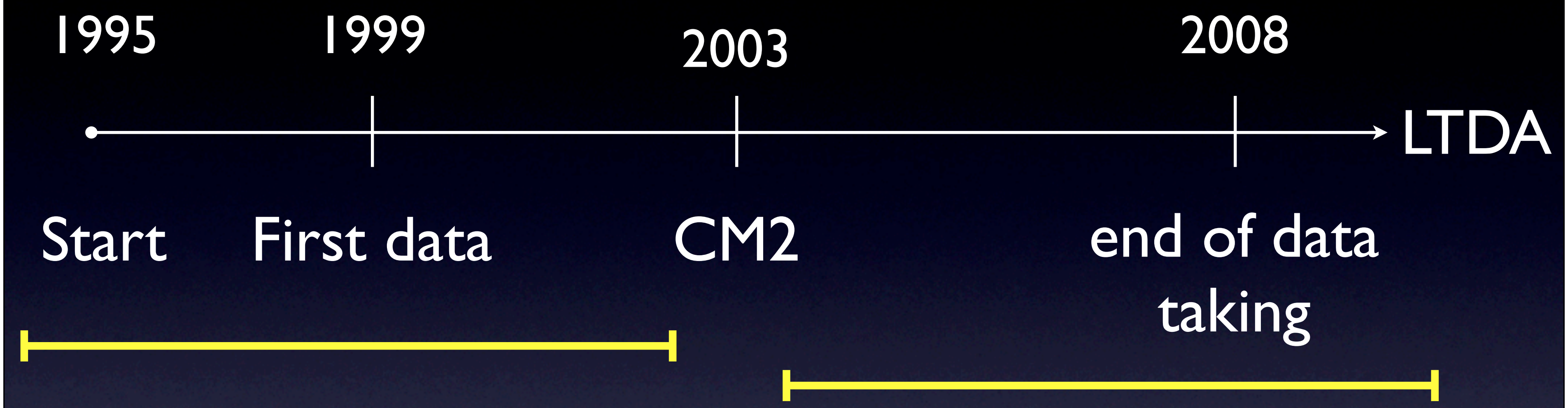
Reuse of BaBar Computing for SuperB

Personal perspective of

David Brown, LBNL

SuperB Computing Workshop
9 March 2010

BaBar Computing


1995 1999 2003 2008 LTDA

Start First data CM2 end of data taking

S2B era (R. Jacobsen)

- C++
- UNIX
- Objectivity
- OO
- Web

Precision era (GPDF)

- Root
- LINUX
- Distributed computing
- MYSQL

BaBar Computing Paradigms

- Online-Offline separation
- Multi-Site distributed computing
- User=Developer
- SRT
- Skimming
- Release cycles (production cycles)
- Sequential processing model

Online-Offline Separation

- Code separation
  - separate build, release systems
  - separate coding standards and requirements
    - Embedded processor compatibility
    - realtime reliability
    - performance
- Separate personnel
- TC + Database as communication channels
  - required a set of common code
- OEP, Prompt Calib, L3 also tied these together

Distributed Computing

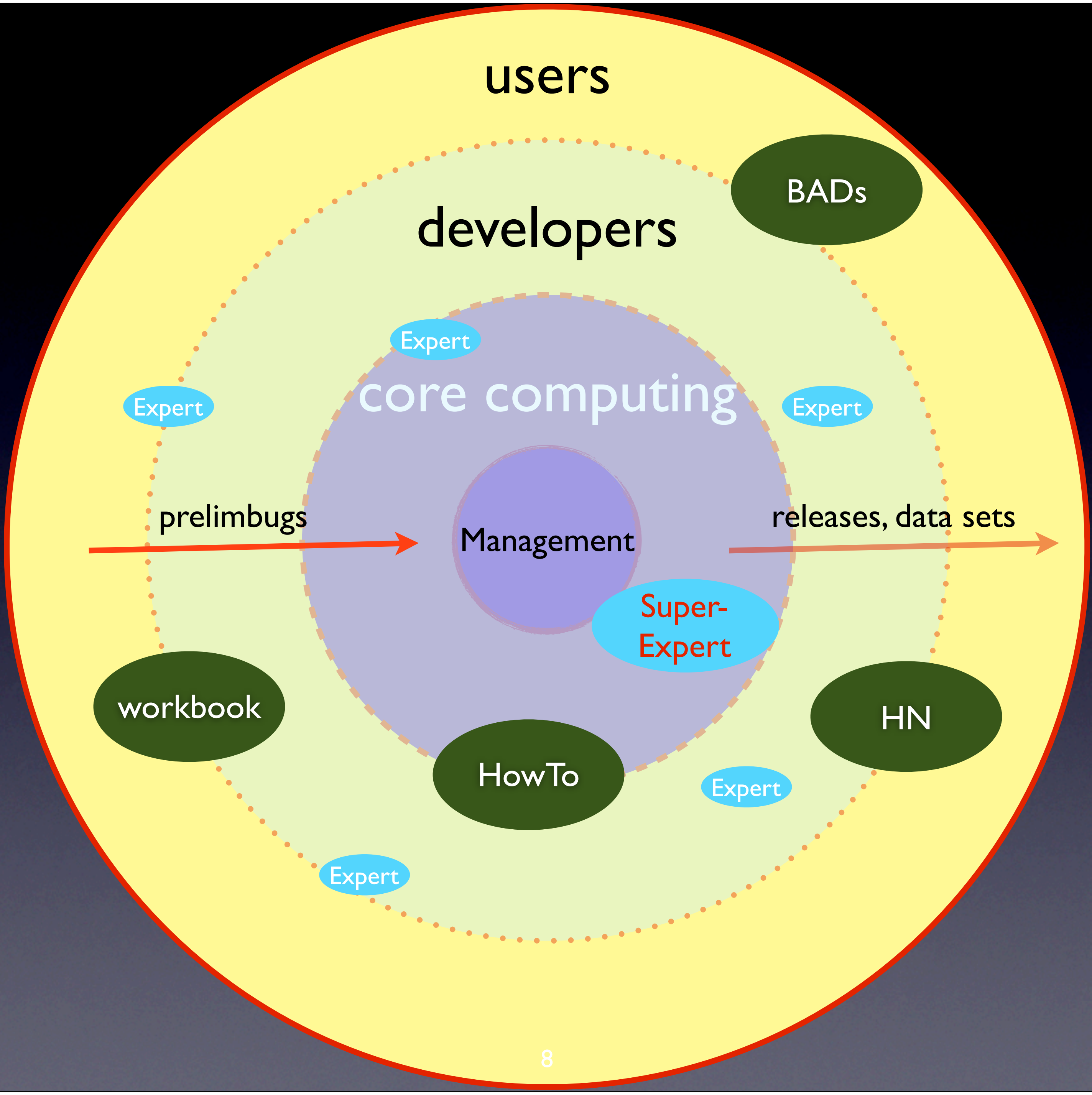
- Heterogenous set of processing sites
    - local installation + maintenance of BaBar Env. by onsite experts
- Network file transfer between sites
- XROOTD within sites
- Web (HN) communication
- Remote code development
- SLAC as master (repository) site
    - SPOF for many processes

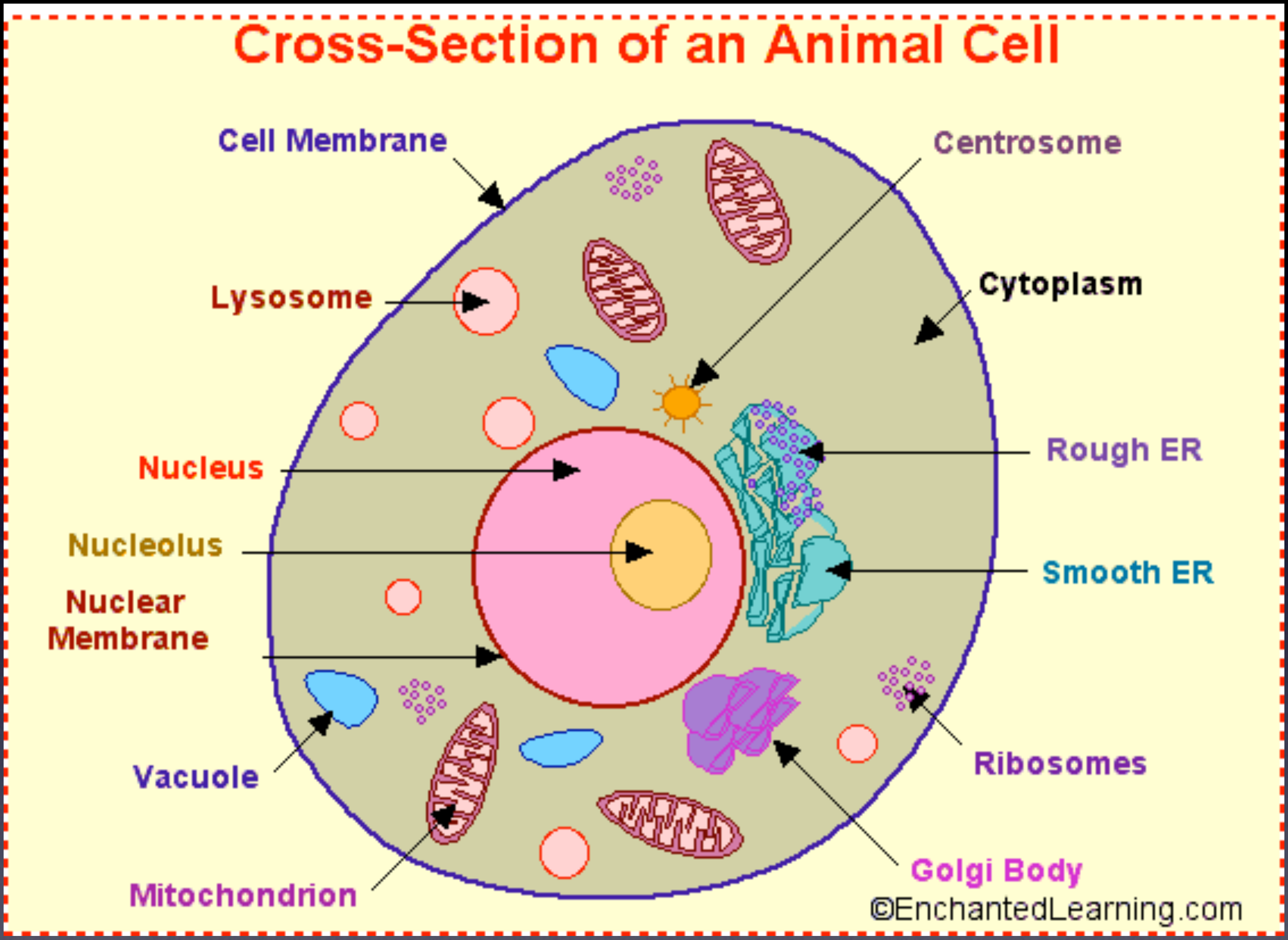
Release System Structure

- Release defined outside revision control
  - web page forms, tags files, ...
  - No formula to create a release from repository
- Lettered build ~ release patch
- Release cycle=parallel development threads
  - associated with specific data samples
  - Very Long analysis through times (up to 4 years)
- No Branches, subdirectories
  - source conflicts, development complications
- Nightly Release
  - based on hand-made tags, not trunk

User = Developer

- Open rw access to source code
  - restrictions applied only during release
- Formal management roles
- Informal (subsystem) development roles
- Large pool of motivated, talented participants
- Many unclear aspects
  - who provides documentation?
  - who responds to bug reports?
    - PC? super-expert? management?
  - what training is required?
  - what are the standards? How are they enforced?
- Babar standard: does it work?

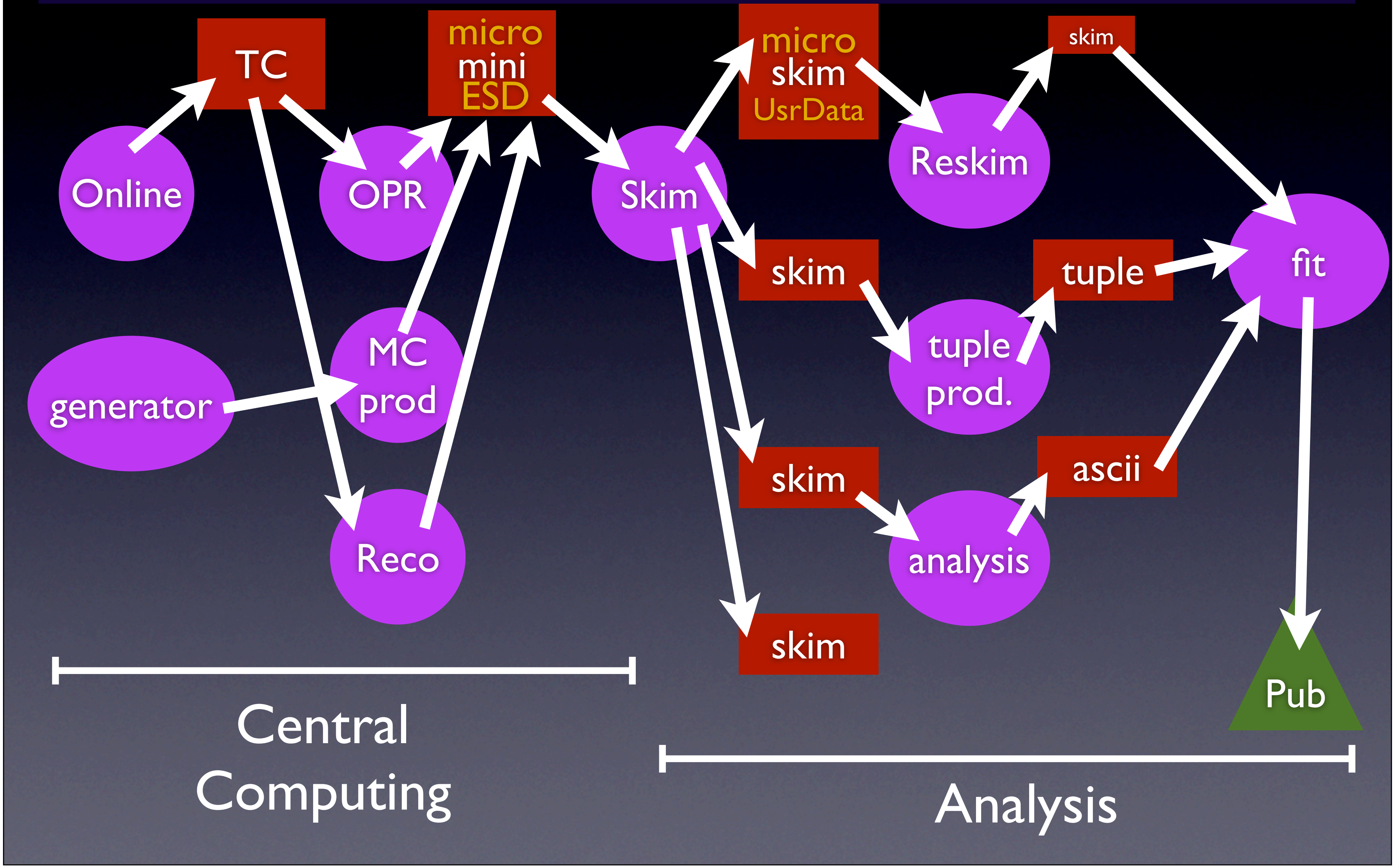

users
BADs
developers
Expert
core computing
Expert
Expert
prelimbugs
releases, data sets
Management
Super-
Expert
workbook
HN
HowTo
Expert
Expert



Cross-Section of an Animal Cell

©EnchantedLearning.com

BaBar Processing Model



What of BaBar computing should SuperB keep?

- Hardware?
- Online-offline separation?
- distributed computing?
- release/repository system?
- processing model?
- Organization?
- code base?

SuperB must process ~100X BaBar data using ~50% BaBar manpower

SuperB Comp. Organization

- Must use physicist manpower
  - cannot afford a pure-professional comp. org.
  - Physicists provide unique qualities
- Must allow for migration of experts
- Must improve reliability
- Must increase autonomy
- Reduce reliance on 'super-experts'
- Centralize 'private' computing
  - tuple production, AWG storage, ...

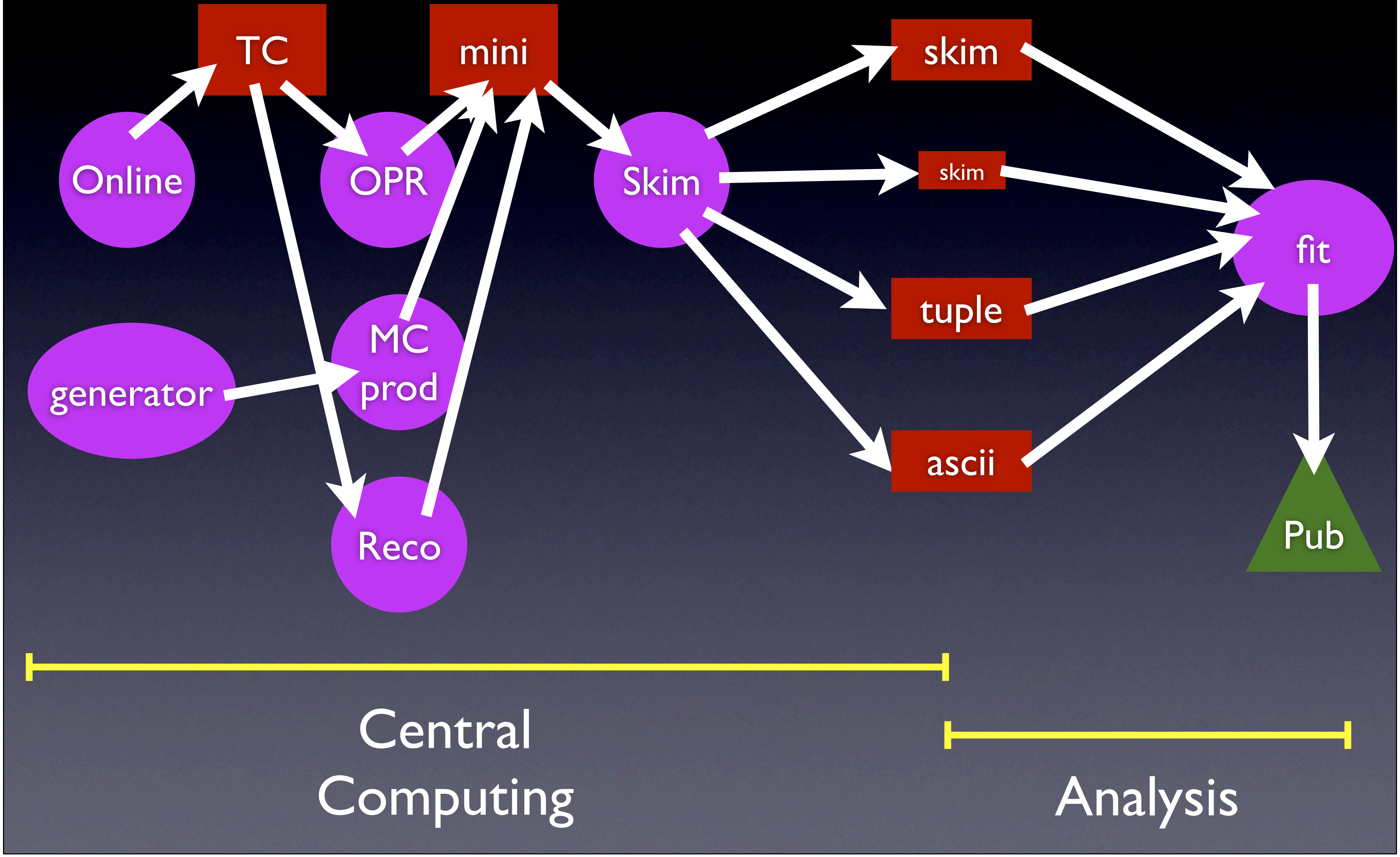
Comp. Org. ‘Proposal’

- Formal definition of roles
  - management = us
  - core = comp. professionals
  - developers = are allowed to commit code
  - users = can access resources (jobs, disk)
- Require Training appropriate to each level
- Assign Responsibilities to each level
- Allow self management within each level
  - Users maintain user-level documentation
  - Users verify/allocate user jobs
  - Developers review code development

Processing Model Principles

- Keep what works
  - mini + micro + UsrData + BtaTuple + ...
- Directly provide what analysts use most
  - micro/mini, tuples, ascii, ...
- incorporate ‘analysis’ into production
  - formalize and standardize formats, procedures
  - bring analysts into computing organization
- Bookkeeping to tie it all together
  - extend to include ‘analysis’ formats

SuperB Processing?



BaBar code base

- BaBar code must be reused
  - 100s (1000s?) of man-years of effort
  - ‘it works’ (solves SuperB physics case)
  - We’re already reusing it
- How to mitigate the ‘toxic waste’?
  - poor design
  - outdated implementation

→

- poor performance
- poor reliability
- difficult maintenance

What and How to Reuse?

- Replace core infrastructure
  - framework, persistence, database, ...
- Define standards
- Rewrite code in sections (1 repository?)
  - prioritized by need, difficulty, expert availability
  - reverse-engineer functionality
  - re-design, re-implement, review
  - Physicist + comp. expert teams
- Migrate all code to new standards
  - hire ‘undergrads’ to fix all gcc4 warnings
- Migrate user code into computing

Centralization?

- Do we want INFN-Frascatti to become another SLAC/CERN computing center?
- Can we design a peer-to-peer distributed computing system?
  - GET vs svn/CVS
  - cloud/GRID vs sites
  - background MC production on _every_ SuperB CPU?

R&D Projects

- Core Infrastructure
- Core technologies
- Code Standards
- Migration procedures
- Prioritization of repository rewrites
- Prototype repository rewrite (Emc reco?)
- Formal definition of Comp. Org.
- Formal definition of processing model
- Formal definition of Dist. Comp. model