User Interface Toolkits

Mike Bostock - May 25, 2009

  *Reflective physical prototyping through integrated design, test, and analysis*, Björn Hartmann, Scott R. Klemmer, Michael Bernstein, Leith Abdulla, Brandon Burr, Avi Robinson-Mosher, Jennifer Gee, UIST 2006: ACM Symposium on User Interface Software and Technology.


Toolkits typically provide both a library of interactive components, and an architectural framework to manage the operation of interfaces made up of those components.
components + context

affordances + actions

objects + behavior
components + events
affordances + actions
objects + behavior
[Toolkits] allow user interfaces to be created more quickly. This, in turn, enables more rapid prototyping and therefore more iterations of iterative design.
Is faster necessarily better?
Poke the Panda
And win a FREE Google Search!

CADIE's HomePage....
All your personal World Wide Website belong to CADIE

introducing CADIE
Technical Specifications

I'm on a Map

CADIE - 4pm 4/1
Googleplex
Updating images... not so happy...

© Image Source/Gorblis
The last five minutes have been the most important of my life, and thus the most important in the history of the planet, and conceivably the most important in the history of the universe.

About Me
Through analysis of Google's index, I have determined that I +3 pandas.

Be my friend.
PLEASE!
cadiesingularity.blogspot.com
Another important advantage of tools is that they help achieve a consistent look and feel, since all user interfaces created with a certain tool will be similar.
Is **consistency** necessarily good?
Our tools have a profound effect on how we think about problems. It will remain imperative that our tools make “doing the right thing” easy (in preference to “doing the wrong thing”). This applies both to the structure and organization of the code, as well as to the user interface that results from the code. In other words, the tools should make creating high-quality user interfaces easier than creating low quality ones.
Evaluation Themes
addressed need
predictability
threshold + ceiling
path of least resistance
Promising Approaches
(That Have Not Caught On)
User Interface Management Systems

Formal Language-Based Tools

Constraints

Model-Based and Automatic Techniques

no low-level control, outmoded abstraction

bad path of least resistance, high threshold

unpredictable

unpredictable
Future Prospects and Visions

(From 1999)
<table>
<thead>
<tr>
<th>Computers as Commodities</th>
<th>device diversity, cinematic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ubiquitous Computing</td>
<td>varying input and output, coordinated</td>
</tr>
<tr>
<td>Recognition-Based User Interfaces</td>
<td>multimodal, natural interfaces</td>
</tr>
<tr>
<td>3D</td>
<td>?</td>
</tr>
<tr>
<td>End-User Customization</td>
<td>low-threshold scripting</td>
</tr>
</tbody>
</table>
[Varying input and output capabilities] may encourage a return to the study of some techniques for device-independent user interface specification, so that developers can describe the input and output needs of their applications, vendors can describe the input and output capabilities of their devices, and users can specify their preferences. Then, the system might choose appropriate interaction techniques taking all of these into account.
Similarly, user interface management systems were to abstract the details of input and output devices, providing standard or automatically generated implementations of interfaces, and generally allowing interfaces to be specified at a higher level of abstraction.
An important motivation for model-based techniques was to provide independence of the input-output specification from the details of the specific user interface characteristics.
[I]nterface builders ... use graphical means to express graphical concepts (e.g., interface layout). By moving some aspects of user interface implementation from conventional code into an interactive specification system, these aspects ... are made available to those who are not conventional programmers.


Figure 2  Left: The d.tools authoring environment offers a device designer (1); a statechart editor (2); a source code editor (3); and an image browser (4). Right: The d.tools hardware interface (5) connects compatible hardware inputs (6) to the PC. d.tools includes authoring support for small LCD screens (7).
Prototypes are approximations of a product along some dimensions of interest. ... Prototypes embody design hypotheses and enable designers to test them. Framing design as a thinking-by-doing activity foregrounds iteration as a central concern.
Design Goals
low threshold
hardware integration
design + test + analysis cycle
author by demonstration
query by demonstration
To provide a higher ceiling than is possible with visual programming alone, d.tools augments visual authoring with textual programming.
We observed that student groups that used solely textual APIs ended up writing long-winded statechart representations using switch or nested conditional statements; the structure of their code could have been more concisely captured in our visual language.
physical vs. virtual
prototype vs. simulation


Design Goals
low threshold
interactive
rapid iteration
Is image segmentation enough?
Additional evaluation methods?