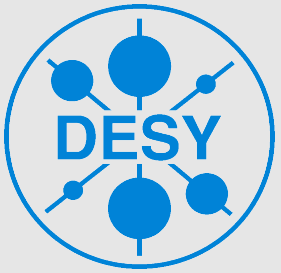


A MERLIN-Based Start-to-End Simulations for the ILC

Dirk Kruecker,
Freddy Poirier, Nicholas John Walker

Outline of this talk:

- Goals, troubles and Software Design
- First Results

Goal

A ”start-to-end” simulation at least RTML-BDS

- For a start ML-BDS
- Ground motion modelling ⇨ BOTH accelerator sides
  - ATL, Seryi ABC to investigate correlated GM
- Support structures for cryomodules, final focus system etc.
- Modelling of steering, tuning, feedbacks
- Total X-section e^+e^-
  - Guineapig (Daniel Schulte)
- ROOT output
- ....

MERLIN Basics

MERLIN is a C++ library developed by N. Walker and A. Wolski and several other people

about 42000 lines of code

Main classes*

- **AcceleratorModel** ← constructed by **XTFFInterface** from XTFF lattice file ~ MAD8 twiss output
- **SupportStructure** ← a handle to move acc. components for ground motion etc. and a way to group elements (i.e. cryomodules)
- **BeamLine** ← seq. of accelerator components
- **Channels** ← e.g. BPMs and correctors
- **Tracker** ← tracks a bunch through a beam line
- **Bunch** ← ParticleBunch / SMPBunch

* do not take any class description in this talk literally everything is simplified for the sake of clarity

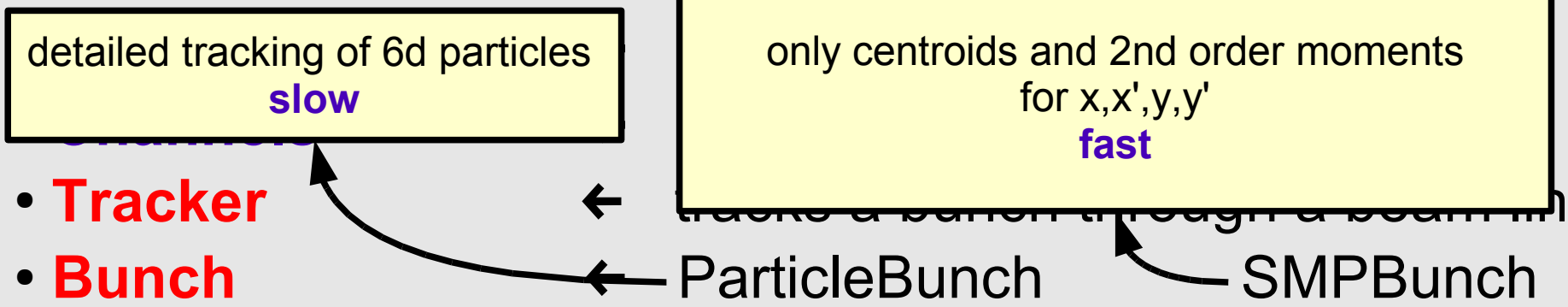
MERLIN Basics

MERLIN is a C++ library developed by N. Walker and A. Wolski and several other people

about 42000 lines of code

Main objects (simplified* class structure)

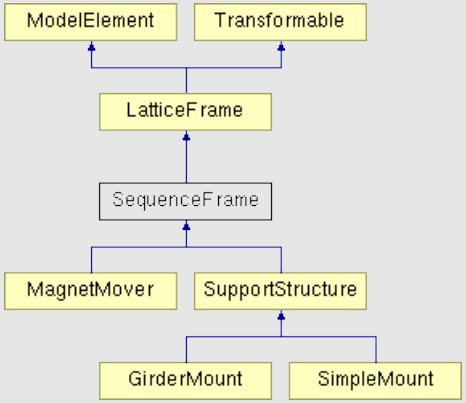
- **AcceleratorModel** ← constructed by **XTFFInterface** from XTFF lattice file ~ MAD8 twiss output
- **SupportStructure** ← a handle to move acc. components for ground motion etc. and a way to group elements (i.e. cryomodules)



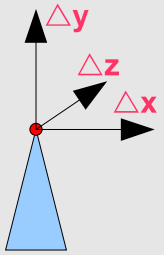
* do not take any class description in this talk literally
everything is simplified for the sake of clarity

SupportStructure

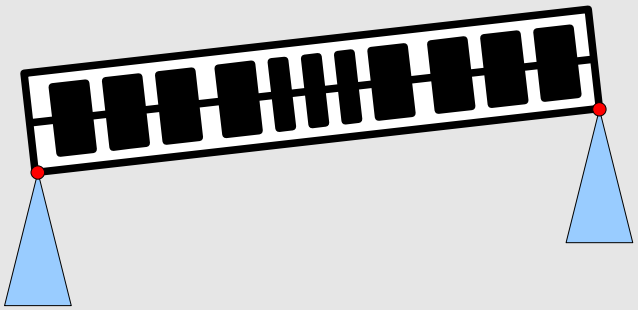
A second layer on top of the geometry of an accelerator element.



- **SimpleMount**
  a point in space
  that can be move



- **GirderMount**
  2 points enclosing several accelerator elements



- implemented in MERLIN for ground motion on top of the geometry

not a MAD type but represented conventionally by **MARK**ers in the XTFF file e.g crymodules

A First Try

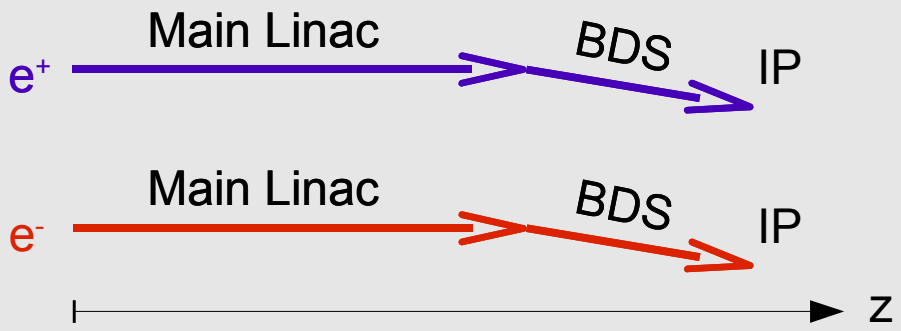
2 **AcceleratorModel** (electron + positron) same lattice file for both sides
  - constructed with from the lattice file we used last year

  (e.g. Failure mode studies and ILCDFS) : ilc_linac_15_250.xtff (?)
  - Separate file for the BDS layout
- 2 tracker for each model = 4 in total
  - **SMPTracking** in ML / **ParticleTracking** in BDS *
  - a converter **SMPBunch**⇔ **ParticleBunch**
- An interface and a wrapper to use code from Daniel Schulte for groundmotion according to models ABC
- Separate One2One steering in ML and BDS
- A simple FFB: fix centroids x,x',y,y' of a bunch at begin of the BDS
- All split magnets on BDS (e.g. Q1-BPM-Cor-Q2) and cryomodules on girders

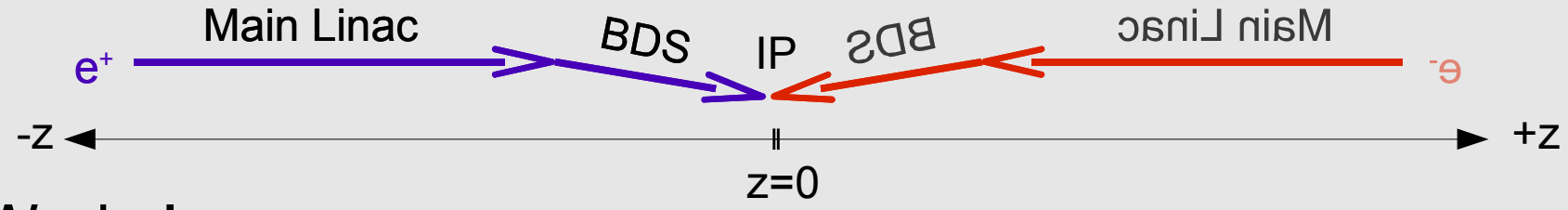
*for the full simulation the gain in time is about a factor of 5

How to collide?

- after construction we have 2 independent models both starting at z=0



- the IP in both accelerator models must be the same point in space
- the concept of supports is used move and reflect the accelerators (the relative position is only relevant for GM)



- Works!

Some *minor* changes ...

So far we had a reasonable concept but we should use the latest ILC lattice files from Mark Woodley's web page http://www.slac.stanford.edu/~mdw/ILC/ :

- ILC2006c:
  Slightly different naming convention in new files
  Changes in naming convention produce two kind of problems;
  - parser (XTFFInterface) does not work (position of girders not recognized)
  - simulation code cannot access elements by name (e.g. BPMs)

- ILC2006e:
  Slightly different naming convention in new files
  - Undulator appears
    only partly implemeted: bypass and drifts for undulator cell
    second half of ML: additional string to compensate for the energy loss
    -> electron side is clearly different now from positron side
    but no lattice for positron ML

Lattice files

- Lattice file contains in addition to the standard MAD types MARKers for girder, supports.

Original lattice file convention (as implemented in the present Merlin lib)
(ilc_linac_15_250.xtff – 20389 elements ML + separate BDS file)
MARKG_CM – MARKG_CM (girder for cryo modules)
...
MARKVPIV (kicker to follow earth curvature)
VKICYCOR / MONIBPM

ILC2006c (42634 elements ML-BDS)
girders
MARKBEGMLCM - MARKENDMLCM
Markers(?) for correctors and BPMs in ML
MLXCOR MLYCOR MLBPM

ILC2006e (46778 lelements ML-BDS)
MARKBEGMLCM – MARKENDMLCM + MARKBEGMLQ - MARKENDMLQ
ML BPMs and correctors back as corresponding MAD types
electron undulator appears but no bpms and correctors
added my own: MARKUNBPM MARKUNXCOR MARKUNYCOR
MARKBEGUNCM MARKENDUNCM
magnet mover in BDS only a few correctors

and other differences ...

Difficulties

- syntax parsing in the original **XTFFInterface** turned out to be inflexible
- Handling of BPMs/correctors and supports depends on naming
- Extra BPMs/correctors needs modification to MAD files
- a new element undulator (additional energy smearing for SMPTracking)
- no MULTIpole element in Merlin
- Particle tracking in undulator : 4 -> 7 separate trackers
- ...

Handling of code became difficult!

Wish list

- The code has to be flexible
  - Keep identifiers (names) in one place (if possible)

  the lattice files defines a vocabulary that is used in different places of the simulation code. Changes in an evolving ILC cannot be avoided
- Use a generic approach to build Bunches and Tracker

  more general: generic SubSystems
- debugging, x-checks:
  - Ability to change easily between Particle and SMP
  - Slice the xtff input file to run only a part of the model
  - Ability to define additional girders/BPMs/Correctors

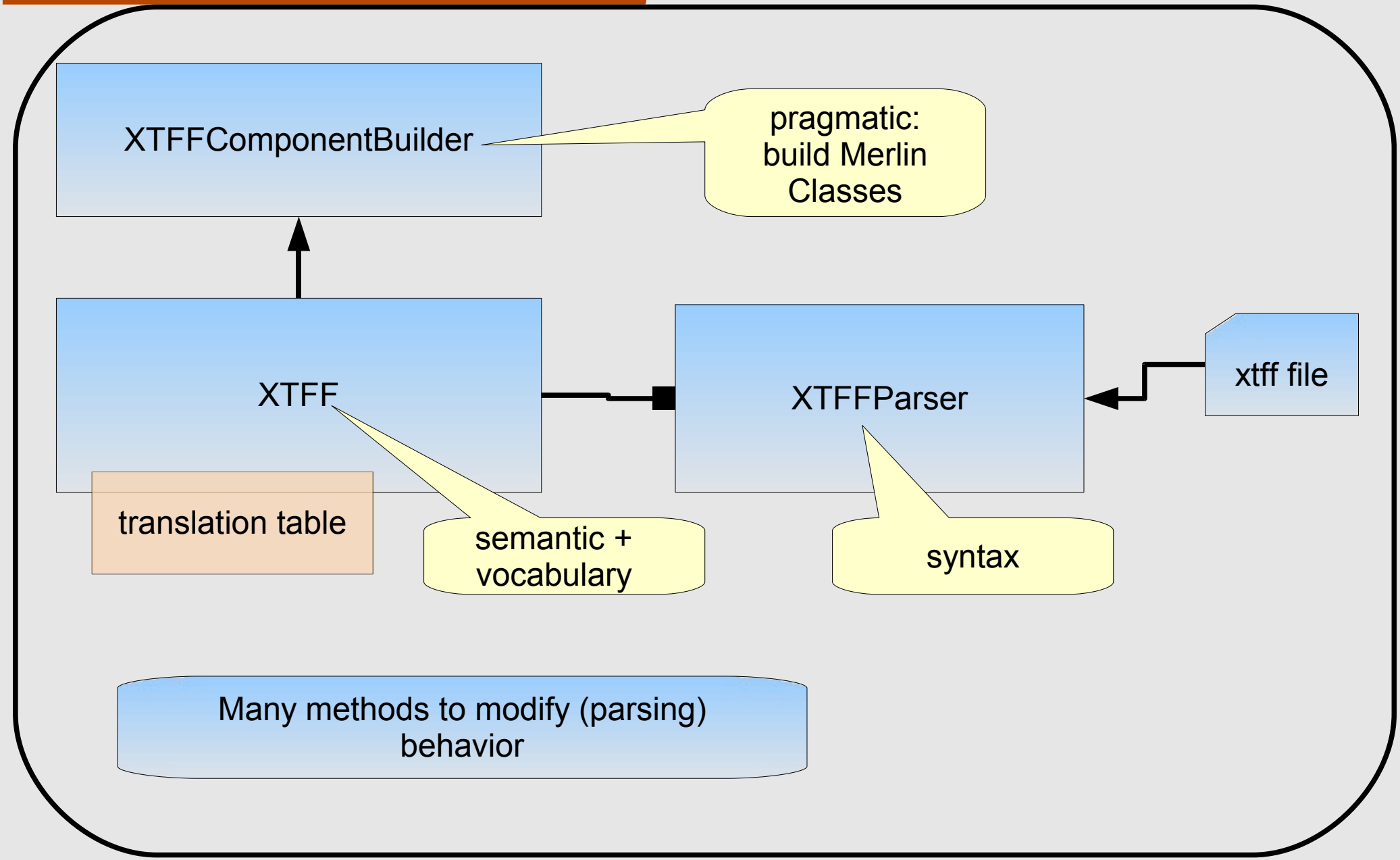
  (without changing the input (mad) files)

New parser

Generic code to model subsystems:

Accelerator & SubSystem

New XTFFInterface



New Functionality

- model from only part of XTFF files
  - Stepwise construction of AcceleratorModel
  - Additional girders (e.g. final focus) by name or z-position
  - Elements can be forced to be on supports
    - WARNINGS if active element is not on a support
  - High Order Magnets on/off
  - ...

Accelerator

simplified

loop:
eSubs & pSubs

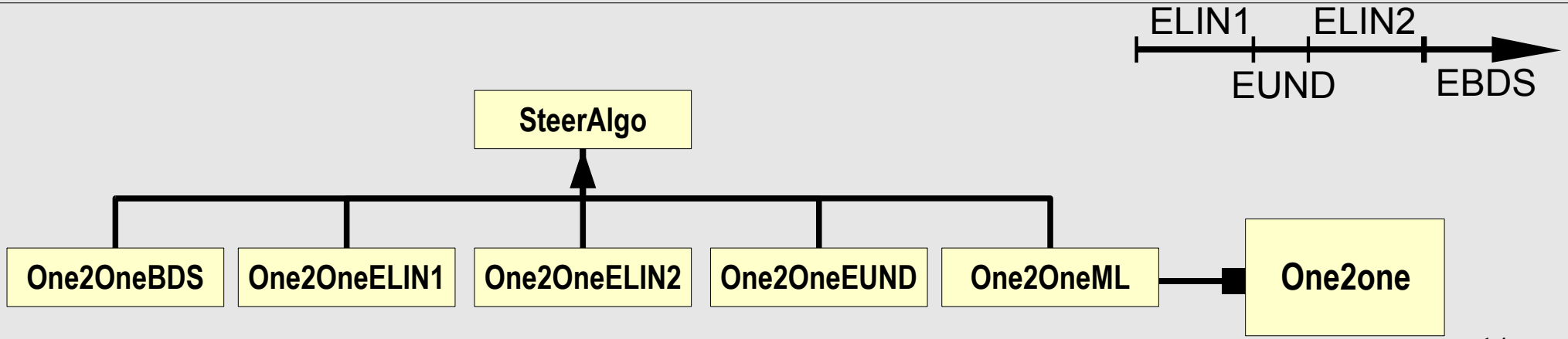
Accelerator

void **Track**()
void **Steer**()
AcceleratorSupportList* **CreateSupportslist**()
...

loop:
steering algorithms

vector< SubSystem* > eSubs
vector< SubSystem* > pSubs
vector< SteerAlgo* > eSteer
vector< SteerAlgo* > pSteer
...

export
list of support



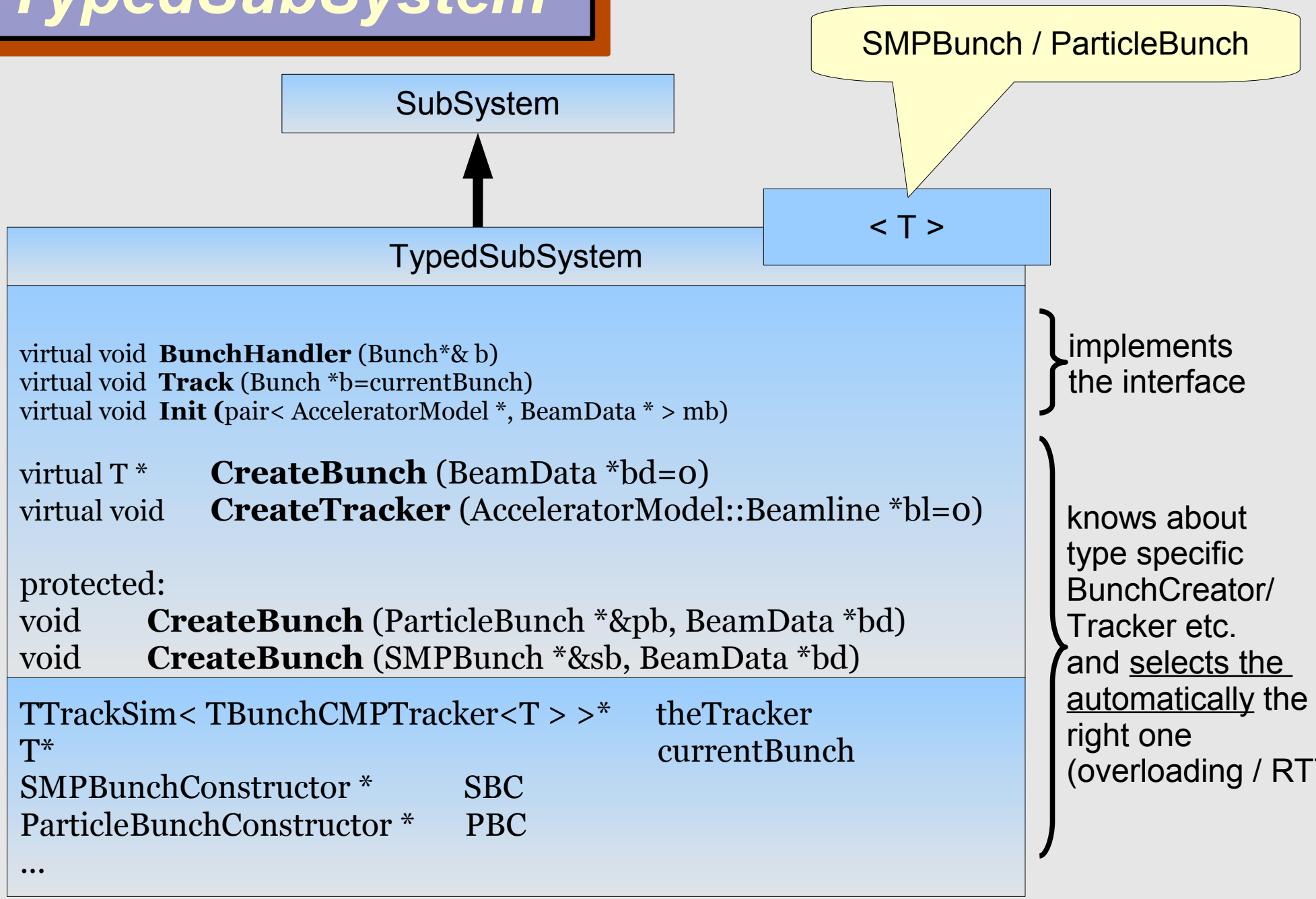
SubSystem

SubSystem
virtual void **BunchHandler** (Bunch*& b) virtual void **Track** (Bunch *b=currentBunch) virtual void **Init (**pair< AcceleratorModel *, BeamData * > mb) ...
string begMark string endMark AcceleratorModel* accMod AcceleratorModel::Beamline bline BeamData* bdat ...

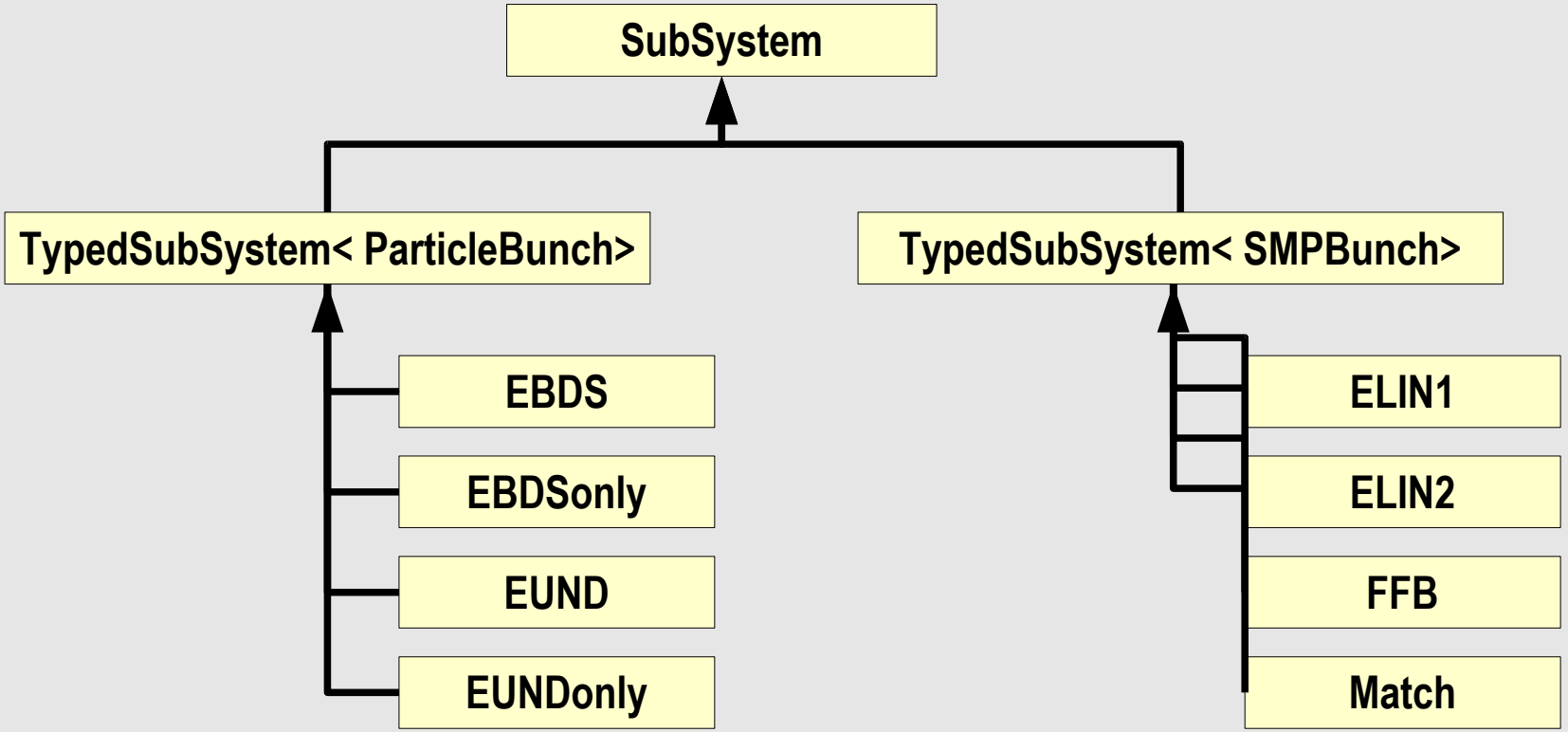
Interface allows a list of **SubSystem**s in **Accelerator**

loop on subsystems: BunchHandler(theBunch) <- create/pass/convert the bunch
Track(theBunch)

TypedSubSystem



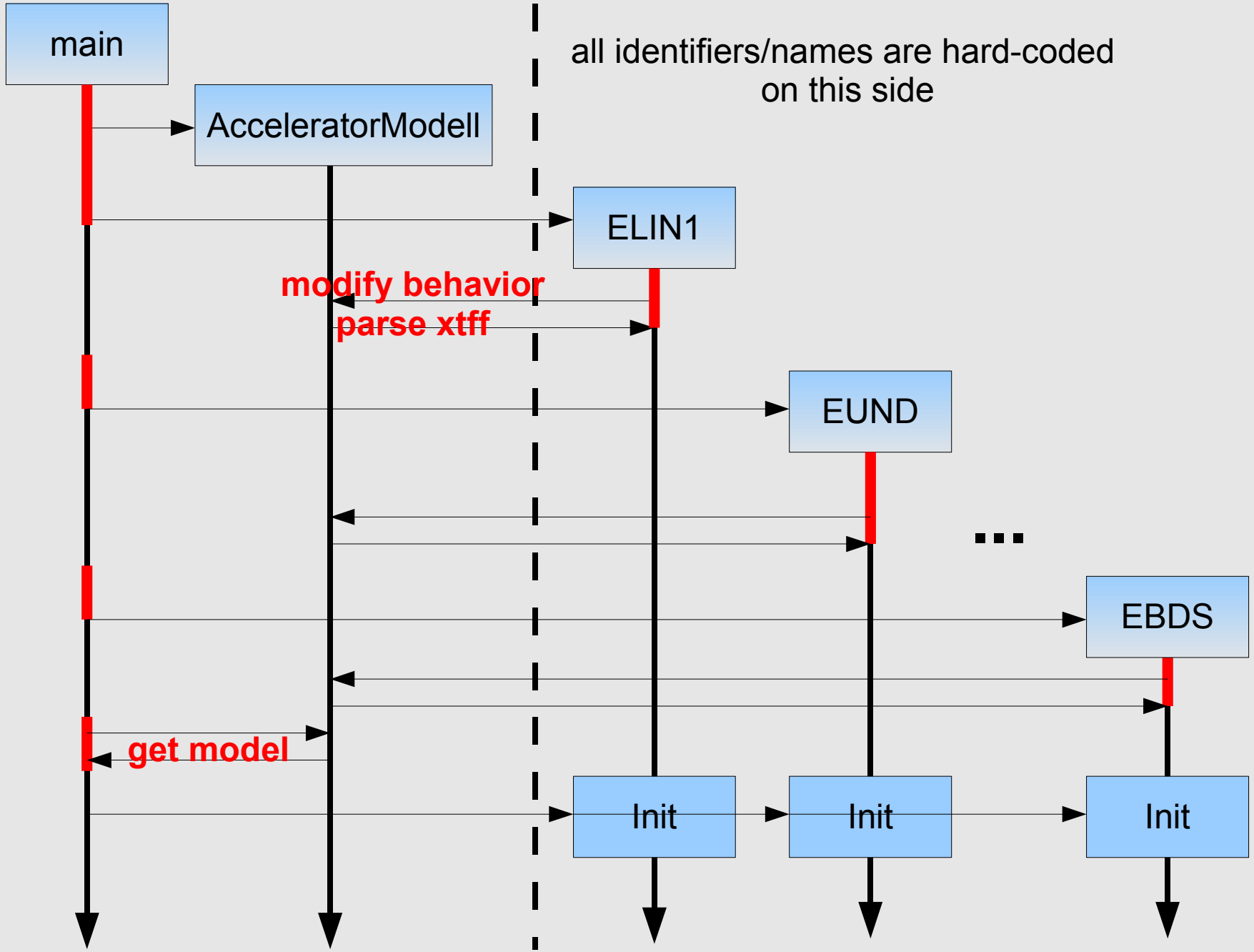
concrete SubSystems



for testing useful

- each subsystem can easily be used with SMPTracking and ParticleTracking
- only small modifications to run a subsystem in stand alone mode (e.g. EBDSonly)

Building the Model



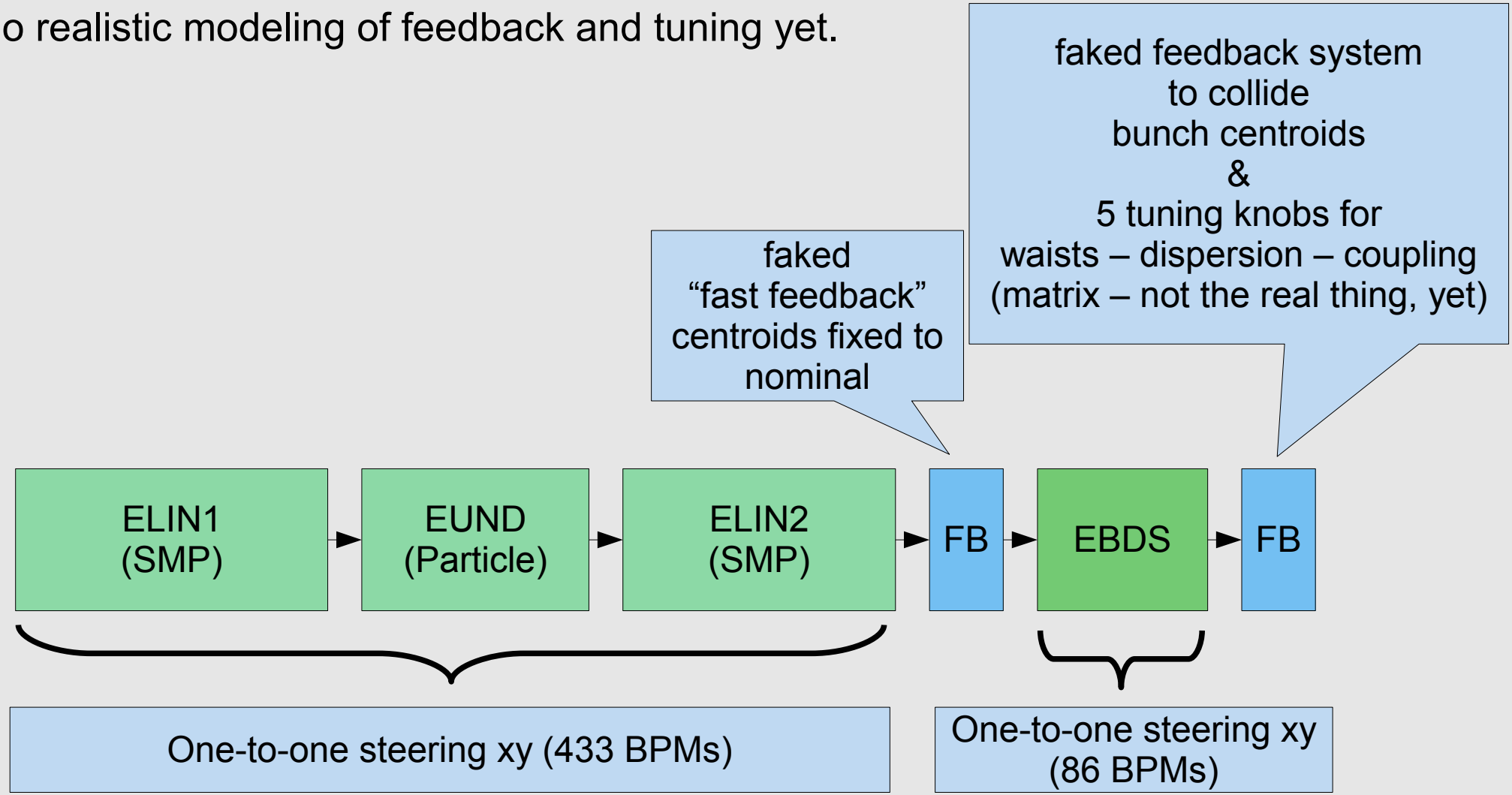
If you have got the impression that my code has become a little complicated :) – the opposite is true e.g BDS:

not simplified

```
class EBDS : public TypedSubSystem< ParticleBunch >
{
public:
        EBDS(XTFF&);
};
//------------------------------ EBDS-----------------------------------
EBDS::EBDS(XTFF& eXTFF){

        name = "EBDS";

        // special markers
        begMark="MARKBEG_EBSY1";
        endMark="MARKEND_EFF1";

        // modify XTFF behavior
        eXTFF.TreatTypeAsDrift("INST");                 // switch of warnings
        eXTFF.ConstructGirdersForSplitMags(begMark,endMark);
        eXTFF.AllSplitQuadsBXY(true);       // additional BPM/XCor/YCor on

        //FINAL FOCUS
        //OC1-SF1-QF1-SD0-OC0-QD0
        eXTFF.AddGirderPair(14745.510,14752.260);

        // append lattice file from-to (inclusively)
        pair<BeamData*,BeamData*> bb =
                eXTFF.AppendModelAB(ModPar::eFileName,begMark,endMark);
        bdat = bb.first;
};
```

Preliminary Model

No realistic modeling of feedback and tuning yet.



First Results – ML

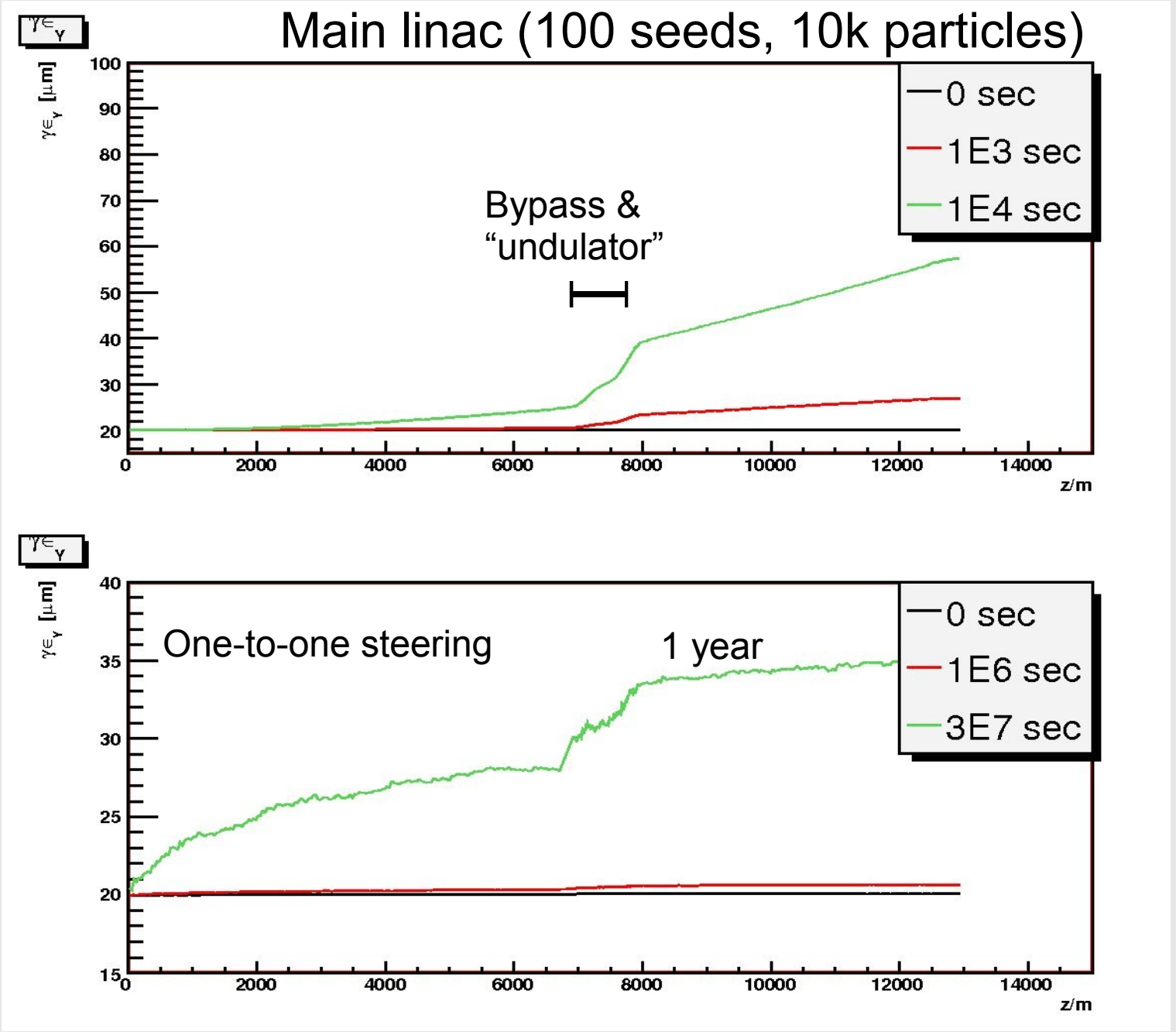
ATL ground motion this plot only y errors

$A = 4 \cdot 10^{-18}$

No further alignment errors

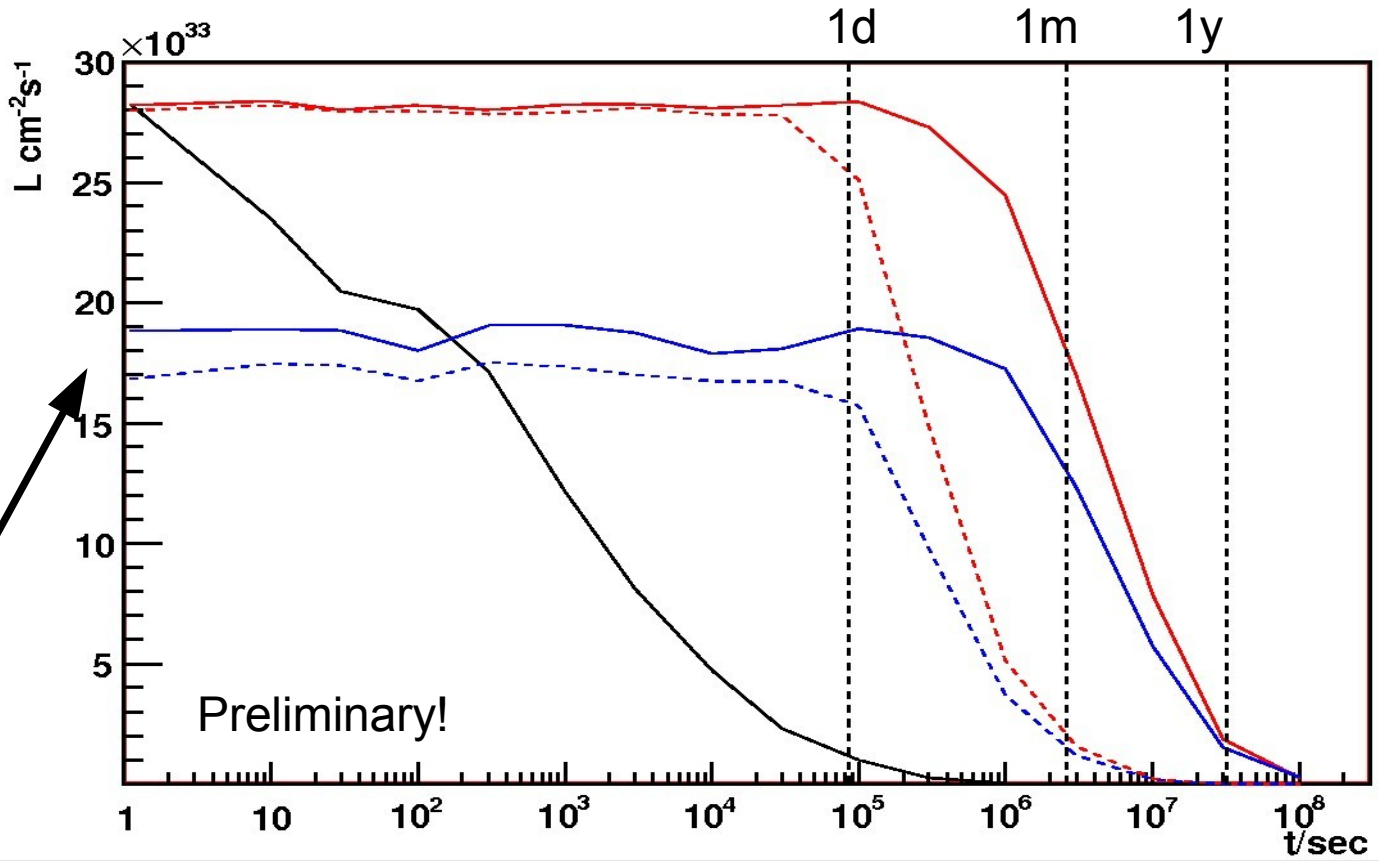
Bypass for undulator seems to be a stability issue

Additional 86 BPMs+YCorrectors in bypass



First Results – ML+BDS stability

- ATL in x and y
- 1-2-1 steering
- 40 collision each point
- 5 Tuning knobs: $w_x, w_y, d_x, d_y, c_{xy}$
- GUNIEAPIG for
- x-section calc.



L about 12% smaller larger $\beta_x\beta_y$ in ILC2006e lattice at IP as in RDR

ATL: A = 410^{-18} m/s - ML-BDS - $\gamma\epsilon_x$= 8, $\gamma\epsilon_y$=0.02 [μm]

- no steering
- 1-2-1
- 1-2-1 + tuned
- noise + 1-2-1
- noise + 1-2-1 + tuned

Noise:
add. Transverse errors, ML 300 nm, BDS 100 nm
approx RDR nominal values: $\gamma\epsilon_x$=10 $\gamma\epsilon_y$ = 0.04 mu

Summary

In the attempt to model the ILC one encounters several difficulties

- Evolving system
- Partial lattice files
- Naming
- ...

Naming conventions ?
Common repository / version management?

We tried to develop a flexible framework

- General solution for technical details (SMP/Particle tracking)
- Modular and and easy to extend
- Reduced dependency on names (vocabulary)
- Interfaces to plug in
  steering algorithms – tuning – feedback systems – ground motion models

...

To Do

- Simulation runs for correlated ground motion models – positron side?
- Realistic errors
- Merlin DFS package
- BDS Tuning studies
- Multi bunch modeling
- RTML

...