**Chapter 30: Programming Tools**

* Modern programming tools decrease the amount of time required for construction
* Using leading edge tools can increase productivity by 50% or more

**Design Tools**

* A good and easy to use design and drawing tool is huge

**Source Code Tools**

* IDEs and desired features
  + Compilation and error detection from within the editor
  + Integration with source-code control
    - Build, test and debugging tools
  + Compressed or outline views of programs
    - Class names only
    - Logical structures without contents
  + Jump to definitions of classes, routines and variables
  + Jump to all places where a class, routine, or variable is used
  + Language specific formatting
  + Interactive help for the language being edited
  + Templates for common language constructs
  + Smart indenting
  + Automated code transformations or refactoring
  + List of search strings so common strings don’t need to be retyped
  + Search and replace across a group of files
  + Editing multiple files simultaneously
  + Side by side diff comparisons
* Class Hierarchy Generators
  + These are useful for breaking out inheritance trees
* Data Dictionaries
  + Database that describes all the significant data in the project
  + Actual data
  + Class definitions

**Building Your Own Programming Tools**

* Good idea, also fun
* Project Specific Tools
  + Tools to generate special kinds of test data
  + Tools to verify quality of data
  + Tools to emulate hardware
  + Part of planning for a project should be thinking about the tools that might be needed and allocating time for building them
    - Like that ridiculous tool box that the forklift picked up in KY
* Scripts
  + Tool that automates a repetitive chore

**Tool Fantasyland**

* We will always need people who can bridge the gap between real world problems and the computer that is supposed to solve the problem