In Place of a summary...

Ties Behnke, DESY

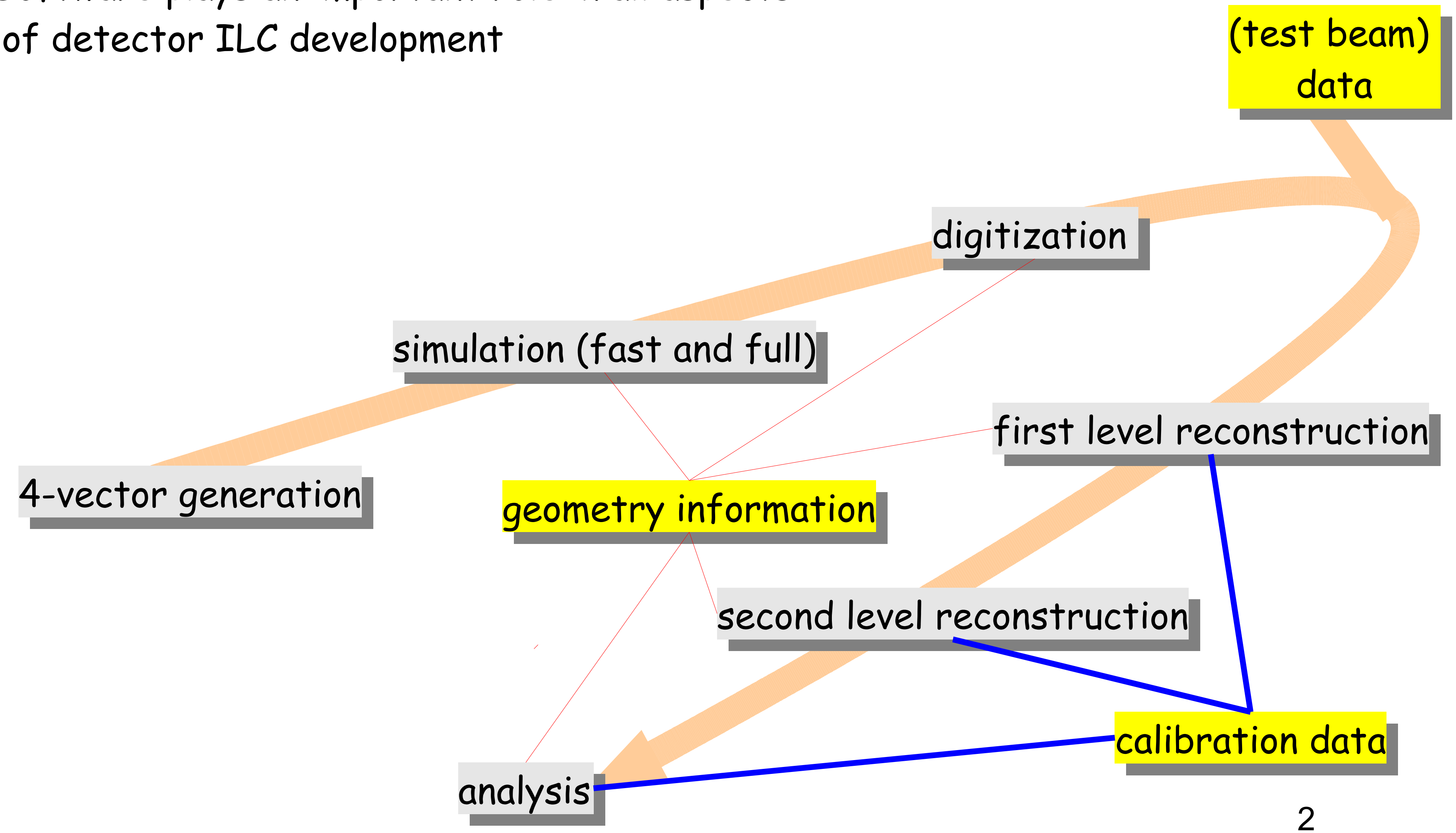
Mark: will talk about particle flow and the future of PFA developments

TB: will bring back up a number of points which have been raised during the meeting

This is a discussion meeting: please interrupt at any time with comments / questions / remarks

Software for the ILC

Software plays an important role in all aspects of detector ILC development



Software in the C++ world

Core software packages:

- MOKKA
- LCIO
- MARLIN
- GEAR
- CED (optional)
- LCCD

Applications
- processors in MARLIN
  - MARLIN Reco
  - MAGIC
  - WOLF
  - PandoraPFA
  - ....

Dependent packages (not ILC)

- GEANT4
- mysql
- xml

- CLHEP
- Root
- AIDA
- OpenGL
- others

The installation problem

The structure of the software becomes more and more complicated

Installation is not trivial

How to proceed:

- Try to automatize the system (-> proposal by Goetz)
  potentially very user friendly
  but puts lots of the burden on the developer
  question: can we maintain such a system?

- Try to be as simple as possible: make files + documentation

The best way: not clear.

LCIO

LCIO is the basic persistency model

C++ framework	Fortran
JAVA framework	Phyton binding

Asian framework at least can write LCIO (read??)

4th concept is orthogonal: can we do something about this?

LCIO experts: Frank Gaede, Tony Johnson

The future of LCIO

The “trivial” task: continue the development of LCIO

The next steps: improve the support for real data (test beam efforts)

Make LCIO more efficient (important already for test beam efforts)

The non-so trivial task:

move towards a true data model for the ILC
which will be used as a transient data model as well

Ease of portability and compatibility

LCIO Questions

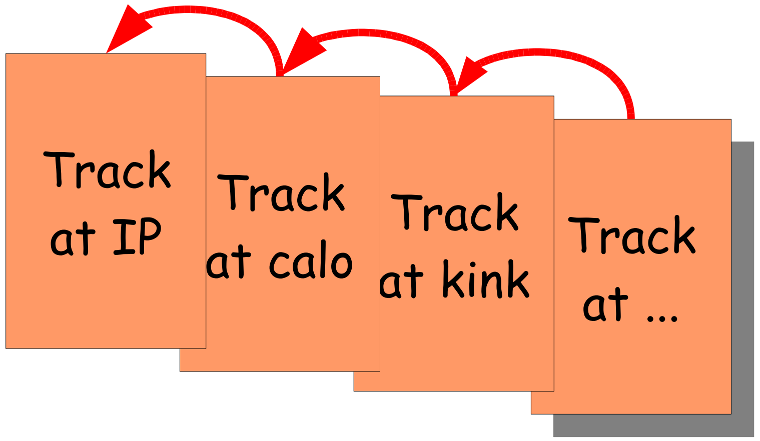
Some concrete questions raised by Frank:

Further development of "reconstructed" entries like tracks:

how do we add information to these?

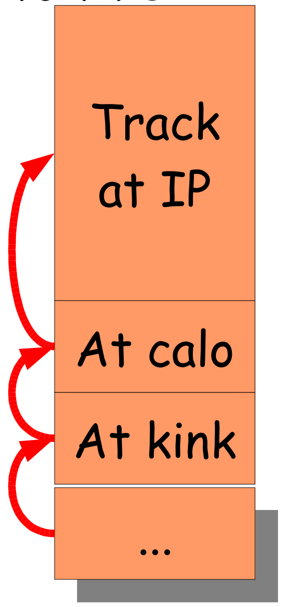
Current LCIO concept:

Keep the objects small and generic
put additional information into
new collections of the same objects



Alternative scheme

Add more information to th e object: the object grows during its lifetime



LCIO Version 2?

LCIO overall has been a success story for the ILC

At some point though we should step back and re-evaluate things:
time for LCIO Version 2, the next major release?

When?
What should be significantly improved?
What is really missing?
Can we improve the multi-language support for LCIO?
.. probably many more questions ...

MOKKA

Support from LLR is quite good for MOKKA kernel

But: closer collaboration with SLIC e.g could save lots of effort

(example: treatment of MCparticle, backscattering, etc etc)

At least we should do in-depth comparisons between SLIC and MOKKA

Biggest problem in MOKKAL:

we need better drivers for the sub-detectors

- Improved representation in the database (more structured)
- More precise geometries
- Natively scalable drivers (concept of superdrivers is rather complicated)

The GRID

LHC computing: depends heavily on the GRID

For us: there is no real alternative to the GRID

- Powerful data manager
- Powerful “batch system”
- Can use empty CPU cycles at many places: very efficient to get larger resources for ILC work

We know:

at the moment it is not user friendly
you have to get a certificate
at the moment it only works under linux
the control language is painful

But other people will improve all that for us over the next few years

Why the GRID?

Why the GRID already now?

Its mostly a matter or resources

Example: at DESY we have around 10 machines which are dedicated ILC batch machines (outdated in addition)

on the GRID we have around 300 machines (and growing) which are “open” for ILC
a few 1000 if we go beyond DESY

At least in Europe for serious processing of ILC data there is simply no alternative to the GRID!
...end of discussion ...

Languages

MARLIN/ MOKKA/ etc rely on C++

US framework relies on JAVA

In many ways JAVA is the better language

BUT: the LHC is 100% C++: to profit from the experience of the people
and possibly from the software
C++ is still a viable alternative to JAVA

The goal: eventually arrive at a point where switching between C++ and JAVA is possible even within one job
(within one org.lcsim reco job, or within one MARLIN job)

Important: we do not want to be religious about either choice, but be open for future developments

Interfaces

Interfaces play a central role in the software concept

Examples:
LCIO
AIDA
are widely accepted

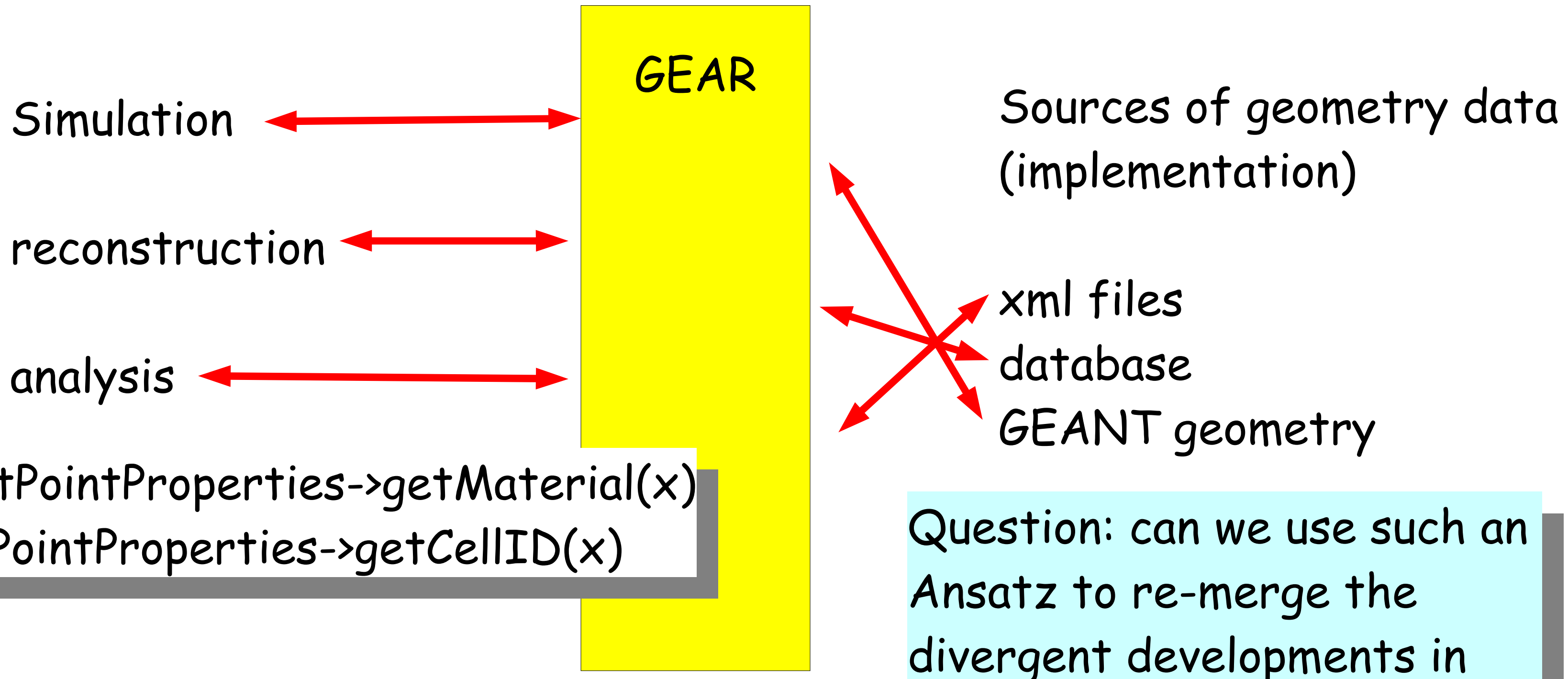
GEAR is an attempt to extend this to the geometry
RAVE is going the same way for vertexing tools

Designing the software around a set of well defined interfaces
- makes software portable, extensible
- and it isolates the user from many technological (and potentially religious) questions.

Interfaces: Example

Geometry information is used in many places

- Very detailed, but local: simulation
- Less detailed, but know surroundings: reconstruction
- Little detail: e.g. Event display



```
Mat=getPointProperties->getMaterial(x)
ID=getPointProperties->getCellID(x)
```

Question: can we use such an Ansatz to re-merge the divergent developments in EU and US?

GEAR

GEAR as an Ansatz looks quite promising

BUT

there are holes and problems:

- Link to MOKKA not yet there (under development, MOKKA driver will produce correct GEAR file
- Urgently needed: implementation of the full GEAR interface (only partially done at the moment)

Ideally: have enough information in the database
to feed the complete GEAR chain (either directly, or
via an automatically created xml file)

The Future of CORE software

The ILC software community is small

The expectations are large

Resources:

- DESY is committed to continue to support core ILC software developments
- EUDET will provide some (small) amount of additional personpower for this
- LLR is committed to continue support for MOKKA

Progress is possible only if:

- we concentrate on the urgent and important tasks at hand
- everyone pulls in the same direction
  - (openeness – standards – responsibility - documentation)

(in my personal opinion: the excellent but insular solution in software is less useful than the well documented, boring solution which respects the standards!)

The Future

Our goal: "lightweight" software: are we still true to this goal?

Lets remember the important NOTs in our community:

we are NOT a collaboration
we do NOT have broad support for the software from professionals
we do NOT have huge resources as the LHC experiments do
we do NOT have system managers/ responsibles dedicated to the ILC
we do NOT in general cater to full time people

This influences the way the software
is conceived
is installed
is run and used

Missing Applications

Lots of applications (processors) are missing, incomplete, non optimized

(see Marks talk as well)

Particularly important

- Better tracking (particular forward)
- Vertexing tools (Rutherford, Vienna, others?)
- Lots of tools and helpers

We need people to contribute to the pool of tools
make your work widely available!

Cooperation

At Bangalore I said:

We are dublicating efforts on 50% of the needed functionality

and never get around to attack the other 50% because of lack of personpower

My pledge:

we should work together more closely
we should try to bridge the Atlantic more efficiently
we should try to learn from each other

We will start to compete for the IP's early enough
there is no need to start this already now.

Summary of the non-Summary

We need to find a proper forum to keep discussing open questions

and to come to some sort of decisions

Linear collider forum
simulation meetings like this one
phone conferences etc?

Lots of good presentations and discussions in this meeting

When shall we have the next simulation meeting? Where?