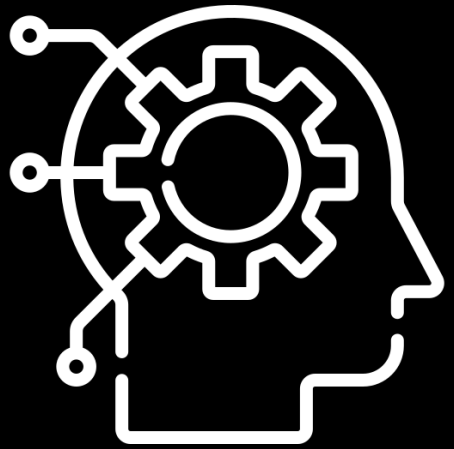


Learnability

Web App Development

Winter 2023
Thompson Rivers University

User Interface Hall of Shame

Source: Interface Hall of Shame

This dialog box, which appeared in a program that prints custom award certificates, presents the task of selecting a template for the certificate.

This interface is clearly graphical.

It’s mouse-driven. No memorizing or typing complicated commands.

It’s even what-you-see-is-what-you-get (WYSIWYG)

the user gets a preview of the award that will be created.

So why isn’t it usable?

User Interface Hall of Shame (repeated)

Source: Interface Hall of Shame

The first clue that there might be a problem here is the long help message on the left side.

Why so much help for a simple selection task?
Because the interface is bizarre!

User Interface Hall of Shame (repeated)

Source: Interface Hall of Shame

The scrollbar is used to select an award template.

Each position on the scrollbar represents a template,
and moving the scrollbar back and forth changes the template shown.

This is a cute but poor use of a scrollbar.

Notice that the scrollbar doesn’t have any marks on it.

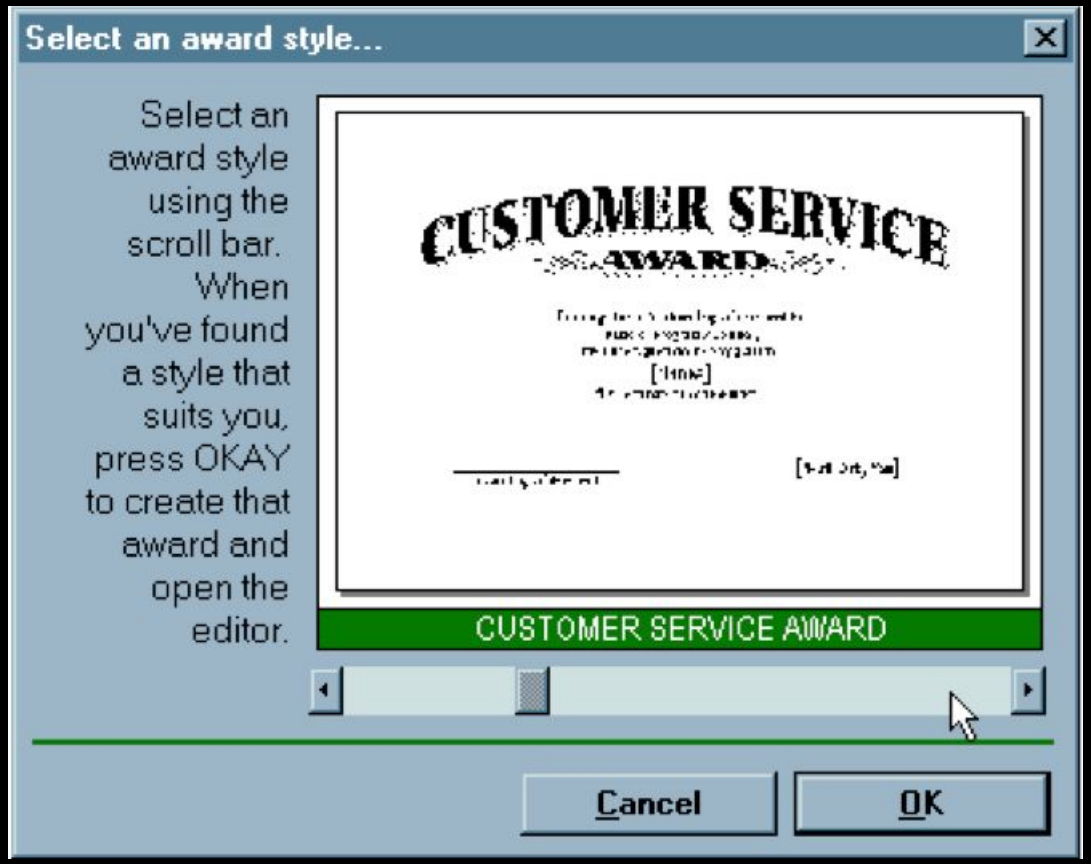
How many templates are there?

How are they sorted?

How far do you have to move the scrollbar to select the next one?

You can’t even guess from this interface.

User Interface Hall of Shame



Source: Interface Hall of Shame

Normally, a horizontal scrollbar underneath an
image (or document, or some other content)
is designed for scrolling the content horizontally.

A new or infrequent user looking at the window
sees the scrollbar, assumes it serves that function, and ignores it.

Inconsistency with prior experience and other applications
tends to trip up new or infrequent users.

We’ve all seen those apparently-pullable door handles with a little sign that says “Push”; and many of us have had the embarrassing experience of trying to pull on the door before we notice the sign.

The help text on this dialog box is filling the same role here.

But the dialog doesn’t get any better for frequent users, either.

If a frequent user wants a template they’ve used before, how can they find it?

Surely they’ll remember that it’s 56% of the way along the scrollbar?

This interface provides no shortcuts for frequent users.

In fact, this interface takes what should be a random access process and transforms it into a linear process.

Every user has to look through all the choices,
even if they already know which one they want.

The computer scientist in you should cringe at that algorithm.

Even the help text has usability problems.

“Press OKAY”?

Where is that?

And why does the message have a ragged left margin?

You don’t see ragged left too often in newspapers and magazine layout, and there’s a good reason.

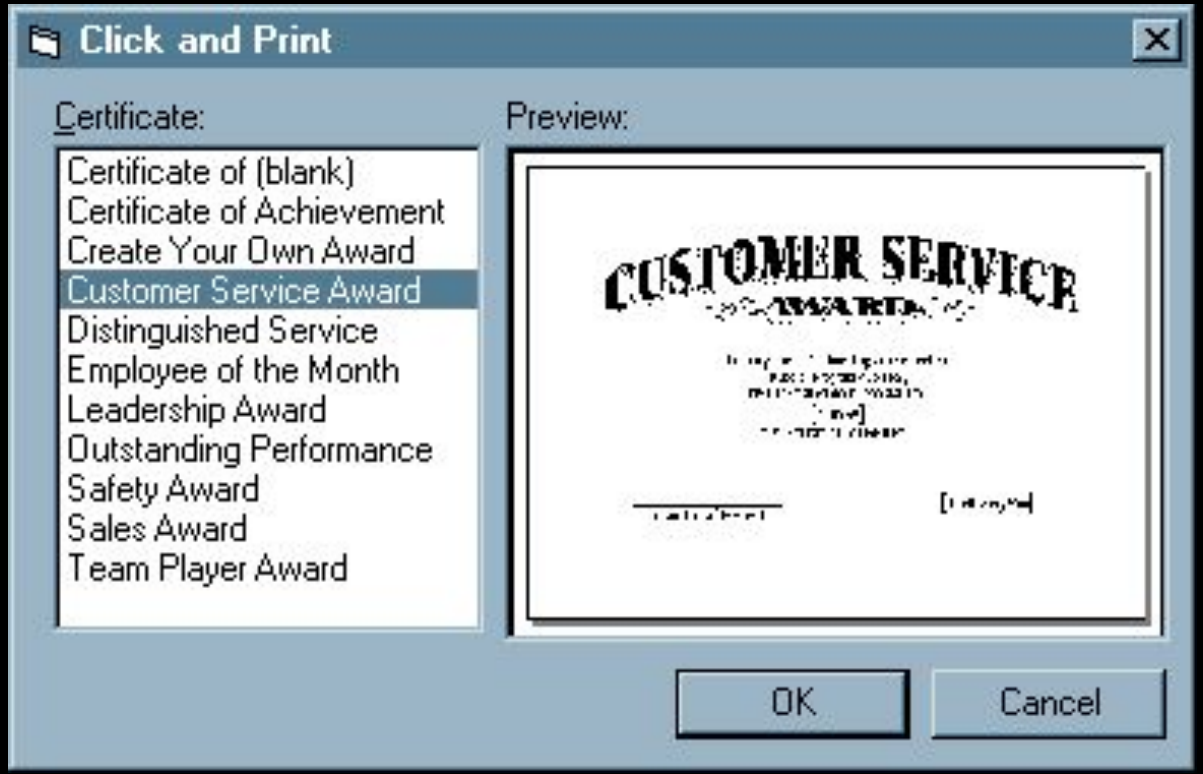
Usability can’t be left until the end of software development, like package artwork or an installer.

It can’t be patched here and there with extra messages or more documentation.

It must be part of the process, so that usability bugs can be fixed, instead of merely patched.

How could this dialog box be redesigned to solve some of these problems?

The Example, Redesigned



Source: Interface Hall of Shame

Here’s one way it might be redesigned.

The templates now fill a listbox on the left;
selecting a template shows its preview on the right.

This interface suffers from none of the problems of its predecessor:
list boxes clearly afford selection to new or infrequent users;
random access is trivial for frequent users.

And no help message is needed.

Definition

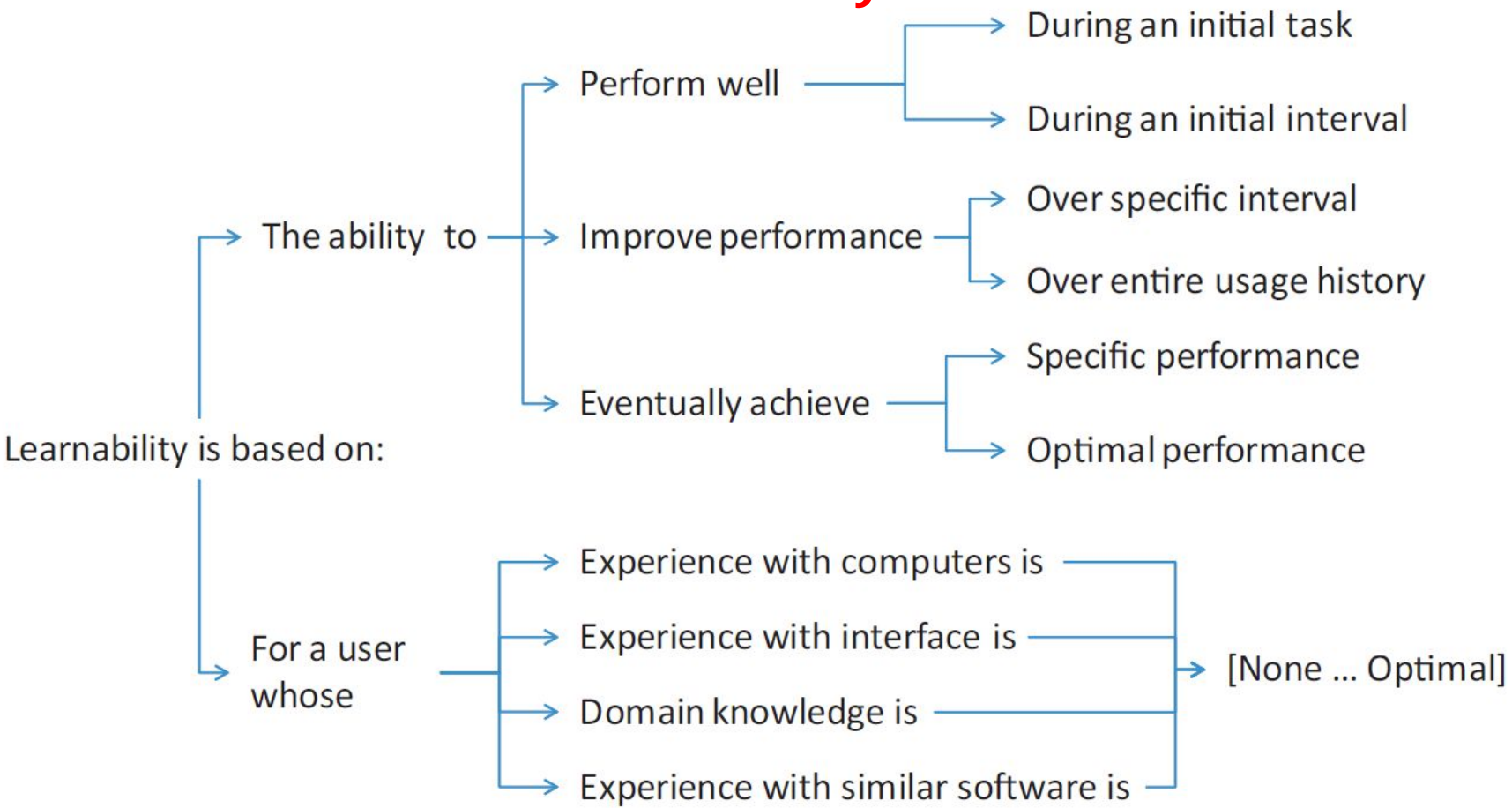
Initial Learnability

- The system should be easy to learn by the class of users for whom it is intended.
  - Michelsen 1980
- the effort required for a typical user to be able to perform a set of tasks … with a predefined level of proficiency
  - Santos & Badre
- Allowing users to rapidly begin to work with the system
  - Holzinger

Extended Learnability

- ease at which new users can begin effective interaction and achieve maximal performance
  - Dix et al.
- Minimally useful with no formal training, and should be possible to master the software
  - Riemann

Taxonomy



Source: Grossman et al., A Survey of Software Learnability Metrics

Combination

Learning = Discovery + Understanding + Memory

Discoverability: what can be done?

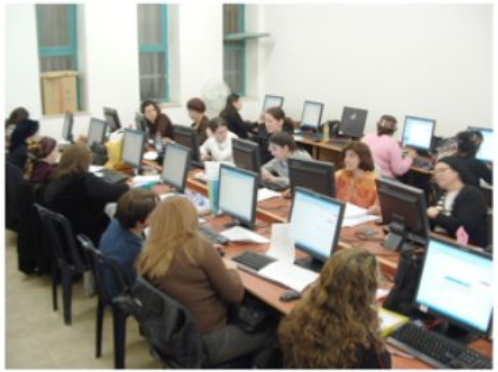
Understanding: what are the effects?

Memory: retain for next time

How we Learn

How We Don't Learn

Not by reading a manual*



Not by taking a class*

Not by reading the help first*

* Standard caveat: “it depends”

Source: Interface Hall of Shame

When computers first appeared in the world,
there were some assumptions about how people would learn how to use the software.

Programmers assumed that users would read the manual first!

obviously not true.

Companies assumed that their employees would take a class first

not always true.

Even now that we have online help built into virtually every desktop application,
and web page help often just a search engine query away,
users don’t go to the help first or read overviews.

All these statements have to be caveated,
because in some circumstances–some applications, some tasks, some users–these might very well be the way the user learns.

Very complex, professional-level tools might well be encountered in a formal training situation–that's how pilots learn how to use in-cockpit software, for example.

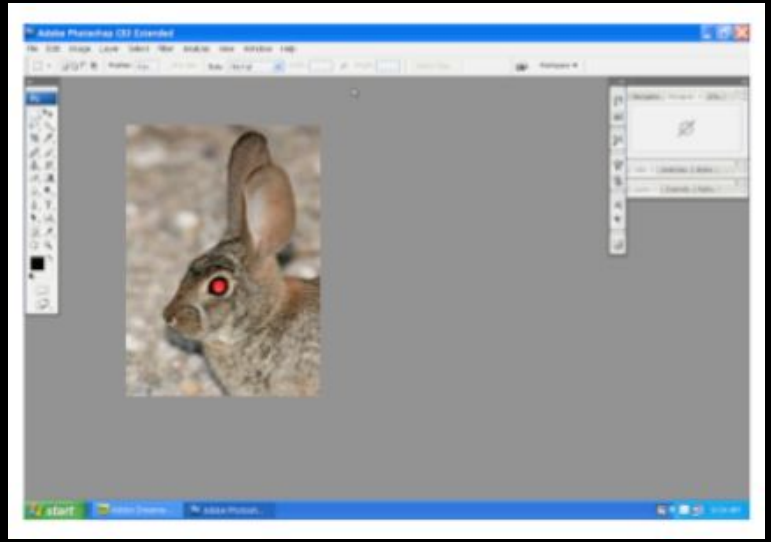
And some users (very few of them) do read manuals.

Nearly all the general statements we make in this lecture should be interpreted as “It Depends.”

There will be contexts and situations in which they’re not true, and that’s one of the complexities of UI design.

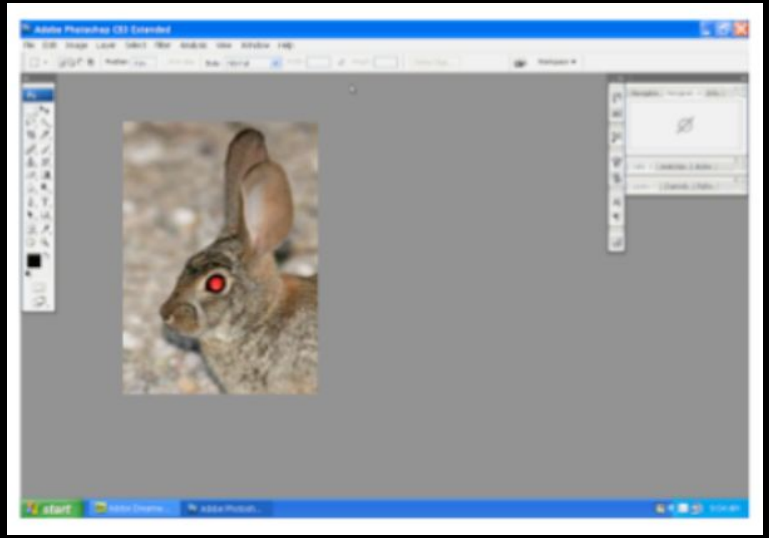
Learning by Doing

- User arrives with a **goal**
  - “Get rid of the redeye from my photo.”
- **Explores** interface for way to satisfy the goal.
  - doesn't care about learning interface, except for goal
- Key: next step should be **discoverable**
  - ideally, **obvious**
- With **feedback** to show progress



Learning by Doing

- User arrives with a **goal**
  - “Get rid of the redeye from my photo.”
- **Explores** interface for way to satisfy the goal.
  - doesn't care about learning interface, except for goal
- Key: next step should be **discoverable**
  - ideally, **obvious**
- With **feedback** to show progress
- Caveat: Highly sophisticated UIs require classes & manuals:
  - pilots, air traffic control, nuclear reactors, etc.

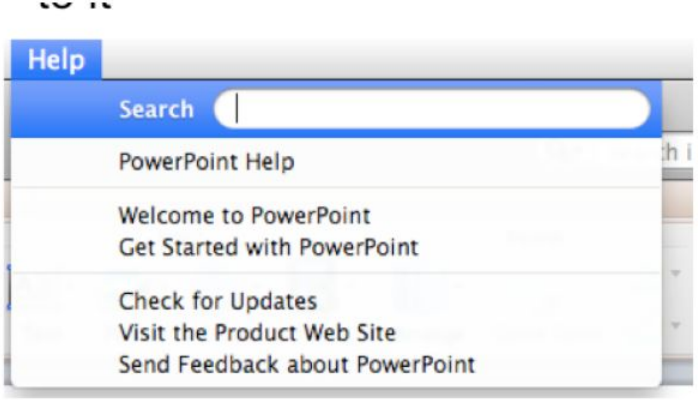


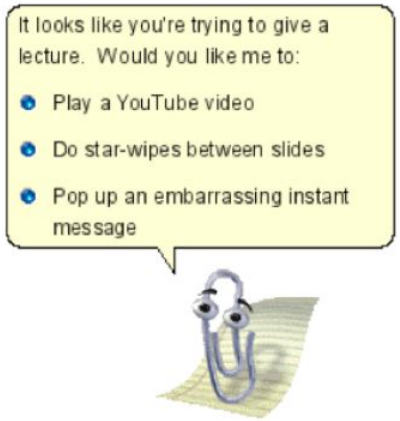
So users don’t try to learn first – instead, they typically try to do what they want to do, and explore the interface to see if they can figure out how to do it.

This practice is usually called **learning by doing**,
and it means that the user is starting out with a goal already in mind;
they are more interested in achieving that goal
than in learning the user interface.

The burden is on the user interface to
clearly communicate how to use it and
help the user achieve their first goal at the same time.

Seeking Help



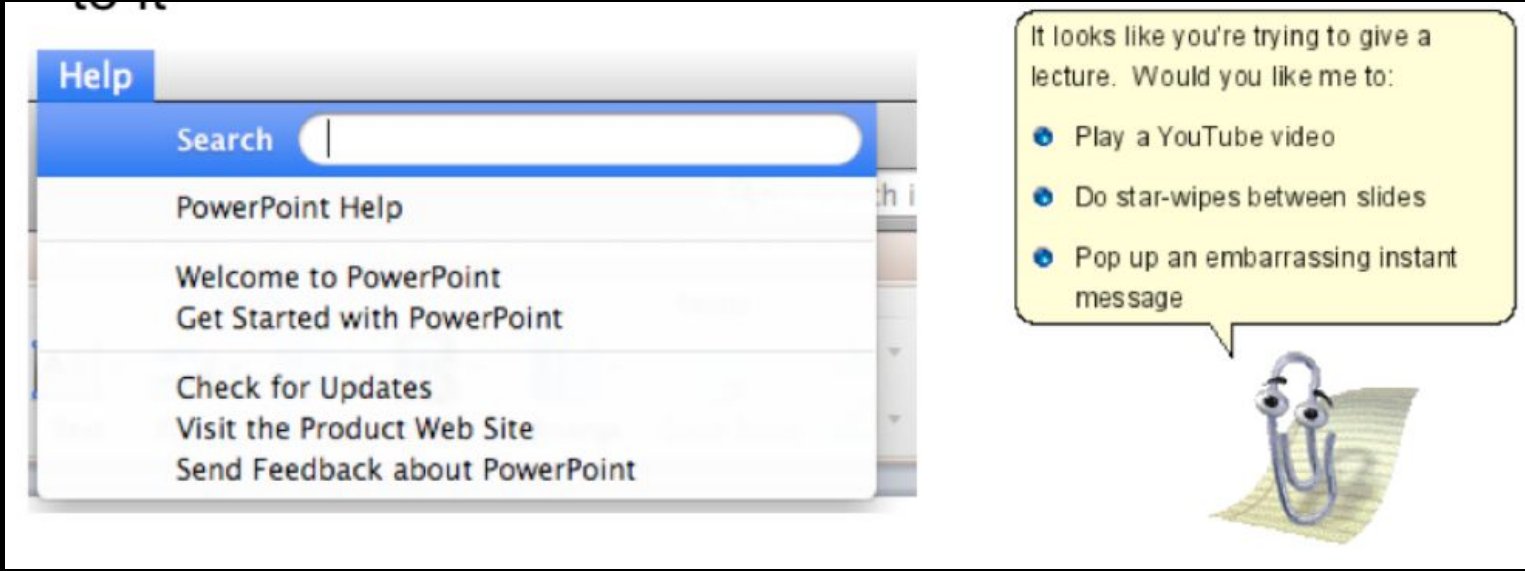


- User resorts to seeking help when they get stuck
- Already has a problem when they arrive
- Looking for concrete solutions to it
- **Not** interested in philosophy of system
- RTFM vs. Stack Exchange

Seeking Help

- User resorts to seeking help when they get stuck
- Already has a problem when they arrive
- Looking for concrete solutions to it

- Not interested in philosophy of system
- RTFM vs. Stack Exchange
- Reading textbooks: to learn, or to answer the homework question?



Only when they get stuck in their learning-by-doing
will a typical user look for help.

This affects the way help systems should be designed,
because it means most users (even first-timers)
are arriving at help with a goal already in mind
and an obstacle they've encountered to achieving that goal.

A help system that starts out with a long text explaining
The Philosophy of the System will not work.

That philosophy will be ignored,
because the user will be seeking answers to their specific problem.

Modern help systems understand this,
and make it easy to ask for the user to ask the question up-front,
rather than wading through pages of explanation.

Lessons for Designers

- Know the user’s goals when we design
- User interface should communicate how it works and how to use it
- Next step to goal should always be discoverable
- Help must be searchable and goal-oriented

The fact that users are learning our interfaces by actually using them has some implications for how we should design them.

First, we should know something about what the users' goals actually are.

Collecting information about that is a critical feature of the user-centered design process that we'll talk about in a these lectures.

If we're designing for the wrong goals,
users are going to struggle to figure out
how to do what they want in our system.

Second, the UI should be the primary teacher of how to use it.

The UI itself must communicate as clearly as possible
how it's supposed to be used, so that users can match their goals with appropriate actions in the system.

Later, we'll talk about a few specific techniques for doing this

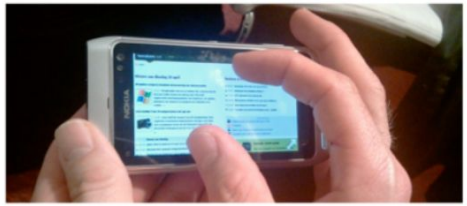
affordances, feedback, and information scent.

Third, when the user does have to resort to help,
that help should be searchable and goal-directed.

Providing a 30-minute video tutorial probably won't help
people who learn by doing.

Learning by Watching

- Learn lots from watching other users
- Unfortunately, many tools are used alone



How did you learn Alt-Tab?

Another way we learn how to use user interfaces is by watching other people use them.

That’s a major way we navigate an unfamiliar subway system, for example.

Unfortunately much of our software
–whether for desktops, laptops, tablets, or smartphones–
is designed for one person, and you don't often use it together with other people, reducing the opportunities for learning by watching.

Yet seeing somebody else do it may well be the only way you can learn about some features that are otherwise invisible.

For example, you probably know how to use Alt-Tab to switch between windows.

How did you learn that?

The UI itself certainly didn't communicate it to you.

Social computing is changing this situation somewhat.

We'll look at Twitter in a this lecture, and see that you can learn some things from other people even though they're not sitting next to you.