Graphical Designer - Reverse Engineering

- **Task:** Graphical designer set of tools, helps understand already written code in a graphical language which you can pick. We suggest UML since it will probably become the standard language. It can also generate detailed class documentation for you.
- **Doc:** http://www.fnal.gov/cd/sweng/avail_tools.html
- **Setup on cdfsga:** setup gdesigner development
- **Examples:** There is an already reverse engineered example in ~sexton/RevEng/Trybos-V03-04-20
- **Usage:** To use on your own software, you need to: 1) specify the *.cc files; 2) specify include directories $GCC_DIR/lib/g++-include, $PROJECT_DIR/include; 3) and -D directives eg. CDF then click continue
- **Output:** Doc can be saved as text or html, graphics as Postscript.
- **Limitations:** Reverse Engineering will only give you class and implementation diagrams, NOT all the most interesting views/diagrams

OOBR from Xemacs

- **Task:** Object Oriented Browser, eases viewing of code (especially others')
- **Doc:** http://www.xemacs.org/
- **Setup on cdfsga:** setup xemacs old
- **Configuration files:** same as xemacs (.emacs, .xemacs-options, .tpu-xemacs-keys)
- **Usage:** 1) Bring up OO-Browser from tools menu ; 2) Load/Create OO-Browser; 3) try using $PROJECT_DIR/results/OOBR in order to browse cdf software. If you want to make your own database just hit return instead of giving a file name and answer lots of questions:
  1) Language? ; 2) complains but just hit <cr>; 3) List of systems and libraries <cr> terminates the lists; 4) specify output file (y); 5) Background (n) ; h=help, f=expand class, v=view source file, e= edit source file
- **Limitations:** You get what you pay for and this is free. FNAL would have to pay money for support.

Insure++ by Parasoft

- **Task:** Checks for memory leaks, array reference out-of-bounds, and other memory-reference related problems. Static tests done by reading source alone. Insure++ can work with C or C++, and does work with GNU compilers.
- **Doc:** http://www.parasoft.com/insure/index.shtml
- **Setup on cdfsga:** setup insure
- **Examples:** $INSURE_DIR/examples
- **Usage:** To use on your own software, you need a .psrc file (see examples area for contents). Then, compile your source using "insight" in place of your compiler. Remember to turn off all optimization. Static analysis is done on all code during compilation. Dynamic analysis and testing is done during run-time, but only for code that is ***actually executed***. TCA is associated tool which determines fraction of code actually tested by line, method, class, file, etc.
- **Output:** Can be sent to a text file, or to a convenient GUI tool (insra) which enables interactive sorting and filtering of reported errors.
- **Limitations:** Static analysis can be confused by Standard C++.

Kai C++ by Kuk & Assoc.

- **Product:** Traditional or Standard C++ compiler. Implements nearly all of Dec. 1996 draft C++ standard. Standard library is more complete and robust than libg++ (which leaks memory in strstreams, lacks some stdlib classes, ...). It can be used, if for nothing else, to recheck errors reported by GNU g++.
- **Platforms:** SGI (but not IRIX6-n32 yet), Linux, AIX, Dec Unix, ...
- **Doc:** http://www.kai.com/C_plus_plus
- **Setup on cdfsga:** setup kai
- **Usage:** http://www.kai.com/C_plus_plus/v3.2/doc/UserGuide/man_page/index.html
- **Mixing codes:** Use a preprocessor macro __STANDARD_CPLUSPLUS to isolate standards-conforming code from traditional (GNU C++) code. Kai C++ implement namespaces correctly, for instance, so most all of stdlib is in "std::" namespace. GNU's g++ and libg++ do not fully implement namespaces.
- **Limitations:** Kai C++ is not yet supported by all third-party products being considered for FNAL Run 2 use: Objectivity, Insure++. Also, unless adapted as primary compiler, Kai C++ may simply complicate source code support.