>
      <aidl:choice aidl:str="Up" />
      <aidl:choice aidl:str="Down" />
   </aidl:strings>
</aidl:selection>
```

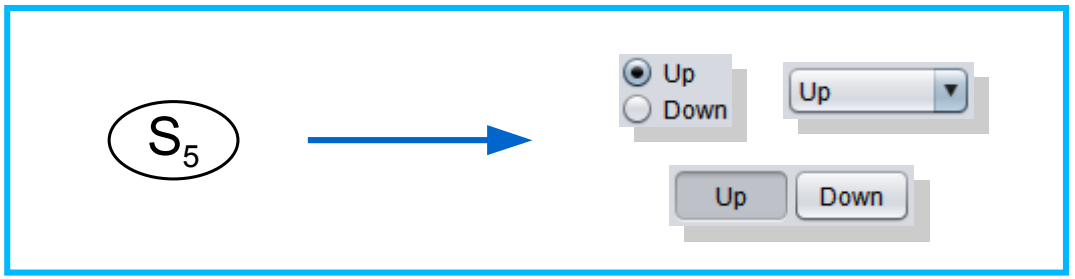
```
<aidl:selection><aidl:description aidl:caption="Find next" /></aidl:selection>
```

```
<aidl:selection><aidl:description aidl:caption="Cancel" /></aidl:selection>
```

Candidate sets

- Flexible widget layout problem
  - To determine **widget candidate sets**
    - Each element of the model is mapped to them.



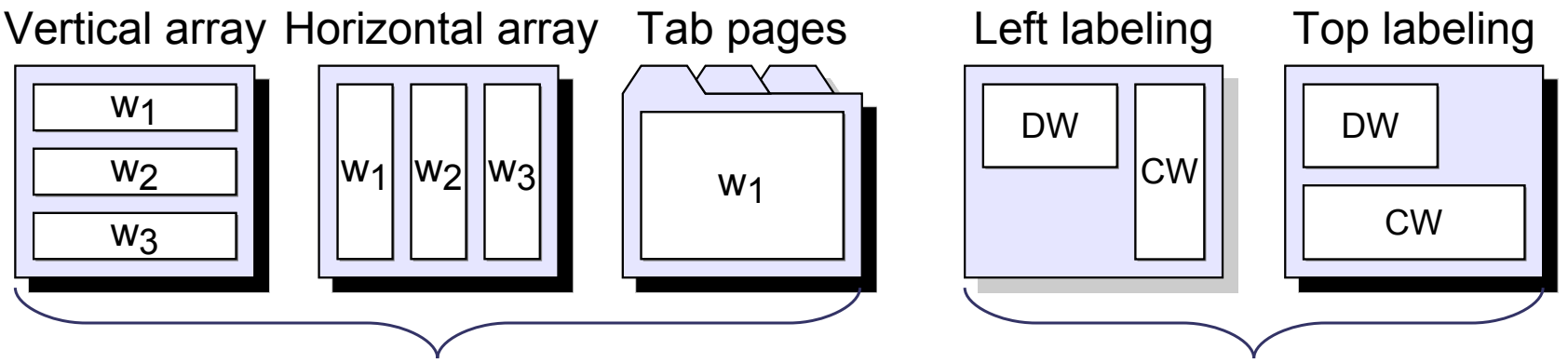


FWL (1/4)

- Container widgets (5 widgets)
  - **Selection of container widgets** expresses selection of positioning.
  - Group elements and *positioning of description*

⟷ Container widget candidate set $W_i \subset W_C$

$W_C =$



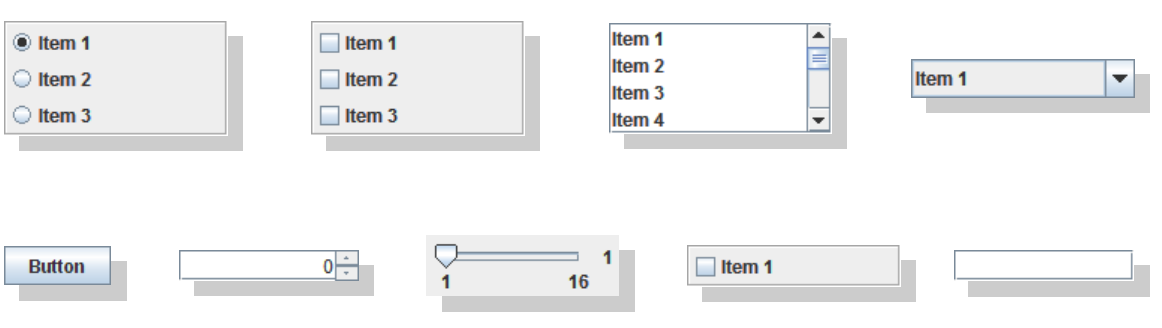
FWL (2/4)

- Normal widgets (11 widgets)
  - A subset adopted in many toolkits (9+2 types)
  - Selection and description elements

⟷ Normal widget candidate set $W_i \subset W_N$

$W_N =$

For selection elements



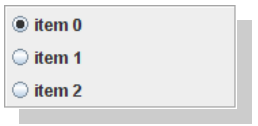
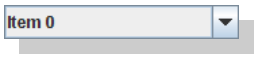
For description elements

Caption labels, Abbreviation labels

FWL (3/4)

- Properties of widgets
  - Minimum size: $ms_w = \langle ms.width_w, ms.height_w \rangle$
  - **Desirability** for each type: $0 \leq \alpha \leq 1$

    You can define it for each user (adaptation)
  - Trade-off between usability and ease of layout

	Radio buttons	Drop-down list box
Appearance		
UI function	Same	Same
The ease of layout	Worse	Better
Usability	Better	Worse

FWL (4/4)

- Layout rules (constraints on layouts)
  - Feasibility of Layout
    - Whether or not all widgets can be in a dialog box?

      → Must be satisfied
  - Desirability of layout
    - Minimum of desirability of selected widgets

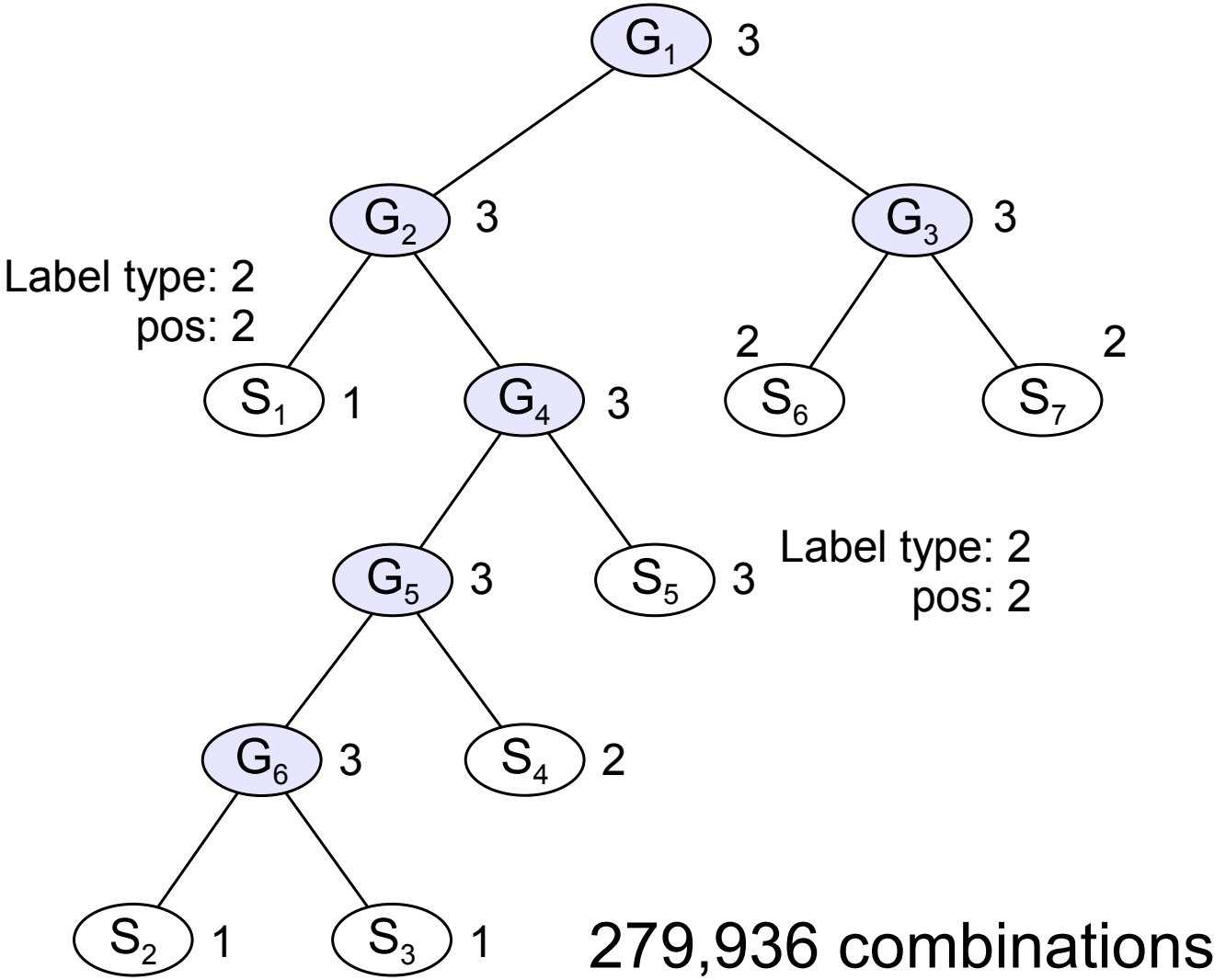
      → To be maximized

"A layout-able and desirable **solution**"

↓↓

A combination of widgets

Combination of widgets



Formulation (1/3)

- Flexible widget layout problem
  - Widget selections
  - Desirability of layout

  - Combinatorial search
  - Fuzzy constraints

Formulated as

Fuzzy constraint satisfaction problems (FCSPs)

- Combinatorial search problem that decides assignments to variables that **almost** satisfy all constraints among variables

Formulation (2/3)

- **Fuzzy** constraint satisfaction problem (FCSP)
  - a set of variables: $X = \{x_1, \ldots, x_m\}$
  - a set of domains: $D = \{D_1, \ldots, D_m\}$
  - a set of constraints: $C = \{c_1, \ldots, c_r\}$
    - c_k: **membership function** $\mu R_k(v[S_k])$ → **satisfaction degree**
    - S_k: scope (variables related to c_h)
    - v: assignment for all variables
  - **A solution of an FCSP**
    - An assignment v is a solution if $Cmin(v) > 0$.
    - The **minimum** of all constraint satisfaction degrees.

$$Cmin(v) = \min(\mu R_k(v[S_k]))$$

Formulation (3/3)

Sizes and positions of widgets are **NOT** represented directly as variables.

- Variables: widget selections by its assignment
- Domains: sets of widget candidates
- Constraints : desirability and parental relations

The scale of domains is reduced.

Personalization of GUIs

Personalization of GUIs on FWL

=

Problem: how to define the desirability?
- UIs for deciding desirability by users

↓

A method for customizing
fuzzy membership functions

Pairwise comparison method (1/2)

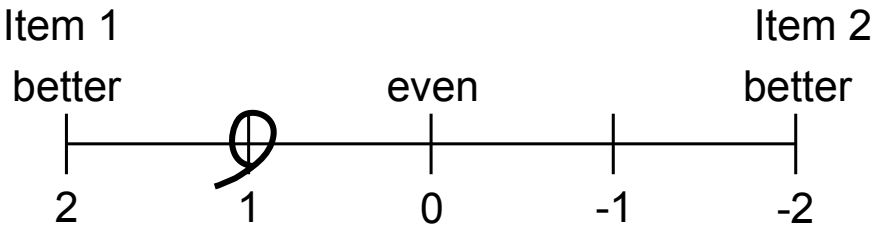
- Pairwise comparison method (PCM)
  - computes *weights* of elements by **comparing elements two by two** in a certain criterion
    - Thurstone's method

      ➡ Order of elements

    - **Scheffé's method**

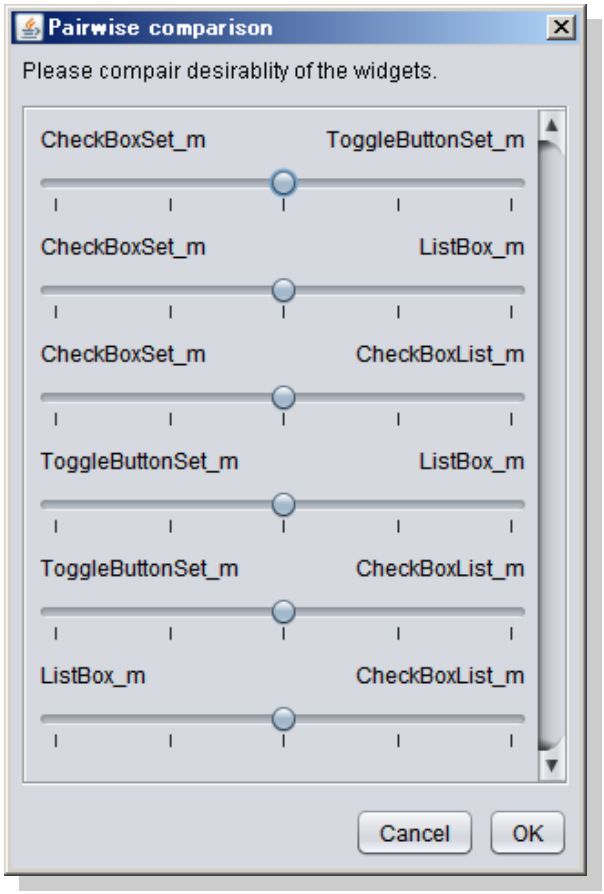
      ➡ Weights of elements

    - Analysis hierarchy process (AHP)

Pairwise comparison method (2/2)

- Implementation of PCM
  - `JPairwiseComparisonDialog`
    - The 3 methods
    - Sliders or pairs of toggled buttons to represent pairs of elements



To utilize PCM to calculate desirability of widgets...

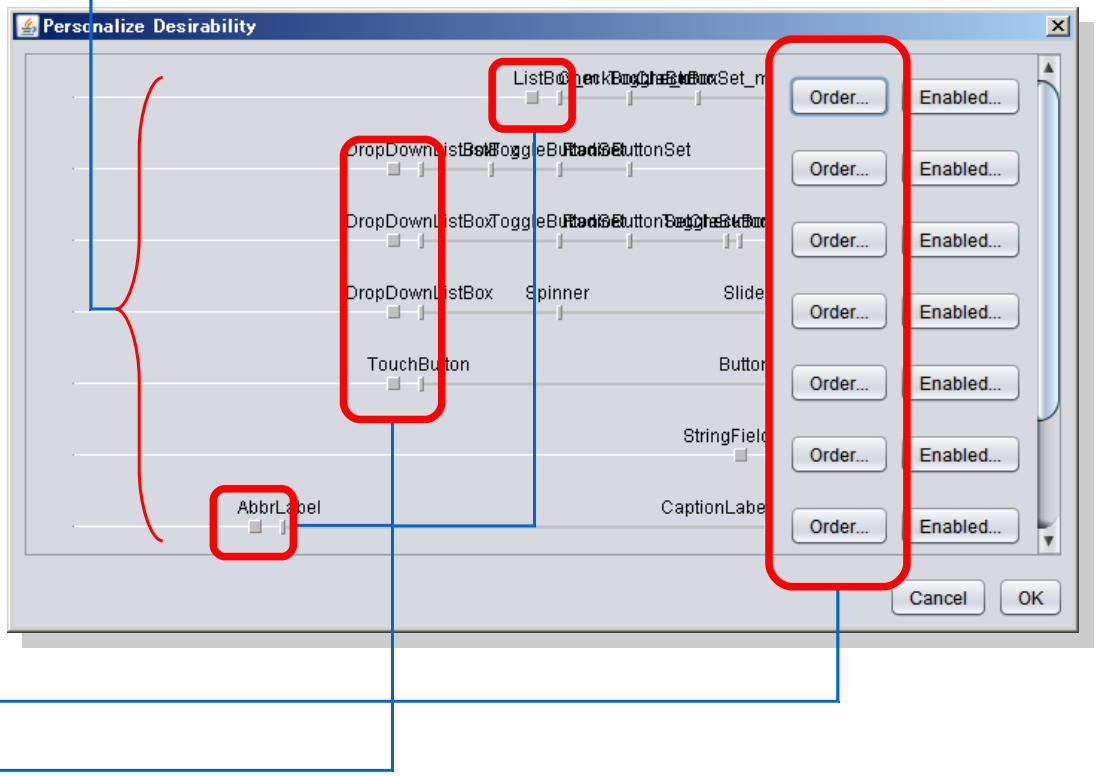
Problem of PCM

- Total order is required
  - Desirability is represented as degree of constraint satisfaction of FCSP
  - For calculating desirability of 16 widgets, 120 pairs need to be evaluated by users

Too many pairs to get a total order of desirability of widgets

Personalization dialog

- UI for evaluating subgroups of widgets
  - Separate widgets into some groups
  - Define the order of them with PCM
  - Marge the groups with wights



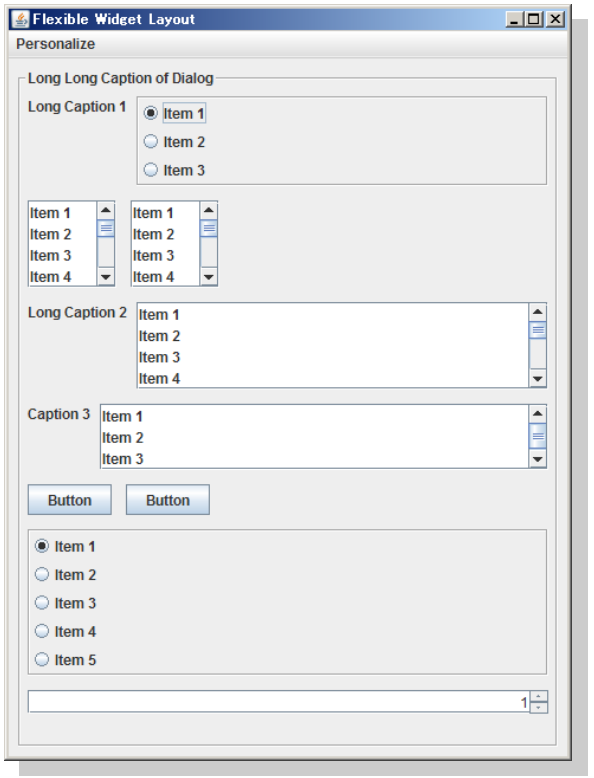
Preliminary experiment

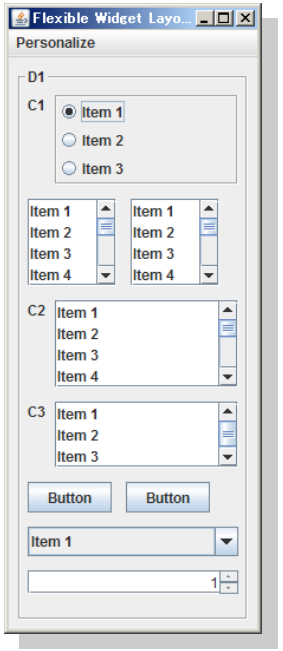
- To verify that our method actually personalize the FWL for each user
  - 4 subjects familiar with GUIs but not experts
  - A sample model
  - No situations specified

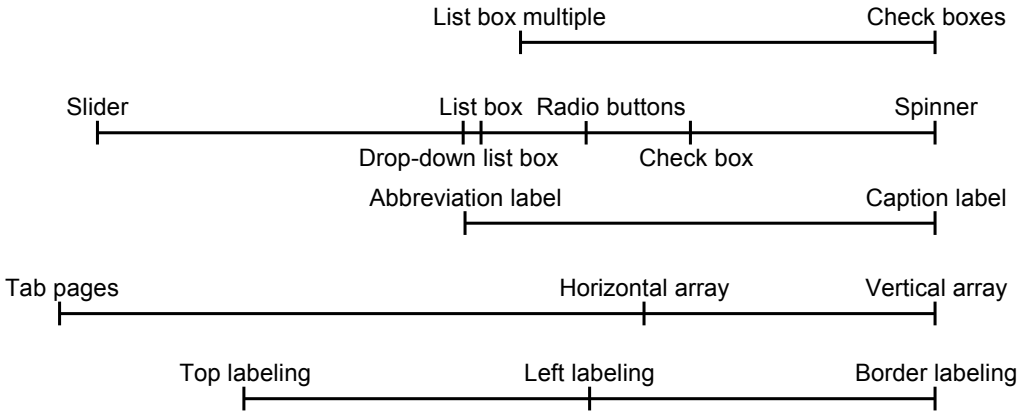
  - Subjects freely use our system for a while
  - Questionnaires

Results of personalization (1/4)

- Subject A



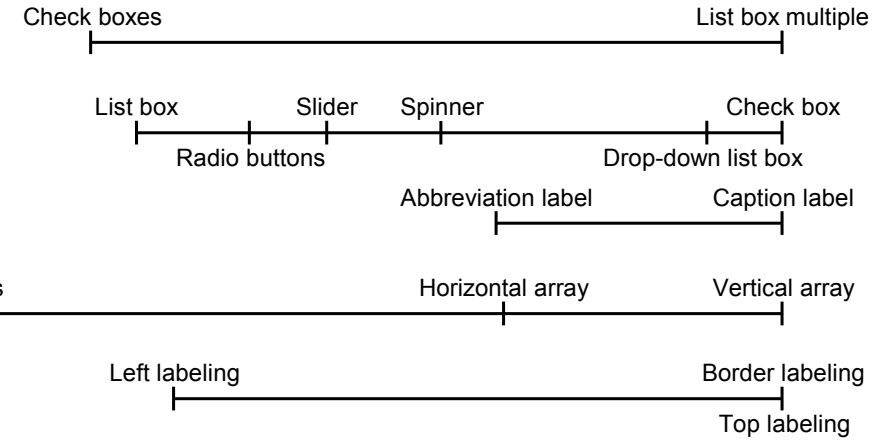


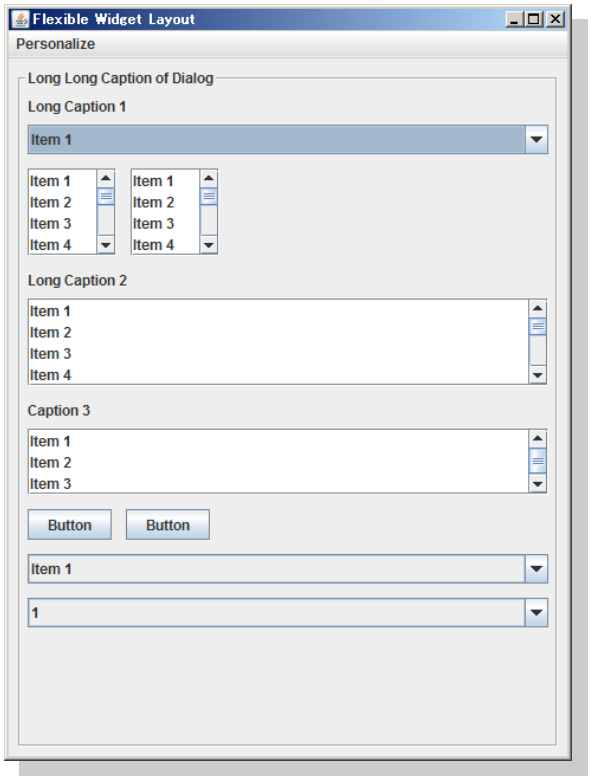


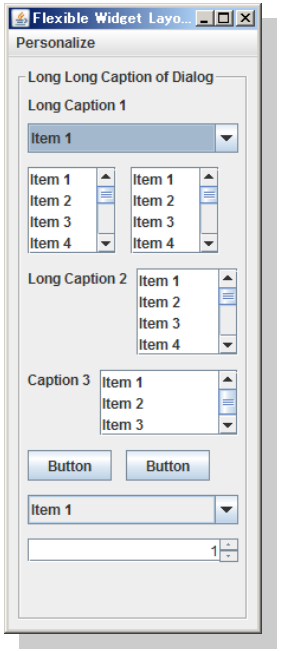
Assumption:
A kind of web forms usable by a mobile phone

Results of personalization (2/4)

- Subject B



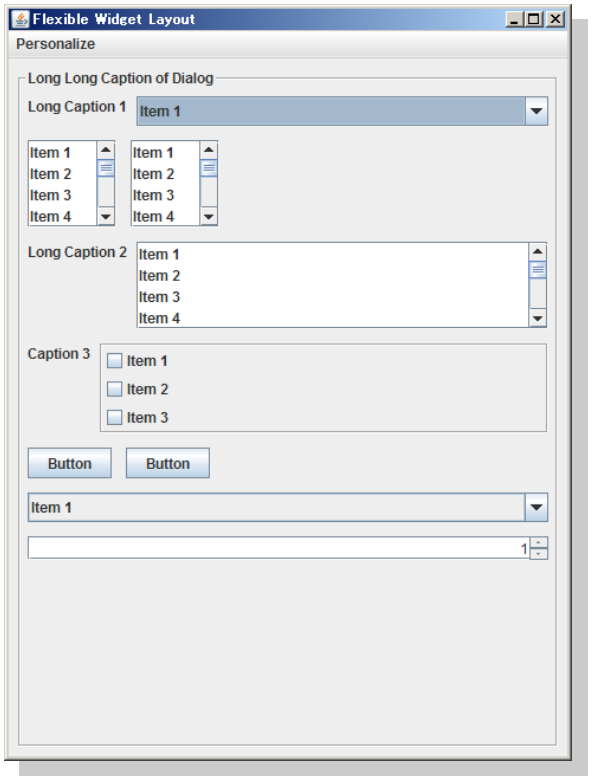


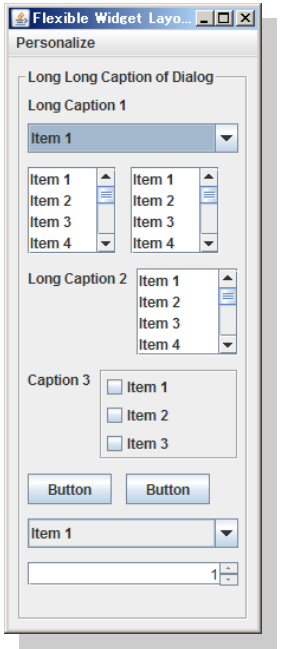


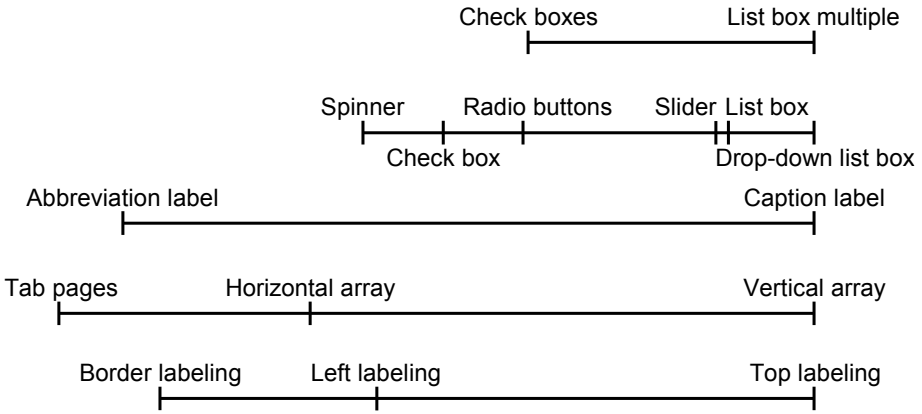
Assumption:
A vertically long dialog box and widgets fit in it

Results of personalization (3/4)

- Subject C



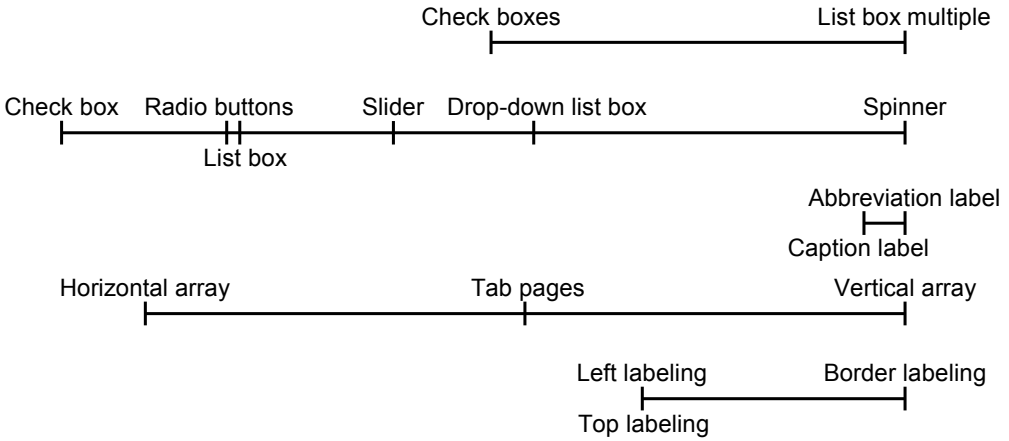


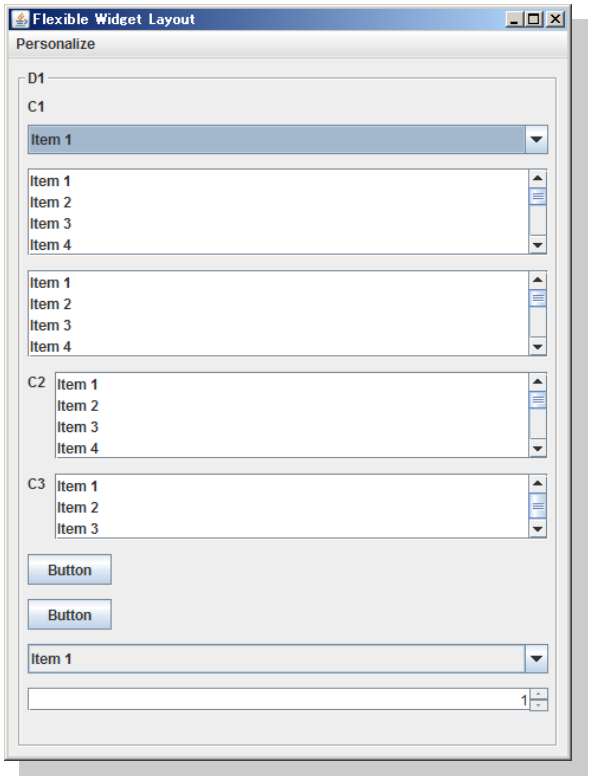


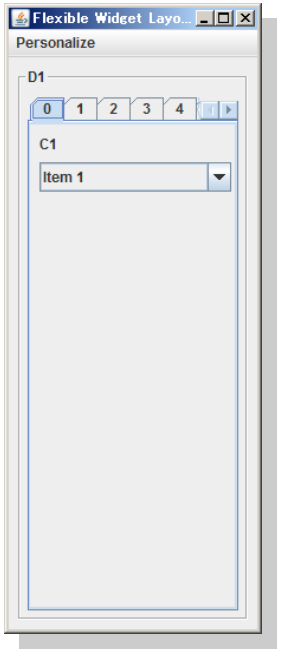
Assumption:
Smallest layout with no wasted space

Results of personalization (4/4)

- Subject D







Assumption:
Smallest layout with no wasted space

Comments

- Were they able to personalize layouts?
  - **Yes, they can do it to an extent.**
  - Sometimes it is too sensitive.
- Was the operation easy?
  - No, it is not understandable and intuitive.
  - Quick responses and PCM itself are intuitive.
- Other comments
  - We should add an adjustment operation.
  - Sub-grouping is not appropriate.

Conclusion

- Method for personalizing the FWL
  - Introducing **pairwise comparison method** and sub-grouping of widgets
  - Indicating the method can generate desirable GUIs for each user
- Future work
  - To include other criterion for evaluating widgets
  - To investigate automatic personalization methods

Thank you

Personalizing Graphical User Interfaces on Flexible Widget Layout

Takuto Yanagida, Hidetoshi Nonaka, and Masahito Kurihara

Hokkaido University, Japan