CS 465 Homework 2
Alex Lambert (alamber3)

1. (a) My air conditioner

i. The air conditioner is used in my apartment. It’s used by both novice and experienced users. Nobody I know has any formal air conditioning training, nor have we read the air conditioner manual. Users are easily frustrated: when they turn on the air, they’re too warm (and probably irritable.) The air conditioner panel is in a public area. As a result, the air conditioner needs to be aesthetically pleasant but not draw attention. (Sophisticated people have art on their walls; normal people have posters; I have a giant faux-wood-paneled air conditioner.) It should be easy to learn and use. Some learning by trial and error is acceptable, since users tend to operate it daily. However, it still must be accessible to novices.

ii. A. Instant acknowledgment. The control dial clicks loudly when you choose a new option, so you know that the machine is listening. (Compare this to thermostats with soft, mushy buttons – you never really know if you pressed them hard enough.)

B. Simple controls. It just has two knobs and a switch. I’ll confess that I never figured out how to use the thermostat in my house over the summer; it had too many options.

C. Easily manipulatable controls. The master dial is big and sculpted for pinching.

iii. A. Wrong offset on master dial. It turns off when I choose “medium air” and turns on the fan when I choose “off.” The pointer on the dial is out of sync with the instruction text. I’m not sure when I have the air on. A redesign should replace the dial.

B. Hard-to-find control panel. The face of the air conditioner opens to reveal the panel, but I couldn’t figure out how to open it initially. A redesign should add an unobtrusive invitation to open the front of the unit (e.g. with finger slots.)

C. Instructional text too close to dial. The dial has worn off some of the lettering on the options. A redesign should ensure the dial does not scrape against the base.

(b) Self-checkout kiosks

i. The self-checkout kiosks are operated by inexperienced users who are in a hurry. Their goal is to get out of the store, not to use the checkout system. The store wants to prevent inventory shrinkage and reduce their labor costs by automating checkout. As a result, the kiosks are designed to be easy, fast, and secure.

ii. A. Easy to learn. The workflow is simple: scan, bag, repeat, pay.

B. Accommodates experienced users. The audio instructions aren’t necessary for experienced users, and the system doesn’t force one to listen to them. You can, for example, tap the “credit” icon as the system is explaining how to pay.
C. **Proactive assistance.** If the system sees that you’re having trouble scanning an item, it will automatically summon a clerk. This is much less embarrassing than asking for help.

iii. A. **Slow for some cases.** When you buy large quantities of a single item, you have to scan each package and bag it separately. It should instead have a quantity option (though it may be hard to do this securely.)

B. **Leaning on the platform yields a scolding.** The bags are continuously weighed; if you lean on the platform, it tells you to remove the new item from the bags. It should be more difficult to accidentally disturb the scale.

C. **Difficult to scan barcodes.** It’s not always clear where the scanner is “watching.”

If the system projected a red block on the target area, users could learn how to position their items.

(c) The alarm feature on my cell phone

i. The phone has few resources, and the time-to-market for this industry is short, forcing rapid software development. It is used by nontechnical people who don’t want to learn how to use another alarm clock interface.

As a result, the system must be simple and easy to use.

ii. A. **Fast access.** I can get to the alarm system in two button taps. This is great, because it’s the second most-used feature on my phone.

B. **Some preset options.** It has presets for 10-, 20-, 30-, and 60-minute alarms. This makes a common case (“wake me in an hour”) easy.

C. **All settings visible on one page.** I can easily see if I’ve set the time properly before changing other options.

iii. A. **Designed to handle multiple alarms simultaneously.** This is not a common case, at least for me. As a result, the first screen is confusing; it’s a gray window that says “No alarms set”. You have to explicitly choose “New” to create an alarm. A redesign should automatically create a new alarm if none are set (or eliminate the multiple-alarm feature entirely.)

B. **Preset options incomplete.** It has presets for 10-, 20-, 30-, and 60-minute alarms. It doesn’t have presets for 5 or 15 minutes; I find those much more common. I have to do the addition manually, and I’m not good with numbers. A redesign might replace the presets with a general way to specify the number of minutes until the alarm triggers (a relative time) instead of the absolute time.

C. **Default alarm tone is awful.** The default alarm tone is the Cingular logo-tone. I never liked it in the first place – it sounds pretentious to me. The clever marketers who chose this default have guaranteed that I’ll recognize the logo-tone anywhere. But now when I hear it I’m angered: I associate the tone with having to wake up in the morning.

2. (a) Jon Caruana

i. Jon’s Windows logon prompt sometimes disappears. He can only bring it back by mashing keys for a while. He thinks this is caused by his graphics driver. An alternative design could automatically update drivers for hardware; this is already implemented in Vista.

ii. His instant message client automatically resizes itself. But it continues to resize itself even when minimized – so when it’s restored, it takes on a uselessly small size.
This is caused by a bug in the resizer. This problem could have been prevented by having developers constantly test the new features in their software. It’s difficult to test every possible set of preferences, but a developer could reconfigure his personal workstation frequently to test more configurations. Jon hasn’t given feedback to the developers; if they had designed a low-friction way to receive bug reports from users, he could have mentioned the problem and offered to test a fix.

iii. iTunes and QuickTime take over the file association for portable network graphics (PNG) files. QuickTime fails to properly display these graphics when they’re first loaded. QuickTime gives no way to revert to the standard handler. He had to reinstall Windows to fix the association; now he avoids apple software. A better design would make it easy to revert QuickTime’s aggressive changes to the system configuration. Something like this has already been implemented for autoplay handlers in XP SP2 and Vista. ([http://technet.microsoft.com/en-us/magazine/cc162493.aspx](http://technet.microsoft.com/en-us/magazine/cc162493.aspx))

(b) Ross Nelson

i. FileZilla, his FTP client, has an automatic update feature. When the developers release a new version, he’s asked to update the software. But the updater user experience is awful: it expects him to choose a place to save the update, extract it manually, and then replace the right files in the Applications folder to fully update the client. The developers assumed both that users were technically savvy (able to replace the right files manually) and willing to devote more than a few minutes to installing each update. While this is true for developers, neither of these are true for most users; the developers designed for themselves rather than for their customers. Other Mac applications (like Adium and Pixelmator) seem to use a common, familiar, and user-friendly update UI; FileZilla should adopt that existing work.

ii. Ross uses Shift+Command+S to access “Save As” in most of his programs; this is a Mac convention. A few applications, like Word 2008, don’t map this keystroke to “Save As.” Word instead shows or hides the formatting toolbar. I think this happens because the Mac version of Office uses both Windows and Macintosh conventions, and the developers aren’t necessarily aware of some of the subtle differences between the platforms. (I didn’t know about this convention until he mentioned it to me; I don’t think there’s an equivalent keystroke in Windows.) To fix this, Apple should explicitly note such implicit conventions in its human-interface guidelines, and Microsoft developers should carefully consider their users’ habits, because even minor issues like these can be frustrating. (Ross says he hits this problem at least daily.)

iii. Ross wanted to use a tool called Graphic Converter to convert all of the images in a particular directory. He knew that it could handle directories, and he could convert individual files, but he couldn’t figure out how to convert a directory. He said, “The problem is I knew that it worked with directories, so I kept my directory selected. At least in the default view, that doesn’t work. You have to select [all of the] files [– not just the directory] . . . Everything I tried failed to make the Go button become enabled.” This happened because the same system-standard dialog box is used to select both individual files and directory trees. It’s not clear that the box can (or should) be used for both. To fix this, the interface for selecting a directory should make it clear that the user’s objective is to select a directory, not a file. Windows,
for example, provides a directory selector dialog that shows the folder tree view but does not show individual files.

(c) Jordan Timmermann

i. Jordan said: “It seems like each laptop keyboard I’ve used (even within the same brand) place command keys in different locations, so it’s really hard to adjust to a different keyboard.” This happens not because there’s been any recent innovation in keyboard layouts, but because the nuances of keyboard layout are relatively unimportant compared to the rest of the design (e.g. the sleek sculpting of the MacBook.) To remedy this, each company should decide on a reasonable keyboard design (like having the arrow keys in a cluster in the lower-right corner) and require future designs to use it. I don’t necessarily see this as a huge problem, though, since most people use at most one laptop frequently.

ii. Jordan complained that word processors don’t make it easy to create attractive documents (“as \LaTeX can,” I inferred.) He says he ends up using \LaTeX because it’s painful to do anything besides font manipulation in Word. In particular, he wants to “set up nice templates, links within documents, insert graphics nicely, [configure] indentation, [and] get rid of annoying grammar checks.” I’ve seen this issue too. I think that the first GUI word processors gave the user too much control over formatting; they assumed he would be coming from the publishing world and would know how to use that control to create attractive documents. That’s not the case anymore; I’ve seen some amazingly awful Word documents. But it’s too late to change that, because users would complain if it became harder to play around with the typeface (even if they really shouldn’t.) Word 2007 has partially addressed this by emphasizing “styles” (like “Heading 1”) and creating attractive “themes” by displaying previews for them in the ribbon.

iii. Jordan’s Motorola phone won’t let you associate multiple numbers with a single contact. If you want to enter someone’s home, mobile, and work number, you’ll need to set up three different entries. The designer was probably aiming for simplicity, because entering contacts into a phone is (from what I’ve seen) one of the most difficult tasks for inexperienced users. (Unlike text messaging or other enhanced services, everyone has to use the contact list.) I don’t necessarily disagree with this design decision; I don’t think it should be fixed. Even though my own phone supports multiple numbers for a single contact, I’m not always sure which number it will use when I select the contact. If I want to be sure, I have to visit a submenu and explicitly select the right number. I’d rather see this information in the list.

3. We’re looking for an equation of the form $T_n = T_1 \cdot n^{-\alpha}$. I used Matlab’s curve-fitting toolbox with a power-law model ($f(x) = ax^b$) and fixed $a = 12.56$ to match the initial trial’s result. The learning constant is $\alpha = 0.362$. As for strategies: the subject said “well, I used my left hand because that’s the hand most of those keys are naturally typed with.” He speculated that his typing time “started high, [went] down relatively quickly as I figured out how to type the sequence, fluctuated slightly for a bit, and then gradually went back up as my hands got sore.”

General model Power1:
$$f(x) = a \cdot x^b$$

Coefficients (with 95% confidence bounds):
\begin{verbatim}
#!/usr/bin/env perl

use strict;
use warnings;
use Time::HiRes qw(gettimeofday tv_interval);

my $target = "zyxwvuts";

\end{verbatim}
my $trials = 30;
my @results;

for (my $trial = 0; $trial < $trials; ) {
    my $start = [gettimeofday];

    printf "type '%%s' using only one hand (trial %%d of %d)\n",
        $target, $trial + 1, $trials;

    my $input = <STDIN>;
    chomp($input);
    print "\n";

    if ($input eq $target) {
        $results[$trial] = tv_interval($start);
        $trial++;
    }
}

print "\n";

for (my $trial = 0; $trial < $trials; $trial++) {
    printf "%%d %%f\n", $trial + 1, $results[$trial];
}

4. Per the newsgroup, we assume that operation is error-free and that the user knows the target item and its location. We are only interested in movement time.

Using Microsoft Mouse data, $MT = 548 + 420 \log_2\left( \frac{A}{W} + 1 \right)$ (in milliseconds). We'll assume they're using standard drop-down menus. I measured a Firefox menu item; it was 175 pixels wide by 17 pixels high. We'll assume the user's cursor starts at 17 pixels above the center of the first menu item, and goes to the center of the correct menu item. Therefore, $A$ for menu item $i$ will be $17 \cdot i$ (with the first menu item having $i = 1$.) Since, according to Wikipedia, “$W$ is the width of the target measured along the axis of motion,” let $W = \frac{17}{2}$.

(a) We'll find the movement time for each item they could choose and then take average. The time to choose menu item $i$ is given by $MT = 548 + 420 \log_2\left( \frac{17i}{17/2} + 1 \right)$.

```
>> MT = inline('548 + 420 * log2((17 * i)/(17/2) + 1)', 'i'); mean(MT(1:12))
an = 2.0476e+03
```
2047 milliseconds.

(b) Assume the dynamic menu is the first four items at the top. The average time to select from the dynamic menu is 1589ms; for the static, 2279ms.

```
>> mean(MT(1:4))
an = 1.5858e+03
>> mean(MT(5:12))
an = 2.2785e+03
```
i. A. 50-50: 1934ms  
   B. 75-25: 1762ms  
   C. 90-10: 1658ms

ii. If the user’s choice is always in the dynamic menu (first four items), we can ignore the static items completely. Then the average time is 1589ms.

iii. If the item is always in the static menu, the top four items are not user; the user must always pass them on his way to the desired option. Then the average time is 2279ms.

(c) The constants in Hick’s Law depend on the tested scenario. In reality, our scenarios change all the time. For example, my reaction time for responding to my phone ringer is different when I’m in class, in a meeting, walking to class, and waiting for an interview call. For many products, there’s no one representative usage scenario for testing.