Automatically Generating User Interfaces for Appliances

The Personal Universal Controller Project

Jeffrey Nichols
jeffreyn@cs.cmu.edu
Human Computer Interaction Institute
Carnegie Mellon University

Abstract

Today's complex appliances are plagued by difficult-to-use and inconsistent user interfaces.

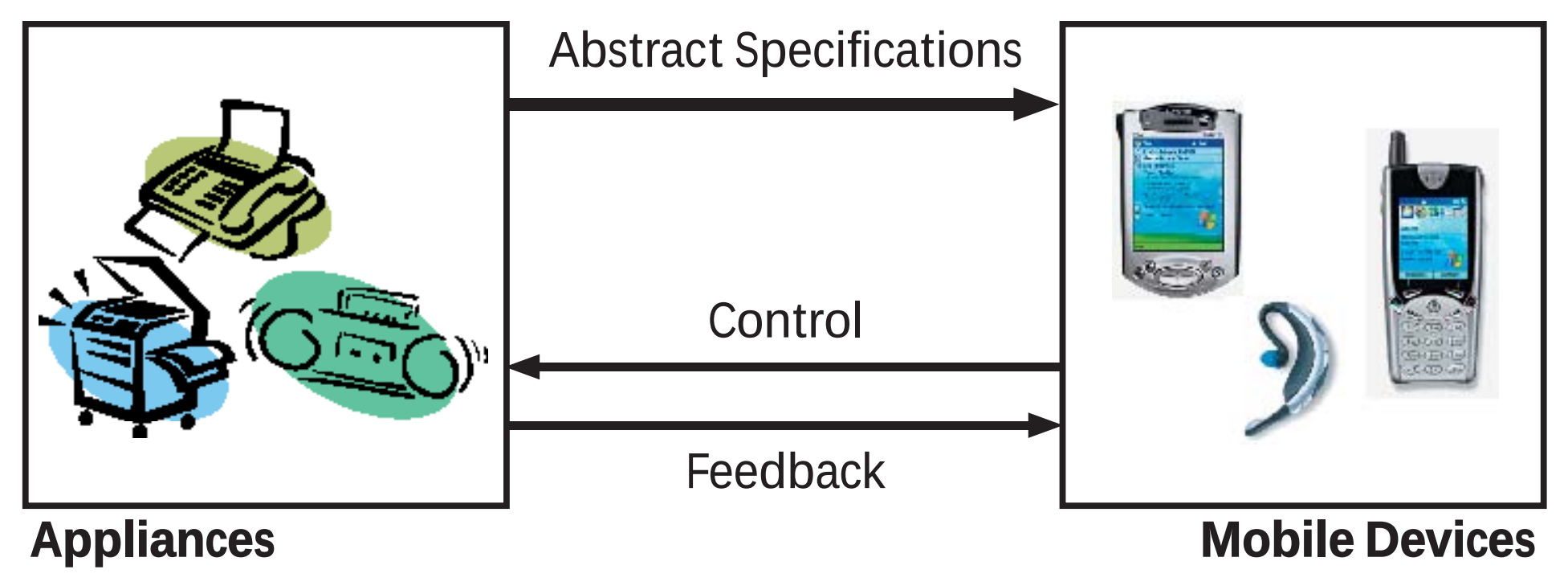
Some of these problems can be addressed by moving appliance user interfaces from the appliance to a mobile device, such as a PDA or mobile phone.

I have conducted studies showing that users are twice as fast and make half as many errors when controlling appliances through a handheld as compared to using the manufacturers' interfaces.

I have built a system for automatically generating user interfaces that allow users to control their appliances. My thesis addresses some of the issues that arise when automatically building appliance user interfaces for end-users.

Personal Universal Controller System

Use mobile devices to control appliances. The following diagram shows how the Personal Universal Controller (PUC) system separates the interface from the appliance.



There are four pieces to the PUC system:

Abstract Specification Language

The language is designed to allow for complete specification of any appliance. I have put significant effort into ensuring the language is as concise and easy-to-use as possible.

The language is XML-based with full documentation at: http://www.cs.cmu..edu/~pebbles/puc/specification.html

The example to the right illustrates almost all features of the language.

```
<?xml version="1.0" encoding="utf-8"?>
<spec name="MediaPlayer" version="PUC/2.2">
  <labels>
    <label>Media Player</label>
  </labels>

  <groupings>
    <group name="Controls" is-a="media-controls">
      <labels>
        <label>Play Controls</label>
        <label>Play Mode</label>
        <text-to-speech text="Play Mode"
                        recording="playmode.au"/>
      </labels>

      <state name="Mode">
        <type>
          <enumerated>
            <item-count>3</item-count>
          </enumerated>
          <value-labels>
            <map index="1">
              <labels><label>Stop</label></labels>
            </map>
            <map index="2">
              <labels><label>Play</label></labels>
            </map>
            <map index="3">
              <labels><label>Pause</label></labels>
            </map>
          </value-labels>
        </type>

        <labels><label>Mode</label></labels>
      </state>

      <group name="TrackControls">
        <command name="PrevTrack">
          <labels><label>Prev</label></labels>

          <active-if>
            <greaterthan state="PList.Selection">
              <constant value="0"/>
            </greaterthan>
          </active-if>
        </command>

        <command name="NextTrack">
          <labels><label>Next</label></labels>

          <active-if>
            <lessthan state="PList.Selection">
              <ref-value state="PList.Length"/>
            </lessthan>
          </active-if>
        </command>
      </group>
    </group>

    <list-group name="PList">
      <state name="Title">
        <type><string/></type>
        <labels><label>Title</label></labels>
      </state>

      <state name="Duration" is-a="time-duration">
        <type><integer/></type>
        <labels><label>Duration</label></labels>
      </state>
    </list-group>
  </groupings>
</spec>
```

Automatic Interface Generators

I have built automatic interface generators for multiple platforms (PocketPC, Smartphone, and desktop) and modalities (graphical and speech).

Two-Way Communication Protocol

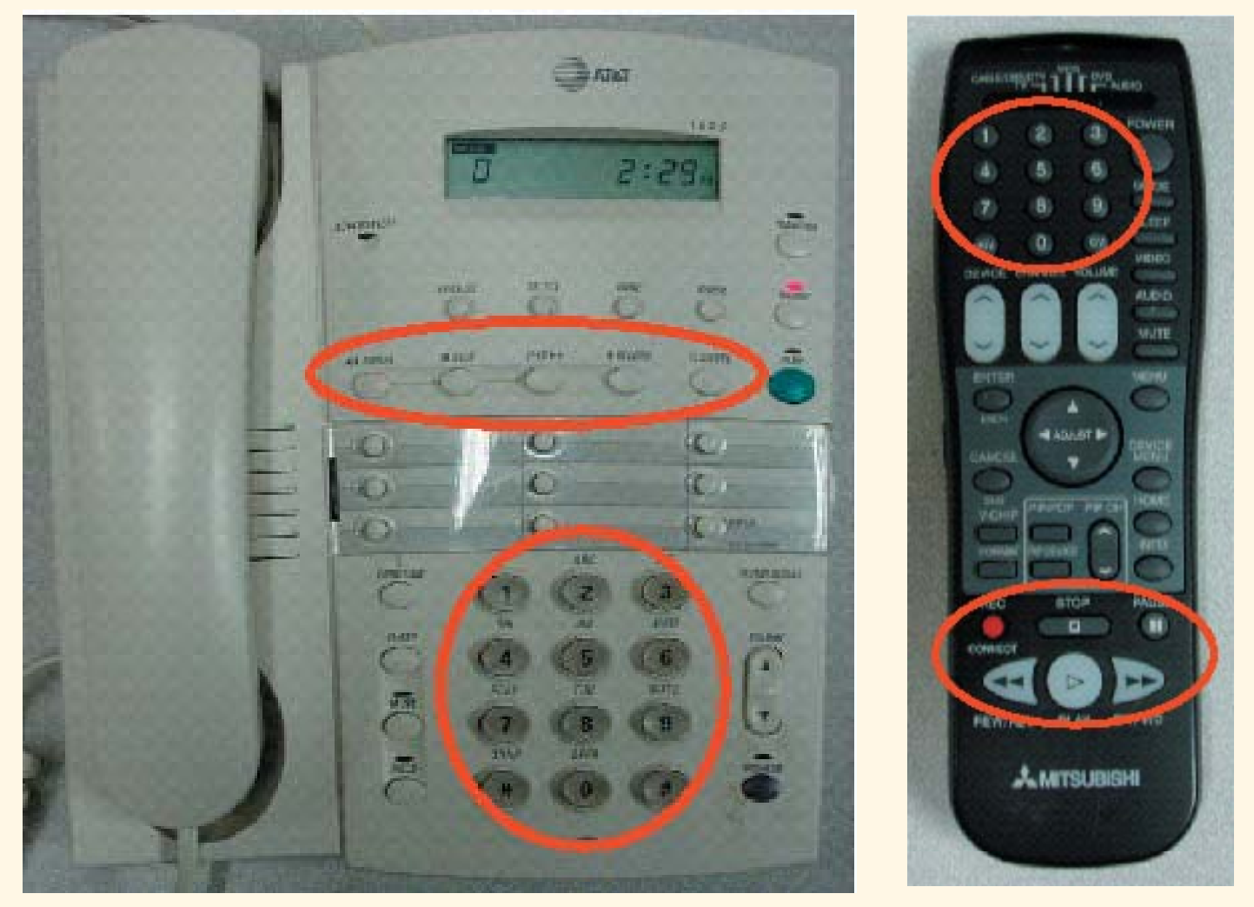
My communication protocol allows for better appliance UIs by enabling the controller to show the appliance state. For example, this allows the UI to gray out commands that are disabled.

Appliance Adaptors

The PUC system is able to control real appliance through adaptors, which translate proprietary protocols into the PUC protocol. Adaptors have been built for specific appliances, like Windows Media Player, and general protocols, like UPnP and AV/C.

Domain-Specific Design Conventions

A common problem for automatic generators has been that their designs do not conform to domain-specific design conventions that users are accustomed to.



Examples of domain-specific conventions in appliance interfaces

In many systems, a designer will add conventional designs to the user interfaces after generation. This is not a viable approach in my system.

Smart Templates

Allow hand-designs for conventions to be integrated into an auto-designed user interfaces.

Standardize in advance on a set of specification restrictions for language groups that should be rendered conventionally.

Because templates are defined in terms of the primitive language, interface generators are not required to implement each template.

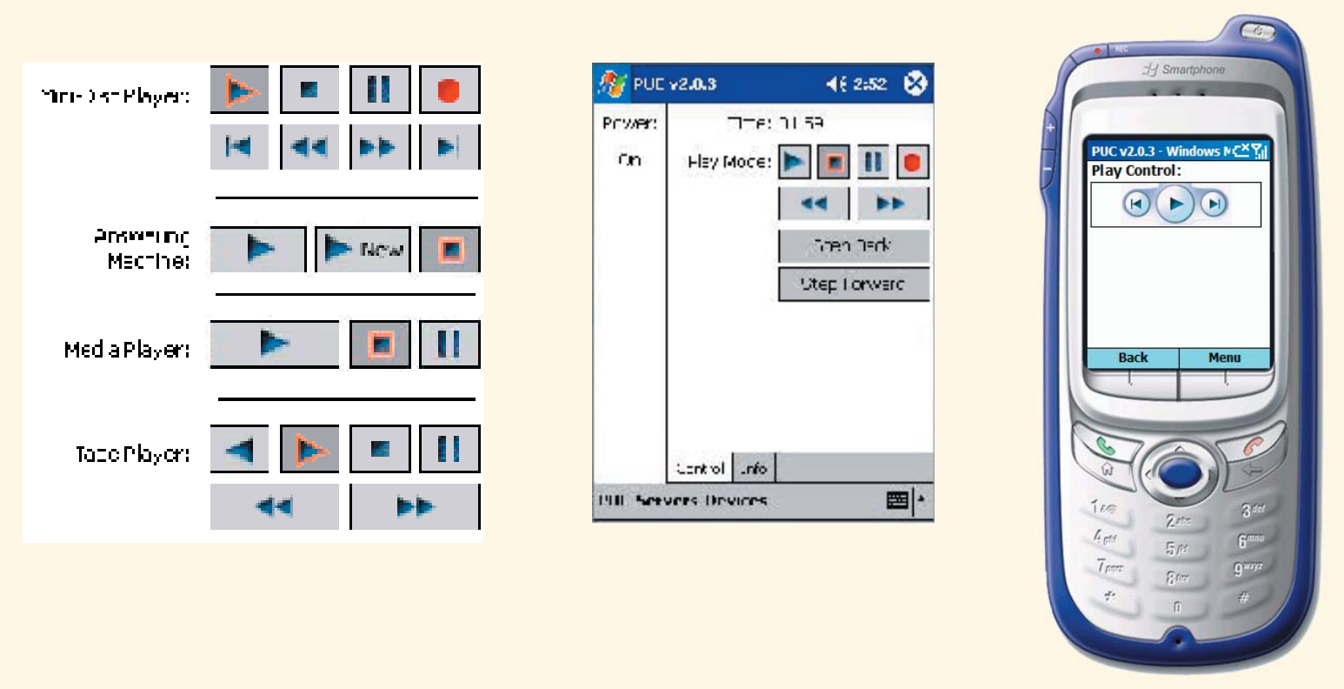
I have defined templates for:

- date
- datetime
- dimmer
- image
- image-list
- media-controls
- mute-mic
- mute-speaker
- power
- phone-dialpad
- time-absolute
- time-duration
- volume
- list-add
- list-delete

Smart Templates are designed to be flexible, allowing for different underlying representations. This parameterization allows templates to cover the common and unique functions of each appliance.

media-controls Example

Used for controls of a media stream, such as sound or video.



Example renderings of the media-controls Smart Template

The specification example to the left shows an example of using this Smart Template in an appliance specification.

Interface Consistency

The PUC system has a unique opportunity to ensure internal and external consistency among all interfaces that a user generates, because each PUC user has their own personal device.

Interfaces can be made internally consistent using the standard toolkit on each device, like our PocketPC interface generator does.

Interfaces can be also be made externally consistent. E.g., a newly generated interface for a VCR in the conference center should look and feel like the interface for my VCR at home.

Creating externally consistent interfaces can be broken into two problems:

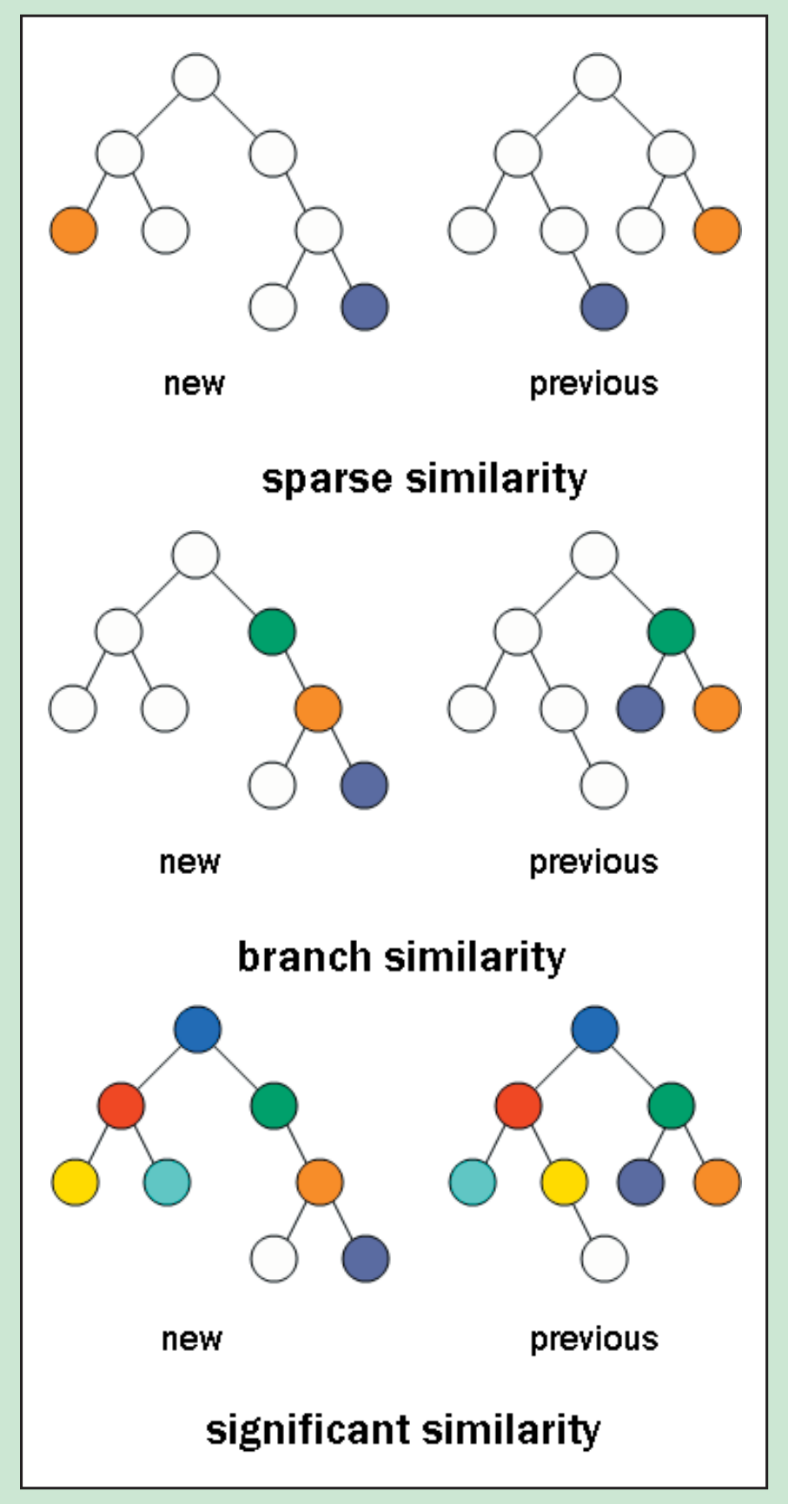
Consistency Problem

How do we make user interfaces consistent for similar appliances?

If the appliances only share a few similar functions, then try to reuse widgets. For example, always use a slider for volume even if a combo box would normally be used.

If the appliances share a small set of grouped functions, then try to use the previous interface for that group. If necessary, use the panel structure the group was previously in.

If the appliances are mostly similar, try to put the functions of the new appliance into the organization of the old interface.



Similarity Problem

How can we determine that two appliances are similar from their specifications?

It is difficult to conclusively know whether two functions from different appliances are the same because there is very little semantic information in the spec. language

Can estimate similarity based on properties of state variables

- Smart Template
- Name
- Group Name
- Labels
- Type

May be able to improve the estimate by looking at relationships between multiple similar variables

Multi-Appliance User Interfaces

Appliances are increasingly being connected together to form systems, such as a home theater or a presentation room. The user interfaces for these systems are not connected however, requiring users to separately control each appliance to get a desired behavior from the system.

MY NEW HOME THEATER IS AMAZING.
IT'S GOT A DVD, HD, DVR, FM, SATELLITE DISH, MP3, WIDESCREEN TV, SEVEN SPEAKERS AND A UNIVERSAL REMOTE.
IT'S FUN TO INVITE PEOPLE OVER SO THEY CAN SHOW ME HOW TO TURN IT ON.
©UFS, Inc.

Improving Interfaces with the PUC

Generate an interface that aggregates all functions into one set of screens organized by screen instead of appliance

Automatically create macros for frequently used functions

E.g., "Play DVD" would:

1. turn on television, DVD player, stereo
2. turn off VCR
3. set the TV and stereo sources to the DVD player
4. instruct the user to insert a DVD (if necessary)
5. play the DVD

Extending Appliance Descriptions

Track inputs and outputs of appliances, and tag each state variable and command with information on how inputs and outputs are modified

Include partial task information that can be taken from each appliance and constructed into a full task specification for the entire system.

This information can be combined to build interfaces for systems of appliances.

Acknowledgements

I would like to thank Brad Myers for advising me on this work. This work was funded in part by grants from NSF, Microsoft, General Motors, and the Pittsburgh Digital Greenhouse, and equipment grants from Mitsubishi Electric Research Laboratories, VividLogic, Lutron, and Lantronix. The National Science Foundation has funded this work through a Graduate Research Fellowship and under Grant No. IIS-0117658.