



A C++ Class Library
for performing
Charged Particle Accelerator Simulations

Status of Merlin

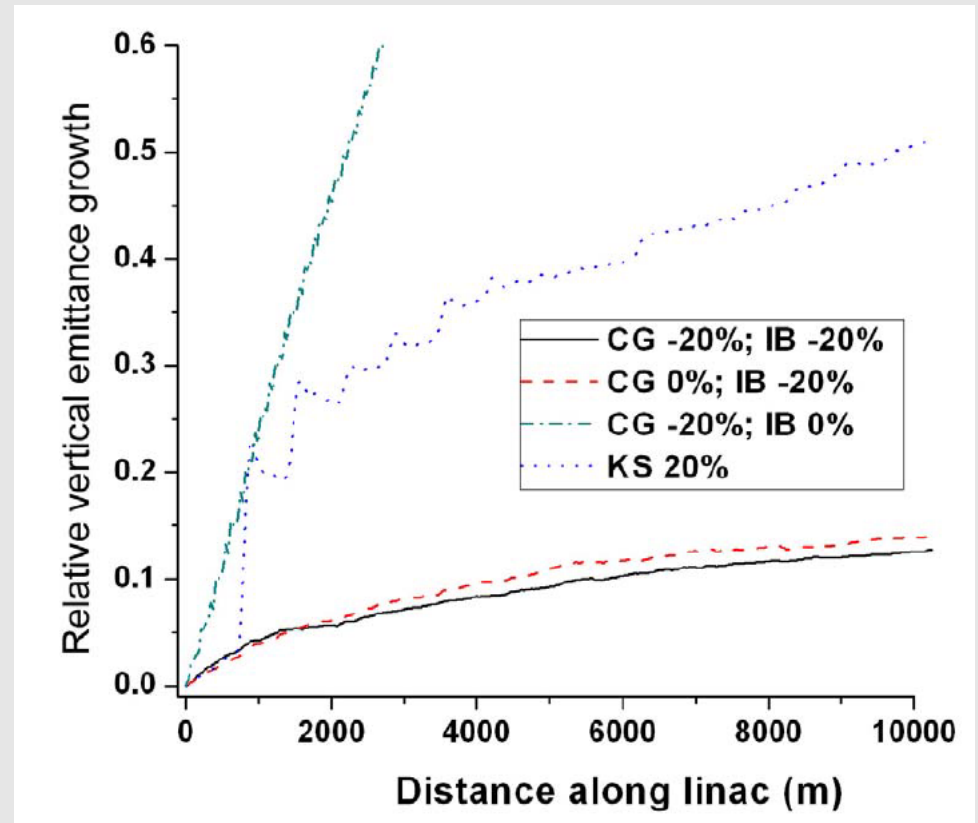
Dirk Krücker - DESY
Uppsala, August 2008

Intro

- MERLIN had been used for many EUROTeV studies.
- New code of general interest has been developed that it worth to become part of the MERLIN library. Other pieces of code are not finalised.
 - DFS
 - Component errors etc.
 - Framework for a Start-to-end Simulation
 - ROOT
 - Wakefields
- New version management at DESY:
CVS to Subversion

New Code – Dispersion Free Steering

- DFS algorithm allows for different energy adjustment policies
 - Constant gradient
 - Initial beam energy
 - Klystron shunting
- Had been used in many studies
- **EUROTeV-Report-2006-106**
- Available from the webpage but depends on development version of MERLIN



New Code – Smaller Contributions

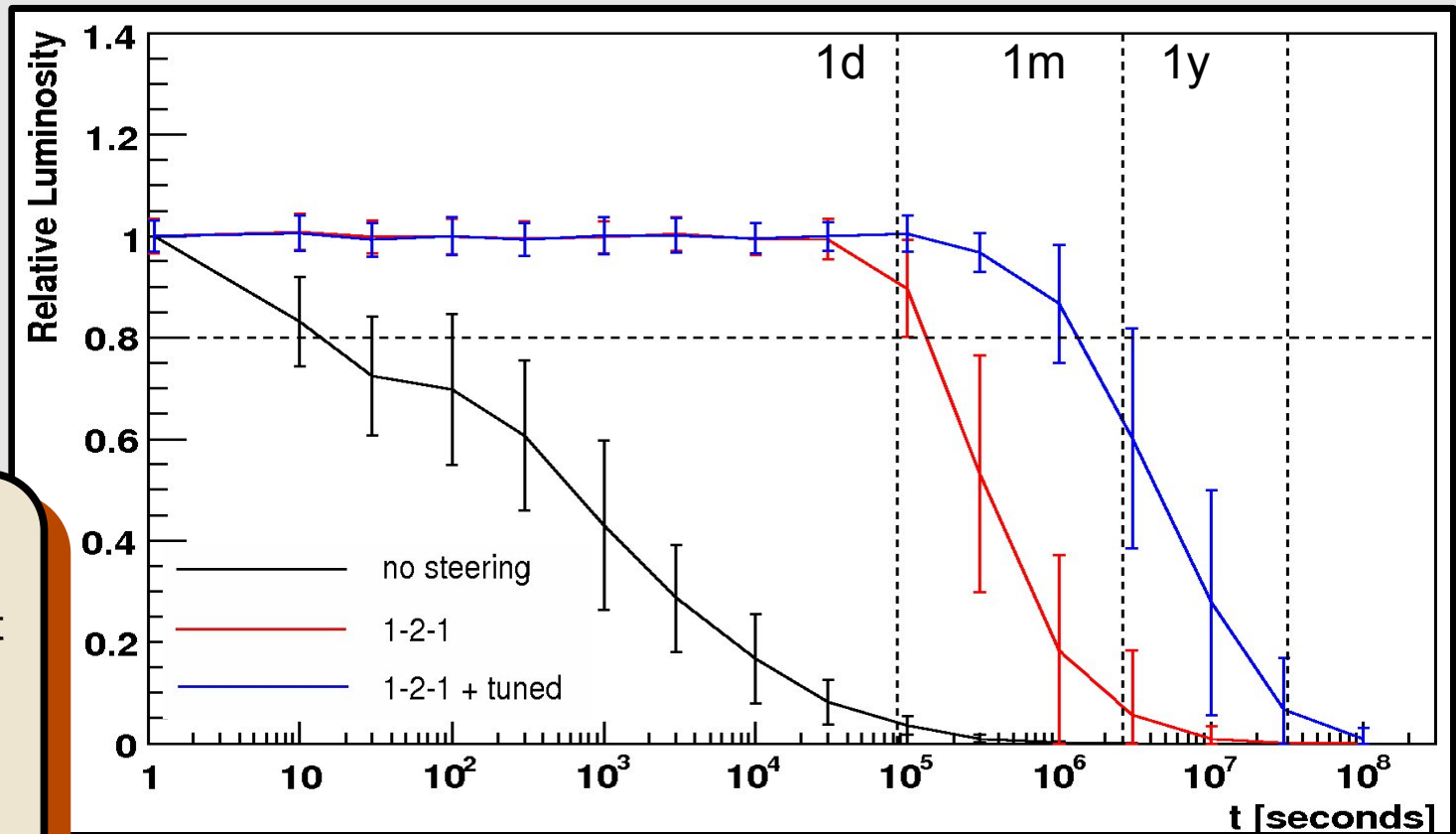
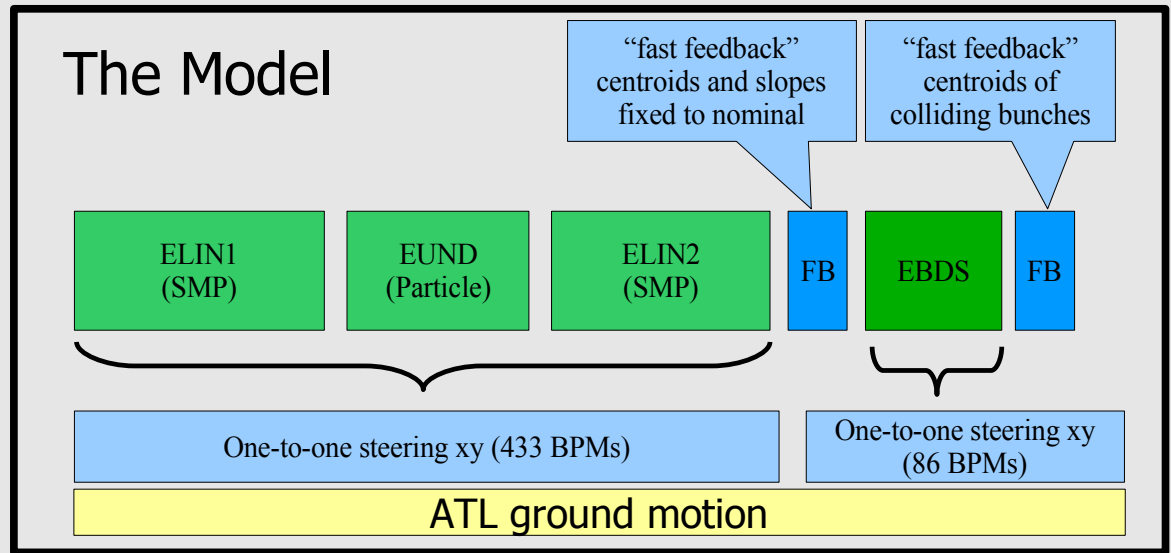
- General classes to simplify the application of component errors
EUROTeV-Report-2007-020 (Evaluation of the Component Tolerances ...)
 - Quadrupole alignment errors, klystron voltage errors etc..
`AddTransverseErrors(20*micrometer, 20*micrometer, bline, "BPM.*");`
- SMPBunch to ParticleBunch converter*
- More ideas around, but not yet in a state to enter the next release
 - Magnetic field errors
 - Seryi model of ground motion
 - Kubo et al., survey line model
 - ...
- An interface to read arbitrary alignment data (see talk by F.Poirier)

*SMP(Sliced Macro Particles) bunch - transverse phase space by 1st and 2nd order moments
Particle bunch – full 6D phase space

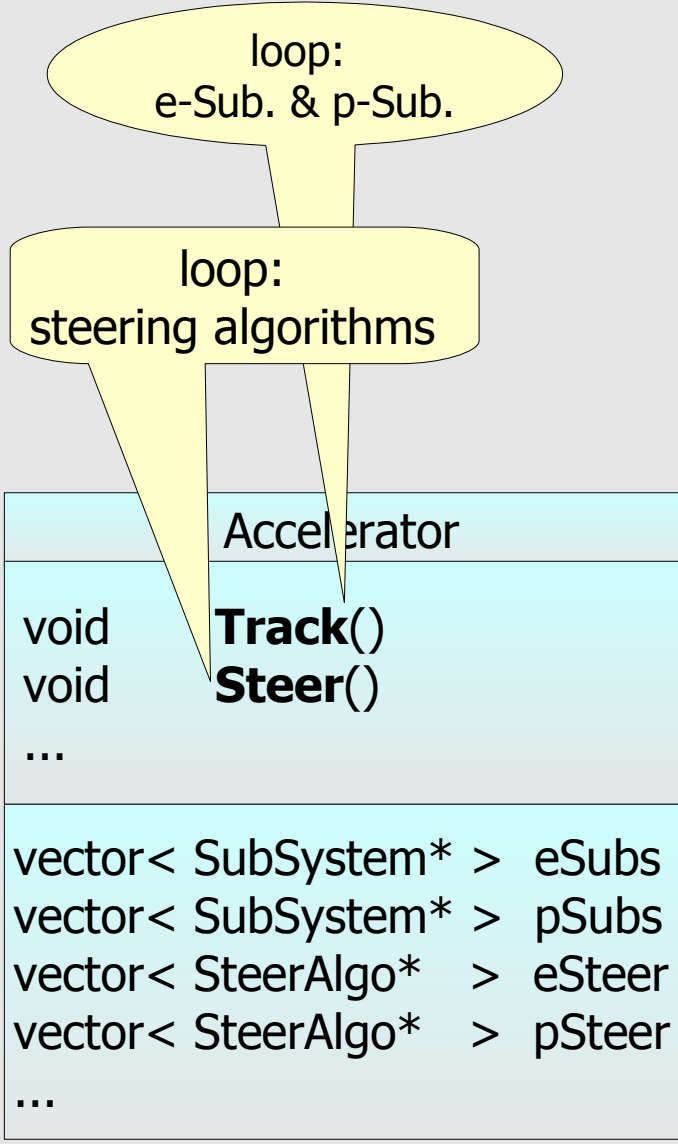
1. Luminosity Stability

- $\gamma\epsilon_x = 10 \mu\text{m}$
- $\gamma\epsilon_y = 0.02 \mu\text{m}$
- ATL in x and y
 $A = 4 \cdot 10^{-18} \text{ m/s}$
- 1-2-1 steering
- Idealistic feedback and beam tuning
- 5 linear tuning knobs
 $w_{x'}, w_{y'}, d_{x'}, d_{y'}, c_{xy}$
- Cross section from GUNIEAPIG with 40 collision / point

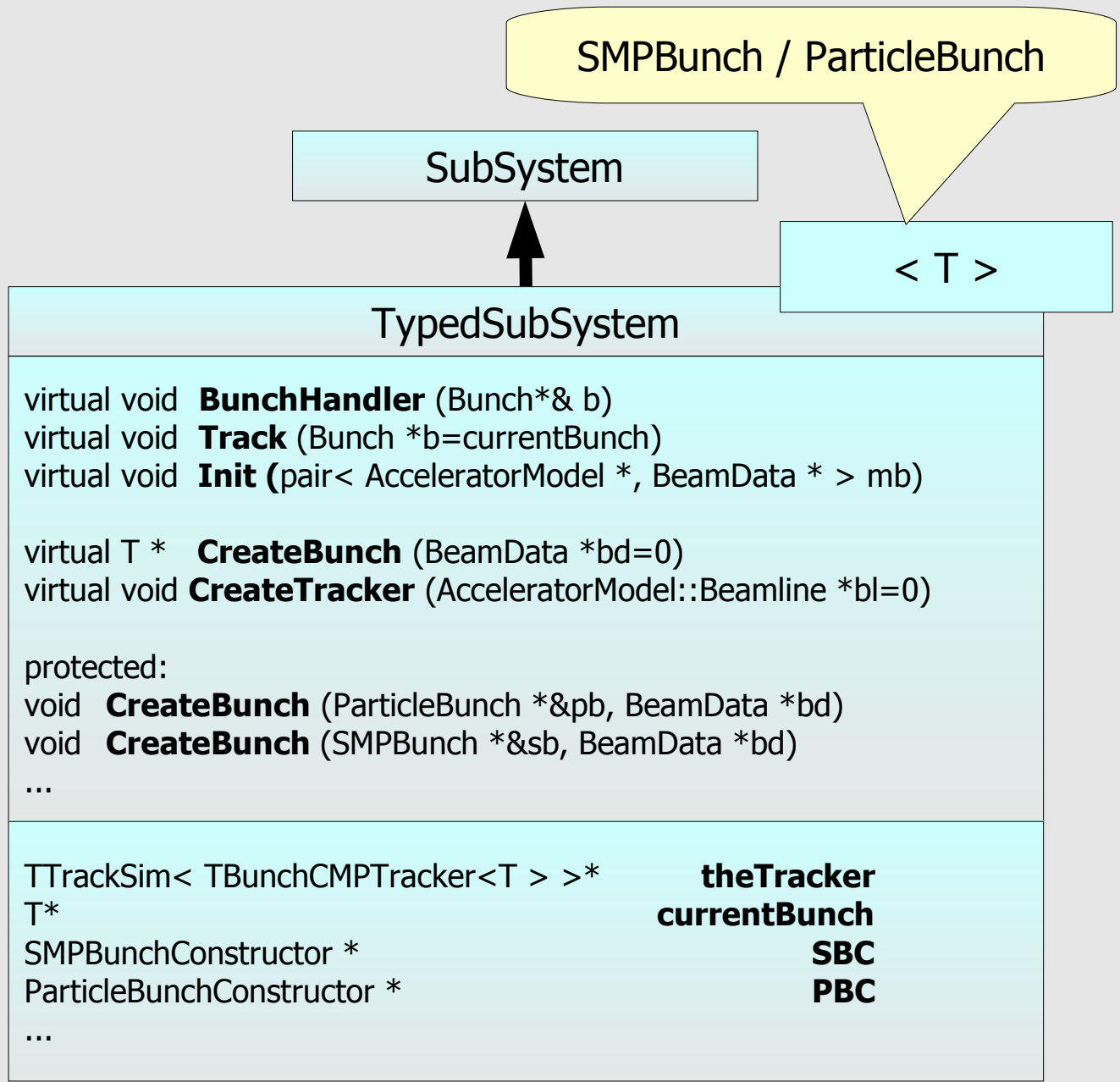
In our model the luminosity can be kept above 80% of the nominal values for about 15 (-7+15) days. Luminosity needs to be re-established then by a re-application of beam-based alignment.



Unfinished new Code - Subsystems Template Class



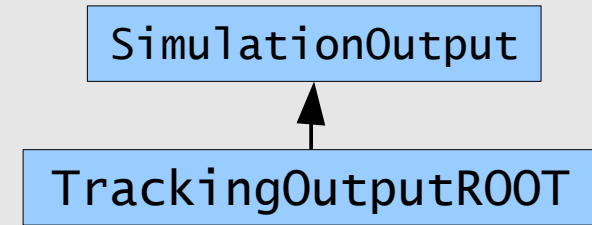
common reference system
for ground motion



- Start to End Simulation – General Framework
 - [EUROTeV-Report-2007-019](#)
 - Template class for a multi system simulation
 - Working
 - New Parser for an easier adaption to changes in lattice files (naming conventions etc.)
 - Turned out to be the the difficult part
 - Postponed for AML integration
 - ROOT interface separated

New Code - ROOT

- Common task for any study
 - Write out beam parameters at component xyz
 - Save bunch particles or lost particles
- ROOT is a widely used tool for analysis
 - Interactive data analysis, visualisation
- A simple interface derived from `SimulationOutput`
 - Modification in `SimulationOutput` for output at position z (MERLIN uses string pattern to identify accelerator components.
 - Names are not always unique.)
- As a MERLIN example i.e. not as part of the library
 - Merlin lib should not depend on ROOT libraries



ROOT Trees

- **Tracking Tree**

$z_{component} \quad p_0$
 $\langle x \rangle \quad \langle y \rangle \quad \langle x' \rangle \quad \langle y' \rangle$
 $\langle ct \rangle \quad \langle dp/p_0 \rangle$

$\sigma_x \quad \sigma_y \quad \sigma_{dp/p_0}$

$\alpha_{x/y} \quad \beta_{x/y} \quad D_{x/y} \quad D'_{x/y}$

$\gamma \epsilon_{x/y} \quad \gamma \epsilon_{x/y}^c \quad BPM \quad YCor \quad V_{cavity} \quad \text{etc.}$

- **Bunch Tree**

SMPBunch and ParticleBunch

$x \quad y \quad x' \quad y'$
 $ct \quad dp/p_0 \quad Q$

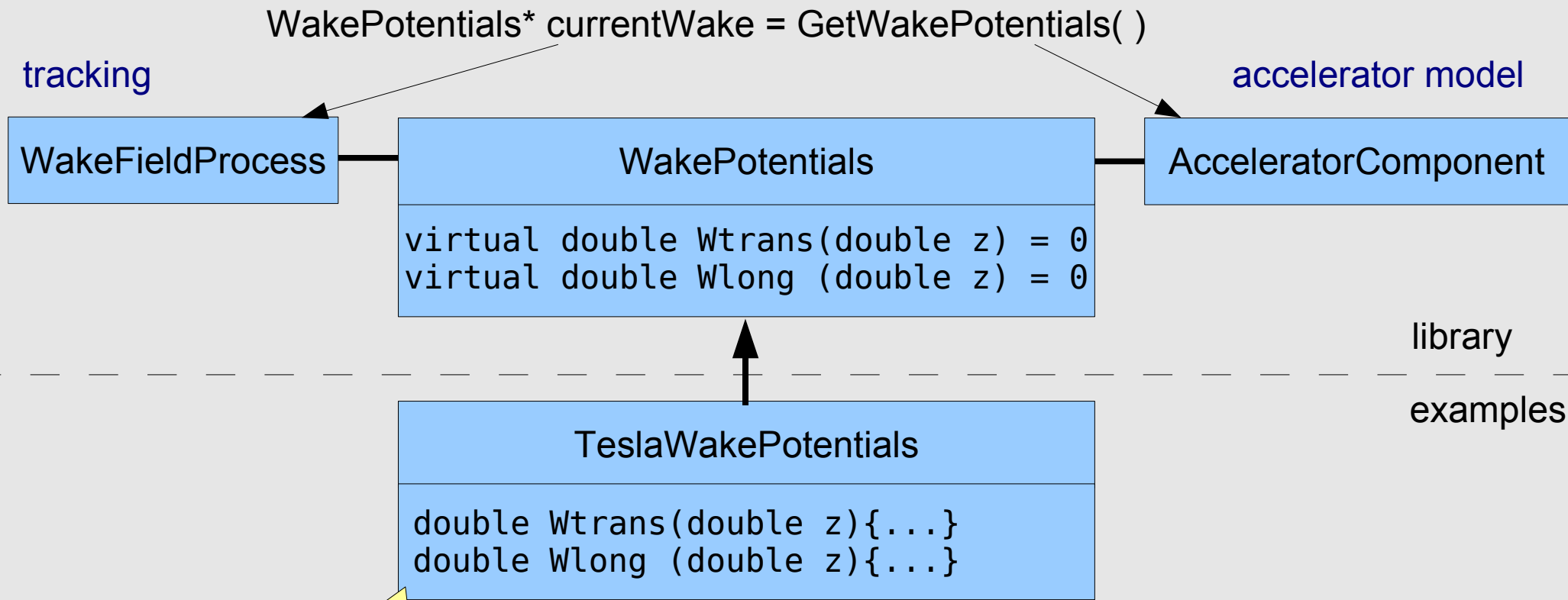
- Both trees are filled at arbitrary positions given by
 - name pattern = accelerator component or z position
- Automatic ROOT file creation
- Multiple trees
- **Successfully tested**: Isabell, Fabian* as guinea pigs :)
- Productivity booster for everybody who knows a bit of ROOT
 - Think about physics not code

*Summerstudent at DESY

New Code - Wakefields

- MERLIN was written mainly for cavity wakefields but already designed flexible enough to accommodate other wakefields
 - Roger Barlow and Adriana Bungau extended the scheme to include higher order short range wakefield
EUROTeV-2006-051
 - My own work on coupler wakefields needs similar modifications to MERLIN
EUROTeV-2008-003 and talk this meeting
- A general scheme for arbitrary wakefield types in one simulation without interference

WakeFieldProcess and WakeFieldPotential in MERLIN



ILCDFS & ILCML examples
ILC specific → example, not in library

- In Merlin WakeFieldPotentials is used as interface
- TeslaWakePotentials implements Wlong Wtrans
 - Cavity/ILC specific
- WakeFieldProcess does not know anything about the derived type (TeslaWakePotentials)
- For the new code we have different classes derived from WakeFieldProcesses and WakeFieldPotential

Derived XYZWakeFieldProcess and XYZWakePotential

```
typeid(*(wake->GetExpectedProcess()))==typeid(*this))
```

WakeFieldProcess

```
WakePotentials  
virtual double Wtrans(double z){return 0;}  
virtual double Wlong (double z){return 0;}  
void GetExpectedProcess(WakeFieldProcess*);
```

inherits functionality

SpoilerWakeFieldProcess

```
SpoilerWakeFieldPotentials  
virtual double Wtrans(double s, int m) = 0  
virtual double Wlong (double s, int m) = 0
```

to allow **type** check

Asks accelerator component for wakefield and **ask the wakfield if itself is the right process**

TaperedCollimatorPotentials

```
double Wtrans(double s, int m) {...}  
double Wlong (double s, int m) {...}
```


```
spoiler_wake = static_cast<SpoilerWakePotentials*>( currentWake);
```

New classes from Roger and Adriana

General WakeFieldProcesses

- Backward compatible
- Allows to derive different classes from `WakeFieldProcess`
 - `SpoilerWakeFieldProcess`
 - `CouplerWakeFieldProcess`and to have multiple `WakePotentials` in one simulation
- Presently in a separate CVS branch
 - will be merged for the new release

- At DESY the CVS to SVN migration has started
- A Very Short Introduction to Subversion (SVN):
 - Similar to CVS
 - svn checkout (get a local copy)
 - svn update (keep it up-to-date)
 - svn commit, add, move, delete etc.
 - But a different system: a database, different concept of tagging:
svn copy <https://svnsrv.desy.de/svn/merlin/trunk> \
https://svnsrv.desy.de/svn/merlin/tags/version_3.20 \
-m "tagging the 3.20 release of merlin"

- Runs on Linux and Windows
 - There is a Windows GUI: TortoiseSVN 
- Biggest disadvantage: **Authentication**
 - As a developer you will need a
 - **SSL certificate** (aka GRID certificate) **or** a
 - **DESY account**
 - Browser and download is open for everybody
- <http://svnsrv.desy.de/public/merlin> public
- <https://svnsrv.desy.de/svn/merlin> SSL
- <https://svnsrv.desy.de/desy/merlin> DESY kerberos
- No decision yet concerning MERLIN

New MERLIN Release

- The development version contains about 25 modified files compared to the last release (version 3.1)
- There is the wakefield branch
- ROOT and other new code
- The DFS example needs the development version

Updated Web Page

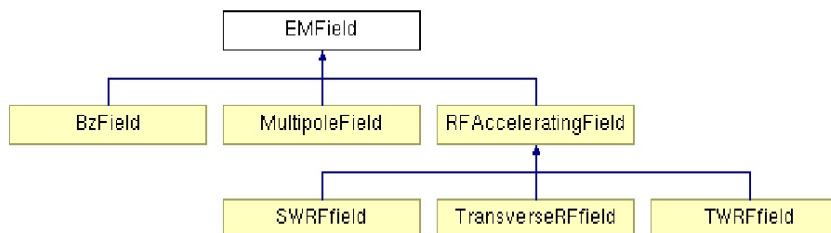
- Updated documentation
- New examples
- Short subversion introduction
- doxygen source code documentation and class Browser



EMField Class Reference

```
#include <EMField.h>
```

Inheritance diagram for EMField:



[List of all members.](#)

Public Member Functions

```
virtual ~EMField ()  
virtual Vector3D GetBFieldAt (const Point3D &x, double t=0) const=0  
virtual Vector3D GetEFieldAt (const Point3D &x, double t=0) const=0  
virtual Vector3D GetForceAt (const Point3D &x, const Vector3D &v, double q, double t=0) const
```

Constructor & Destructor Documentation

```
EMField::~~EMField( ) [virtual]
```

Definition at line [17](#) of file [EMField.cpp](#).

Member Function Documentation

```
virtual Vector3D EMField::GetBFieldAt ( const Point3D &    x,  
                                     double                t = 0  
                                     ) const [pure virtual]
```

Implemented in [BzField](#), [MultipoleField](#), [SWRFfield](#), [TransverseRFfield](#), and [TWRFField](#).

Referenced by [SpinParticleProcess::DoProcess\(\)](#), and [GetForceAt\(\)](#).

```
virtual Vector3D EMField::GetEFieldAt ( const Point3D &    x,  
                                     double                t = 0  
                                     ) const [pure virtual]
```

Implemented in [BzField](#), [MultipoleField](#), [SWRFfield](#), [TransverseRFfield](#), and [TWRFField](#).

Referenced by [GetForceAt\(\)](#).

```
Vector3D EMField::GetForceAt ( const Point3D &  x,  
                              const Vector3D & v,  
                              double           q,  
                              double           t = 0  
                              ) const [virtual]
```

Reimplemented in [MultipoleField](#), [SWRFfield](#), [TransverseRFfield](#), and [TWRFField](#).

Definition at line [22](#) of file [EMField.cpp](#).

References [cross\(\)](#), [GetBFieldAt\(\)](#), [GetEFieldAt\(\)](#), [Point3D](#), and [Vector3D](#).

The documentation for this class was generated from the following files:

- [EMField.h](#)
- [EMField.cpp](#)

Conclusions

- Migration to Subversion has started at DESY
 - CVS has been copied to SVN
 - No decision yet concerning MERLIN
- New release soon
 - ROOT
 - Wakefields
 - etc.
 - Doxygen code documentation