
Accelerator Description Formats

Nikolay Malitsky (BNL) and Richard Talman (Cornell)
malitsky@bnl.gov

Outline

Abstract: Being an integral part of accelerator software, accelerator description aims to provide an external representation of an accelerator's internal model and associative effects. As a result, the choice of description formats is driven by the scope of accelerator applications and is usually implemented as a tradeoff between various requirements: completeness and extensibility, user and developer orientation, and others. **Moreover, an optimal solution does not remain static but instead evolves with new project tasks and computer technologies.** This talk presents an overview of several approaches, the evolution of accelerator description formats, and a comparison with similar efforts in the neighboring high-energy physics domain. Following the Accelerator-Algorithm-Probe pattern, **we will conclude with the next logical step, Accelerator Propagator Description Format (APDF),** providing the flexible approach for specifying associations between physical elements and evolution algorithms most appropriate for the immediate tasks

- ☐ Multi-Purpose Input Language
- ☐ Application Programming Interface
- ☐ Exchange Format
- ☐ Propagator Format Extension
- ☐ Composite Approach

Accelerator Multi-Purpose Input Language

Intend: Accelerator-oriented user-friendly Interface

Earlier (and present) issues:

- Variety of accelerator programs (E. Keil, Computer Programs in Accelerator Physics, AIP, 1985): COMFORT, DIMAD, MAD, MARYLIE, SYNCH, TEAPOT, TRANSPORT,
- Variety of different projects
- Variety of different tasks: design optimization, correction, tracking, instabilities, ...
- Variety of algorithms. For example, non-linear approaches : aberration formalism, Lie-algebra technique, symplectic integrator, differential algebra ...

Solution:

A workshop for the standardization of **MAD input language** for beam optics program, SLAC, May 1984.

MAD Input Language

Major features:

- Comprehensive accelerator description model based on two major concepts: accelerator element and beam line
- Classification of element types
- Classification of element attributes
- Conventional grammar based on a single rule :
label : keyword {, attribute}
- Support of the procedural programming mechanisms such as subroutine, DO loop, variables, and others

Open issues:

- FORTRAN constraints and limitations prevented the adoption of the MAD parser as a sharable module among the different accelerator programs. This lead to the development of numerous dialects.
- The convenient set of the MAD programming features could not substitute the power of standard programming languages, such as C or FORTRAN. That prevented the description of the complex scenarios, such tune modulation, etc.

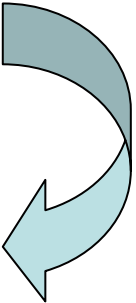
Application Programming Interface

Intend: open accelerator programs for extensions and complex scenarios

Solution: replacing the accelerator input language parser with the standard programming languages

Influential Examples: COSY INFINITY, MXYZPTLK

```
INCLUDE 'COSY'  
PROCEDURE RUN;  
  PROCEDURE SQ PHI L B D;  
  
  ....  
  ENDPROCEDURE;  
OV 5 2 0;  
UM ;  
DL .1; { drift }  
SQ 30 .2 .1 .1;  
PM 6;  
ENDPROCEDURE;  
RUN;  
END;
```



adding a new element type “on fly”

Application Programming Interface (Continued)

Other examples:

- ❑ **PAC++ (1994):** $\text{SBend hb} = \text{length} * L + 2 * \text{PI} / N * \text{ANGLE};$
- ❑ **UAL Perl API (1996):** `$hb->set("l" => $length, "angle" => 2*$pi/$n);`
- ❑ **ROOT Detector Geometry (now):**

```
TGeoCombiTrans* ct_tpad31000 = new TGeoCombiTrans();  
tpss->AddNode(tpad3,1000,ct_tpad31000);  
ct_tpad31000->RotateX(0.0);  
ct_tpad31000->RotateY(0.0);  
ct_tpad31000->RotateZ(15.0);
```

Open Issues:

strong bias towards the associated software environment.

Accelerator Exchange Format

Intend: Context-free format for connecting the multi-lab and multi-system heterogeneous environment

TEAPOT fort.7 (1987)

- ❑ TEAPOT
- ❑ **Structure:** thin lens machine
- ❑ **Technology:** Fortran

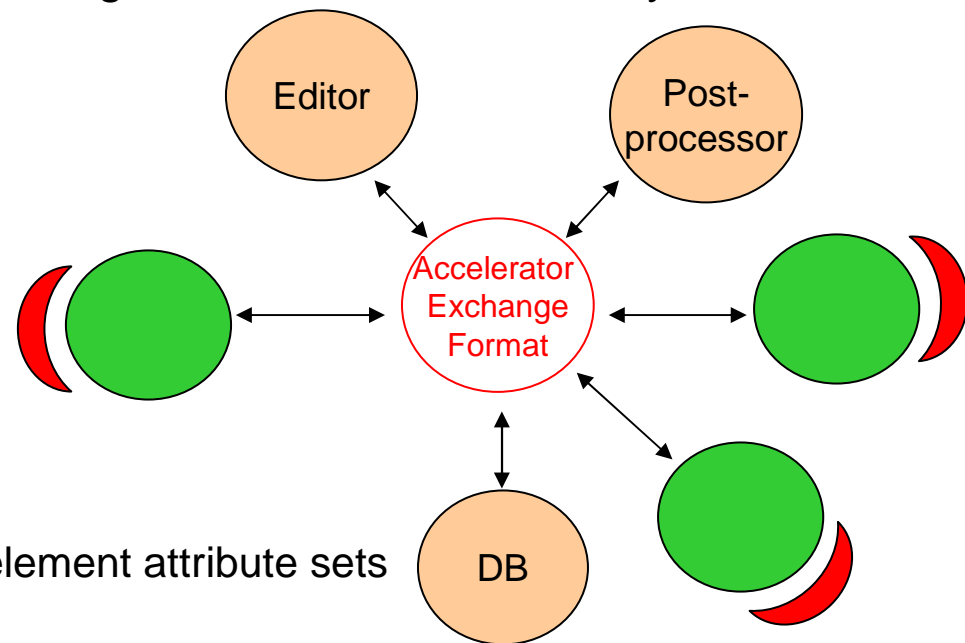
SXF (1998)

- ❑ US-LHC collaboration; MAD-X, UAL
- ❑ **Structure:** MAD sequence and UAL element attribute sets
- ❑ **Technology:** C++, flex/yacc

ADXF 1.0 (1998)

- ❑ Proposal
- ❑ **Structure:** MAD sequence and UAL element attribute sets
- ❑ **Technology:** XML

ADXF 2.0, AML (2005)



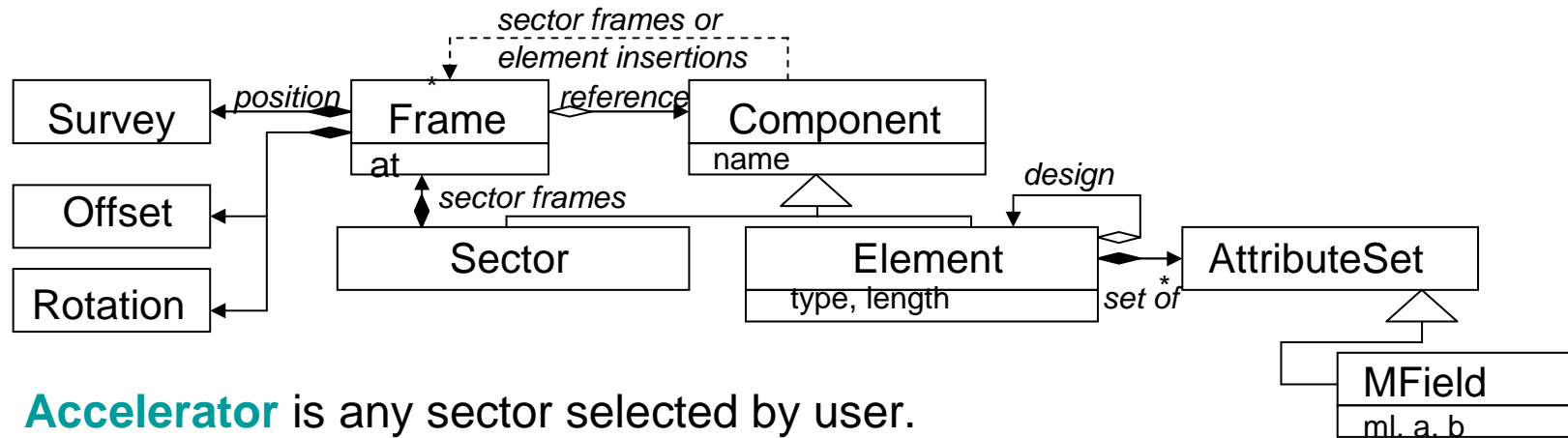
ADXF 2.0 New Objectives and Features

- ❑ **Inspired** by E.Forest's "fibre bundles" and ROOT Detector geometry description (*)
- ❑ **Adds** a new concept, "installed" element.
- ❑ **Addresses** several issues:
 - families, buses, ramp, *etc.*
 - sharing of common "real" elements by more than one sector;
 - supporting of design and operational accelerator descriptions;
 - connecting of accelerator and detector models;

*ROOT Geometry Manual:

"The basic components used for building the logical hierarchy of the [detector] geometry are the **positioned volumes** called nodes. Volumes are fully defined geometrical objects having a given shape and medium and possible containing a list of nodes. Nodes represent just positioned instances of volumes inside a container volume".

Dictionary of Accelerator Model Concepts



- ❑ **Accelerator** is any sector selected by user.
- ❑ **Sector** is a named sequence of frames with installed accelerator components.
- ❑ **Frame** is a layout of an installed component. It contains a relative position (longitudinal position “*at*” or Survey coordinates), misalignments, and a reference to an associated component, sector or accelerator element.
- ❑ **Accelerator Element** is a leaf component in the accelerator tree organization. There are many different types of accelerator elements (e.g. sbend, quadrupole, etc.) But all of them have the same structure: name, length, and an open collection of attribute sets. Element may have a reference to the design element.
- ❑ **Element Attribute Set** is a container of attributes relevant to the single physical effect or feature (e.g. magnetic field, aperture, etc.)

UML-to-XML Schema Mapping

Definition:

Element is a leaf component in the accelerator tree organization. There are many different types of accelerator elements (e.g. sbend, quadrupole, etc.) , but all of them have the same structure: name, length, and an open collection of attribute sets. Elements may have a reference to the design element.

XML Schema:

```
<element name = "element">
  <complexType>
    <attribute name="type" />
    <attribute name="name" />
    <attribute name="l" />
    <attribute name="design" />
    <sequence>
      <element name="attributeSet" minOccurs="0" maxOccurs="unbounded" />
    </sequence>
  </complexType>
</element>
```

Example:

```
<!-- design quadrupole q1 -->
<element type="quadrupole" name="q1" l="1.2" >
  <mfield b = "0. 0.024" />
</element>
<!-- real quadrupole q1_1 with a set of measured multipole harmonics -->
<element name="q1_1" design="q1">
  <mfield b = "0. 0.025 0.0004" />
</element>
```

ADXF 2.0 vs MAD-X Design Lattice description

MAD-X:

```
LD0 := -RHOD0 * (THD0 - THDX);  
D0MP08: SBEND, L := LD0, ANGLE := THD0 - THDX;  
D008B: LINE = ( OD0FLX, ERD0MP08, D0MP08, ELD0MP08, OD0FLA );
```

ADXF:

```
<constant name="ld0" value = "-RHOD0 * (THD0 - THDX)"/>  
<element type="sbend" name="D0MP08" l = "ld0">  
  <bend angle="THD0 - THDX" />  
</element>  
  
<sector name="D008B" line="OD0FLX, ERD0MP08, D0MP08, ELD0MP08,  
  OD0FLA" />  
  
<sector name="D008B" >  
  <frame ref = "OD0FLX" />  
  <frame ref = "ERD0MP08" />  
  <frame ref = "D0MP08" />  
  <frame ref = "ELD0MP08" />  
  <frame ref = "OD0FLA" />  
</sector>
```

Two variants



ADXF 2.0 vs SXF Operational Lattice Description

SXF:

```
bi8-dh0
  sbend {
    tag = d0mp08
    at = 661.74662424
    l = 3.58896623069
    body = { kl = [ -0.0151862520728 ] }
    body.dev = { kl = [ 0 0 0.005476 0.033503 -16.112848 316.932487 1324982.270185 2770273.563708
                        -10008730294.68691 207554314866.7792 ]
    kls = [ 0 0 -0.010166 0.024366 19.256818 316.932487 -318977.213193 -2770273.563708
            2859637227.053402 ]
  }
};
```

ADXF:

```
<sbend name = "d0mp08" l = "3.58896623069" angle= "-0.0151862520728 ">
  <element name = "bi8-dh0" design = "d0mp08" >
    <mfield b = "0 0 0.005476 0.033503 -16.112848 316.932487 1324982.270185 2770273.563708
                -10008730294.68691 207554314866.7792"
    a = "0 0 -0.010166 0.024366 19.256818 316.932487 -318977.213193 -2770273.563708
        2859637227.053402"
  />
</element>
<sector name="blue" >
  ...
  <frame at="661.74662424" ref="bi8-dh0">
  ...
</sector>
```

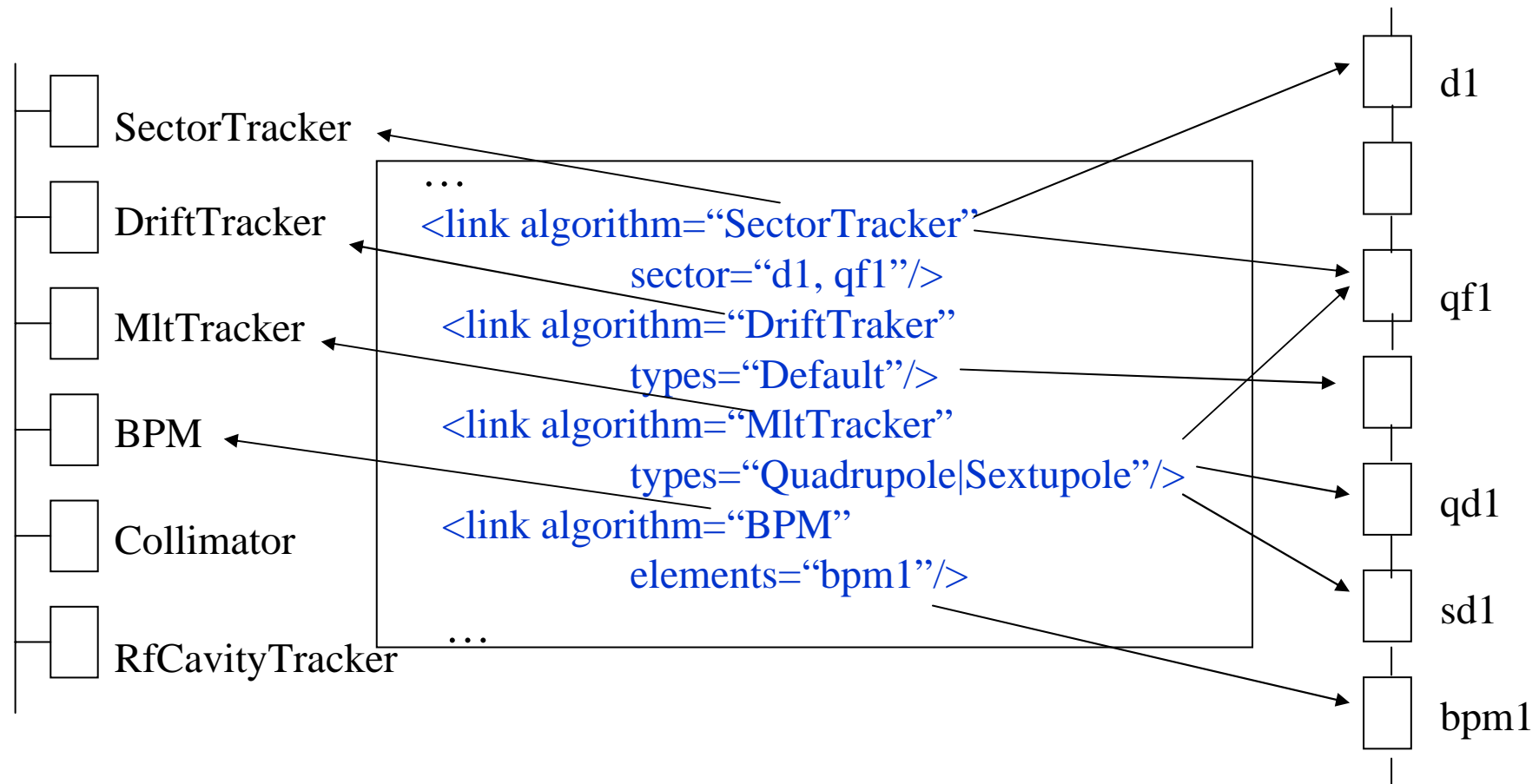
ADXF Accelerator vs ROOT Detector Models

ADXF Concept	ADXF Accelerator Model	ROOT Detector Model
Accelerator	Accelerator: any selected sector	Detector: any selected volume
Sector	Sector: sequence of frames with installed components. Sector does not have its own attribute sets	VolumeAssembly: container of nodes (positioned volumes); it does not have its own attributes, such as shape and medium
Frame	Frame: an layout of an installed accelerator component. It has a relative position, misalignments, and a reference to a component (sector or element)	Node: positioned volume. It has a relative position (geometrical transformation) and a reference to a volume.
Position	Longitudinal position “at” or its Survey coordinates in the sector local coordinate system.	Matrix: geometrical transformation from parent to local reference frame.
Element	Element: is a fully defined accelerator object having a type, collection of attribute sets, and possibly containing a list of node-insertions.	Volume: fully defined geometrical object having a given shape, medium, and possibly containing a list of nodes.
Element Type	Component’s attribute used by the corresponding tracker. There are several element types, such as sbend, quadrupole	There is no special volume attribute. But for the sake of comparison, it could be associated with a Shape EShapeType. There are several shape types (primitives)
Type-specific sets of attributes	AttributeSet ’s of magnetic and RF fields, e.g. MField, Bend, RfField, etc.	Primitive Shape ’s
Common sets of attributes	Aperture	Material

Accelerator Propagator Description Format (APDF)

Catalog of Algorithms

Accelerator (SXF)



(Some) APDF-based applications

- **Longitudinal Dynamics**: sector 2D matrices + RF
- **Dynamic Aperture**: element-by-element tracker with type-based associations
- **Online model**: sector maps + trackers of selected elements
- **Instrumentation modeling (e.g. MIA, BTF)** : conventional tracker + diagnostics devices
- **Special localized effects (e.g. beam-beam)**: conventional tracker + propagator for special effect
- **Space charge studies**: conventional tracker + space charge insertions + impedance
- **Spin studies**: Spin tracker as a wrapper of conventional tracker
- ...

Composite Approach

based on our experience with UAL

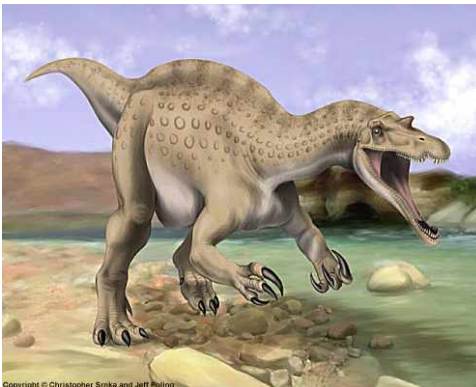
1. Accelerator file
2. Propagator files
3. C++/CINT Shell with a dynamic collection of arguments: **shell.run(Args << Arg("bunch", bunch) << ...**

UI has been divided into

1998

1. Accelerator file (SXF, XML)
2. Perl User Shell with a dynamic collection of arguments: **\$shell->run("bunch" => \$bunch, ...);**

B.C. (Before C++)



Perl API

1996

\$element->set ("angle" => 0.2, ...

C++ API

1994

element.set(0.2*ANGLE + ...