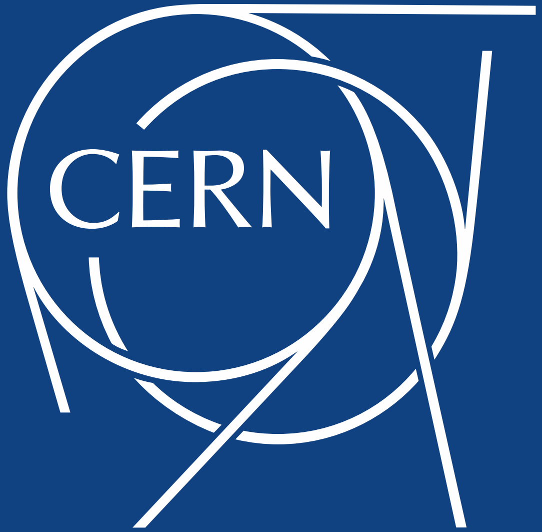
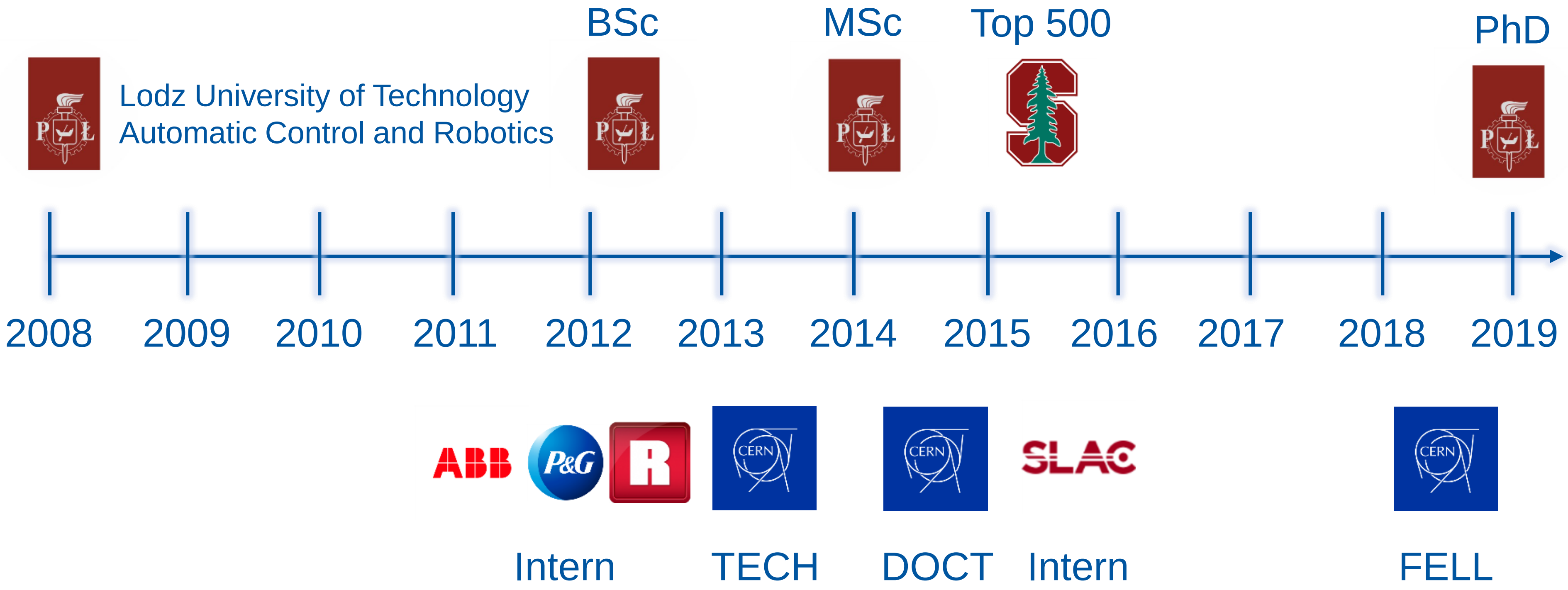

CERN

Scientific Software Engineering

Michał Maciejewski
on behalf of TE-MPE

With countless contributions from great colleagues including, but not limited to:
A. Verweij, B. Auchmann, L. Bortot, M. Prioli, M. Mentink, E. Ravaioli, K. Król, M. Koza, Z. Chariffouline, P. Hagen, J.B. Ghini,
S. Schops, H. De Gerseem, I. Cortes Garcia, A. Fernandez Navarro, Ch. Obermair, K. Andersen, M. Wilczek, K. Wolf, P. Bayrasy, …

About me



BSc

MSc

Top 500

PhD

Lodz University of Technology
Automatic Control and Robotics

2008 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019

Intern

TECH

DOCT

Intern

FELL

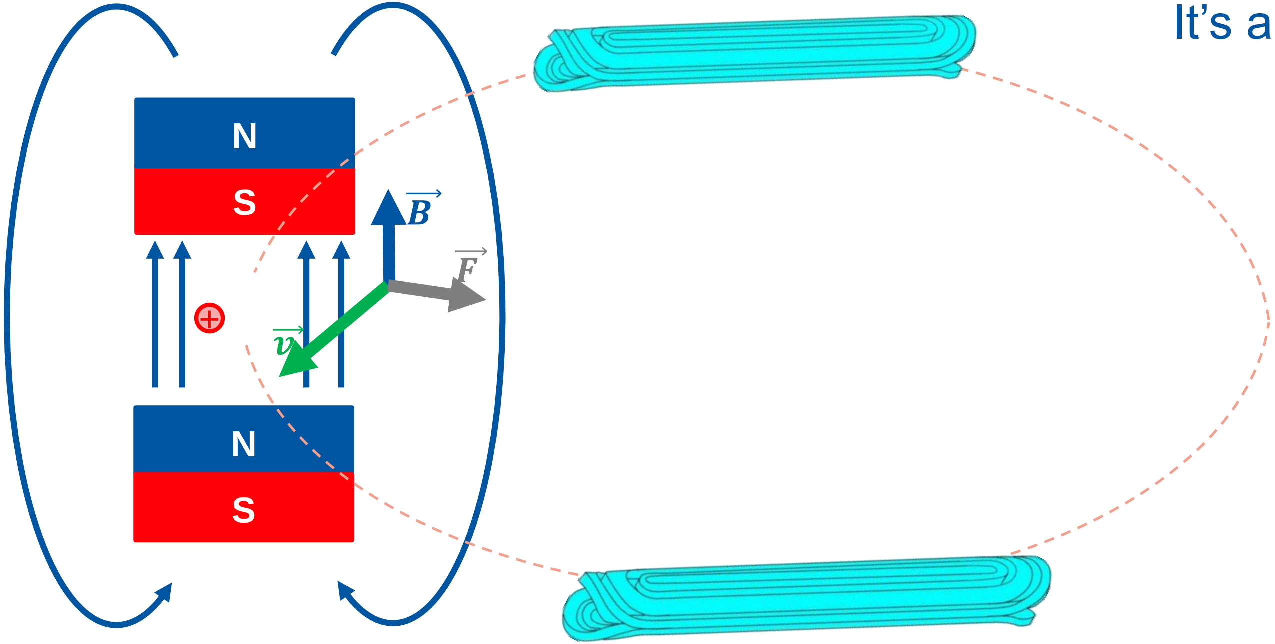
Outline

1. Introduction
2. Modelling of Superconducting Accelerator Circuits
3. Monitoring of Superconducting Accelerator Circuits
4. Software Quality
5. Summary

Superconducting Accelerator Magnets - *Introduction*

It's all about the Lorentz force

$$F_{\mathrm{L}} = q(E + v \times B)$$



Large Hadron Collider 27 km

Why do we Need Simulations?

The stored energy in a magnet is

7 MJ

20-ton truck @ 95 km/h

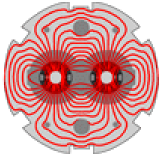
SAFETY & MAINTENANCE

RESEARCH & DEVELOPMENT

The stored energy in a circuit is

1.1 GJ

380-ton TGV @ ~50 km/h

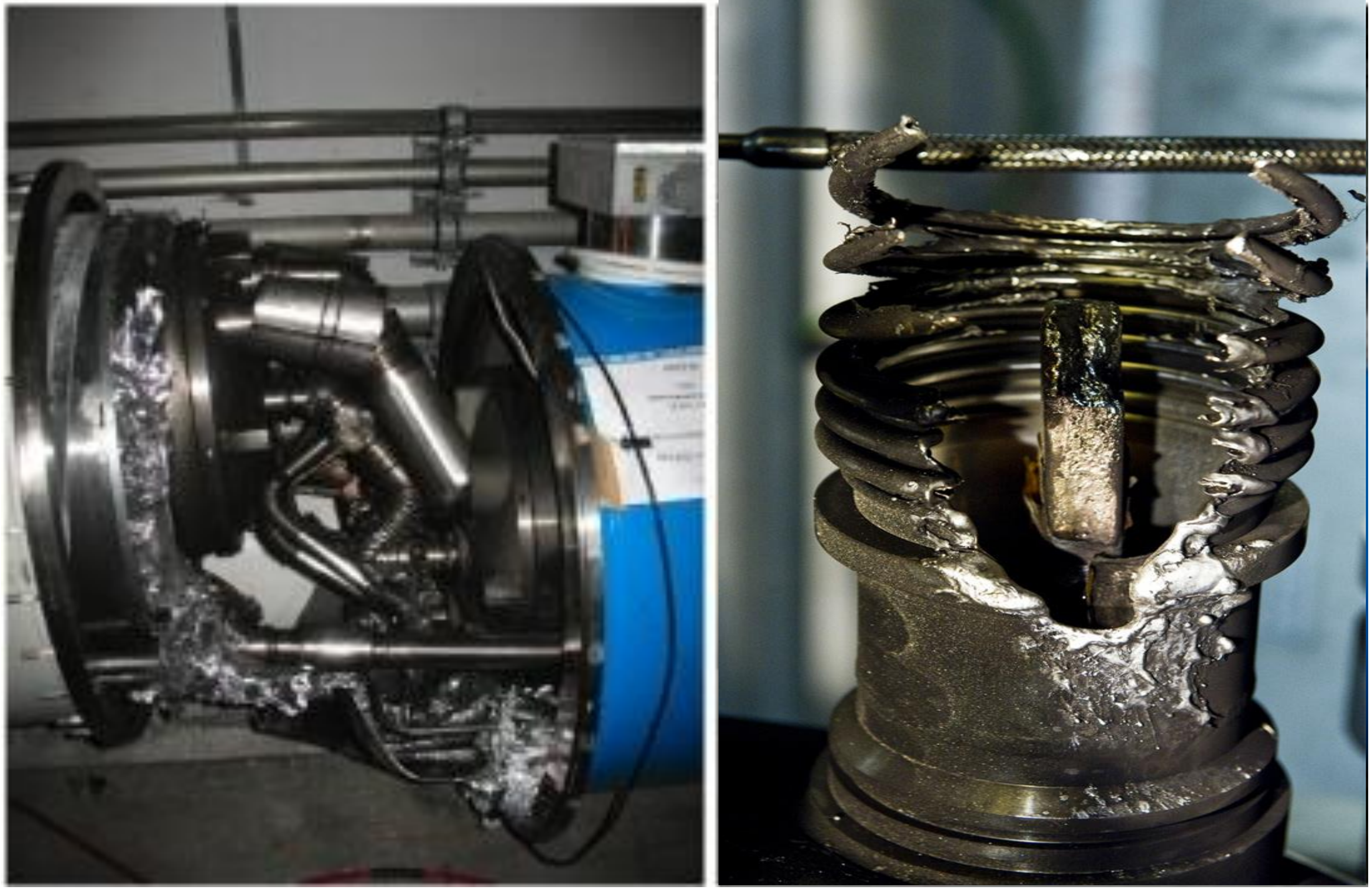


LHC

HiLumi
HL-LHC PROJECT

FCC
hh ee he

Model-based System Engineering!

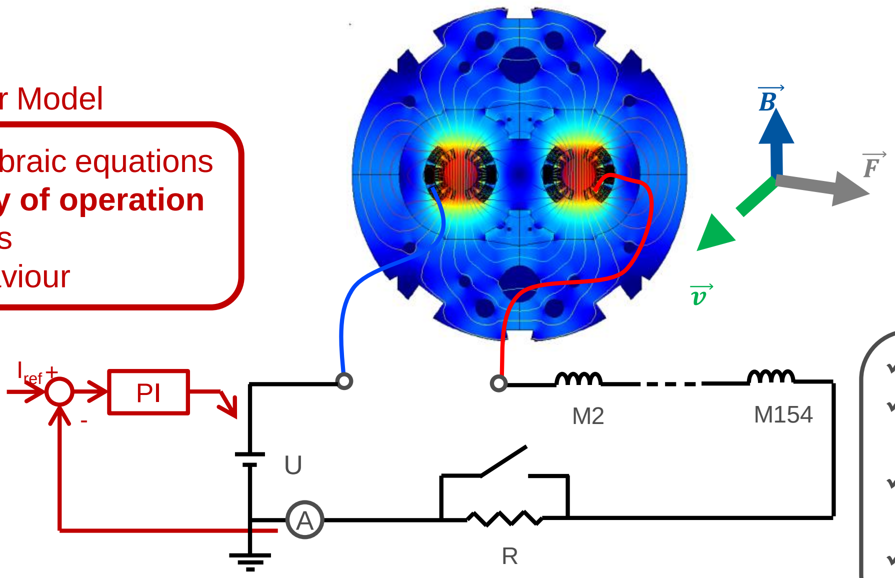


https://home.cern/news/press-release/cern/cern-releases-analysis-lhc-incident

Superconducting Accelerator Circuits – *Numerical Challenges*

Controller Model

- ✓ Differential-algebraic equations
- ✓ **fixed frequency of operation**
- ✓ 10-100 elements
- ✓ Non-linear behaviour



Field model of a magnet

- ✓ Partial differential equations
- ✓ **Varying time constants**
  ~10us (quench) - ~10ms (losses)
- ✓ **Varying geometric scales**
  ~10 um (filaments) - ~10 m (magnet)
- ✓ ~10 k degrees of freedom
- ✓ Highly non-linear material properties and equations

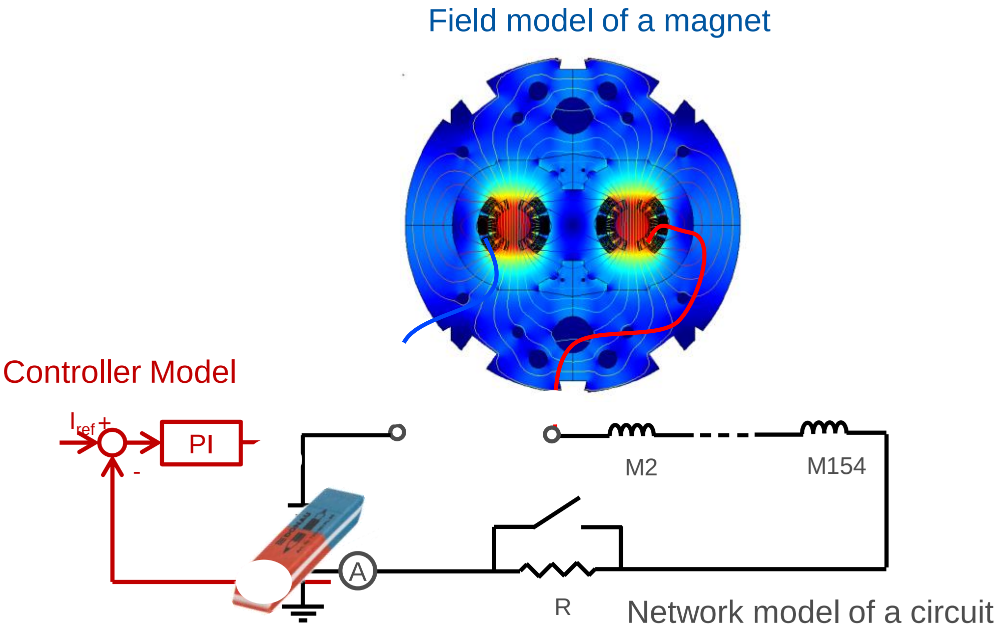
Network model of a circuit

- ✓ Differential-algebraic equations
- ✓ **Varying time constants**
  ~1 ms (switch) - ~10 min (circuit discharge)
- ✓ **Varying geometric scales**
  ~10 cm (diode) - ~10 km (circuit)
- ✓ ~10 k elements
- ✓ Non-linear behaviour

Mutli-Domain, Multi-Physics, Multi-Rate and Multi-Scale Problem

Superconducting Accelerator Circuits – *Divide et Impera*

These Multi-X phenomena can't be simulated with the desired accuracy in a single simulation tool.



Research Questions

1. How to represent a Multi-X problem in a consistent and generic way?
2. How to characterize the coupling between the domains?
3. What algorithm to choose in order to couple the models?
4. How to ensure consistency of the coupled simulation results?

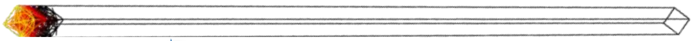
Coupling of dedicated models promises to tackle these numerical challenges

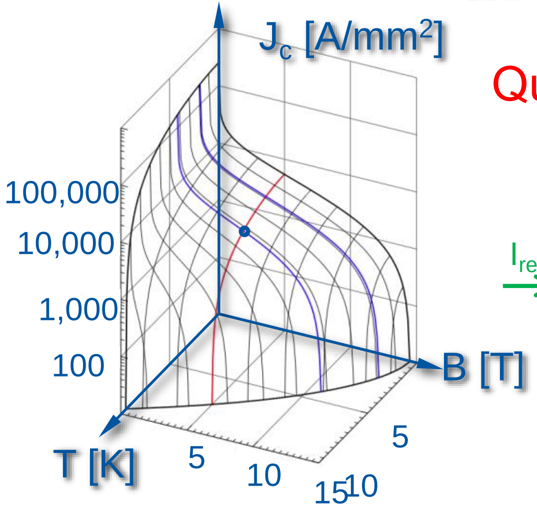
Simulation of Accelerator Magnets – *Quench*

COMSOL

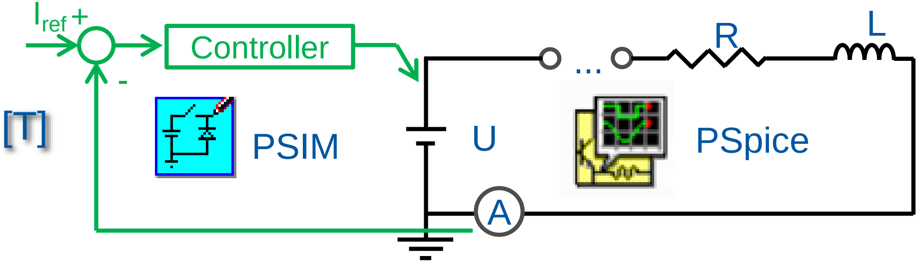
Quench → **1D Quench Propagation**

~micrometres
~microseconds





Quench is a part of magnet’s life – we need to prepare for battle!



~metres - kilometres
~milliseconds - minutes

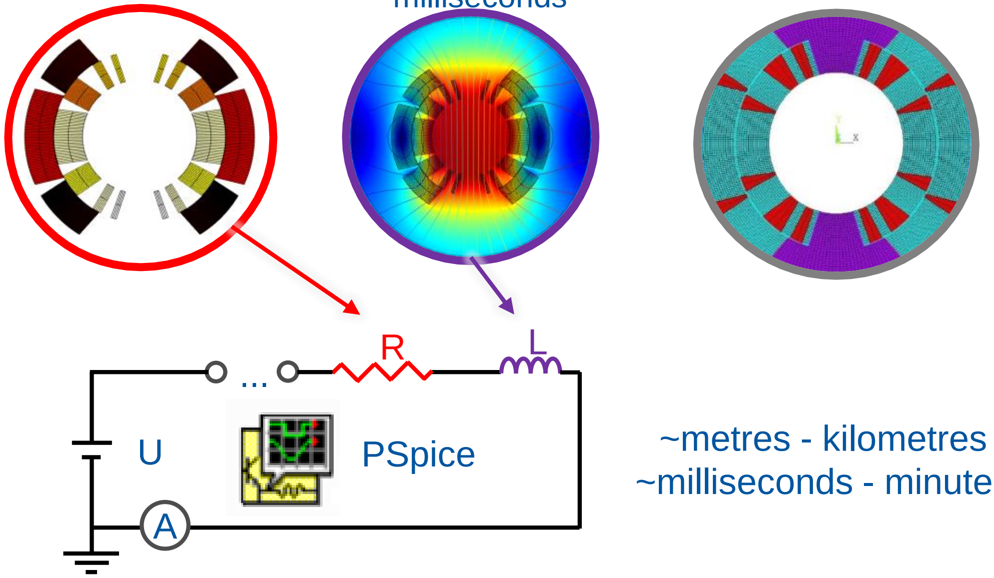
Simulation of Accelerator Magnets – *Protection*

COMSOL COMSOL COMSOL ANSYS

Quench → 1D Quench Propagation → 2D Quench Propagation → 2D Magnetic Model → 2D Mechanical Model

~micrometres
~microseconds

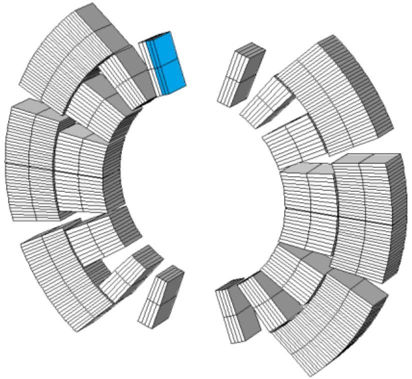
~millimetres-centimetres
~milliseconds



Multi-physics
Multi-rate
Multi-scale

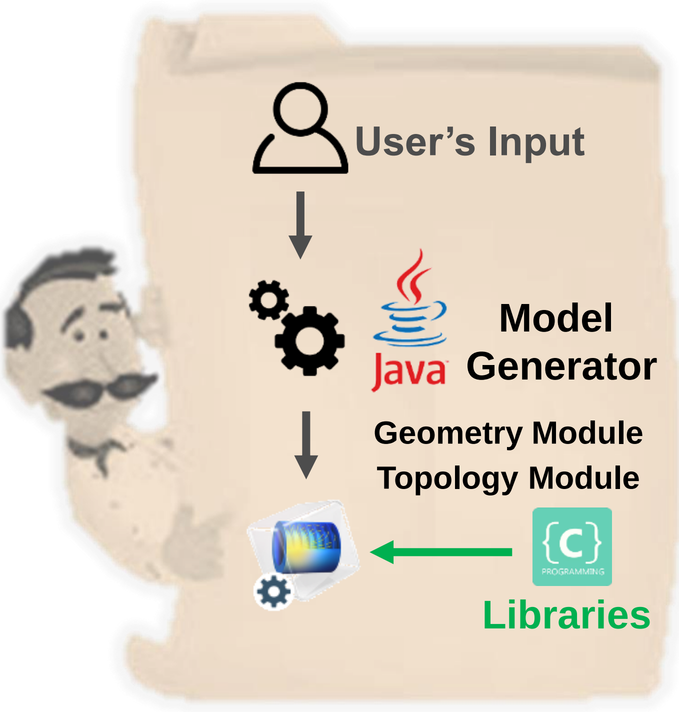
~metres - kilometres
~milliseconds - minutes

Automated Model Generation



~400 parametrized elements

COMSOL



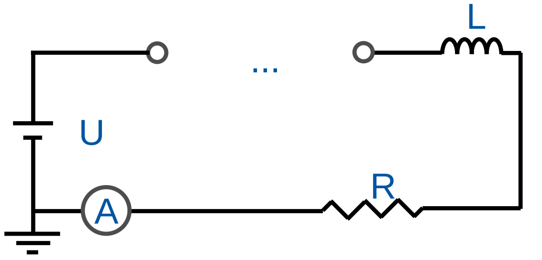
User's Input

Model Generator

Geometry Module

Topology Module

Libraries



~10 000 parametrized elements

PSpice

Finite element model

← How to connect them? →

Network model

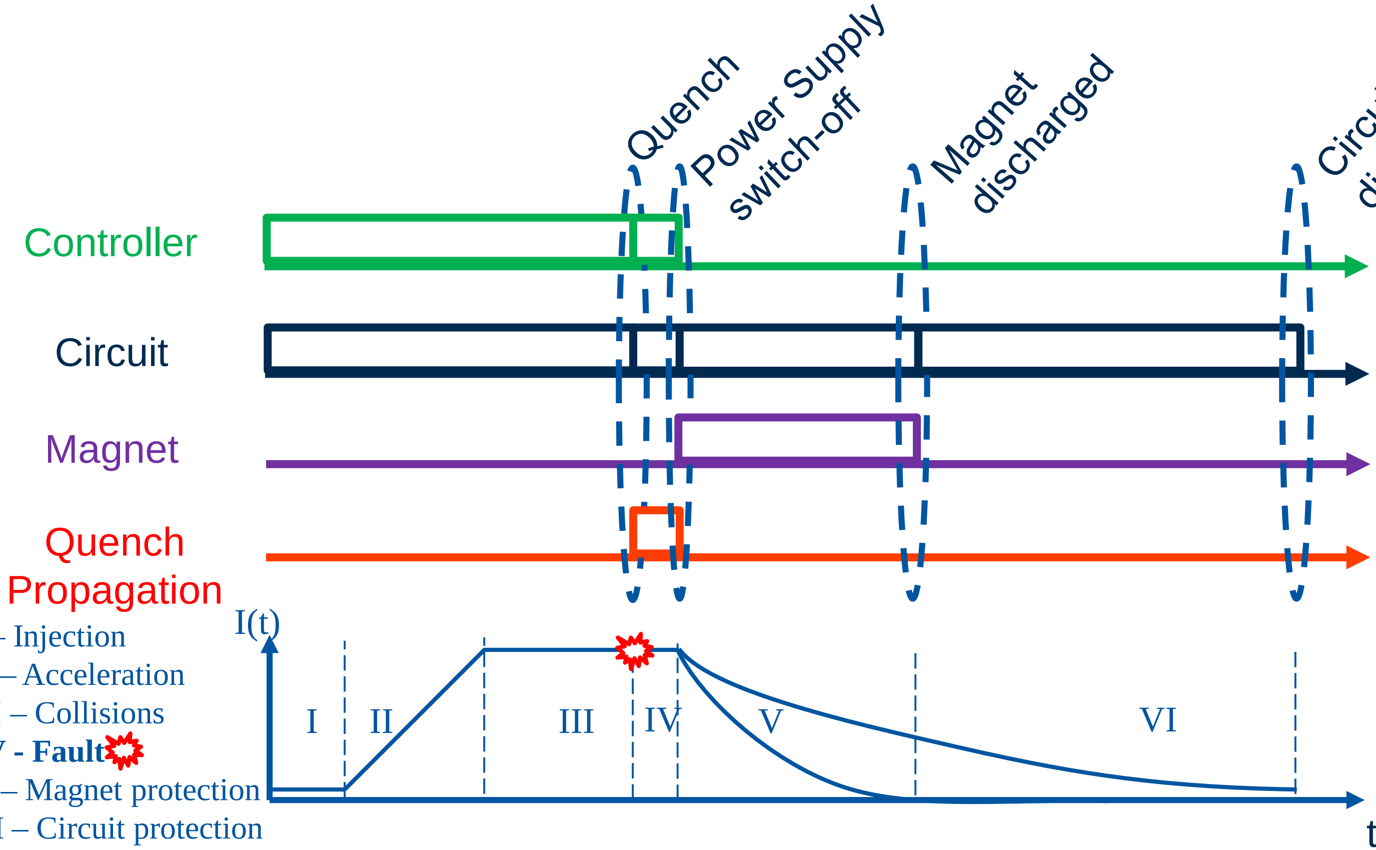
What is Simulation Coupling?

Convergence is **not guaranteed – external supervision** is required!

When the hands of the clock will first overlap?

more iterations
=
better accuracy

Hierarchical Co-Simulation



I – Injection
II – Acceleration
III – Collisions
IV - Fault
V – Magnet protection
VI – Circuit protection

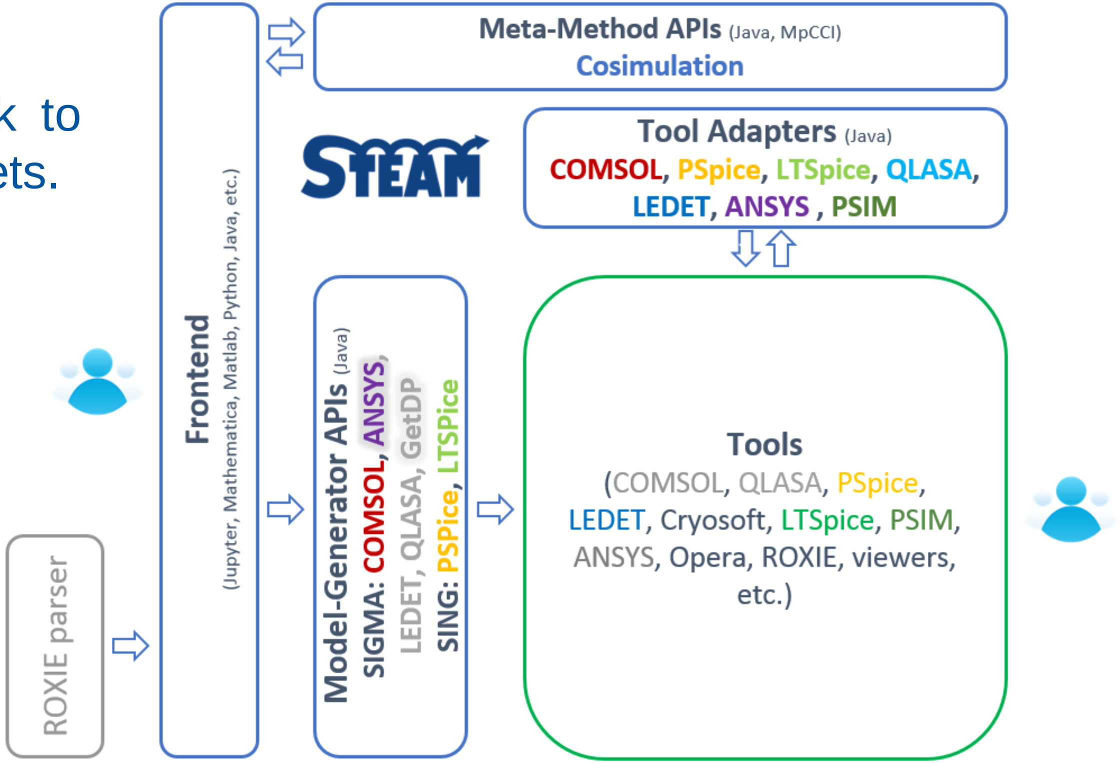
1+1 > 2

Simulated
Together
Everyone
Achieves
More

STEAM Architecture

STEAM is a simulation framework to study transient effects in SC magnets.
It consists of several pillars:

1. Validated tools
   → standalone, co-simulation
2. Model generation API
3. Tool adapter API
4. Meta-Methods
   →co-simulation, optimization
5. Front-end to interact with APIs



Meta-Method APIs (Java, MpCCI)
Cosimulation

Tool Adapters (Java)
COMSOL, PSpice, LTSpice, QLASA, LEDET, ANSYS , PSIM

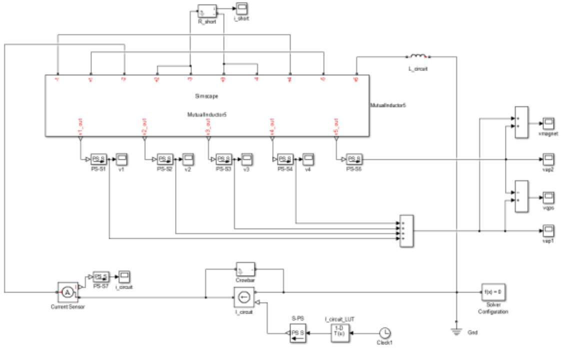
ROXIE parser

Frontend
(Jupyter, Mathematica, Matlab, Python, Java, etc.)

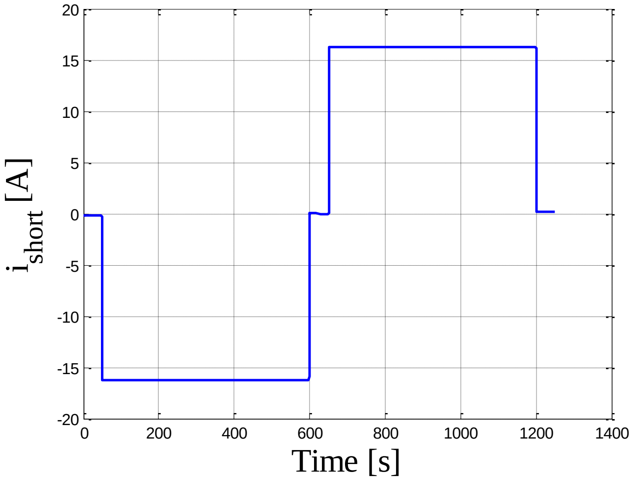
Model-Generator APIs (Java)
SIGMA: COMSOL, ANSYS,
LEDET, QLASA, GetDP
SING: PSPice, LTSPice

Tools
(COMSOL, QLASA, PSpice, LEDET, Cryosoft, LTSpice, PSIM, ANSYS, Opera, ROXIE, viewers, etc.)

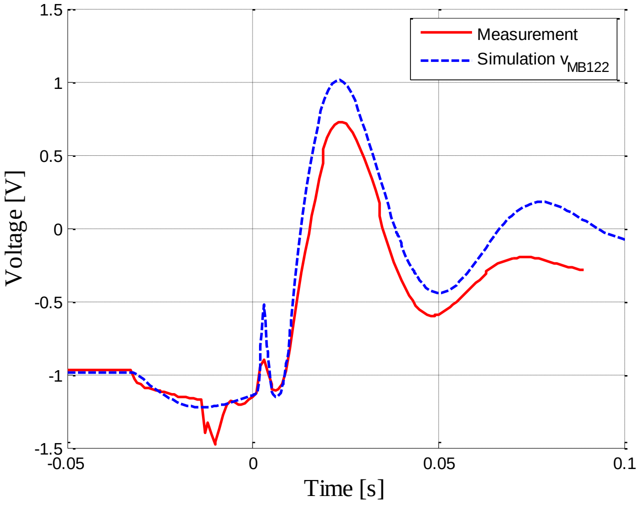
Internal short-circuit in a LHC main dipole



Equivalent circuit



Simulated short-circuit current

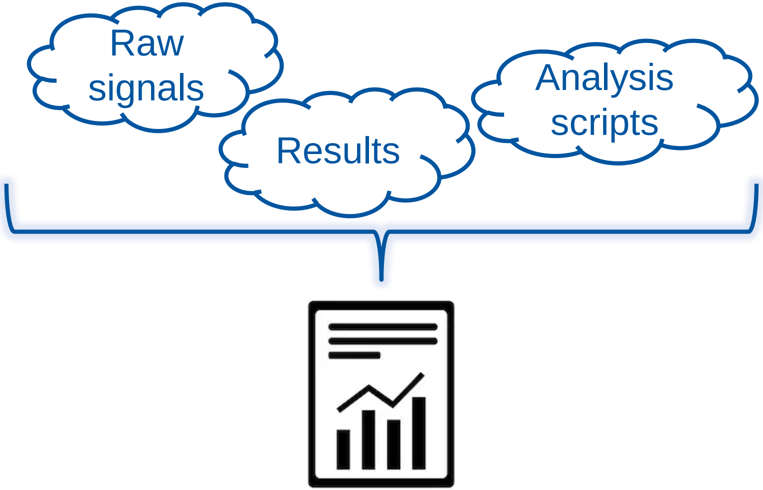


Voltage difference between magnet halves

Challenge

A typical event analysis includes

- ✓ collecting signals
- ✓ performing analysis
- ✓ post-processing results
- ✓ summarizing results
  (paper, report, presentation)



Python is very intuitive to use and comes with wealth of **powerful** libraries

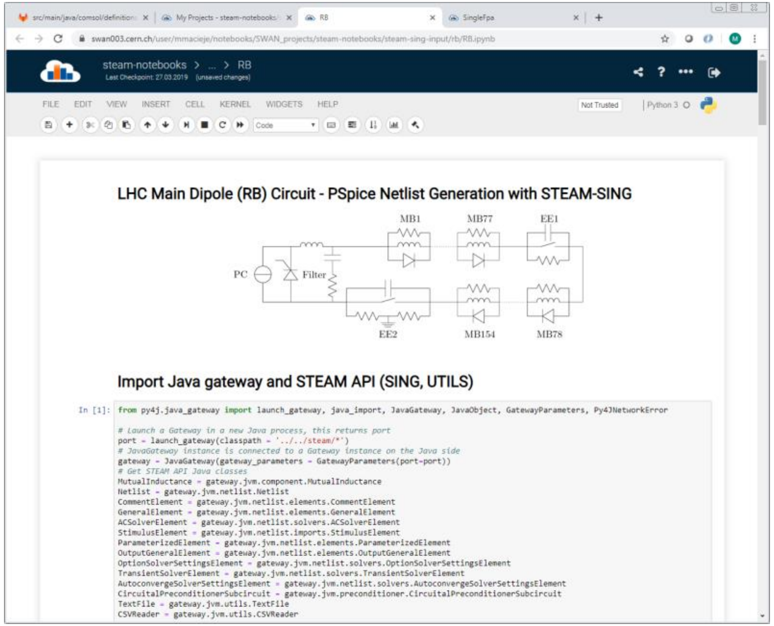
→ Little code to be developed and maintained

Notebooks do not require any installation and run in a **browser** (phone, tablet, laptop)

→ Notebooks can be immediately exported as a pdf file and stored for future reference

Digital Twin

Model Generation



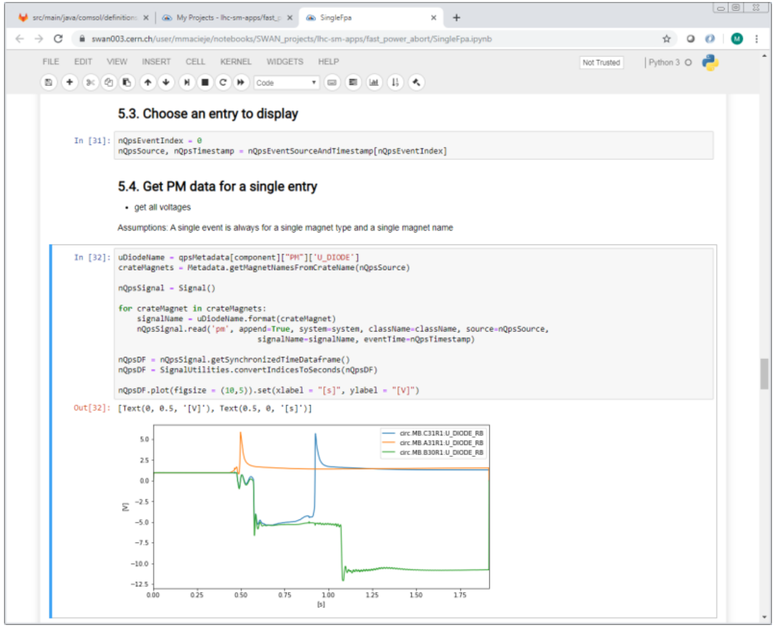
LHC Main Dipole (RB) Circuit - PSpice Netlist Generation with STEAM-SING

Import Java gateway and STEAM API (SING, UTILS)

```
In [1]: from py4j.java_gateway import launch_gateway, java_import, JavaGateway, JavaObject, GatewayParameters, Py4JNetworkError

        # Launch a Gateway in a new Java process, this returns port
        port = launch_gateway(classpath = '../../steam/*')
        # JavaGateway instance is connected to a Gateway instance on the Java side
        gateway = JavaGateway(gateway_parameters = GatewayParameters(port=port))
        # Get STEAM API Java classes
        MutualInductance = gateway.jvm.component.MutualInductance
        Netlist = gateway.jvm.netlist.Netlist
        CommentElement = gateway.jvm.netlist.elements.CommentElement
        GeneralElement = gateway.jvm.netlist.elements.GeneralElement
        ACSolverElement = gateway.jvm.netlist.solvers.ACSolverElement
        StimulusElement = gateway.jvm.netlist.imports.StimulusElement
        ParameterizedElement = gateway.jvm.netlist.elements.ParameterizedElement
        OutputGeneralElement = gateway.jvm.netlist.elements.OutputGeneralElement
        OptionSolverSettingsElement = gateway.jvm.netlist.solvers.OptionSolverSettingsElement
        TransientSolverElement = gateway.jvm.netlist.solvers.TransientSolverElement
        AutoconvergeSolverSettingsElement = gateway.jvm.netlist.solvers.AutoconvergeSolverSettingsElement
        CircuitalPreconditionerSubcircuit = gateway.jvm.preconditioner.CircuitalPreconditionerSubcircuit
        TextFile = gateway.jvm.utils.TextFile
        CSVReader = gateway.jvm.utils.CSVReader
```

=

Signal Acquisition



5.3. Choose an entry to display

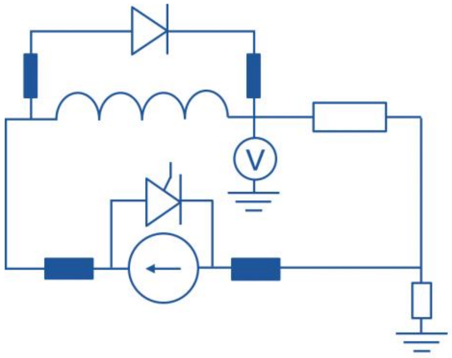
```
In [31]: nQpsEventIndex = 0
         nQpsSource, nQpsTimestamp = nQpsEventSourceAndTimestamp[nQpsEventIndex]
```

5.4. Get PM data for a single entry

- get all voltages

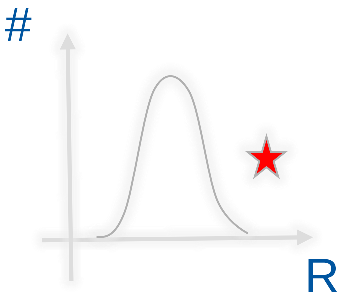
Assumptions: A single event is always for a single magnet type and a single magnet name

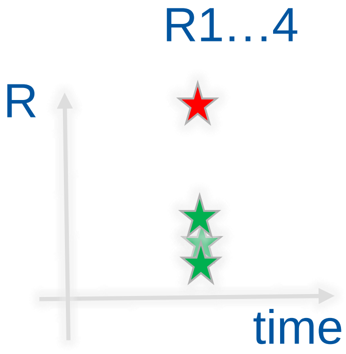
```
In [32]: uDiodeName = qpsMetadata[component]["PM"]['U_DIODE']
         crateMagnets = Metadata.getMagnetNamesFromCrateName(nQpsSource)

         nQpsSignal = Signal()

         for crateMagnet in crateMagnets:
             signalName = uDiodeName.format(crateMagnet)
             nQpsSignal.read('pm', append=True, system=system, className=className, source=nQpsSource,
                                          signalName=signalName, eventTime=nQpsTimestamp)

         nQpsDF = nQpsSignal.getSynchronizedTimeDataframe()
         nQpsDF = SignalUtilities.convertIndicesToSeconds(nQpsDF)

         nQpsDF.plot(figsize = (10,5)).set(xlabel = "[s]", ylabel = "[V]")
Out[32]: [Text(0, 0.5, '[V]'), Text(0.5, 0, '[s]')]
```

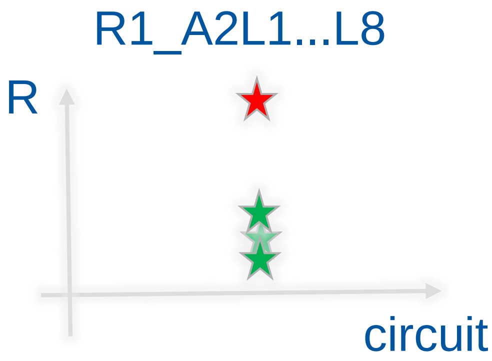


Signal Analysis

1. With historical data we derive expected behavior and trends.
2. With on-line data we compare behavior with others.







digital-twin

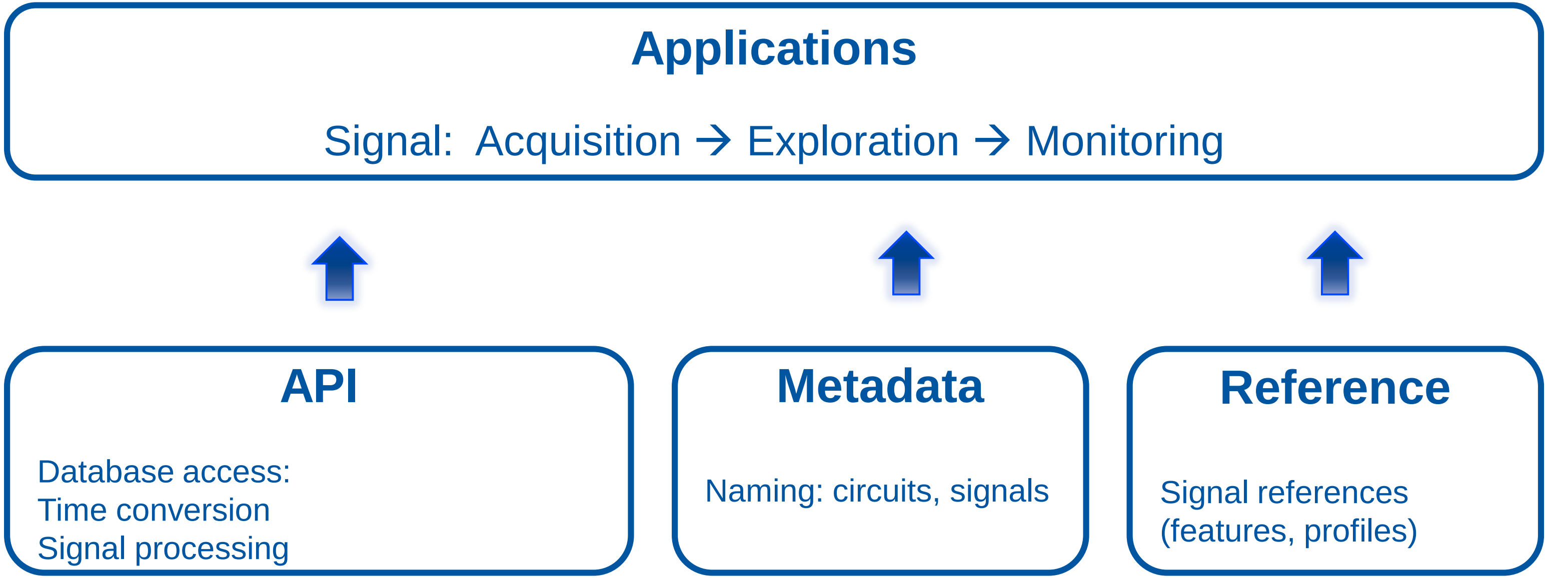
trends

intra-component

cross-population

Predicatail: https://othello.predictail.com/#toggle-id-4

LHC Signal Monitoring - Architecture



Applications

Signal: Acquisition → Exploration → Monitoring

API

Database access:
Time conversion
Signal processing

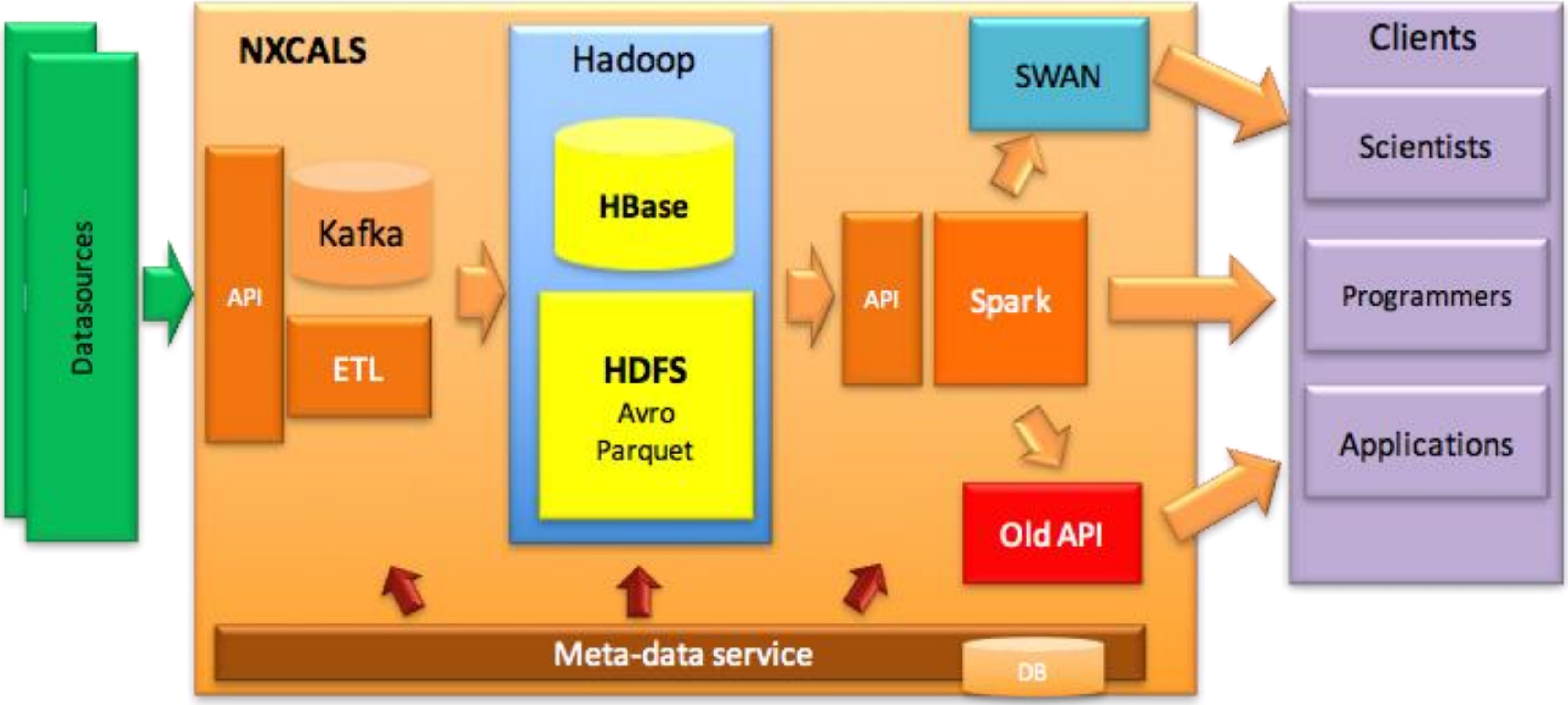
Metadata

Naming: circuits, signals

Reference

Signal references
(features, profiles)

New Accelerator Logging System - NXCALS



Courtesy: NXCALS team, BE-CO-DS, source: P. Sowiński: SWAN for NXCALS

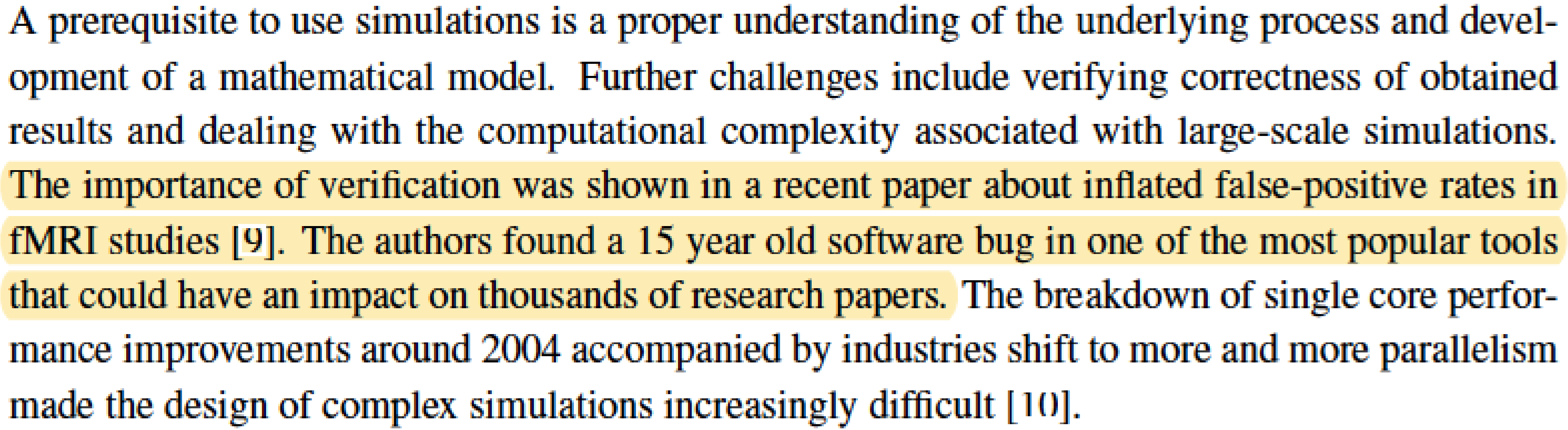
Outline

1. Introduction
2. Modelling of Superconducting Accelerator Circuits
3. Monitoring of Superconducting Accelerator Circuits
4. Software Quality
5. Summary

```
// When I wrote this, only God and I understood what I was doing
// Now, God only knows
```

-anonymous

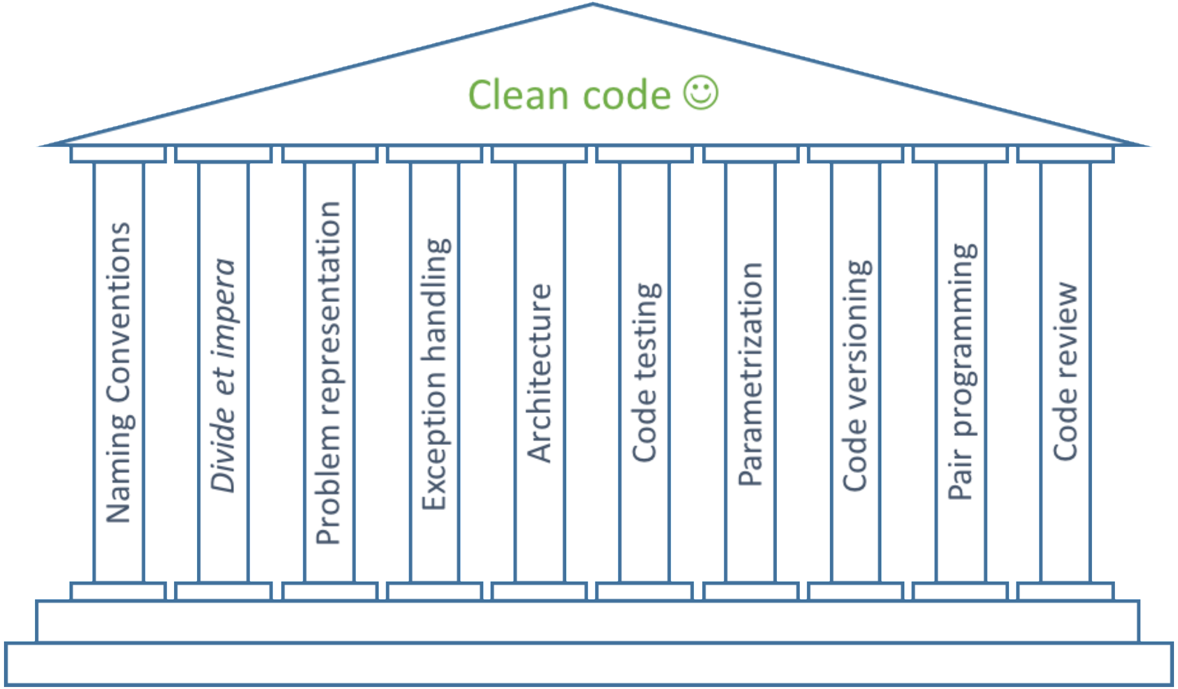
Costs in Research



A prerequisite to use simulations is a proper understanding of the underlying process and development of a mathematical model. Further challenges include verifying correctness of obtained results and dealing with the computational complexity associated with large-scale simulations. The importance of verification was shown in a recent paper about inflated false-positive rates in fMRI studies [9]. The authors found a 15 year old software bug in one of the most popular tools that could have an impact on thousands of research papers. The breakdown of single core performance improvements around 2004 accompanied by industries shift to more and more parallelism made the design of complex simulations increasingly difficult [10].

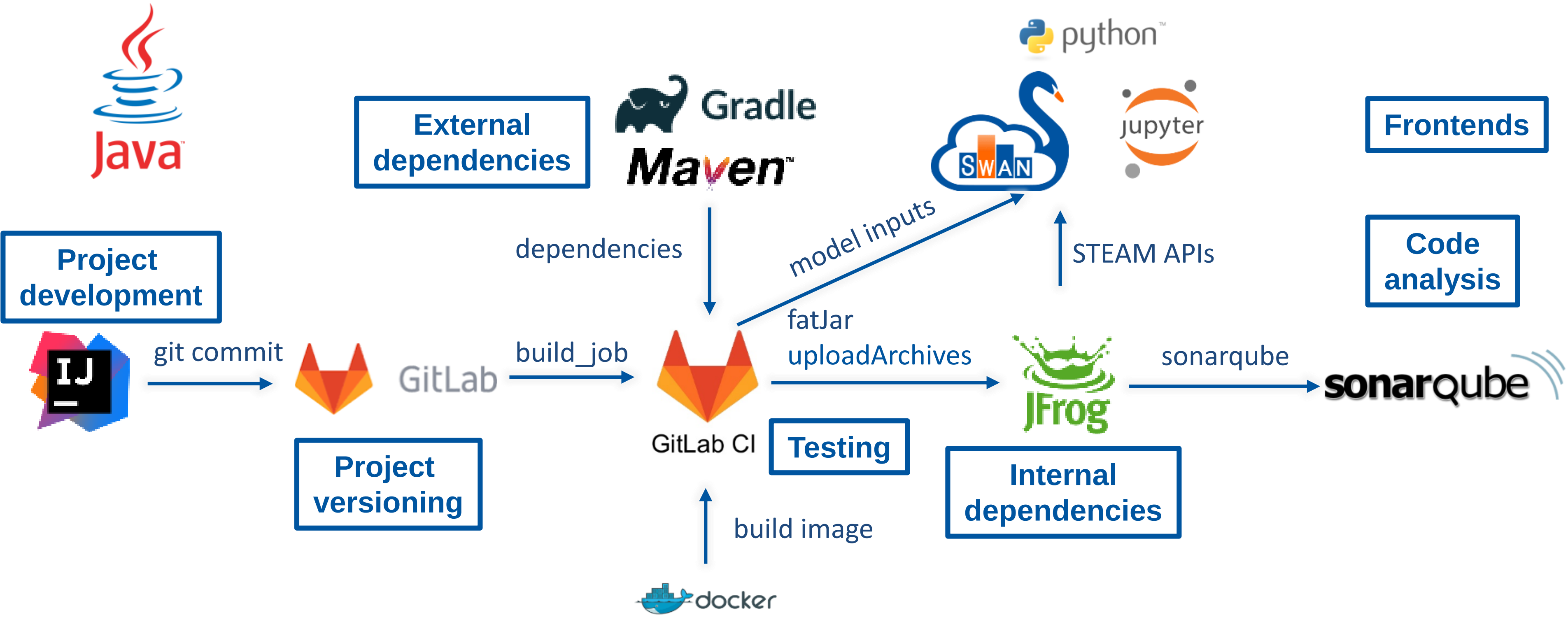
Source: L. Breitwieser, From Cortex3D to BioDynaMo The Birth of a New Platform for Large-Scale Reproducible Biological Simulations

The Temple of Clean Code



9 July 2015, Clean code development workshop, jointly with MPE-MS
13 Aug 2015, Object oriented programming workshop, jointly with MPE-MS

STEAM Continuous Integration



Project development

git commit

Project versioning

build_job

External dependencies

dependencies

model inputs

STEAM APIs

fatJar uploadArchives

Testing

Internal dependencies

sonarqube

Frontends

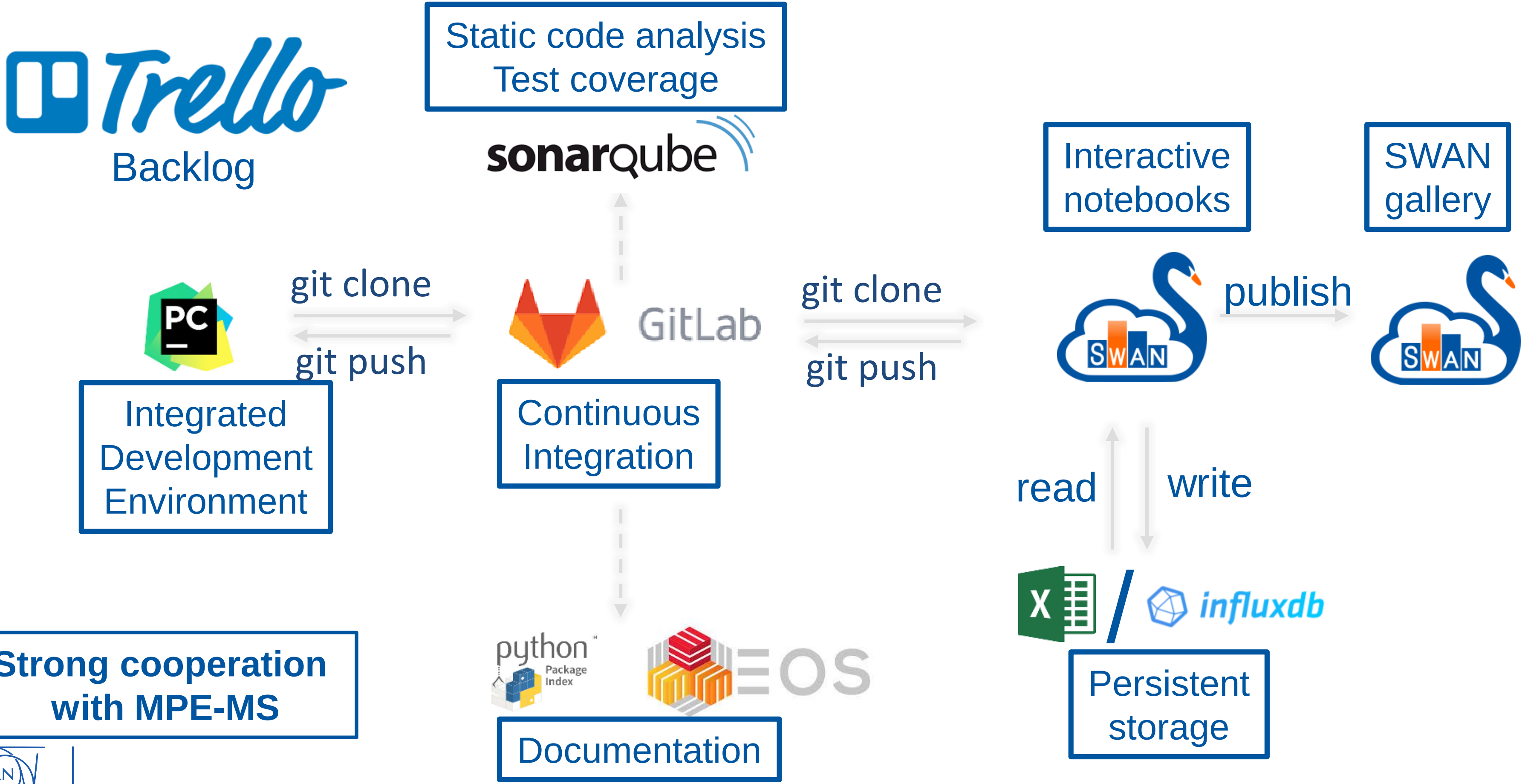
Code analysis

build image

*Strong cooperation with TE-MPE/MS (K. Król, JC. Garnier)

The pipeline automatically executes project build, testing, sharing, and analysis.
This ensures the maintainability of the project.

LHC Signal Monitoring Continuous Integration



Backlog

Static code analysis
Test coverage

Integrated Development Environment

git clone

git push

Continuous Integration

git clone

git push

Interactive notebooks

SWAN gallery

publish

read

write

Persistent storage

Documentation

Strong cooperation with MPE-MS

Majority (except for PyCharm IDE and Python Package Index) services are supported by CERN IT

Agile manifesto

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity--the art of maximizing the amount of work not done--is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Conclusion

Analysis of superconducting accelerator circuits involves a good understanding of

1. physics (electrical, magnetic, thermal, mechanical phenomena)
2. numerical simulations (FEM and network models, co-simulation)
3. software development (various technologies, conventions, infrastructure)
4. team dynamics (communication, presentation, conflict resolution)
5. engineering practices (documentation, consistency, simplicity)

And is a lot of fun! ;-)