A platform to collaborate around CFD simulations

Claudio Gargiulo
R&D - Aerothermal CFD
Fiat Chrysler Automobiles, Italy
Email: claudio.gargiulo@fiat.com

Donato Pirozzi
ISISLab, Dip. di Informatica
Università di Salerno, Italy
Email: dpirozzi@unisa.it

Vittorio Scarano
ISISLab, Dip. di Informatica
Università di Salerno, Italy
Email: vitsca@dia.unisa.it

Giuseppe Valentino
ISISLab, Dip. di Informatica
Università di Salerno, Italy
Email: giuval@me.com

***Abstract*—This paper describes Floasys, a web-based platform to foster the collaboration among Computational Fluid Dynamics analysts and to promote model reuse by centrally managing simulation data and providing metadata annotations and search functionality over them. In this way, CFD analysts access to simulation data and results performed by different engineers and are able to leverage on them to make the right design decisions.**
***Index Terms*—Data sharing, model sharing, stakeholders communication, CFD simulators integration, Web-based platform, Tools integration, simulation tagging, simulation search, simulation data version control.**

I. Introduction

Nowadays, SMEs and large industries extensively use simulations to design new products. Products have become even more complex, they integrate many components (e.g. more than 20000 separate components for automotive product [1]) and are available to customers in many configurations. To manage this complexity, to get "*a better insight into product behaviour*" [2] and to reduce costs for prototypes [3], industries simulate products with Computational Fluid Dynamics (CFD) simulations that enable the investigation of physical product behaviour. In addition, SMEs and large industries have many locations, so dispersed teams need to collaborate together and share knowledge to design products. Also co-located engineers who work in the same room need to communicate sharing simulation, know-how, best practices and other information.

The paper introduces Floasys, a web-based platform designed to support data, knowledge and result sharing among CFD analysts in Fiat Chrysler Automobiles (FCA). The aim is to promote the model reuse and the result sharing to make design decisions. Floasys collects and centralizes data from CFD simulators integrating their functionality and data in a user-friendly GUI. Floasys provides a tool to search data independently by the specific simulator. The search tool is very useful to get simulations performed by different engineers to compare performance about multiple design revisions. In addition, it allows the data sharing through URLs exchange. The Floasys targets are industries who use CFD simulators to design their products. One of the main requirements is the integration of validated and widely used CFD simulators.

The paper is organized as follows. Section II describes the industrial FCA use case about CFD simulations and the need to share data and collaborate. This section also reports the stakeholders Functional and Non-Functional Requirements (NFR). These have led the Floasys platform design in terms of Front-End (Section III), architecture and engineers tools (Section IV).

II. FCA Use case

This section introduces Fiat Chrysler Automobiles use case and the stakeholders requirements about a platform to foster the collaboration among engineers, promote data sharing and automate repetitive and error-prone tasks. Some space is allocated to describe the Product Development Process (PDP), the CFD Workflow and the internal industrial organization.

A. Product Development Process and CFD Workflow

Manufacturers aim is to bring products on market quickly within the budget and performance constraints [2]. The PDP describes the process adopted to design, develop and bring products on the market [4] and involves a continuous information exchange among many tasks. Large industries are dislocated and organized in multiple organization structures of different types. One type of structure is the functional area. Functional areas have technical know-how about a specific sector (i.e. engineering, cost engineering, marketing, commercial). The CFD unit is the engineering functional area with highly skilled engineers who perform simulations to analyze the aerodynamic and aerothermal automotive product behaviour. Automotive designers, CFD analysts and performance engineers collaborate together to design the vehicle products. This use case focuses on the CFD functional area and its relationships with the PDP. CFD is a numerical computer simulation able to solve and analyze problems that involve the fluid flow and other related physical phenomena. CFD is widely adopted in many industrial sectors such as automotive, aerospace, high-tech and chemical sectors. CFD benefits are a "*better insight into product behavior*" [2], product optimization in according to the performance objectives, the simulation of extreme environmental conditions (i.e. low or high temperatures) and a cost reduction due less number physical prototypes. Experimental tests, on the other hand, with real prototypes are very expensive (i.e. wind tunnel infrastructure).

Style designers create the exterior and interior product design with manual drawings that will become 3D models using CAD software. CFD engineers use the 3D model to simulate and analyze the product performances (e.g. aerodynamics, aerothermal, aeroacoustic, air conditioning and cabin climatization) with CFD simulators. CFD analysts perform simulations and report data in documents. In order to meet

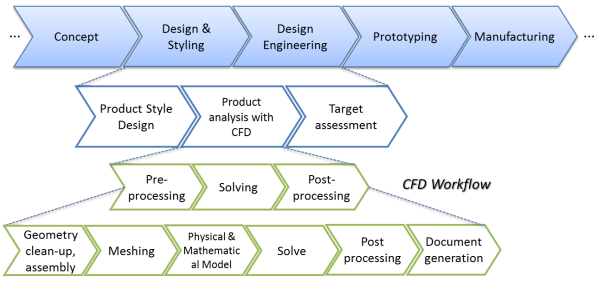

Fig. 1: Product Development Process and CFD Workflow.

the engineering targets, performance engineers use simulation results to decide the changes to make and constraints for the next style revisions (style constraints). At this stage, engineers decide which prototypes to build and test in the wind tunnel infrastructure. Finally, experimental data are correlated to numerical simulation data, and additional style constraints are defined in according to the performance goals. CFD Workflow is iterative and consists of three phases (Fig. 1): pre-processing, solving and post-processing. In the pre-processing phase, CFD analysts take the vehicle geometries from stylists, and perform clean-up and meshing (both surface and volume mesh) tasks. Vehicle geometry is inserted into a virtual wind tunnel. So, CFD analysts define the geometric and physical-mathematical models to simulate. The physical-mathematical model contains the solid materials and fluid flow characteristics as well as the boundary and initial conditions. Volume mesh is a spatial discrete representation of the geometric domain. At each simulation step, physical values (i.e. velocity and pressure) are computed for each mesh cell. The size, shape and number of volume cells determine how many time and how many computational resources (e.g. the number of processors) are required by the simulation. Solving phase consists in running model simulation using HPC resources. The tuning of CFD simulations is a time consuming task because geometries are complex and the number of parameters to set is high. The simulation running takes about several hours (currently up to 12 hours with at least 40 processors). It is very important to monitor the running simulation to check periodically the simulation convergence. CFD analysts monitor the residual and the physical quantities about the examined phenomena (i.e. the pressure forces under the vehicle body). In post-processing, CFD analysts use simulation results to create documents about the simulated product. Simulation results are tabular data, contour-plots and streamline images. The document creation (e.g. spreadsheets and slides) requires manual copy-and-paste operations to obtain artifacts compliant to the industrial templates.

B. Stakeholders Requirements

Requirements elicitation activity is performed using interviews and surveys involving performance engineers, technical manager and CFD analysts. Obviously, stakeholders have different requirements. Performance engineers and technical manager ask management features such as the opportunity to monitor resources, projects timeline and goals. On the other hand, CFD analysts, who perform simulations, require engineering features (e.g. simulation monitoring, automatic document generation) and aim to share and exchange both simulation data and results. In order to support collaboration among engineers (Fig. 2) and to make the right design decisions, they must access to centrally available simulation data. Data sharing is required in many design phases and with different granularity. Performance engineers and technical manager need aggregate data (e.g. statistical data, trends about performance) while CFD analysts need access to simulation data (e.g. model, simulation case). In addition, to further foster the model reuse, an advanced search tool is suggested. In this way CFD analysts perform simulations starting by previous works changing some parameters or some geometry components.

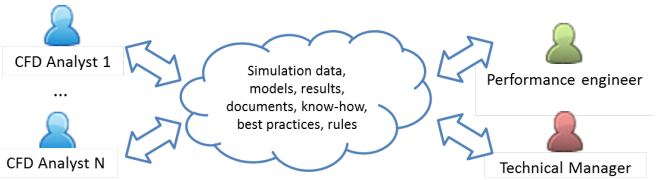

Fig. 2: CFD functional area stakeholders.

This section outlines the requirements pointed out by CFD analysts. Some requirements and the issues that we are facing in the automotive context seem to be very similar to what is depicted in market researches [2] about other sectors that use CFD. So, we are confident that many of our findings and proposed solutions can be take into account and adopted also in other sectors (i.e. aerospace, high tech). The market research "*Getting Product Design Right the First Time with CFD*" [2] has interviewed 704 companies who use CFD to design and develop products impacted by fluid flow. Aberdeen Group has introduced a performance index used to classify companies in three groups: laggard, industry average and best-in-class. Industries aim to centrally manage results and to "*use CFD results to improve collaboration between design engineers, R&D engineers, and analysts*" [2]. Idea is that to support collaboration between engineers and analysts they must have access to CFD results and leverage on them to make design decisions. The following is a detailed description of the requirements gathered by our analysis.

1) Centralize Simulation data and results: In according to Aberdeen Group market research [2], industries aim to centralize simulation results. Our goal is to centralize not only the results but all simulation data such as 3D geometries, simulation setup, results and documents allowing their easy retrieval. Industries already centralize the files using network shared folders without impose any rules on how to store them. The difficulties to centralize data concern mainly the single simulation file size (each one takes up to ten gigabytes) and the high number of files (based on our internal survey each engineer performs at least two hundred simulations per year). Actually, to save space many old simulation files are removed

from the shared network folder and are stored only on backup devices, creating some issues on their retrieval. The main issue concerns the 3D geometry and meshing, to face these issues our aim is to store only the surface meshing and rebuild volume mesh from it basing on simulation case setup. Our idea is to provide a reduced 3D model, so engineers could display and identify instantly which components are used to perform simulations (e.g. spoiler presence).

2) Provide metadata over simulation data: The need, here, is to store additional metadata over simulation data and use them to provide other services. CFD simulators do not consider this aspect so engineers can not store additional data such as project name, revision and so. Actually, to overcome this issue, engineers use simulation file name to encode metadata.

3) Provide searching facility over simulation data and metadata: Search functionality both for data and metadata are mandatory due the huge amount of data. Actually, simulators do not provide search functionality and existing operating system search tools are not so useful because the most useful data are stored in a closed simulation file format. This requirement is also related to avoid vendor lock-in requirement (see below in requirement II-B7) because in order to provide search functionality data must be stored in an open and neutral format (i.e. XML). One query could be the opportunity to search simulation data about a specific project, brand and revision.

4) Support data sharing: The centralization of data and their retrieval are initial steps towards a collaborative CFD platform. CFD analysts need a mechanism to exchange references about data. On Internet a common way to share resources is exchanging URLs. So, our idea is to identify univocally simulation data with URL and use them to share data among engineers. An important aspect of this technique is *who can see what data*. Multiple industrial roles exists (Fig. 2), so an access control is important to control the sharing of confidential data.

5) Provide version control over data: The requirement to provide version control over the simulation data is reported in literature [2]. The main difficulty concerns the version control of 3D geometry data. To give an idea, now each simulation file takes up to ten gigabytes due to huge geometry data.

6) Integrate multiple industrial CFD Simulators: CFD analysts use multiple CFD simulators. They have validated these software over the years and analysts are confident with them. So, it is important to support and collect data from multiple daily used CFD Simulators. This is the key difference with other educational platforms that integrate simplified or in-house developed solvers. In the analyzed context, engineers use mainly a proprietary CD-adapco Star-CCM+ and an open source OpenFOAM software. For some products both design and simulations can be performed by the same team using an all-in-one integrated CAD-CFD software. Automotive products are complex since they integrate mechanical components, embedded systems, as well as electronics devices with many configurations. In addition, to design an automotive product, many teams are involved. Styling, CAD design and simulations are performed by different teams. Each team has its know-how and focuses on the design of a specific product feature.

7) Avoid Vendor Lock-In and Data Lock-In: The *support of multiple industrial CFD simulators within the platform* (requirement II-B6) consists in the integration of industrial open source and closed commercial CFD simulators. The main issue is the potential Vendor Lock-In both in terms of services and data. Vendor Lock-In is a well-known Anti-Pattern [5]: the phenomenon that causes customer dependency on given vendor software about a specific good or service [6] with high switching costs [7]. Data Lock-In concerns the data and occurs when the only way to access to data is using the vendor software. For instance when the platform must access to a closed file format and the vendor software does not have export functionality to open standard. Data Lock-In is very common in Cloud Environments [8] and is an obstacle to cloud computing [9]. Vendors lock users in to make harder for them to leave product because they cannot get their data. Stakeholders aim to integrate existing CFD simulators but at same time it is very important to avoid the Vendor Lock-In.

III. FLOASYS

In order to meet the stakeholders requirements, the proposed solution centralizes simulation data (requirement II-B1), provides functionality to add metadata over simulation data (requirement II-B2) and finally has a structured and assisted Search tool to get simulations performed by different engineers (requirement II-B3). The proposed features are designed and developed over the Floasys architecture. In particular the added tools are: the repository tool to navigate the simulation data repositories, a Search tool to get the simulations performed by different analysts. Floasys is a web-based platform to support CFD engineering tasks in FCA. Floasys was designed to be modular and extensible to meet the future needs.

A. Front-end

Floasys provides a re-configurable web-based GUI based on Perspectives and Views concepts provided by Eclipse Remote Application Platform (RAP) [10]. Our idea is that the virtual workbench GUI changes according to the engineering tasks. In this way, the system is able to show only the functionality more relevant to perform the task. A *perspective* is a specific configuration of the workbench and contains many views to show information. A perspective provides a well-organized software functionality access because they divide them into semantically homogeneous sections.

1) System independent Repository Tool: It allows the repositories navigation and simulations selection. The Repository tool is able to handle simulation data from different simulators using the Floasys framework services. Floasys supports multiple simulators: the first difference to face is how they store simulation data. For instance, OpenFOAM stores data in a well-defined directories structure consisting in three folders (e.g. system, constant and iteration directories) and data are stored in multiple files. Instead, Star-CCM+ stores all simulation data in one single-vendor format file. OpenFOAM files are plain text readable without the software, instead Star-CCM+ files are in closed format and they can be read only

through the vendor software. Repository tool inherits the user file system access permissions, so the logged user can access only to the files he/she is authorized. Floasys can access to network folder through a server using a SSH connection with the logged user credentials.

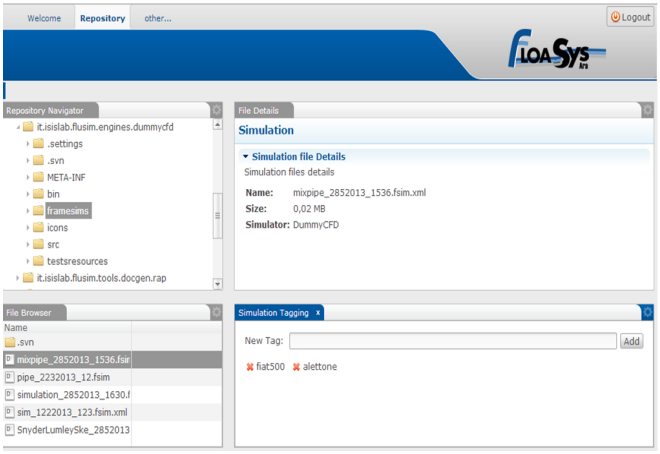

Fig. 3: Repository tool with the tagging file operation.

2) Tagging features: CFD analysts perform a lot of simulations per year storing them in the repositories. By using an internal survey and interviews we asked question about CFD Analysts simulation file organization. Actually, existing tools do not support the efficient search of simulation data, not even the operating systems search tools, simply because simulation are usually in a closed binary file format. In addition, survey responders pointed out that simulations do not contain desired searching information. For instance, simulations are about a specific brand, project name, revision etc., information that can not be stored within the simulation files. These additional information are very useful to retrieve the simulations especially to share data among engineers. Our idea is to provide the tagging feature to enrich simulation files with metadata (i.e. project, brand, and free tags). In this way, our system meets requirement II-B2. The tagging feature is available in the Repository tool. CFD analysts can annotate the selected files with free tags or recommended tags (Fig. 3). The tag operations are both unstructured with free tags and structured tags taken by the forms shown in Fig. 4.

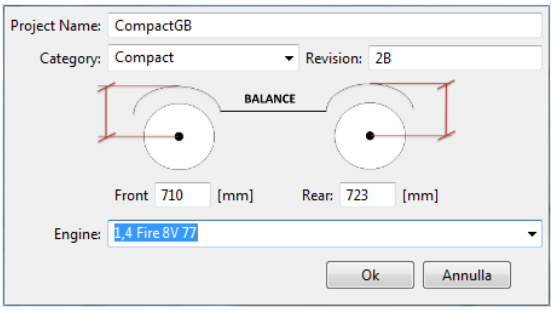

Fig. 4: An example of structured data to store.

3) Search tool: Floasys Search tool provides functionality to search simulation data stored in the repositories and to get files with found data. Both tags and simulation data are used to search files so the tool meets requirement II-B3. In order to avoid data lock-in and to manage the search over closed file format, we decide to extract some other important simulation data (e.g. the names of components, simulation parameters) and to store them in XML files. In this way, the search operation is faster because it does not need the direct access to the closed files format and it does not require to open the simulation file using the proprietary software. Every time the analyst opens a simulation through Floasys, the platform automatically extracts the simulation data storing them in open format. The data extraction is already required to support the engineering tasks. The Search Tool (Fig. 5) is another Floasys perspective useful to perform searches inserting the keywords. The system performs the search by using indexed data and simulation file names displaying results in a list. At same time, Floasys recommends further keywords to refine the search (Fig. 5). End-user can select a listed simulation to display the revisions history. In this way, the tool supports the search activity suggesting further search keys to reduce the total number of potential results.

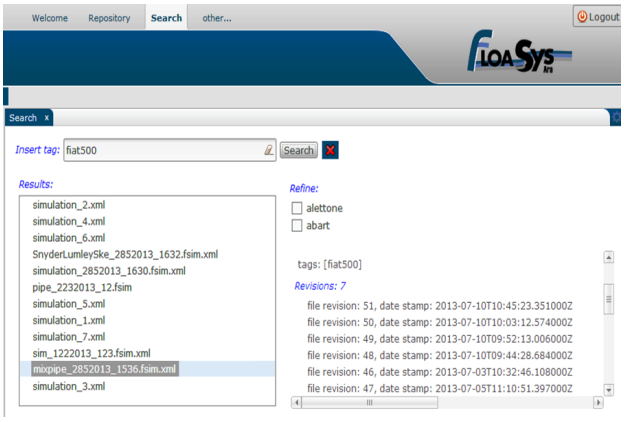

Fig. 5: Search Tool

4) Data sharing: Each simulation file has a unique ID within the platform, all relevant data are linked to this ID such as documents, simplified 3D geometry, surface mesh and so on. Both repository and search tools provide a unique URL for each selected simulation. Our idea is to share data simply by sharing the unique reference to the specific simulation data. Now, URLs identify simulation data and inherits the file system permissions. The URL is private and is accessible only within the industry boundaries. Considering the Computer Supported Cooperative Work (CSCW) space-time quadrants [11], Floasys supports the *asynchronous* data sharing both for dislocated and co-located teams.

5) Web-based 3D Model Visualization: Floasys shows a reduced 3D geometries of the simulated vehicle. With this tool engineers have a quick feedback on which components are used to simulate the product without the CAD software. It has a list of components with their Property IDs (PID). The

user can activate or deactivate some parts and can perform the basic zoom and pan operations. The 3D vehicle geometries usually are very complex. To give an idea, each geometric model takes up ten gigabytes and engineers use very performing hardware to open and manipulate them. An important requirement for any engineering platform is the visualization of 3D geometric. As many other platforms, Floasys is web-based. The vehicle geometries are impossible to render in the browsers using WebGL because they are very detailed and heavy; also the quantity of data to transfer from server to client is very huge. To overcome this common issue and considering that the geometric representation is useful to give an immediate feedback on which components are included in the simulation, Floasys generates a simplified geometry representation to render in the browser. Engineers need to have numerical tabular data, contour-plots and the 3D geometric model in the same view. Floasys provides a reduced geometry visualization so engineers can quickly check which are the vehicle components at a glance. For instance, an engineer can quickly visual check if the vehicle is simulated with the spoiler.

IV. Architecture

This section covers the architectural solution to centralize data and to provide tagging and search tools as well as data sharing. In order to meet the stakeholders requirements the main issues to face are the proprietary nature of the CFD software, the closed file formats and the huge amount of data. Our aim is to centralize data (e.g. 3D geometry, model, results, documents) and provide services over them (e.g. metadata and search functionality) so it is important to access simulation data independently by the CFD simulators. CFD Analysts use proprietary simulators that generate closed file format and do not have export functionality in open format (e.g. XML). For instance, exporting to standard format is a well-established functionality for geometry data (e.g. in STL format) but is not available for the entire simulation case, setup and parameters.

A. Data centralization, tagging and search

For each simulation, Floasys generates an XML file to store simulation data and metadata. The XML files are centrally managed, are indexed to have high search performances and are linked to the original simulations. In addition, both tagging and search features are independent by the specific simulator avoiding Vendor Data Lock-In. Every time an end-user uses Floasys, it extract all useful data storing them in XML format, independently by the specific software. In this way, we face the issue that simulator files are in closed file format by storing data in open format. Our idea is to perform data search using only data stored within XML files using Apache Solr [12]. Each XML file is linked with the original simulation file using an unique ID. So, the search operation is performed over the XML file and then, when requested by end-user following the link to the original simulation file. Two Java libraries are used (Fig. 8): SolrJ to interact with the Solr Server and SVNKit to commit and update data to SVN repository. Our solution meets also other industrial constraints, such as the

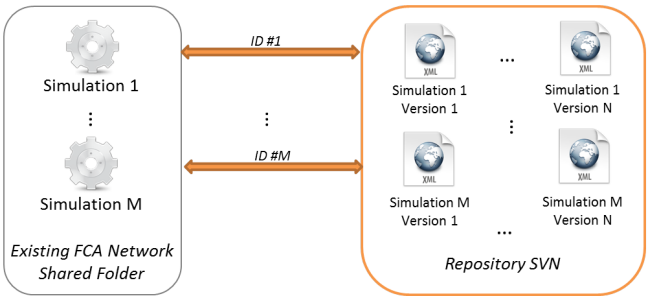

Fig. 6: Simulation Data versioning.

impossibility to move existing files and folders or to store them within a database. Finally, the solution must be independent by the specific simulator, so it can not store metadata within the simulation files, also because files are in closed file format.

B. Simulation Data Version Control

Simulations are stored in closed file format and contain detailed and heavy vehicle geometries, so it is hard to provide version control directly over these files. Our solution extracts data from the closed files and stores them in open format (e.g. XML). It extracts also the original surface mesh (STL format) avoiding to store the volume mesh (the most heavy part created form the surface mesh) and a simplified 3D geometry. Floasys provides the version control over the XML files. The store of surface mesh reduces the overall required amount of storage. In addition, though it happens rarely, CFD Analysts can use the stored surface mesh to run the simulation again. After many attempts the best trade-off between running time and the 3D geometry quality is to use the Matlab `reducepatch` command. So, in the post-processing phase Floasys in batch connects to Matlab server and reduce the original STL file creating the lightweight version. This simplified version contains all vehicle parts separately. Proposed solution has an interesting advantage. XML files store the most important and useful simulation data including a simplified 3D geometry. So, users can open the XML files using the repository tool and access to all simulation data without the original software and without the HPC resources. It is a useful feature because sometimes CFD analysts need to open simulations to consult data, in this way no proprietary software license nor HPC resources are used. Floasys interacts with Matlab as a black box, it gives in input the original mesh and gets in output the simplified mesh, so in future we could replace Matlab with another system. For each simulation file (left-side of Fig. 7) an XML file exists in the SVN repository (right-side of Fig. 7) that contains extracted simulation data and metadata. The linking between simulation files and XMLs metadata is very important to retrieve the original simulation file, and is created by using a unique ID. Floasys generates a unique ID for each simulation file and stores it with metadata in the XML file. The ID is based on the original simulation file content and path. This solution has the following advantages. Floasys does not change the simulation file content to add other information such as the ID. It performs search operations

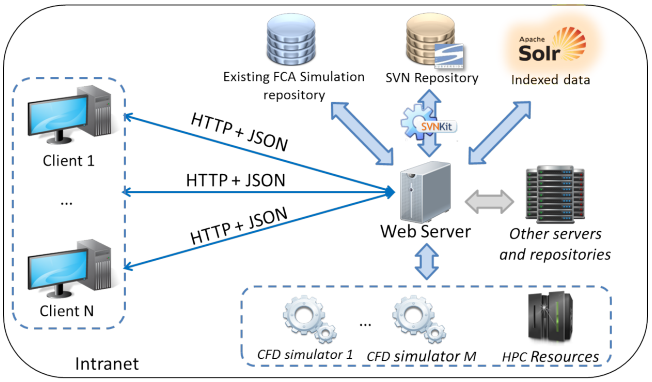

Fig. 7: Floasys Client/Server Architecture.

using indexed XML content getting high performances and providing version control for them. Other alternative were discarded such as to add metadata directly to simulation files avoiding the creation of XML files. The drawbacks in this case are: 1) it is difficult to find available and unused fields in the simulation files; 2) the simulation files are stored in closed file format, so the solution is vendor software specific; 3) the metadata management requires the access to files through the vendor software using HPC resources due the geometry data and 4) it is difficult to provide version control over simulation files because they takes up to ten gigabytes.

C. Floasys Platform Architecture

Floasys is based on a Client/Server architecture (Fig. 8) developed using Eclipse Remote Application Platform (RAP) [10]. Floasys is Intranet-based for security reasons. It could be exposed also on Internet, but limitations exists such as the huge amount of simulation data (gigabytes) to transfer. In addition, trusting and security issues must be taken into account to avoid espionage. Industries usually have their own repositories for simulation data (i.e. shared network folders) that must be used by the platform. The architecture needs an additional repository to store metadata (req. II-B2) about simulation data. The repository can be an internal SVN server or a shared network folder (without the version control support). The server-side software architecture [13] follows a three layers approach. The tools are in the top layer and contribute both in Front-end and functionality. A CFD tool is an end-user GUI that provides one or more useful functionality to engineers. Each tool performs a well-defined engineering task. Some tools depend on the internal team organization and business process. For example, document generator tool accepts document templates and creates documents from them. The architecture has an isolation layer that isolates the front end tools from the CFD software. The isolation layer has multiple aims: it provides common service APIs to the front-end managing the simulators differences, it decouples front-end from the simulators wrappers and it allows the automatic or manual simulators selection that are able to provide the needed services. Finally, the bottom layer consists of CFD software wrappers. The server-side architecture is based on a pure plug-in architecture in which everything is a plug-in [14]. Each box of the software architecture is a plug-in. To achieve extensibility and modularity requirements, we choose a pure plug-in architecture because it supports the extensibility by plug-ins. Every plug-in provides well-defined hook points called *extension points* that describe the way to extend the plug-in's functionality. Other plug-ins can add new functionalities by implementing an extension point. A plug-in can be modified or replaced by another equivalent implementation. For instance, Floasys framework defines extension points (hook points) to extend its functionality by providing other simulator wrappers or Front-end plug-ins.

V. Future Works

Planned future work aims to support simulation data sharing among groups of engineers providing more control over the shared data. For instance, it is important for the technical manager to control what data are exchanged and reconfigure which groups must exchange data.

References

[1] D. Sörensen, *The automotive development process*. Springer, 2006.

[2] C. K.-R. Michelle Boucher, "Getting Product Design Right the First Time with CFD," 2011.

[3] D. H. Michelle Boucher, "Engineering Envolved: Getting Mechatronics Performance Right The First Time," 2008.

[4] J. Weber, *Automotive Development Process*. Springer, 2009.

[5] W. H. Brown, R. C. Malveau, and T. J. Mowbray, "AntiPatterns: refactoring software, architectures, and projects in crisis," 1998.

[6] M. Perry and T. Margoni, "Floss for the canadian public sector: open democracy," in *Digital Society, 2010. ICDS'10. Fourth International Conference on*. IEEE, 2010, pp. 294–300.

[7] R. Shah, J. Kesan, and A. Kennis, "Lessons for open standard policies: a case study of the Massachusetts experience," in *Proc. of the 1st inter. conf. on Theory and practice of electronic governance*, 2007.

[8] S. Sakr, A. Liu, D. M. Batista, and M. Alomari, "A survey of large scale data management approaches in cloud environments," *Communications Surveys & Tutorials, IEEE*, vol. 13, no. 3, pp. 311–336, 2011.

[9] C.-W. Chang, P. Liu, and J.-J. Wu, "Probability-based cloud storage providers selection algorithms with maximum availability," in *Parallel Processing (ICPP), 2012 41st International Conference on*. IEEE, 2012.

[10] "Eclipse rap remote application platform," "http://eclipse.org/rap/", last checked on 03/02/2013.

[11] J. Rama and J. Bishop, "A survey and comparison of cscw groupware applications," in *Proc. of the 2006 annual research conf. of the South African institute of computer scientists and information technologists on IT research in developing countries*, 2006.

[12] Solr, "Apache Solr," "https://lucene.apache.org/solr/", check 09/04/2014.

[13] C. Gargiulo, D. Pirozzi, V. Scarano, "An architecture for CFD Workflow Management," *IEEE conference on Industrial Informatics*, 2013.

[14] D. Birsan, "On plug-ins and extensible architectures," *Queue*, vol. 3, no. 2, pp. 40–46, Mar. 2005.