Balancing the Needs of Expert and Novice Users

Daniel Y. Kimberg
University of Pennsylvania

cfn.upenn.edu

Multiple lightweight interfaces

For many (but not all) programs, different kinds of interfaces make sense for different kinds of users, even for the same application:

- command line interfaces
- interactive text interfaces
- config file interfaces
- graphical user interfaces

Are tradeoffs necessary?

- If you're only going to develop one interface:
  - there's a common core of basic things both kinds of users generally need exposed
  - you don't have to make advanced functions *inaccessible* to support novices
  - you don't have to make advanced functions *prominent* to support experts

- Supporting complex functionality means exposing stuff that most users will never use.
- But why limit yourself to just one interface?

New interfaces galore

- Any program that either takes a config file or exposes an elaborate command line interface can be given multiple additional interfaces trivially.
  - UIs written in un-cool languages
  - UIs that retrieve and organize the data
  - UIs that bridge the gap between metadata systems and metadata-naive processing tools
  - UIs that take care of a variable amount of the prep work and that know about local usage
  - shell scripts and aliases

Code Level

- For simple, modular tools, we can separate interfaces and guts
  - keep the guts in a nicely encapsulated library
  - develop as many simple interfaces as we like
  - as long as there's a detailed text interface, anyone can develop a custom interface
- It's helpful in transitioning users to have GUIs produce the same config files (or display the command lines) that are consumed by the final program in the chain, even if they don't do the work that way.