iterative design and prototyping

what you might be looking for

- functionality users (or other stakeholders) want
  - participatory techniques, requirements animation
- how to present this functionality to users
  - basic design metaphor, look-and-feel of the system
  - the sequence of user-system interactions
  - how to represent the information users will see
  - how to gather input needed from users
- evaluation of specific UI features
  - icon design, terminology, data visualization
- requirements for user support
  - training required; design of help & reference info
downsides?

the many flavors of prototyping
some important dimensions

◆ representation
◆ scope
◆ executability
◆ maturation
◆ horizontal versus vertical
◆ fidelity
◆ global versus local
◆ who owns and/or builds it

choosing the right prototyping tools

◆ basic hypermedia tools (e.g., HTML, HyperCard)
  – good for requirements animation or scenario machines, especially where focus is on screen design
  – PLUS: users might be able to contribute, modify
◆ hypermedia with more extended scripting (e.g., VB, ProGraph, Director, AuthorWare)
  – more support for data processing & manipulation
  – could support evolutionary prototype for some apps
◆ interpreted languages (e.g., Smalltalk, tkl, Python)
  – suitable for evolutionary prototypes (but must struggle with rationalization of the emergent software design)
specific features to look for

◆ screen creation and editing (including link logic)
◆ the user interaction styles that are supported
◆ range of I/O devices
◆ import from or export to other tools
◆ ease of learning and use
  – version control, design rationale capture, print-outs
◆ who makes it!

surprise, surprise ... you can evaluate usefulness and usability of prototyping tools just like any other interactive system (cf. H&H Chapter 11)

building your project prototype
—what kind of prototype are you being asked to construct?—

◆ five fully functioning scenarios
  – e.g., adding UI details to what you turned in at design
◆ no need to have complete data, logic programmed
  – can “fake” things like authorization, network retrieval, database look-up, etc.
  – BUT must support user input, give feedback, etc.
◆ sufficiency test: can it support usability testing?
  – i.e., consider error blocking, feedback during use
◆ (note: deliverable includes usability specifications, the topic of next week’s lecture)
demoing your prototype

* decide soon (now!) on platform, let Vlad know
  - default will be to use PC/LiveBoard in 104c
* assume you have 1/2 of a class period (30-35 min)
* go through all five scenarios
  - include discussion of user confusions/errors you’ve considered, i.e., not just a simple chauffeured demo
* finish with short discussion of highlights of the current design and of the prototyping process
  - what’s most interesting, did you learn or change anything as you went?
* everyone on the team participates!