On the evolution of CDF data model

P.Murat, Jul 22 1998

1 where we are right now

- design'97: all the interactions between modules go through (TR)YBOS banks, other globals are not allowed
- event processing could be interrupted in between each two modules and resumed later on started from intermediate results written into external file. The results should be identical to those of the single pass processing

2 why do we think about the evolution

- strategy: more C++, less FORTRAN
- "bank-like" vs "non-bank" objects: D-bank vs track or MC particle object
- eficiency issue: redundancy of run-time representation saves time (pt, et, eta, phi, px, py, pz, energy), so for "non-bank" objects persistent and transient rep's may be different (should be if we care about the disk space usage)
- efficiency again: each C++ module has to do packing/unpacking for "non-bank" objects
- logically there are global objects in the system (for example, geometry objects) which we in fact are already using
- the idea is to state explicitly that persistent and transient representations are different and to decouple persistency mechanism and offline code (BaBar has already done this)

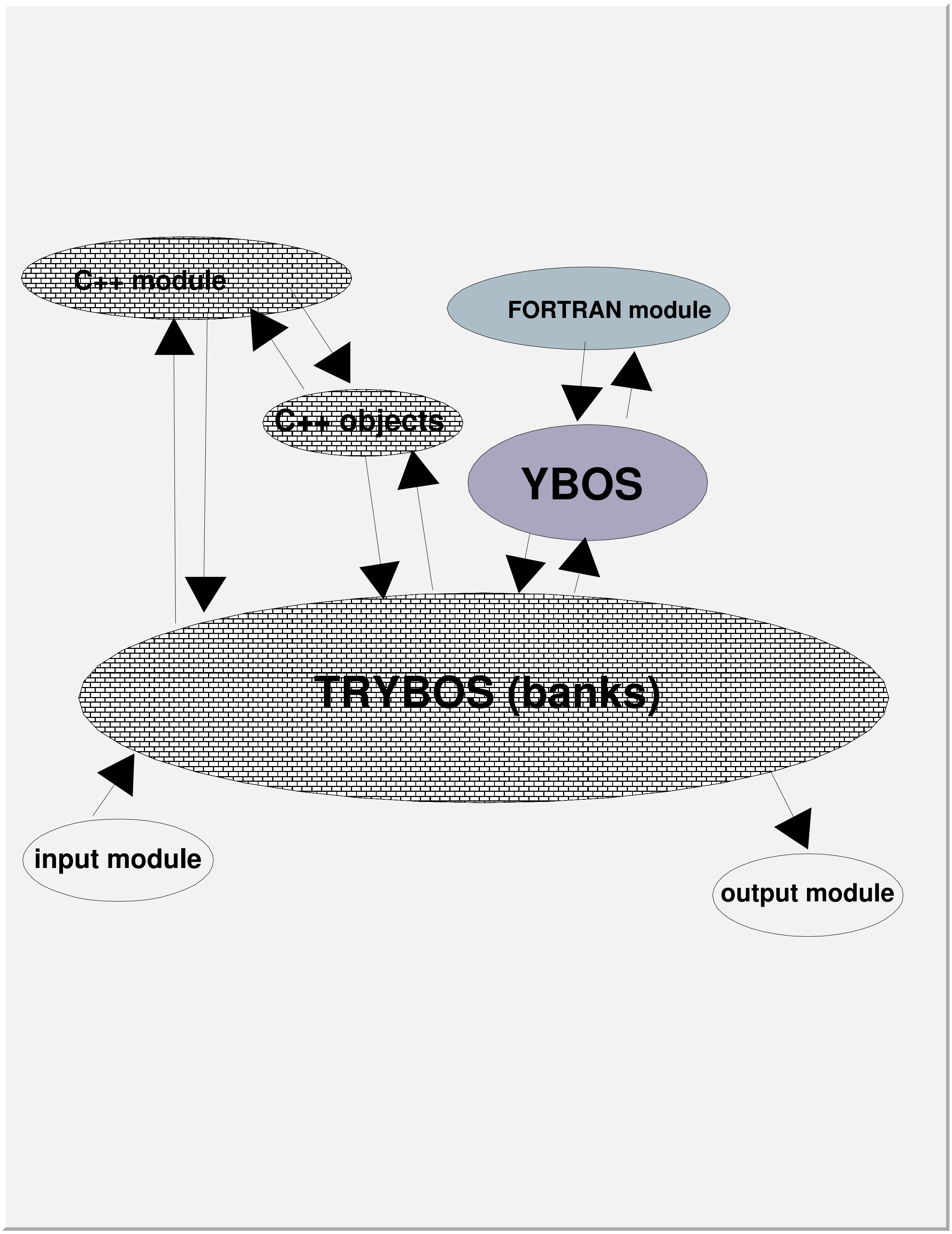

Figure 1: Present CDF data model

3 where and how to evolve

- C++ modules deal with C++ objects, transient representation of objects may be redundant, hierarchical (tracks-vertices-hits-MC truth), objects may have cross-pointers
- persistent representation of objects: bank-like (don't want to write pointers out)
- deferred input: need event data manager
- FORTRAN modules request for the necessary YBOS banks, send back the list of modified/created banks
- keeping in mind long-term development goals may want to limit FORTRAN development in use of YBOS: allow only **blocat, bmake, blist** and may be 3-5 other routines
- ROOT I/O : we know from Rob Kennedy how to write a flat array of YBOS banks into a ROOT file
- next step in evolution of event model: event is a C++ structure

  ```
  LRID*
  TPID*
  ...
  QTRK*
  COTD*
  ...
  array of other banks (TRY_Generic_Bank's)
  ```

- there are banks which format is known to ROOT, at the same time a user have a way of adding an arbitrary piece of data to the event (this is where we need self-describing bank structure)
- iterate on the branch structure of the tree

- ROOT **Streamer** functions provide a natural place for the conversion between persistent and transient rep's
- plan to prototype output of MC with ROOT I/O, everything is ready for it

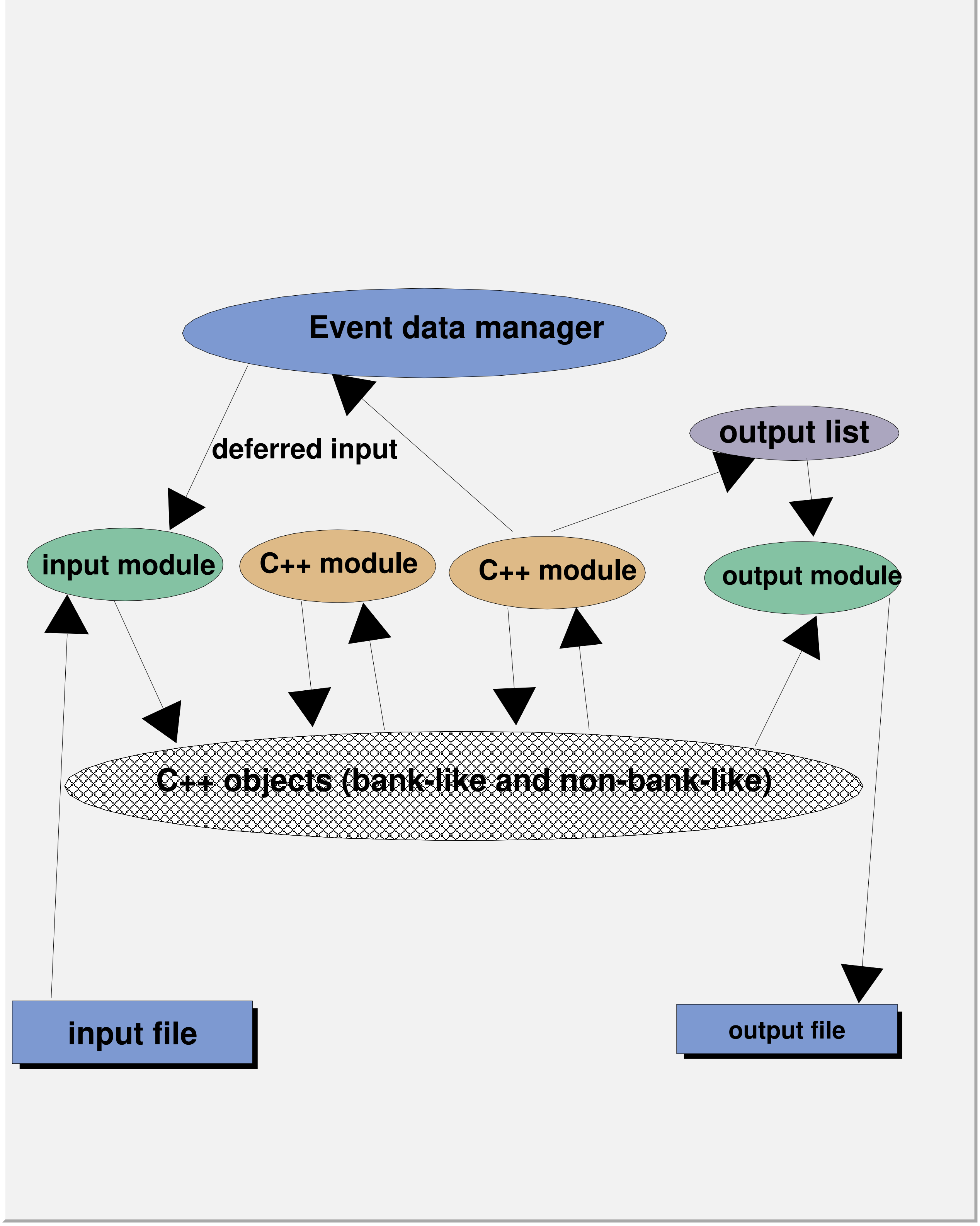
Event data manager
deferred input
output list
input module
C++ module
C++ module
output module
C++ objects (bank-like and non-bank-like)
input file
output file

Figure 2: