

Philosophy and Plans

Introduction

- Philosophy behind our code management system
- Constraints on the system
- Choices that we've made
 - Details
- Some operational details
- How to improve it

Philosophy and Plans

Philosophy behind our code management system

- Use tools that exist
- Improve the existing tools as needed
- Write new ones only when necessary

Upside

- Could get started ASAP
- Saved development time/work

Downside

- Had to make compromises
- Had to rewrite some tools after we'd started using them
- Maintenance is, perhaps, harder

Philosophy and Plans

Constraints

- Tight version control, history
- C++
 - Severe coupling requires “monolithic” builds
 - “non-standard” compilers
- Multi-platform support
- Multiple simultaneous versions
- Multiple widely dispersed developers
 - most/many are non-expert
- Remote development sites
 - Remote distribution by non-experts

Philosophy and Plans

Choices

- CVS
 - free, widely used
 - good version control
 - allows/facilitates multiple remote developers
- SoftRelTools build system (Babar)
 - free
 - natural use of CVS
 - naturally allows good version control
 - naturally allows monolithic builds
 - multi-platform support
 - multi-compiler support
- UPS/D, Fermilab's Unix Product Support/Distribution
 - free
 - allows multiple versions to co-exist on a single machine
 - has good remote distribution capabilities

Philosophy and Plans

Choices (details)

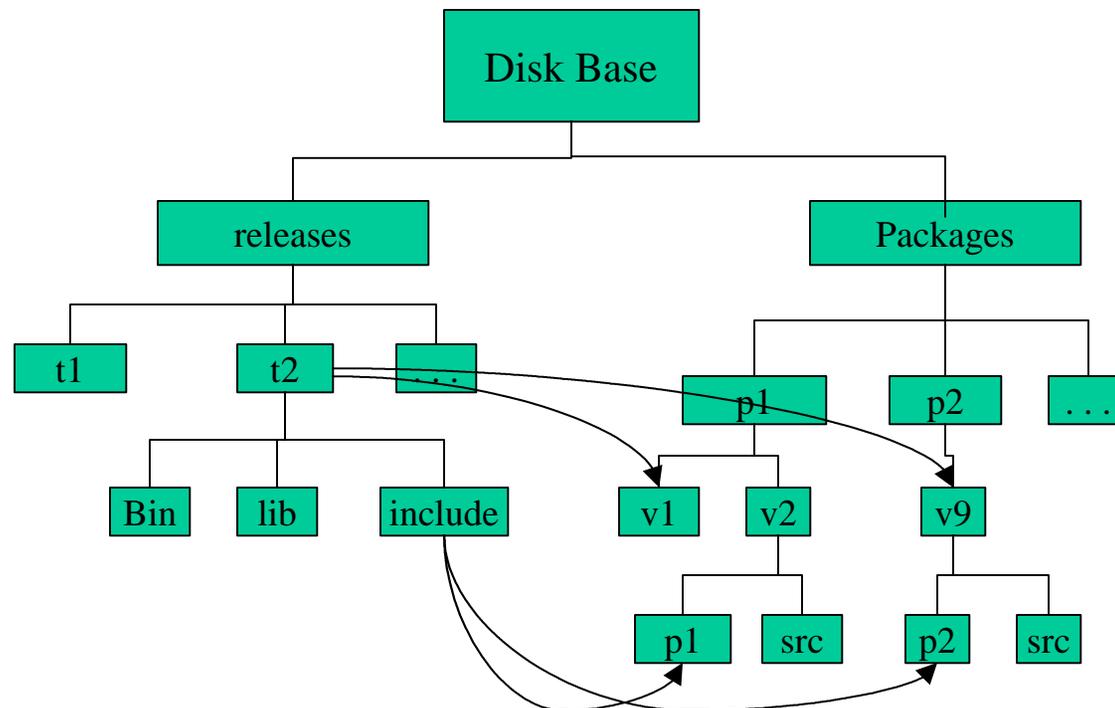
– CVS

- <http://www.loria.fr/~molli/cvs-index.html>
especially the reference manual at:
http://www.loria.fr/~molli/cvs/doc/cvs_toc.html
- We “release” only *tagged* (cvs rtag) versions.
This allows remote developers to share code
using cvs (cvs checkout and update from the
trunk’s “head)

Philosophy and Plans

Choices

- SoftRelTools build system
- <http://runIIcomputing.fnal.gov/runiweb/cmgt.html>
- Basic structure is:



Philosophy and Plans

Choices (details)

- SoftRelTools
 - Build system is based on **gmake**
 - Collection of makefile fragments
 - supply the make targets and rules
 - Packages contain
 - GNUmakefile in each directory
 - define make variables that direct that specify what needs to be done in each directory and if there are subdirectories that need to be processed.
- Ctest/Ctbuild
 - <http://www-d0.fnal.gov/software/cmgt/ctest/ctest.html>
a bit out of date, but...
 - an interface to SoftRelTools that replaces the GNUmakefiles (except the top level one) with straight text files to direct the build.

Philosophy and Plans

Choices (details)

- UPS/D, Fermilab's Unix Product Support/Distribution
 - <http://www.fnal.gov/docs/products/ups/>
 - UPS (Unix Product Support)
 - users see as “setup <product>”
 - Allows multiple versions of a single “system” product to be available simultaneously
 - totally rewritten by FNAL Computer Division
 - less intrusive, can be installed and used totally within a single non-privileged user's resources.
 - Can be installed in a shared mode either privileged or non-privileged.
 - » Privilege allows “#!/usr/local/bin xxx”
- UPD (Unix Product Distribution)
 - used is *pull* new versions of products from a distribution node and declare them to the local ups database:
 - fnkits.fnal.gov
 - www-d0.fnal.gov
 - any others

Philosophy and Plans

Choices (details)

- UPD (Cont)
 - Can be used to pull only the structure of a given release, or the whole thing
 - `upd install -h www-d0.fnal.gov`
`DORunII <ver> -q dist`
all packages needed
the links etc
ready to build
 - `upd install -h www-d0.fnal.gov`
`DORunII-bin <ver> -q $SRT_SUBDIR`
everything from the previous one
plus binaries corresponding to \$SRT_SUBDIR

Philosophy and Plans

Choices (details)

- UPC (Unix Product Census)
 - Allows a remote site to compare their UPS database with the one(s) at the distribution sites.
- UPP (Unix Product Pull)
 - Allows automated Census (ala UPC) with mail various forms of notification.
 - Allows automated installation (UPD) if you've got the fortitude.
- We have never tested either of these for lack of time.

Philosophy and Plans

Some Operational Details

These remarks are in response to an email question from Kors Bos which seems to reflect a lot of confusion from users, especially the remote and new users.

- A release
 - Has a name, ie) t00.84.00, preco03.07.00
we will be able to identify particular data sets by the name of the release that produced them.
 - A release includes everything needed to do development and to build custom executables.
 - A collection of tagged (cvs rtag) package versions. These are specific instances of each of the D0 packages.
 - Instructions (via ups table file) to “setup” specific versions of external products needed during a build.
 - The binaries, (bin/, lib/ etc) needed by the build system to build your executables
 - The executables built by the build system except for those used only for regression testing.
 - The rcp files and other data files needed to run pre-built executables or the ones you build.

Philosophy and Plans

Some Operational Details

- A “golden” release
 - Originally was meant to be a release that built and tested without errors.
 - Due to the difficulty of producing these, it now means (for the “t” series) a release with a d0reco executable. Soon this will be tightened to mean a d0reco that runs.
 - These are tar’d up for distribution via UPD from www-d0.fnal.gov

Philosophy and Plans

Some Operational Details

– Release Sequences

- “**t**” releases are the normal “test” releases done roughly one/week on IRIX, Linux and OSF. *Most development will be done against these.*
- “**nt**” releases contain a subset of the packages in the corresponding “**t**” release. They are done only on the NT operating system. *Development for NT (Level 3 etc) will be done against these.* They also catch errors that are missed by the other compilers.
- “**production**” releases are releases meant for a particular production environment. They are meant to satisfy the need of the experiment to be able to trace **exactly** what code was used at each step of the data taking, reconstruction, analysis chain. They are based on a particular “**t**” release but the code goes through much more verification. *Since verification takes a while, these are usually too old to use for code development.* Currently we have “pmc” (monte carlo), “preco” (reconstruction) and will have “l3” (level3) plus others.

Philosophy and Plans

Some Operational Details

- Release Sequence Disk Residency
 - “**t**” and “**nt**” releases will typically stay on disk for 3-4 weeks. They are removed when we need the disk space.
 - “**production**” releases will stay on disk for *much* longer and will be available in an archived form “forever”. Though they are extremely stable, they can’t change, *they are a poor choice to use for code development.*

Philosophy and Plans

Some Operational Details

- Which releases to use.
 - Code development
 - “t” or “nt” releases
 - Looking at some muon tracks.
 - Depends on what you are doing:
 - » to develop methods: “t” or “nt”
 - » to study the results produced by a “preco” release: “preco”, the one that produced the data probably
 - Reconstruct all events and look at efficiencies
 - Probably “t” or “nt”
 - BUT if you want the efficiency for a given data set, you need the preco that was used to create the data set.
 - Reconstruct all events just simulated
 - depends
 - » if part of a “challenge”, both the pmc release and the preco release that needs to be used will be specified. Accountability the key. We need a well defined, single valued efficiency.

Philosophy and Plans

Some Operational Details

- Which releases to use (cont)
 - Run the latest D0 Monte Carlo
 - “t” release
 - BUT if part of a challenge or to measure the efficiency of a real data set, the “pmc” release will have to be agreed upon by the collaboration.

Philosophy and Plans

How to improve it

- We all know the system isn't perfect
 - Ask three people and they'll give you three different answers as to how it isn't perfect. If any two of those don't conflict on the majority of points I'd be surprised.
- Don't go off and invent something new!
 - We don't have time to do major build/distribution system development until after we have gotten the experiment up and get the first data out.
- Work with what we have
 - incremental improvements are always welcome
 - fixes/correction are actively sought.
- Be patient
 - Most of the components are maintained by others (non-D0) who have to satisfy more than just D0. It takes time to get all the approvals and to do the testing needed.

Philosophy and Plans

How to improve it

- Be patient (cont)
 - BUT if you don't hear a response in a reasonable time, remind us, and keep doing it. You are entitled to a yes/no response at least, and a reason.
- Don't be too clever at getting around perceived deficiencies in the system.
 - If there is a real problem/error it needs to be fixed.
 - If there is functionality that needs to be added, that needs to be done too.
 - Most of the time there is a “legal” way to do what you need to do. You just don't know it.

Too often workarounds are done in ways that make the system more fragile. Those come back to bite us again and again. We are trying very hard to make the system robust. Cute/clever workarounds don't help.

Philosophy and Plans

How to improve it

- Who to contact
 - For errors/corrections
 - d0-release-mgr@fnal.gov (Paul and I)
 - For questions as to what's available, possible
 - d0-release-mgr@fnal.gov (Paul) for SoftRelTools and the build system
 - jonckheere@fnal.gov (me) for the rest
 - For suggestions for added functionality
 - jonckheere@fnal.gov to see if it might already exist and is possible followed by
 - d0rug@fnal.gov to see if there is any support or need for it.
- Real code!
 - If you can show that a changed is “easy” by supplying real code, that would be best
- Documentation!
 - Corrected/new documentation is always needed.
 - Best to send us the **entire** corrected text to just replace the old version unless it's an extremely simple correction.

Philosophy and Plans

How to improve it

- Bottom line
 - Ask before you do a lot of work! We've seen too much good work thrown away because it couldn't be fit into the whole.
 - People willing to work within the system improving it are welcome but:
 - The improvements must be coordinated. That's part of my job.
 - The improvements must be agreed upon by the collaboration which in this case means D0Rug.
 - People who want to rewrite it should come back in a couple of years.