DOI 10.12694/scpe.v15i4.1053
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 15, Number 4, pp. 309–329. http://www.scpe.org © 2014 SCPE

SIMULATION DATA SHARING TO FOSTER TEAMWORK COLLABORATION

CLAUDIO GARGIULO[*], DELFINA MALANDRINO[†], DONATO PIROZZI[‡] AND VITTORIO SCARANO[§]

Abstract.
This paper introduces Floasys, a Web-based platform to foster the collaboration among engineers involved in Computational Fluid Dynamics (CFD) simulations. The platform has been designed around the simulation data, i.e., fostering and stimulating sharing, re-use and aggregation of models, simulation results and engineers annotations. Floasys requirements come directly from an extensive requirements study that we conducted with two different teams in Automobiles (FCA), geographically distributed, who daily perform an intense activity of CFD simulations to design vehicle products. Collaborative requirements were gathered through stakeholders' interviews and a user survey. We describe, first, Functional and Non-Functional requirements as suggested by relevant literature (both in scientific and industrial setting) and by the user survey performed within FCA teams. Then, we show Floasys functionalities and its architecture, that is based on a centrally managed repository of simulation data. By enriching the repository with metadata annotations, Floasys provides all the desired functionalities to allow CFD analysts an easy and immediate access to simulation data and results performed within the teams so that they can leverage them to make the right design decisions.

In this paper, we were able to (1) identify key collaborative requirements for CFD design, (2) address each of them with an integrated, extensible and modular architecture, (3) implement a working industrial prototype (currently under testing and evaluation in a real setting like FCA), and (4) identify the possible extensions to different contexts (like aeronautic, rail and naval sectors).

Key words: Data sharing, model sharing, collaboration, simulation survey, CFD simulators integration, Web-based simulation, simulation tagging, simulation search, simulation data version control.

AMS subject classifications. 68U20

1. Introduction.

Nowadays, SMEs and large industries extensively use simulations to design new products. Products became 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 use different types of computer simulations [3] to simulate and analyse the products behaviour. One type of simulation is Computational Fluid Dynamics (CFD) used to investigate the physical product behaviour, such as external aerodynamics, underhood cooling, air conditioning and so on. In addition, SMEs and large industries have many locations, therefore, both co-located and geographically distributed engineers need to collaborate together, share simulation models, know-how, best practices and other important information.

This paper introduces Floasys, a Web-based platform designed to support simulation data, knowledge and result sharing among CFD analysts in Fiat Chrysler Automobiles (FCA). The goal is **to promote the sharing of simulation models and results to foster their reuse** among engineers. This work introduces, analyses and discusses Functional and Non-Functional collaborative requirements (Section 2) as suggested by relevant literature (both in scientific and industrial setting) and by results of an extensive user survey performed within FCA teams. The collaborative requirements are: *simulation data centralisation, metadata over simulation data, search facility, version control over data and data sharing.* Functional and Non-Functional requirements led the design of Floasys' architecture and its functionalities. Floasys collects and centralises simulation data over time. Simulation data are collected from multiple simulators and are stored in open format (e.g., XML). Floasys provides additional services over collected simulation data. It provides a Search tool that is independent by the specific simulator. It 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 target customers are industries who use CFD simulators to design their products. From architectural point of view, Floasys meets the extensibility and modularity Non-Functional requirements since it can be tailored to customer needs, accommodate future needs and used in many departments. Although this work concerns an automotive use case, issues that we are facing within this sector seems to be very common issues also in other

[*]R&D - Aerothermal CFD, FIAT Chrysler Automobiles, Italy, claudio.gargiulo@fiat.com
[†]ISISLab, Dipartimento di Informatica, Università di Salerno, Italy, delmal@dia.unisa.it
[‡]ISISLab, Dipartimento di Informatica, Università di Salerno, Italy, dpirozzi@unisa.it
[§]ISISLab, Dipartimento di Informatica, Università di Salerno, Italy, vitsca@dia.unisa.it

sectors as highlighted in [2, 4], especially for the list of gathered requirements. Therefore, we believe that many of our considerations and design decisions could be adopted also for other type of simulations and in other contexts (i.e., aeronautic, rail and naval sectors).

The paper is organized as follows. Section 2 briefly introduces the use case study and outlines who are the stakeholders. The aim is to provide an overview about the context and its internal organization. Then, it analyses the collaborative Functional and Non-Functional requirements. Section 3 introduces the Floasys prototype with its functionalities. Section 4 discusses the Floasys architecture design decisions to meet stakeholders' requirements. Through the paper, we track and map the collaborative requirements with the solution ideas and the specific implementation technologies (i.e., libraries) used to develop the Floasys architecture. Section 6 concludes the paper and discusses possible future works.

2. Collaborative Requirements. This section analyses the key collaborative Functional and Non-Functional requirements to design a platform to foster the collaboration among industrial simulation practitioners and promote the sharing of models, results, and know-how. These requirements come from a relevant literature study and an extensive requirements elicitation activity performed through observations, stakeholders interviews and a user survey (Appendix A).

Fiat Chrysler Automobiles (FCA), as many other large industries, is organised in multiple geographically distributed teams that collaborate together. Through our survey analysis, we get that all analysts collaborate at least with another engineer in the same office and more than half analysts collaborate with at least one engineer who works in another location. They collaborate together sharing file geometries (CAD files), simulations and documents (e.g., slides, spreadsheets).

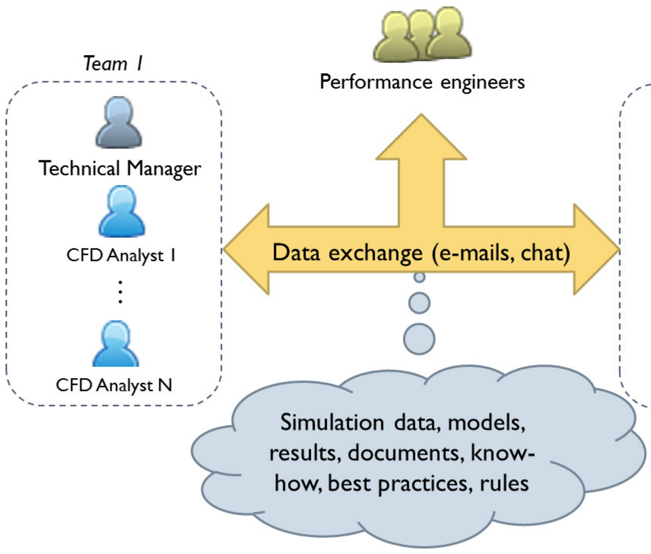


Fig. 2.1. *Geographically distributed teams that collaborate together in asynchronous way.*

Large industries have multiple locations around the world and are internally organized in multiple 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). Specifically, engineering functional areas perform tasks to design products and constantly invest in Research and Development (R&D) to improve their know-how and to be ready to provide innovative design solutions. The Computational Fluid Dynamics (CFD) unit is the engineering functional area with highly skilled engineers, called CFD analysts, who perform numerical computer simulations to analyse problems that involve fluid flow and other related physical phenomena, such as aerodynamic, aerothermal and aeroacoustic automotive product behaviour. CFD is widely adopted in many industrial sectors, such as automotive, aerospace, high-tech and chemical sectors. CFD analysts perform simulations following the CFD Workflow [5] that is iterative and consists of three phases: (1) pre-processing to prepare simulation, (2) solving and (3) post-processing to analyse results. The CFD unit and the CFD Workflow are our use cases. In each CFD unit, there are analysts and a technical manager who is responsible for the internal team organisation, resources monitoring and their allocations.

In a large industry, many CFD units collaborate together (Fig. 2.1). The collaboration is among geographically distributed CFD units and, among CFD units and other industrial teams, such as the product style designers and the performance engineers. In order to design an automotive product many engineers collaborate together. Especially, to perform aerodynamics/aerothermal analyses, CFD analysts, automotive designers, and performance engineers collaborate together.

The prerequisite to enable the collaboration among analysts is the ***simulation data centralisation***. Industries perform many simulations per year, therefore, in order to foster the ***model reuse*** and promote the ***data sharing***, it is fundamental how easy it is to retrieve the needed data stored in multiple repositories with different formats (often in closed file format). In order to improve data retrieval, users aim to annotate simulation files with additional ***metadata over data***, such as free tags or structured data, and to have a ***search tool*** able to get desired data. Search tools should support at least the search through files' names, annotated metadata and simulations' contents. ***Simulation data version control*** is another desired feature. The aim is to have a history of modifications made to simulations. It is a desired feature because the same simulation is often performed changing only some parameters (e.g., inlet velocity).

Table 2.1
Stakeholders' Collaborative Requirements.

	Requirement	**Notes**
Req. 1	*Simulation Data centralisation*	
Req. 2	*Metadata over simulation data*	Link metadata to simulations (e.g., free tags).
Req. 3	*Search facility*	Search based on file names, file content and tags.
Req. 4	*Version control over data*	
Req. 5	*Data sharing*	
Req. 6	*Integrate multiple simulators*	Avoid Vendor Lock-In
Req. 7	*Extensibility and modularity*	
Req. 8	*Do not change how engineers work*	

In order to gather the collaborative requirements (Table 2.1), we worked closely with a team of professionals in Pomigliano D'Arco (Italy) who extensively use CFD simulations to design automotive products. We observed their daily work annotating, collecting and analysing their tasks and workflows. We constantly discussed with analysts and technical managers trying to get a deep understanding of their work and answer to our questions. Requirements are refined through continuous iterations. FCA has multiple geographically distributed teams, therefore in order to get the collaborative requirements directly from stakeholders, we issued an electronic survey (shown in Appendix A) created with Google Forms[1]. The survey questions were divided in the following main sections: *participants' experience, collaboration among engineers and data sharing, data centralisation and data search*, and *simulation data versioning*. The survey responders are seventeen FCA professionals half from Pomigliano D'Arco (Naples, Italy) and half from Orbassano (Turin, Italy). Both groups design products using Computational Fluid Dynamics simulations. Through the paper we sometimes differentiate the technical managers and the analysts because they have different roles and requirements. Technical managers usually ask management features, such as the opportunity to monitor resources, projects timeline and performance goals. On the other hand, CFD analysts, who perform simulations, require engineering features (e.g., simulation monitoring, automatic document generation). Of course, both roles aim to collaborate over centralised data at different granularity. Floasys has been designed to also support engineering tasks, such as the simulation convergence monitoring, engineering wizards to automate repetitive tasks, simulation templates and so on. In this paper we mainly focus on the collaborative aspects overlooking the engineering Floasys's features. An important consideration is the impossibility to change how the employers actually work. Any architectural software solution to meet the requirements shown in Table 2.1 must rely on existing internal procedures and must not change them. During the requirement elicitation activity we also tried to understand the ways on how a collaborative platform could be introduced and deployed over existing practices ***without hardly change***

[1]http://www.google.com/google-d-s/createforms.html

how the engineers work but at same time improving their work. The following section will analyse each requirement listed in Table 2.1.

2.1. Simulation Data Centralisation. In order to support collaboration among engineers (Fig. 2.1) they must access to centrally available simulation data (Req. 1, Table 2.1). The idea is to collect data from different sources over time (i.e., from different simulators) and store them in open format like XML. In this way, data and results can be aggregated in different ways and can be compared within the same project or among different projects. Performance engineers and technical managers need to work on aggregate data (e.g., statistical data, trends about performances) whereas CFD analysts access to fine grain simulation data (e.g., model, simulation case) and their results to perform comparison. Obviously, data aggregation is not feasible with classic shared network folders that store data in a closed file format. Actually it is manually performed with continuous copy-and-paste operations among simulators and documents. In according to Aberdeen Group's whitepaper "*Getting Product Right the First Time with CFD*" [2], in order to improve the company competitiveness, they should centralise simulations results. Our aim is to centralise simulations and all their related data, such as the 3D geometries, simulation setup parameters and documents supporting their retrieval. In order to centralise data and provide additional services over them, software designers should consider: file size, total number of performed simulations and closed file format. In our use case, both geometries and simulations are very large files. In the survey, we asked which are usually the geometries and the simulations file sizes (questions **Q5** and **Q6** of the Survey shown in Appendix A). Figure 2.2a shows that the CAD file size is about one gigabyte in the fifty percent of answers. The file geometry can also contain the surface mesh and/or the volume mesh, explaining the differences of file size answers depicted in the chart of Figure 2.2a. Instead, the simulation file size (Fig. 2.2b) is more than ten gigabyte in 80% of answers. Simulations are so large because they contain the entire detailed vehicle geometry, the surface and the volume mesh as well as the physical/mathematical data to describe the model.

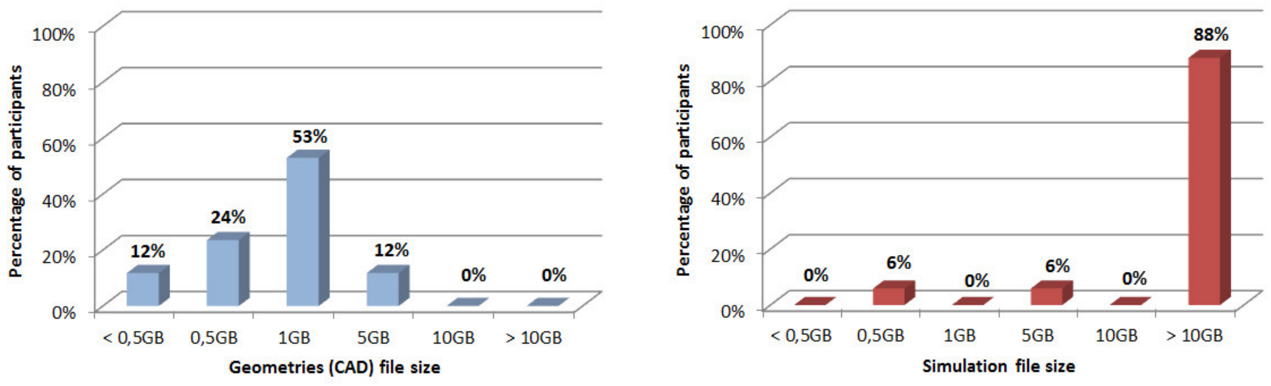


FIG. 2.2. *Geometries (question Q7) and simulations file size (question Q8).*

An alternative idea to provide services like data search or results aggregation, is to use a rational database to store simulation data, but considering file sizes and huge number of simulations we excluded it. In order to solve simulations the original files can not be moved and must be stored in their original format on file system. The use of a database leads to continuous transfers of data from the database to the file system and vice versa, compromising performance and response time.

2.2. Provide Search Facility. The aim is to provide a search tool able to find data using simulation file names, simulation content (e.g., its model, parameters, etc.) and metadata (e.g., tags). Simulators software often store simulation data as binary files in a closed file format. In addition, the used CFD simulator does not have an export functionality to an open format. Therefore, classical search tools are not useful to find simulation files based on their content (files are in binary format). For instance, the Windows OS search utility can not be used to search within the file content. To overcome this issue, users actually insert a lot of information in the simulation file name that will be useful to find data the next time. As shown in Figure 2.3, the main information inserted in the file name are (questions Q13-Q18): the project, the release, the revision number, the engine model and the vehicle trimming. Users decide to put the most important information, regarding their personal opinion, in the file name with the drawback to have very long file names. In addition, not all information can be stored in the file name so a lot of data remain within the simulation closed file and can not be used for next retrievals.

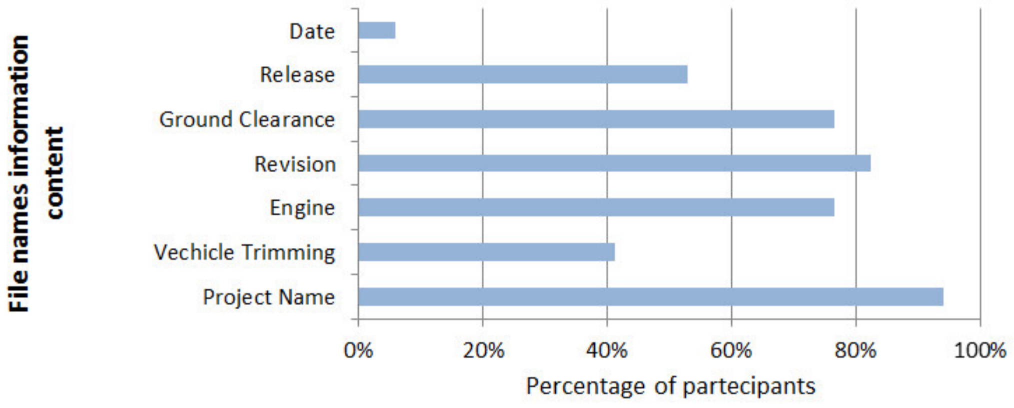


FIG. 2.3. *Information inserted in the simulation file names (multiple choices question Q12).*

More than half of analysts follow roughly some rules to store files in shared file system trying to follow them over time. Here, the term "rules" mainly means how engineers give a name to a file and how they decide the directories structures to improve the future simulation retrieval. Nevertheless these rules are mostly a personal choice (82%), engineers add essentially the same information to file names because the analysed engineering field is very specific. The limitation of this approach emerges when an engineer needs to search a simulation performed by other employees, mostly because he can not use existing search tools (e.g., the Windows Search tool) to search simulations based on the file content. An example of query is: "*search all simulations performed at inlet velocity X [km/h] that has the spoiler*". Unfortunately these data are not inserted in the file name and remain inside the closed files. This also limits the aggregation of data at different levels based on specific keywords and the relative results comparison of multiple different simulations to generate performance history charts.

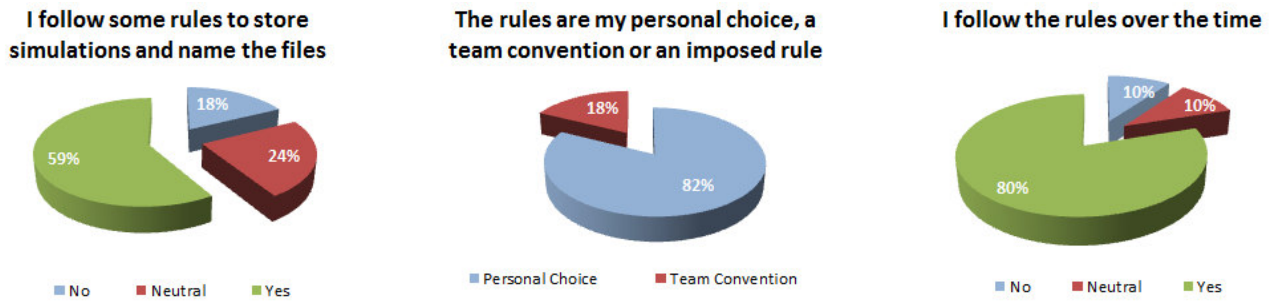
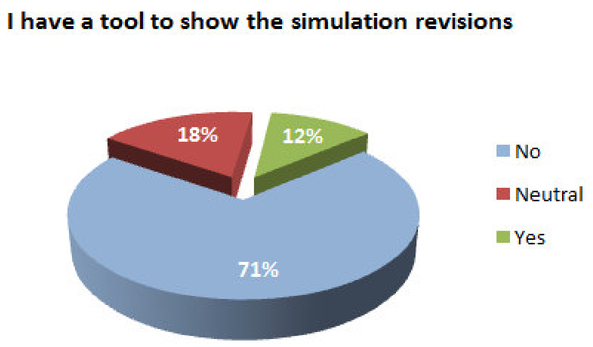


FIG. 2.4. *Rules to store the files on shared network folder (questions Q11, Q13, Q14).*

2.3. Provide Metadata over Simulation Data. Engineers use multiple simulators software, some of them store data in closed file format. As stated in the previous section, the file content can not be used to retrieve the files using the classical search tools such as using the Operating System find tool. Actually, to overcome this issue, engineers insert a lot of information in the simulation file name such as project name, revision and engine type (Fig. 2.3). Obviously, the file name can not host too many data, so other useful data are not annotated with simulations (e.g., free comments, descriptions). To get this requirement through interviews and the survey we asked whether the engineers desire to link other data to files (question Q15). All analysts (100%) desire a system to link other information to the files, such as the file tagging.

2.4. Simulation Data Versioning. As reported in the Aberdeen Group market research [2], an action to improve the company competitiveness is to provide version control over data. Our survey aims to further investigate this need especially to understand its value for stakeholders. Version control means that users can track modifications made to a simulation over time. It is interesting because engineers usually do not start simulations from scratch but they copy an existing file changing some parameters. In addition, starting from the same simulation file many other simulations can be performed just changing few parameters (e.g., the inlet velocity). In according to our survey, more than 60% of participants declared that they do not have a tool to track the simulations modifications. In addition more than 80% of participants said that the feature could be useful.



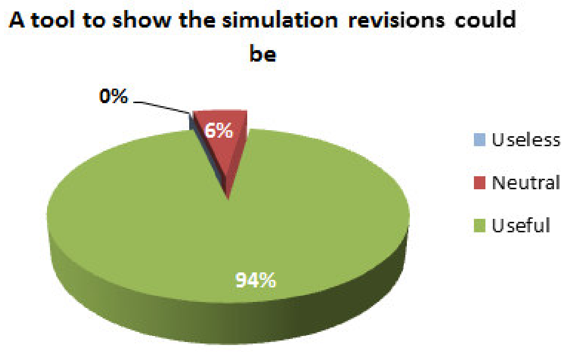


FIG. 2.5. *Version control (question Q20).*

2.5. Support Data Sharing. CFD analysts need a mechanism to exchange references about data. On Internet a common way to share resources is exchanging URLs. Hence, our idea is to univocally identify simulation data with URLs 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. 2.1), so an access control is important to control the sharing of confidential data.

2.6. Simulator Independence and Integration of Multiple CFD Simulators. The previous requirements must work independently by specific used simulators to generate data. For instance, tagging and search functions must work on a repository of heterogeneous simulations coming from multiple simulators. This requirement is very important because in the analysed context, analysts use multiple CFD software and actually one single software can not be used to perform all simulation types. In our use case and large industries, there are different teams that use different software to perform tasks. For instance, a team is responsible for the CAD design whereas another team simulates models using other software. Obviously, in other contexts both design and simulations can be done by the same team with an all-in-one CAD/CAE software. Through the survey, we asked (multiple choices question Q21) to indicate which simulator software the analysts use, to give an idea about their multiplicity. All analysts use Star-CCM+ and more than half of them use OpenFOAM. Other used software are: CFD++ (35%) and PowerFlow (18%). Analysts have used software over the years and they are confident with them. Moreover, industries are unwilling to invest in training engineers on other software products. Therefore, in order to meet the requirements is fundamental to support and collect data from multiple daily used CFD simulators. This is a key difference with other platforms (i.e., e-Science) that often integrate simplified or in-house developed solvers [6].

It is evident that any platform must consider the integration of multiple simulators. The integration of multiple simulators (Req. 6 in Table 2.1) has some difficulties especially because CFD analysts use often proprietary software and actually a lack of simulator standardisation exists so that many software do not have function to export data in open format. The import/export in open format are important functions to evaluate during the choose of a CAD/CAE software [7] otherwise simulation data are locked in the vendor software. Vendor Lock-In is a well-known Anti-Pattern [8] [9] [10]: the phenomenon that causes customer dependency on given vendor about a specific good or service [11] with high switching costs [12]. Vendor Lock-In occurs both in terms of services and data. Vendor Lock-In Anti-Pattern in terms of services occurs when the architecture heavily relies on a closed vendor software and strictly depends on vendor choices, so the architecture is product-dependent [13]. Data Lock-In occurs when the only way to access to data is using the Vendor Software because data are stored in a proprietary file format or on a vendor server that does not provide an export functionality to open format or a public customer API. The exporting and importing of geometric data are well-established functionalities for CFD software, simply because they must commercially support the interaction with other CAD software. Conversely, it is not the same for simulation data such as case setup, simulation results and so on. Data Lock-In is very common in Cloud Environments [14] and is an obstacle to cloud computing [15]. Vendors lock users in to make difficult to change product because they cannot get their data; despite, as reported in literature, giving the opportunity for customers to get their data increases their trust in the product [16]. A design solution useful to mitigate the Vendor Lock-In is to design the system with an additional layer called isolation layer [8].

2.7. Extensibility and Modularity. The combination of modularity and extensibility [17, 18] system qualities advantages are: the opportunity to compose a system with the only needed modules, the introduction of new functionalities tailored to customers' needs, and the creation of customers own modules to automatise specific tasks keeping them private to protect the know-how. Extensibility is the ability of a software system to allow and accept significant extension of its capabilities without major rewriting of code [17] [18]. Extensibility is a quality architecture attribute useful during the development and especially in future when more and more simulators' features will be integrated in the architecture [19]. Industries want to deploy the same system with different features. Modularity "*is the degree to which a system or computer program is composed by discrete components such that a change to one component has minimal impact on other components*" [17]. The architecture must be modular to support both the adding of new simulators and the removing of existent simulators. The modularity requirement has an interesting advantage for the architecture design: the engineering tools and simulators are loosely coupled. An important consideration concerns the software license. Two opposite needs must be taken into account: on one hand, industries want to protect their know-how, on the other hand, the architecture must be also adopted in other contexts. Based on our use case, modularity, extensibility and EPL license [20] are the right mix. The architecture, the framework and some other modules are open source. At same time industries can protect their know-how developing their own private and closed modules.

3. Floasys Functionalities. This section describes the Floasys functionalities and shows its graphical user interface (GUI). Floasys provides a *simulator independent repository tool* to navigate open format simulation data repositories and annotate selected files through free and structured tags (Req. 2). Floasys has a structured and assisted *Search tool* to get simulations performed by different engineers (Req. 3) and *share* them (Req. 5). Floasys's screenshots contain CFD related data but its GUI and its ideas are general to be reused in other engineering areas (e.g., ergonomics).

Floasys is a Web-based platform to support both engineering tasks (e.g., run simulation, monitor simulations, generate documentation automatically etc.) and data sharing among dispersed engineers. Floasys centralises simulation data in open format and provides a search tool able to browse and query the simulation database using tags identifying versions, interesting features and open comments. The Figure 3.1 depicts a real-world Floasys workflow that is difficult or time-consuming without our platform. It is composed by six tasks executed in sequence. In Task 1, user finds a simulation using keywords like project name, revision, velocity and so on. The velocity is an internal simulation parameter. It is embedded in the closed file format, so the task to search by velocity can not be accomplished without Floasys or at least, as come to light in Section 2, the user can remember where he stored the simulation file and open it to check the velocity value. In addition, Operating System find tool can not be used to get the simulation because velocity is not included in the simulation file name (Fig. 3.2). With Tasks 2 and 3, the user selects a simulation from the list of results to get the original simulation file and open it with the proprietary software. Unfortunately, the original simulation file is not in the repository. Using Floasys, nevertheless the original file was deleted, the user can get the simulation data, setup

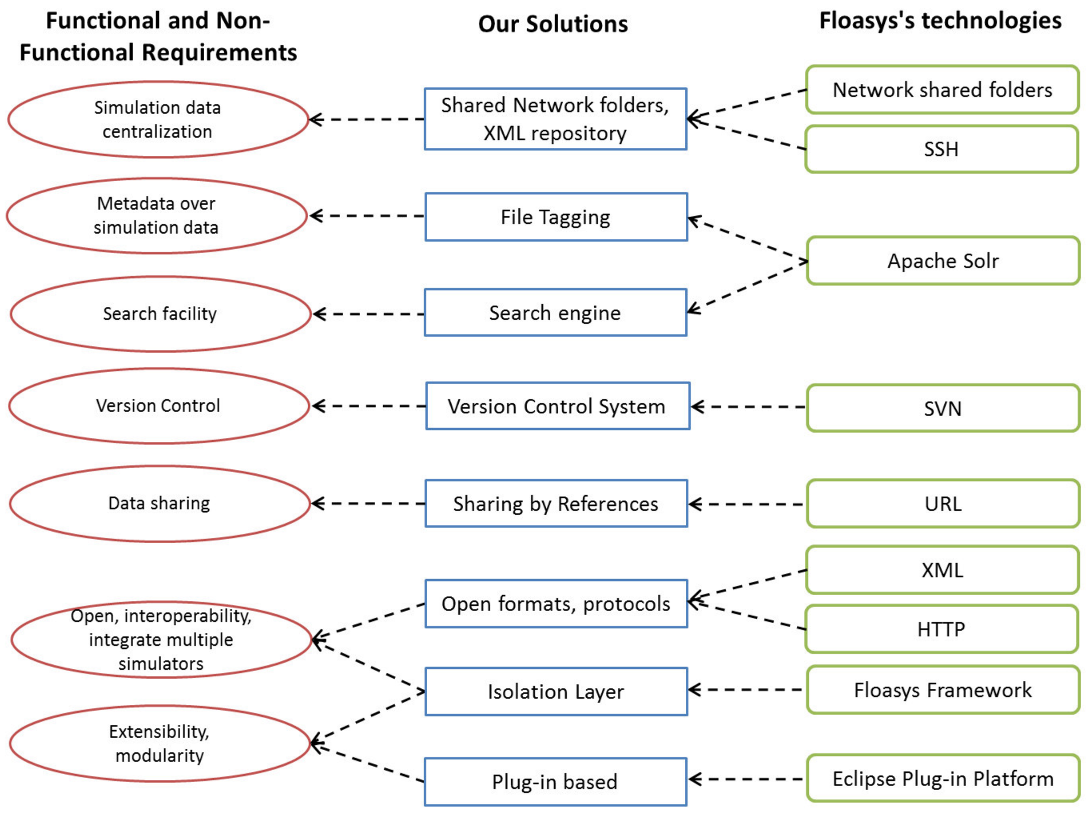


FIG. 2.6. *Mapping among Stakeholders' requirements, solutions and prototype technologies.*

and results. Of course, these data can not be used directly to simulate it again. Anyway, an expert engineer can recreate the simulation starting from the provided surface mesh and simulation setup (boundary conditions, physical model, used parameters, previous reports and so on). The Task 6 concerns the sharing of a simulation URL to another user via a preferred medium (e.g., e-mail, chat). Of course, the shared URL is available only within the industry's Intranet.

Floasys provides a re-configurable GUI based on Perspectives and Views concepts provided by Eclipse Remote Application Platform (RAP) [21]. The idea is that the virtual workbench changes according to the engineering tasks. In this way, the system is able to show only relevant functionalities to perform the actual task. A *perspective* is a specific configuration of the workbench and contains many views to show information. A perspective provides well-organized software functionalities access because it divides them in semantically homogeneous sections.

3.1. System Independent Repository Tool and Simulations Tagging. The Repository tool supports the navigation of central simulation repositories. Floasys integrates multiple simulators, so data heterogeneity is one of the issues to face. For instance, OpenFOAM stores data in a well-defined directories structure of 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 using the vendor software. The Repository tool, relaying on Floasys framework services, is simulator independent and is able to manage data from different simulators. The Repository tool inherits the user file system access permissions, so logged user can access only to files he/she has authorised. Floasys can access to network folder through a server using a SSH connection with logged user credentials.

The Repository tool provides file annotation and tagging features. The idea is to enrich simulations files

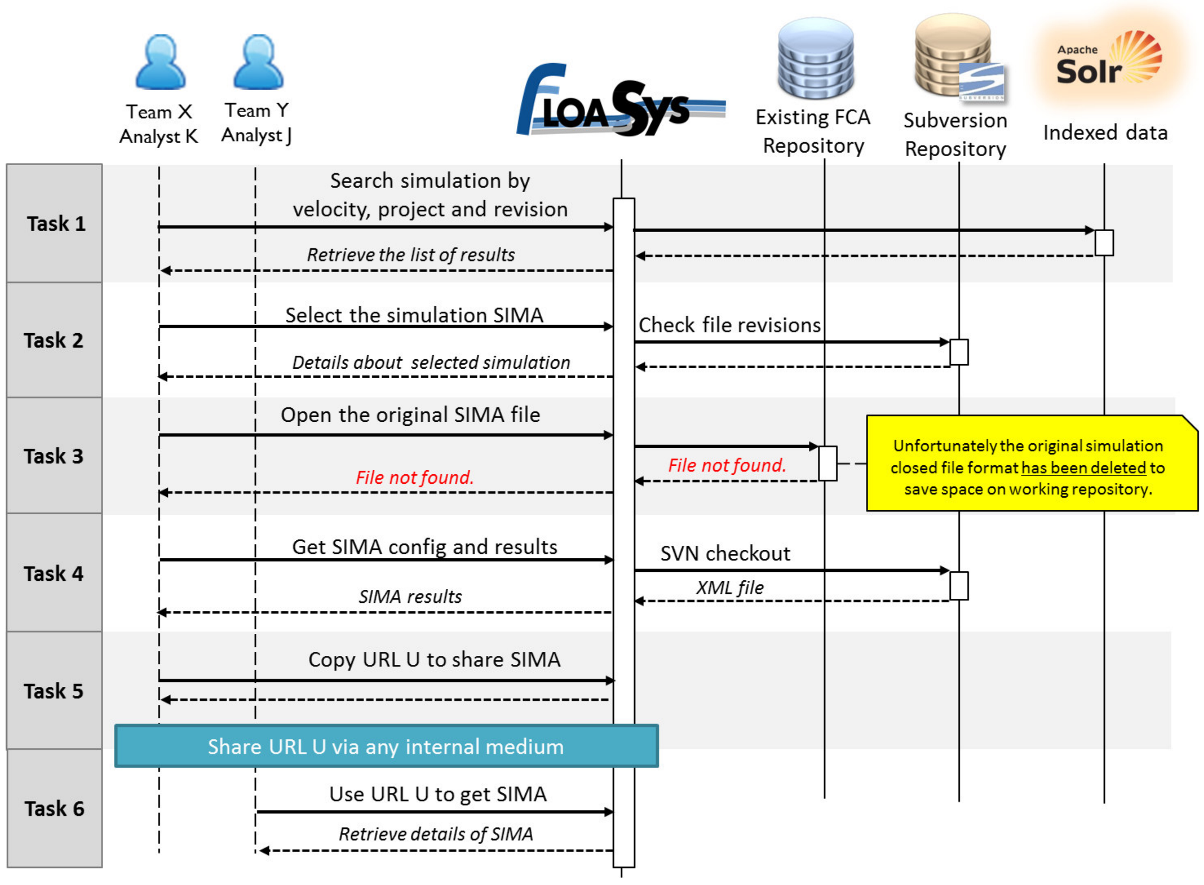
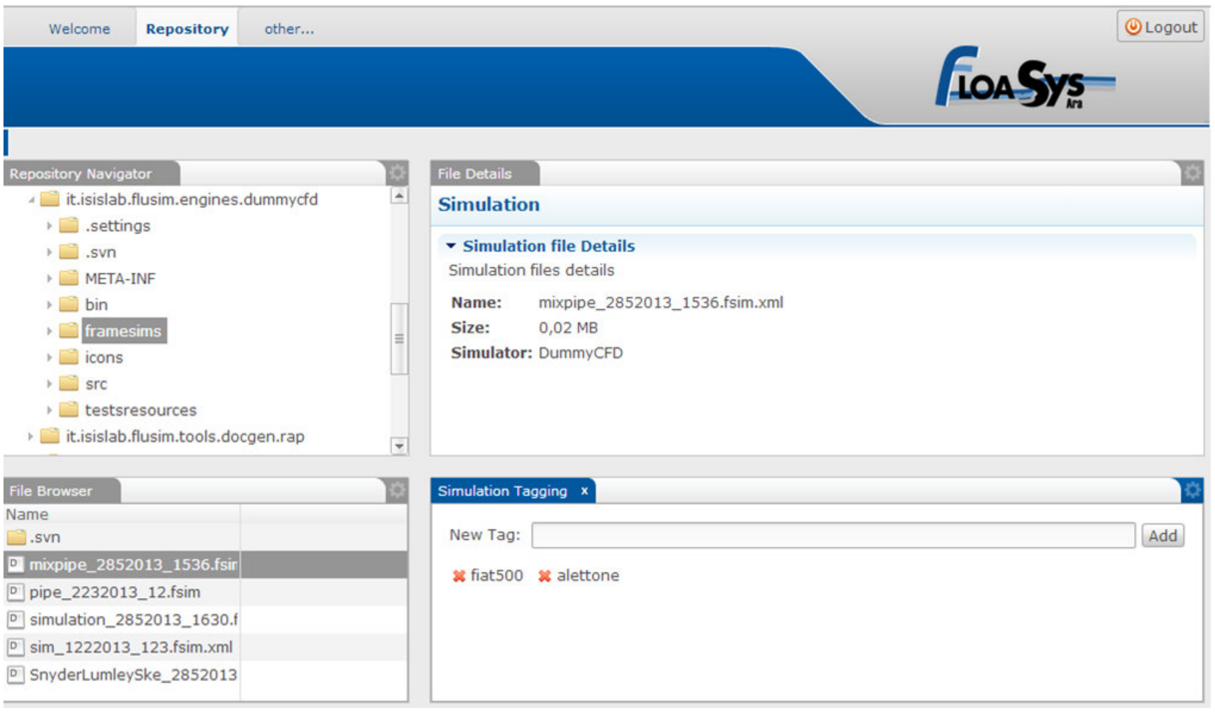

Team X Analyst K
Team Y Analyst J
FLOASys
Existing FCA Repository
Subversion Repository
Apache Solr
Indexed data
Task 1
Search simulation by velocity, project and revision
Retrieve the list of results
Task 2
Select the simulation SIMA
Details about selected simulation
Check file revisions
Task 3
Open the original SIMA file
File not found.
File not found.
Unfortunately the original simulation closed file format has been deleted to save space on working repository.
Task 4
Get SIMA config and results
SIMA results
SVN checkout
XML file
Task 5
Copy URL U to share SIMA
Share URL U via any internal medium
Task 6
Use URL U to get SIMA
Retrieve details of SIMA

FIG. 3.1. *Example of a typical workflow supported by Floasys.*

with metadata: a user can annotate a simulation file and provide additional information useful to retrieve and share it in future. Examples of free tag categories are: brand, project name, revision and engine type; all information that can not be stored directly within simulation files, whereas Floasys allows it. Analysts are free to add any tag to files. In order to uniform the provided tags, during typing, Floasys suggests the tags to use (Fig. 3.2). Tags are both unstructured with free tags and structured inserted filling out standard forms like in Figure 3.3.

3.2. Search Tool and Data sharing. The Search tool (Fig. 3.4) is a Floasys perspective developed to provide the search of simulation data stored in central repositories. The tool supports the search by file name, simulation content, free tags and structured data (Req. 3 in Table 2.1). When a user types the search keywords, Floasys recommends further keywords to refine the search (Fig. 3.4). In this way, the tool supports the search activity suggesting further search keys to reduce the total number of potential results. The system performs search using only indexed data without accessing (e.g., open) to original closed format files. The results are displayed in a list. In order to display the revisions history, the user can select a simulation from the list of results.

Each simulation file has a unique ID within Floasys and all relevant data (e.g., documents, simplified 3D geometry, surface mesh and so on) are linked to this ID. Both repository and search tools provide a unique URL for each selected simulation. Our idea is to share data by simply exchanging unique reference to the specific simulation data. URLs identify simulation data and inherit file system permissions. The URL is private and is accessible only within the industry boundaries. Considering the Computer Supported Cooperative Work



it.isislab.flusim.engines.dummycfd
.settings
.svn
META-INF
bin
framesims
icons
src
testsresources
it.isislab.flusim.tools.docgen.rap

Simulation

Simulation file Details

Simulation files details

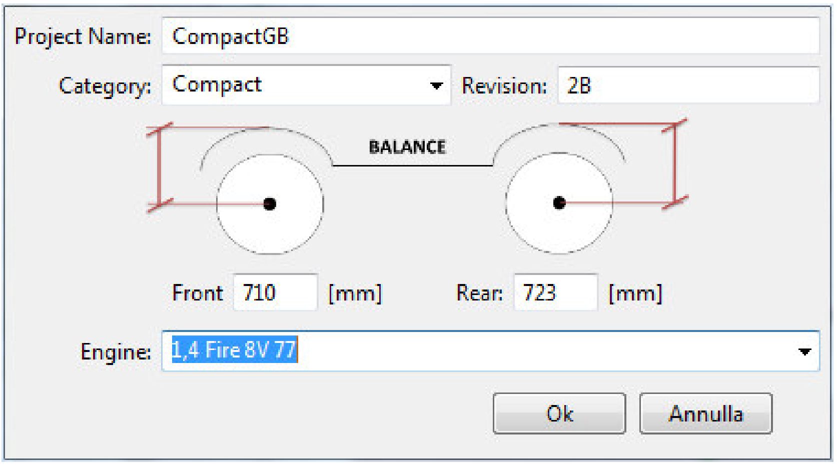
Name: mixpipe_2852013_1536.fsim.xml
Size: 0,02 MB
Simulator: DummyCFD

Name
.svn
mixpipe_2852013_1536.fsir
pipe_2232013_12.fsim
simulation_2852013_1630.f
sim_1222013_123.fsim.xml
SnyderLumleySke_2852013

New Tag:

fiat500 alettone

FIG. 3.2. *Repository tool to navigate a simulation repository and tag the resources (e.g., files).*

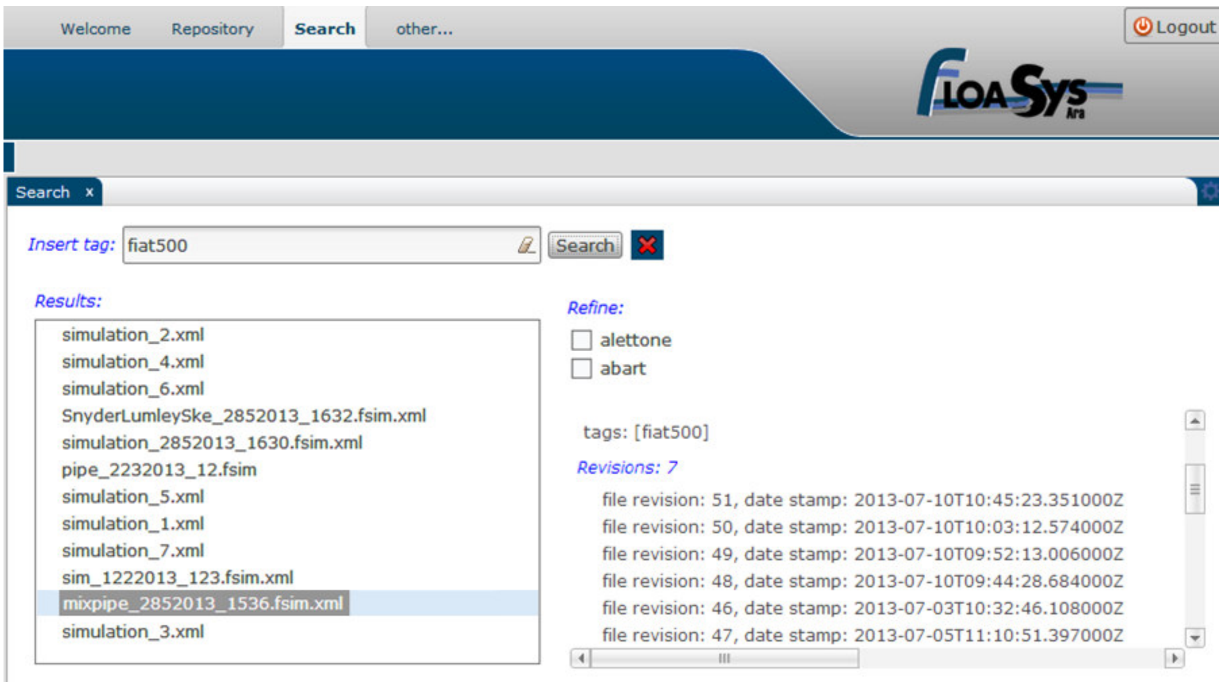


Project Name: CompactGB
Category: Compact Revision: 2B
Front 710 [mm] Rear: 723 [mm]
Engine: 1,4 Fire 8V 77

FIG. 3.3. *An example of structured data to store.*

(CSCW) space-time quadrants [22], Floasys supports the *asynchronous* data sharing for both co-located and distributed teams.

3.3. Web-based 3D Model Visualisation. Floasys shows a reduced 3D geometry of the simulated vehicle. Through this tool, engineers can quicly discover which components have been used to simulate the product without opening the CAD software. The tool shows a list of components with their Property IDs (PID) on the left (Fig. 3.5). The user can activate or deactivate some parts and can perform the basic zoom and pan operations. Figure 3.5 shows the simplified 3D surface geometry of a FCA production vehicle. The 3D vehicle geometries usually are very complex. To give an idea, each geometric model takes up ten gigabytes and engineers



Insert tag: fiat500

Results:

simulation_2.xml
simulation_4.xml
simulation_6.xml
SnyderLumleySke_2852013_1632.fsim.xml
simulation_2852013_1630.fsim.xml
pipe_2232013_12.fsim
simulation_5.xml
simulation_1.xml
simulation_7.xml
sim_1222013_123.fsim.xml
mixpipe_2852013_1536.fsim.xml
simulation_3.xml

Refine:

☐ alettone
☐ abart

tags: [fiat500]

Revisions: 7

```
file revision: 51, date stamp: 2013-07-10T10:45:23.351000Z
file revision: 50, date stamp: 2013-07-10T10:03:12.574000Z
file revision: 49, date stamp: 2013-07-10T09:52:13.006000Z
file revision: 48, date stamp: 2013-07-10T09:44:28.684000Z
file revision: 46, date stamp: 2013-07-03T10:32:46.108000Z
file revision: 47, date stamp: 2013-07-05T11:10:51.397000Z
```

FIG. 3.4. *Search Tool*

use very performing hardware to open and manipulate them. An important requirement for any engineering platform is the visualisation of 3D geometric data. As many other platforms, Floasys is a Web-based platform. 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 the server to the clients 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 be rendered 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 visualisation allowing engineers to quickly check which are the vehicle components at a glance. For instance, an engineer can visually check if the vehicle is simulated with the spoiler.

4. Floasys Architecture. This section introduces the Floasys architectural solution to centralise, annotate, tag, search and share simulation data. In order to meet the stakeholders' requirements, our solution collects simulation data from already existing simulation repositories (e.g., network shared folders), transforms, indexes (to provide high data retrieval performance) and store them in open format (e.g., XML).

4.1. Architecture Overview. Floasys is based on a Client/Server architecture (Fig. 4.1) developed using Eclipse Remote Application Platform (RAP) [21]. Clients are Web-based components. Therefore, Floasys is accessible through any browser installed on the company workstations. The Web-Based RAP clients communicate with the server exchanging commands and messages in JSON text format [23] over the HTTP protocol. Servers tend to interact with user browsers using the JSON exchange format [24] because it is easily parsed in client-side JavaScript language [23]. The Floasys's server can access to a set of already existing repositories (mainly shared network folders) that store the simulation files in their original format. In according to the internal policies, Floasys accesses to these existing FCA repositories in a read-only mode through the SSH protocol with the logged user credentials. Therefore, the architecture needs an additional repository to store simulations in open format (e.g., XML) with annotations, tags and additional metadata (Req. 2). Floasys supports two types of repository: an internal Subversion server or a shared network folder (without the version control support). In order to improve retrieval performances, Floasys indexes open format XML documents

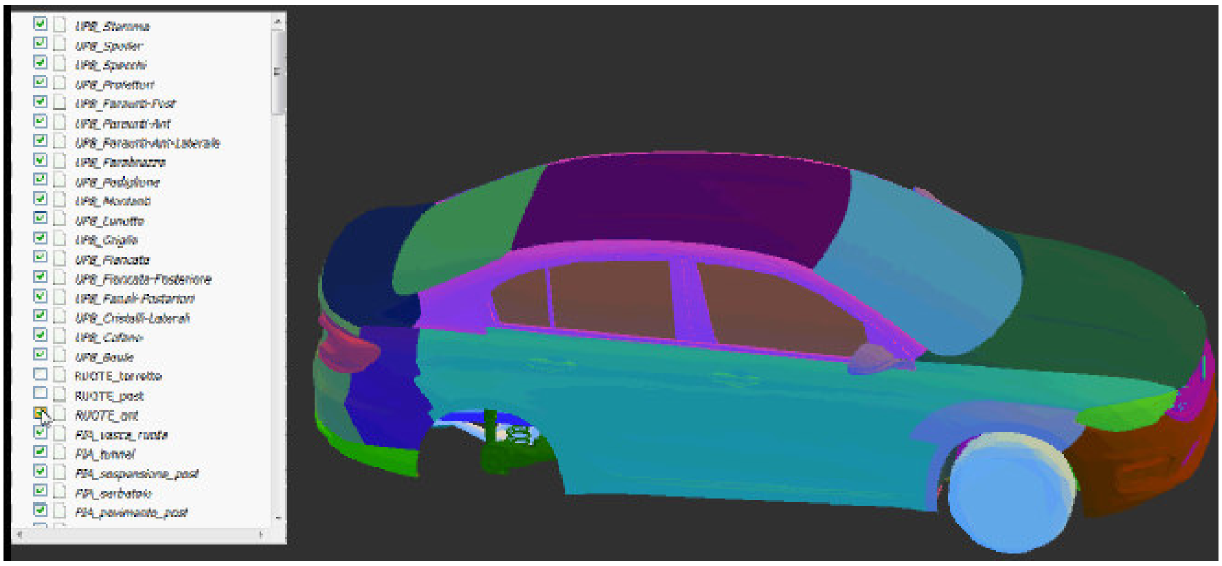


FIG. 3.5. *Floasys 3D model visualisation.*

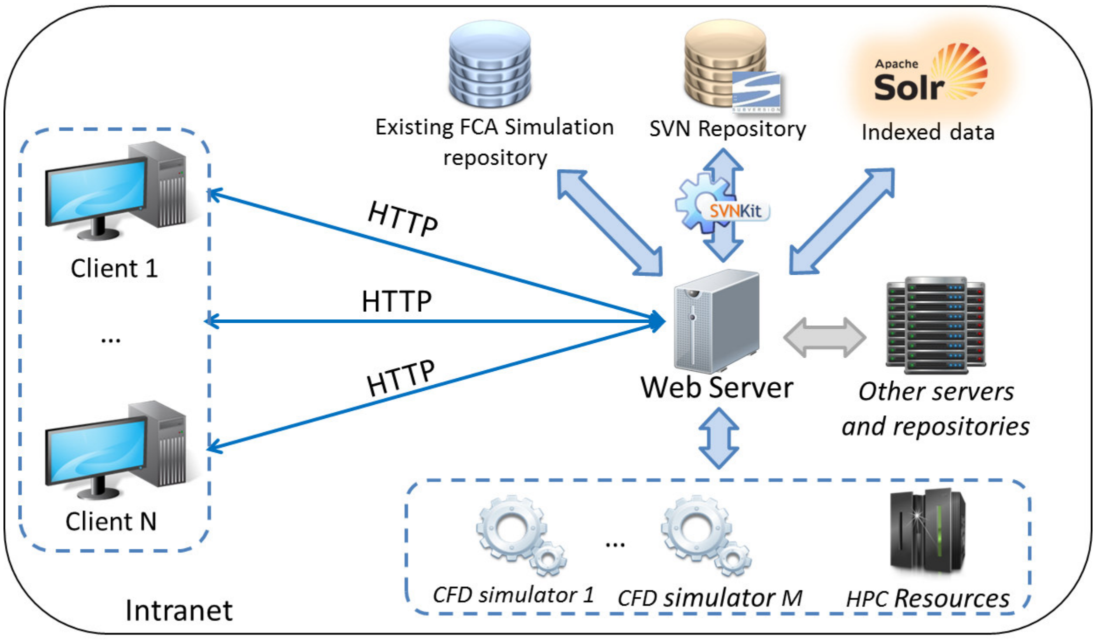


FIG. 4.1. *Floasys Client/Server Architecture.*

relaying on a well-established search engine technology like Apache Solr [25, 26, 27]. The server can access also to simulator software and High Performance Computing (HPC) resources as well as other internal services like the authentication service. Floasys is Intranet-based for security reasons. In addition, any kind of control access to data must be compliant with the industries internal policies and can not be override. To provide authentication and to manage both users and groups, Floasys can rely on existing industrial internal Lightweight Directory Access Protocol (LDAP) servers [28, 29] or use existing Secure SHell (SSH) accounts comply with existing file and directories access permissions. Floasys could be exposed also on Internet, but limitations exist such as the huge amount of simulation data (gigabytes) to transfer. Trusting and security issues must be taken into account (e.g., to avoid espionage activities). Floasys is designed, developed and tested following an Agile methodology based on short iterations of two weeks each in average, delivering small functionalities every time. During the development, especially for server-side features, we wrote black box unit tests using JUnit [30]. From functionalities testing point of view, for each planned release we had a test plan with the test cases to execute and check on a controlled environment software installation. In addition, during the Floasys development, we worked closely to analysts in Fiat Chrysler Automobiles to get the user feedback as soon as possible that were recorded in an issue tracking system (e.g., Edgewall Software Trac[2]) and scheduled for the next plans in according to the issue/enhancement priority.

4.2. Server-side Software Architecture. The Floasys server-side component interacts with the simulator software to collect closed format data and transform them in open format. The architecture is a three layers approach (Fig. 4.2). It integrates multiple simulators in the bottom layer wrapping the vendor software. The top layer is the front-end that contains the Web-based GUI tools (or applications). The middle layer (1) provides a common APIs to the front-end tools, (2) provides a common unified data representation called Simulation Model for data coming from different vendor systems, (3) it is an isolation layer [8] to decouple the front-end from vendor-specific simulator wrappers and, (4) it allows the vendor-product switching at run-time to choose which ones are able to provide the needed services and data.

The middle isolation layer contains the common APIs exposed to the upper applications layer. In order to keep its use easy, it mainly contains interfaces (or abstract classes) which are implemented by vendor-specific wrappers. The architecture is able to provide the middle layer services also with other technologies such as Restful and Web Services to support the interaction and data exchange among other devices (i.e., mobile devices) and/or industrial systems. In this way, another third application (i.e., mobile application) can access to the central simulation repositories and provide other service over open format data. Actually Floasys Meeting Mobile is under development to provide statistical information about projects during the meetings.

An alternative solution to our architecture could be the introduction of a separate isolation layer for each vendor software. The *support of multiple replaceable* vendor products and the *simulators selection process* requirements impose the introduction of a common isolation layer. In fact, the alternative solution has the following drawbacks: (1) the selection process is performed in the application layer and (2) separate isolation layers means also different APIs, differences that must be handled in the application layer. However, the use of a common isolation layer does not exclude that each wrapper itself is designed with an isolation layer using a proxy pattern.

The extraction of data from closed file format generally is a tricky task and the solution depends on the specific proprietary software and it is strictly coupled with it. The reverse engineering of the binary file content is an extreme solution and we definitively tried to avoid it during Floasys development. Our idea is to interact with the simulator taking advantage of its specific features. Specifically, CFD simulators have an interesting built-in feature: the opportunity to write (or record) a macro to automate tasks within the software. In addition, CFD simulators run "*headless*" without the graphical user interface (GUI) and can execute macros from the command line. It is a built-in feature because every CFD simulation requires and runs on High Performance Computing (HPC) resources. For instance OpenFOAM, an open source CFD software package, is a set of command line tools without GUI so that the aim of many projects [31] both open and commercial is to design a GUI for OpenFOAM. Another CFD simulator is CD-Adapco Star-CCM+, it has a Java-based macro language to automate repetitive tasks. Therefore, Floasys takes advantage of this built-in CFD software feature. In order to extract the data from a closed file format, the specific Floasys Wrapper runs the original simulator and execute a macro within the simulator. The macro reads the simulation content and stores everything in a plain intermediate file that after it is managed by Floasys platform. Floasys reads this plain intermediate file, transforms it to a common open format creating a XML document stored in the central open repository.

In order to meet extensibility and modularity requirements (Req. 7), the server is based on a pure plug-in architecture [32]. A plug-in can provide well-defined hook points called *extension points* to define and describe the way to extend its functionality. Other plug-ins (or modules) can add new functionalities implementing

[2]Edgewall Software Trac official web site: `http://trac.edgewall.org/`

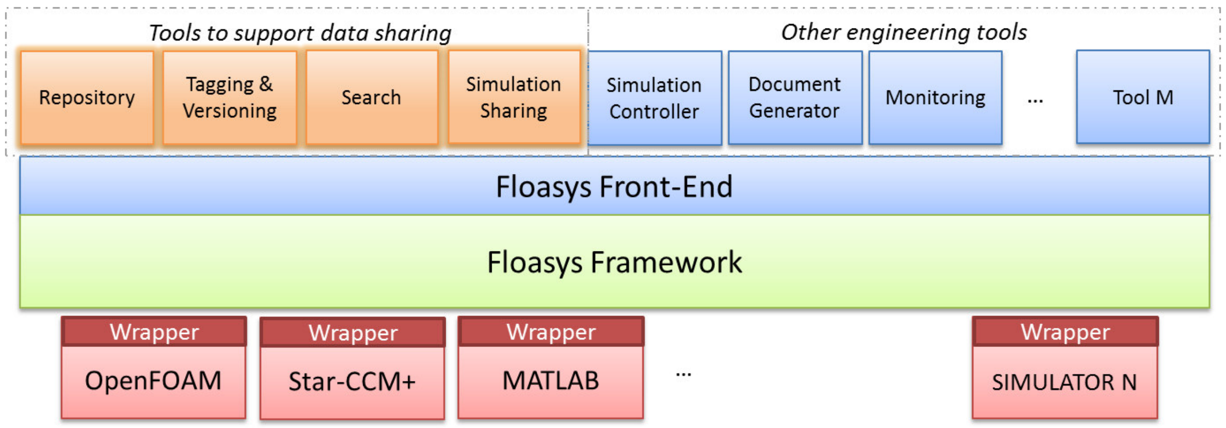


FIG. 4.2. *Floasys Server-side architecture.*

an extension point. In addition, a module can be replaced with another equivalent implementation also at run-time. The Floasys core provides two extensions points to extend its functionalities: (1) one hook point to introduce new tools in the upper layer and (2) another hook point for new wrappers. In this way, the following opportunities exist for the final customers: 1) multiple Floasys instances can be deployed choosing which modules will compose the overall architecture in according to the industrial needs; 2) the industry can identify exactly which modules contain their specific know-how; 3) each company can decide to invest money for the development of its own internal modules to customise Floasys and meet specific internal requirements; 4) in according to Eclipse Public License [20] (EPL), each plug-in can be released open sources or with a closed license.

Floasys has two kind of modules: wrappers on bottom to collect data and tools on top to provide engineering features (Fig. 4.2). An interesting Floasys extension planned for the future is to develop a wrapper that collects experimental data (e.g., wind tunnel experimental data, engine test bed). This is a challenging goal but the advantage would be a central repository that contains both simulation and experimental in open format supporting the comparison among them. An important task is the validation of simulation results and the comparison among the computer results and experimental data is very important.

Floasys relies on mainstream technologies. The server-side components are Java servlet-based. Floasys is developed upon Eclipse Remote Application Platform (RAP) that "*uses standard servlet technology and runs on any JEE servlet container*" [21]. Therefore, the outcome of the deployment phase is a Web application ARchieve (WAR) file that is deployed on a JEE servlet container (e.g., JBoss or Tomcat). This software stack can be installed upon any operating system (e.g., Mac, Windows or Linux). Actually in according to the industrial internal policies, the server is a Red Hat Linux distribution with JBoss[3] but any other Linux distribution can be used.

4.3. Simulation Model: Managing Simulator Differences. Floasys aims to collect data from multiple different simulators that often use closed file formats. A lack of interoperability among CFD software exists (see Section 2) so Floasys must directly handle these heterogeneities. Heterogeneities among vendor products are both syntactic and semantic. The syntactic heterogeneity concerns the vendor product APIs differences or the way to interact with them trough command line. The architecture has an isolation layer (Floasys Framework in Fig. 4.2) to face these syntactic differences that remain within the simulator wrappers and one common API has provided to upper layers. Semantics and data heterogeneities deal with data differences: software are often similar but they use different concepts. This issue becomes evident when architectures try to "*support the concurrent use of multiple infrastructures, transparently*" [8]. Floasys introduces an intermediate common representation for simulation data called **Simulation Data-Model**. It is based on a tree-like data structure as CGNS [33] format. In order to be reusable, it consists mainly of interfaces and abstract classes. In

[3]Red Hat JBoss official web site: `http://www.jboss.org/`

addition, Floasys provides a basic implementation based on the composite design pattern [34]. Each wrapper (architecture bottom layer Fig. 4.2) knows how to interact with a specific simulator and can extract data from a closed file format. The same wrapper is responsible to create the Simulation Data-Model instancing the basic implementation and translating simulation content in nodes of Data-Model. The Simulation Data-Model is serialisable. Floasys serialises the Simulation Data-Models in XML documents that are indexed using Solr and stored in a Subversion repository. Floasys uses Java XStream [35] Library to serialize Simulation Data-Model in XML. This Data-Model is very powerful because Floasys can enrich the original data adding meta-data as a new node of the tree structure. Both Floasys Framework and wrappers can add metadata over data inserting additional nodes in the tree (i.e., documents, automatic extracted information) during extraction phase. Also users can enrich the Data-Model providing tags and comments through repository tool that become nodes in Data-Model. All the information stored in Simulation Data-Model can be used during within the Search Tool to find simulations.

The advantages of our intermediate Simulation Data-Model representation are: (1) metadata over data adding custom nodes, (2) serialisation in open format such as XML, (3) decoupling of wrappers from tools so it is possible to replace a wrapper limiting changes to upper layers and (4) opportunity to compare results that came from simulators with the results that came from the experiments with real prototypes in future. Finally, we experienced a great advantage of using a Data-Model during Floasys development and for the stakeholders after. Using the Data-Model has the advantage to use the Floasys front-end without simulators. The idea is to have a dummy simulator that reads data from the XML file and provides them through the described architecture as a real simulator. This is a cost-saving in terms of HPC resources and available simulator licenses for closed software. Considering the 3D geometry complexity, to open a simulation file, engineers access to a computer cluster using a software license that are fixed by the project budget. Therefore, the requirement to avoid data lock-in leads to a cost-saving feature.

4.4. Simulation Data Centralisation, Version Control and Data Indexing. The architecture integrates multiple simulators, collects and centralises simulation data. Each simulation contains textual, numerical (e.g., results), images and geometrical data. Floasys extracts all simulation data embedded in closed file format and stores them in open format files. The textual and numerical data are stored in XML files in according to the Simulation Data-Model and are committed to the Subversion repository. These XML files are relatively small (MB) so they are easily managed by the Subversion repository. Obviously, most Subversion operations are recursive but Subversion 1.5 introduced the sparse directories [36] (or shallow checkout) to checkout a portion of the working directory with the freedom to get more files and directories later [36]. Therefore, Floasys relies on the shallow checkout to get a partial group of XML files. Floasys can use multiple Subversion servers to accommodate future needs. Version control granularity concerns the specific simulation file. In this way, simulation XML files can be distributed among multiple Subversion servers. Floasys architecture has designed to store the SVN URL within the Solr search engine during the indexing phase. Hence, when the user search a simulation and gets the search results, for each result there is the SVN URL to a specific Subversion repository. Hence, every time Floasys exactly knows the Subversion server used to store the open format XML document. In addition, in order to provide high search performance, the generated simulation XML files are indexed using Apache Solr [25]. Apache Solr provides extensions, configuration, infrastructure and programming languages bindings around Apache Lucene. In according to the official documentation [25], Apache Solr is is highly reliable, scalable and fault tolerant, providing distributed indexing, replication and load-balanced querying, automated failover and recovery, centralized configuration and more. In particular, Apache Solr can be run in a standalone configuration or it is possible to setup a cluster of Solr servers through SolrCloud to combine fault tolerance and high availability as well as scalability using replication and distributed indexing dividing the index into partitions called shards.

Floasys does not use the Subversion repository for the geometrical data because they are very huge (GB). A simulation contains mainly two meshes (geometrical data): (1) a *surface mesh* that is the vehicle shapes used to build the (2) *volume mesh* used at solving time to solve the simulation. Floasys extracts only the surface mesh and makes two outputs: a simplified geometry that serves just as overview of the vehicle product (it is fast to retrieve and render with WebGL, see Section 3.3) and a surface mesh file (e.g., STL file). Floasys does not store geometric volume mesh (the most heavy part of a simulation) reducing the overall required amount of physical space. In this way it saves space on repositories and it is always possible to build volume mesh from surface mesh.

In order to get the simplified 3D geometry version used only for the visualisation on web, Floasys in batch connects to the Matlab server and reduces the original STL surface mesh creating the lightweight version. This simplified version contains all vehicle parts separately. After many attempts the best trade-off between running time and the 3D geometry quality is to use the Matlab `reducepatch` command. The quality of the obtained mesh is assessed asking to CFD analysts. 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.

The proposed solution has an interesting advantage. XML files store the most important and useful simulation data including a simplified 3D geometry. Therefore, 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.

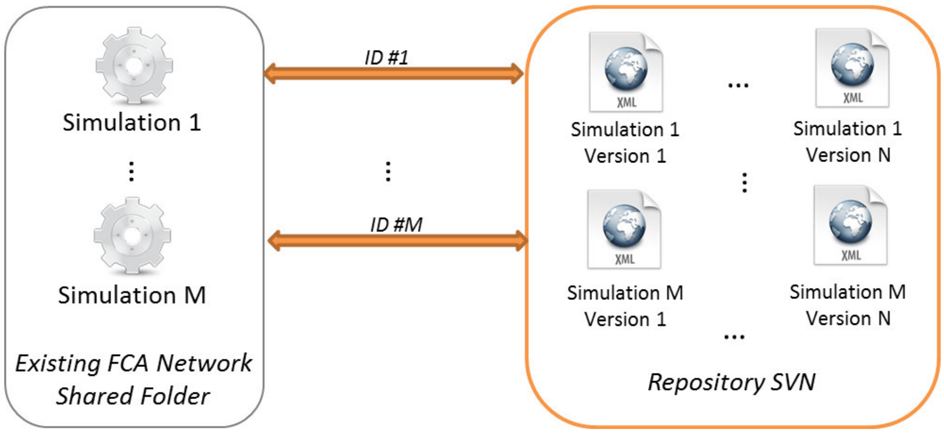


FIG. 4.3. *Simulation data versioning.*

For each simulation file (left-side of Fig. 4.3) stored in closed file format, an XML file exists in the SVN repository (right-side of Fig. 4.3) that contains extracted simulation data and metadata in open format. In addition, each XML file is indexed using Apache Solr [25].

Each XML file is always linked with its original simulation file using an unique ID. In this way, the users can always get the original simulation following the provided link. 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 using indexed XML content getting high performances and providing version control for them. Another alternative solution is to add metadata directly to simulation files avoiding the creation of XML files. This solution has been discarded because has the following drawbacks: 1) it is difficult to find available and unused fields in the simulation files; 2) the simulation files are still 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.

From implementation point of view, two Java libraries have been used (Fig. 4.1): SolrJ to interact with the Solr Server and SVNKit to commit and update data to Subversion repository. Our solution meets also other industrial constraints, such as the 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.

5. Related Works. Aberdeen Group conducts market research studies to help businesses worldwide to improve performance[4]. They use a research methodology called P.A.C.E. to classify companies in three categories: best-in-class, average and laggard. Then they identify and compare companies using the internal and external pressures, their capabilities and the actions used to face the market challenges. The market research "*Getting Product Design Right the First Time with CFD*" [2] by Aberdeen Group studied the experience of 704 companies that perform simulations to design their products. Specifically, they use the Computational Fluid Dynamics (CFD) simulations to design the products. Their leading market research question is how the CFD simulations impact the product design and which are the key advantages of using them. The white paper includes a list of "actions" that are the steps to perform in order to increase the competitiveness of the companies on the market. Some of the actions are: *capture and document best practices for conducting simulations*, *centrally manage the simulation results and the best practices*, *take advantage of predefined wizards or templates to guide less experienced users.* The market research provides some starting points that must be further investigated, such as "*promote the collaboration*" among engineers, *ensure the right people have access to the results* and *offer version control.* Obviously, the market research does not discuss the technical solutions to achieve these actions.

We had the opportunity to work closely with professionals in Fiat Chrysler Automobiles (FCA) who use CFD simulations to design vehicle products. Our work further investigates the collaborative requirements of dispersed teams and co-located engineers gathered using interviews and a survey. Here, we analyse the survey requirements results enriching them with the stakeholders observations and feedback. Our work contributes also with technical solutions to meet the reported requirements. In [4], authors conducted a survey *to understand the needs and perception of practitioners about the Cloud-based simulation (CBS).* In their survey results come to light the need to share, store and retrieve models in CBS.

Many Web-based platforms have been created over the years to support Computational Fluid Dynamics. The "*e-Science Aerospace Integrated Research System*" (e-AIRS) [6] is an educational Web portal developed in Korea to help students to understand the aerodynamic simulation process [37]. EDISON_CFD [38] is the e-AIRS improvement in terms of stability, faster data response time and waiting time [39, 40]. Such systems have remarkable differences with our use case requirements and with Floasys. The systems target is the first difference, both e-AIRS and EDISON_CFD have an educational target, instead Floasys aims to industrial sectors (e.g., automotive sector). The e-AIRS target is educational and therefore it has been used in undergraduate and graduate classes. This have an impact on the integrated tools, that is the other difference. e-AIRS integrates custom in-house meshing tools and solvers. It operates with its own Fortran-based in-house CFD solvers [6]. Industries use widely adopted and validated CFD software, so Floasys platform aim is to integrate existing both commercial and open source solvers (Req. 6). In addition, the meshing is very important because it impacts on simulation quality results and running times. e-AIRS adopts a custom software called e-AIRSmesh to mesh the geometry storing the mash in a specific custom file format. Each CFD simulator works with a specific mesh topology. A Floasys requirement is to integrate multiple industrial adopted and validated CFD solvers (Req. 6). Industries have assistance contracts with CFD software vendors, so industrial platforms can not ignore their integration. In addition the aim is to avoid Vendor Lock-In adopting open format data.

Many other platforms proposed to manage simulations on HPC resources but they do not focus on collaboration among engineers. For example, a Web-based system for Management of CFD simulations for Civil Engineering was proposed with the goal to develop tools for civil engineers who are not CFD experts but need to perform CFD analysis [38]. It allows the "*dispatching and controlling of long-running simulations*" [38]. The system targets are civil engineers and CFD beginner users. The system was tested with a group of students in civil engineering class. The main differences concern the system end-user target and the correlated requirements to achieve. Our system target is automotive industry where CFD analysts need to collaborate, share data, result and knowledge, simulation data and result centralisation with the aim to promote collaboration. An interesting emerged common requirement is the need to use templates both for expert and beginner users. The nature of CFD simulations with high number of parameters to consider forces the creation of standard templates both to support beginner and expert users.

Another research avenue comes from the Semantic Web field. Many works in literature proposed software

[4]Aberdeen Group official web site `http://www.aberdeen.com/`

platforms for modelling and simulation. Simantics [41] is ontology based modelling; it uses ontologies to semantically describe the simulation model and the data. The two mainly applications that have built on Simantics platform are: the proprietary Apros6 for power plant M&S and an open source Simantics System Dynamics Tool based on Melodica language and the OpenMelodica environment. The Simantics's [41] developers are working on the integration of OpenFoam, an open source CFD software package.

6. Conclusions and Future Works. In this paper, we were able to identify key collaborative requirements for CFD design through the use of stakeholders interviews and a user survey. In addition we were able to address these requirements with an integrated, extensible and modular architecture. In this way, the paper provides the solutions and the technologies able to address the collaborative requirements. Requirements, solutions and technologies are tracked through the paper and their links are depicted in the Figure 2.6. Floasys is Web-based platform designed and developed to meet the collaborative requirements and is the industrial prototype currently under testing and evaluation in FCA.

Ideas behind Floasys, such as the integrated, extensible and modular architecture, could be adopted also in other contexts. The great opportunity to have different modules to plug in the architecture allows the deployment of a system tailored to engineers needs and development of some custom modules to embed team know-how. The solution to integrate existing engineering software and extract data from closed file format enables the creation of value added services over open format industrial data. In addition, large industries, independently by the sector, have multiple geographically distributed teams so, the collaboration around open format data and the sharing of data at different granularity and aggregation are great features. All features that could boost the industry competitiveness.

Floasys relies on mainstream open source solutions and its architecture is made integrating widely used existing enterprise technologies. The architecture can be divided into four main uncoupled parts: (1) simulators wrappers that communicate with the simulator software to get the simulation data and transform them in XML open format, (2) the version control repository for the XML files (e.g., SVN), (3) an enterprise search engine to index, cache and search the XML documents (e.g., Apache Solr), and (4) the central web server that provides the Web content (e.g., JBoss servlet container). Actually, we choose Apache Solr because it can scale using SolrCloud, Subversion because Floasys supports multiple SVN repositories and a mainstream servlet container. As future works, we have planned a controlled benchmark test to quantitatively assess and evaluate the Floasys performance, reliability and robustness. In addition, we are planning an evaluation study to analyse the usability of the Floasys user interface, and the user satisfaction when interacting with it [42, 43]. The user acceptance of the software will be investigated as well [44].

Finally, an interesting future work is the opportunity to link our SVN that contains simulation data in open format with an internal social network and enable the discussion on artefacts [45]. The aim is to understand and evaluate the benefits of using the social in the field of industrial CFD simulations.

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] S. Onggo, S. Taylor, and A. Tulegenov, *The need for cloud-based simulation from the perspective of simulation practitioners*, Proceedings of the Operational Research Society Simulation Workshop (SW14), 2014.

[5] C. Gargiulo, D. Pirozzi, V. Scarano, and G. Valentino, *A platform to collaborate around CFD simulations*, 23rd IEEE International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 2014), Parma, Italy, 23-25 June, 2014, pp. 205–210.

[6] J. Moon, C. Kim, Y. Kim, and K. W. Cho, *CFD Cyber Education Service using Cyberinfrastructure for e-Science*, in Fourth International Conference on Networked Computing and Advanced Information Management (NCM'08), 2008, vol. 2. pp. 306–313.

[7] V. Bertram and P. Couser, *Aspects of Selecting the Appropriate CAD and CFD Software*, 9th Conference on Computer and IT Applications in the Maritime Industries, Gubbio, Italy, 2010.

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

[9] Cunningham & Cunningham, Inc., *Anti-Pattern*, [Online]. Available: http://c2.com/cgi/wiki?AntiPattern, checked on 19/01/2015.

[10] J. Vlissides, R. Helm, R. Johnson, and E. Gamma, *Design patterns: Elements of reusable object-oriented software*, Reading: Addison-Wesley, vol. 49, p. 120, 1995.

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

[12] R. Shah, J. Kesan, and A. Kennis, *Lessons for open standard policies: a case study of the Massachusetts experience*, in Proceedings of the 1st international conference on Theory and practice of electronic governance, 2007.

[13] V. Varma, *Software Architecture: A Case Based Approach.* Pearson Education India, 2009.

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

[15] C.-W. Chang, P. Liu, and J.-J. Wu, *Probability-based cloud storage providers selection algorithms with maximum availability*, IEEE International Conference on Parallel Processing (ICPP), pp. 199–208, 2012.

[16] B. W. Fitzpatrick and J. Lueck, *The case against data lock-in*, Queue, vol. 8, no. 10, 2010.

[17] A. Geraci, F. Katki, L. McMonegal, B. Meyer, J. Lane, P. Wilson, J. Radatz, M. Yee, H. Porteous, and F. Springsteel, *IEEE standard computer dictionary: Compilation of IEEE standard computer glossaries.* IEEE Press, 1991.

[18] B. Bruegge and A. H. Dutoit, *Object-Oriented Software Engineering Using UML, Patterns and Java-(Required).* Prentice Hall, 2004.

[19] J. Humble and D. Farley, *Continuous delivery: reliable software releases through build, test, and deployment automation.* Pearson Education, 2010.

[20] *Eclipse public license.* [Online]. Available: http://www.eclipse.org/legal/epl-v10.html, checked on 19/01/2015.

[21] *Eclipse RAP Remote Application Platform*, [Online]. Available: http://eclipse.org/rap/, checked on 19/01/2015.

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

[23] X. Chen and K. Kasemir, *Bringing control system user interfaces to the web*, TUPPC078, ICALEPCS, vol. 13.

[24] G. Wang, *Improving data transmission in web applications via the translation between XML and JSON*, in Third International Conference on Communications and Mobile Computing (CMC), 2011, pp. 182–185.

[25] A. Solr, *Apache software foundation Solr*, 2014. [Online]. Available: http://lucene.apache.org/solr/, checked on 19/01/2015.

[26] R. Kuć, *Apache Solr 4 Cookbook.* Packt Publishing Ltd, 2013.

[27] D. Smiley and D. E. Pugh, *Apache Solr 3 Enterprise Search Server.* Packt Publishing Ltd, 2011.

[28] T. A. Howes, M. C. Smith, and G. S. Good, *Understanding and deploying LDAP directory services.* Addison-Wesley Longman Publishing Co., Inc., 2003.

[29] B. Arkills, *LDAP directories explained: an introduction and analysis.* Addison-Wesley, 2003.

[30] P. Tahchiev, F. Leme, V. Massol, and G. Gregory, *JUnit in action.* Manning Publications Co., 2010.

[31] *OpenFOAM GUIs* [Online]. Available: http://openfoamwiki.net/index.php/GUI, checked on 19/01/2015.

[32] D. Birsan, *On plug-ins and extensible architectures*, Queue, vol. 3, no. 2, Mar. 2005.

[33] T. H. Christopher L. Rumsey, Bruce Wedan and M. Poinot, *Recent Updates to the CFD General Notation System (CGNS)*, 50th AIAA Aerospace Sciences Meeting, 2012.

[34] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design patterns: elements of reusable object-oriented software.* Pearson Education, 1994.

[35] J. Walnes, J. Schaible, M. Talevi, G. Silveira, *et al.*, *XStream*, [Online]. Available: http://xstream.codehaus.org, 2011, checked on 19/01/2015.

[36] C. M. Pilato, B. Collins-Sussman, and B. W. Fitzpatrick, *Version control with subversion.* O'Reilly Media Inc., 2008.

[37] J. Moon, K. W. Cho, S.-H. Ko, J.-H. Kim, C. Kim, and Y. Kim, *A Cyber Environment for Engineering Cyber Education*, in IEEE Fourth International Conference on eScience, 2008, pp. 532–539.

[38] S. Lee and C. Kim, *Development and utilization of online computational environment for education and research in fluid engineering*, 2013.

[39] Y. Jung, J. Moon, D. Jin, B. Ahn, J. Seo, H. Ryu, O. Byeon, and J. Lee, *Web simulation service improvement on EDISON_CFD*, Computer Science and Technology, 2012.

[40] Y. J. Jung, J. Moon, D.-S. Jin, B.-Y. Ahn, J. H. Seo, H. Ryu, O.-H. Byeon, and J. R. Lee, *Performance Improvement for Web based Simulation Service on EDISON_CFD*, 2013.

[41] *Simantics platform*, [Online]. Available: https://www.simantics.org/simantics/about-simantics/simantics-platform/, checked on 19/01/2015.

[42] *Questionnaire for user interface satisfaction.* [Online]. Available: http://oldwww.acm.org/perlman/question.cgi?form=QUIS, checked on 19/01/2015

[43] *Computer system usability questionnaire*, [Online]. Available: http://oldwww.acm.org/perlman/question.cgi?form=CSUQ, checked on 19/01/2015.

[44] F. D. Davis, *Perceived usefulness, perceived ease of use, and user acceptance of information technology*, MIS quarterly, pp. 319–340, 1989.

[45] D. Malandrino, I. Manno, A. Negro, A. Petta, V. Scarano, and L. Serra, *Social team awareness*, in 9th International Conference Conference on Collaborative Computing: Networking, Applications and Worksharing (Collaboratecom), 2013, pp. 305–314.

Appendix A. Requirements elicitation Survey.

The survey aims to identify the most important requirements for a collaborative simulation platform to support the engineering activities. Survey is confidential and all data will be processed in aggregated way. Thank you for your time and your advice. At the end we will provide you the survey results and considerations.

Your experience.

Q1. Which is your role in the company?
- CFD analyst
- Technical Manager
- Performance Engineer

Q2. Which is your place of work?
- *(FCA) Pomigliano D'Arco (Naples)*
- *(FCA) Orbassano (Turin)*

Q3. How many years you spent working in the CFD field?
- *(write the number of years)*

Q4. How many simulations do you perform per year?
- *(write the number of simulations per year)*

Collaboration among analysts and data sharing.

Q5. In my office I daily work with a number of analysts equal to
- *(write the number of analysts)*

Q6. I daily work with a number of analysts in a different place equal to
- *(write the number of analysts)*

Q7. On average, the geometries file size in average is
- *(write the geometry file size)*

Q8. On average, the simulations file size in average is
- *(write the simulation file size)*

Q9. In order to exchange geometries and simulations files I usually use

E-mail:	(Never) 1	2	3	4	5	6	7 (Always)
Chat:	(Never) 1	2	3	4	5	6	7 (Always)
Phone:	(Never) 1	2	3	4	5	6	7 (Always)
FTP:	(Never) 1	2	3	4	5	6	7 (Always)
Ask to a colleague:	(Never) 1	2	3	4	5	6	7 (Always)
Internal Platform:	(Never) 1	2	3	4	5	6	7 (Always)

Q10. In order to exchange documents I usually use

E-mail:	(Never) 1	2	3	4	5	6	7 (Always)
Chat:	(Never) 1	2	3	4	5	6	7 (Always)
Phone:	(Never) 1	2	3	4	5	6	7 (Always)
FTP:	(Never) 1	2	3	4	5	6	7 (Always)
Ask to a colleague:	(Never) 1	2	3	4	5	6	7 (Always)
Internal Platform:	(Never) 1	2	3	4	5	6	7 (Always)

Data centralisation and simulation data search.

Q11. I follow some rules to store simulations and assign the name to their corresponding files

(Never) 1 2 3 4 5 6 7 (Always)

Q12. The information that I store in the simulation file name are *(Multiple choice)*
- *Project Name*

- *Release*
- *Ground Clearance*
- *Revision*
- *Engine*
- *Vehicle Trimming*
- *Date*

Q13. The rules are:
- *Personal Choice*
- *Team Conventions*
- *Fixed imposed rules*

Q14. I follow the rules over time
(Never) 1 2 3 4 5 6 7 (Always)

Q15. The opportunity to link other information (e.g., tags) to files could be:
(Useless) 1 2 3 4 5 6 7 (Useful)

Q16. In order to find simulation files I usually use

My mind:	(Never) 1	2	3	4	5	6	7 (Always)
Free directory navigation:	(Never) 1	2	3	4	5	6	7 (Always)
Windows Find Tool:	(Never) 1	2	3	4	5	6	7 (Always)
See the file name:	(Never) 1	2	3	4	5	6	7 (Always)
Open the simulation:	(Never) 1	2	3	4	5	6	7 (Always)
File History:	(Never) 1	2	3	4	5	6	7 (Always)
Unix Find Tool:	(Never) 1	2	3	4	5	6	7 (Always)
Ask to a coworker:	(Never) 1	2	3	4	5	6	7 (Always)

Q17. I have a tool to search simulations according to their content
(Never) 1 2 3 4 5 6 7 (Always)

Q18. A tool to support the search operations based on simulation data could be:
(Useless) 1 2 3 4 5 6 7 (Useful)

Simulation data versioning.

Q19. I have a tool to show the simulation's modification over time
(Never) 1 2 3 4 5 6 7 (Always)

Q20. A tool to show the simulation revisions could be
(Never) 1 2 3 4 5 6 7 (Always)

Used simulators.

Q21. During my work I use these simulator software (multiple choices):
- *Star-CCM+*
- *OpenFOAM*
- *SolidWorks*
- *PowerFlow*
- *CFD++*

Edited by: Giacomo Cabri
Received: September 15, 2014
Accepted: January 5, 2015