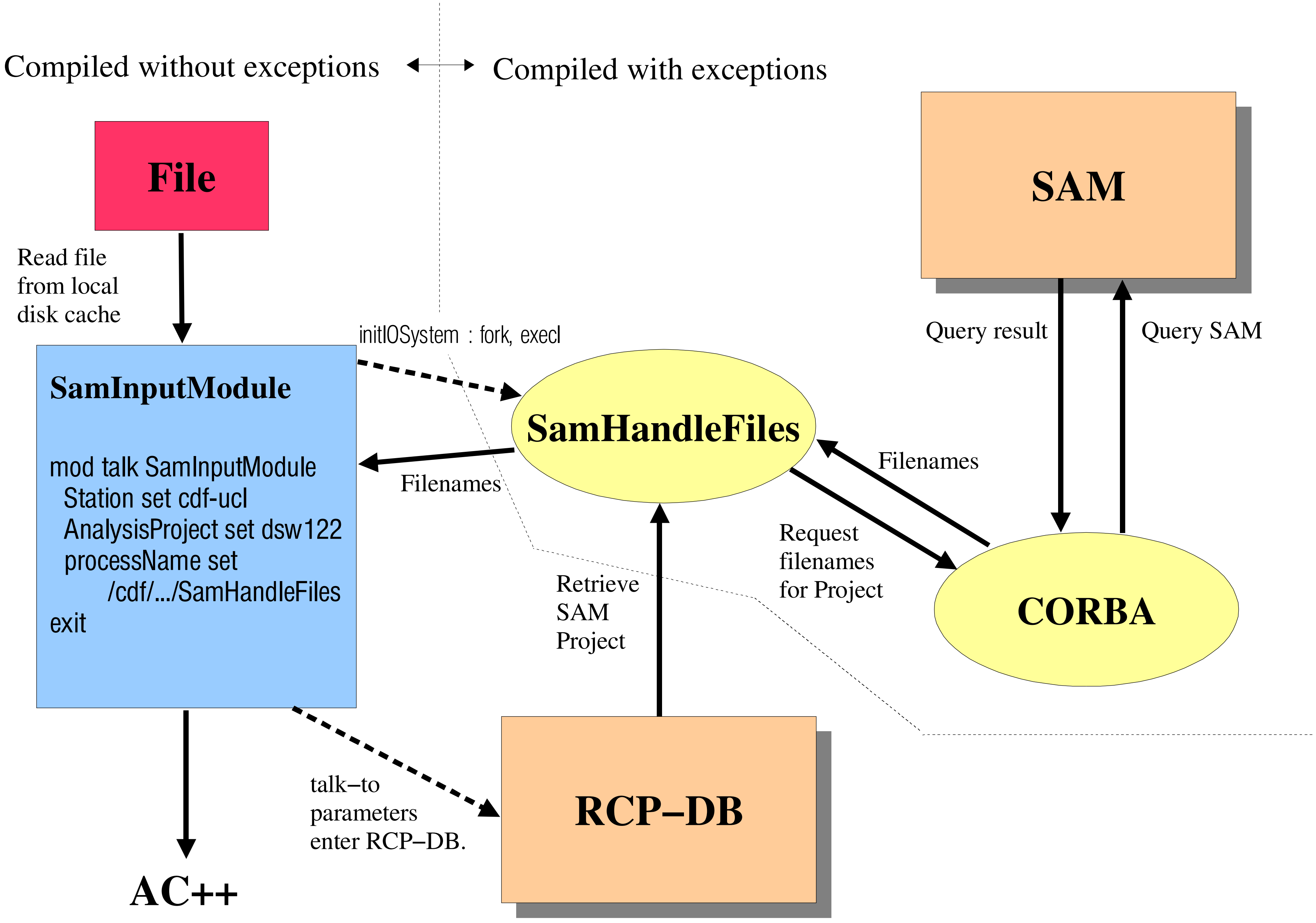
SAM Interface to AC++

(Code largely "borrowed" from D0. CDF interface written by Rick St. Denis and Jim Kowalkowski)

Sequence :

• Wrapper script "runsam" takes as arguments a tcl–file and dataset.
• runsam launches a SAM project ("sam start project ..."). This queries the SAM database for the files constituting the dataset. If not already available on the local SAM station, SAM initiates the copy of the files to local disk cache.
•runsam creates a SamInputModule talk–to fragment containing, amongst other things, the station name, project name and path to the binary "SamHandleFiles".
•runsam launches the AC++ job including SamInputModule : see next slide.
•runsam stops the SAM project.


Compiled without exceptions
Compiled with exceptions
File
Read file
from local
disk cache
SAM
Query result
Query SAM
initIOSystem : fork, execl
SamInputModule
mod talk SamInputModule
Station set cdf-ucl
AnalysisProject set dsw122
processName set
/cdf/.../SamHandleFiles
exit
SamHandleFiles
Filenames
Filenames
Request
filenames
for Project
CORBA
Retrieve
SAM
Project
talk–to
parameters
enter RCP–DB.
RCP–DB
AC++

Development Options :

(1) Keep the structure as it exists : separate SamHandleFiles process.

- Advantages : we keep as much of the SAM/Orbacus interface code as possible in common with D0 – lower maintenance.
- Disadvantages : complex two–process interface. Have to ensure that SamHandleFiles is always available – in part a code management issue.

(2) Put SAM/Orbacus interface code into SamInputModule without exception handling.

- Advantages : single process interface. No need for availability of SamHandleFiles.
- Disadvantages : have to strip exception handling out of existing D0 code. Maintenance issues.

(3) Put SAM/Orbacus interface code into SamInputModule with exception handling.

- Advantages : as above.
- Disadvantages : unknown consequences for rest of CDF software if exception handling is enabled.