Ability-Based Design: Concept, Principles, and Examples

Ability is context-dependent for everyone

Environment
- Temperature
- Light
- Noise
- Weather
- Terrain

Social
- Distraction
- Collaboration
- Confinement
- Clothing

User
- Preferences
- Goals and Tasks
- Physical state
- Orientation
- Emotional state

Technology / Device
- Screen resolution
- Connectivity
- Speed
- Browser
- Battery
Walking User Interfaces

Users are “situationally impaired” while walking (Yambe and Takahashi 2007) - Re: “getting off the treadmill”

Figure 2. Our music player user interface in two sizes. (left) The player while standing; (right) the player while walking.
Barrier Pointing
[Froehlich et al. 2007]

How to provide stability for a user to press a stylus against a screen, particularly during motion?
“Design for One”

**Adaptivity:** How systems and interfaces are made **user-customizable** to incorporate user’s preferences.

**Adaptability:** How systems and interfaces are made to **adapt to our abilities**. For example through sensing and performance modeling.
Discussion:

In what ways does designing with an ability-based approach help all users, not just those with impairments?
TUSHAR
"Situation aware" systems. [...] might be helpful for the unimpaired in special cases where "Ability" changes drastically under stress or duress.

MICHELLE
Designing with an ability-based approach helps all users because it changes the mindset of designers.
FRANCESCO
I think that in the majority of the cases the best design always includes a “perfect match” between hardware and software. In the past [...] there was a battle between hardware engineers that want to fix all the problems by improving the hardware and software engineers that want to solve them by pushing on the software.

CALVIN
A problem I noticed with this "design-for-one" methodology is that it [...] is not always feasible from a technology standpoint. While computation is only getting cheaper by the day, the speed of the process of capturing, cleaning, analysing and spitting out insights is of concern.
Win-win: Muting and Captioning Advertising Videos

80% of people react negatively when mobile video ads play sound in the feed.

Facebook will automatically caption video ads, so videos can be played but with the sound off.

Captioning ads makes them more accessible for everyone, while increasing view time by 12% in a Facebook study (!)
Dustin: “...an interface designed with blind users in mind can also be useful for users with able vision who, at the end of a long day of staring at a screen, may benefit from an interface which does not require sight...”
Ability-based systems may both adapt to user’s abilities, or be adapted or customized directly by the user (Edwards 1995).

Figure 2. (a) A user whose abilities match those presumed by the system. (b) A user whose abilities do not match those presumed by the system. Because the system is inflexible, the user must be adapted to it. (c) An ability-based system is designed to accommodate the user’s abilities. It may adapt or be adapted to them. Our symbols are based on those from prior work (Edwards 1995).
“No more than 60% of people who indicate a need for access technologies actually use them…”

(Fichten et al. 2000)

Why? . . .

Reasons for abandonment include:

- Procurement and maintenance costs
- Configuration time
- Compatibility (or lack of)
- Stigmatizing or humiliating aspects
- Painstaking effort of use (learning cost)
<table>
<thead>
<tr>
<th>Stand</th>
<th>Principle</th>
<th>Description</th>
<th>Recommendation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stance</td>
<td>Ability.</td>
<td>Designers will focus on ability not dis-ability, striving to leverage all that users can do.</td>
<td>Required</td>
</tr>
<tr>
<td></td>
<td>Accountability.</td>
<td>Designers will respond to poor performance by changing systems, not users, leaving users as they are.</td>
<td>Required</td>
</tr>
<tr>
<td>Interface</td>
<td>Adaptation.</td>
<td>Interfaces may be self-adaptive or user-adaptable to provide the best possible match to users’ abilities.</td>
<td>Recommended</td>
</tr>
<tr>
<td></td>
<td>Transparency.</td>
<td>Interfaces may give users awareness of adaptations and the means to inspect, override, discard, revert, store, retrieve, preview, and test those adaptations.</td>
<td>Recommended</td>
</tr>
<tr>
<td>System</td>
<td>Performance.</td>
<td>Systems may regard users’ performance, and may monitor, measure, model, or predict that performance.</td>
<td>Recommended</td>
</tr>
<tr>
<td></td>
<td>Context.</td>
<td>Systems may proactively sense context and anticipate its effects on users’ abilities.</td>
<td>Recommended</td>
</tr>
<tr>
<td></td>
<td>Commodity.</td>
<td>Systems may comprise low-cost, inexpensive, readily available commodity hardware and software.</td>
<td>Encouraged</td>
</tr>
</tbody>
</table>
Prior Approaches

Assistive Technology
Rehabilitation Engineering (non HCI specific)
Universal Design
Universal Usability
Design for All
User Interfaces for All
Inclusive Design
Extra-Ordinary Human-Computer Interaction
Assistive Technology

Park MacArthur.  
*Ramps* (2013)
How the SUPPLE system works
[Gajos and Weld, IUI ‘04]

The optimization algorithm **automatically adapts user interfaces** using a predictive model based on a user’s performance, and dexterity in tasks such as pointing, dragging, list selections, and multiple clicks.
User behavior is measured and then used to automatically customize interfaces which render to the user's usage pattern to make accurate typing easier.

SUPPLE

[Gajos and Weld, IUI '04]
Discussion:

If we are able to generate predictive models based on a user’s performance, is it better to make local or global adaptations to the interface? **What are the trade-offs?**

To what extent is it important **to ask the user to approve a customization** (perhaps in the form of a preview), that has been generated based on performance assessments?
**Takeaways**
- Interface generation treated as an optimization problem
- Performance evaluation is a form of customization
- Usefulness of heuristics to find a solution

**Outstanding questions**
- How do we model abilities and impairments?
- How often are abilities measured and how?
- How do we map certain abilities to interface changes?
- How can the automatic generation of interfaces based on performance evaluation also be customized by users?
Example-Centric Programming: Integrating Web Search into the Development Environment

Joel Brandt, Mira Dontcheva, Marcos Weskamp, Scott R. Klemmer
Scenario

Suppose you need to retrieve power-usage data from a Web service, using Adobe Flex builder.

Method contains “URL”? => urlLoader => Can’t remember the usage

Problems with this approach?
Scenario

Web browser is independent of other tools in the development chain.

Code editor assumes all code is typed by hand.

Search engine has no notion of a user’s development context.
Blueprint

● A Web search interface for accessing online example code from within the development environment and thus enable programmers to write better code more easily.

● Provides a UI for search queries and results.
● Automatically augments queries with code context.
● Focused on getting code examples.

Adobe Flex Builder
Scenario

Suppose you need to retrieve power-usage data from a Web service, using Adobe Flex builder.

Method contains “URL” ? => urlLoader => Can’t remember the usage

Benefits of this approach?
Implementation

Blueprint Server Query Process

1. User query: "chart"
2. Query interface: rendered examples
3. Augmented query: "chart + flex3"
4. Google
5. Result: (list of examples, docs, suggestions)
6. Result: (list of examples)
7. Result: (list of urls, suggestions)
8. Result interface: rendered examples

Blueprint Client
Blueprint Plugin
Eclipse
Example Cache Database
Evaluation

- Lab study:
  - 20 participants
  - Control: Firefox + Adobe Community Help Search Engine
  - Treatment: Blueprint
  - Measures: Time, Rank
    - Tutorial task
    - Directed programming task
    - Exploratory programming task

- Hypotheses:
  1. Programmers will complete direct tasks more quickly because they will find example code faster
  2. Code produced will have the same or higher quality as code written with traditional means
  3. Programmers will produce better designs on an exploratory design task

small-scale
Evaluation

- Longitudinal study:
  - 2,024 Blueprint users + 13,283 Community Help users
  - Control: Users who used the Community Help search engine over same duration
  - Treatment: Blueprint users
  - Measures: Click-throughs, syntax, queries etc.

- Hypotheses:
  1. If additional context is not necessary, Blueprint queries should have a significantly lower click-through rate
  2. If users are using Blueprint with other IDE features, queries should contain more correctly formatted code
  3. If Blueprint is used for reminders, users should repeat queries more frequently across sessions

large-scale
Lab Study - Results

1. **Programmers will complete direct tasks more quickly because they will find example code faster**
   - Time to first copy/paste of example (57s vs. 121s)
   - Time to completion (346s vs. 479s)
   - Paste time strongly correlated with task completion time
2. Code produced will have the same or higher quality as code written with traditional means

- Professional software engineer external to the project rank-ordered participants code
- Treatment produced significantly higher-rated code
Lab Study - Results

3. **Programmers will produce better designs on an exploratory design task**

- Professional software engineer external to the project rank-ordered participants charts
- Treatment produced higher-rated designs but the result was not statistically significant
Longitudinal Study - Results

1. If additional context is not necessary, Blueprint queries should have a significantly lower click-through rate

- Number of click-throughs to source pages on search (~43,000 queries measured)
- Treatment is much lower (0.38 vs. 1.32)
Longitudinal Study - Results

2. If users are using Blueprint with other IDE features, queries should contain more correctly formatted code

- Check for camelCase in queries
- Treatment contains much more camelCase (49.6% vs. 16.2% of queries)
Longitudinal Study - Results

3. If Blueprint is used for reminders, users should repeat queries more frequently across sessions

- Repeat queries during sessions
- Treatment has much more queries issued again by the same user (12.2% vs. 7.8% of queries)
Small scale vs. Large scale

Other example:

- Joel Brandt et al. “Two studies of opportunistic programming: interleaving web foraging, learning, and writing code”
- **Lab**: 20 programmers to rapidly prototype a Web application in the lab.
- **Longitudinal**: quantitatively analyzed a month-long sample of Web query data. 24,293 programmers produced the 101,289 queries in the sample.
Discussion - Groups of 2 (1 min)

- What are the pros and cons of small-scale and large-scale studies?
- How would you apply them into your own projects?
How would you relate Blueprint to Ability-Based Design?
Thank you!