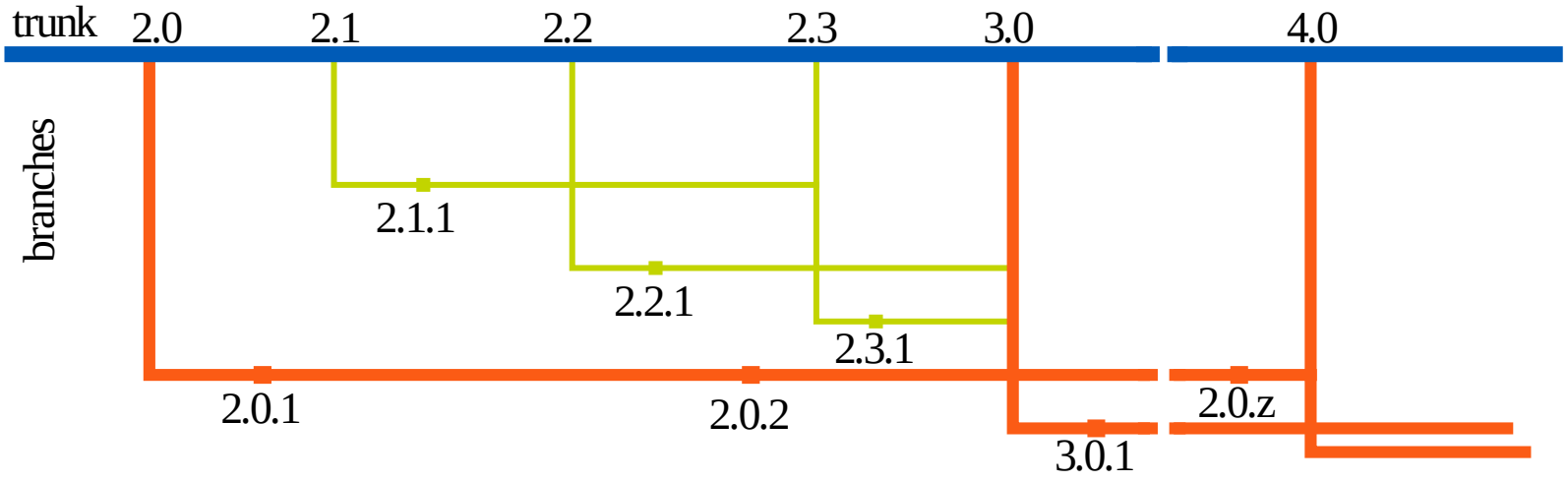
LATEST NEWS AND PROSPECTS IN *Code_Saturne*

From version 3.0 to 3.3, on the road to 4.0

April 2 2014

Code_Saturne versioning scheme reminder

- **From version 2.0 on, different kinds of versions "x.y.z" are released**
  - Production version every two years (x increasing)
    - With the release of a Verification & Validation summary report
  - An intermediate version every six months (y increasing)
    - With non-regression tests to ensure the code quality
  - Corrective versions when needed (z increasing)
    - To make sure the users are provided with bug fixes and ports
    - XML and user subroutines remain compatible, so upgrading is encouraged



Code_Saturne version history

- **Pre-open source versions**
  - 1998: prototype (long time EDF in-house experience, ESTET-ASTRID, N3S, ...)
  - 2000: version 1.0 (basic modeling, wide range of meshes)
    - 2001: Qualification for single phase nuclear thermal-hydraulic applications
  - 2004: version 1.1 (complex physics, LES, parallel computing)
  - 2006: version 1.2 (state of the art turbulence models, GUI)

- **Open source (GPL) versions (retired, old stable, stable, intermediate)**
  - 2008/11: version 1.3 (massively parallel, ALE, code coupling, ...)
    - 2008/11: version 1.4 (parallel I/O, multi-grid, atmospheric, cooling towers, ...)
  - 2010/08: **version 2.0** (parallel joining, code coupling, easier install, extended GUI)
    - 2011/10: version 2.1 (parallel mesh partitioning, dynamic memory, improved scripts, coupling with Syrthes 4)
    - 2012/03: version 2.2 (EBRSM, ALE improvements)
    - 2012/07: version 2.3 (many physical model additions, Cp handling, BC formulation changes, coupled velocity)
  - 2013/03: **version 3.0** (AFM, DFM thermal wall laws)
    - 2013/06: **version 3.1** (Lagrangian additions and post-processing, k-ω robustness, radiative quadratures)
    - 2013/12: **version 3.2** (Joining-based rotor-stator, Lagrangian coal combustion, atmospheric chemistry)

Code_Saturne version 3.0

- **Released March 22, 2013**
  - Just before last year's user meeting
  - Now at patch release 3.0.3
    - 3.0.4 will be released very soon
  - Check NEWS file to see if you should upgrade
    - http://code-saturne.org/viewvc/saturne/branches/Version3_0/
- **Described in detail in 2013 user meeting**

PRODUCTION VERSION

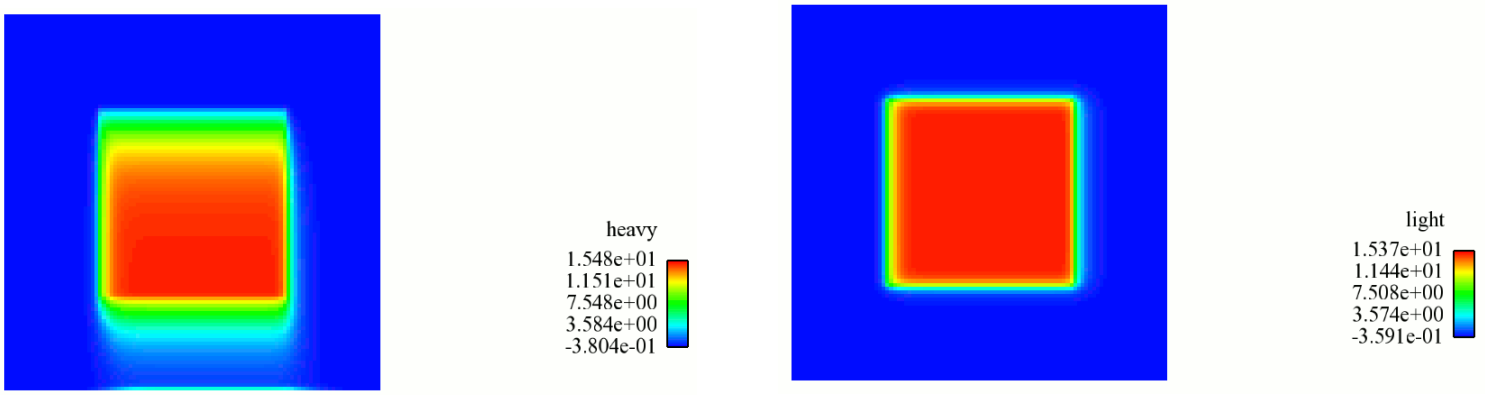
- Current stable
  - recommended for most studies under quality assurance
  - will become "old stable" when version 4.0 is released (April 2015)
  - will be maintained until release of version 5.0 (2017)
- Old stable is 2.0
  - will be retired when 4.0 is released (April 2015)

Code_Saturne Version 3.1 (1/4)

- **Released June 2013**
  - shortly after version 3.0
  - contains mainly developments that could have gone into 3.0, but did not make it in time before feature freeze
- **General changes**
  - documentation updates
  - improve k-ω robustness with low $y+$
    - fixes longstanding bug, detectable on diffuser test case
      - versions 1.3 and 2.0 forced relaxation to work around this, at the expense of unsteady computations
      - version 3.0 identified the issue, forcing relaxation only in affected cases
        - » fix is not merged into 3.0, as it is deemed too intrusive
  - hybrid parallelism (OpenMP)
    - add numbering options for threads
  - optional support for NEPTUNE's Equations of State Library
- **radiative model**
  - add new S4 S6 S8 and Tn quadratures

Code_Saturne Version 3.1 (2/4)

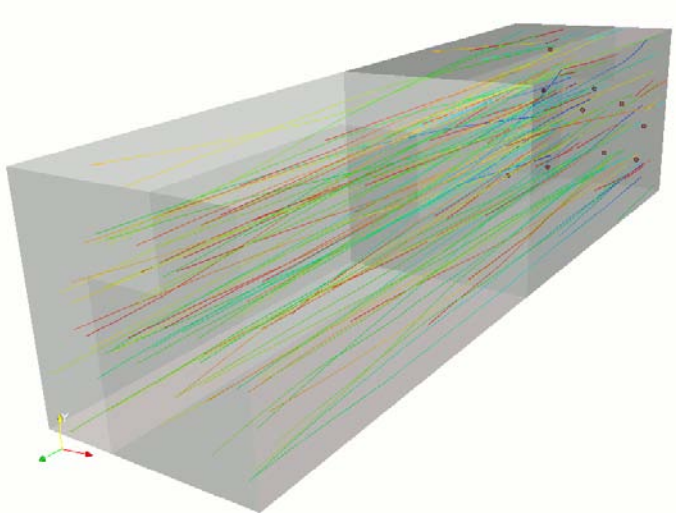
- **Coal combustion**
  - increase max coals to 5, add coke composition
  - remove old coal combustion model
- **add a drift model**
  - first used for coal combustion
  - general framework (radionuclide transport, …)



Code_Saturne Version 3.1 (3/4)

- **GUI and scripts**
  - prepare for Windows (with features useful for all, and essential to windows population)
    - handle whitespace in paths
    - cases may now be created (not just edited) directly from the GUI
  - using CFD_Study, display the monitoring points on the SALOME VTK viewer

- **Lagrangian model**
  - add zero-flux particle boundary condition to be applied with Eulerian symmetries.
  - with combustion, use a formulation of the coal density local to a particle and improve the numerics.
  - add a particle resuspension model.
  - implement a wall law for fluid velocity, k and ε for the deposition sub-model
  - implement a BC based on the DLVO theory
  - full rewrite of the postprocessing output
    - now based on the standard mechanisms ; no restrictions for parallel runs
    - trajectories are now really usable



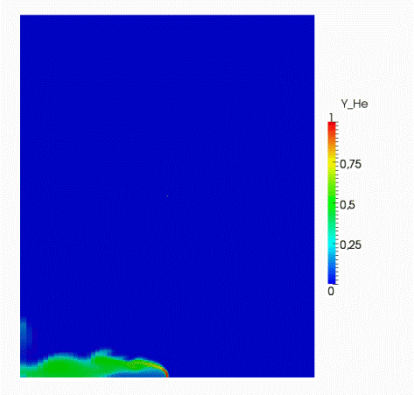
Code_Saturne Version 3.1 (4/4)

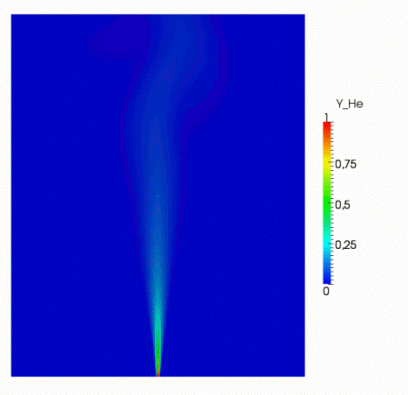
- **Autovnv improvements**
  - global postprocessing
  - prescribe results name,
  - many other useful additions
- **Automatic installer changes**
  - The installer is now in the top-level directory, and does not download *Code_Saturne* anymore
    - download code sources `code_saturne_x.y.z.tar.gz` file first, the installer is inside
  - The setup file template is generated by a first call to install_saturne.py.
  - MPI should now be installed upstream, but PT-SCOTCH and ParMetis are now handled.
- **For more details, see NEWS file in:**
  - http://code-saturne.org/viewvc/saturne/branches/Version3_1/

Code_Saturne Version 3.2 (1/6)

- **Released December 2013**

- **General changes**
  - Remove uncoupled velocity solver (`ivelco = 0`), deprecated since version 3.0
  - add a new Boundary Condition type for free inlet
    - can be used for natural convective flows in free atmosphere for instance (plumes, flame, etc.).
  - Turbulence:
    - Major change in Rij-epsilon models:
      - the Daly Harlow model on the diffusive term is now by default for SSG
      - the GGDH brick is used for all the models (LRR, SSG, EBRSM)
      - the "diffusivity_tensor" is added as a field key word
      - Rij-epsilon routines are cleaned up and documented using Doxygen.

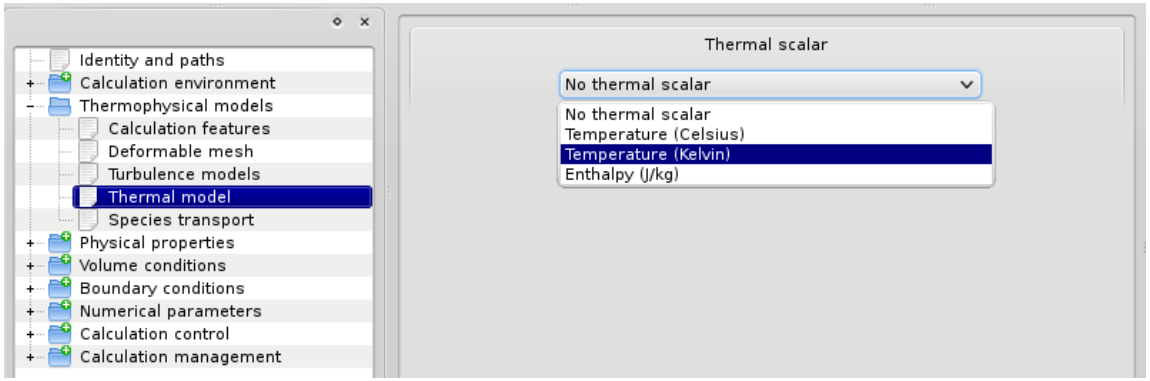




- **Turbomachinery modeling:**
  - add a rotor-stator model based on mesh joining.
    - see specific presentations on this subject today

Code_Saturne Version 3.2 (2/6)

- **Thermal model**
  - the thermal model is now defined by the "`itherm`" keyword/variable, which replaces `iscsth(iscalt)`.
    - In the case of temperature, the scale used is defined by a separate variable (`itpscal`). For additional user scalars, a new array `iscacp` is defined, such that `iscacp(iscal)` defines whether the scalar behaves like a temperature, so the possibility of modeling multiple passive "temperatures" is not lost.
  - This change allows for better consistency between the standard and specific physics, as the thermal variable is now always a "model" scalar, and user scalars remain separate.
    - so `nscapp = 1` using a thermal model but no specific physical model
    - It also allows better consistency between the GUI and user subroutines logic
    - It also allows querying the thermal model with one less indirection.



Code_Saturne Version 3.2 (3/6)

- **Atmospheric module:**
  - add gazeous chemistry models.
  - plug the SIze REsolved Aerosol Model (SIREAM).
  - see general presentation on atmospheric module today
- **Particle tracking module:**
  - add a modeling of the drying phase of the coal particle combustion
  - add a new boundary condition to simulate coal fouling mechanism
  - implementation of a particle discretization in the coal combustion model:
    - backwards compatibility is ensured (set nlayer = 1)
    - computation of intra-particle thermal gradients
    - adaptation of chemical source terms to temperature discretization
    - reworked the particle injection for coal (clear difference between standard and user-defined coal composition)
    - adapted the particles and trajectories export routines to be able to output variable information for a specific layer

Code_Saturne Version 3.2 (4/6)

- **Compressible module:**
  - change the compressible algorithm from a density formulation to a pressure formulation
  - merge the compressible algorithm with the coupled velocity components algorithm
  - adapt standard operators (`codits`, `bilsc*`) in order to make them compatible with the compressible algorithm
  - implement analytical flux boundary condition
    - plus a new total enthalpy / total pressure boundary condition with a fixed point algorithms, generalization of the subsonic outlet
  - new set of BC coefficients for the convection operator for compressible flows
  - density is now a property only, not a solved variable
- **Coal combustion module:**
  - added new NOx model for coal combustion;
  - introduction of the coal thermal conductivity
    - for the calculation of intra particle gradients in particle-tracking module

Code_Saturne Version 3.2 (5/6)

- **Documentation**
  - moved tutorials outside the codebase
    - this allows looser synchronization with the code base, as tutorials may be updated somewhat less frequently
    - for non-EDF users, pdf's are available on the web site; to contribute, please contact us
  - Progress in Doxygen documentation
    - Fortran modules
    - user examples
    - Fortran routines
    - install Doxygen documentation from tarball (as built by "make dist")
- **Post-processing**
  - Added experimental ParaView Catalyst co-processing output option
    - developed with the SALOME visualization team
    - see STFC presentation today
  - for CFD_Study
    - update to PARAVIS instead of VISU.

Code_Saturne Version 3.2 (6/6)

- **Programming changes**
  - moved to PyQt API 2 to plan for future Python version upgrades
    - leads to some issues in complex combinations, such as with SALOME, so version 3.3 will add a compatibility layer to handle both API 1 and API 2
  - replaced `propfa` and `propfb` arrays by distinct fields
    - use `field_get_val_...` functions to access values
    - for cell properties, more work remains before `propce` may be removed, but use of field API is recommended to avoid requiring future changes
  - added cs_c_bindings.f90 module for general definitions of C bindings
    - For large modules, it is recommended to use separate files (see field.f90 and post.f90 for example), but for smaller modules, this avoids requiring the definition of specific module files
  - added `cs_field_pointer` API for quick access to main fields from C
  - moved the convection-diffusion balance (`bilsc2.f90`) to C

- **For more details, see NEWS file in:**
  - http://code-saturne.org/viewvc/saturne/branches/Version3_2/

Code_Saturne Version 3.3 (1/2)

- **Version 3.3 to be released late April 2014**
  - Automated test cases will be run
- **Lagrangian module**
  - Improvements in roughness and resuspension models
    - added a user keyword for roughness surface (calculation of the energy barrier in the case of rough wall)
    - consideration of the electrostatic force in the adhesion force for the resuspension
    - mass flux update for particles rolling on the wall
- **Atmospheric module**
  - Add imbrication module (boundary condition coefficients interpolation)
- **Rewrite of temporal moments handling.**
  - Moments handling is now more modular, and allows for variances in addition to means.
  - Numerically stable recurrence relations are used to update moments, whose values are now directly usable at any given time.
    - Weight accumulators are handled inside the module, and not seen as fields anymore.
  - Also, support for user functions is added.
  - Currently, this is mapped to the legacy data setup, and tested only in this context, but the added functionality will be exposed with future changes in case setup.

Code_Saturne Version 3.3 (2/2)

- **Code Architecture**
  - Sharing in C of many Fortran keywords
  - Migration to C of many finite volume operators
    - allows for future increased sharing of code with NEPTUNE_CFD
    - C's local variable declarations allow for safer OpenMP hybrid parallelism deployment
    - handling of structures is much simpler
    - handling of optional arguments is much simpler and safer
      - test for NULL
    - easier for many tools, such as Doxygen and debuggers

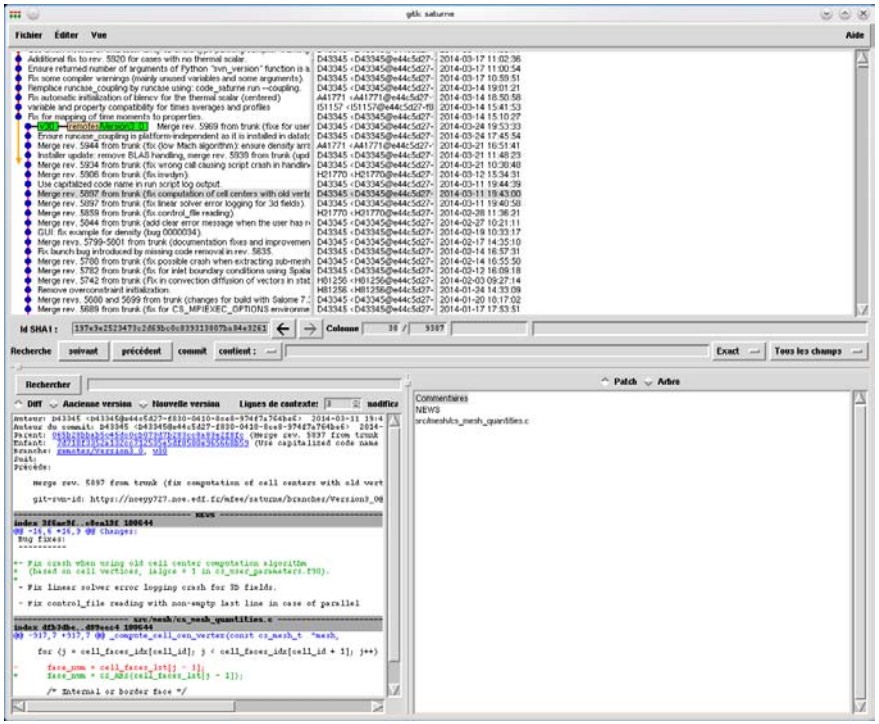
```
const cs_lnum_2_t *i_face_cells = (const cs_lnum_2_t *)m->i_face_cells;
const cs_lnum_t *b_face_cells = (const cs_lnum_t *)m->b_face_cells;

const cs_real_t *cell_vol = fvq->cell_vol;
const cs_real_3_t *diipb = (const cs_real_3_t *)fvq->diipb;
const cs_real_t *b_face_surf = (const cs_real_t *)fvq->b_face_surf;

/* Get physical fields */
const cs_real_t *dt = CS_F_(dt)->val;
const cs_real_t *rho = CS_F_(rho)->val;
const cs_field_t *h = cs_field_by_name_try("enthalpy");

/*----------------------------------------------------------------------------
 * This example computes energy balance relative to enthalpy
 * We assume that we want to compute balances (convective and diffusive)
```

- **For further details, browse**
  - http://code-saturne.org/viewvc/saturne/trunk
- **or checkout the code**
  - git svn clone \ http://code-saturne.org/svn/saturne/trunk \ code_saturne

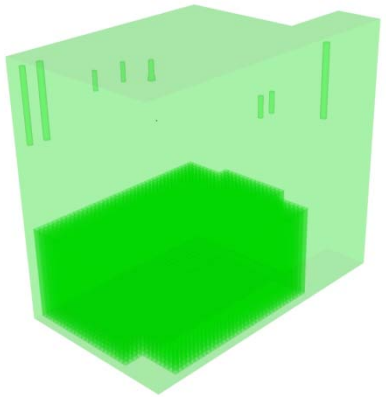


On the road to *Code_Saturne* 4.0

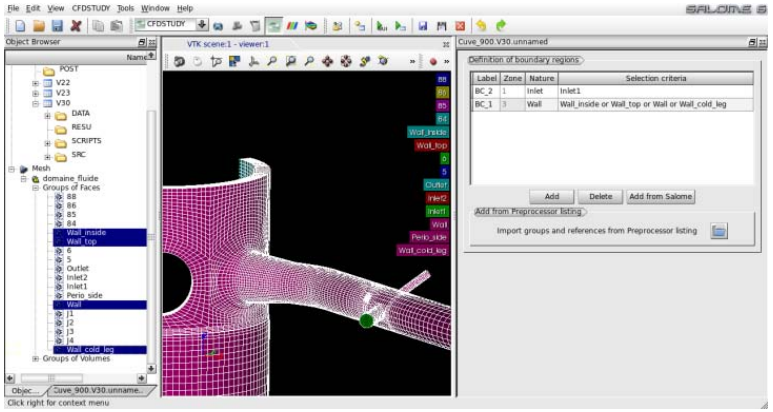
- **4.0 feature freeze with branching of version 4.0 in November 2014**
  - branch will be known as “4.0 beta”, until it is deemed ready for “4.0 release candidate” status
    - all test cases must run successfully to come out of beta
    - snapshots will be released regularly (every 2 weeks or so)
    - when 4.0.0 is released, support for beta and rc versions is discontinued
- **Focus on verification and validation**
  - version 3.0 brought versioned test case setups and automated runs
  - version 4.0 will leverage those tests, which are run frequently in the development process
    - in general, the earlier a bug is detected, the less costly it is to fix it
      - and the less time it has to annoy users
- **Distributed inside EDF as part of SALOME_CFD**

On the road to *Code_Saturne* 4.0

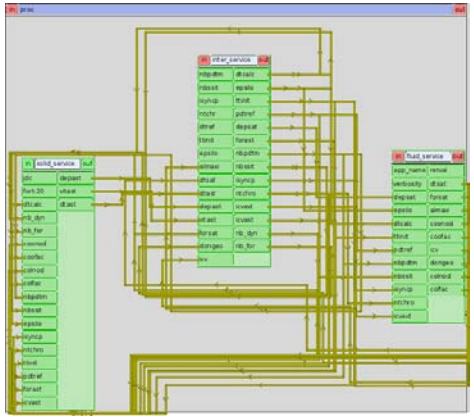
- **Feature list not frozen yet, but should include**
  - postprocessing output improvements
    - rewrite of probes and profiles output, using a consistent writer / submesh paradigm
    - zone-based balance computation and extraction
  - additional physical models
  - additional HPC oriented features
    - optional use of external linear solver packages (targeting PETSC)
    - more cache and thread-friendly mesh numbering
    - deployment of hybrid MPI-OpenMP builds
      - already functional today, but need more systematic testing within the AutoVnV framework
- **And after that ?**
  - Additions to code structure to prepare for new numerical schemes may start appearing shortly after 4.0
    - will require time and effort, so start as early as possible



SALOME_CFD



- ***Code_Saturne* and SYRTHES are already well integrated with the SALOME platform**
  - CFD_STUDY is the main GUI entry point for *Code_Saturne* and NEPTUNE_CFD
  - SYRTHES also has a SALOME component
- **SALOME_CFD aims to be a complete CFD platform**
  - at least SALOME, *Code_Saturne*, SYRTHES
  - NEPTUNE_CFD for EDF internal builds
  - other SALOME modules, such as ADAO and OpenTurns
    - exact list not fixed yet; possibly also JobManager and HOMARD



- **What does this mean for *Code_Saturne* users ?**
  - version releases synchronized (since 2013)
  - For EDF users
    - On workstations, *Code_Saturne* 4 will be distributed as part of SALOME_CFD
      - users will thus immediately have a more complete, preconfigured environment, rather than chasing packages
    - No change for clusters

Organisation

- **New internal EDF quality assurance manual to be released soon**
  - clarifies roles of core development team and contributors
  - builds on identified best practices
    - based on feedback of the last couple of years, versus older projects
- **We realize we ask more of contributors than several years ago**
  - with more users and developments, integration work needs to be spread to more people, or better prepared by contributors.
  - The old way of doing things did not scale
    - at least not without proportionately increasing the development team
    - or integrating unrevised code
      - would lead to rapid accumulation of technical debt and skyrocketing support issues 1 to 3 years later
- **To help with this, we added a 1 day developer's course this year**
  - 1st "test" session in 2014
    - thanks to our participants for their patience for this first try
  - expect 1 session per year
    - we'll be happy to help you make the best use of the code

THANK YOU

Reminder:
best practices to cope with version changes

- **GUI vs. user subroutines**
  - GUI advantage: mostly automatic update from one version to the next
  - User subroutine advantage: slightly less layers, so slightly lower risk of bugs
    - we are progressively aligning user subroutines with the GUI logic so as to make these layers thinner, and avoid GUI translation bugs altogether.
- **Recommended approach**
  - do as much as possible using the GUI, and only the rest using user subroutines
    - for example, for a complex inlet boundary condition, you may define all conditions using the GUI, except the complex one
      - reducing the size of the code that may need updating
      - enhancing its readability
- **Validation test cases are versioned**
  - Updated regularly
  - EDF intranet only