Accelerator Markup Language and the ED Phase

PT

(With *a lot* of help from David Sagan)

SLAC

Beamline Description – The Situation Today

- ILC standard language is Extended Standard Input Format (XSIF)
  - **Descended from an early form of MAD's Standard Input Format (SIF)**
    - Add apertures, linear acceleration, arbitrary changes of accelerator alignment axis
    - Retrofitted with many useful features available in MAD-8 era SIF (ie, `CALL, filename = "whatever.xsif"`)
  - **All the files in the ILC EDR lattice directory are XSIF**
- Most applications can read XSIF
  - **Some exceptions**
    - SAD

Drawbacks with XSIF

- XSIF can't adequately capture the complexity of a modern accelerator complex
  - **No notation for wraparound elements**
    - Needed for IR, low energy acceleration sections
  - **No notation for pulsed extraction lines branching off the main line**
  - **No notation for capturing which paths through the complex are allowed and which are forbidden**
  - **No notational means for describing beamlines that do complicated transport**
    - Example: SLAC linac, which transports e- and e+ in same direction, so focusing and steering magnets have opposite polarities for the 2 beams
  - **Capacity for describing field maps, wake fields, and other important effects limited or nonexistent**
- Tools for translating to other important formats limited

Drawbacks with XSIF (2)

- Subtle compatibility problems with MAD-8
  - **MAD-8 uses SIF to process application commands as well as lattice description**
    - Have to strip MAD-8 commands out of SIF before posting
      - …which we sometimes forget to do!
  - **MAD-8 does not support arbitrary coordinate transforms**
    - Makes injection / extraction line description error-prone
    - Complicates description of vertical curvature in linac
- XSIF is old!
  - **Originally described in a 1983 paper**
  - **Codebase of XSIF library probably goes back almost that far**
  - **Library has become a huge plate of FORTRAN spaghetti**
  - **Lots of modern functionality (ie, default directory specification) nonexistent**
- SIF / XSIF parsers all written and maintained by physicists
  - **Takes away time we could use for doing physics, working on physics application software, etc**

Accelerator Markup Language (AML)

- Project led by David Sagan (Cornell)
  - **Develop a new standard for accelerator description**
    - Address shortcomings in XSIF
    - Ease maintenance / leverage modern software developments
    - Truly independent of any particular application program
- Based on Extensible Markup Language (XML)
  - **HTML-like syntax, “look and feel”**
  - **Lots of widely-available XML parsers which can easily be used to parse AML**
  - **Quite easy to add features without touching the code itself**
- Incorporates lessons learned from 20+ years of accelerator design since SIF first appeared
  - **IE, developers asked themselves, “How do we represent the trickiest features of existing accelerators?”**

AML (2)

- AML does address the issues with XSIF brought up earlier in this talk
- Other refinements
  - **Straightforward solution to the "magnet family" problem**
    - IE, "I want to define this quad once, but then when I generate the full lattice I want every instance to be unique and to have a unique name!"
  - **Capacity to capture engineering details**
    - Sophisticated documentation capacity for components
    - Representation of magnets powered in series
    - Representation of RF cavities excited by common klystron
    - Representation of common support girders (with or without movers) for beamline components
  - **Other things I've doubtless forgotten to mention!**

AML Look and Feel

AML looks like HTML. Here's a FODO lattice in XSIF and in AML

XSIF:

```
QF : quad, L = 1.0, k1 = 0.55
QD : QF, k1 = -QF[k1]
DR : drift, L = 32.4

CELL : line = (QF, DR, QD, DR)
FODO : line = (100 * CELL)
```

Note: the one major drawback to AML as compared to XSIF is that XSIF is generally more compact.

AML:

```
<element name = "QF" >
   <length design = "1.0" />
   <quadrupole>
      <k design = "0.55" />
   </quadrupole>
</element>
<element name = "QD" inherit = "QF" />
<set attribute = "QD[quadrupole:k]"
        value = "-QF[quadrupole:k]" />
<element name = "DR">
   <length design = "32.4" />
</element>

<sector name = "cell">
   <element ref = "QF" />
   <element ref = "DR" />
   <element ref = "QD" />
   <element ref = "DR" />
</sector>

<sector name = "fodo">
   <sector ref = "cell" repeat = "100" />
</sector>
```

Universal Accelerator Parser

- Once you have a language specification, the next job on the agenda is a parser
  - **Standalone library**
    - Linkable by any application program
  - **Read in AML and generate a data structure which represents the beamline**
    - Can be used by application or translated to the application's internal data structure
- Sagan and co. went further
  - **Wrote a standalone library for parsing AML**
  - **Can also parse other formats (BMAD, SIF)**
    - Generates same data structure
  - **Can write the formats it reads**
    - Thus translating any format to any other
- *Universal Accelerator Parser (UAP)*

AML Status

- The current AML *draft* standard is available

http://www.lepp.cornell.edu/~dcs/aml/AMLUAP/doc_repository/aml_doc-0.48.pdf

  - **Still undergoing modifications and clarifications**
- Cornell plans to use AML for their ERL project
- ATF / ATF2 “flight simulator” collaboration tentatively selected AML as their lattice interchange format
  - **Applications**
    - Lucretia
    - PLACET

AML and the ILC ED Phase

- Interest in using AML as the official ILC lattice description format
  - **Features for managing complex beamlines**
  - **Capacity for capturing engineering information**
    - As the ED progresses, would like to use the lattices as the “official” data source for as much of the ILC as possible
      - Beamline components and CFS layout
      - PS, klystron, girder, mover information
      - Other tunnel hardware (pumps? Radiation monitors?)
      - “Formal device names”
      - Connection to other engineering data (drawings, specifications, etc) via advanced documentation options in AML
  - **Provides a true standard for the ILC lattice descriptions**
  - **True independence from any particular application**
  - **Easier to maintain and expand the parser**
    - “You don’t really expect me to keep debugging this 30 year old FORTRAN-77 codebase, do you?”
- Supported at the highest levels of the ILC
  - **IE, by Nick Walker**

What's the Plan?

NOTE: This plan has not been approved by anybody!

- Near term: first 6 months of 2008
  - ***Get GDE EC approval of this plan!***
  - **Finalize and review the AML standard**
  - **Develop an ILC coding standard for use with AML**
    - Emphasis on readability, ease of maintenance of the deck files
  - **Add SAD parser to UAP**
  - **Complete first pass of XSIF lattices for ED phase**
  - **Bring XSIF lattices into compliance with XSIF coding standards**
    - A topic for another meeting!

The Plan (2)

- Summer of 2008
  - **Roll out AML duplicates of existing XSIF lattices and AML lattice file tree structure**
    - At this point two ~equivalent versions of the lattice files exist – one in XSIF, one in AML
      - AML lattices do not yet include any of the advanced engineering features supported by the standard
- Fall 2008 – Spring 2010
  - **Maintain duplicate lattice tree structures**
  - **Gradually add engineering info to AML version**
    - As this progresses, it will be harder to regenerate AML version of the lattice from XSIF after changing the XSIF
  - **Implement AML support into applications**
    - AT, BDSIM, BMAD, Lucretia, MAD-8, Merlin, PLACET, SAD, …

The Plan (3)

- Summer 2010
  - **AML version of the lattices becomes the *only* official version supported by ILC**
    - Probably at about the time the EDR is relased
  - **What happens if somebody wants an XSIF version of one of the lattices after that date?**
    - They can use UAP to translate AML to XSIF
      - Same argument for other alternate languages
  - **Given existence of UAP, why not keep multiple versions updated indefinitely?**
    - Sure to fail – eventually the versions *will* get out of synchronization
    - AML version includes engineering details – at some point, when you mess around with the lattice you *should* have to confront engineering issues implied by those changes!

Some Final Remarks

- Everyone recognizes that changing language formats is *a big deal*
  - **Only happens about once per generation**
- Nothing in this presentation represents a final decision by anybody
- Acceptance of AML by applications developers and users (ie, the people at this meeting) is recognized as crucial

- I was originally skeptical of AML concept
- After studying the issue, I am completely convinced that migration to AML is the right thing to do

Comments / Questions

"I'm talking darkest night, a shoddy simulation of paradise…"

-Machines of Loving Grace