HepML: status and plans

S. Belov, JINR, Dubna
on behalf of MCDB group

LCG MCDB group:
S. Belov, JINR
L. Dudko, SINP MSU
A. Ribon, CERN
A. Sherstnev, Univ. of Oxford

LCG Generator Services monthly meeting
CERN, 8 April 2009

Outline

- What is HepML for
- Why HepML
- Main features
- Examples
- HepML in real projects
- Further developments
- Conclusion

Few problems in Monte-Carlo simulation

- Interfaces between generators
- Passing information between different simulation levels
- Standards for metadata representation
  - Partially solved with LHEF

MCBD uses HepML information structure

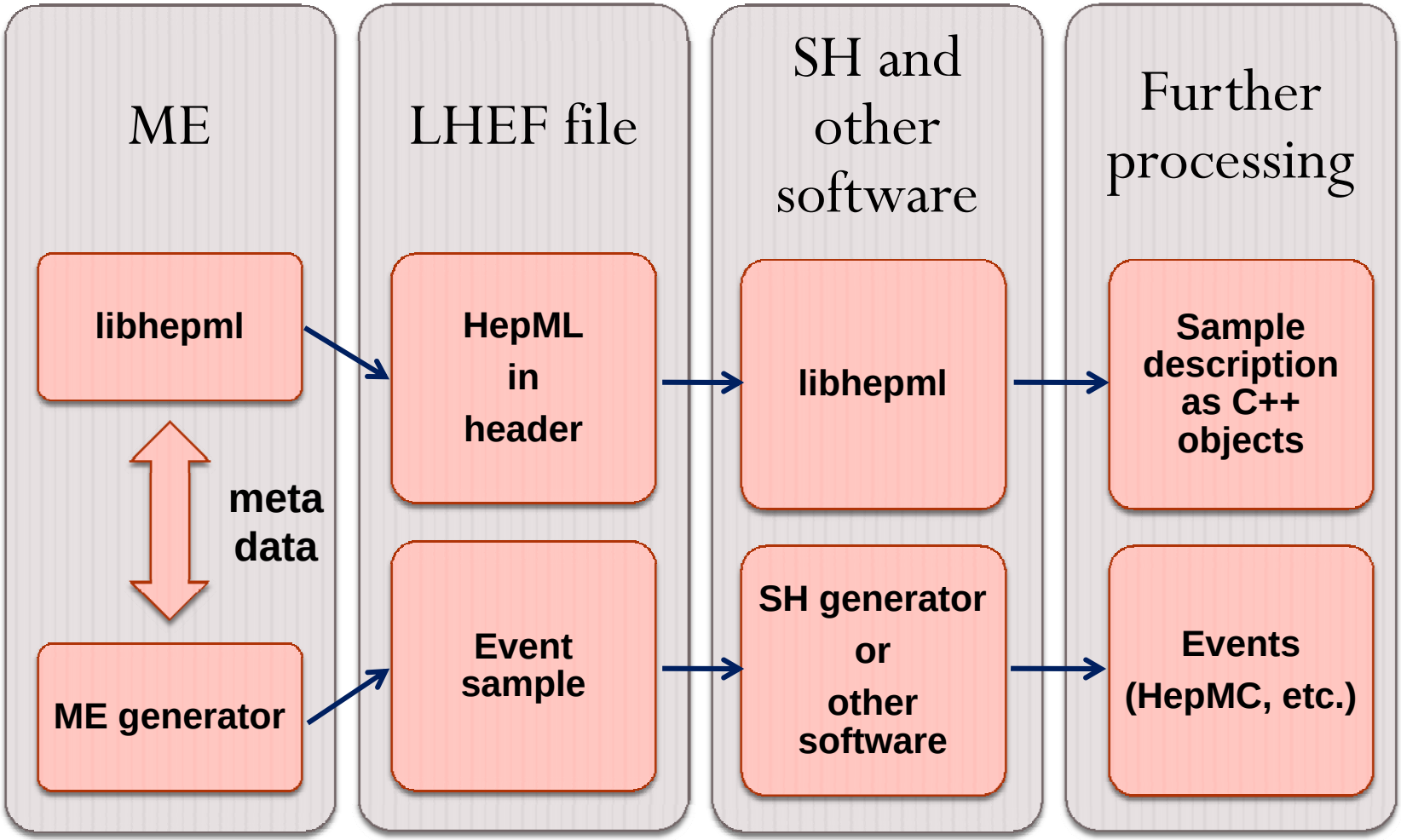
HepML markup language

- Unified XML format of MC event files metadata
  - to store all possible information from MC generators in XML format
  - to store generator input parameters and setup
  - an effort to fix a unified extensible way of MC events description
  - is an allowed part of LHEF standard event file header
- Main purposes
  - to unify MC event files description
  - to facilitate information transfer from Matrix Element generators to Shower generators
  - usage in MC generators tuning and testing
- Contributors
  - LCG MCDB http://mcdb.cern.ch
  - CEDAR http://www.cedar.ac.uk
- Homepage
  https://twiki.cern.ch/twiki/bin/view/Main/HepML

HepML structure

- XML schemas
  - https://twiki.cern.ch/twiki/bin/view/Main/Schema
- Library (libhepml)
  - language: C/C++
  - could be used from Fortran, Perl, Python, etc.
  - for download: http://mcdb.cern.ch/distribution

libhepml in MC simulation



HepML inside LHEF

J. Alwall et al., A standard format for Les Houches Event Files (2006) (*hep-ph/0609017*) structure:

```
<LesHouchesEvents version="1.0">
  <header>
    <hepml>
      <!--
        HepML sample description here
      -->
    </hepml>
  </header>
  <init> ... </init>
  <event> ... </event>
  <event> ... </event>
  ........
</LesHouchesEvents>
```

Main in ME event sample description

General information

- title
- abstract
- authors
- experiment and/or groups

Physics process

- initial state
- final state
- QCD scale
- process PDF

Event files

- physics process/subprocesses
- events number
- cross section and uncertainty
- file name, location(s)

Used generator

- name and version
- description
- home page

Theoretical model

- name
- description
- set of parameters and their values with author's description

Applied cuts

All MCDB interfaces use HepML information structure

What is libhepml

- Library to read/write/modify HepML blocks of information (LHEF, MCDB, ...)
- Simple and reliable way to deal with ME generator metadata
- Intended to automatize sample description in the MC simulation chain

Some C++ classes of *libhepml*

```
namespace mcdb
{
    class Article;
    class File;
    class Author;
    class Cut;
    class Generator;
    class Model;
    class Process;
    class Subprocess;
}
```

```
class Generator{
public:
    Generator();
    ~Generator();
    string& name();
    string& name(const string&);
    string& version();
    string& version(const string&);
    string& homepage();
    string& homepage(const string&);
private:
    ....
};

class Process{
public:
    Process();
    ~Process();
    string& initialState();
    string& initialState(const string&);
    string& finalState();
    string& finalState(const string&);
    string& factScale();
    string& factScale(const string&);
    string& renormScale();
    string& renormScale(const string&);
    string& pdf();
.....
```

```
class Model{
public:
    Model();
    ~Model();
    class ModelParameter;
    string& name();
    string& name(const string&);
    string& description();
    string& description(const string&);
    vector<ModelParameter>& parameters();
    vector<ModelParameter>&
    parameters(const vector<ModelParameter>&);

     class ModelParameter
     {
       public:
           ModelParameter();
         ~ModelParameter();
          string& name();
          string& name(const string&);
          string& value();
         string& value(const string&);
         private:
         string name_;
         string value_;
      };
       private:
```

Some C functions of *libhepml*

```
void * init_lhaef_document (void);
int set_general_description(void * doc, lhaef_general_description * sample);
int set_file_description(void * doc, lhaef_file * file);
int set_generator(void * doc, lhaef_generator * gen);
int set_model(void * doc, lhaef_model * model);
int add_model_parameter(void * doc, lhaef_parameter * par);
int set_cutset(void * doc, int cutsetnumber);
int add_cut(void * doc, lhaef_cut * cut);
int set_author_record(void * doc);
int add_author(void * doc, lhaef_author * author);
int create_process(void * doc, lhaef_process * proc);
int add_process_beam(void * doc, int i, lhaef_beam * beam);
int add_process_beam_pdf(void * doc, int i, lhaef_pdf * pdf);
int add_process_beam_pdf_alphas(void * doc, lhaef_alphas * alphas);
int add_process_alphas(void * doc, lhaef_alphas * alphas);
int add_subprocess(void * doc, lhaef_subprocess * subproc, int
cutsetnumber);
char * form_hepml_document(void * doc);
```

Features of libhepml

- Creating standard HepML headers from C++, C and Fortran programs (generators) in a very simple way
- Reading and parsing HepML
  - result is C++ objects
  - obtained data could be easily converted to other formats (HTML, plain text, ROOT trees, etc.)
- Mixing HepML headers of event files
- Modification and saving HepML headers

Current state of developments

- Full support of HepML schema release 0.1
  - read/write/parse support
- Partial support for new – significantly improved - HepML schema release 0.2
  - header writing support
  - HepML parsing (in progress)
  - mixing (in progress)

- Release for public use is LibHepML 0.2

HepML Header Writing example – C (I)

```c
int main()
{
    void * doc;
    char * hepml_doc;

    doc = init_lhaef_document();
    form_sample_article( doc );
    form_hepml_document( doc, &hepml_doc, 0, 1 );

    printf( "%s\n", hepml_doc );

    release_hepml_document( hepml_doc );
    release_lhaef_document( doc );

    return 0;
}
```

HepML Header Writing example - C (II)

```
void form_sample_article(void * doc)
{
    lhaef_general_description * sample;
    sample = calloc(1, sizeof(lhaef_general_description));
    strcpy(sample->title, "p,p->Wbbj->l,nu,b,b,j process from CompHEP");
    strcpy(sample->abstract, " There are about 1.1 MEvents of CompHEP p,p-
    >Wbbj with leptonic decays of W. Spin correlations and contribution from
    the initial sea quarks were taken into account.");
    sample->experiment = calloc(1, sizeof(lhapdf_experiment) );
    strcpy(sample->experiment->experiment, "CMS");
    strcpy(sample->experiment->group, "RDMS");

    set_general_description(doc, sample);
   ...
```

HepML Header Writing example - C (III)

```
    lhaef_model * model;
    model = calloc(1, sizeof(lhaef_model) );
    strcpy(model->name, "SM, Feynman gauge");
    set_model(doc, model);
    …
    lhaef_parameter * param;
    lhaef_math_notation * notation;
    param    = calloc(1, sizeof(lhaef_parameter) );
    notation = calloc(1, sizeof(lhaef_math_notation) );
    param->notation = notation;

    strcpy(param->name, "Ms");
    strcpy(param->value, "0.117");
    strcpy(notation->plain, "Ms");
    strcpy(notation->html, "m&lt;sub&gt;s&lt;/sub&gt;");
    add_model_parameter(doc, param);
  ……
}
```

HepML Header Writing example – C++ (I)

```
#include “hepml.hpp“
…
mcdb::Article article;
article.title() = “article title”;
article.abstract() = “abstract is here”;
…
article.generator().name() = “gen name”;
article.generator().version() = “ 1.2.3”;
….
mcdb::Subprocess sub;
sub.crossSection().unit = mcdb::CrossSection::mub;
 article.process().subprocesses().push_back( sub );
article.process().subprocesses().push_back( mcdb::Subprocess() );
…
```

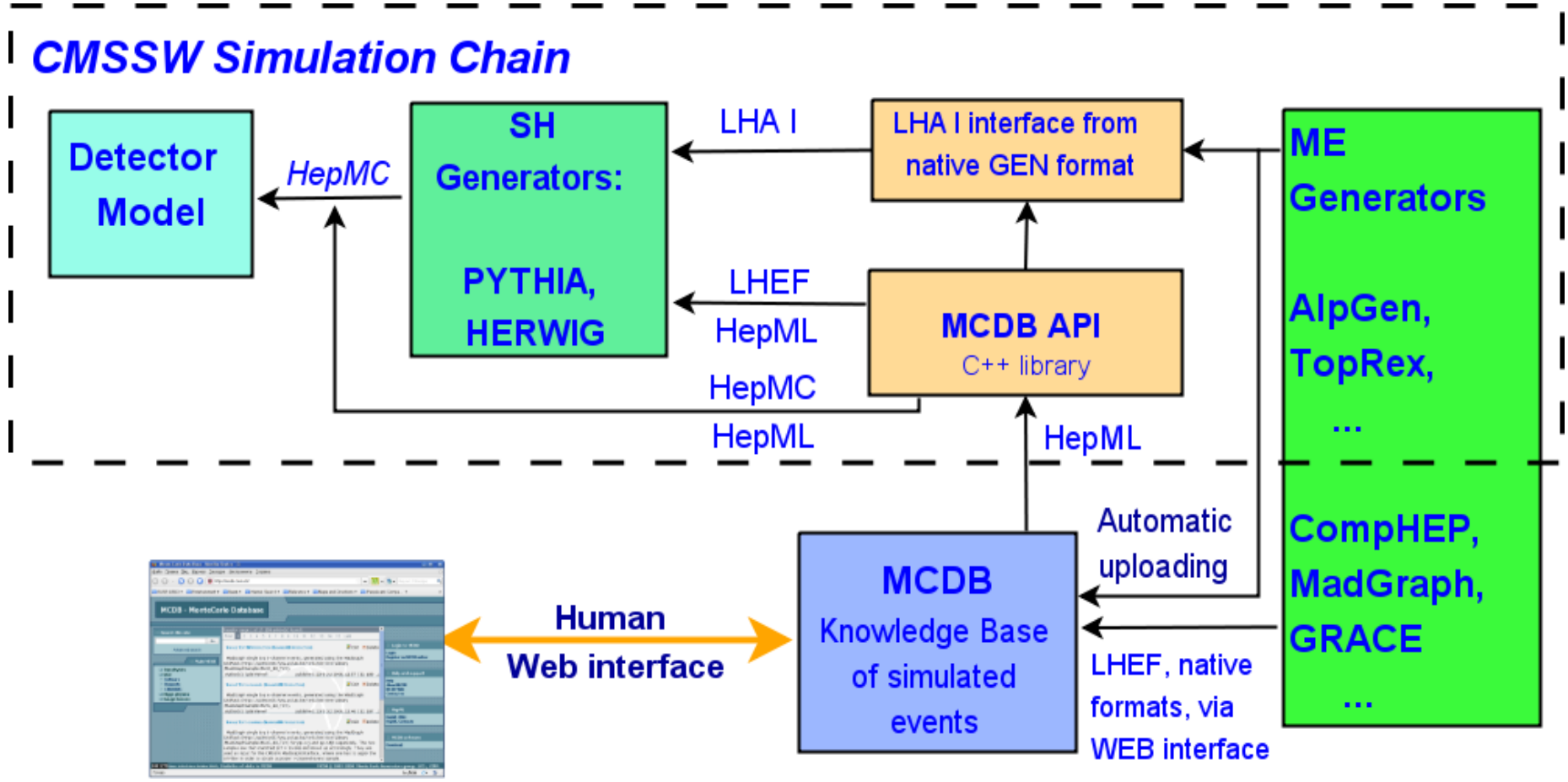
HepML Header Writing example– C++ (II)

```
mcdb::Model::Parameter mp;
mp.name() = “s12";
mp.value() = "s12 value";
article.model().parameters().push_back( mp );
….
mcdb::Author author1, author2;
au1.firstName() = “Sergey";
au1.lastName()  = “Belov";
…
a.authors().push_back( author1 );
a.authors().push_back( author2 )
…..
cout << hepml::articleToXml( article);
```

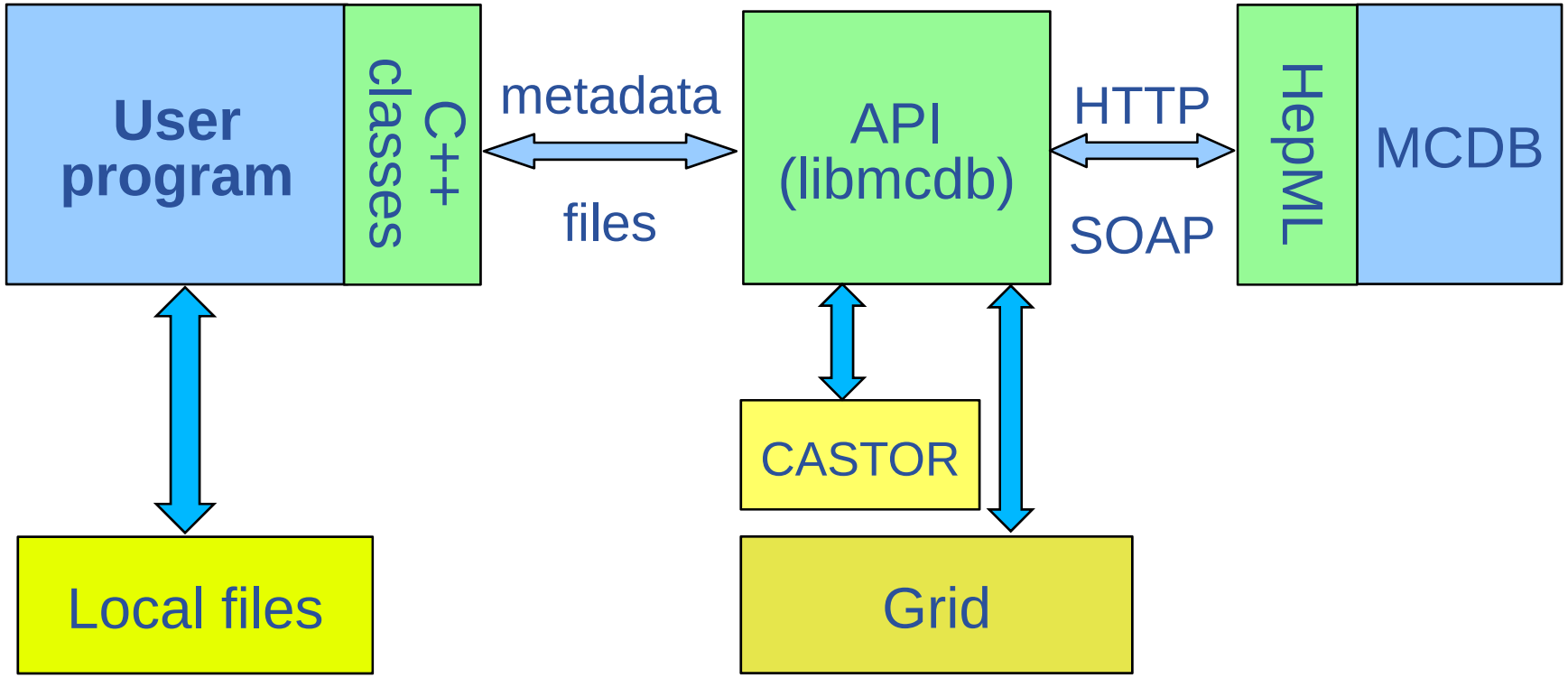
HepML current usage

- CMSSW
  - In LHEInterface: access to MCDB content
- CompHEP
  - generation metadata in HepML format as part of LHEF header (*self-documented event sample*)
  - event samples mixing using information from HepML block
- MCDB
  - In API to automated access of MCDB (upload, download event samples and its' descriptions in HepML form)
  - libmcdb: LCG AA external library /afs/cern.ch/sw/lcg/external/mcdb
- libhepml – standalone library
  - C/C++ interfaces
  - writing HepML documents, parsing HepML
  - available here: http://mcdb.cern.ch/distribution

Simulation chain: CMSSW example



Software access to MCDB



Plans

- Parsing for HepML 0.2
  - Support for common libraries (libxml2, xerces, etc.)
- Full join of CEDAR and LCG branches
- Make HepML an Open Source project
  - Appropriate place – HepForge
    http://hepforge.org

Conclusion

- Schemas are developed and public available
  - https://twiki.cern.ch/twiki/bin/view/Main/HepML
- libhepml is an easy way to store and get metadata from standard HepML descriptions
- HepML is already in use in several projects
- We propose HepML for the standard way to keep ME generator metadata in LHEF
- All suggestions are welcome

- Library is available for download:

  http://mcdb.cern.ch/distribution

  http://simu.cvs.cern.ch/cgi-bin/simu.cgi/simu/GENSER/MCDB/hepml/

Thank you for your attention!

Questions?

BACKUP

Few tasks in Monte-Carlo simulation

- Data reliability, correctness and availability
  - verified reliable Monte-Carlo data
  - comprehensible detailed description (for the future use)
  - public available data (simulated events, models, cuts, etc.)
- Passing data through the simulation chain (formats, standards)
  - different data formats (now less problems with LHEF introduction)
  - need to have additional information along with Monte-Carlo events
- Metadata handling and persistence (models, processes, cuts, etc.)
  - unified way to describe and store these objects
  - re-usability of such descriptions
  - good place to have it – with Monte-Carlo events