Introduction to Software Engineering

3. User Interface Design

Mircea F. Lungu

Based on a lecture by Oscar Nierstrasz.

Roadmap

> Interface design
> Design principles
> Graphical User Interfaces (GUI)
> Usability Testing

Roadmap

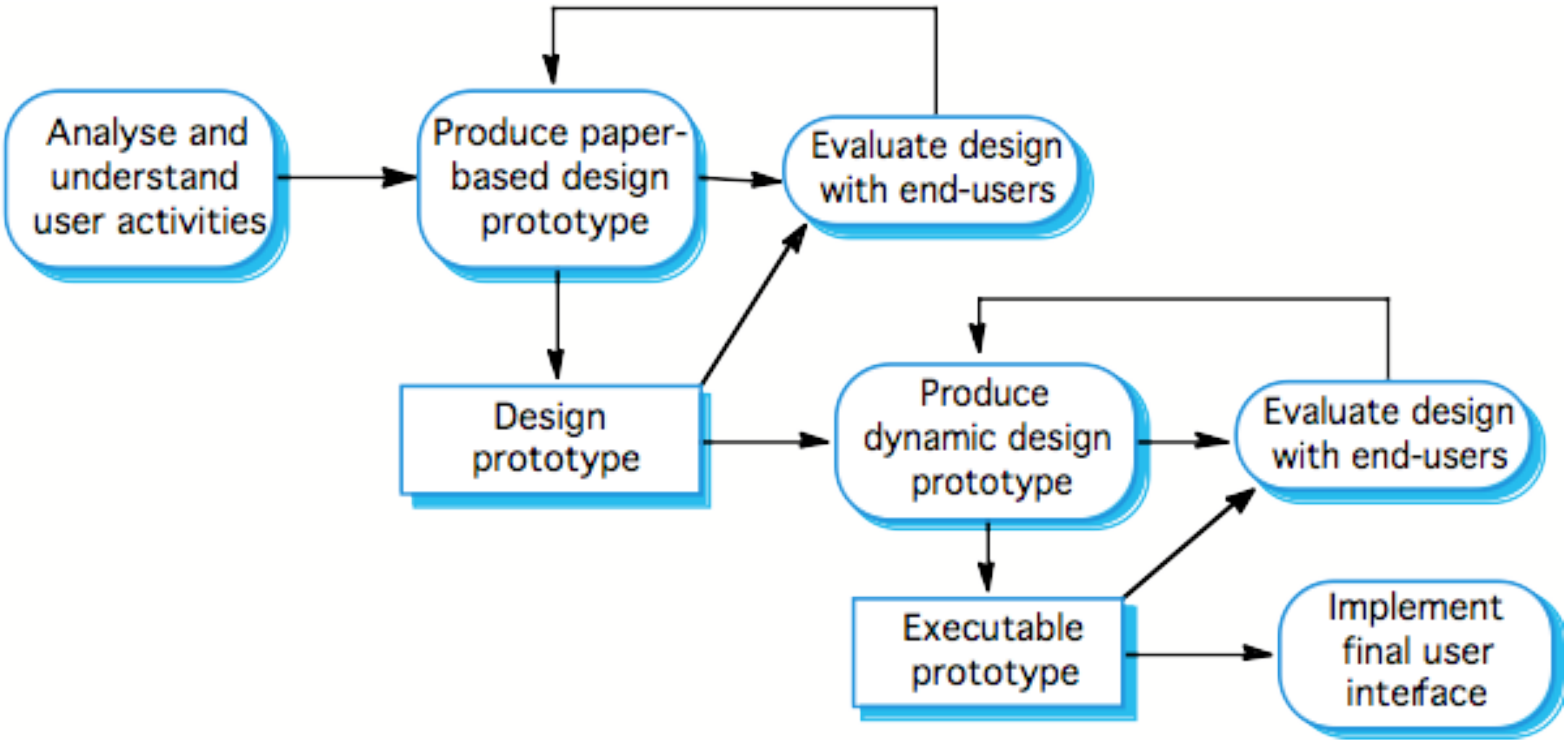
- > **Interface design**
- > Design principles
- > Graphical User Interfaces (GUI)
- > Usability Testing

The interface design process

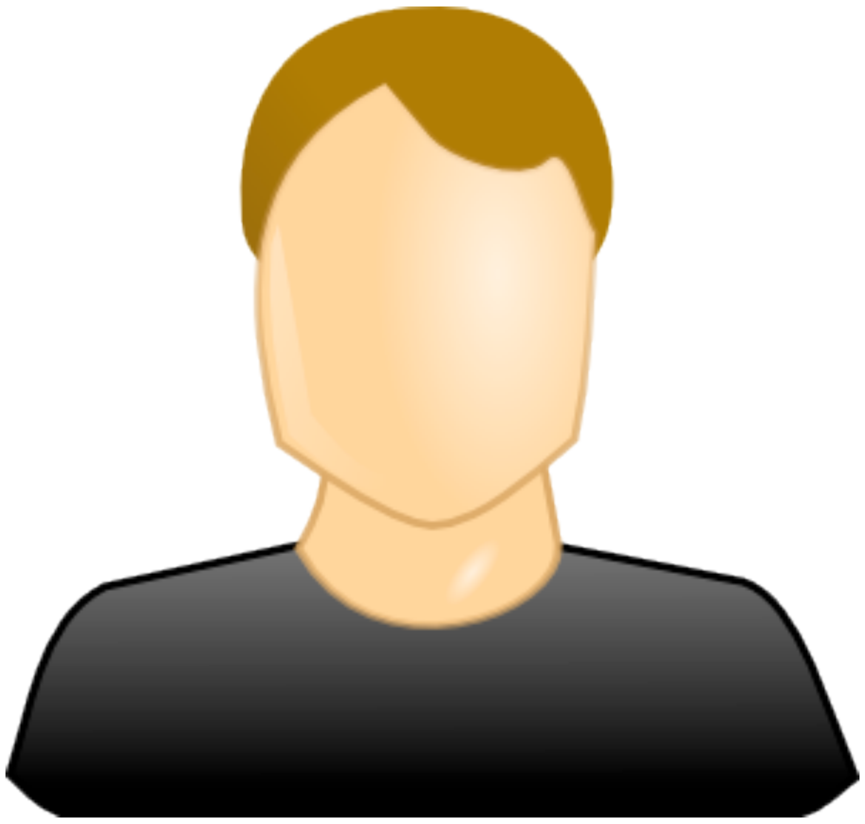
- User-Interface (UI) design is an ***iterative process*** involving close liaisons between users and designers.

- The 3 core activities in this process are:
  - *User analysis*. Understand what the users will do with the system;
  - *System prototyping*. Develop a series of prototypes for experiment;
  - *Interface evaluation*. Experiment with these prototypes with users.

The design process


Analyse and understand user activities
Produce paper-based design prototype
Evaluate design with end-users
Design prototype
Produce dynamic design prototype
Evaluate design with end-users
Executable prototype
Implement final user interface

Personas



The User

Maud

Elmer

It is sometimes better not to talk about “the user” but think about a clear customer.

Technique from Marketing
- generated after interviews with users
- helps in focusing a product’s features
- a single persona should be the main focus a design

Popularized by Alan Cooper in his book “The Inmates are Running the Asylum”

Roadmap

- Interface design models
- **Design principles**
- Graphical User Interfaces (GUI)
- Usability Testing

User Interface Design Principles

Principle	***Description***
User familiarity	Use terms and concepts *familiar* to the user.
Consistency	Comparable operations should be activated in the *same way*. Commands and menus should have the same format, etc.
Minimal surprise	If a command operates in a known way, the user should be able to *predict* the operation of comparable commands.
Feedback	Provide the user with visual and auditory feedback, maintaining *two-way communication*.

User Interface Design Principles

Principle	***Description***
Memory load	Reduce the amount of information that must be remembered between actions. *Minimize* the memory load.
Efficiency	Seek efficiency in dialogue, motion and thought. *Minimize keystrokes and mouse movements*.
Recoverability	Allow users to *recover from their errors*. Include undo facilities, confirmation of destructive actions, 'soft' deletes, etc.
User guidance	Incorporate some form of *context-sensitive user guidance* and assistance.

Roadmap

- > Interface design models
- > Design principles
- > **Graphical User Interfaces (GUI)**
- > Usability Testing

Command Interfaces

With a command language, the user types commands to give instructions to the system

- May be implemented using ***cheap terminals***
- *Easy to process* using compiler techniques
- Commands of *arbitrary complexity* can be created by command combination
- *Concise interfaces* requiring minimal typing can be created

Command Interfaces

Advantages

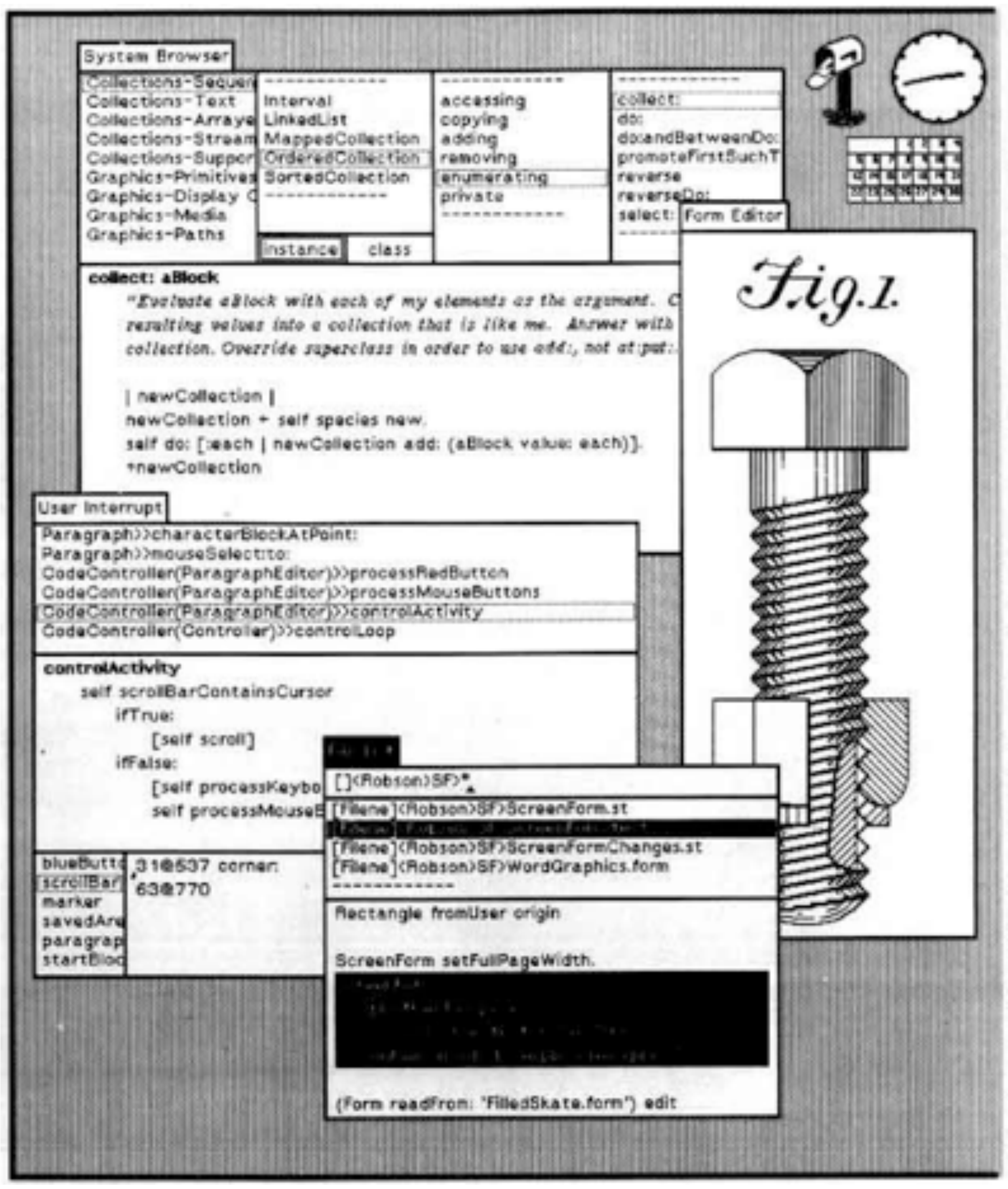
- Allow experienced users to *interact quickly* with the system
- Commands can be *scripted* (!)

Problems

- Users have to *learn and remember* a command language
- Not suitable for *occasional* or inexperienced users
- An *error detection* and recovery system is required
- *Typing ability* is required (!)

GUIs

XEROX Alto



```
collect: aBlock
    "Evaluate aBlock with each of my elements as the argument. C
    resulting values into a collection that is like me.  Answer with
    collection. Override superclass in order to use add:, not at:put:

    | newCollection |
    newCollection ← self species new.
    self do: [:each | newCollection add: (aBlock value: each)].
    ↑newCollection
```

```
Paragraph>>characterBlockAtPoint:
Paragraph>>mouseSelect:to:
CodeController(ParagraphEditor)>>processRedButton
CodeController(ParagraphEditor)>>processMouseButtons
CodeController(ParagraphEditor)>>controlActivity
CodeController(Controller)>>controlLoop
```

```
controlActivity
    self scrollBarContainsCursor
        ifTrue:
            [self scroll]
        ifFalse:
            [self processKeybo
            self processMouseB
```

```
31@537 corner:
63@770
```

```
[]<Robson>SF>*
[Filene]<Robson>SF>ScreenForm.st
[Filene]<Robson>SF>ScreenFormChanges.st
[Filene]<Robson>SF>WordGraphics.form
```

```
Rectangle fromUser origin

ScreenForm setFullPageWidth.

(Form readFrom: 'FilledSkate.form') edit
```

Smalltalk 80

XEROX Alto was the first computer to use the **desktop metaphor.** And a **mouse.**

More about the history of XEROX PARK in **Dealers of Lighting**.

GUIs

Advantages

- They are *easy to learn* and use.
  - —Users without experience can learn to use the system quickly.
- The user may *switch attention* between tasks and applications.
- *Fast, full-screen interaction* is possible with immediate access to the entire screen

Problems

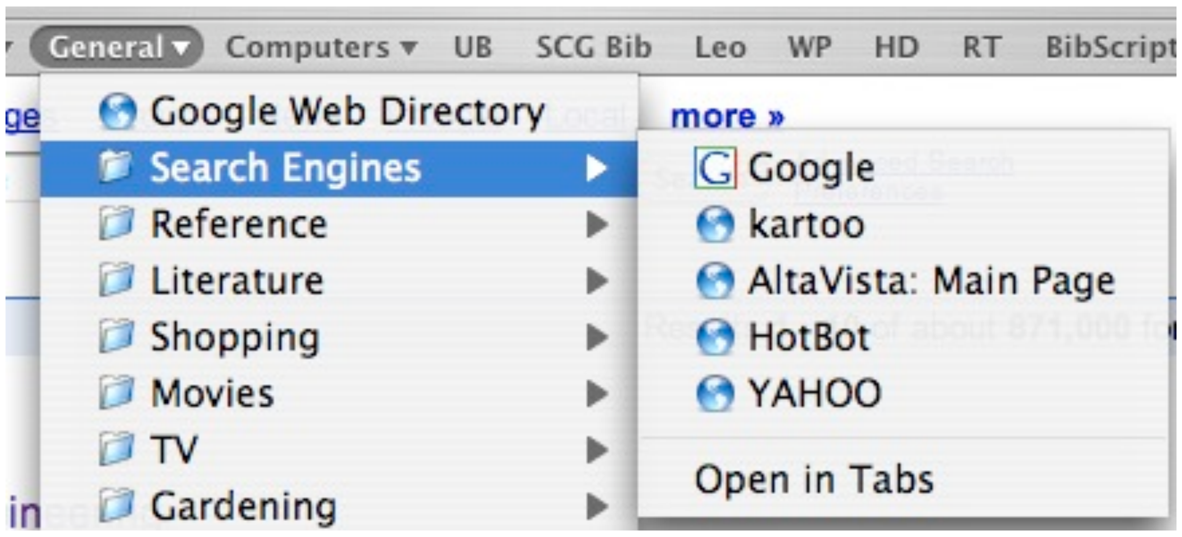
- A GUI is not automatically a good interface
  - —Many software systems are *never used* due to poor UI design
  - —A poorly designed UI can cause a user to make *catastrophic errors*

Components

Characteristic	***Description***
Windows	Multiple windows allow *different information to be displayed simultaneously* on the user's screen.
Icons	Usually icons represent *files* (including folders and applications), but they may also stand for *processes* (e.g., printer drivers).
Menus	Menus bundle and organize *commands* (eliminating the need for a command language).
Pointing	A pointing device such as a mouse is used for *command choices* from a menu or indicating items of interest in a window.
Graphics	Graphical elements can be *commands* on the same display.

Supporting Consistency and Minimal Surprise.

Menu Systems



Advantages

- Users don't need to remember command names
- Typing effort is minimal
- User errors are trapped by the interface
- Context-dependent help can be provided (based on the current menu selection)

Problems

- Actions involving *logical conjunction* (and) or disjunction (or) are *awkward* to represent
- If there are many choices, some *menu structuring* facility must be used
- *Experienced users find menus slower* than command language

Menu Structuring

Scrolling menus

> The menu can be scrolled to reveal additional choices

> Not practical if there is a very large number of choices

Hierarchical menus

> Selecting a menu item causes the menu to be replaced by a sub-menu

Walking menus

> A menu selection causes another menu to be revealed

Associated control panels

> When a menu item is selected, a control panel pops-up with further options

Colour Use Guidelines

Colour can help the user *understand complex information structures.*

> Don’t use (only) colour to *communicate meaning*!

—Open to *misinterpretation* (colour-blindness, cultural differences ...)

—*Design for monochrome then add colour*

> Use colour coding to support user tasks

—highlight exceptional events

—allow users to control colour coding

> Use *colour change* to show *status change*

> Don't use too many colours

—Avoid colour pairings which clash

> Use colour coding *consistently*

Platform Specific GUI Patterns

http://developer.android.com/design/patterns

Roadmap

> Interface design models
> Design principles
> Graphical User Interfaces (GUI)
> **Usability Testing**

Usability Testing

> Observe a group of test subjects performing a predefined scenario

- Which test subjects?
- How many test subjects?
- Which scenarios?
- What to observe?

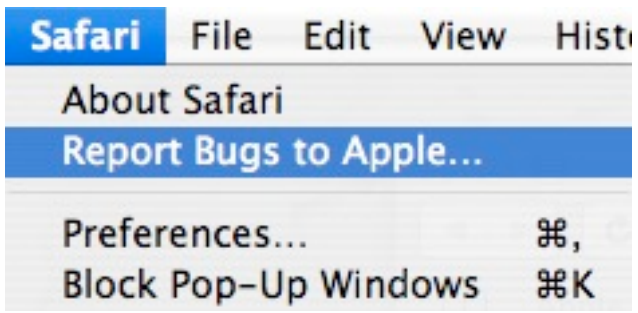
Jakob Nielsen, *Usability Engineering*

User interface evaluation

- Some evaluation of a user interface design should be carried out to assess its *usability*.
- Full scale evaluation is very *expensive* and *impractical* for most systems.
- Ideally, an interface should be evaluated against a *usability specification*. However, it is rare for such specifications to be produced.

Simple evaluation techniques

- *Questionnaires* for user feedback.
- *Video recording* of system use and subsequent tape evaluation.
- *Instrumentation* of code to collect information about facility use and user errors.
- The provision of code in the software to collect *on-line user feedback*.



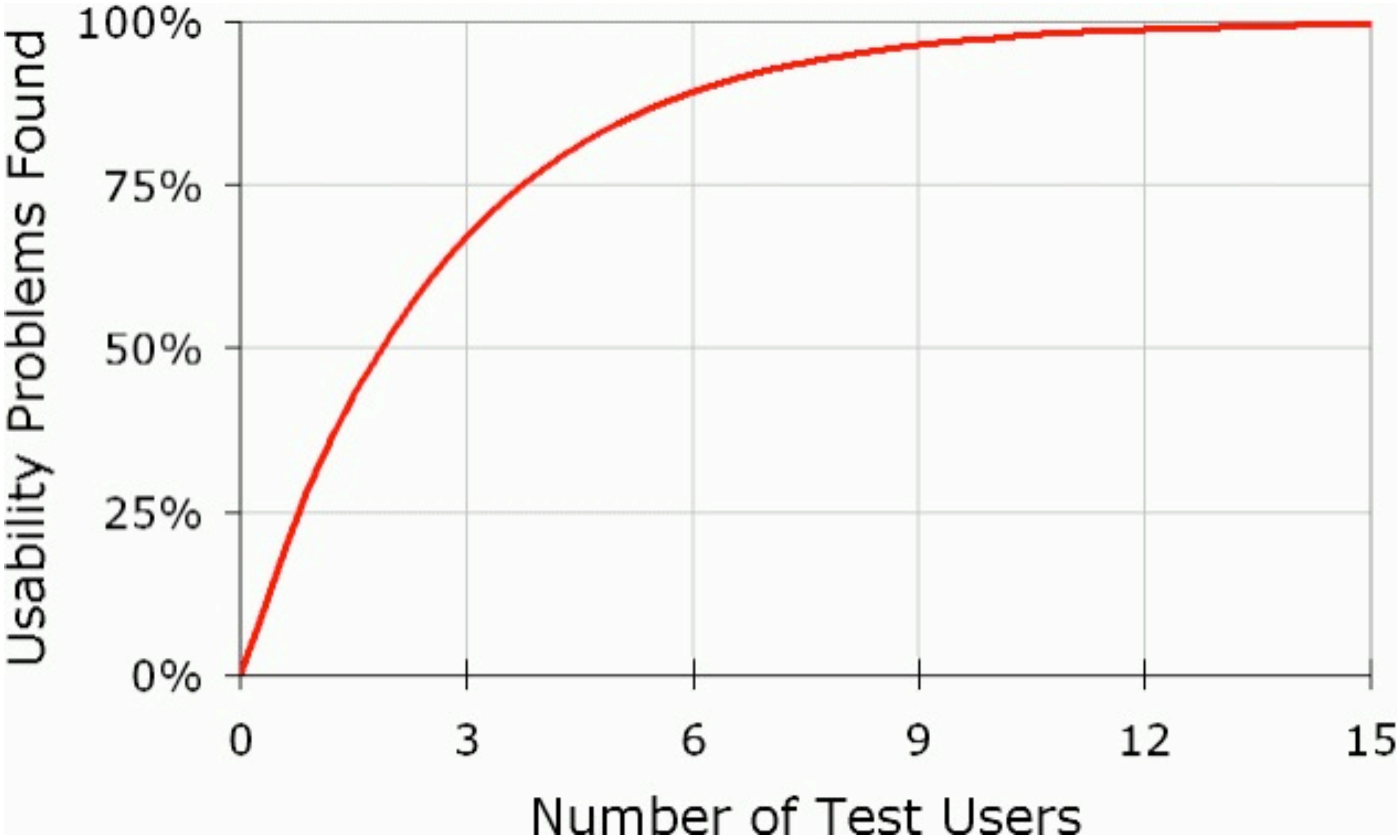
Hints

- Establish concrete goals — what do you want to achieve?
  - What criteria will you use to establish “success”?
  - What data will you collect?
  - Choose representative test tasks.
- Carry out a pilot test first.
- Test users should truly represent the intended users.
- Use experienced experimenters. (Get trained!)
  - Make the test subjects feel comfortable.
  - Don’t bias the results.

Usability Attributes

Attribute	***Description***
Learnability	How long does it take a new user to become *productive* with the system?
Speed of operation	How well does the system *response* match the user's work *practice*?
Robustness	How *tolerant* is the system of user error?
Recoverability	How good is the system at *recovering* from user errors?
Adaptability	How closely is the system tied to a *single model* of work?

Why you need to test with 5 users



http://www.useit.com/alertbox/20000319.html

Nielsen, Jakob, and Landauer, Thomas K.: "A mathematical model of the finding of usability problems," *Proceedings of ACM INTERCHI'93 Conference* (Amsterdam, The Netherlands, 24-29 April 1993), pp. 206-213.

Roadmap

> Interface design models
> Design principles
> Graphical User Interfaces (GUI)
> Usability Testing
> **Summary**

Key points

- The user interface design process involves *user analysis*, *system prototyping* and *prototype evaluation*.
- *User interface design principles* should help guide the design of user interfaces.
- *Interaction styles* include direct manipulation, menu systems form fill-in, command languages and natural language.
- *Graphical displays* should be used to present trends and approximate values. *Digital displays* when precision is required.
- *Colour* should be used *sparingly and consistently*.
- The goals of *UI evaluation* are to *obtain feedback* on how to improve the interface design and to assess if the interface meets its *usability requirements*.

What you should know!

> Interface design principles
> What are personas and why are they useful
> Trade-offs between menus and command languages
> How to use color to improve a UI
> Android UI design patterns

Can you answer the following questions?

- Why is it important to offer “keyboard shortcuts” for equivalent mouse actions?
- How would you present the current load on the system? Over time?
- What is the worst UI you every used? Which design principles did it violate?
- What’s the worst web site you’ve used recently? How would you fix it?

Literature

Sources

- *Software Engineering*, I. Sommerville, 7th Edn., 2004.
- *Software Engineering — A Practitioner's Approach*, R. Pressman, Mc-Graw Hill, 5th Edn., 2001.

Recommended reading

- Jakob Nielsen, *Usability Engineering*, Morgan Kaufmann, 1999.
- Alan Cooper, *About Face — The Essentials of User Interface Design*, Hungry Minds, 1995.
- Alan Cooper, *The Inmates are running the Asylum*, SAMS, 1999.
- Jef Raskin, *The Humane Interface*, Addison Wesley, 2000.
- Jeff Johnson, GUI Bloopers, Morgan Kaufmann, 2000.
- *The Interface Hall of Shame*, (link)

http://homepage.mac.com/bradster/iarchitect/shame.htm
http://www.frankmahler.de/mshame/