Software Development at Belle II
Thomas Kuhr, Thomas Hauth

- **Aim:** Reliable, sophisticated, and easy-to-use software for acquisition, simulation, reconstruction, and analysis of Belle II data
- **Challenge:** Regional distribution, different (cultural) backgrounds and skills of developers
  - State-of-the-art tools, balance between common rules and individual freedom, communication

**Tools:**
- C++11 (gcc 4.7, clang 3.4, icc 14), python for steering files
  - metaframework for analysis (see talk by M. Staric)
- Central svn repository, code browser
- Astyle for C++ and pep8/autopep8 for python code style, checked on commit
- Redmine for issue tracking
- Doxygen and twiki for documentation
- Googletest for unit tests, valgrind for memory check
- Validation framework for regression tests
- **Buildbot** for several automated tasks:
  - Incremental build on commit for fast feedback
  - Full builds with different compilers and system on commit, email of new errors/warnings to authors
  - Nightly build, mem-check, and validation, email of problems to librarians
  - Installation of new tools and releases at sites
  - Compilation and creation of binary distribution tarballs for different system for externals and releases
  - Monthly build of tagged packages

**Organization:**
- Subgroups for database, generators, simulation, background, tracking, alignment
- Software organized in packages with a responsible librarian who controls write access and can tag the package
- Monthly integration builds of tagged packages and feature driven releases
- Weekly developers meetings, two workshops per year, joint sessions with other groups at collaboration meetings