PROTOMOL – Quo Vadis?

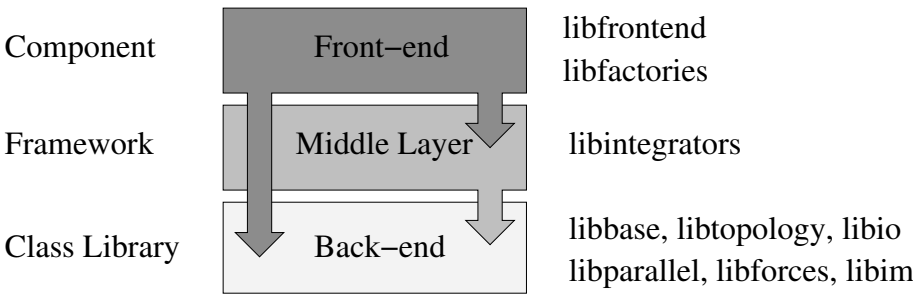
PROTOMOL – Quo Vadis?

2.0

Overview

- Introduction
- Patterns
- Adding new Features

Overview



- `Makeable` and `Value`
- IO, IMD and Factories lib's
- Customized applications

IO

- Read & write to files
- Building blocks for output objects and input
- File; Reader and Writer base classes
- Format (PSF, PDB, XYZ, DCD, PPM, PNG, etc.)
- << and >>

Value

- Generic value with type and constraint
- ValueType – traits (<< and >>)
- ConstraintValueType – traits
- Conversion operatators and safe conversion

Makeable

- Base class of all classes to be create dynamically (Factory)
- Parameter list and check
- Id
- Alias
- Virtual make method at family class level (Output, Force, etc.)

Output

- Output is defined by a collection of output objects
- Output objects are invoked one by one – initialize, run and finalize
- Concrete class implements hook methods of initialize, run and finalize
- Optional fallback method to fix undefined parameters
- Virtual make method at family class level (Output, Force, MTS, STS, etc.)
- Shared cache object (OutputCache: temperture, minial image, etc.)

Patterns

- Abstract description of core solution of a problem
- Pattern name, problem, solution and consequences

Factory (- Method)

Define an interface for creating an object, but let subclasses decide which class to instantiate. Factory Method lets a class defer instantiation to subclass.

Prototype

Specify the kind of objects to create using a prototype instance, and create new objects by copying this prototype.

Singleton

Ensure a class only has one instance, and provide a global point of access to it.

Policy (Strategy)

Define a family of algorithms, encapsulate each one, and amke them interchangeable. Policy lets the algorithm vary independently from clients that use it.

General Design Description of a Force

R1 Algorithm to select an n-tuple of particles

R2 Boundary conditions

R3 Retrieve efficiently ($\mathcal{O}(1)$) the spatial information of each particle

R4 Potential defining the force and energy on the n-tuple

R5 Switching function to modify the potential

Adding a New Integrator

- Find a similar integrator (MTS or STS)
- Take a copy and change it
- Add it to the Makefile
- Register your new integrator (libfactories)
- Reconfigure and recompile

Adding a New Force

- Find a similar force/ potential (CoulombForce, PaulTrapExtendedForce)
- Take a copy and change it
- Add it to the Makefile
- Register your new force (libfactories)
- Reconfigure and recompile

Adding a New Output

- Find a similar output object (OuputDiffusion) – derived from OutpuFile
- Take a copy and change it – `doInitialize`, `doRun` and `doFinalize`
- Add it to the Makefile
- Register your new force (libfactories)
- Reconfigure and recompile

```
class OutputDiffusion : public OutputFile {
public:
  OutputDiffusion();
  OutputDiffusion(const std::string\& filename, int freq);
public:
//  From class Output
public:
  virtual Output* make(std::string\& errMsg, std::vector<Value> values) const;
  // Factory method
protected:
  virtual void doInitialize();
  virtual void doRun(int step);
  virtual void doFinalize(int step);

// From class Makeable
public:
  virtual std::string getIdNoAlias() const{ return keyword;}
  // Returns the identification string

  virtual unsigned int getParameterSize() const {return 2;}
  virtual void getParameters(std::vector<Parameter> \&parameter) const;
// My data members
public:
  static const std::string keyword;

private:
};
```

```
  const string  OutputDiffusion::keyword("diffusionFile");

  OutputDiffusion::OutputDiffusion(): OutputFile(){}

  OutputDiffusion::OutputDiffusion(const string& filename, int freq): OutputFile(filename,freq){}

  Output* OutputDiffusion::make(std::string& errMsg, std::vector<Value> values) const{
    string err;
    string filename;
    int freq;
    if(!values[0].get(filename))
      err+=" Filename \'"+values[0].getString()+"\' for "+getId()+" not valid.";
    if(!values[1].get(freq))
      err +=" Output frequency \'"+values[1].getString()+"\' for "+getId()+" not valid.";

    if(!err.empty()){
      errMsg += err;
      return NULL;
    }

    return adjustAlias(new OutputDiffusion(filename,freq));
  }

  void OutputDiffusion::getParameters(std::vector<Parameter> &parameter) const{
    parameter.push_back(Parameter(getId(),Value(myFilename,ConstraintValueType::NotEmpty())));
    parameter.push_back(Parameter("diffusionOutputFreq",Value(myFreq,ConstraintValueType::Positive())));
  }
}
```

```
void OutputDiffusion::doInitialize(){
  ofstream diffusionHeaderFile(string(myFilename + ".header").c_str(), std::ios::out | std::ios::trunc);
  if(!diffusionHeaderFile)
    report << error <<" Can not open \'"<<myFilename<<".header\' for "<<getId()<<"."<<endr;
  diffusionHeaderFile << setw(18)
                      << "Time(fs)" << " "
                      << setw(24)
                      << "Diffusion(10^-5 cm^2/s)" << " "
                      << setw(14)
                      << "Volume(AA^3)";
  diffusionHeaderFile << endl;
  diffusionHeaderFile.close();
  open();
  close();
}

void OutputDiffusion::doFinalize(int){
  close();
}
```

```
void OutputDiffusion::doRun(int step){
  if(!reopen())
    report << error <<" Can not open \'"<<myFilename<<"\' for "<<getId()<<"."<<endr;

  Real diffusion = 10000.0*myCache->diffusion()*(step > myFirst?1.0/((step-myFirst)
                     * myIntegrator->getThisLevelTimestep()):0.0);
  myFile << resetiosflags(std::ios::showpoint |  std::ios::fixed | std::ios::floatfield)
         << setw(18)
         << setprecision(2)
         << setiosflags(std::ios::showpoint |
                        std::ios::fixed)
         << myCache->time() << " "
         << resetiosflags(std::ios::showpoint |
                          std::ios::fixed)
         << setiosflags(std::ios::floatfield)
         << setprecision(8)
         << setw(24)
         << diffusion << " "
         << setw(14)
         << myCache->volume();
  myFile << endl;
  reclose();
}
```

Resume

- Good sequential and parallel scaling
- General MD framework for research & education
- Platform to develop and evaluate novel methods & algorithms
- C++, 40k LOC & freely available at SourceForge
- Supported computer platforms: AIX, IRIX, Linux, HP-UX, PowerPC and Windows
- Supported compilers: GNU, Intel, MIPSPro, M$ Visual C++, VisualAge C++ and HP-UX aC++.

Acta est fabula, plaudite!