SUPPLE: Automatically Generating User Interfaces

Krzysztof Gajos
and Daniel S. Weld

University of Washington
Computer Science & Engineering
Motivation

• Current interfaces: complex & “One size fits all”
  ⇒ Adapt to users and tasks

• Variety of display devices & interaction contexts makes hand-designed interfaces expensive
  ⇒ Adapt to device characteristics

⇒ Automatic interface generation is a scalable solution
Approach

• Develop abstract representation for:
  • Interfaces
  • Display devices
  • Users

• Automatically generate interfaces from the abstractions
SUPPLE Architecture

- Interface Model
- Application or Appliance

SUPPLE

- Device Model
- Display

Target Device

- User Model
- User's Info Space
SUPPLE Architecture

- Interface Model
- Application or Appliance
- Device Model
- Target Device
- User Model
- User's Info Space
Examples of Applications

• A Classroom Controller
• An interactive FTP Client
• A distributed jukebox
• A stereo controller
• Print dialog box
Automatically Rendered Interfaces for Classroom Controller
Road Map

- Introduction
- Modeling
  - Interfaces
  - Devices
  - Users
- Inner workings of SUPPLE
- Results and evaluation
- Related Work
- Conclusions and future work
Modeling User Interfaces

- **simple types**: \( \text{int} | \text{float} | \text{string} | \text{bool} \)
- **derivative types**: \( \langle \tau, C_{\tau} \rangle \)
- **vectors**: \( \text{vector}(\tau) \)
- **containers**: \( \{\tau_i \mid i \in 1...n\} \)
- **actions**: \( \tau \rightarrow \text{nil} \)
Modeling User Interfaces

- simple types: \( \text{int}|\text{float}|\text{string}|\text{bool} \)
- derivative types:
- vectors:
- containers:
- actions:
Modeling User Interfaces

- simple types:
- derivative types: $\langle \tau, C_\tau \rangle$
- vectors:
- containers:
- actions:
Modeling User Interfaces

- simple types:
- derivative types:
- vectors:
- containers: \( \{ \tau_i \mid i \in 1...n \} \)
- actions:

{bool,
 <string, {Computer 1, Computer 2, Video} >}
Modeling User Interfaces: Optional Attributes

- Label
- Set of likely values
- Exact value required
- ...

Road Map

☑ Introduction
☐ Modeling
☐ Interfaces
☐ Devices
☐ Users
☐ Inner workings of SUPPLE
☐ Results and evaluation
☐ Related Work
☐ Conclusions and future work
Modeling Device Capabilities

- Device constraints
- Available widgets
  - Primitive widgets
  - Container widgets
- Match cost function for primitive widgets
- Navigation cost function for containers
Examples of Available Widgets

Pointer and Keyboard

Touch Screen
Match Cost Function
For Primitive Widgets

\[
\text{Match}(\langle \text{int}, [0,10]\rangle, \text{Level } 7 \text{, change value from 7 to 8 } ) = 3
\]

\[
\text{Match}(\langle \text{int}, [0,10]\rangle, \text{Level } 7 \text{, change value from 7 to 10 } ) = 5
\]

\[
\text{Match}(\langle \text{int}, [0,10]\rangle, \text{, change value from 7 to 10 } ) = 1
\]
Navigation Cost Function For Container Widgets

• Inputs:
  • A transition type
  • A container widget

• Output: an estimate of user effort to navigate the interface
Example of Navigation Cost

[Diagram of a user interface showing navigation controls for a classroom, including options for light, sound, and video inputs.]
Example of Navigation Cost
Road Map

- Introduction
- Modeling
  - Interfaces
  - Devices
- Users
- Inner workings of SUPPLE
- Results and evaluation
- Related Work
- Conclusions and future work
Modeling Users With Traces

- **Trace** as a model of usage pattern composed of **Trails** [Wexelblat and Maes, CHI 1999]

- **Trail** as a “coherent” sequence of elements the user interacted with

- Trail format independent of rendering

\[
T = \{ <\text{root}, -, -> > \\
<"\text{Left light:Power}", \text{off}, \text{on}> \\
<"\text{Vent}", \text{1}, \text{3}> \\
<"\text{Projector:Input}", \text{video}, \text{computer 1}> \\
.... \}
\]
Road Map

- Introduction
- Modeling
- Interfaces
- Devices
- Users
- Inner workings of SUPPLE
- Results and evaluation
- Related Work
- Conclusions and future work
Rendering As Optimization

- **Cost function** ($\$\$) -- estimated user effort to manipulate a rendering of the interface

- **Inputs:**
  - A rendering
  - A user trace

- **Cost function derived from**
  - Match cost function
  - Navigation cost function
Rendering Algorithm: Properties

- A constrained branch-and-bound search algorithm (like A*) -- at each step assign a widget to an abstract interface element
- Guaranteed to find a rendering with the lowest cost
Design Choices:

- **Constraint propagation methods:**
  - None, Forward Checking (FC), Full

- **Variable ordering:**
  - Bottom-up, Top-down, MRV

- **Admissible Heuristic:**
  - Estimate of the total cost of the entire interface
Road Map

- [✓] Introduction
- [✓] Modeling
  - [✓] Interfaces
  - [✓] Devices
  - [✓] Users
- [✓] Inner workings of SUPPLE
- [☐] Results and evaluation
- [☐] Related Work
- [☐] Conclusions and future work
Results And Evaluation

- Adapting to device characteristics
- Building an interactive interface
- Conceptual study
  - Adapting to usage patterns
Adapting to Device Characteristics
An Interactive Interface
An Interactive Interface
An Interactive Interface
Preliminary Study

- Four “experts”
- Same widget library and conditions as for SUPPLE
Preliminary Study

Human Designer A

SUPPLE
Preliminary Study

Human Designer B
Preliminary Study

Human Designer C
Preliminary Study

Human Designer C

SUPPLE with a trace
Road Map

- Introduction
- Modeling
  - Interfaces
  - Devices
  - Users
- Inner workings of SUPPLE
- Results and evaluation
- Related Work
- Conclusions and future work
Related Work

- Personal Universal Controller (PUC) [Nichols, et al, UIST’02]
- iCrafter [Ponnecanti, et al, UbiComp’01]
- Xweb [Olsen, et al, UIST’00]
- GADGET [Fogarty, et al, UIST’03]
- XIML [Puerta & Eisenstein, IUI’02]
Contributions

- Formal definition of the problem
- User-specific rendering
- An efficient algorithm
- Evaluation of speed and quality
Future Work

• Cross-device consistency (with Anthony Wu)
• Transforming interfaces to adapt to observed usage
• Automatically learning match and navigation cost functions
• Explicit customization
• Incorporate design heuristics
• More complex applications
• The anonymous subjects

• Comments from: Mark Adler, Alan Borning, Gaetano Borriello, Tessa Lau, Jeffrey Nichols, Steven Wolfman, Alexander Yates
More Information

• SUPPLE:
  http://www.cs.washington.edu/ai/supple

• Krzysztof Gajos:
  kgajos@cs.washington.edu

• Daniel S. Weld:
  weld@cs.washington.edu