INTRODUCTION

Computational Fluid Dynamics (CFD) has interested professionals who work with fluid flow and heat transfer processes in buildings from a very early stage in its development. Nielsen (1973) showed the potential of the technology whereas Jones and Whittle (1992) signalled how much was already possible with due care. Important specifics of CFD usage in building applications have also been addressed such as the relative merit of different turbulence models (Chen 2009) and, more recently, new areas of research and development are being opened up through CFD (Li and Nielsen 2011). If the former works focused on internal flows, the subject of modelling and simulation of external flows has undergone a similar development, from early papers (Baskaran and Stathopoulos 1989) to later publications that put forward best-practice recommendations (Franke et al. 2004, Tominaga et al. 2008). CFD success stories can be found almost everywhere, but several obstacles remain for an even wider adoption by professionals in the building industry (Marques et al. 2014). High tool costs, lack of access and expertise requirements are amongst those obstacles. To help surpass them, the authors have been working on a software product (blueCFD®-AIR) that is based on a number of Open Source Software products with a dedicated GUI for work process integration. The central CFD component employs OpenFOAM® (Anon.) due to its wide acceptance and quality. This paper details the software structure of the said product. In particular, the next section describes the several components that went into the product, followed by a section where we show results from validation runs. A section of conclusions terminates the paper.

SOFTWARE STACK

Several decisions had to be taken in order to produce a new software that, due to its reliance on an Open Source Software stack, also signals the emergence of a new paradigm in CFD use for building applications. This paradigm tries to facilitate access and use of advanced CFD techniques and models for all types of interested users, thereby removing several obstacles to CFD use such as price, ease of use, lack of sophistication or even, to some extent, expertise requirements. To varying degrees, this paradigm is being built into several building-orientated tools around the world. Figure 1 shows the components employed in our own effort.

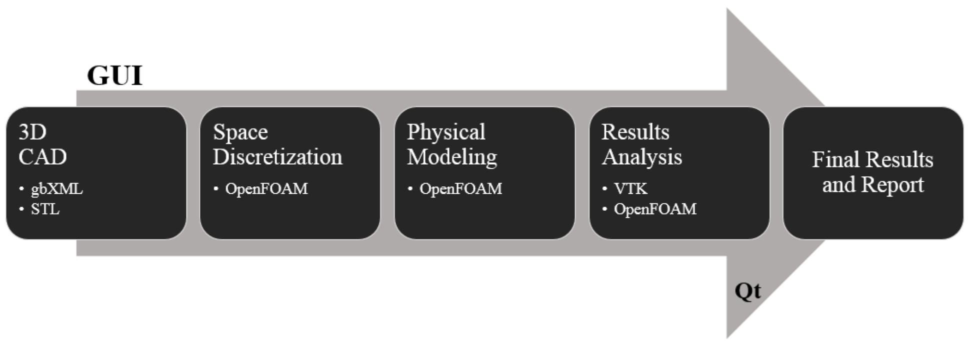

Figure 1. Main steps in simulation methodology and software components.

Work process integration

A tool has to be used within the scope of the work process that exists on the entity where it is deployed. The integration of this tool on the work process should then be as seamless as possible to ensure effectiveness and maximize efficiency. We have tried to meet these goals with: common 3D CAD formats in the building's world for the exchange of geometrical data; a powerful CFD engine that either provides or allows the easy development of physical and numerical models considered relevant in building applications; a GUI that focuses on the needs of the end application; built-in set of physical properties relevant for building applications; capacity to leverage local and cloud computing resources; automation of the GUI use through third-party tools; self-contained post-processing capabilities; automated reporting and, finally, suitable documentation. The user also has the possibility to export his or her model to a native OpenFOAM format for more advanced simulations or post-processing tasks.

CAD interoperability

The exchange of surface definitions with CAD software is done using one of two formats: gbXML (Anon. A) or STL (Anon. B). The gbXML format is open for any proposed changes/additions/improvements and is specifically oriented for Building Information Modelling (BIM). It is therefore the ideal interoperability format for use with other applications in this area of engineering. The main (and possibly only) downsides with gbXML at the present is that it misses specific geometrical details that are essential for CFD, on the one hand, and that, on the other, it has yet to allow the setting of boundary conditions directly in the applications that generate the gbXML files. Only a few XML elements are used from the gbXML format:

- <Building> - to set the rooms names and the main surfaces that define their internal volume. The child XML elements are: <Space>, <Name>, <ShellGeometry>, <ClosedShell>, <PolyLoop>, <CartesianPoint>, <Coordinate>.
- <Surface> - to define the openings that are associated to each room, mostly on the respective walls. The following child elements are used: <Opening>, <PlanarGeometry>, <PolyLoop>, <CartesianPoint>, <Coordinate>.

When importing compatible gbXML files, blueCFD-AIR correlates the internal volumes defined in the shells for each room with the walls defined in the independent surfaces, projecting the openings therein onto the room volumes. This allows the estimation of the wall thickness in order to connect the openings on both sides of the shared walls. However, for more complex geometries, this projecting algorithm sometimes fails. This sets the need to have gbXML produce a non-ambiguous definition of the geometrical structure of rooms, buildings and possibly HVAC ducts, in order to have them properly represented in 3D CAD for a CFD simulation. Beyond this, the ability to define the boundary conditions for each geometrical component directly in the gbXML format would be very welcome, as it would facilitate the integration of CFD directly into CAE/BIM tools, either through a self-contained plugin or an external tool.

STL stands for STereo Litography and is a file format originally designed for 3D printing. Any solid is defined by all of the triangles that describe its surface. The original STL standard has provisions for both binary and coded formats. The binary format is the most compact but it does not allow the definition of more than one solid by name, as opposed to the coded format that does. As such, in our tool we take advantage of the coded format to define the relation between solids directly in their names. For example, we can define that a window is part of a specific room by defining a solid named "|Room" for the room and another named "Room < Window" for the window. This customized variant of the coded form of these files can be manipulated manually or through scripting in order to convert and/or merge several regular STL files into a single file. One such example is the customized STL export plug-in created for SketchUp and provided with blueCFD-AIR. At the present, the STL format is the one where the geometrical model can more easily be represented with little or no flaws, albeit limited to geometrical data only.

Space discretization

Mesh generation (or space discretization) is arguably the single most important step in CFD simulations. Since practicality is also a concern, the mesh generation tool has to: allow its use with scripting; be able to create a discrete representation of the model's characteristics without losing the original edges; generate, or allow the conversion to, a quality mesh that works well in OpenFOAM; be available in a license that allows use and distribution with any type of application, Open Source or not. In the end, snappyHexMesh, a part of OpenFOAM, was chosen for the following reasons: its development is ensured with each major OpenFOAM release; it has the same open-source license as OpenFOAM, which allowed us to fix bugs or adapt it to our needs; it keeps track of mesh quality parameters, therefore maximizing the compatibility with OpenFOAM. snappyHexMesh is not the fastest tool but it has three important characteristics:

- Hexahedral based mesh – specifically for refinement, the base mesh cells are divided into smaller ones simply by splitting them in equal parts along all of the 3 major axis. For example, a level 1 refinement equates to a cell being divided into two over each of the 3 major axis. This enables a flexible control of regions in space with increased or decreased levels of mesh refinement;
- Cell snapping - When cells cut the surface definition of the computational domain, they are projected onto the said surface. This ensures an accurate surface and volumetric representation of the physical space;
- Layer addition - layers of cells can be generated at the surface of the model. This feature has not yet been made available due to higher complexity in automating it. However, for simulating flows such as the ones that concern us, it has proved more beneficial to use a mesh as uniform as possible and rely instead on flexible wall boundary conditions, valid for all y+ values.

Integration of model equations

OpenFOAM (Anon.) is currently the basis for all the CFD components. A description of its core principles can be found in Weller et al. (1998). It implements basic as well as advanced numerical, discretization and algorithmic techniques useful in CFD, particularly in the Finite Volume Method (Ferziger and Perić 2002) context. Our use of OpenFOAM relies on customized solvers for the approximate solution of the steady-state Navier-Stokes equations using the SIMPLE algorithm, with or without the Boussinesq hypotheses (Ferziger and Perić 2002) in case of non-isothermal flow, the latter requiring the solution of the energy conservation equation. The most relevant turbulence models (Chen 2009) are available in OpenFOAM and new ones are easily incorporated such as algebraic models (Chen and Xu 1998). The ever important discretization of convective terms in the transport equations is performed with second order accurate techniques that blend upwinding and Total Variation Diminishing requirements (Ferziger and Perić 2002). The approximate solution of linear systems is mostly performed with Algebraic Multigrid (Ferziger and Perić 2002) for the sake of generality. Additional transport equations for scalars such as age of air or tracers are easily developed and incorporated, the same being true for special boundary conditions such as the ones for partially-open windows or, more importantly, wall boundary conditions valid for all y+ values (Mockett et al. 2012). Due to its extensive set of physical and numerical modelling techniques that matches and even surpasses many commercial codes, OpenFOAM has the potential to spark more advanced simulations in the future, fostering innovation in building design.

Computing resources

Even relatively small cases benefit from runs performed in parallel machines. Since OpenFOAM implements Domain Decomposition parallelism (Anon. C) and employs the Message Passing Interface (Gropp et al. 1996) for communication between the different partitions, it becomes easy to leverage this capability in machines of all sizes, from multicore workstations to departmental clusters and even cloud based services, from the many that have native support for OpenFOAM simulations.

Post-processing and GUI development

The following software toolboxes assist the development of the GUI: Qt (Anon. D) as the framework for the GUI and blueCFD-AIR's infrastructure; VTK (Anon. E) as the framework for manipulation and visualization of both geometrical and results data. These packages are portable to the most common Operating Systems in use.
Qt (pronounced "cute") is employed using the open-source LGPL solution (Anon. F). Since it is a mature and powerful framework, Qt provides all of the necessary tools to create a cross-platform application in C++ with advanced GUI capabilities, including the abilities to locally control processes, interpret XML files and network access, which blueCFD-AIR uses extensively.
VTK stands for Visualization Toolkit and is an open-source cross-platform solution. It

is released under a BSD license (Anon. F), which explains its use in commercial applications. It supports all of blueCFD-AIR's necessities for importing, exporting, handling, manipulating and representing both the geometrical models and the CFD produced data. In addition, it enabled an easy expansion of its features, such as the implementation of our own 3D controls, filters, import and export algorithms. As a downside, VTK might not have the best memory conservation possible. A bird’s eye view of the GUI can be seen in Figure 2.

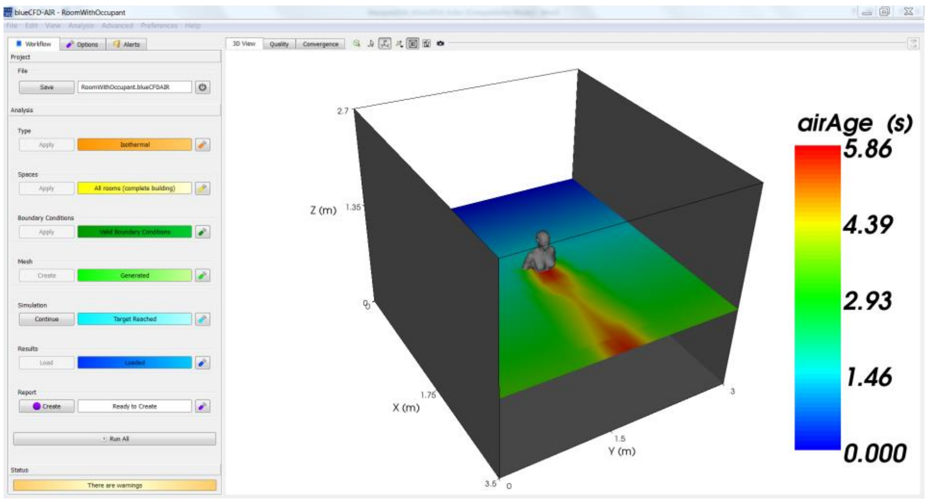

Figure 2. Bird’s eye view of the blueCFD-AIR GUI.

VALIDATION RESULTS

The validation of CFD tools is a critical aspect in the confidence building process that allows their use in real-life engineering problems. In this section we present results for two benchmarks.

Turbulent natural convection in an enclosed tall cavity

A three-dimensional cavity (Betts and Bokhari 2000) was simulated using a uniform mesh with 91520 cells. The flow is assumed to be turbulent, incompressible and non-

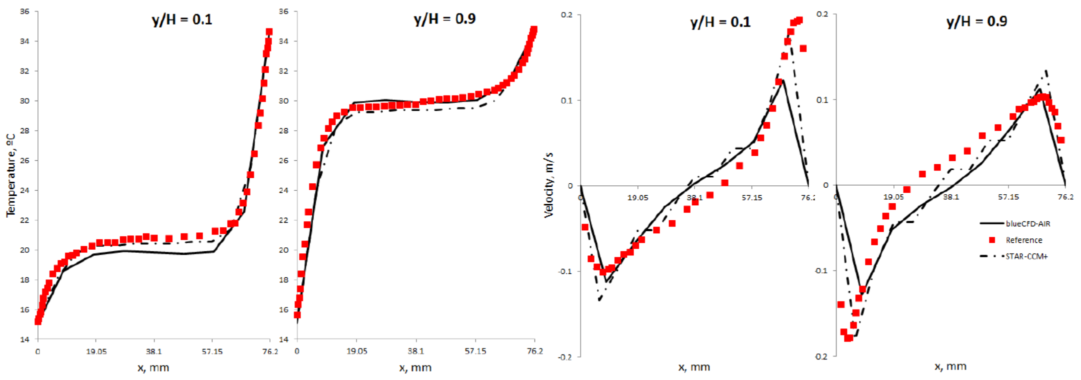

Figure 3. Temperature and vertical velocity across the cavity at different heights.

isothermal. The cold wall is set at 15.1 ºC while the hot wall is at 34.7 ºC. The remaining walls are considered adiabatic. The Rayleigh number is Ra = 8.6E5, the specific heat capacity C_p = 1005 J/kg.K, the thermal conductivity k = 25.3E-3 W/mK, the fluid density ρ = 1.2 Kg/m^3 and dynamic viscosity μ = 1.752E-5 Pa.s. The buoyant-incompressible solver was used, together with the SST k-omega turbulence model with standard wall functions. The results of the simulation are plotted in Figure 3 against the results obtained with STAR-CCM+ (2011) and measured data from the literature (Betts and Bokhari 2000).

Urban landscape

The model described in Tominaga et al. (2008) consists of low-rise buildings with a high-rise building (proportions of 1:1:4) located at the centre of the urban area. Each block is enclosed by four roads. The urban landscape is enclosed in a box with 720 m x 720 m x 720 m. This case is simulated at the lab scale. The three-dimensional non-uniform mesh has 1184314 cells. The flow is assumed to be turbulent, incompressible and isothermal. Inlet values for velocity and turbulent quantities are prescribed according to Tominaga et al. (2008), which also supply reference results.

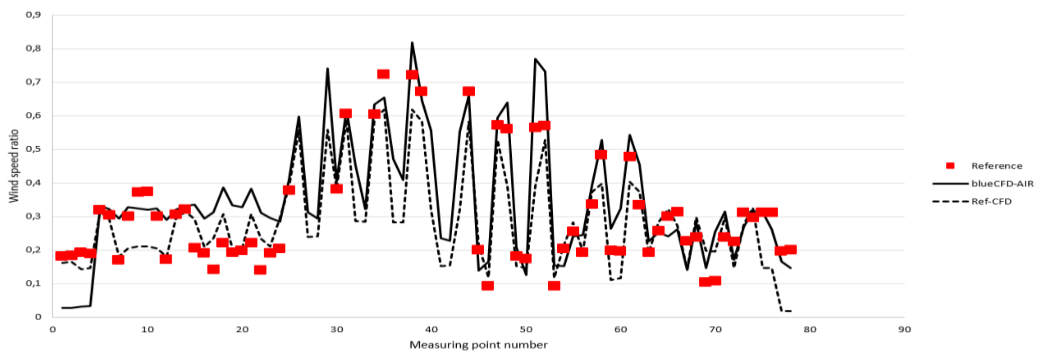

Figure 4. Wind speed ratio at the measuring points for 0º wind direction.

CONCLUSION AND IMPLICATIONS

We have shown in this paper how it was possible to build a viable and effective CFD analysis tool specifically for buildings. The technologies that were employed are described and so are basic validation results. More could be said if space was available. A reasonably powerful and accessible tool, with room for more advanced analysis provided by its reliance on OpenFOAM, it is meant not only to allow the tackling of problems by professionals that would not have the means to do so otherwise, but also to foster the creation of more specialists in CFD.

REFERENCES

Anonymity. http://www.openfoam.org/, accessed May 5 2014.
Anonymity A. http://www.gbxml.org/, accessed May 5 2014.
Anonymity B. http://en.wikipedia.org/wiki/STL_(file_format), accessed May 5 2014.
Anonymity C. http://www.ddm.org, accessed May 5 2014.

Anonymity D. http://qt-project.org, accessed May 5 2014.

Anonymity E. http://www.vtk.org/VTK/project/license.html, accessed May 5 2014.

Anonymity F. http://www.gnu.org/licenses/gpl.html, accessed May 5 2014.

Baskaran, A. and Stathopoulos, T. 1989. Computational evaluation of wind effects on buildings. *Building and Environment*, Vol.24, pp.325-333.

Betts, P.L. and Bokhari, I.H. 2000. Experiments on turbulent natural convection in an enclosed tall cavity, *International Journal of Heat and Fluid Flow*, Vol.21, pp.675-683.

Chen, Q. and Xu, W. 1998. A zero-equation turbulence model for indoor airflow simulation. *Energy and Buildings*, Vol. 28, pp.137-144.

Chen, Q. 2009. Ventilation performance prediction for buildings: a method overview and recent applications, *Building Environment*, Vol.44, pp.848–858.

Ferziger, J. and Perić, M. 2002. *Computational Methods for Fluid Dynamics, 3rd Ed.* Berlin: Springer-Verlag.

Franke, J., Hirsch, C., Jensen, A. G., Krüs, H. W., Schatzmann, M., Westbury, P. S., Miles, S.D., Wisse, J.A., and Wright, N. G. 2004, May. Recommendations on the use of CFD in wind engineering. In Cost Action C (Vol. 14, p. C1).

Gropp, W., Lusk, E., Doss, N., and Skjellum, A. 1996. A high-performance, portable implementation of the MPI message passing interface standard. *Parallel computing*, Vol.22, pp.789-828.

Jones, P. J. and Whittle, G. E. 1992. Computational fluid dynamics for building air flow prediction—current status and capabilities. *Building and Environment*, Vol.27, pp.321-338.

Li, Y. and Nielsen, P. V. 2011. CFD and ventilation research. *Indoor Air*, Vol.21, pp.442-453.

Marques, N., Santos, B., and Ramalho, S. 2014. Efficient and Effective BIM to CFD, *ASHRAE/IBPSA-USA Building Simulation Conference*.

Mockett, C., Fuchs, M., and Thiele, F. 2012. Progress in DES for wall-modelled LES of complex internal flows, *Computers and Fluids*, Vol.65, pp.44-55.

Nielsen, P. V. 1973. Berechnung der Luftbewegung in einem zwangsbelüfteten Raum. *GI-Gesundheits Ingenieur*, Vol.94, pp.299-302.

Nielsen, P. V. 2004. Computational fluid dynamics and room air movement. *Indoor Air*, Vol.14, pp.134-143.

Star-CCM+ Version 6.04.014 - User Guide. 2011. CD-adapco Inc.

Tominaga, Y., Mochida, A., Yoshie, R., Kataoka, H., Nozu, T., Yoshikawa, M., and Shirasawa, T. 2008. AIJ guidelines for practical applications of CFD to pedestrian wind environment around buildings, *Journal of wind engineering and industrial aerodynamics*, Vol.96(10), pp.1749-1761.

Weller, H. G., Tabor, G., Jasak, H., and Fureby, C. 1998. A tensorial approach to computational continuum mechanics using object-oriented techniques. *Computers in Physics*, Vol.12, pp.620-631.