

Non-Visual Access to GUIs: Leveraging Abstract User Interfaces

Kris Van Hees & Jan Engelen

Katholieke Universiteit Leuven

Department of Electrical Engineering

ESAT - SCD - DocArch



KATHOLIEKE UNIVERSITEIT
LEUVEN



Overview

- Introduction
- Related work
- HCI design issues: non-visual presentation of GUIs
- Leveraging Abstract User Interfaces
- Comparison with other approaches
- The road ahead...

Introduction

- Graphical User Interfaces:
 - Fact of life
- In the past:
 - Mostly single-platform: MS Windows
 - Remote access for everything else
- Recent years:
 - More platforms (Unix-Linux flavours, Mac OS-X, ...)
 - Multiple graphical toolkits in client-server environment (Qt, GTK+, Athena, ...)

Related work

- EU project: GUIB – Screen reader
- Mercator – Screen reader
- HAWK – Non-visual interaction toolkit
- Gnome Accessibility Architecture
- Fruit – Abstract widget toolkit
- **UsiXML** – Abstract user interface definitions

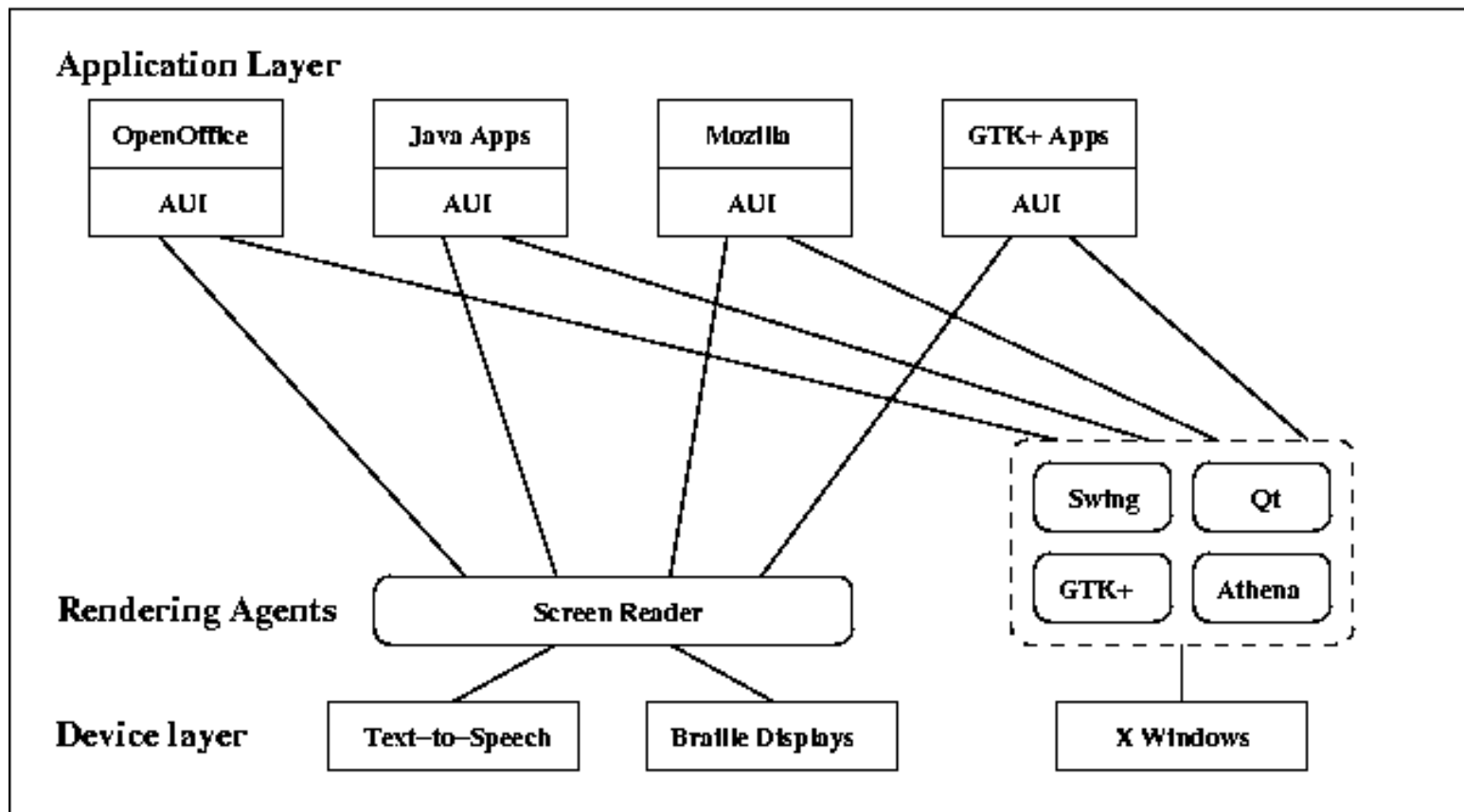
HCI design issues: non-visual presentation of GUIs

- Coherence between visual and non-visual interfaces
 - Consistent mental interaction model
 - Blind / sighted user interaction
- Exploration in a non-visual interface
 - Access beyond the content windows
 - Spatial parameters can carry information
- Conveying graphical information in a non-visual interface
 - Graphical object attributes (appearance, style, ...)

HCI design issues: non-visual presentation for GUIs

- Interaction in a non-visual interface
 - Interaction objects and techniques
 - Alternatives to visual idioms (clicking, dragging, sliding, ...)
- Ease of learning
 - ... because otherwise no one would use it!
- Maintainability
 - Ensure accessibility throughout the UI life cycle

Leveraging Abstract User Interfaces



Leveraging Abstract User Interfaces

- Applications provide an AUI description
 - Hierarchy (tree)
 - Elements (nodes)
 - Properties
 - Data source (if any)
 - Relations
- The UI description is in a standardised format (**UsiXML** or a similar format)

Leveraging Abstract User Interfaces

- The UI description can be transformed for a specific output modality
 - Preservation of semantics and presentation coherence
 - Augmenting UI description
- The UI description is rendered by agents:
 - Graphical toolkit agents for visual presentation
 - Non-visual agents for auditory and/or tactile presentation

Comparison with other approaches

- Development time UI construction
 - Usually automatically generated (e.g. UsiXML, ...)
 - Distinct UIs for specific modalities
- Runtime UI selection
 - Model-driven UI customisation
 - UI adaptation

Comparison with other approaches

- Development time UI construction
 - Model-driven UI creation
 - Essentially different versions of the same application
 - Collaboration suffers from UI differences
 - Collaboration suffers from modality imposed limitations
 - No simultaneous presentation in multiple modalities

Comparison with other approaches

- Runtime UI selection
 - User and context models drive UI adaptation
 - Uses most appropriate interaction objects and techniques
 - Collaboration suffers from model imposed limitations
 - No simultaneous presentation in multiple modalities
 - Requires design time planning for supported adaptations

Comparison with other approaches

- Parallel UI rendering
 - Single UI, multiple renderings
 - Coherent presentation of the UI in multiple modalities
 - Collaboration is implicitly supported
 - Alternative rendering engines do not require UI design changes

The road ahead...

- Prototype implementation of visual and non-visual AUI rendering agents (*in progress*)
 - Speech and Braille output
- Integration with existing AUI frameworks (UsiXML, ...)
- Definition of transformation rule sets to augment UI descriptions

At all stages, feedback from real users will be crucial!

Contact information

Kris Van Hees & Jan Engelen
Katholieke Universiteit Leuven
Department of Electrical Engineering
ESAT – SCD – DocArch
Kasteelpark Arenberg 10
B-3001 Heverlee
Belgium

kris@alchar.org, jan@docarch.be