1. Thinking about user interface design.
A. Mobile/Cellular Phones
   - “Modern” mobile phones have only been in existence for a relatively short period of time. They have become common and are used for social communication as well as business correspondences. For the most part, they’ve been modeled after cordless home phones with the standard screen and number pad, albeit standalone and with more functionality.
   - 1. The familiarity that many people have with the home phone means that they can learn how to basically operate a cell phone. They know that you can just dial the number to call and press [Talk/Send/etc.] or vice versa.
   - 2. The size of most phones is well-suited for use by most people. They have reached the point where if they were any smaller, they’d become difficult to use (Zoolander), but if they were bigger, people would complain about bulkiness and lack of portability again.
   - 3. Being able to set the way a phone rings is a standard feature nowadays. Having a personalized ringtone ensures you know it’s your own phone ringing (and maybe even from a specific caller), vibrate mode is usually called “manner mode” for a reason, and silence allows you to receive calls without disturbing others.
   - 1. “Hold” commands are still present in some phones. Without reading the instructions, a user might not know, for instance, that he/she can hold a button on the side of the phone to toggle between ring modes. If such a button inadvertently gets held down, an unwanted ring may come up. A possible solution would be to put a dedicated and less accident-prone toggle on the phone (e.g., an indented switch).
   - 2. Some buttons on my phone are used for services to which I don’t subscribe. They can neither be re-mapped nor disabled. If a friend were to borrow my phone and try out one of these buttons, I would get charged at an increased rate for “impromptu usage” of the service. Perhaps the companies collaborated and designed the phones with this in mind, I don’t know. Otherwise, they can just get rid of the space-wasting buttons.
   - 3. This may just be a matter of personal preference, but I don’t like all of the useless features being dumped into cell phones. For those who prefer simple phones with minimal functionality there are few choices. Sure, they could just ignore the extra features, but each one adds more junk to skip while navigating through options.
B. TVs and Remote Controls
   - Everybody knows everyone knows how to operate a TV even if he/she doesn’t watch often. As technologies advanced, the evolution of TV sets has seen the screen sizes increase while the TV sets get thinner and the remotes (and number of remotes) get bigger.
   - 1. Remote controls add a level of convenience that is necessary by today’s standards. People do not want to get up to change the channel or volume. That is the reason.
   - 2. The actual television usually only has the minimal set of controls (such as power, volume, and channel). Not only does this keep the TV cleaner-looking; it also allows people who temporarily have no access to a remote to quickly start watching what they want, how they want, without having to go through a maze of buttons.
3. Many relatively new TVs have front-mounted audio/video input jacks for easy access to people who need to temporarily set up, say, a game console. These ports can be covered while not in use, which preserves these newer TVs’ aesthetic beauty.

- 1. Remote controls have too many controls! This problem especially plagues the multi-device or multifunction remotes. Some of the more seldom-used buttons can be put back on the TV itself (like the ones used during setup/calibration), and others can be omitted entirely (like “eject,” since you have to get up to change the CD anyway).

2. Also related to too many controls on remotes is the dependence on remotes. If certain functions like captions/sleep are available only on the remote, it would become impossible to use them if the remote ran out of batteries or got lost. Newer TVs don’t suffer major problems, but I’ve been in a situation where I couldn’t turn captions off because our remote control had been broken.

3. Most TV remotes today still use infrared to communicate with the TV, which means a direct line of sight is required between them. Simple little things like coffee mugs or stacks of paper can block the signal. A simple alternative is to use radio-frequency communications, but I’m not sure of the consequences (increased energy use?).

C. Microwave Ovens

- 1. Again, this is a common household accessory. Nearly all supermarkets have exclusively microwaveable frozen meals with easy cooking instructions. Most ovens are well-designed for their primary purpose but leave room for improvement.

2. Safety features built into most, if not all, ovens today include an automatic shut-off of the radiation if the front door is opened mid-cook. I’ve never seen firsthand what could happen if this mechanism weren’t in place, but I hope I won’t have to.

3. The most visible buttons are the ones that are most often used. Numbers for specifying the cook time, buttons for fine-tuning the time, and a defrost mode are usually among the first and foremost ones you see.

3. A rotating plate inside the main cooking chamber eliminates the old-fashioned cooking instruction of “turn food 180 degrees after X minutes.” Since the food being cooked is constantly rotating, a more even distribution of energy is ensured.

- 1. Some of the useless features I’ve seen include toggles for specific types of food. Cooking instructions for those foods are generally specified on the packaging, if it is microwaveable at all.

2. Multi-function buttons sometimes can’t be used, such as during cooking operations. A few are nice (like “add 30 seconds”), but the others should just replace the otherwise useless buttons mentioned in point 1.

3. I can’t really think of any more major negatives for microwaves, but something nice to have would be some sort of temperature monitor, such as an infrared thermometer. This would be able to warn consumers when the food or container that’s just been heated is too hot to handle or still too cold to eat.
2. Learning to listen to users.

1. Interviewee:
   “A problem I frequently encounter while using my system or applications on my system is the problem of their existence. Some programs such as Adobe Reader and Mozilla Fire should have never been made. They are simply horrible pieces of garbage that leak immense amounts of memory as if the computer had Alzheimer’s. They crash, freeze, and occasionally have a massive out on my machine forcing me to reboot. To answer the question about how frustrating it is; let’s put it this way, if I were a masochist I’d be having orgasms using these pieces of crap. They were simply made to retard any progress on your computer.”

This disgruntled computer user has technical knowledge about the problems he encounters; memory leaks consume extra memory and consequently slow down the rest of the system if serious enough. There really is nothing that can fix this problem other than fixing the software themselves. An alternative would be to use different programs, but then new problems may crop up.

2. Interviewee:
   “Problems encountered: system crashes etc., semi-hard-to-use organization system with folders, driver incompatibilities, file sizes
   Respective frustration levels: very frustrating, frustrating, frustrating, very frustrating
   Effect on ability to work: high, low, medium, medium
   Possible corrections: better OS programming/implementation, brainstorm for more creative features for file storage, better driver programming, lower file size for everything!!!”

System crashes and bad drivers are most likely related, folder management is user preference, and file sizes are the result of compression and encryption algorithms. These problems are cited by a notebook/laptop user, and I can understand the frustration with drivers and file sizes. There isn’t much we can do about drivers and crashes; it’s usually up to manufacturers to fix them for us. Smaller file sizes can be achieved by compressing more heavily, but for some files (multimedia, specifically) there’d be a tradeoff in quality. Lastly, I haven’t seen any other hierarchical system besides directories, but I have no suggestions at this time.

3. Interviewee:
   “1.) I don't like when my computer freezes. It decreases my productivity in games, such as TF2, by decreasing my rate of killing directly proportional to the time I'm stuck frozen plus reboot time. What can be done about this? You figure it out, CS major.
   2.) I really dislike blue screens of death. Solution? Make them a more friendly color, such as lavender.
   3.) Wireless mice with short ranges. Make the wireless range longer and better, to prevent choppy scrolling and poor aiming of pipe bombs/Natascha.”

It looks like this person is angered most by problems that occur during a game. Once again, crashing and freezing are due to manufacturer-provided drivers or faulty management by the OS, and there’s almost nothing, besides fixing the drivers ourselves, that we can do. Friendlier colors to use with error messages could be nice, though. Wireless mouse manufacturers can do more testing or have better QA when releasing their products. This would minimize the amount of faulty items sold.

<table>
<thead>
<tr>
<th>Trial Number</th>
<th>Time Taken (seconds)</th>
<th>Correct Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>11.8713</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>8.33861</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>7.94059</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>6.31186</td>
<td>Yes</td>
</tr>
<tr>
<td>5</td>
<td>6.60986</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>5.76040</td>
<td>Yes</td>
</tr>
<tr>
<td>7</td>
<td>5.41069</td>
<td>Yes</td>
</tr>
<tr>
<td>8</td>
<td>5.68911</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>5.16322</td>
<td>Yes</td>
</tr>
<tr>
<td>10</td>
<td>4.69111</td>
<td>Yes</td>
</tr>
<tr>
<td>11</td>
<td>5.83553</td>
<td>Yes</td>
</tr>
<tr>
<td>12</td>
<td>5.00265</td>
<td>Yes</td>
</tr>
<tr>
<td>13</td>
<td>4.11200</td>
<td>Yes</td>
</tr>
<tr>
<td>14</td>
<td>4.16622</td>
<td>Yes</td>
</tr>
<tr>
<td>15</td>
<td>5.56515</td>
<td>Yes</td>
</tr>
<tr>
<td>16</td>
<td>7.58291</td>
<td>Yes</td>
</tr>
<tr>
<td>17</td>
<td>5.89135</td>
<td>Yes</td>
</tr>
<tr>
<td>18</td>
<td>5.82228</td>
<td>Yes</td>
</tr>
<tr>
<td>19</td>
<td>6.41644</td>
<td>Yes</td>
</tr>
<tr>
<td>20</td>
<td>3.75292</td>
<td>Yes</td>
</tr>
<tr>
<td>21</td>
<td>5.72393</td>
<td>Yes</td>
</tr>
<tr>
<td>22</td>
<td>5.44364</td>
<td>Yes</td>
</tr>
<tr>
<td>23</td>
<td>3.13511</td>
<td>Yes</td>
</tr>
<tr>
<td>24</td>
<td>3.76561</td>
<td>Yes</td>
</tr>
<tr>
<td>25</td>
<td>3.33443</td>
<td>Yes</td>
</tr>
</tbody>
</table>

\[ T_n = 10.266n^{0.27} \]
The learning constant ($\alpha$) is about 0.27. This may be attributed to the fact that the subject sacrificed time to achieve a “perfect” result set (i.e., have no mistakes listed). As we can see from the chart, only when typing became fast did the mistakes and fixing become more common. My program recorded trials with incorrect input in other test takers, but not all trials were finished in those experiments, and thus those data were inadequate.

Source:

// Power Law of Practice:
// This is a simple prompter to test a user’s
// proficiency over time of a repeated task.
// Results are logged to a file.
// By Jason Chen
// Created for CS 465 Homework 2 Problem 3 in Fall 2008

#include <iostream>
#include <fstream>
#include <string>
#include <sys/time.h>

using namespace std;

const int numTrials = 25;
const string prompt1 = "Using one hand, quickly type the last 8 letters of";
const string prompt2 = "the alphabet backwards, followed by Enter/Return.";
const string correct = "zyxwvuts";
const bool countIncorrectAsTrial = false;
const char fileName[] = "results.txt";

void makeLowerCase(string in) { 
    for (int i = 0; i < in.size(); i++) {
        in[i] = tolower(in[i]);
    }
}

int main(int argc, char* argv[]) {
    int trialCount = 0;
    string input;
    timeval tim;
    double startTime;
    double time;

    ofstream log;
    log.open(fileName, ios::out);
    if (!log) {
        cerr << fileName << " failed to open for writing." << endl;
        return EXIT_FAILURE;
    }

    cout << "Welcome to the test. You must complete "
        << numTrials << " trials." << endl;

    while (trialCount < numTrials) {
        cout << endl << prompt1 << endl << prompt2 << endl;
        cin >> input;
        cout << endl;
        trialCount++;
    }

    log << "Type anything and press Enter/Return to start.";
    cin >> input;
    log << "\n";
    log << "";
    log.close();
    return EXIT_SUCCESS;
}
gettimeofday(&tim, NULL);
startTime = tim.tv_sec + (tim.tv_usec / (double)CLOCKS_PER_SEC);
cin >> input;
gettimeofday(&tim, NULL);
time = tim.tv_sec + (tim.tv_usec / (double)CLOCKS_PER_SEC) - startTime;
makeLowerCase(input);
if (input == correct) {
    log << "trial " << trialCount << ": "
        << time << " seconds" << endl;
    trialCount++;
} else {
    cout << "What you entered was incorrect. Try again." << endl;
    log << "trial " << trialCount << ": "
        << time << " seconds (incorrect input)" << endl;
    if (countIncorrectAsTrial) {
        trialCount++;
    }
}

cout << endl << "Thanks for completing the test.";
log.close();
return EXIT_SUCCESS;

   (a) 3.585 * 0.15 = 0.538 seconds
   (b) 50/50: 0.419 seconds
       75/25: 0.359 seconds
       90/10: 0.324 seconds
       Minimum choice time: This is about 0.300 seconds, when the probability split is ~100/~0.
   (c) Hick’s Law shouldn’t be used when the list of options is extremely lengthy. It also probably wouldn’t work in situations where the items from which to be selected are randomly arranged. Also, it would diminish in usefulness after the user gets used to the system (as seen in the power law of practice).