



User-Centred Design

CITS1220 Software Engineering



Lecture Overview

- Users and Usability: some definitions
- Usability Principles
- Evaluating user interfaces

- Next: Building GUIs with Java Swing



Users and Usability



Use Cases: some definitions

- **Use Case:** A general series of interactions between one or more actors and a system
- **Scenario:** An *instance* of a use case. A scenario represents a concrete sequence of interactions.
- **Actor:** An external entity that needs to exchange information with the system. An actor can represent either a *user role* or *another system*.

Use case example

<i>Use case name</i>	PlanTrip
<i>Entry condition</i>	1. The Driver activates her home computer and logs into the trip planning web service
<i>Flow of events</i>	2. Upon successful login, the Driver enters constraints for a trip as a sequence of destinations
	3. Based on a database of maps, the planning service computes the shortest way visiting the destinations in the specified order. The result is a sequence of segments binding a series of crossings and a list of directions.
	4. The Driver can revise the trip by adding or removing destinations.
<i>Exit condition</i>	5. The Driver saves the planned trip by name in the planning service database for later retrieval.

Use case example

<i>Use case name</i>	ExecuteTrip
<i>Entry condition</i>	1. The Driver starts her car and logs into the onboard route assistant.
<i>Flow of events</i>	2. Upon successful login, the Driver specifies the planning service and the name of the trip to be executed.
	3. The onboard route assistant obtains the list of destinations, directions, segments, and crossings from the planning service.
	4. Given the current position, the route assistant provides the driver with the next set of directions.
<i>Exit condition</i>	5. The Driver arrives at her destination and shuts down the route assistant.



User Centred Design

Software development should focus on the needs of users

- Understand your users
- Design software based on an understanding of the users' tasks
- Ensure users are involved in decision making processes
- Design the user interface following guidelines for good usability
- Have users work with and give their feedback about prototypes, on-line help and draft user manuals

Why focus on users ?

- Reduced training and support costs
- Reduced time to learn the system
- Greater efficiency of use
- Reduced costs by only developing features that are needed
- Reduced costs associated with changing the system later
- Better prioritizing of work for iterative development
- Greater attractiveness of the system, so users will be more willing to buy and use it



Characteristics of Users

SW engineers need to understand users:

- Goals for using the system
- Potential patterns of use
- Demographics (characteristics of the population)
- Their knowledge of the domain and of computers
- Physical ability
- Psychological traits and emotional feelings



Basics of UI Design

- User interface design should be done in conjunction with other software engineering activities.
- Do use case analysis to help define the tasks that the UI must help the user perform.
- Do *iterative* UI prototyping to address the use cases.
- Results of prototyping will enable you to finalize the requirements.

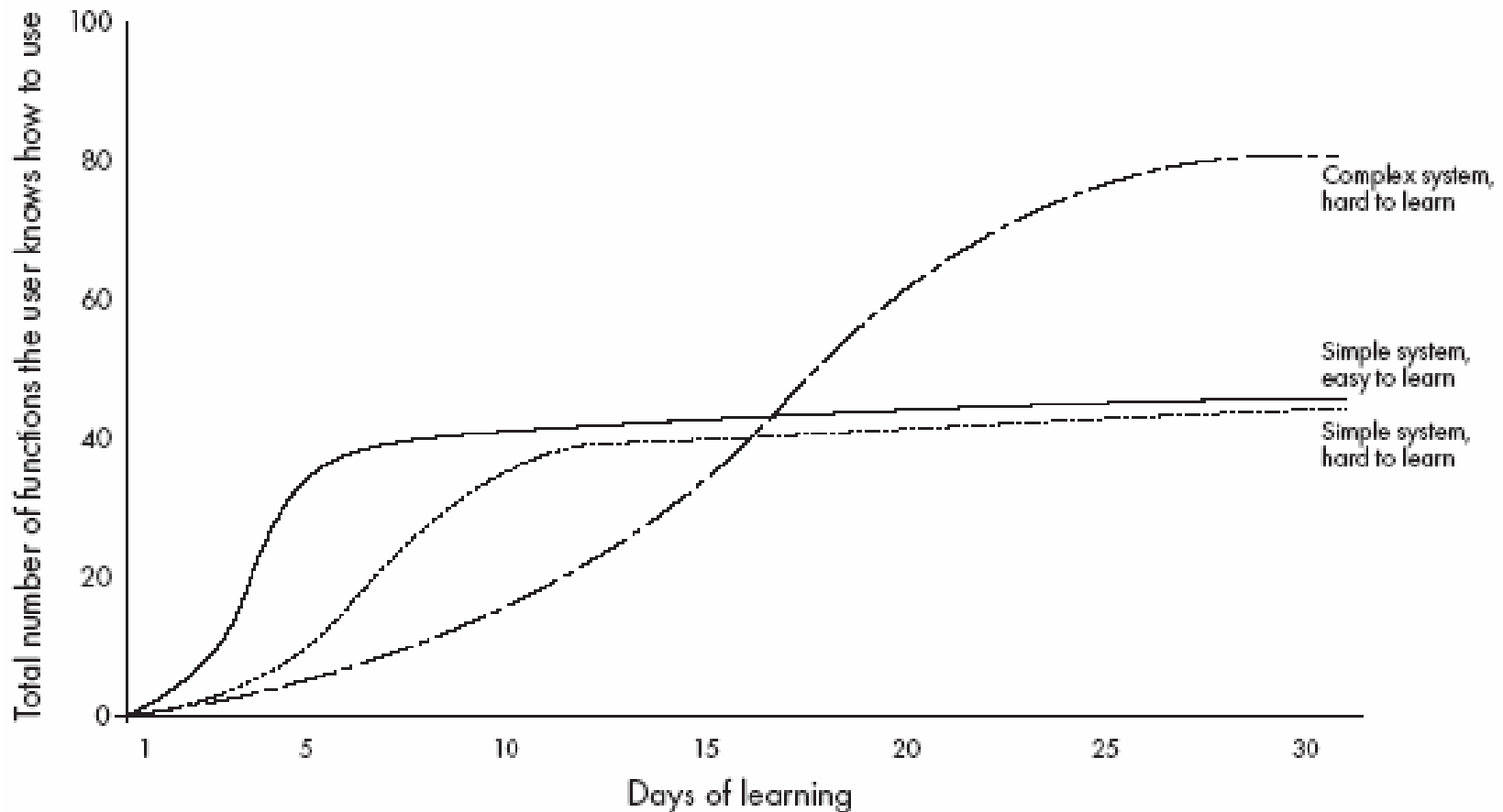
Usability vs. Utility

- Does the system provide the *raw capabilities* to allow the user to achieve their goal?
 - This is *utility*.
- Does the system allow the user to *learn and to use* the raw capabilities *easily*?
 - This is *usability*.
- *Both utility and usability are essential*
 - They must be measured in the context of particular types of users.

Aspects of usability

- Learnability
 - The speed with which a new user can become proficient with the system.
- Efficiency of use
 - How fast an expert user can do their work.
- Error handling
 - The extent to which it prevents the user from making errors, detects errors, and helps to correct errors.
- Acceptability.
 - The extent to which users *like* the system.

Different learning curves

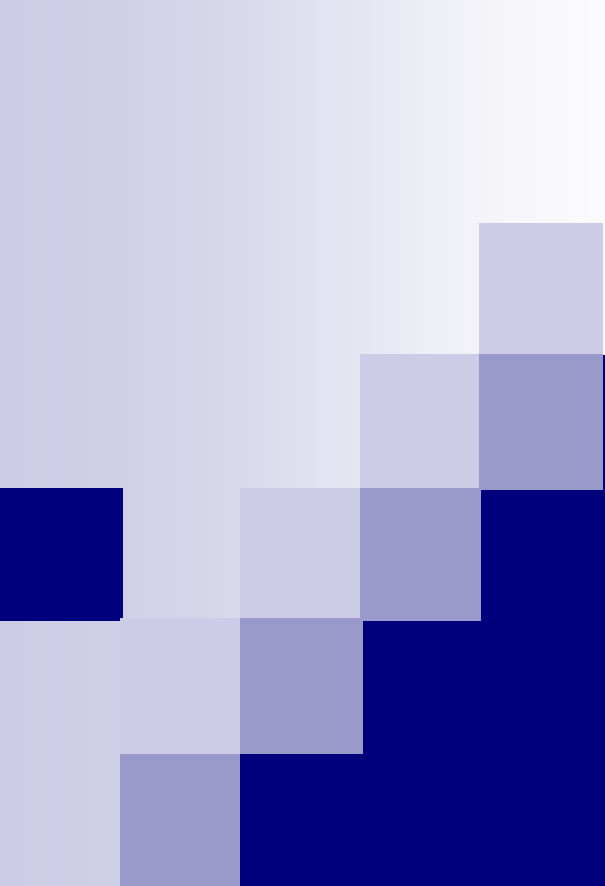


Terminology for UI design


- **Dialog:** A specific window with which a user can interact, but which is not the main UI window.
- **Control** or **Widget:** Specific components of a user interface.
- **Affordance:** The set of operations that the user can do at any given point in time.
- **State:** At any stage in the dialog, the system is displaying certain information in certain widgets, and has a certain affordance.
- **Mode:** A situation in which the UI restricts what the user can do.
- **Modal dialog:** A dialog in which the system is in a very restrictive mode.
- **Feedback:** The *response from the system* whenever the user does something, is called feedback.
- **Encoding techniques.** Ways of encoding information so as to communicate it to the user.

Some encoding techniques


- Text and fonts
- Icons
- Photographs
- Diagrams and abstract graphics
- Colours
- Grouping and bordering
- Spoken words
- Music
- Other sounds
- Animations and video
- Flashing





12 Usability Principles

- 
- 1. Do not rely only on usability guidelines – *always test with users.*
 - Usability guidelines have exceptions; you can only be confident that a UI is good if you test it successfully with users.

 - 2: Base UI designs on users' *tasks.*
 - Perform use case analysis to structure the UI.


- 
- 3: Ensure that the sequences of actions to achieve a task are as *simple* as possible.
 - Reduce the amount of reading and manipulation the user has to do.
 - Ensure the user does not have to navigate anywhere to do subsequent steps of a task.

- 
- 4: Ensure that the user always knows what he or she can and should do next.
 - Ensure that the user can see *what commands are available* and are not available.
 - Make the *most important commands stand out*.


- 
- 5: Provide good feedback including effective error messages.
 - Inform users of the *progress* of operations and of their *location* as they navigate.
 - When something goes wrong explain the situation in adequate detail and *help the user to resolve the problem*.

- 6: Ensure that the user can always get out, go back or undo an action.
 - Ensure that all operations can be *undone*.
 - Ensure it is easy to *navigate back* to where the user came from.

- 7: Ensure that response time is adequate.
 - Users are very sensitive to slow response time
 - They compare your system to others.
 - Keep response time less than a second for most operations.
 - Warn users of longer delays and inform them of progress.

- 
- 8: Use *understandable encoding techniques*.
 - Choose encoding techniques with care.
 - Use labels to ensure all encoding techniques are fully understood by users.

 - 9: Ensure that the UI's appearance is *uncluttered*.
 - Avoid displaying too much information.
 - Organize the information effectively.

- 
- 10: Consider the needs of *different groups* of users.
 - Accommodate people from different *locales* and people with *disabilities*.
 - Ensure that the system is usable by both *beginners* and *experts*.

 - 11: Provide all necessary *help*.
 - Organize help well.
 - Integrate help with the application.
 - Ensure that the help is accurate.



■ 12. Be *consistent*.

- Use similar layouts and graphic designs throughout your application.
- Follow look-and-feel standards.
- Consider mimicking other applications.

Example (bad UI)

Dotumlia Sign Up

File Edit Help

- Personal Info...
- Add Addresses...
- Add Services...

Welcome to
Ootumlia Services

To sign up, use the **Edit** menu

Dotumlia Sign Up

Payment

Name:

Number:

Expiration date:

Amount: \$20.00

Cancel OK

Dotumlia Sign Up

Personal Information

Name:

Your Email:

OK Cancel

Street:

Municipality:

Country:

Postal Code:

Phone:

Type:

OK Cancel

Dotumlia Sign Up

Signing you up...

Cancel

Example (better UI)

Ootumlia Sign Up

Welcome to
Ootumlia Services

To sign up, click on Start

Cancel Start

Ootumlia Sign Up

Step 1: Personal Information

Name:

Existing Email:

Addresses

Home Work Mailing

Street:

Municipality:

Country:

Postal Code:

Phone:

<< Prev Next >>

Ootumlia Sign Up

Step 5: Payment

Amex Visa MasterCard

Number:

Expiration date:

Total monthly fee: \$20.00

My credit card will be debited
the first day of each month
for the above amount

<< Prev Cancel I agree

Ootumlia Sign Up

The system is now dialing in
to register you for our services.

Please stand by...

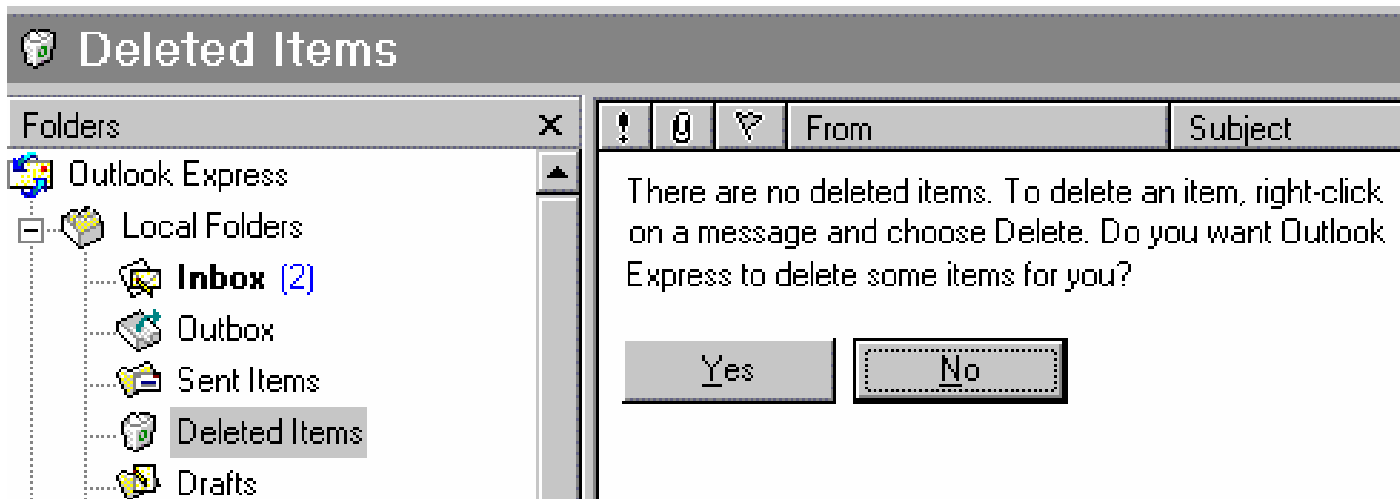
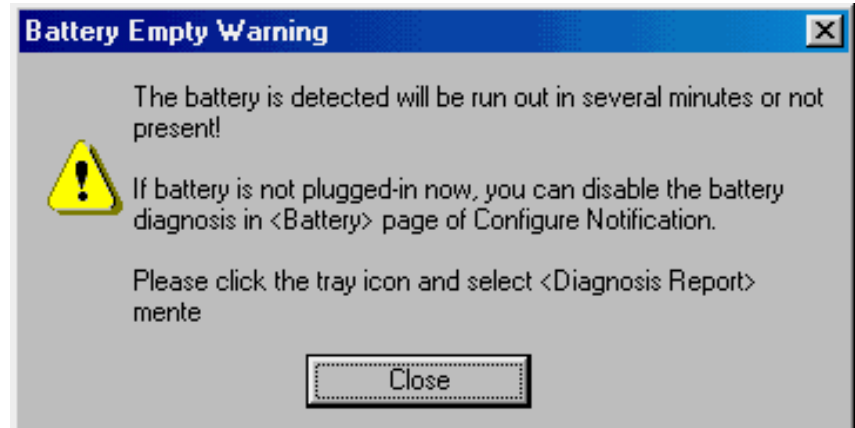
About 5 seconds remaining...

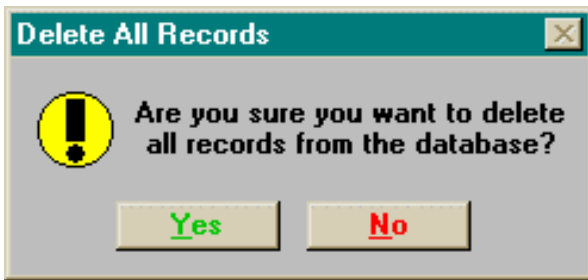
Cancel

Interface Hall of Shame

<http://homepage.mac.com/bradster/iarchitect/new.htm>

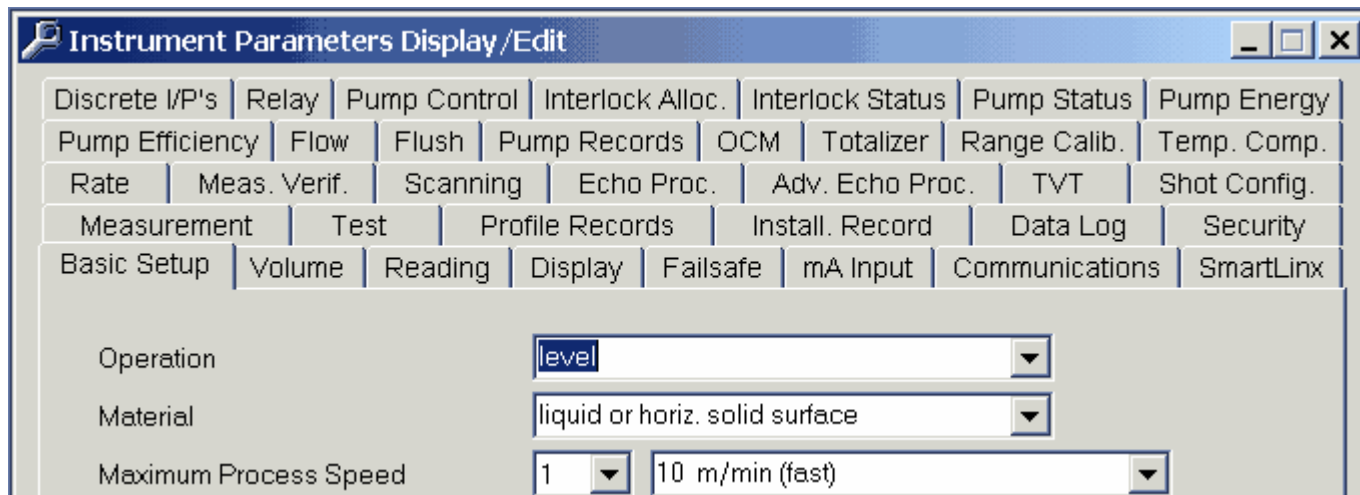
Nobody likes a stupid computer. However, many applications interrupt the user to ask stupid questions, provide meaningless information, or require the user to make what should be an obvious selection.

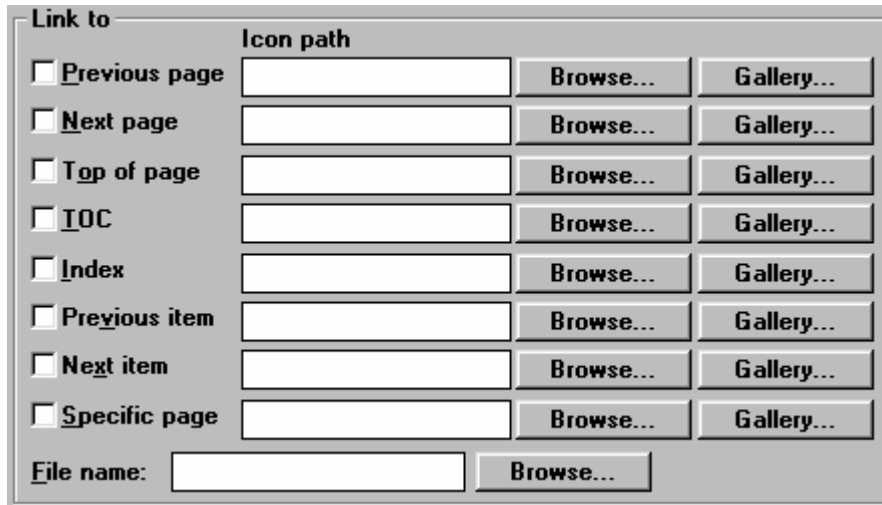




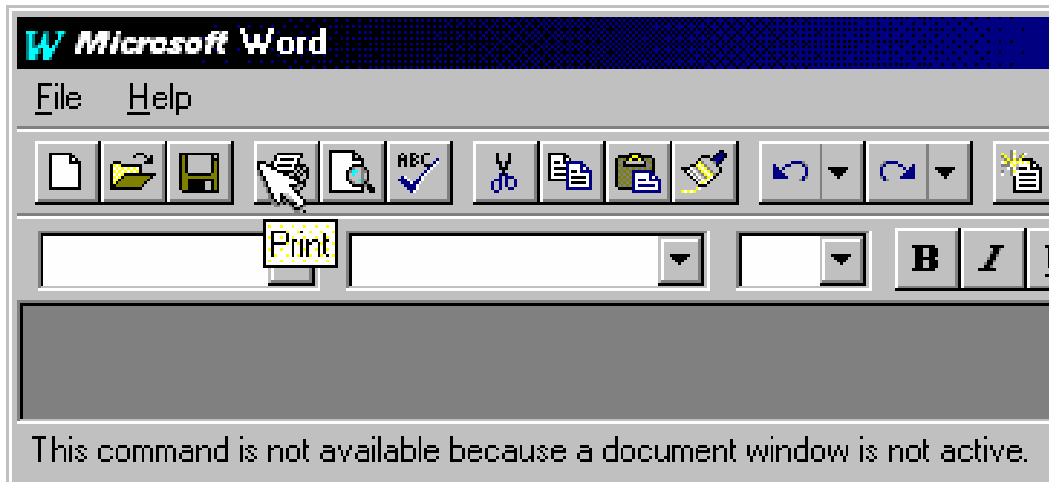
All affirmative buttons (OK, Yes, Open) have green-colored text and All negative buttons (Cancel, No, Close) have red-colored text. But beware the question!

One row of tabs is enough.
Again: One row is enough.
And once more: One row is enough.





when you have to devote more than, say, 25% of your window's real estate to command buttons, you probably have too many buttons. You should never have to duplicate a command button more than once on the same window, never mind 9 times, and certainly not 15 times.



One of the basic rules of GUI design is that controls that are disabled should *appear* to be disabled. One can normally tell at a glance whether or not a particular menu item, command button, list, or drop-down control is available.

How *not* to use drop down lists



Inconsistent use of icons can potentially lead to problems. The toolbar icon for the Delete command appears remarkably similar to the Window Close button (in the upper right corner), and is exactly the same as the Cancel icon in the widely-used but frowned-upon Borland-style command buttons.





Evaluating User Interfaces

Heuristic evaluation

1. Pick some use cases to evaluate.
2. For each window, page or dialog that appears during the execution of the use case
 - Study it in detail to look for possible usability defects.
3. When you discover a usability defect write down the following information:
 - A short description of the defect.
 - Your ideas for how the defect might be fixed.

Evaluation by observing users

- Select users corresponding to each of the most important actors
- Select the most important use cases
- Write sufficient instructions about each of the scenarios
- Arrange evaluation sessions with users
- Explain the purpose of the evaluation
- Preferably videotape each session
- Converse with the users as they are performing the tasks
 - When the users finish all the tasks, de-brief them
 - Take note of any difficulties experienced by the users
 - Formulate recommended changes

Difficulties and Risks

□ **Users differ widely**

- *Account for differences among users when you design the system.*
- *Design it for internationalization.*
- *When you perform usability studies, try the system with many different types of users.*

□ **User interface implementation technology changes rapidly**

- *Stick to simpler UI frameworks widely used by others.*
- *Avoid fancy and unusual UI designs involving specialized controls that will be hard to change.*

Difficulties and Risks

- **User interface design and implementation can often take the majority of work in an application:**
 - *Make UI design an integral part of the software engineering process.*
 - *Allocate time for many iterations of prototyping and evaluation.*
- **Developers often underestimate the weaknesses of a GUI**
 - *Ensure all software engineers have training in UI development.*
 - *Always test with users.*
 - *Study the UIs of other software.*