LIBPF

Technical Introduction

Summary

Introduction

LIBPF (**LIB**rary for **P**rocess **F**lowsheeting) is a modelling tool for industrial, continuous processes. Its purpose is to make it possible to rapidly prototype and deploy **solutions** for training, process engineer support, on-line process diagnostic, and data reconciliation.

The necessary models (physical properties and unit operations) and tools (solvers, input/output) are made available in the form of a **C++ library**; the **model developer** writes simple C++ code and links it to the LIBPF library to create lightweight, special-purpose application to compute a specific process or process family.

The **model user** receives the model as a black-box executable with no access to the underlying equations and assumptions; she can play around with model inputs and configuration options but no change in the structure is possible.

Comparison with alternatives

With respect to general purpose equation solving environments[1] it offers facilities to handle chemical engineering specific issues like:

- Quantities with physical units of measurement;
- Streams with sub-streams;
- Physical property computations;
- Equilibrium and flashes etc.

With respect to commercial process simulation tools for chemical engineering[2] its scope is much narrower but it is more compact, modular and lightweight, and the licensing can be arranged more flexibly.

Finally LIBPF is most suited for modelling of consolidated processes with a fixed topology, a change of topology requiring a new compilation of 2 % of the code and linkage to a library.

[1] Such as Maple, Mathematica, Matlab, Maxima etc.

[2] Such as AspenPlus, ChemaSim, ChemCAD, gProms, Hysys, ProSimPlus etc.

Features

The unique features of the LIBPF library are:

- Compatibility: it is written in C++, 100% compliant with the «Programming Language C++» ISO/IEC 14882:1998 international standard;
- Leanness: just about 30k LOC (Lines Of Code), thanks to extensive code reuse;
- Completeness: over 250 object types (classes) make it possible to code the model directly in C++ effectively and reliably.

The features of the calculation kernels created by linking to the LIBPF library are:

- Portability: can be ported to any host where a C++ compiler is available;
- Small footprint: the size of the calculation kernel executable is about 2 MB;
- Zero-configuration;
- Is a black-box executable with your models encoded in an opaque manner for the user, for maximum IP (Intellectual Property) protection;
- Interfaces to relational databases (ODBC, SQLITE) and process control equipment (OPC) available as options;
- User interface and enhanced reporting (UI) available as options.

LIBPF maximises reuse of existing libraries and external tools, thereby reducing risks associated with reimplementation:

- STL (Standard Template Library) for data containers;
- OpenMP for multi-threading, to exploit performance enhancements in multi-core or multi-processor workstations;
- boost::graph for graph manipulation;
- AT&T GraphViz 2.17 for graph visualization;
- ADOL-C for Automatic Differentiation;
- ODBC 3.0 for database connectivity;
- OPC Data Access 2.0 for plant connectivity;
- GMM++ 3.05 (Generic Template Matrix C++ Library) for vector and matrix manipulation;
- SUNDIALS for Differential and Differential-Algebraic equation solving.

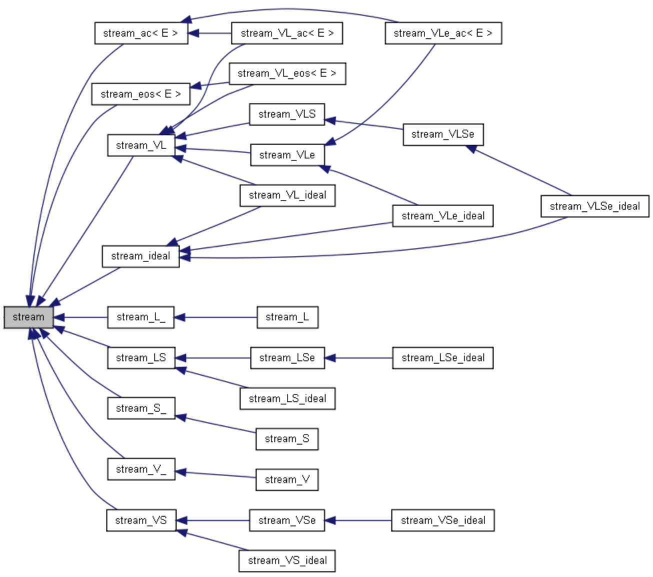
Modelling capabilities

LIBPF natively supports the following modelling features:

- Problem Resolution:
    - Static and dynamic processes;
    - Sequential resolution of flowsheets with material and heat recycle (direct substitution or accelerated direct substitution);
    - Process specifications (feed-back);
    - Sparse analytical derivatives;
    - Simultaneous solution of flowsheets, sparse linear solver;
- Physico-chemical properties:
    - Ideal, NRTL activity coefficient model and basic (Peng-Robinson and Soave-Redlich -Kwong) cubic equations of state;
    - Gas phase, liquid phase or vapour-liquid phase processes (latter only static).

The key object type (`class`) is the material stream type, based on the abstract class `stream` . Every concrete stream type contains at least one total phase such as `stream_V` (just one phase): ; multi-phase streams are composed in a similar way: `stream_VLe`  or the less realistic `stream_VLLSSSe` .

The graph showing the arrangement of the classed derived from stream illustrates these concepts:

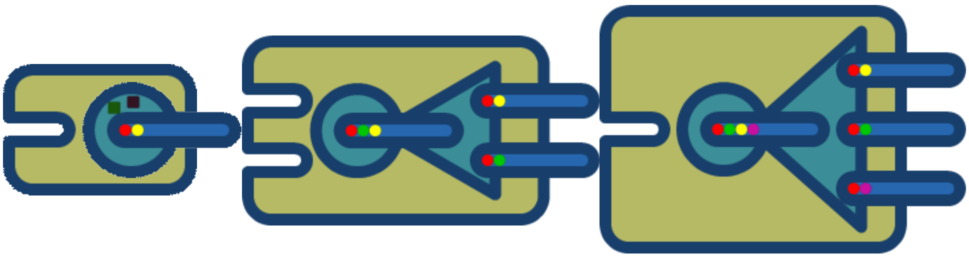


Since the stream type only handles phase equilibrium, we need some additional gadget to handle reactions: the generic flash (`genflash`) mixin:

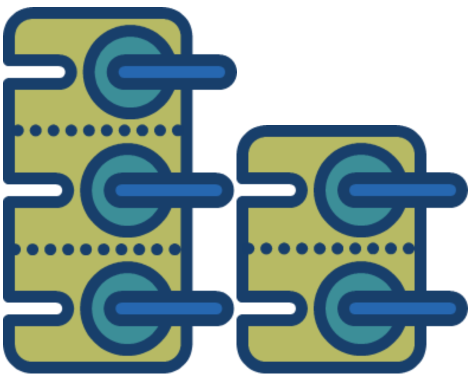
Reactive flashes are obtained by composing the different stream types with reactions:

To do flowsheeting, the connectivity mixins are introduced such as `three_two`, `one_one`:

To get a concrete class representing a unit operation, one connectivity mixins, one genflash and a stream are compose, creating unlimited variations with very high code reuse:



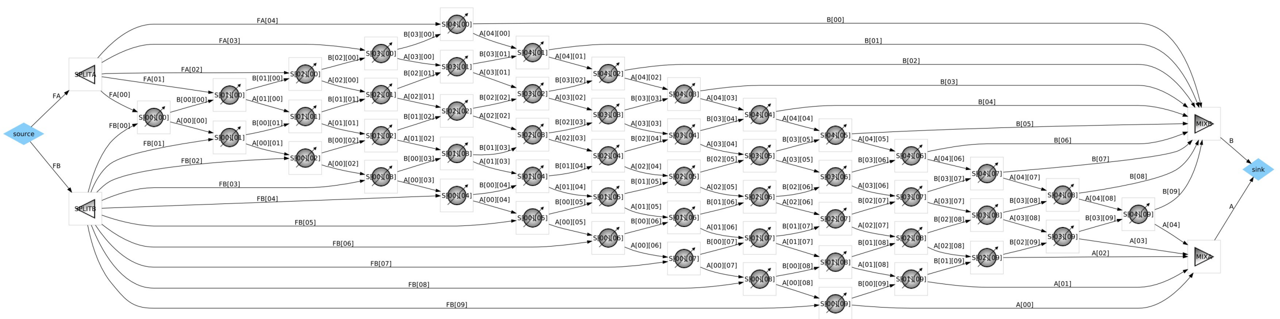
Finally the flexible multi-stream concentrated parameters model multihx is introduced, with N optionally reactive streams exchanging heat or mass; this versatile model can be used for (reactive) multi-stream heat exchangers, (reactive) membrane units or fuel cells:



The unit operations models available in this way in LIBPF are:

- Concentrated parameters:
  - Stream mixer, 2 or more inlets;
  - Stream splitter (tee), 2 or more outlets;
  -
  - Spawn (duplicates the inlet);
  - Fixed-yield separator, 2 or three outlet streams;
  - Optionally reactive (fixed yield stoichiometric or equilibrium) flash with phase equilibrium;
  - Isentropic compressor/expander;
  - Counter-current non-reactive adiabatic HTU/NTU absorber/stripper;
  - Reactive multi-stream heat exchanger;
  - Fuel cell with 2 or more streams, each supporting multiple equilibrium or fixed conversion reactions; one or more electrochemical reactions supported;
  - Flowsheet-in-Flowsheet;
- Distributed parameters:
  - Generic multi-stage 1-D unit, with co-/counter- current flow pattern;
  - Generic multi-stage 2-D unit, with spiral, co-, cross- or counter-flow pattern.

More complex unit operations (distributed parameter fuel cell, plug-flow reactor, zone heat exchanger, distillation column etc.) are obtained as combination of the generic multi-stage 1-D and 2-D unit with the concentrated parameter models. For example a 2-D distributed parameter planar fuel cell, with anode/cathode cross-flow can be represented as a network of 5 x 10 two-stream `multihx` objects:



Solution Architecture

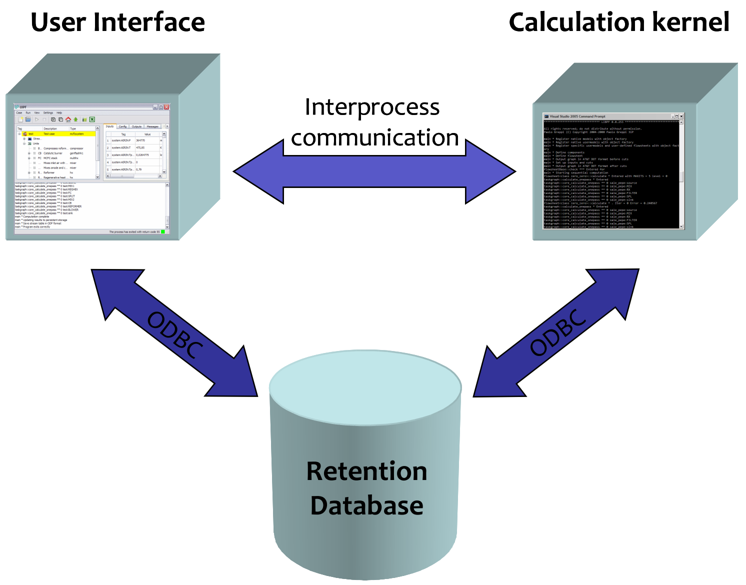
The LIBPF library and products allow to compose a variety of different architectures to suit the requirements of the application. Three basic configurations can be envisaged:

1. Desktop;
2. Enterprise;
3. Embedded.

Desktop Architecture

The architecture for a desktop solution is composed of three components:

1. User interface (LIBPF UI);
2. Calculation kernel: console application without any user interaction developed in C++ using the LIBPF library;
3. RDB (Relational Data-Base) for data storage / retrieval, via ODBC 3.0 driver.



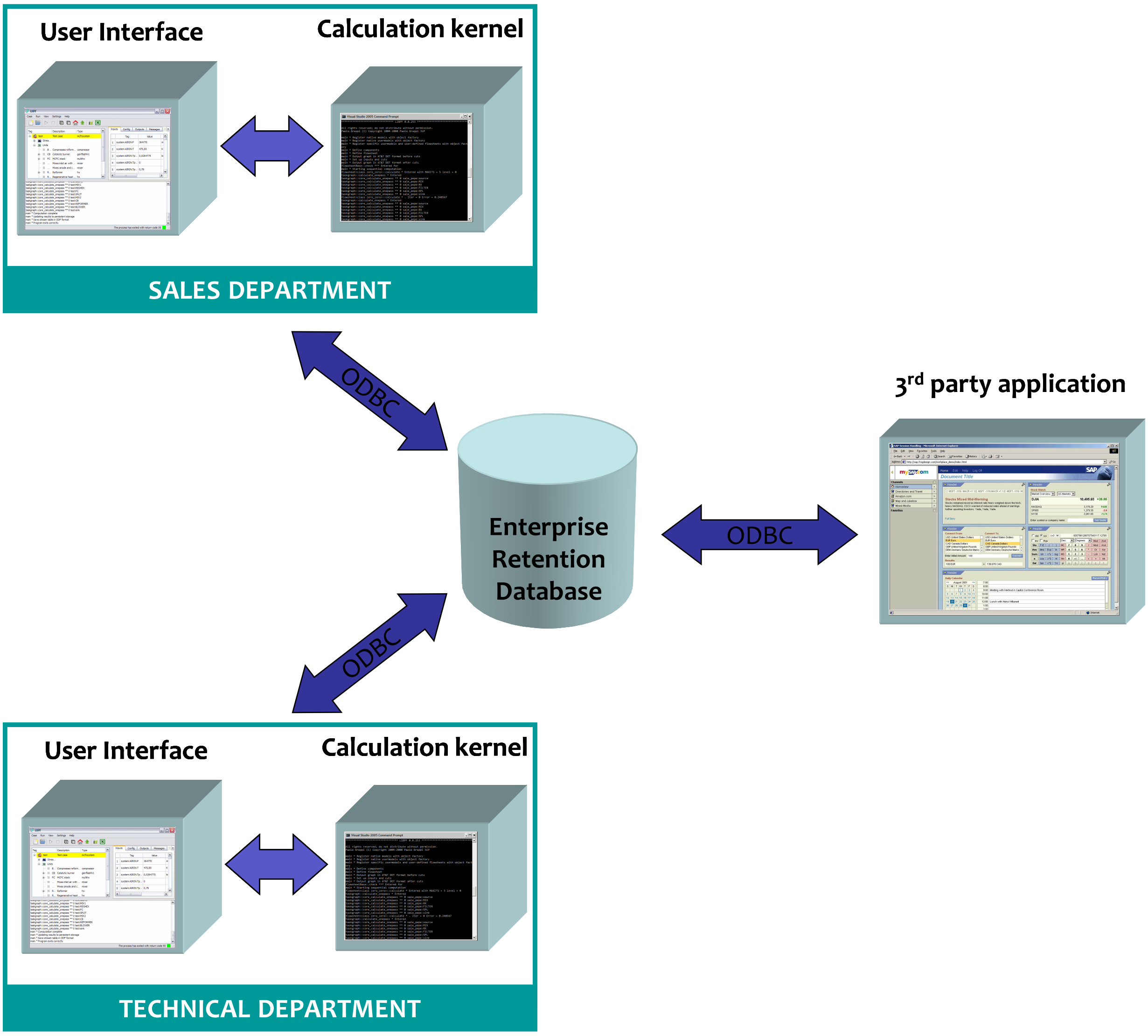
The workflow is: user interacts with the UI to initially instantiate an object of a give type (i.e. a process model with a given structure).

User interacts further with the UI to change the specification; UI writes the new specification to the database via ODBC; user launches the calculation kernel to perform the computation and controls it via interprocess communication; UI interfaces to the database to get and display results.

Enterprise Architecture

The architecture for an enterprise solution is composed of several components:

1. A number of user interface / calculation kernel pairs running locally and concurrently on the dektop of each user;
2. One centralized RDB (Relational Data-Base) to store models and data for all users and makes it possible to exchange data;
3. 3rd party applications can interact with the RDB to integrate with the other information infrastructure.



Embedded Architecture

LIBPF can be used to develop the process-specific part of an on-line application. The calculation kernel is portable and lightweight so that it can be directly installed on the low-cost embedded hardware (such as PLC or SCADA), in a stripped-down configuration consisting only of calculation kernel and RDB (Relational Data-Base):