CS 696 Mobile Phone Application Development
Fall Semester, 2010
Doc 21 Design 2
Nov 16, 2009

References

User Interface Design For Programmers, Joel Spolsky,
http://www.joelonsoftware.com/uibook/fog0000000249.html

Classic Software Development Mistakes

People-Related Mistakes
Process-Related Mistakes
Product-Related Mistakes
Technology-Related Mistakes

Classic mistakes by Steven McConnell, http://www.stevemcconnell.com.nyud.net/rdenum.htm

People-Related Mistakes

1. Undermined motivation
2. Weak personnel
3. Uncontrolled problem employees
4. Heroics
5. Adding people to a late project
6. Noisy, crowded offices
7. Friction between developers and customers
8. Unrealistic expectations
9. Lack of effective project sponsorship
10. Lack of stakeholder buy-in
11. Lack of user input
12. Politics placed over substance
13. Wishful thinking

Process-Related Mistakes

14. Overly optimistic schedules
16. Insufficient risk management
17. Contractor failure Insufficient planning
18. Abandonment of planning under pressure
19. Wasted time during the fuzzy front end
20. Shortchanged upstream activities
21. Inadequate design
22. Shortchanged quality assurance
23. Insufficient management controls
24. Premature or too frequent convergence
25. Omitting necessary tasks from estimates
26. Planning to catch up later
27. Code-like-hell programming

Product-Related Mistakes

28. Requirements gold-plating
29. Feature creep
30. Developer gold-plating
31. Push me, pull me negotiation
32. Research-oriented development

Technology-Related Mistakes

33. Silver-bullet syndrome
34. Overestimated savings from new tools or methods
35. Switching tools in the middle of a project
36. Lack of automated source-code control

One Agile Development Practice

Customer ranks features in order of importance

Developer estimates development time per feature

Features implemented per iteration

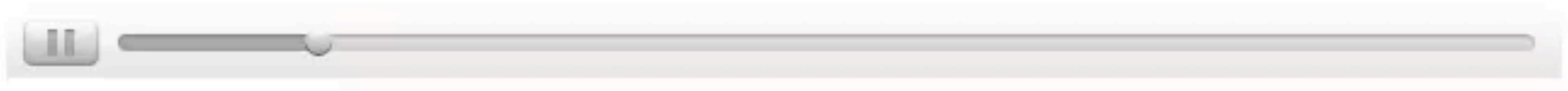
- Important features first
- Only those features that fit in to the iteration period

Another Agile Development Practice

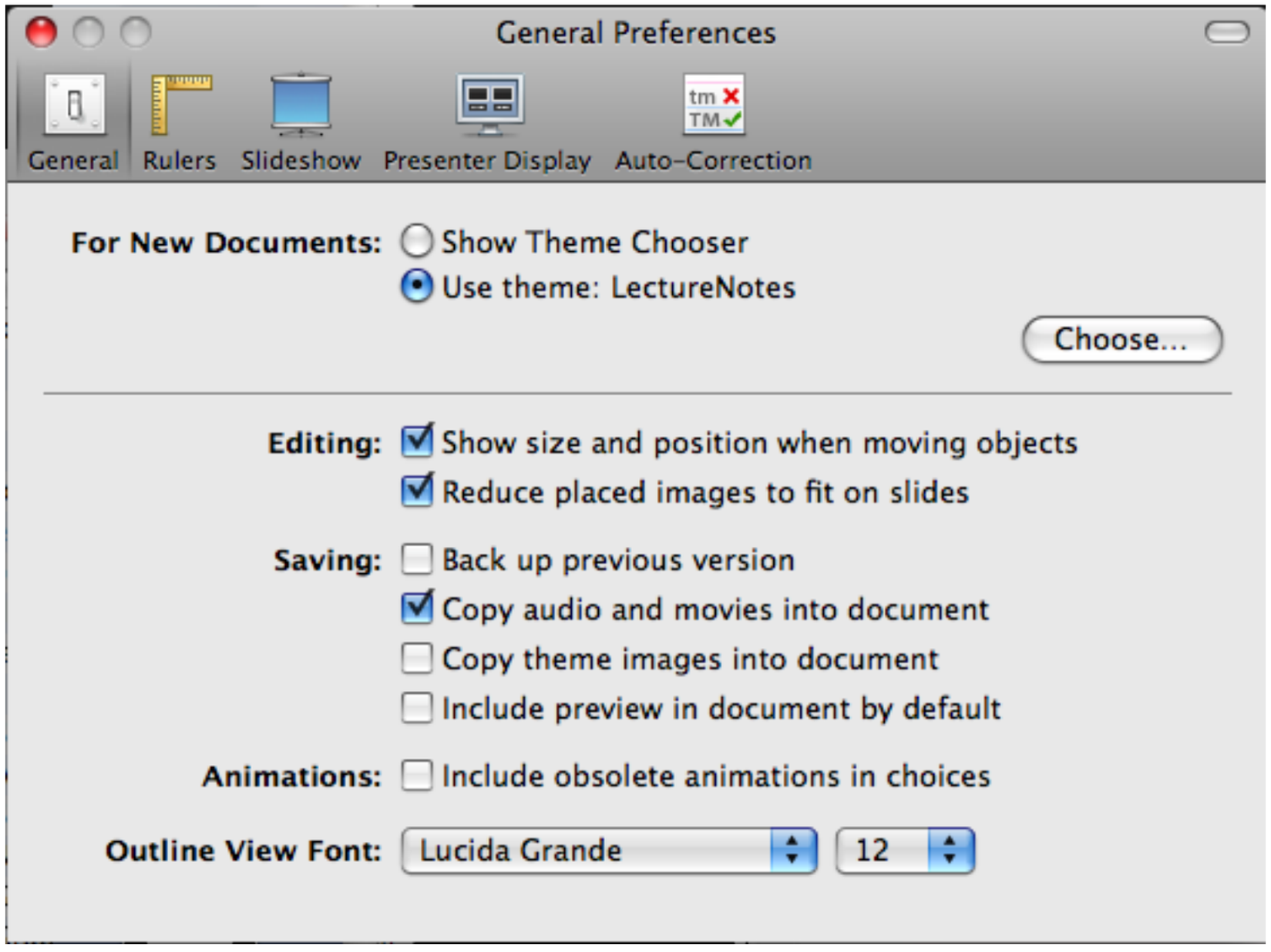
Always have a working program

User Interface Design For Programmers

A user interface is well-designed when the program behaves exactly how the user thought it would.



From http://www.apple.com/ilife/tutorials/#iphoto-intro


General Preferences
General
Rulers
Slideshow
Presenter Display
Auto-Correction
For New Documents:
Show Theme Chooser
Use theme: LectureNotes
Choose...
Editing:
Show size and position when moving objects
Reduce placed images to fit on slides
Saving:
Back up previous version
Copy audio and movies into document
Copy theme images into document
Include preview in document by default
Animations:
Include obsolete animations in choices
Outline View Font:
Lucida Grande
12

Who is your user?

Patricia is an English professor who has written several well-received books of poetry. She has been using computers for word processing since 1980, although the only two programs she ever used are Nota Bene (an ancient academic word processor) and Microsoft Word. She doesn't want to spend time learning the theory of how the computer works, and she tends to store all her documents in whatever directory they would go in if you didn't know about directories.

What does the user expect?

What is their mental model of the computer/application

Ask them

Perform usability studies

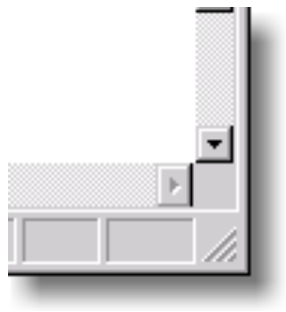
Choices

Every time you provide an option, you're asking the user to make a decision.

Image from http://www.joelonsoftware.com/uibook/fog0000000249.html

Metaphors

Affordance



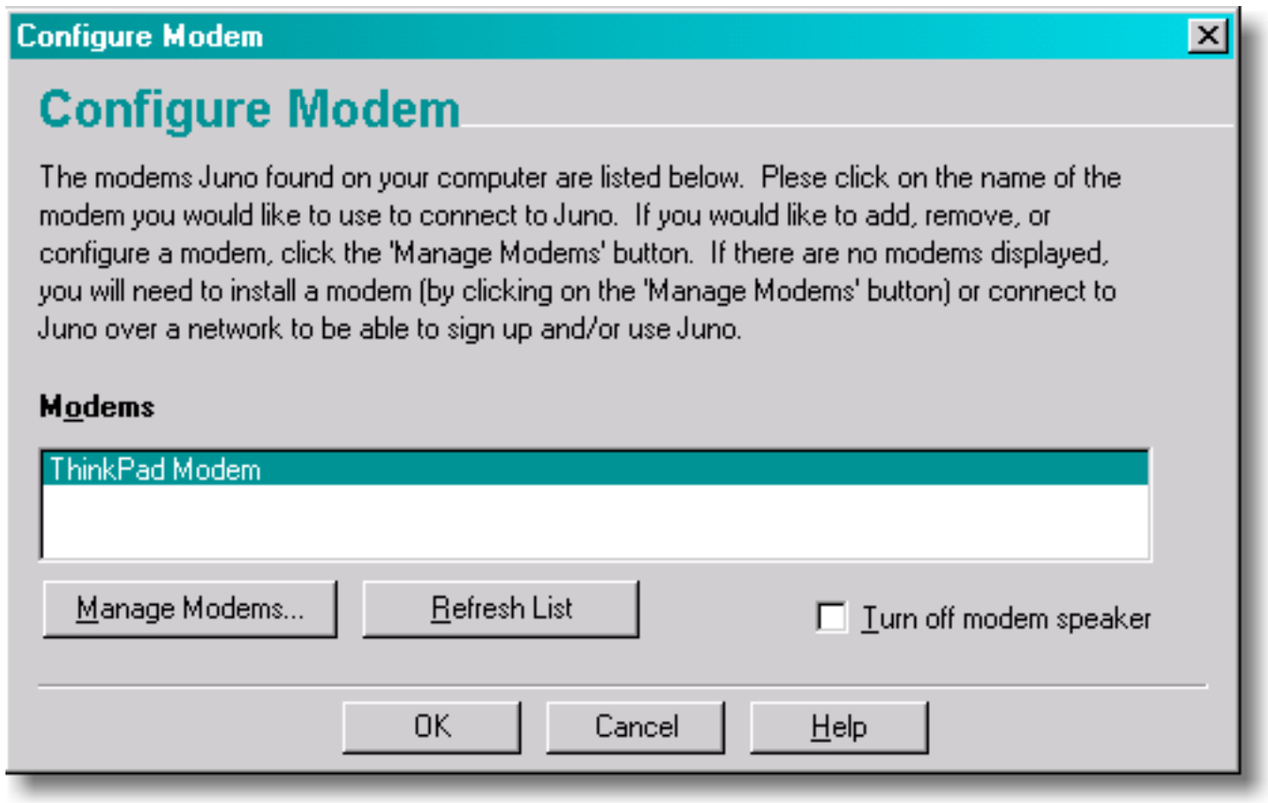
Consistency

Designing for People Who Have Better Things To Do With Their Lives

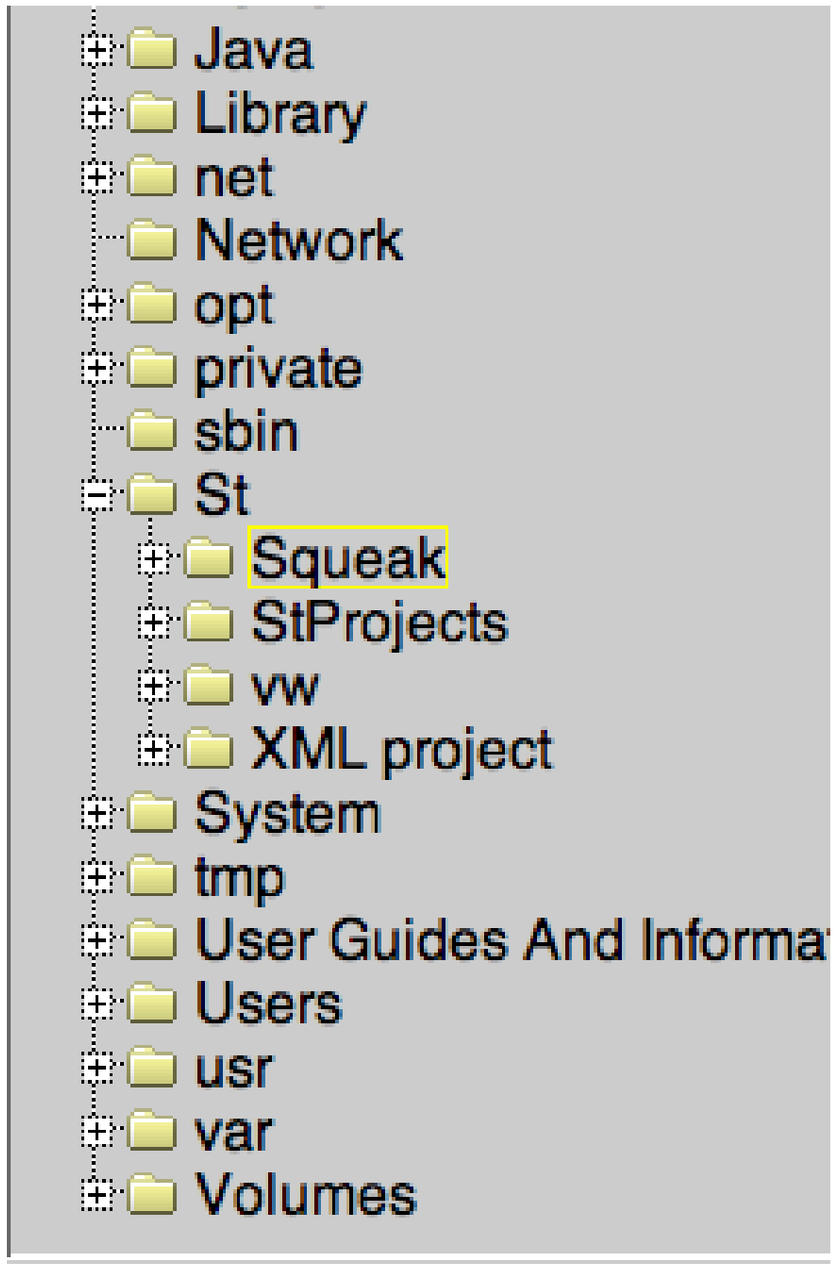
Users Don't Read the Manual

Users don't read anything

People dont Read



Users can't use the mouse



Use the Standards for you platform

It is what the users are used to

Six steps for designing good software

Invent some users

Figure out the important activities

Figure out the user model

Sketch out the first draft of the design

Iterate over your design again and again

Watch real humans trying to use your software.

Paper Prototype