UNIVERSITÀ
DEGLI STUDI DI
SALERNO

.DIEM

Ministero dell'Istruzione,
dell'Università e della Ricerca

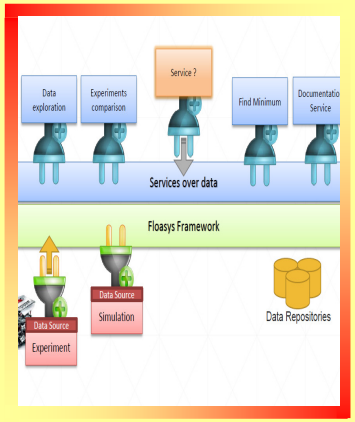
Dottorato di Ricerca in Informatica e Ingegneria dell'Informazione

C. Gargiulo - Teamwork Collaboration around CAE Models in an Industrial Context

Dipartimento di Ingegneria dell'Informazione ed Elettrica e
Matematica applicata/DIEM

Teamwork Collaboration around CAE Models in an Industrial Context

Claudio Gargiulo



Coordinatore: **Prof. Alfredo De Santis**

Tutor: **Prof. Vittorio Scarano**

Anno Accademico 2016-2017, XXIX Ciclo

Università degli Studi di Salerno

Dipartimento di Ingegneria Elettronica ed Ingegneria Informatica

Dottorato di Ricerca in Ingegneria dell'Informazione
XXIX Ciclo

TESI DI DOTTORATO

Teamwork Collaboration around CAE Models in an Industrial Context

CANDIDATO: **CLAUDIO GARGIULO**

TUTOR: **PROF. VITTORIO SCARANO**

COORDINATORE: **PROF. ALFREDO DE SANTIS**

Anno Accademico 2016 – 2017

Supervisor

Prof. Vittorio Scarano, Dip. di Informatica,

Università degli Studi di Salerno, Italy

Graduate Group Chair

Prof. Alfredo De Santis, Dip. di Informatica,

Università degli Studi di Salerno, Italy

Date of defense

June, 2017

Claudio Gargiulo
ISISLab, Dip. di Informatica
Università degli Studi di Salerno
Via Giovanni Paolo II, 84084 Fisciano (Salerno), Italy
E-mail: c.gargiulo25@studenti.unisa.it

Acknowledgements

I am using this opportunity to express my gratitude to University of Salerno that has provided to me and my colleagues a great environment with all the infrastructures to study and live over five years since my first day in Lancusi on September 2012. In particular, I would like to thank professor Vittorio Scarano and ISISLab to be involved in an active and stimulating research group for my PhD thesis.

I have to thank Donato Pirozzi, Delfina Malandrino and Andrew Fish for the interesting and stimulating discussions about very different technical and non-technical topics.

And ... of course a special thanks to my wife Maria Silvia.

Abstract

Le aziende di medie e grandi dimensioni devono ogni giorno competere in un contesto mondiale. Per raggiungere una maggiore efficienza nei loro prodotti/processi sono costrette a globalizzarsi aprendo più sedi in luoghi geograficamente distanti tra loro. In tale contesto, persone dello stesso team o afferenti a team diversi devono lavorare insieme indipendentemente dal fuso orario e dal luogo in cui si trovano. Pertanto un team "virtuale" è costituito da gruppi di persone geograficamente lontane che riescono a coordinarsi con l'ausilio delle nuove tecnologie.
Gli strumenti e le metodologie a supporto del *"Computer Supported Cooperative Work"* (CSCW) possono agevolare la collaborazione riducendo problematiche legate alla distanza e al tempo.
I principali obiettivi che il CSCW si prefigge di ottenere all'interno di una organizzazione complessa sono di seguito elencati:

- Pianificare, monitorare, e visualizzare l'avanzamento delle varie fasi di progetto fino al suo completamento (Project Management)
- Condividere, revisionare, approvare o respingere proposte di progetto provenienti da altri membri del gruppo di lavoro (Authoring Systems)
- Gestione Collaborativa di attività e reportistica all'interno di un processo di business basato sulla conoscenza (Workflow Management)
- Collezionare, organizzare, gestire e condividere l'informazione sotto varie forme (Knowledge Management)
- Poter taggare, organizzare, condividere, e ricercare dati globali attraverso un engine di collaborazione (Enterprise Bookmarking)
- Collezionare, organizzare, gestire e condividere informazioni legate alla delibera finale di un progetto (Extranet Systems)
- Condividere informazioni di natura generale fra tutti i membri di una organizzazione (Intranet Systems)
- Organizzare relazioni sociali di gruppo (Social Network)
- Collaborare e condividere dati strutturati (Online SpreadSheet)

Il presente lavoro prende spunto dagli obiettivi principali su esposti e attraverso una attività di ricerca con esperienza diretta sul campo ne verifica la rispondenza e ne garantisce la applicabilità.
Il contesto reale è costituito da team virtuali di ingegneri e al modo in cui collaborano all'interno dell'industria automotive. L'iter della attività di ricerca si può riassumere nei seguenti passi: (1) sono stati identificati i principali requisiti collaborativi ed ingegneristici facendo riferimento ad un caso d'uso reale all'interno di Fiat Chrysler Automobiles; (2) ogni requisito è stato soddisfatto implementando un`architettura integrata, modulare ed estendibile; (3) è stata progettata, implementata e testata una piattaforma chiamata Floasys di raccolta, centralizzazione e condivisione di simulazioni; (4) è stato progettato un tool denominato ExploraTool per esplorare visivamente un repository di simulazioni all'interno di Floasys; (5) sono state identificate le possibili estensioni della piattaforma in termini di multidisciplinarietà e di multisettorialità; (6) a valle di tutto il processo, sono stati verificati tutti i requisiti che un CSCW si prefigge di soddisfare.

La fase iniziale del lavoro si è concentrata sulla raccolta di requisiti collaborativi e delle relative esigenze che emergono nel momento in cui differenti team geograficamente lontani (virtual teams) si ritrovano a collaborare per perseguire un risultato comune.
I requisiti collaborativi identificati per supportare la collaborazione tra team geograficamente lontani sono: centralizzare i dati delle simulazioni, fornire la possibilità di annotare ed aggiungere metadati ai file, fornire un motore di ricerca per ottenere simulazioni completate da altri analisti, fornire il versioning dei dati e supportare la loro condivisione. In linea con i requisiti individuati è stato sviluppato un prototipo di piattaforma collaborativa (CSCW) chiamato Floasys.
I clienti finali di Floasys sono in prima istanza tutte le industrie che utilizzano le simulazioni CAE per progettare i loro prodotti, quindi, le industrie automotive, aeronautiche e navali, etc.
Floasys colleziona i dati delle simulazioni, li memorizza in formato aperto XML e li centralizza in un repository condiviso; fornisce inoltre ulteriori servizi sui dati raccolti memorizzati in formato aperto, ad esempio la possibilità di annotare i file oppure di cercare all'interno del repository indipendentemente dal simulatore con cui sono stati generati.
Risulta di estrema utilità il poter recuperare le simulazioni effettuate da altri membri dello stesso team o di team diversi al fine di poter confrontare le prestazioni di un progetto in corso. Per poter fornire questi servizi vanno considerati vari aspetti: sicuramente i servizi appena elencati debbono essere immersi in un contesto aziendale già esistente con relative pratiche, workflow e sistemi software esistenti. Per portare un esempio concreto la sola centralizzazione dei dati delle simulazioni implica la comunicazione con i software di simulazione esistenti mitigando il problema del Vendor Lock-In ovvero la forte dipendenza dagli stessi simulatori.
Da un punto di vista architetturale, Floasys soddisfa i requisiti non funzionali di estendibilità e modularità. In questo modo il sistema può essere adattato alle necessità dei clienti, aperto a soddisfare necessità future ed essere usato in altri dipartimenti.
L'architettura modulare ed estendibile di Floasys è stata ottenuta basandosi sul concetto di plug-in. Sebbene l'attività di ricerca riguarda direttamente il settore automotive, i requisiti raccolti e le difficoltà descritte sono comuni anche ad altri settori come descritto in letteratura. Per cui molte delle considerazioni fatte in questo lavoro e le soluzioni adottate possono essere riutilizzate per altri tipi di simulazione oltre che per i dati ottenuti da esperimenti.
Infine, all'interno di Floasys è stato integrato un tool interattivo detto “ExploraTool” per la visualizzazione, l'esplorazione e l'interrogazione di repository di simulazioni. Sebbene l'idea di questo tool sia nata nel contesto della navigazione dei repository di simulazioni, esso è generico e può essere utilizzato con qualsiasi dataset. Il tool si basa sui diagrammi di Eulero-Venn. L'universo è l'insieme di tutte le simulazioni memorizzate in uno o più repository. I gruppi di simulazioni vengono rappresentati mediante ellissi innestate. Usando tale tool, gli analisti possono esplorare il repository attraverso operazioni di drill-down e roll-up per ottenere più o meno dettagli. Andando giù nella gerarchia l'utente filtra gli item all'interno del dataset effettuando a tutti gli effetti una query grafica. In questo modo l'utente esplora il repository ottenendo alla fine due o più simulazioni da comparare. Dopo la fase di ideazione, progettazione ed implementazione, il tool è stato testato con utenti reali allo scopo di ottenere dati sulla sua usabilità.

Publications

The content of this thesis has been published in the following conferences and journal:

1. Gargiulo, Claudio; Pirozzi, Donato; Scarano, Vittorio. ***An Architecture for CFD Workflow Management.*** In Proceedings of the 11th IEEE International Conference on Industrial Informatics (INDIN), Bochum, Germany, July 29-31, 2013, pp. 352-357.

2. Gargiulo, Claudio; Pirozzi, Donato; Scarano, Vittorio; and Valentino, Giuseppe. ***A platform to collaborate around CFD simulations.*** In Proceedings of the 23rd IEEE International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), Parma, Italy, June 23-25, 2014, pp. 205-210.

3. Gargiulo, Claudio; Malandrino, Delfina; Pirozzi, Donato; Scarano, Vittorio. ***Simulation Data Sharing to foster Teamwork Collaboration.*** Journal on Scalable Computing: Practice and Experience, Scientific International Journal for Parallel and Distributed Computing 2014.

4. Fish, Andrew; Gargiulo, Claudio; Pirozzi, Donato; Scarano, Vittorio. ***Simulation Repository Visualisation and Exploration.*** In Proceedings of the 13th IEEE International Conference on Industrial Informatics (INDIN), Cambridge, UK, 22-24 July, 2015.

5. Fish, Andrew; Gargiulo, Claudio; Malandrino, Delfina; Pirozzi, Donato; Scarano, Vittorio. ***Visual Exploration System in an Industrial Context.*** Journal IEEE Transactions on Industrial Informatics 12 (2), 567-575, April, 2016.

The papers are attached at the end of this thesis.

Abbreviations

This section is a list of abbreviations and acronyms used through the thesis.
They are listed in alphabetic order.

API Application Programming Interface
BOM Bill of Materials
CAD Computer-Aided Design
CAE Computer-Aided Engineering
CAM Computer-Aided Manufacturing
CBS Cloud-Based Simulation
CFD Computational Fluid Dynamics
CLI Command Line Interface
CRM Customer Relationship Management
CSCW Computer Supported Collaborative Work
CSW Comma-separated values
DM Data Management
DNS Domain Name System
DOE Design Of Experiments
ERP Enterprise Resource Planning
FCA Fiat Chrysler Automobiles
FEA Finite Element Analysis
FEM Finite Element Method
GUI Graphical User Interface
HCI Human Computer Interaction
HPC High Performance Computing
ICT Information and Communication Technology
IDE Integrated Development Environment
JAR Java ARchive
JSON JavaScript Object Notation
MRP Material Resource Planning
M&S Modeling and Simulation
NFR Non-Functional Requirement
NPD New Product Development Process
PDM Product Data Management
PLM Product Lifecycle Management
RPM Revolutions Per Minute
SCM Supply Chain Management
SME Small and Medium Enterprise
SMEs Small and Medium Enterprises
SSH Secure SHell
TTM Time-To-Market
UI User Interface
WBS Web-Based Simulation
WWW World Wide Web
XML eXtensible Markup Language

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Contents

This introductory chapter provides an overview of the thesis content. It introduces briefly the terminology used since the title page and through the entire work. It evolves around the concepts of virtual teams, Enterprise 2.0 and Trialogic learning discussed in a real industrial use case. The chapter aims to discuss the dissertation goals and contributions giving to the reader a link between them and the next chapters.

1.1 Enterprise Collaboration

Nowadays small and medium enterprises (SMEs) as well as large industries are world-wide and work in a global market. Their organisation is often dislocated over multiple nations and companies and faces time, space and cultural barriers. In a high competitive world market and economy, companies face the need for a fast time-to-market, low cost and rapid product development [1]. The introduction of Internet and new communication technologies can enable the collaboration among dispersed team members overcoming geographical, temporal, cultural and organisational boundaries.

The term **collaboration** refers to the process of two or more parties who work jointly towards a common goal [2]. In according to it, we define the **collaborative technology** as computer tools that support *communication, coordination, and/or information processing needs of two or more people working together on a common task* [2].

Supported Cooperative Work (CSCW) and groupware are well-known computer technologies to collaborate together, and study how people use technology to work together.

Enterprise collaboration refers to communication among the employees of a corporate that may encompass the use of collaborative technologies like collaboration platforms, enterprise social networks and corporate Internet. For instance, within the enterprise well-established communication tools are the e-mails and the video-conferencing systems.

In according to the collaborative technology there are two dimensions: *space* and *time*. *Space* means where the persons who collaborate together are located. For instance, persons could be in the same room discussing face-to-face or they can be geographically distant each other discussing through a video-conference system. *Time* concerns whether the persons collaborate at same time or can collaborate also in different moments, so the key terms are respectively **synchronous** and **asynchronous** collaboration. To further understand this distinction it is really useful to consider the CSCW Quadrants. The combination of the time and space dimensions generates four quadrants useful to classify the existing or future collaborative technologies. For instance, the e-mail is a tool that supports the asynchronous communication among distant people and people who work in the same room. The communication is asynchronous just because the sending and receiving events occur in different times. Instead a video conference system is mainly used by users that are far from each other and communicate together at same moment.

		TIME	
		Same Time (Synchronous)	Different Time (Asynchronous)
SPACE	Same Space	**1st Quadrant** Spontaneous collaborations, formal meetings, classrooms	**2nd Quadrant** Design rooms, Project scheduling
	Distributed	**3rd Quadrant** Video conferencing, net meetings, phone calls	**4th Quadrant** Emails, writing, voice mails, fax

Figure 1.1: CSCW Quadrants (Adapted form [2]).

In the context of Enterprise Collaboration, another very common keyword is **virtual team** that encompasses all the collaboration within the enterprise among teams that are geographically far from each other.

1.1.1 Virtual Teams Collaboration

In team-based organisations, participants are grouped into teams. A **team** is a group of people with a full set of complementary skills who work on the same activity or tasks towards a common goal [3,4]. The term **teamwork** highlights exactly the concept of people (a team) who work together toward the same goal. Their actions are interdependent, but they are fully committed to a single result. Teamwork means that people will try to cooperate, using their individual skills providing constructive feedback, despite any personal conflict between individuals.
When the members of a team are geographically distributed over different places and use the modern communication technologies to coordinate their work we talk about **virtual teams** also known as dispersed teams. A team is virtual when all the following characteristics are true:

- geographically dispersed teams sometimes over different time zones;

- driven by common purpose;
- enabled by communication technologies [5], such as e-mail, video-conferencing, telephone, etc.

Today complex products like automotive products are designed in collaboration with the Suppliers directly involved in the design process. Hence, also teams belong to different organisations can collaborate together. Therefore, additional characteristics for virtual teams could be:

- they can belong to different companies;
- the teams are not necessary permanent;
- often the teams have small size;
- the team members are knowledge workers.

Summarising these characteristics, the definition of virtual team is:

Virtual Team definition *[1]: (small temporary) groups of geographically, organizationally and/or time dispersed knowledge workers who coordinate their work predominantly with electronic information and communication technologies in order to accomplish one or more organization tasks.*

An important keyword of the Virtual Team Definition is the **coordination** predominantly with electronic technologies so all interactions are virtual. Of course, it is important to highlight also the virtual teams disadvantages or their difficulties [1]. The lack of physical interactions, reduced face-to-face synergies and the lack of social interactions are difficulties to take into account. The introduction of Virtual Teams within an enterprise is not easy and, as known in literature [7], requires a heavy change in the project management. Hence, difficulties are both technological and non-technological, and must be evaluated before to leverage on a virtual team.

1.1.2 Enterprise 2.0

The **Enterprise 2.0** known also as E2.0 is the use of *emergent social software platforms* within an organisation to pursue its goals [8]. Social software supports the rendezvous, connection, and collaboration among users who form online communities. The term *emergent*, used in the previous paragraph, refers to the use of E2.0 software that is freeform, optional, without predefined workflows and indifferent to formal hierarchies. In contrast, enterprises usually use software that have standard workflows. For instance, Customer Relationship Management (CRM) and Enterprise Resource Planning (ERP) software are two type of system that every enterprise uses within and outside its organisation. They define exactly the workflows to follow, the roles and what does everyone.
On the opposite side E2.0 are completely free without predefined structure. Of course, Enterprise 2.0 technologies can be used with Virtual Teams to overcome their limitations. For instance, E2 aims to introduce social interactions within the enterprise that is exactly a virtual team limitation. Of course, the introduction of E2.0 technology must be carefully evaluated and actually not all industries have it.

1.1.3 Trialogic Learning

In the **Trialogic Learning** [9] learners collaborate around shared objects developing, transforming, or creating other shared objects in a systematic fashion for some later use.
The triad in the Trialogic Learning definition is made by:

- The subjects or learners,
- The collaboratively development, transformation or object creation
- For later use or other users

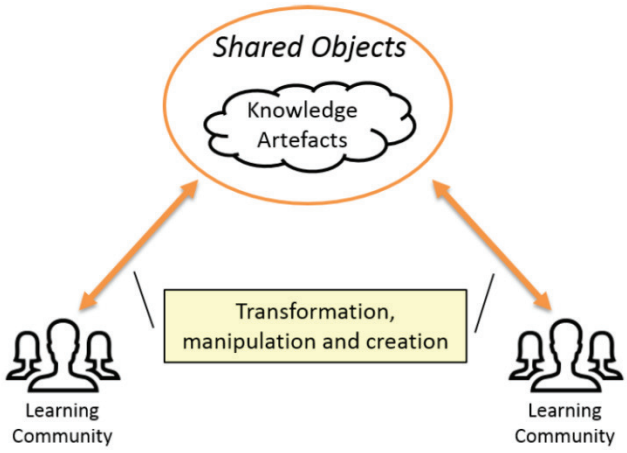


Figure 1.2: Trialogic Learning elements.

Trialogic learning concentrates on the interactions among people through developing common shared objects. The community gets collaboratively insight into knowledge objects and also collaboratively works with knowledge artefacts to develop other knowledge objects or transform the previous ones.
Specifically, **shared objects** [9] are knowledge artefacts, practices or processes developed collaboratively for later use. Artefacts can be conceptual or concrete. Examples of conceptual artefacts are ideas, plans and designs, whereas concrete artefacts are prototypes. These objects have a knowledge content and one can perform actions on them transforming objects and their knowledge content. The representative example of Trialogic Learning is Wikipedia. The shared objects are the wikipedia articles that are updated by the communities of users for communal use.
In the context of this thesis the three elements of the Trialogic Learning exist. In the engineering context, multiple engineers collaborate together to simulate products (the shared objects) to use later to make the products.

1.2 CSCW Research topics

Collaborative software was originally designated as *groupware* and this term can be traced as far back as the late 1980s, when Richman and Slovak (1987) wrote:

"Like an electronic sinew that binds teams together, the new *groupware* aims to place the computer squarely in the middle of communications among managers, technicians, and anyone else who interacts in groups, revolutionizing the way they work."

Even further back, in 1978 Peter and Trudy Johnson-Lenz coined the term groupware; their initial 1978 definition of groupware was, “intentional group processes plus software to support them.” Later in their article they went on to explain groupware as “computer-mediated culture... an embodiment of social organization in hyperspace."

Groupware integrates co-evolving human and tool systems, yet is simply a single system.

In the early 1990s the first commercial groupware products were delivered, and big companies such as Boeing and IBM started using electronic meeting systems for key internal projects. Lotus Notes appeared as a major example of that product category, allowing remote group collaboration when the internet was still in its infancy. Kirkpatrick and Losee (1992) wrote then:

"If GROUPWARE really makes a difference in productivity long term, the very definition of an office may change. You will be able to work efficiently as a member of a group wherever you have your computer. As computers become smaller and more powerful, that will mean anywhere."

1.2.1 Design and implementation issues

The complexity of groupware development is still an issue. One reason for this is the socio-technical dimension of groupware. Groupware designers do not only have to address technical issues (as in traditional software development) but also consider the organizational aspects and the social group processes that should be supported with the groupware application. Some examples for issues in groupware development are:

- Persistence is needed in some sessions. Chat and voice communications are routinely non-persistent and evaporate at the end of the session. Virtual room and online file cabinets can persist for years. The designer of the collaborative space needs to consider the information duration needs and implement accordingly.
- Authentication has always been a problem with groupware. When connections are made point-to-point, or when log-in registration is enforced, it's clear who is engaged in the session. However, audio and unmoderated sessions carry the risk of unannounced 'lurkers' who observe but do not announce themselves or contribute.
- Until recently, bandwidth issues at fixed location limited full use of the tools. These are exacerbated with mobile devices.
- Multiple input and output streams bring concurrency issues into the groupware applications.
- Motivational issues are important, especially in settings where no pre-defined group process was in place.
- Closely related to the motivation aspect is the question of reciprocity. Ellis and others have shown that the distribution of efforts and benefits has to be carefully balanced in order to ensure that all required group members really participate.

One approach for addressing these issues is the use of design patterns for groupware design. The patterns identify recurring groupware design issues and discuss design choices in a way that all stakeholders can participate in the groupware development process.

1.2.2 Groupware and levels of collaboration

Groupware can be divided into three categories depending on the level of collaboration:

1. **Communication** can be thought of as unstructured interchange of information. A phone call or an IM Chat discussion are examples of this.
2. **Conferencing** (or collaboration level, as it is called in the academic papers that discuss these levels) refers to interactive work toward a shared goal. Brainstorming or voting are examples of this.
3. **Co-ordination** refers to complex interdependent work toward a shared goal. A good metaphor for understanding this is to think about a sports team; everyone has to contribute the right play at the right time as well as adjust their play to the unfolding situation - but everyone is doing something different - in order for the team to win. That is complex interdependent work toward a shared goal: collaborative management.

1.2.3 Collaborative management (coordination) tools

Collaborative management tools facilitate and manage group activities. Examples include:

- ***Electronic calendars*** (also called time management software) — schedule events and automatically notify and remind group members
- ***Project management systems*** — schedule, track, and chart the steps in a project as it is being completed
- ***Online proofing*** — share, review, approve, and reject web proofs, artwork, photos, or videos between designers, customers, and clients
- ***Workflow systems*** — collaborative management of tasks and documents within a knowledge-based business process
- ***Knowledge management systems*** — collect, organize, manage, and share various forms of information
- ***Enterprise bookmarking*** — collaborative bookmarking engine to tag, organize, share, and search enterprise data
- ***Prediction markets*** — let a group of people predict together the outcome of future events
- ***Extranet systems*** (sometimes also known as 'project extranets') — collect, organize, manage and share information associated with the delivery of a project (e.g.: the construction of a building)
- ***Intranet systems*** — quickly share company information to members within a company via Internet (e.g.: marketing and product info)
- ***Social software systems*** — organize social relations of groups
- ***Online spreadsheets*** — collaborate and share structured data and information
- ***Client portals*** — interact and share with your clients in a private online environment

1.3 Dissertation Goals and contributions

In large industries, such as automotive industries, the engineering teams are distributed over multiple locations. The geographically separation among workers introduces new requirements that are not covered by existing engineering software, and introduces new challenges for software engineers. This thesis is related to a real use case of two separated engineering teams of Fiat Chrysler Automobiles (FCA) that collaborate together with the aim to design automotive products.
The first goal of this work is to get insight into the real use case understanding the engineering context and providing an overview of the actual collaboration among team members. In order to be compliant to a standard methodology, this step refers to the standard software engineering methodology for the requirements elicitation. Therefore, the main outcomes of this requirements elicitation activity is a list of collaborative and engineering requirements gathered within a real use case and compared with the existing literature. The availability of this real use case is a really valuable aspect of this work because the reporting of collaborative requirements within the simulation engineering context represents a significant distinction and novelty compared to the existing work published in literature. The interaction with end-users has potentially enormous advantages for the immediate feedback.
The systematic collection, classification and analysis of both collaborative and engineering requirements has been performed through an **agile methodology** that involves directly, immediately and continuously the end-users and stakeholders from the beginning through short, repetitive and close steps directly on the field. Three approaches have been used: on-site observations working directly with simulation analysts in FCA, stakeholders interviews and a user survey between two dispersed teams. Of course, this requirements elicitation is not a one-shot activity but it has been performed continuously from the beginning until the end of the project.
The collection, classification and analysis of gathered requirements is only the first step, the next step is to design, implement and test a platform to deploy in a real setting like FCA. This work contributes with Floasys a collaborative web-based platform useful to collect, centralise and share simulations among engineers of different dispersed teams.
A collaborative platform is not an isolated island but it must be integrated with the industrial ecosystem. Therefore, the development of a new platform and its deployment within the industry implies the integration of the existing software, procedures, best practices and so on. In addition, different issues must be take into account like the integration of existing software, heterogeneities among software and already existing internal workflows and practices.
To summarise, the main goals achieved through this work are:

- Collection, identification and analysis of the key collaborative requirements of dispersed teams within a real industrial use case (FCA use case);
- Design of an integrated, extensible and modular software architecture;
- Design, implementation and testing of a real working prototype called Floasys with its collaborative and engineering functionalities;
- Design, implement and test a generic tool called ExploraTool to visually explore large datasets in hierarchical way;
- Extension to different disciplines (like Structure, Thermal, Fatigue) and different contexts (like Aeronautic, Rail and Naval sectors).
- Final verification of CSCW research requirements in the developed prototype.

1.4 Dissertation Roadmap

This thesis has been organised logically starting from the requirements elicitation activity and the gathered collaborative and engineering requirements. Then, it introduces Floasys, the Web-based platform designed to address the key stakeholders' requirements and able to collect, centralise and share simulations among engineers. A part of the thesis is allocated to describe the Floasys architecture and how it solves the data heterogeneity issue through a modular and extensible architecture. Another part discusses an interactive tool to visualise, explore and query repository of simulations and experiments. The last chapter covers the multi-context and multi-disciplinary issues.
The overall dissertation has been organised as follows:

- Chapter 2 introduces, describes and analyses the collaborative and engineering requirements gathered through on site observations, stakeholders interviews and an on-line survey.

- Chapter 3 introduces Floasys, its features and its graphical user interface (GUI). Of course, the functionalities have been divided in two parts: collaborative and engineering features.
  In the 2nd part it describes the Floasys integrated, extensible and modular architecture and maps the architecture features with the provided functionalities and the gathered requirements.
  In the final part it introduces the problem of large dataset navigation and exploration like simulation and experimental repositories and describes the general idea behind the ExploraTool.

- Chapter 4 describes the platform extensions in terms of multi-context and multi-disciplinary issues.

- Chapter 5 concludes the thesis verifying the CSCW research topics satisfaction and reporting the possible future directions to investigate.

Chapter 2

Collaborative Requirements

Contents

Small and medium enterprises (SMEs) as well as large industries are organised in multiple geographically distributed teams that collaborate together. Fiat Chrysler Automobiles is a large industry. It has many engineering teams around the world that collaborate together to design vehicle products. The goal is to work closely with a team of professionals to get insight and gather the key collaborative requirements in the engineering field. It is a challenging goal due the different involved stakeholders but at same time represents a significant and relevant use case. This chapter reports, analyses and discusses the key collaborative Functional and Non-Function 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 performed working closely with two engineering teams in Fiat Chrysler Automobiles (FCA) in Pomigliano D'Arco (Napoli) and Torino. The requirements are gathered through observations, stakeholders interviews and a user survey.
This chapter is organised as following. It introduces synthetically the main collaborative requirements in the Section 2.1. The Section 2.3 describes the existing methodologies for the requirements elicitation activity describing how these have been conducted in the Fiat Chrysler Automobiles use case. Then, it deeply discusses every single requirement providing the survey results and user comments collected during the interviews and meetings. After introducing briefly the engineering context and the simulation workflow, the chapter provides an overview of both Functional and Non-Functional requirements with a detailed description for some of them. Finally, the chapter provides the possible future works especially in terms of a platform to automatically manage multiple simulation workflow iterations considering both Multi-Disciplinary and Multi-Objective simulations.

2.1 Introduction

Fiat Chrysler Automobiles (FCA), as many other large industries, is organised in multiple geographically distributed teams that collaborate together. Through the 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).

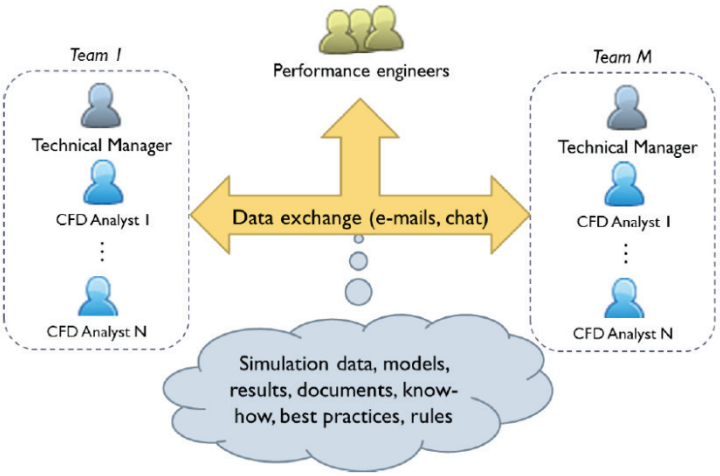


Figure 2.1: Collaboration among geographically distributed teams.

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 topic (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 [10] 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 the 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 analysis, 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).

	Requirement	Notes
Req. 1	*Simulation Data centralisation*	
Req. 2	*Metadata over simulation data*	Free tags, structured information.
Req. 3	*Search facility*	based on file names, 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*	

Table 2.1: Stakeholders' Collaborative Requirements.

Through, the survey and interviews many requirements have gathered. All of them were filtered to get the list of key collaborative requirements shown in Table 2.1. These key collaborative requirements drive the design of a collaborative platform to foster collaboration among group of engineers who perform simulations. Nevertheless, the analysed context considers mainly simulations practitioners, many of these requirements could be valid also in other engineering contexts (i.e., aerospace, naval).

2.2 Related Works

Aberdeen Group conducts market research studies to help businesses worldwide to improve performance. 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*" [11] 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*. The 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. The work contributes also with technical solutions to meet the reported requirements.
In [12], 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.

2.3 Methodology

This section aims to review the existing literature methods to gather stakeholders' requirements and summaries the used method along its advantages and drawbacks. In this way, the work can be replicated in the same field or another field following the same methodology.
The used methodology involves the stakeholders since the beginning in the following activities: plan, design, and test. The used requirements elicitation methodology has the following main steps:

- Stakeholders identification;
- Domain understanding;
- Tasks identification.

The previous requirements elicitation activities are not one shot activities but I followed an agile methodology with short iterations of two weeks each. For each iteration, stakeholders, simulation analysts and technical managers have been immediately involved in the requirements elicitation process. The main tools that I have used are:

- Open discussions and informal meetings with a small group of simulation analysts;
- *R*equirements elicitation survey;
- *D*iscussions using mockups and system prototypes of the gathered requirements.

In the following I focus on two steps: the stakeholders identification and the domain understanding.

2.3.1 Stakeholders identification

The stakeholders are the people affected introducing a new system in an organisation. System stakeholders are not limited to top management that pays for the system, but more important are the **people** (actors [3]) **that will directly use the system**. In addition, also people that do not directly use the system and are indirectly affected by it must be considered as stakeholders. For instance, customers that will place the phone orders will be affected and must be considered as stakeholders. The stakeholders identification and their classification are fundamental activities to perform before requirements elicitation. Of course, the list of stakeholders can change and be updated many times.
The CUSTOM approach [13] as explained by Dix et al. [14] classifies the stakeholders in four groups: primary, secondary, tertiary and facilitating. One should be sure to meet the requirements of all stakeholders but often they can be complex and in conflict each other [14].
One could ask why it is important to identify all the stakeholders. In the book "*Human-Computer Interaction*" [14], the authors described an example of organization with different departments, each one with its computer system, and the decision of the top management to integrate them together to share sales, marketing and stock data. The introduction of the system without taking care of salesmen, responsible for marketing and storekeeper, leads to a paradoxical situation, in which, for instance, salesmen are unhappy to share their customers contacts with the marketing and keep them in personal files. The main concern is that all organisations have formal and informal communication structures that contribute to the overall organisation working. Identifying correctly the stakeholders uncovers hidden information transfers and highlights how the information flow across the structures. Introducing a new system, one must be really careful to not disrupt these communication schemes, like the hierarchy and mutual interactions.

2.3.2 Domain understanding

The research study aims to report how engineers daily work, and identify collaborative requirements to improve their work. Therefore, it can be classified as a descriptive and improving study. In according to the existing literature, the following research methodologies exist:

Methodology	Primary objective	Primary data	Design
Survey	Descriptive	Quantitative	Fixed
Case study	Exploratory	Qualitative	Flexible
Experiment	Explanatory	Quantitative	Fixed
Action research	Improving	Qualitative	Flexible

Table 2.2: Overview of research methodology characteristics.

Case studies are by definition conducted in real world settings, and thus have a high degree of realism [15]. Of course, the high realism corresponds to a low level of variables control. In contrast, controlled experiments usually conducted in laboratory aims to fix all the parameters and change only one at time to measure qualitatively or quantitatively their effect. Conducting case studies usually researches get qualitative data. On the other side the using of a survey is very interesting because it can give quantitative data. As reported in [15], the research methodologies are depicted in Table 2.2.

2.3.3 Impact of a new system in the organization

One reason for which the introduction of new systems fails is due to the mismatch between information systems and organisational and social factors [14]. Another consideration is the impact of the technology introduction within the organisation. The impact should be assessed and evaluated before its introduction [14] as well as its acceptance.
Aspects like free rider problem and critical mass must be evaluated. The **free rider problem** concerns persons that participate, for instance in a meeting, but they do not give any contribution. On the other side, the users will join a system only if they have a benefit. The **critical mass** is the number of users that join the system in which the benefits of using the system became equal or greater then costs.

2.4 Collaborative Requirements Survey

FCA has multiple geographically distributed teams, therefore in order to get the collaborative requirements directly from stakeholders, we issued an electronic survey created with Google Forms. The survey questions were divided in the following main sections: *participants' experience, collaboration among engineers and data sharing, data centralization 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.
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 how the engineers work*** but at same time improving their work.

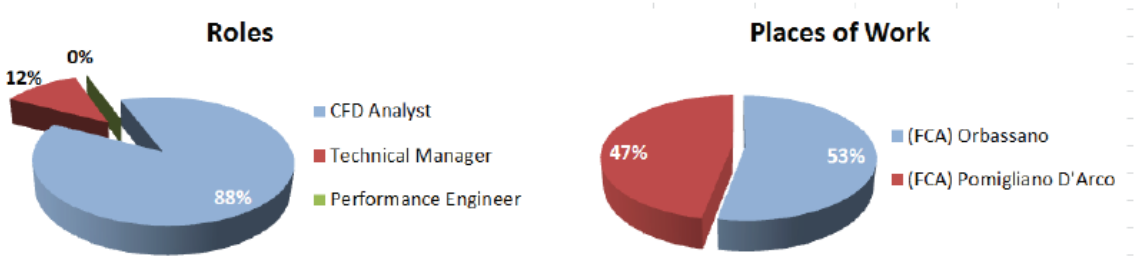


Figure 2.2: Participants' roles and their working place (*questions Q1 and Q2*).

The following section will analyse each requirement listed in Table 2.1.
The survey participants are CFD analysts and technical managers (Figure 2.2a). One survey participant is an academic who works daily with CFD analysts. In order to get the participants experience in the simulation field, the survey asks the years of experience and the number of performed simulations per year. More than 50% of participants perform at least one hundred simulations per year. This gives an idea about the total number of simulations per year, about one thousand simulations per year considering only the survey participants. Technical managers have a high experience in the CFD field but they perform less simulations per year because their tasks concern mainly the team organization.

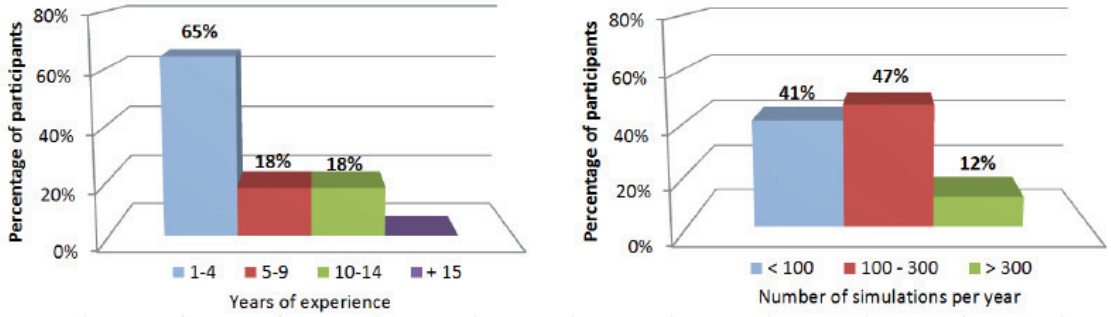


Figure 2.3: Participants' experience.

All analysts collaborate with at least one other engineer in the same office (Fig. 2.4a). More than half analysts collaborate with at least one engineer who works in another location (Fig. 2.4b). Analysts collaborate together sharing file geometries (CAD), simulations and documents (e.g. slides, spreadsheets).

2.5 Simulation data centralization

In order to support the collaboration among engineers (Fig. 2.1) they must access to centrally available simulation data (Req. 1, Table 2.1). Data centralisation means to collect data from different sources over time (i.e., from different simulators) and store them in an open format.

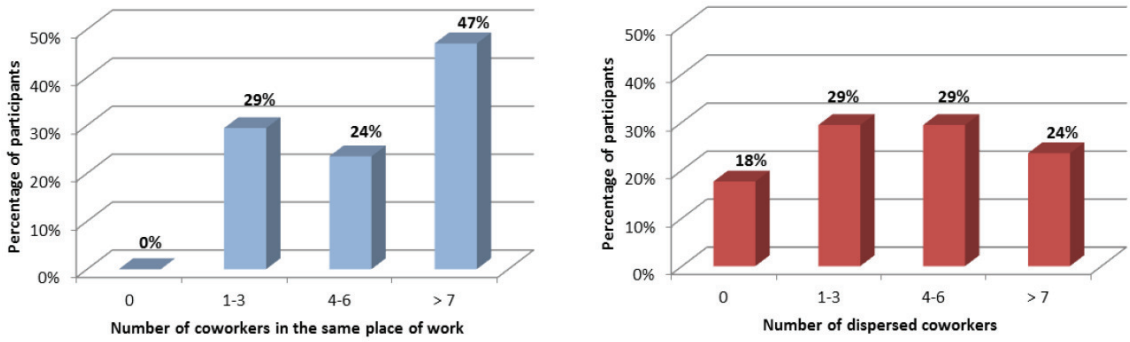


Figure 2.4: Co-located (*question Q8*) vs distributed workers (*question Q9*).

Data centralisation and the open format give an additional advantage: data and results can be aggregated in different ways, possibly in real time through an interactive user interface. Data aggregation means that analysts could compare simulations results performed on the same project or about multiple 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.

In according to Aberdeen Group's whitepaper "*Getting Product Right the First Time with CFD*" [11], in order to improve the company competitiveness, they should centralize simulations results. The aim is to centralise not only the results but all the related data such as the 3D geometries, simulation setup parameters and the documents allowing their easy retrieval. Based on the presented study, in order to centralise data and provide additional service over them, software designers should consider: the file size, the total number of performed simulations and eventually the closed file format.

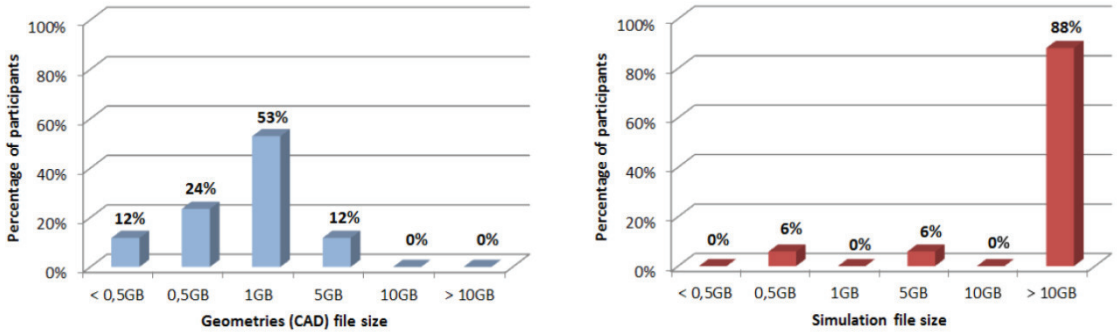


Figure 2.5: Geometries and simulations file size (*question Q10 and Q11*).

In the use case, both geometries and simulations are very large files. In according to the on-field observations and through the survey, we asked which are usually the geometries and the simulations file sizes. Figure 2.5a shows that the CAD file size is about one gigabyte in the fifty percent of answers. The file geometry can contain also the surface mesh and/or the volume mesh, explaining the differences of file size answers depicted in the chart in Fig. 2.5a. Instead, the simulation file size (Fig. 2.5b) is more than ten gigabyte in the 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. The large file size and the huge number of performed simulations exclude the use of a relational database to store data and provide additional services such as data search (see the following section) or results aggregation. In order to perform a simulation, its file must be stored on the 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.

2.6 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 search utility of the Windows OS cannot 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 the next time to find the data.

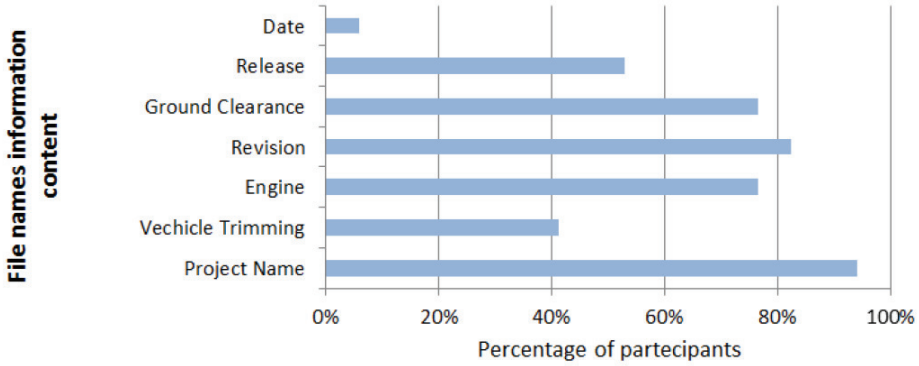


Figure 2.6: Information stored in the file names (*question Q20*).

As shown in Fig. 2.6, the main information inserted in the file name are: the project name, 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 remains within the simulation closed file and cannot be used for next retrievals.

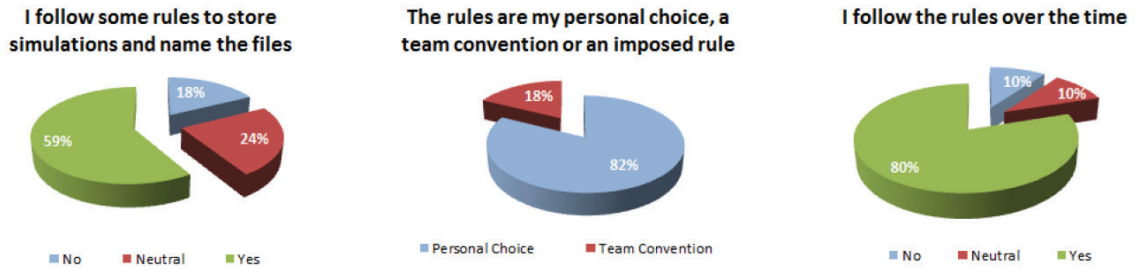


Figure 2.7: Rules followed store files (*questions Q19, Q21, Q22*).

More than half analysts follow roughly some rules to store files in the 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 its future retrieval. Nevertheless these rules are mostly a personal choice (82%), engineers add essentially the same information to the file names because the analysed engineering field is very specific. The limitation of this approach emerges when an engineer must search a simulation performed by other employees, mostly because he cannot use the 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 but are inside the closed file. This limits also the aggregation, based on specific keywords, of data at different levels, results comparison of multiple different simulations and generation of performance history charts about data of the entire simulation repository.

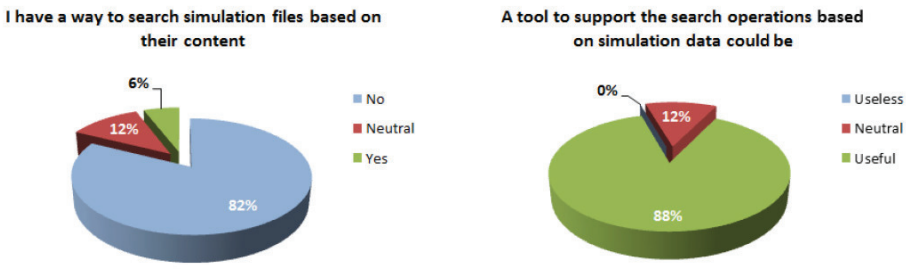


Figure 2.8: Search based on file content (*questions Q27 and Q28*).

To further explore the search of files, we asked to stakeholders whether they have a tool to search simulation based on the simulation file content such as the parameters (e.g. inlet velocity, which parts compose the simulated model). About 60% of participants said that they do not have such tool (Fig. 2.8a). In addition we asked whether they desire a tool to support the searching based on simulation content; about 75% of participants assert that they desire it (Fig. 2.8b).
An interesting consideration is about the technologies used to search files on the shared network folders. Analysts usually work on a specific project so they are confident with it and they try to remember where the files are stored. So, in order to find a file the most used approach are: try to remember where it is stored, navigate the file system seeing the file names and finally ask to a colleague. The most surprising (for the technical managers) survey result is that many analysts open the simulation file. The simulation file open requires a lot of seconds considering the heavy content. The less used techniques (in average) are the file history because the data are accessed from different workstations so the file history is not updated in the Unix and Windows find tools.

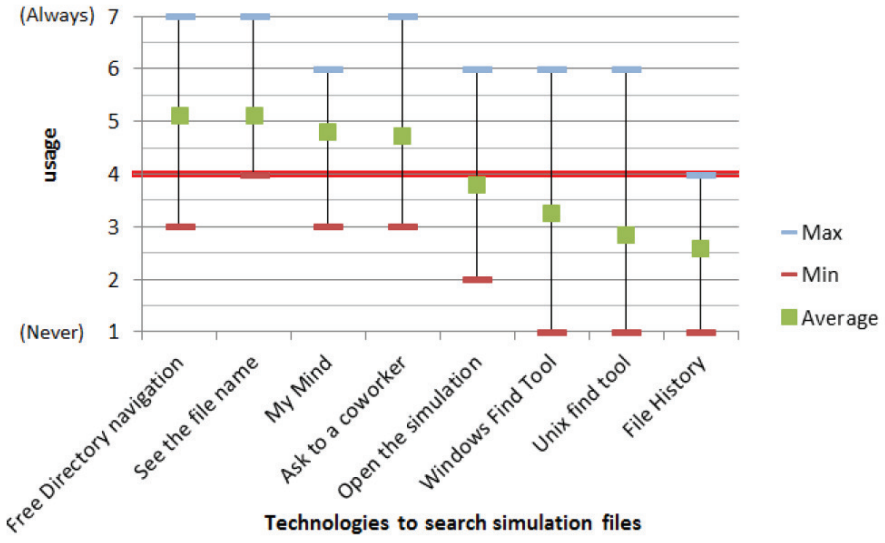


Figure 2.9: Technologies to search simulations (*question Q31*).

It is evident that an improvement can be done on the data search and retrieval. The most difficult and challenging part is that analysts use multiple simulators; each one stores files in different way, some of them in closed file format. The availability of a search tool enables the selection of multiple simulations based on the inserted criteria and the opportunity to extract statistics on the data.

2.7 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 cannot 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. 2.6). Obviously, the file name cannot host too many data, so other useful data are not annotated with simulations (e.g., comments and feedback). These data are very important both to improve the next retrieval but even more important to give a description of what the analyst did, his considerations and comments. To get this requirement through interviews and the survey, we asked to engineers whether they desire to link other data to the files. All analysts (100%) desire a system to link other information to the files, such as the file tagging.

2.8 Simulation data versioning

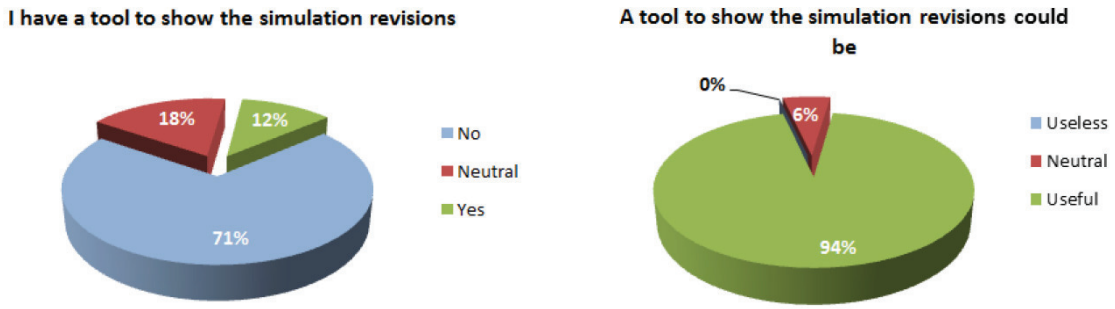


Figure 2.10: Version control (*question Q30*).

As reported in the Aberdeen Group market research [11], an action to improve the company competitiveness is to provide the version control over data. The survey aims to investigate further this need to understand its value for stakeholders. Here, version control means that the user 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, with the same simulation file, could be performed many simulations changing each time few model parameters (e.g., the inlet velocity). According to the 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 this kind of tool could be useful.

2.9 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, the 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.

2.10 Simulator independence

The previous requirements must work independently by specific used simulators to generate data. For instance, the tagging and search function 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 cannot be used to perform all simulation types. In the FCA 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 the model 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 to indicate which simulator software 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. 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 functions to evaluate during the choose of a CAD/CAE [17] otherwise simulation data are locked in the vendor software. Vendor Lock-In is a well-known Anti-Pattern [18] [19] [20]: the phenomenon that causes customer dependency on given vendor about a specific good or service [21] with high switching costs [22]. 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 by the vendor choices, so the architecture is product-dependent [23]. Data Lock-In occurs when the only way to access to the data is by using the Vendor Software because data are stored in a proprietary file format or they are stored on the vendor server and it does not provide an export functionality to an open format or a public customer API. The exporting and importing of geometric data are well-established functionalities for the CFD software, simply because they must commercially support the interaction with other CAD software. Instead, it is not the same for the entire simulation data such as the case setup, the simulation results etc. Data Lock-In is very common in Cloud Environments [24] and is an obstacle to cloud computing [25]. Vendors lock users in to make harder for them to leave the product because they cannot get their data; despite, as reported in literature, giving the opportunity for the customers to get their data increases their trust in the product [26].
A design solution useful to mitigate the Vendor Lock-In is to design the system with an additional layer called isolation layer [18].

2.11 Extensibility and modularity

The combination of modularity and extensibility [27, 28] system qualities allow final customers to compose a system with the only needed modules and to create their own modules to automatise specific tasks keeping them private to protect their know-how. In addition, an extensible and modular architecture allows the introduction of new functionalities tailored to the customer's needs. Extensibility is the ability of a software system to allow and accept significant extension of its capabilities without major rewriting of code [27] [28]. 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 [29].
Industries want 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*" [27]. The architecture must be modular enough to allow 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 also the software license. Two opposite needs must be taken into account: on one hand, industries want protect their know-how, on the other hand, the architecture must be adopted also in other contexts. Based on the presented use case, modularity, extensibility and Eclipse Public License (EPL3 license) are the right mix because the architecture, the framework and some other modules are open source but at same time industries can protect their know-how developing their own private and closed modules.

2.12 Social network

Centralisation of data, metadata and easy retrieval are required to enable the sharing of data among multiple teams. An interesting idea is to enable the discussion around simulation data using a kind of private social network (e.g., elgg). So, through the survey we investigated also this opportunity trying to understand what the users think about it. Actually, in the analysed context do not use a private social network as shown in Figure 2.11a and as stated by 95% of survey participants. Furthermore, more then 65% of participants actually are not involved in discussions about industrial topics about their work as shown in Figure 2.11b.

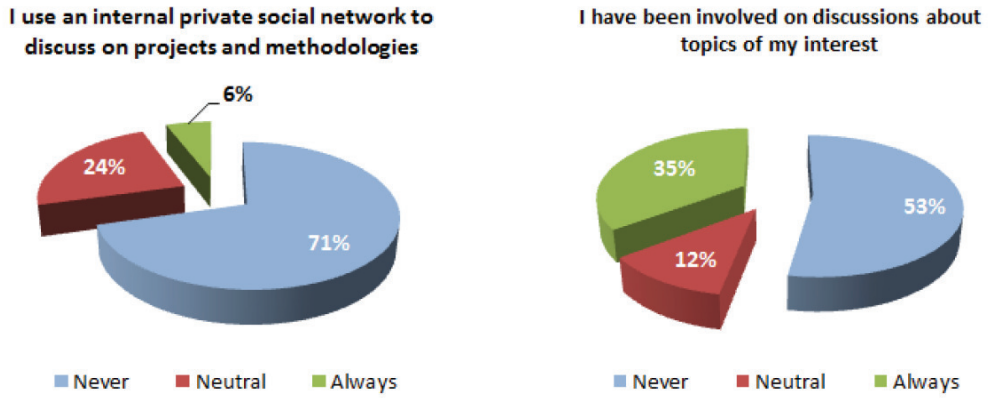


Figure 2.11: Actual use of private social networks (*Questions Q31 and Q32*).
The previous survey questions investigated the existence of a private social network within the company, a further step is to evaluate how the users are prone to use a private social network to discuss around simulation data, issues and interesting topics. Of course, nowadays users are widely exposed to the Social Network platforms (e.g., Facebook), so the term Social Network is a well-known. More than half of survey participants (65%) consider useful to involve other coworkers in discussions about interesting industrial topics. Moreover the 82% of participants consider this opportunity useful to improve their know-how (Figure 2.12b).

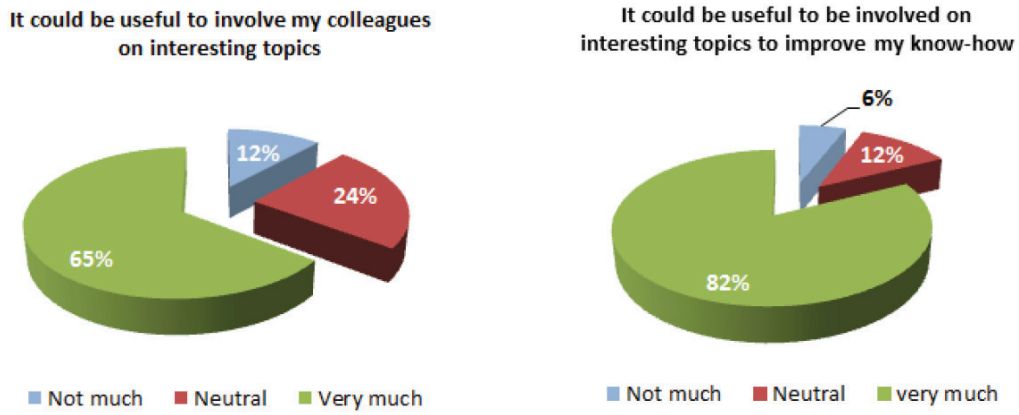


Figure 2.12: Introduction of social networks (*Questions Q33 and Q34*).

About the use of social networks to discuss on simulation data, through the survey and interviews we asked to the users what they think about their use. We get interesting considerations. A new employer declared that a social platform is useful to increase his know-how becoming immediately productive. Another useful comment is about the discussions traceability as the opportunity to find information about a previous faced issue. Therefore, traceability do not means to trace and use social as official place to make decisions or against employers (for it there are the official meeting memorandum) but to use it as a know-how repository.

2.13 Introduction to the Simulation Workflow

Briefly the Simulation Workflow is made by three steps (Figure 2.13): pre-processing, solving and post-processing phases. In the pre-processing phase engineers setup the simulation, in the solving phase engineers use the clusters to solve the simulation and in post-processing they collect, analyse and summarise results generating documents.

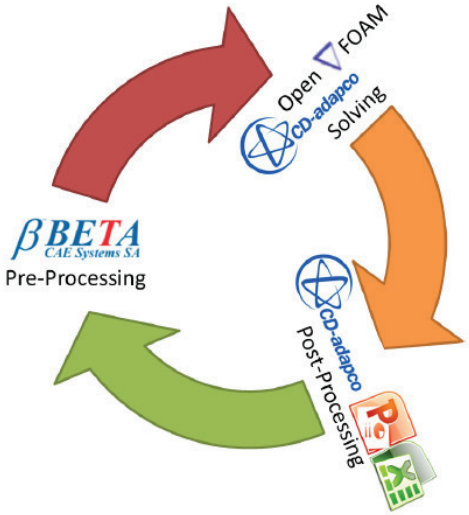


Figure 2.13: CFD Workflow: software used by analysts.

In order to perform simulations, engineers use different software, such as Computer-Aided Design and Simulator software. In addition, they use High Performance Computing Resources (e.g., Cluster) to run simulations. As known in large industries the Computational Fluid Dynamic workflow is not covered by only one software especially for large industries that design complex products, but multiple software are required. For instance, industries use at least a CAD software, a simulator and a post-processor software. In addition, large industries simulate different parts of the vehicle so they use multiple simulators, each one for the specific simulation type.

Typically CFD analysts start from the geometry that describes the vehicle shapes. This geometry is not suitable to be used in the simulators software. Therefore, it is imported in another CAE software called pre-processor. In the analysed use case, engineers use ANSA developed by BETA CAE Systems. Another motivation to use a pre-processor is the need to clean up the geometry and create both the surface and volume mesh indispensable to simulate the vehicle physical properties. The volume mesh is then imported into the simulator software (e.g. OpenFoam or STAR-CCM+). The Fig. 3.1 shows the CFD Workflow with software tools used at each workflow step. At the end of the tree steps, the simulation results (e.g. contour-plots and tables) are used to build the documents.

Engineers use multiple CFD software in the solving phase. Therefore, through the survey we asked to the analysts which software they use. The answers have been depicted in Figure 2.14. All analysts use CD-Adapco STAR-CCM as main simulator software and more than half analysts use OpenFoam. Of course the described workflow is usually performed within a large industry like Fiat Chrysler Automobiles; for small enterprises the workflow essentially is the same but the number of software can be reduced in number as well as the product complexity. Of course, for simple products it is possible to use one integrated software environment to design the product, simulate it on a single computer and process the results.

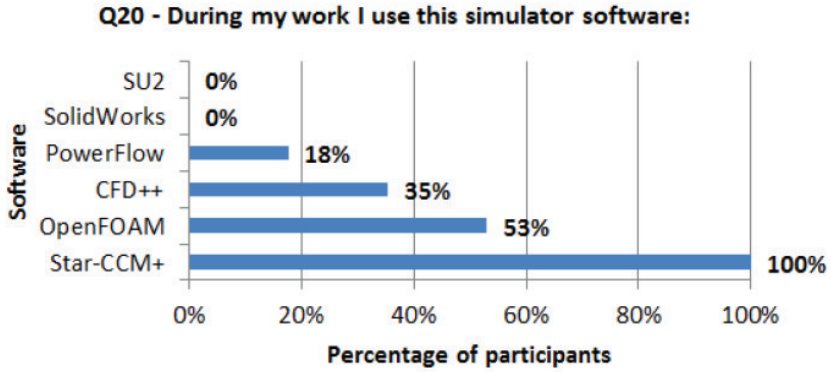


Figure 2.14: Simulation software used to simulate vehicles.

Another difference between small and large enterprises are the number of performed simulations. In the automotive sector as well as in the aerospace each engineer performs many simulations for year. As reported in Figure 2.15, more than half of survey participants in FCA perform at least one hundred of simulations per year. This is very normal because usually engineers perform multiple simulations on the same product changing only some parameters, such as the inlet velocity (the velocity of the air fluid in the wind tunnel) or the position of a component. The repetition of these operations (e.g., simulation running, generation of documentation, etc.) executed mainly manually

raise the need to have an automatic tool to perform them, and Figure 3.6 shows on top the main services to provide within the CFD Simulation Workflow.

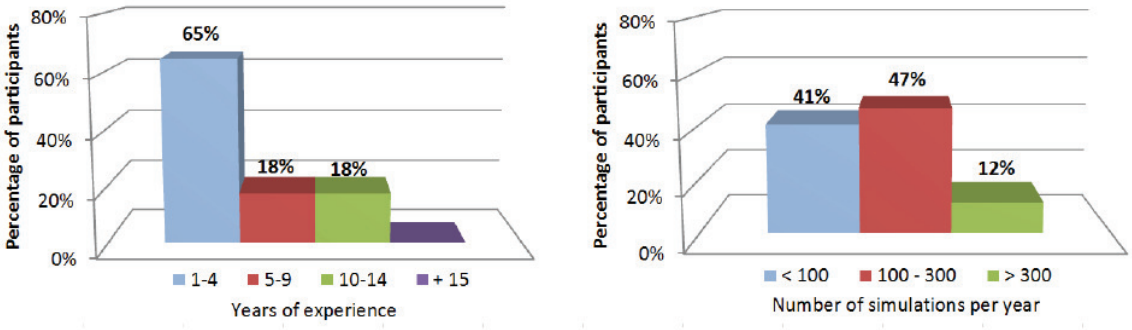


Figure 2.15: Participants' experience.

2.14 Metrics used to evaluate an engineering software

It is very interesting to understand which criteria the engineers use to evaluate a CFD solver. This gives a rough idea on which should be the features important for end-users. Therefore, we asked to CFD analysts which are the criteria used to evaluate and choose a CFD solver. Engineers judge a CFD solver considering the commercial support, its reliability and the validation over many common industrial contexts. Another important criterion is the software product dissemination especially considering solutions used by the competitors.

CFD software are used in many industrial sectors such as automotive, aerospace, high tech, oil/gas and so on. The same CFD software usually is generic enough to be used in multiple sectors. So, end-users (40% of respondents) seek for standard templates and wizards for their specific configurations and simulation setup. In this way, instead to start a simulation from scratch every time, the wizards guide end-users to setup the simulation giving only the basic information. Wizards are useful also for less experienced users reducing the training costs. Moreover, both templates and wizards guarantee that everyone in the team works in the same manner.

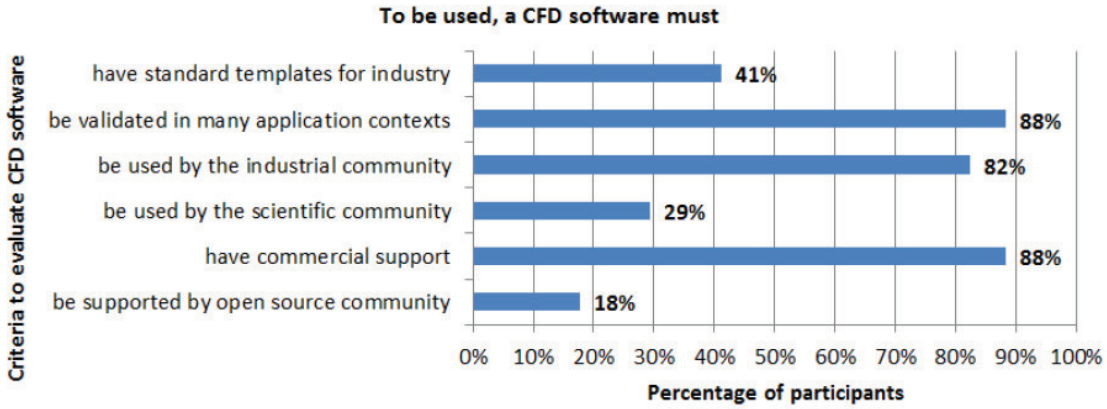


Figure 2.16: Criteria used to evaluate a CFD solver (*Question* Q32).

The "*Getting Product Design Right the First Time with CFD*" [11] market research shows the same result and advises the use of templates and wizards as a way to increase the company competitiveness. In 2010 the 27% of Best-in-Class companies report plans to implement these facility. In the cited market research [11], Best-in-Class are the industries with a high performance index based on likely to release product on time, reduction of development time, meeting of quality and cost targets.

In addition, it is interesting to understand the criteria used by analysts to evaluate an engineer tool in general (not only the CFD solvers as in the previous question). The question outcome is shown in Fig. 2.16. Obviously, the most of analysts consider the solver accuracy the most important software feature. The accuracy value emerges also in the market research "*Getting Product Design Right the First Time with CFD*" [11], the Best-in-Class companies place high value on CFD simulation accuracy so that the 58% of responders aim to have simulations as accurate as possible and they are not willing to sacrifice accuracy. But more accuracy requires more running and solving time. So, in order to reduce the running time a way could be the model simplification but this generates less accurate results.

Other criteria to evaluate a software are the *Easy to use* and the *GUI*. Through the survey, half of analysts think that the system usability is important. In particular, the 76% of respondents consider the software *easy to use* as a criteria to choose or judge an engineering software. This shows an increase importance of the software usability than in the past. In contrast, the Aberdeen Group market research (2010) reports that only the 23% of their responders considered the easy to use of a CFD software important. Therefore, the usability is becoming important also in the engineering field where the end-users are usually very experts, and they daily use software with many options and are prone to accepts also complicated software to perform their tasks.

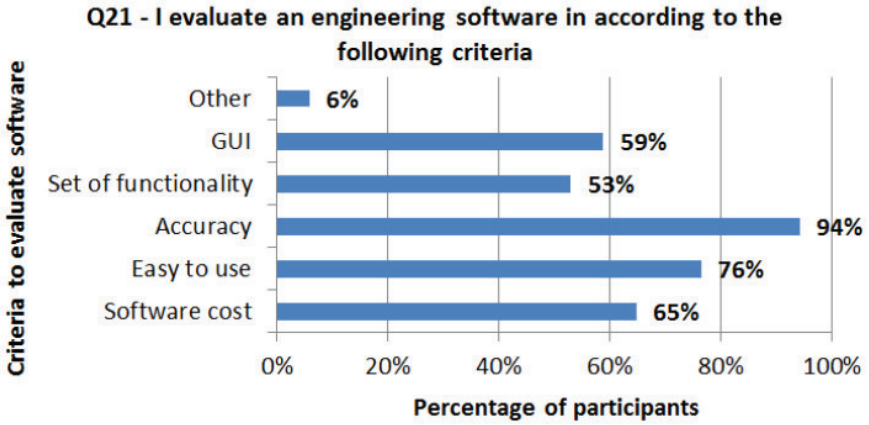


Figure 2.17: Criteria used to evaluate an engineering software (*Question* Q33).

The last criteria is the software costs but obviously this not affect directly the analysts because usually the technical manager are involved in the software purchase.

An interesting survey outcome is that the set of software functionality seems to be less relevant of the accuracy. Finally, another respondent has proposed further criteria to evaluate an engineering software: the customer support and whether the software is used by the competitors.

2.15 Requirements Overview

Requirements have been divided in two main categories [3]: Functional and Non-Functional requirements. Functional requirements define the system functions so they concern mainly the system behavior and what kind of functionalities the system provides. Non-Functional requirements are criteria used to evaluate the system often from a quality point of view, therefore, they are often defined as Quality Requirements.

The table 2.3 lists briefly the main functional requirements gathered within the use case, instead the table 2.4 lists the Non-Functional requirements.

The table 2.4 of Non-Functional requirements, contains also additional constraints to follow that come to light during the discussions with engineers and stakeholders. Engineers use different simulator software so the platform must integrate multiple simulators (Req. 14) and multiple users that work with the platform in according to the available resources (e.g., computing resources).

	Requirement
Req. 9	*Repository Tool, Simulation tagging and search*
Req. 10	*Run Simulation Wizard*
Req. 11	*Monitor Simulation Convergence*
Req. 12	*Results Comparison & Statistics generation*
Req. 13	*Automatic Document Generation*

Table 2.3: Stakeholders' key engineering Functional Requirements.

The simulator must be replaceable because usually enterprises especially in the engineering sector change the used software to get more competitive products. Some simulators are open source so from technical point of view is easy to integrate them because one could change it, but one constraint is to avoid the change of the available source code (Req. 18). The motivation is simple, when one change the source code of a software than it is difficult to be updated with the latest software release because each time a merge operation is required. The integration of this simulators is feasible because engineering software run “headless" that means without user interface interacting with it directly through the command line.

	Requirement
Req. 14	*Support multiple simulators*
Req. 15	*Simulator selection*
Req. 16	*Concurrent simulators use*
Req. 17	*Headless simulators integration*
Req. 18	*Do not change simulators source code*
Req. 19	*Support real-time and batch interactions*
Req. 20	*Avoid Vendor Lock-In*
Req. 21	*Extensibility and modularity*

Table 2.4: Stakeholders' key engineering Non-Functional Requirements.

2.16 Functional Requirements

This section describes briefly the Functional requirements gathered in the Fiat Chrysler Automobiles use case.

In according to the Simulation Workflow, the first step is the selection of an existing simulation, and, in order to do this, Floasys has the Repository Tool to navigate the simulation repository and select the target simulation. It is important to remember that the creation of a simulation is made within a proper Simulation Software. After the selection of the simulation, usually engineers use the command line to run the simulations on the industrial High Performance resources. A need is to provide a wizard to run the simulations in an easy way. Therefore, Figure 2.18 shows above the pre-processing phase, the Run simulation and the post-processing phase. The provided wizard is not only a direct replacement of the command line but provides additional settings that depend on the simulation to run.

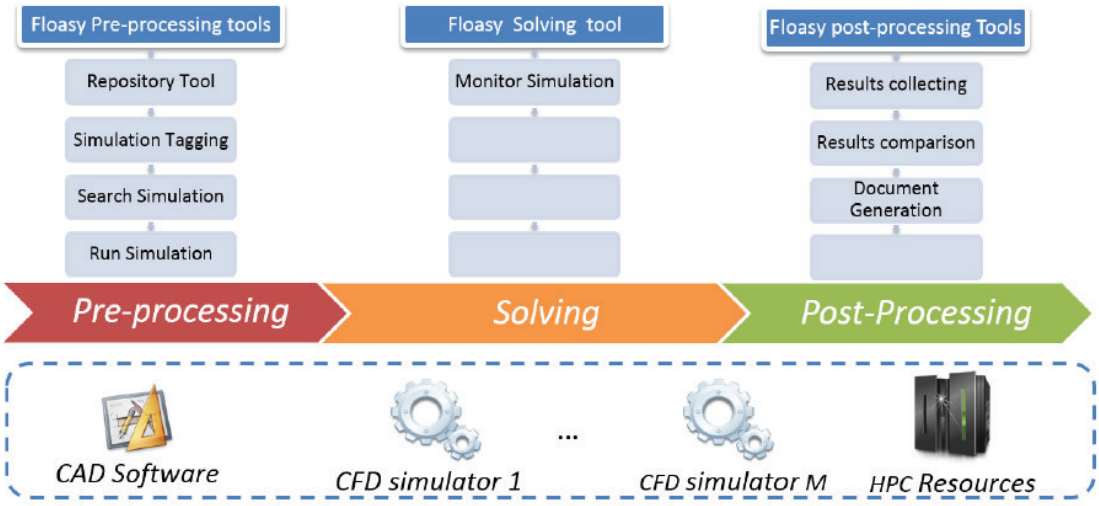


Figure 2.18: CFD Workflow and Floasys Tools

CFD simulations runs on HPC resources and they usually run for many hours, sometimes also an entire day. In addition, the tuning of simulations is a non-trivial task due the high number of parameters and specifically the geometry quality. For instance, the geometry quality is very important because if an engineer is simulating the external vehicle aerodynamic that has some invisible holes especially between two surfaces, the simulator tries to solve the fluid equations in that space and could diverge.

Therefore, as observed in this thesis use case and as noticed also in literature [30], it is very important and useful to have a tool that connects to the HPC cluster to monitor the running simulations convergences. Commercial simulator products already have a GUI made by real-time charts to monitor the simulation convergence. Other products, especially open source, that runs only from the Command Line Interface, do not have a monitor GUI but they write convergence data in appropriate files.

Finally, the last group of functionalities are: the collection of results, the results comparison and the automatic document generation.

2.17 Non-Functional Requirements

This section introduces and describes the Non-Functional requirements.

2.17.1 Support multiple simulators

CFD analysts perform different types of simulations on the same vehicle product (i.e. external aerodynamics, aeroacoustics, air conditioning and engine thermal analysis). Therefore, they use different simulator software, each one suitable and validated internally for its own application. Briefly, in the FCA use case, at time of writing, the most used software are (Fig. 2.14): STAR-CCM+ (commercial product) and OpenFoam (open source). In order to reduce costs or to have better features, industry can decide to change CFD simulator in future. So, the architecture must support and integrate multiple simulators software with the opportunity to remove each one and introduce other implementations.

2.17.2 Simulator selection

The support and the integration of multiple simulators (Req. 6) leads to the issue of selecting the right simulator to perform a specific simulation type. These selection can be done automatically by the system or manually by the user. Based on the performed tasks or the simulation type, the system must automatically select or recommend the appropriate available simulator software. Obviously, engineers must be aware about the selected simulator, so the system must show or give the easy access to feature (at least the name) of the used CFD software.

2.17.3 Concurrent simulators use

Analysts use many instances of the same simulator software opening multiple files and running many solving jobs. The platform must support two cases: the support of multiple different simulators concurrently and multiple instances for the same simulator.
For instance, the Convergence Monitoring Tool (Monitoring Tool) shows the charts about the convergence of multiple running simulations. So, the tool is able to access and show the chart basing on the data generated by multiple running simulations.
Another example is the Automatic Documentation Generation Tool that extracts and collects data from different simulations and merge them in spreadsheets and slides. So, it need to manage multiple simulations file and consequently multiple simulators, also of different types.

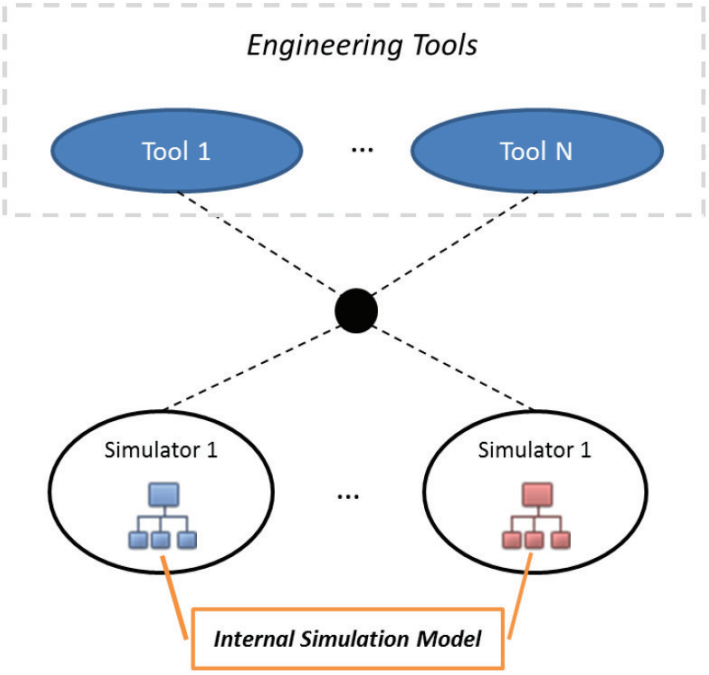


Figure 2.19: Simulation Model Mapping.

Engineers use many instances of the same simulator software opening multiple files. For instance Floasys Automatic Document Generation Tool extracts and collects tabular data from different simulations, and merges them in Excel and Power Point documents. Data are shown in the front-end workbench GUI. Documentation tool needs to manage multiple simulation files and multiple instances of the same simulator software.
The architecture must support multiple CFD simulators used concurrently. Same tools must access to multiple simulators concurrently. For instance the monitoring tool shows charts about the convergence of the running simulations. Engineers monitors the convergence of running simulations from different CFD software.

2.17.4 Headless simulators integration

CFD simulators can run "*headless*" without the Graphical User Interface (GUI). This is a built-in simulators feature because they must run on High Performance Computing (HPC) resources such as computer clusters. For instance, OpenFoam is an open source simulator made by a set of command line tools and text-based input/output files. OpenFoam does not have the GUI, so the aim of many projects both open and commercial is to design and provide a GUI [31] for OpenFoam.
Another example is the simulator CD-adapco STAR-CCM+ , a commercial software to perform CFD simulations. It can run either with GUI or without it. In addition, it has a Java-based scripting language to provide additional custom features called Macro.
Simulators often do not have public APIs to allow other applications to interact with them. Therefore, the only way for another application to exchange data with the simulator is to wrap it. Idea is to think about the simulator as a black box with its input and output. Fortunately, in the CFD field the software have been designed to run also on the command line. Third commercial products rely on this assumption to work and integrate simulators features. For instance, Esteco

modeFrontier, a commercial Design Of Experiments (DOE) software for optimization, has a graphic workflow made by nodes. Among the different node types, modeFrontier has a specific node to integrate external software trough calling to the executable software from command line. It relies on this mechanism providing input and getting the output trough files.

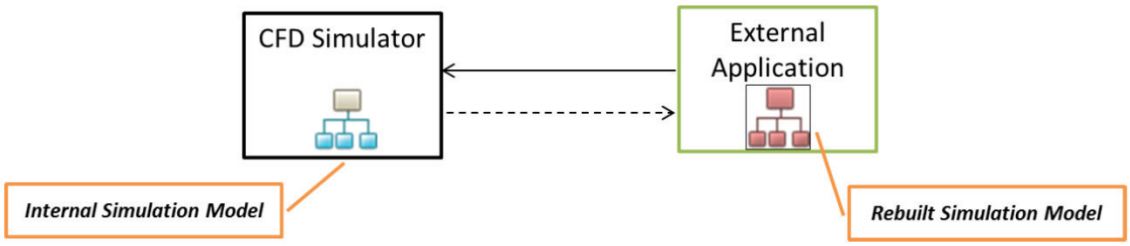


Figure 2.20: Simulator black-box interaction.

The integration among simulators and external applications could be a bit difficult nevertheless they run from the Command Line Interface (CLI). Not all functionalities are available through the CLI and what you can perform significantly varies. For instance, simulators sometimes give a less control on the simulation model changes. Therefore, the interaction with the simulators is more restrictive because they do not have public APIs and do not support sufficient interactions through the command line. Another limitations is the closed file format: it is impossible to access to simulation data without the vendor software and binary files are meaningless to an external application. Generated simulation files are binary files limiting the interactions with external software.

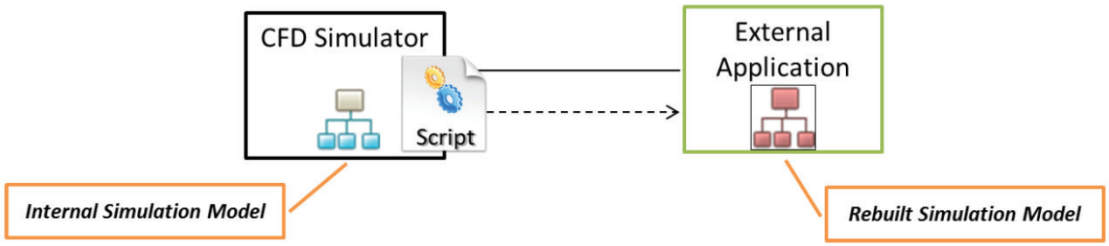


Figure 2.21: Interaction with a simulator through a macro

In these cases, the interaction among simulators and external applications are limited so some workaround are needed. One possible workaround concerns the exploitation of the provided scripting language used in combination with command line options.

Do not change simulators source code

The aim is to integrate both open source and closed simulators. For open source software, we do not change the source code to be always up-to-date with the original software avoiding continuous changes of source code at each release or in the worst case to remain with and old simulator version because the upgrade is too expensive in term of changes. In addition, the overall platform must be

designed in a way that the upgrade of the simulator cannot heavy impact on the overall platform and tools. So, the aim is to reduce the coupling between the simulators and the platform.

Support real-time and batch interactions

Architecture must handle both real-time and batch interactions. Batch interactions are mainly the Job submissions to solve simulations. At same time, engineers monitors the running simulation using the Monitoring Tool that shows real-time data.

Avoid Vendor Lock-In

This requirement is very important because CFD analysts use proprietary software. Vendor Lock-In is a well-known Anti-Pattern [18] [19] [20]: the phenomenon that causes customer dependency on given vendor about a specific good or service [21] with high switching costs [22]. 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 the vendor choices. So, the system architecture is product-dependent [23] because it wraps some or all the vendor software functionalities and there is not clear distinction between them. Data Lock-In occurs when the only way to access to the data is by using the Vendor Software because data are stored in a proprietary file format or they are stored on the vendor server and there is not an export functionality to an open format or a public customer API. CFD analysts use proprietary software that store data in closed file format. The exporting and importing of geometric data are well-established functionality for the CFD software, simply because they must commercially support the interaction with other CAD software. Instead, it is not the same for the entire simulation data such as the case setup, the simulation results etc. Data Lock-In is very common in Cloud Environments [24] and is an obstacle to cloud computing [25]. Vendors lock users in to make harder for them to leave the product because they cannot get their data; despite, as reported in literature, giving the opportunity for the customers to get their data increase their trust in the product [26].

Extensibility & Modularity

Architecture should be extensible to allow and simplify the introduction of new functionalities. Extensibility is the ability of a software system to allow and accept significant extension of its capabilities without major rewriting of code [27] [28]. Extensibility is an important Non-Functional requirement during the system development because systems are constantly under change and, adopting the Continuous Delivery [29], they need to wrap incrementally vendor software functionalities. Extensibility is a quality architecture attribute useful during the development and especially in future when more and more simulators' features will be integrated in architecture. Industries want deploy the same system with different features. The architecture must be modular enough to allow both the adding of new simulators and the removing of existent simulators. The modularity requirement has an interesting advantage for the architecture design: the application or custom engineering tools and simulators are loosely coupled. 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*" [27].
Industries have command line macros that are designed, developed and tested over the years that must be integrated in the architecture. But at same time the system architecture should be deployed in different contexts, so it is very important to identify exactly which system modules contain the industrial know-how and separate them.

2.18 Related Works

This section introduces the process called New Product Development Process (NDP) to conceive a new product and bring it on the market. Within this process we will focus on the engineering activities, especially the simulation workflow to design vehicle products.

2.18.1 New Product Development Process

Manufacturers aim is to bring products on market quickly within the budget and performance constraints [11]. The New Product Development Process (PDP) describes the process adopted to design, develop and bring products on the market [32] and involves a continuous information exchange among many tasks. As shown in Figure 2.22 it is made by the following steps: concept, design, prototyping, manufacturing.

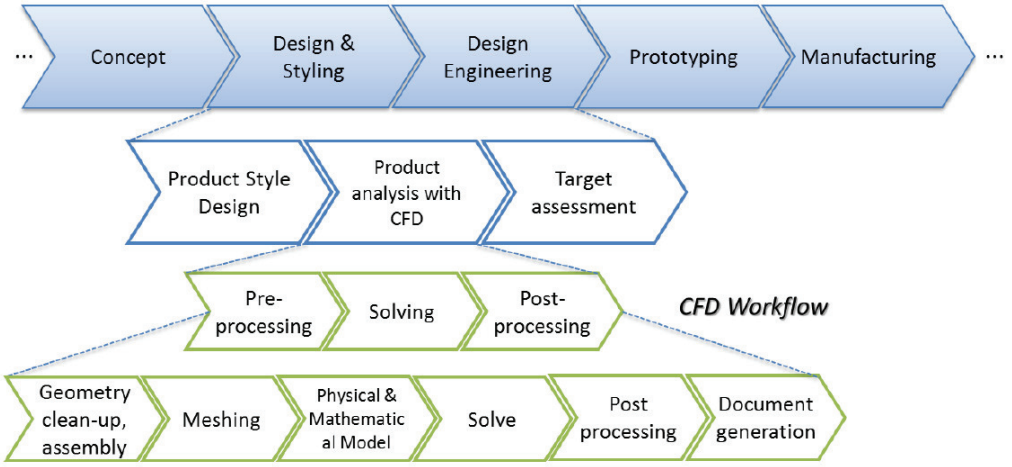


Figure 2.22: Product Development Process (PDP) and CFD Workflow.

Nowadays, in a worldwide and highly competitive market, enterprises face short time to market (TTM), continuous innovations, global collaborations and complex risk management [33]. Enterprises are global and have different organisations around the world. Therefore, intellectual assets, data and know-how must be accessible to anyone within the enterprise and sometimes also outside the enterprise. In order to design products, enterprises collaborate with other external enterprises. For instance, along the supply chain, enterprises get the raw materials to make products or they externalizes tasks performed by external consultants. In this context the use of software systems is very important. From historical point of view two main systems have been evolved separately: the Product Lifecycle Management systems and Product Data Management, although today the aim is to integrate them together to have a unique system.

The **Product Lifecycle Management** (PLM) is the process to manage the product life cycle from its inception, through the design and manufacturing, to customer service until the product disposal. PLM gained even more interest in the last years as a business approach to integrate people, processes, business systems and information to manage the product life cycle management.
As stated in [33] the PLM has two roots:

- enterprise management;
- management of product information.

The **enterprise management** concerns the material resource planning (MRP), enterprise resource planning (ERP), customer relationship management (CRM) and supply chain management (SCM). It is evident that across the years different management systems have been created so it's essential to integrate them through a PLM system.

The **management of product information** concerns the product and its related information, know-how and so on. Therefore, the management of product information refers essentially to the **Product Data Management** systems. Enterprises design their products using different systems in different phases. Generically they use authoring tools called authorware to create new content. Authoring tools are not programming languages and do not require programming skills but they rely on the graphical user interface (GUI) to create content.

Industries design products through Computer-Aided Design (CAD) systems and simulate them through Computer-Aided Engineering (CAE). These Design, Manufacturing and Engineering (CAD, CAM and CAE) software are daily used and generate a lot of data. Therefore, during 1980s, Product Data Management (PDM) systems appeared to control and manage the product information created using engineering authoring tools [34]. PDM systems were born in the engineering field and they were used mainly to store information like geometric models, Bill of materials (BOM) and FEM models. Nevertheless, very similar needs arose also within non-engineering area, such as sales, marketing and supply chain management, PDM systems failed to address these similar needs, mainly because they were designed for engineers to manage engineering data. Later, in the 1990s, with the introduction of Internet and WWW, vendors adopted these new technologies to implement PDM systems. In this way, PDM became web-based and took advantage of *universal*, *inexpensive* and *ubiquitous* nature of Internet to provide their services throughout the enterprises. Nevertheless, PDM were web-enabled, their use were still about the engineering field and they essentially managed engineering documents. In the past, product were designed using pencil and papers; instead, nowadays product are mainly designed using CAD systems to create the geometric models. Pencil and papers are not completely replaced in the concept and more creative phases. Of course, during the years the volume of geometric models is very huge and can get out of control. Therefore, products data must be managed through a **Product Data Management** (PDM) that itself must be integrated with the CAD and other software.

PDM mainly gained their success in the engineering field to manage geometric data, bill of materials (BOM) and finite element analysis models. Product Data Management systems are well-integrated with CAD software, but it is not the same for simulation software that are actually almost isolated islands. A goal of this thesis is to explore how integrate simulator software especially because a lack of interoperability exist; so the idea is to reduce the gap between the simulator software and other software like Product Data Management systems.

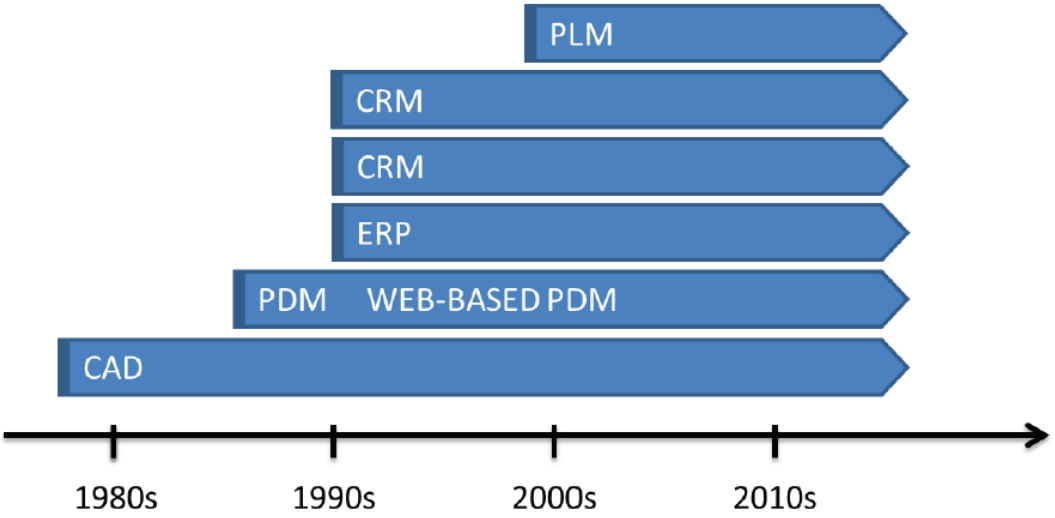


Figure 2.23: Product Lifecycle Management Evolution

From historical point of view, the evolution of PLM and PDM is tracked roughly in the Figure 2.23. It is evident the concurrent evolution of PDM to manage engineering data and other solutions to manage specific aspects of the enterprises, such as ERP, CRM, SCM systems that were integrated together within PLM systems. In the paper [34], the authors identified both internal and external enterprises forces that led the PLM evolution and adoption. The internal forces are: the need for innovation, customer intimacy and operations excellence. The external forces are: globalisation, product complexity, shrinkage in product lifecycle, push into supply chain and environmental issues. An example of Product Lifecycle Management is Aras an open source product, but many other product exist.

2.18.2 Engineering Use Case

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 R&D activities to improve their know-how and to be ready to provide innovative design solutions. Platform area is a transversal area that has a global view on the product and leads the product development from the concept to the final product to sell as well as customers feedback and satisfaction.

The CFD unit is the engineering functional area with highly skilled engineers called CFD analysts who perform simulations to analyse the aerodynamic and aerothermal/automotive product behavior. This functional area is this thesis use case. In a big industry, many CFD unit exists that collaborate together (Fig. 2.1). The collaboration happens among dispersed CFD units and among CFD units and the other industry teams such as the product style designers and the performance engineers. In order to design an automotive product many engineers collaborate together. Specifically, in the automotive sector and for the aerodynamics/aerothermal analyses, CFD analysts, automotive designers, and performance engineers collaborate together. Each CFD unit has a technical manager who is responsible for the internal team organization and needs.

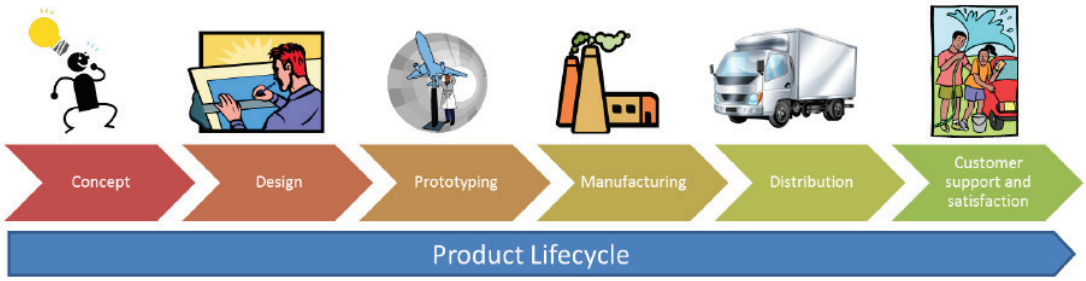


Figure 2.24: Product Lifecycle

Briefly, style designers design the product style concept that is used by other functional area. CFD analysts perform CFD simulations using the 3D vehicle model and finally the performance engineers are responsible for the meeting of performance targets. The thesis use case focuses on the CFD functional area and its relationships within the Product Development Process (PDP). CFD is a numerical computer simulation able to solve and analyse 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 behaviour*" [11], product optimization in according to the performance goals, the simulation of extreme environmental conditions (i.e. low or high temperatures) and a cost reduction due less number of 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 analyse the product performances (e.g. aerodynamics, aerothermal, aeroacoustic, air conditioning and cabin climatisation) with CFD simulators. CFD analysts perform simulations and report data in documents. In order to meet 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. 2.25): 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.

2.18.3 CFD Simulation Workflow

Figure 3.13 shows the classical simulation workflow mainly made by three steps:

- *Pre-Processing*: usually concerns the creation of the model (e.g. geometric model) and the setup of the simulation (e.g. simulation parameters);
- *Solving*: the simulation runs on HPC resources;
- *Post-Processing* may include the calculation of additional quantities, the plotting of results, the visualization of simulations pictures, the analysis of results and the creation of documents.

The Simulation Workflow is a step-wise and iterative process. The same product has always multiple variants and during the design process multiple revisions are created. So, for the same product the Simulation Workflow is iterated many times. The final outcomes are documents that describe the product performances. These documents are used as a support to exchange results and to collaborate among analysts and performance engineers.

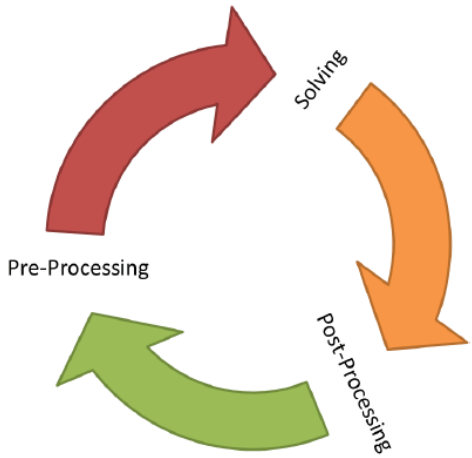


Figure 2.25: CFD Simulation Workflow.

One aim of this work is to improve the data management especially when many simulations exist and engineers perform a lot of simulations per year.

The Pre-Processing (Fig. 2.26) is essentially done manually. The most of time is spent cutting and cleaning the vehicle geometry using a CAD/CAE software (e.g. BETA CAE Systems ANSA). Also the Post-Processing requires a lot of manual work especially to create the documents compliant to the industrial templates. Many accomplished tasks both in the pre and post processing are repetitive and error prone tasks. One aim of this work is to automatise many of tasks.

The Pre-Processing and the Solving phases can be accomplished by one engineer who creates the model, setups and runs the simulation. Nowadays, products are becoming very complex integrating many components, so industries have a specific design team who is responsible to design the product concept and its style. So, more often the geometric model already exists as CAD file. The CFD analyst task is to cut the geometric shapes to remove the unnecessary part for the specific

simulation. This has also another goal: to simplify as much as possible the geometric model and reduce the later simulation time. For instance, to simulate the external vehicle aerodynamic, all the internal vehicle components are removed to have only the vehicle surface. Another important task is the geometry clean-up. For instance, some space in the front of vehicle that is negligible for the mere visualization can be destructive for a simulation solver that tries to simulate the fluid within the hole.

Nevertheless, here the main use case is about the CFD, the Simulation Workflow follows roughly the same steps also for other type of simulations. In addition, it is usual to use separate software applications for each step [35].

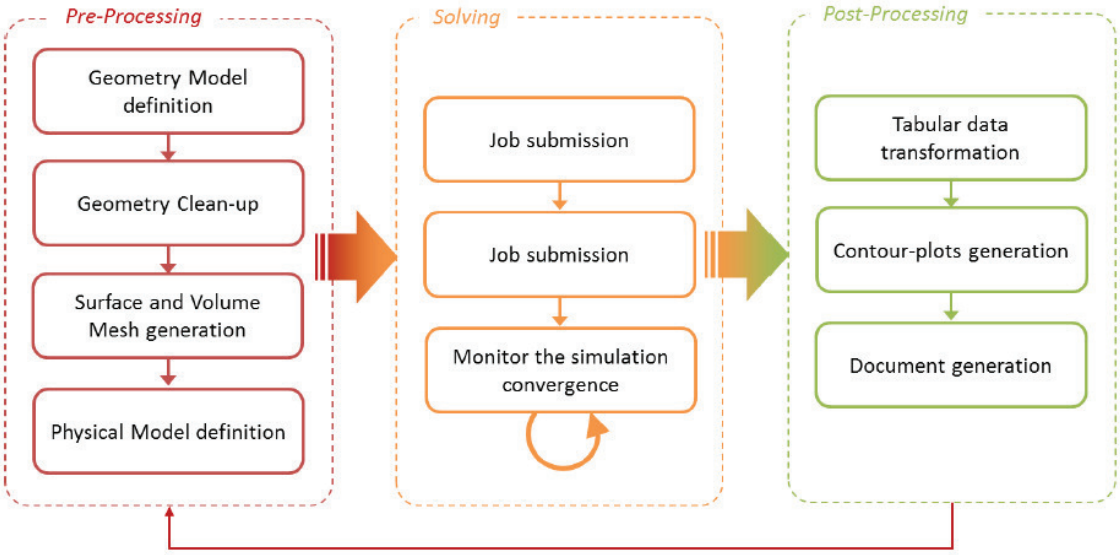


Figure 2.26: CFD Simulation Workflow detailed phases.

2.18.4 Existing platforms for CFD simulations

Many Web-based platforms have been created over the years to support Computational Fluid Dynamics. The "*e-Science Aerospace Integrated Research System*" (e-AIRS) [16] is an educational Web portal developed in Korea to help students to understand the aerodynamic simulation process [36]. EDISON CFD [37] is the e-AIRS improvement in terms of stability, faster data response time and waiting time [38, 39]. 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 [16]. 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 mesh 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 cannot 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 [40]. It allows the "*dispatching and controlling of long-running simulations*" [40]. 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. The 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 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.

2.19 Future works

This section describes the future works that can be further investigated starting from the previous described engineering requirements. These future works have been systematically gathered through the interviews with the stakeholders.

Simulation Workflow is usually made by three main steps (Figure 2.25): pre-processing, solving and post-processing. In the pre-processing phase, engineers setup simulation (e.g., geometries, mathematical model) to be solved in the next Solving phase using HPC resources. Finally, in the post-processing phase, the data in different format are collected and analysed to understand the product behavior.

The Simulation Workflow is iterative. Actually, each iteration takes hours to be completed (sometimes 24 hours) and requires the engineers manual actions. Therefore, analysts perform few workflow iterations about the same vehicle product. At each iteration, the vehicle geometry is the same but placed in different positions. For instance, analysts simulate the same vehicle with different ground clearance also called ride height (the amount of space between the base of an automobile tire and the underside of chassis) or, for example, with different spoiler positions. In addition, at each iteration also the simulation parameters can be changed. For instance, the inlet velocity usually is one parameter to change (i.e., common values are 20 km/h, 30 km/h, 50 km/h). For aerodynamic analysis, considering all the vehicle configurations, the number of simulations to perform is very high. Actually the simulation setup is performed manually. For instance, engineers manually move the spoiler along the vehicle solving a simulation for each discrete spoiler position. One interesting idea is to use the parametric geometry morphing supported by many CAD software (e.g., BETACAE ANSA) to automatically change the spoiler position and perform all simulations without the engineers actions.

The geometric and simulation parameter is only one of parameters to control. In addition, analysts perform different types of simulations. For instance, in the automotive context, engineers perform

aerodynamic, aeroacoustic, underhood cooling, internal air conditioning and other aerothermal simulations. These use of different types of simulation is called **Multi-Disciplinary simulations**. To perform these analysis, engineers use different simulator software because each one is suitable, validated and adapt to simulate a specific physical phenomena. Another example is **Multi-Scale simulations** where the analysed system is simulated at different scales with different simulators. Finally, another important aspect is the **Muti-objective optimisation**.

Therefore, analysts firstly demand for the integration of existing tools to get features not covered by one single product. The most well-known integration requirement in literature is the integration between CAD and CAE/CFD software, but also other integration requirements are becoming even more important. For instance, the integration between a Design Of Experiment (DOE) system and simulators to change the input parameters or the opportunity to use more than one software to perform the simulations.

Industries aim to integrate existing simulators, repositories as well as hardware and infrastructures together to have a unique platform to manage multiple simulation workflow automatically. Idea is to integrate existing industrial facilities (e.g., repositories, simulators, DOE, HPC resources, etc.) and automatically manage multiple simulation workflow through a unique, Multi-Disciplinary, Muti-objective and easy to use platform. Product Lifecycle Management (PLM) and Product Data Management (PDM) systems aim to collect data and aggregate them within the company and do not focus on the simulation management.

Looking at the big picture made by many CFD workflow iterations, stakeholders aim to have a platform to cover requirements that raise when analysts perform multiple simulation iterations (Figure 2.27).

Some requirements that are not currently covered by the existing software are: the data management with tools to automate the data analysis, the automatic execution of multiple multidisciplinary simulations (e.g. underhood cooling, aerodynamics, cabin climatisation etc.) on different simulators, the monitoring of multiple simulations from different simulators. For instance, an interesting feature to provide is the results comparison of multiple simulations. Idea is to provide an interactive visualization tool that collects results from multiple simulations and compare them using a chart. In addition, the aggregation of simulations results gives useful feedback about the overall projects performances, especially to understand the targets achievement.

Unfortunately, a lack of interoperability among CFD software exists. Actually they do not provide export functionality in open format. For instance, the export of geometry data in open format is a well-established functionality (e.g. in STL format) but is not available for the entire simulation case, setup and parameters. Nevertheless the existence of technologies to guarantee interoperability (i.e., SOA architectures, restful and so on), commercial CAD and CAE software often do not provide open access to their services and data. Therefore, in order to provide new functionality (e.g. collaboration among engineers) over existing engineering software the only way is to wrap the software. The main drawback of the software wrapping is the possibility to run into the Vendor and Data Lock-ins AntiPatterns [18].

The integration of CFD software, engineering tools and existing infrastructures as well as the collection of data from different sources provide the base to build new value-added service that cannot exist without such integration. For instance, the collaboration around simulation data and results make sense only within an integrated environment.

In order to successfully integrate software in one end-user environment it is important to integrate software and hardware both syntactically and semantically. We experienced that the main difficulties concern the semantic aspect of the software and the data representation.

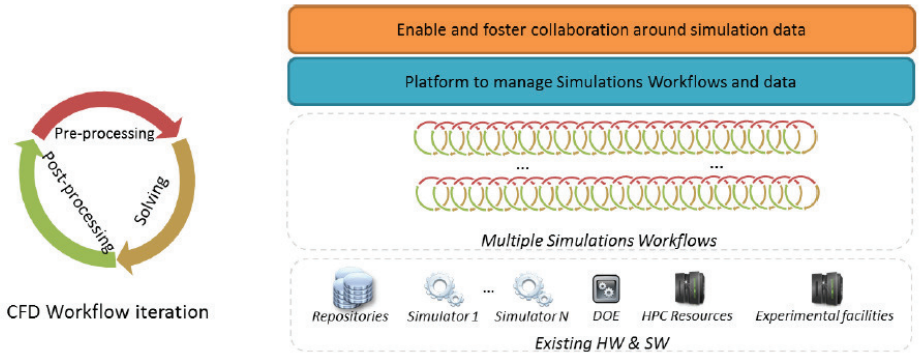


Figure 2.27: Multiple Workflows Management.

2.19.1 Automatic Simulation Workflows Management

The idea is to automatically manage a set of Multi-Disciplinary and Multi-Objective simulations. Of course, some tasks are manual and actually they are difficult to automatise. For instance, the vehicle geometries CAD manipulations are mainly manual activities.

The aim is to semi-automatise the workflow performing automatically some iterations. Considering the simulation workflow phases, the first step is to prepare the mesh (e.g., mesh clean-up), essentially a manual activity that requires the engineer actions. Therefore, it cannot be automatised and must be done manually. The geometry CAD can be parametrised so the geometry parts can be modified passing a numeric value such as the absolute position or numeric value that identify the geometry part translation and rotation. For instance, the spoiler can change the position in according to the specific parameters. Then, engineers decide which simulations must be executed and the parameters to use. In this way, the platform is able to setup the case to simulate, run the simulations on the cluster with different parameters both geometric and simulation parameters. Platform must be able to monitor the simulation jobs over time providing statistical information. At the end, the platform collects data and makes the documentation (e.g. spreadsheet and slides).

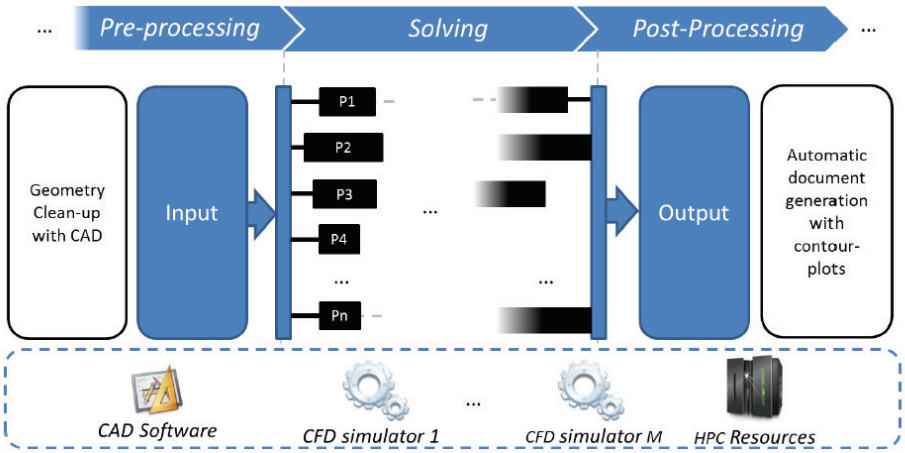


Figure 2.28: Multidisciplinar, mutliobjects and automatic CFD workflows.

An engineering challenge is the time required to perform and complete all the simulations. It is mainly a CFD task to optimise the different simulations and complete them within a useful period in

according to the industrial targets. For instance, the industry could decide to get all results within two weeks. From the information technology point of view the aim is to integrate all these software and be able to communicate with them exchanging the data especially because a lack of interoperability exist. For instance, it is important to be able to set simulation parameters and submit the jobs on the cluster. At same time, the simulation control and monitoring is another important feature. For instance, a CFD simulation can diverge so the platform must be able to monitor the simulation and block the execution of the next jobs. Of course, as described previously, CAD software interaction is important mainly to manipulate the geometry mesh leveraging on the morphing changes of the geometry shape within a continuous range.

Figure 3.16 shows an example of the use case compared with the simulation workflow phases on the top. The platform submits different jobs over the time changing the simulation input parameters in according to the values provided by a Design Of Experiments (DOE) system (e.g., modeFrontier, Dakota). The platform is able to monitor and control the simulation jobs. The platform is integrated with the other internal systems such as the CAD software, the simulators and HPC resources.

2.19.2 Experiment Data Management

Engineers perform experiments in real settings. For instance, in the automotive field, experiments are extremely important. Vehicle engines are constantly tested in controlled environments. A single engine run generates a huge amount of data stored usually in text format (e.g., Comma-separated values - CSV file format). Engineers usually run multiple experiments with different conditions (e.g., experimenting different paths) for many hours. Actually, it does not exist a unique common format to store the experiment data: any test-bed engine system generates data in different format. Later, engineers analyse these experimental data through comparisons.

Sometimes unexpected events can occur. For a real engine, a unexpected event is a high pressure value inside combustion cylinder for particular setting conditions. In this context, engineers face the problem to understand in which conditions the event occurs and why. Therefore, they needs to explore experiments dataset and compare thousands of experiments together. Automatic features as well as exploring and query features to get insight into the experiments will be really useful for engineers. For instance, experiments data aggregation by pressure values gives the opportunity to cluster experiments and identify outliers.

This use case conceptually is very similar to the simulation use case. Instead to have simulation data, engineers deal with experimental data. In both cases, users assert that they deal with big data and they require to explore datasets and compare data. The requirement to explore datasets has pushed the design and development of the Exploration and Visualisation tool further described in the next Chapters.

2.20 Conclusions

This Chapter provides a deep analysis of collaborative requirements to design a platform to collaborate around simulation data and promote the share of models. Through observations, interviews and a user survey we collected many requirements, so through further screening we were able to identify the key and essential collaborative requirements.

The 2nd section has described the main engineering Functional and Non-Functional requirements gathered in the Fiat Chrysler Automobiles use case. The integration of existing software is a relevant topic for the practitioners especially because for simulator software a lack of interoperability exist. In addition, the interaction with the engineering software allows the potential automatic management of the simulation Workflow.

Chapter 3

Floasys Platform Architecture

Contents

3.1 Introduction

This chapter describes the Floasys functionalities and shows its graphical user interface (GUI). Floasys functions have been divided in two groups: collaborative and engineering functionalities. From collaboration point of view, Floasys provides a *simulator independent repository tool* to navigate simulation 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).

From the engineering point of view, Floasys provides the following services to support the CFD Workflow. It provides a service to run, solve and monitor simulation as well as automatic document generation like slides and spreadsheet documents.

In the 2nd part the chapter describes the Floasys architectural solution [44] to meet both collaborative and engineering functional requirements described in the previous chapters as well as Non-Functional requirements (NFRs). The architecture 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). Therefore, the architecture supports the centralisation, annotation, tagging, search and sharing of simulation data to meet the collaborative requirements. At same time it supports the creation of engineering services over simulation data, such as the find of simulation with highest pressure.

The chapter is organised using a top-down approach as follows. The Section 5.1 introduces the general ideas behind the Floasys's architecture that is described further in the next sections of the chapter. The Section 3.5 gives an overview of the architecture in terms of patterns, architectures and protocols to guarantee its reproducibility. The chapter tracks and maps the collaborative requirements with the solution ideas and the specific implementation technologies (i.e., libraries) used to develop architecture.

In the final part, the chapter introduces and describes a tool called ExploraTool to visualise, explore and graphically query large repositories of simulations. Instead of starting with the empty list, ExploraTool provides an initial overview of the repository content, progressively grouping the simulations by their main attributes, such as brand, vehicle model, power source, engine type and so on. Users can interactively navigate the repository view through drill-down, roll-up and rearrangement operations. In this way, using the ExploraTool, simulation analysts can visualise, explore and filter large repository of simulations as well as select groups of simulations to compare their performances. Large industries like FCA have large repository of simulation data and they must be sure that analysts have access to previous generated data. ExploraTool provides an overview of the repository content fostering its exploration selecting the key attributes to limit the space of results to find previous simulations. In addition, ExploraTool is immediately useful to answer questions like *how many simulations we performed for the vehicle X?*, and *why for the vehicle Y with the engine type Z there are few simulations?* ExploraTool gives a further advantage for the technical managers who can periodically check the working in progress on a specific vehicle model. The idea behind the ExploraTool is generic and can be easily used with the repository of experiments as well as other type of big data sets. In order to do this, it is necessary to identify the common and interesting data categories, and build the relative hierarchy that ExploraTool will render.

3.2 Floasys Graphical User Interface

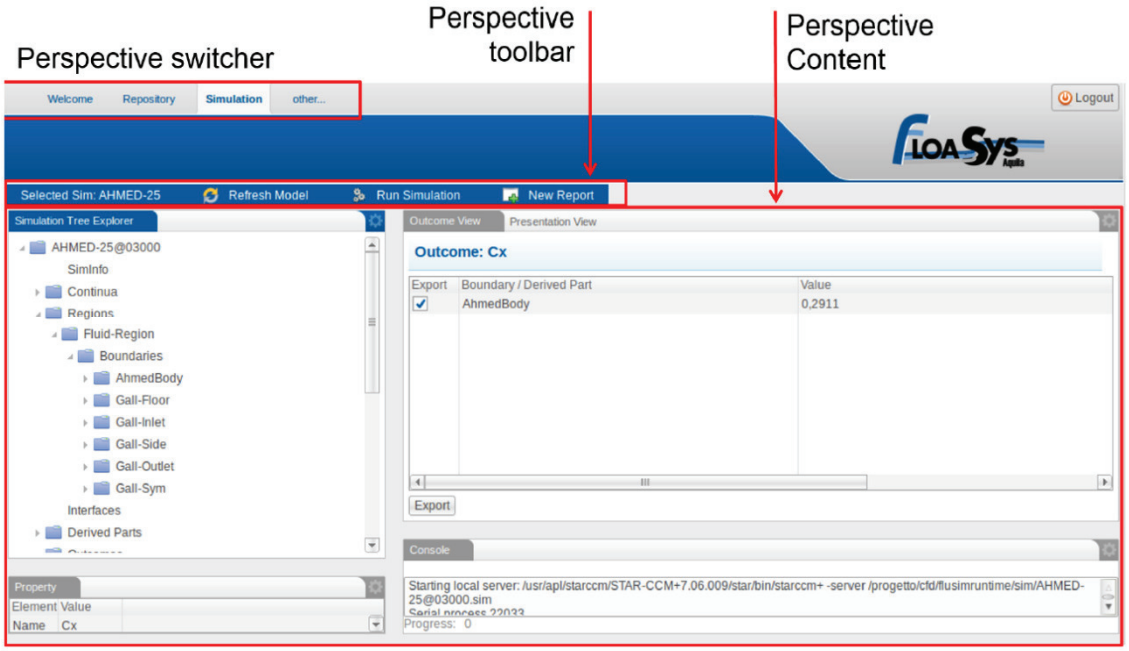


Figure 3.1: Floasys Graphical User Interface.

Floasys provides a re-configurable GUI based on Perspectives and Views concepts provided by Eclipse Remote Application Platform (RAP) [42]. 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. In each perspective the content is organized in multiple views. Each view effectively contains the data using the available widgets. For instance Figure 3.1 shows the “Simulation Controller Perspective" with four views: Simulation Tree Explorer, Property, Outcome View and Console.

3.3 Collaborative Features

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 3.2 depicts a real-world Floasys workflow that is difficult or time-consuming without the designed Floasys 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 cannot 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 cannot 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.

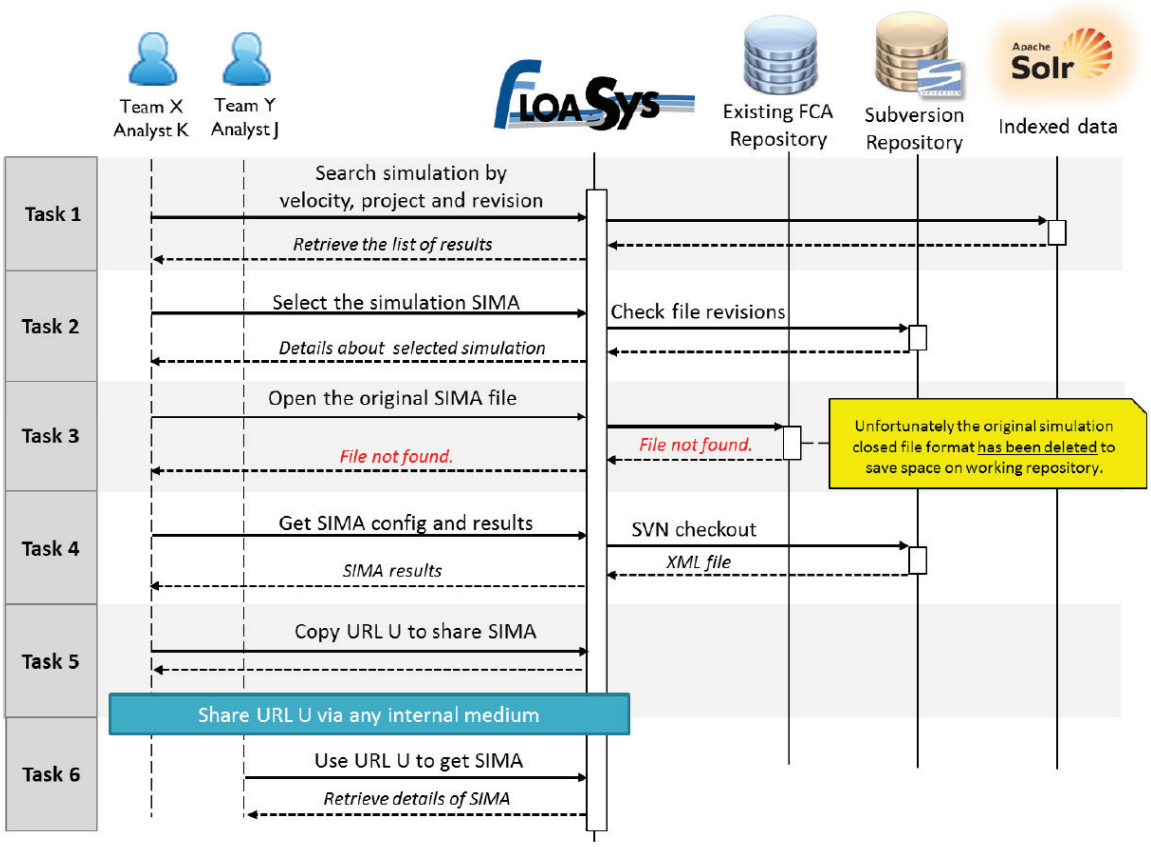


Figure 3.2: Example of a typical workflow supported by Floasys.

Using Floasys, nevertheless the original file was deleted, the user can get the simulation data, setup and results. Of course, these data cannot 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.

3.3.1 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.

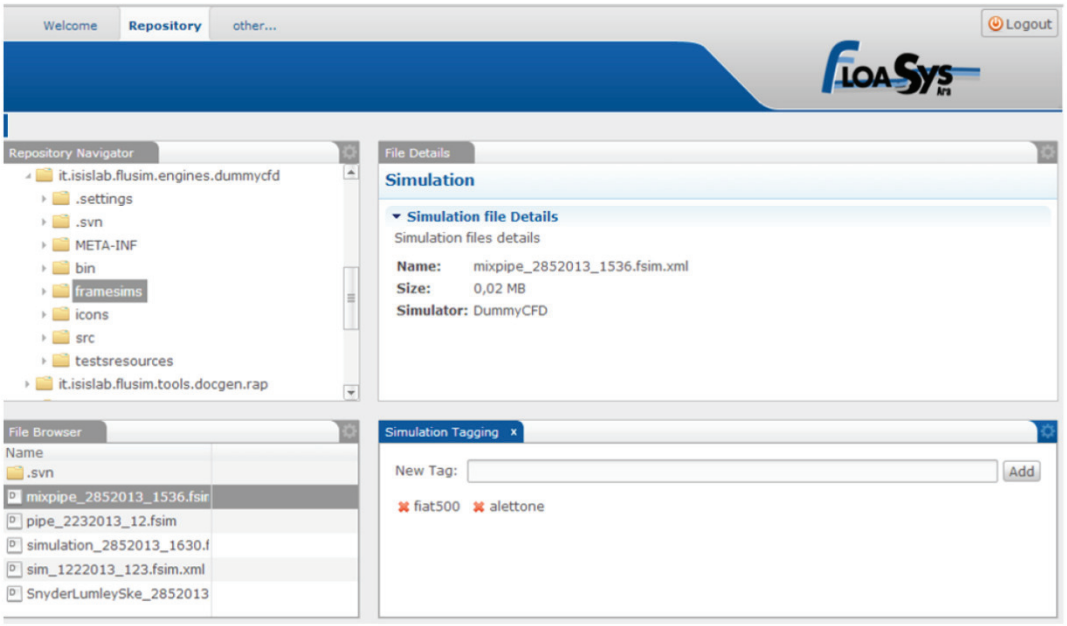


Figure 3.3: Repository tool to navigate and tag a simulation repository.

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 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 cannot 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. 3.3). Tags are both unstructured with free tags and structured inserted filling out standard forms like in Figure 3.4.

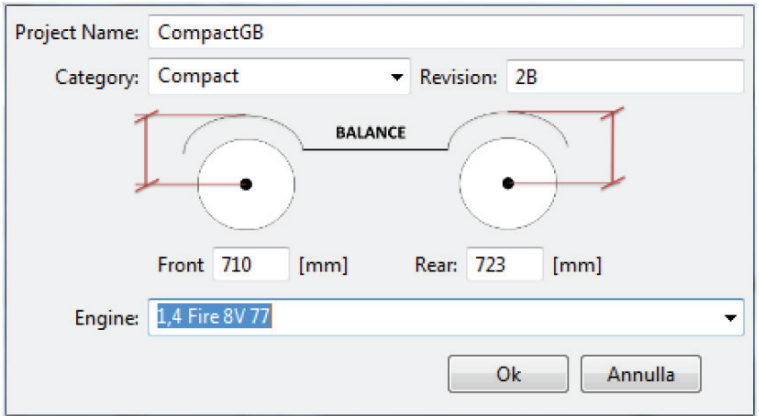


Figure 3.4: Example of structured data.

3.3.2 Search tool and Data sharing

The Search tool (Fig. 3.5) 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. 3.5). 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.

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.

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. The idea is to share data by simply exchanging unique reference to the specific simulation data. URLs identify simulation data and inherits 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 [43], Floasys supports the *asynchronous* data sharing for both co-located and distributed teams.

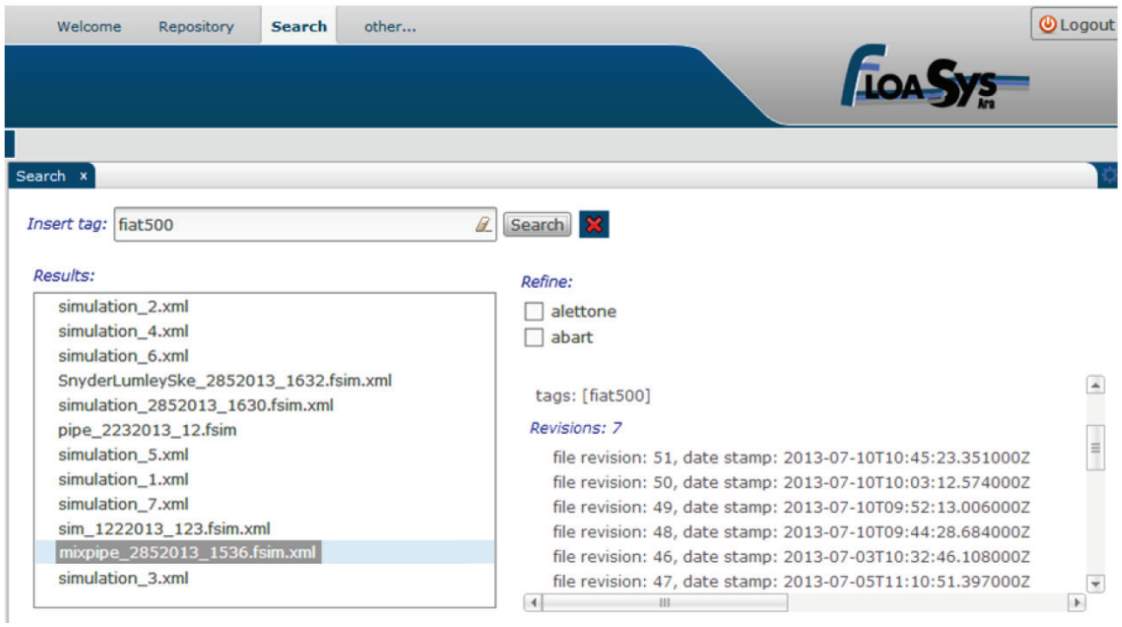


Figure 3.5: Search Tool

		TIME	
		Same Time (Synchronous)	Different Time (Asynchronous)
SPACE	Same Space	**1st Quadrant** Spontaneous collaborations, formal meetings, classrooms	**2nd Quadrant** Simulation data sharing
	Distributed	**3rd Quadrant** Video conferencing, net meetings, phone calls	**4th Quadrant** Simulation data sharing

Figure 3.6: CSCW Quadrants.

3.3.3 Web-based 3D Model Visualisation

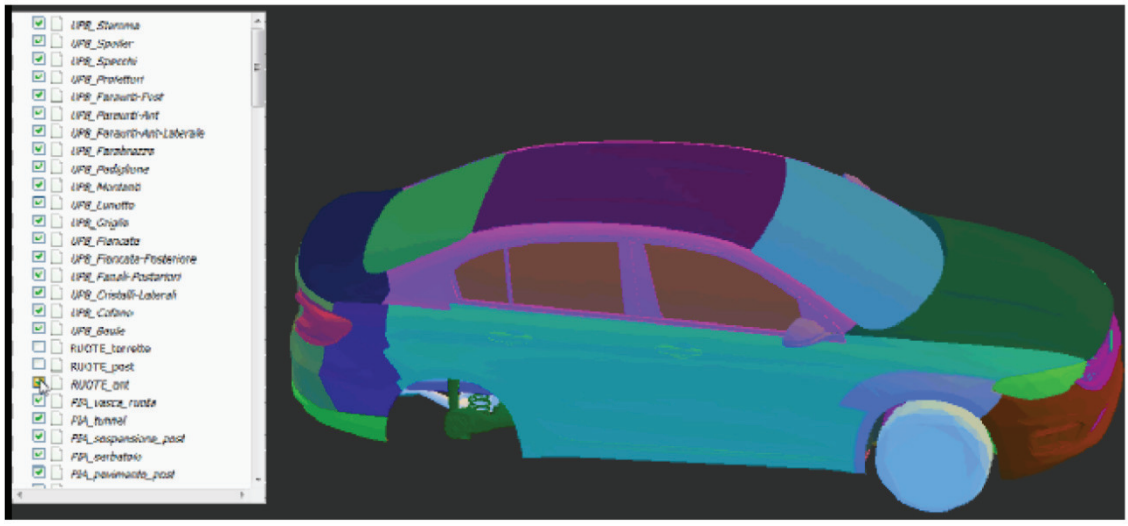


Figure 3.7: Floasys 3D model visualisation.

Floasys shows a reduced 3D geometry of the simulated vehicle. Through this tool, engineers can quickly 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. 3.7). The user can activate or deactivate some parts and can perform the basic zoom and pan operations. Figure 3.7 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 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 visualization 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.

3.4 Engineering Features

This section describes the Floasys engineering functionalities to support the engineering activities. The CFD workflow is made by three parts: pre-processing, solving and post-processing. Floasys has at least one function in any of the CFD workflow phase. The Figure 3.8 shows the CFD Workflow and for each step shows the functionality supported by Floasys.

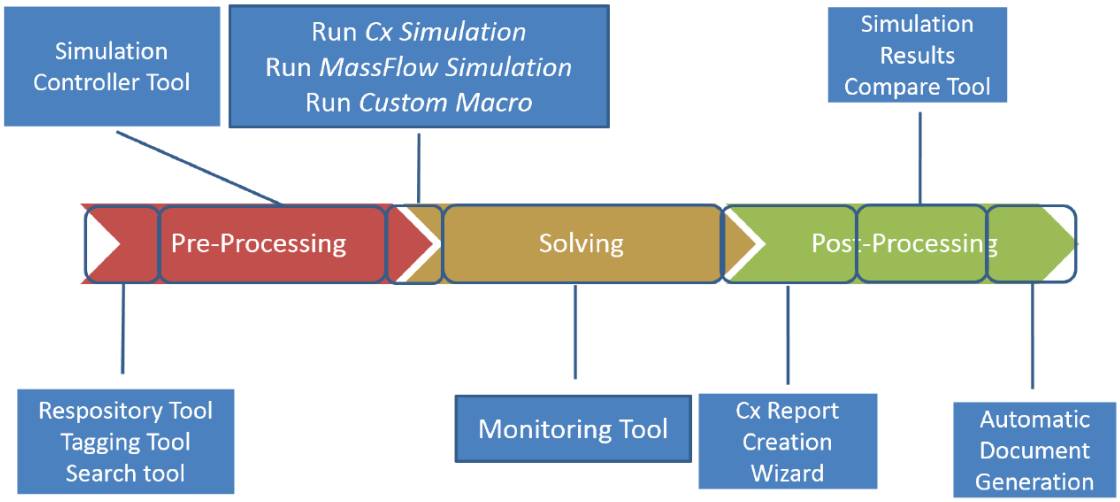


Figure 3.8: CFD Workflow and tools provided by Floasys.

In order to understand how the users interact with the system, the Figure 3.9 shows a typical CFD workflow with the tasks and, for each of them the used Floasys Tools. For instance, after the log-in and at beginning of the pre-processing phase, engineers use the *"Repository Tool"* to navigate the simulation repository and select a simulation file (Task B, Fig. 3.9). Then, the "*Simulation Controller Tool"* opens the simulation and shows its content and details (Task C, Fig. 3.9).

In order to solve a simulation, Floasys has multiple "*Run simulation*" wizards. In the engineering field, the simulation running takes long time (hours) so it is important to monitor them during the solving phase. CFD simulations are numerical simulations so engineers look to convergence charts (Task E, Fig. 3.9) to understand whether the simulation is converging or not.

Finally, the tools available in the last Post-Processing step are the "*Simulation Results Compare tool*" (Task F, Fig. 3.9) and the tool to generate the documentation automatically (Task E, Fig. 3.9).

3.4.1 Simulation Controller Tool

The Simulation Controller Tool shows data of a simulation and provides functionality to interact with it. Obviously, previously the user has selected the simulation using the "*Repository Tool* ". The screenshot in Figure 3.10 shows data about a simulation example provided by CD-adapco STAR-CCM+ : AHMED-25. The Controller Tool shows on the left an overview of the simulation data. The data structure is tree-based and has rendered through a tree widget. For instance, the picture shows on the left the geometric boundaries: the wind tunnel boundaries such as the Floor, Inlet, Outlet and the Side as well as the AhmedBody placed in the middle of the wind tunnel. The same picture shows on the right the Cx value that is the simulation running result and the value in which the engineers are interested.

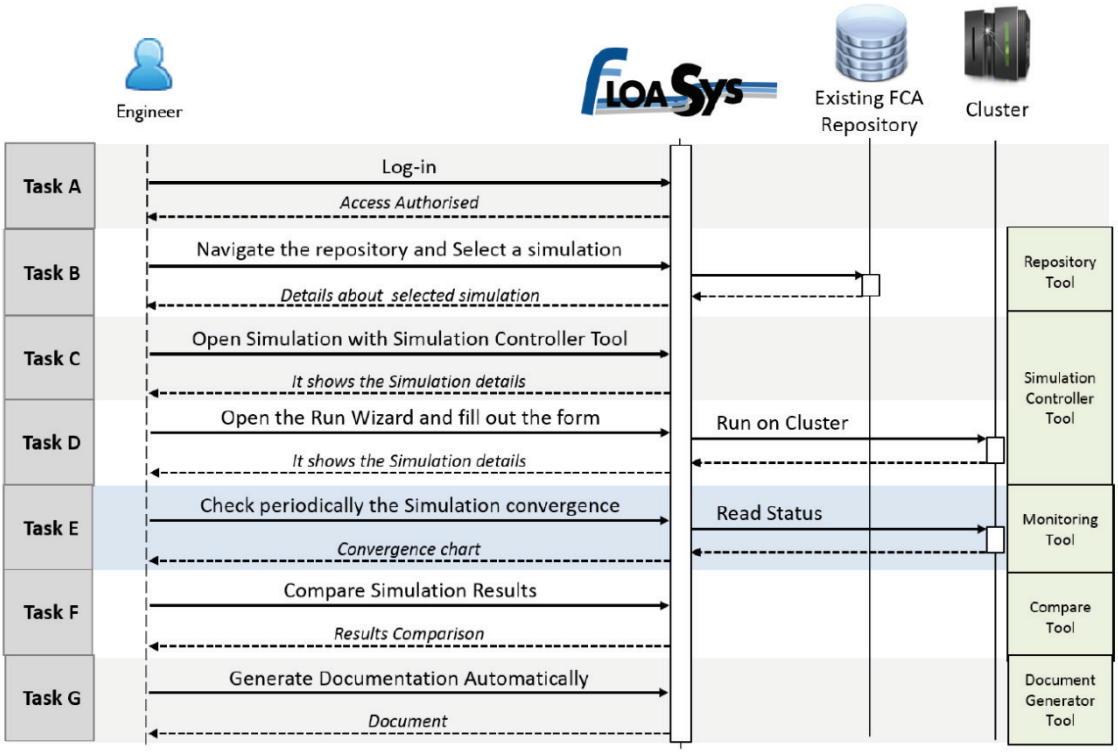


Figure 3.9: A typical engineering workflow.

The tool is based on the Simulation Model concept that contains all the data about the selected simulation in a tree-data structure. Simulation data have a tree hierarchy structure made by nodes that represent the single data. Simulation data are the regions, the boundaries, the interfaces between regions, the physical properties, the outcomes and so on. The tool shows this data in the Simulation Tree Explorer on the left side.

For each simulation model node that is selected in Simulation Tree Explorer (left), the Property view shows all information about the node for example the node name. On the right side the system shows other detailed information on the selected node. In the simulation tool the focus is on the selected simulation. The toolbar provides the functionalities related to the selected simulation. The toolbar shows on the left the simulation name, so the engineer is aware about which simulation he has chosen.

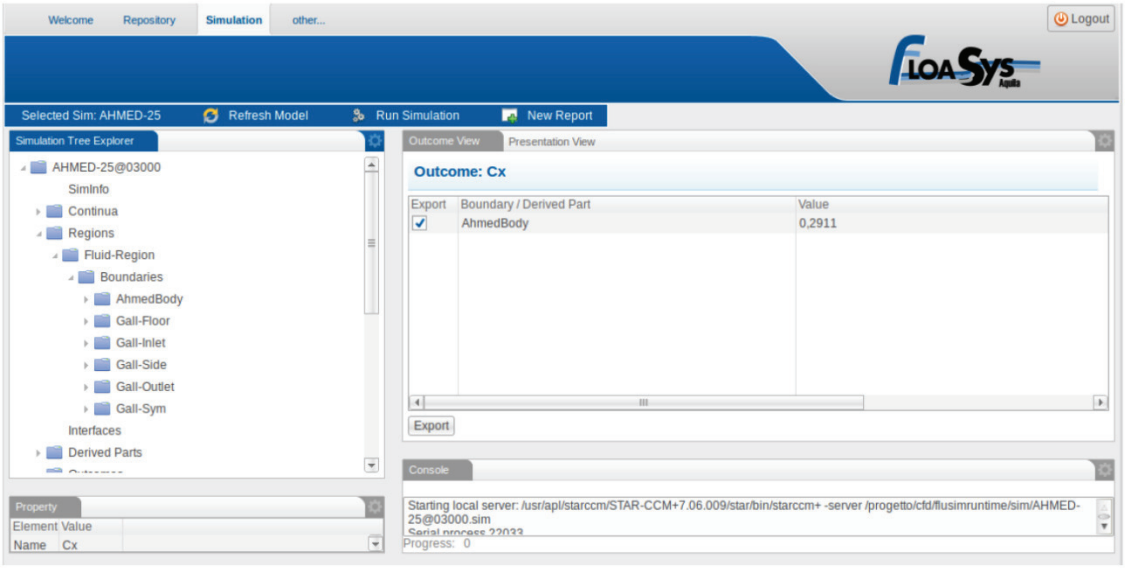


Figure 3.10: Simulation Controller Tool.

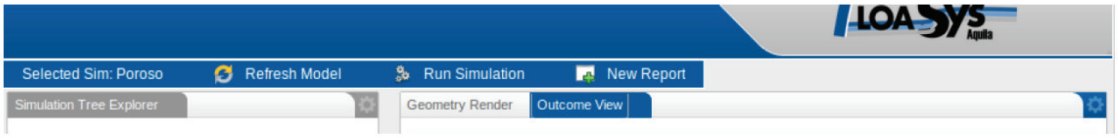


Figure 3.11: Simulation Tool Toolbar

Figure 3.11 shows the Simulation Controller Tool toolbar: the main entry to other functionality of the Selected Simulation. This toolbar continuously evolves providing new features, currently it contains the following functionalities:

- **Selected Sim**: it shows the selected simulation name, if the simulation is stored in a file than the selected simulation is the file name;
- **Refresh model**: to refresh data shown, it extracts the simulation data and shows them in the simulation tree under the toolbar;
- **Run simulation**: it solves the simulation; usually it uses HPC resources to run the simulation.

In particular, Floasys has a wizard of three pages to run a simulation (Fig. 3.12 shows two of them). In the first page the user can choose among multiple standard running types (e.g., Cx simulation running, MassFlow run and so on). The Wizard's pages and input parameters change in according to the selected simulation type and for each choice, Floasys configures the simulation. For instance, the Figure 3.12 shows the parameters to solve a simulation that calculates the Drag Coefficient (Cx value). The running of a simulation requires the access to High Performance Computing resources, so the last Wizard page always asks the cluster name or IP address (industries usually have multiple clusters), the number of processors to use and the user credentials.

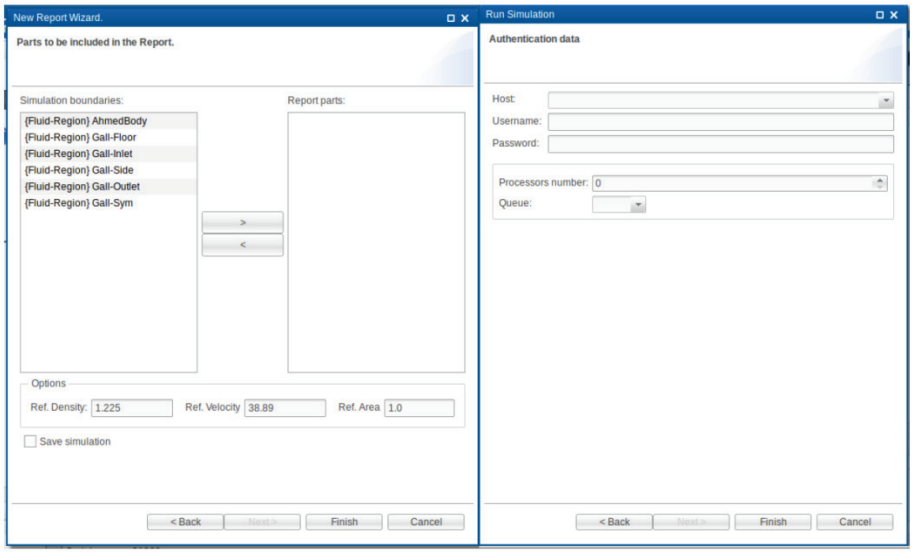


Figure 3.12: Screenshots from the Run Simulation Wizard.

3.4.2 Monitoring Tool

The Monitoring Tool is used to monitor the running simulations. It is generally used when the simulations run on a computer cluster. In order to use the Monitoring Tool the user must connect to the HPC resource by clicking on the **Connect** button. To login the user must insert the IP address or the DNS, and the password. The available usually hosts are fixed at configuration time so the user can select it from the list of available hosts. After the authentication, the monitoring tool shows the jobs submitted on the cluster. The monitoring perspective is divided into two columns. On the left side there is the job list. The list contains both the running and waiting jobs. When the user clicks on a job in the list, Floasys shows on the right the convergence chart. For example, the list shown in the Figure 3.13 contains three jobs. The user can select a job from the list to show the chart on the right. Of course, everything is configurable simply clicking on the Change Graph button. Floasys reads for each simulation, its log file and extracts the quantities to show in the chart.

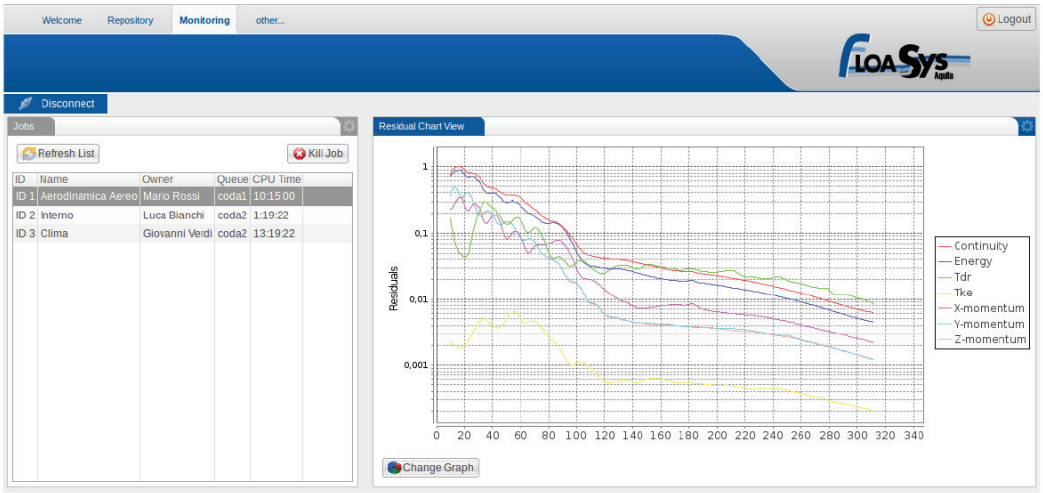


Figure 3.13: MonitoringTool to check the simulation convergence.

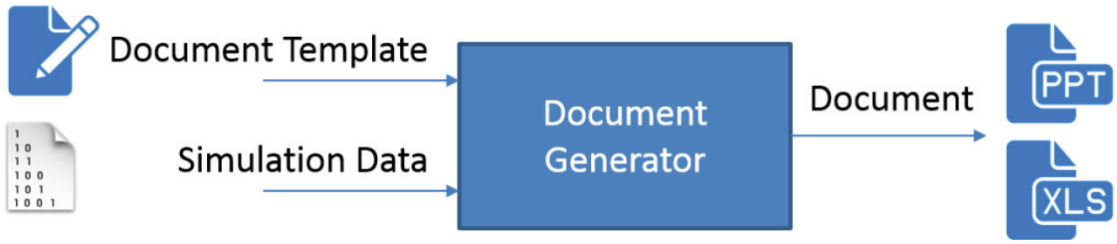


Figure 3.14: Generation of documents from simulation data.

3.4.3 Documentation Tool

In order to generate documents the first approach is to write a specific procedure for each document to get the simulation data and make the document. This approach has the drawback that the document structure is stored within the generator program so to change it, the generator source code must be changed; an activity that cannot perform the system end-user. Therefore, this approach is not flexible. Another approach uses a template document with marker elements inside. These markers will be replaced by data during the document generation process. Figure 3.14 shows the document generator, a black box from the design point of view, with the template and simulation data as input and the generated document as output. This approach is flexible compared to the first one because it is possible to change the document template to change the format of the generated document, but still, for complex documents, it needs to write some code, an operation that end-users cannot perform.

	A	B	C	D	E
1	**Summary**				
2		*Part name*	*Cx Value*		
3		Under Hood	$[SUM(D10)]		
4		Ruote			
5					
6	`<jx:forEach items="${simModel.reports}" var="report" select="${report.type.typeName.equals('CX') == true}">`				
7	*Report:* **${report.name}**				
8		*Boundary Name*	*Boundary Type*	*Cx Value*	
9		`<jx:forEach items="${report.dataItem}" var="uh" select="${uh.component.name.contains('UH')}">`			
10		${uh.component.name}	${uh.component.type.typeName}	${uh.value}	
11		`</jx:forEach>`			
12		`<jx:forEach items="${report.dataItem}" var="wheel" select="${wheel.component.name.contains('RUOTE')}">`			
13		${wheel.component.name}	${wheel.component.type.typename}	${wheel.value}	
14		`</jx:forEach>`			
15	`</jx:forEach>`				
16					

	A	B	C	D	E
1	**Summary**				
2		*Part name*	*Cx Value*		
3		Under Hood	0,96		
4		Ruote			
5					
6	*Report:* **Force Coefficient 1**				
7		*Boundary Name*	*Boundary Type*	*Cx Value*	
8		InLet - UH	Inlet	0,51	
9		Pressure - UH	Empty	0,45	

Figure 3.15: Example of a template and a generated document.

The idea behind the Documentation Tool is to generate automatically the documents from the document template and the simulation data. The template has the same original format with special tags or keywords within the document template. The tags within the template use a specific language, like Velocity or Freemarker that are two Java template engines. Figure 3.15 shows two screenshots, the first one on top shows the Excel document with the tags and the second one shows another document generated by the Documentation Tool with simulation data.

The same idea has been used also for the Power Point presentations. The template has on each slide a keyword or tag formatted properly that will be replaced by the content during the generation phase. Also for pictures on slides there is a tag that will be replaced by images.

3.4.4 Parametric Exploration Tool

Engineers perform multiple simulations for the same product with different parameters. In literature already exist Design Of Experiments software, two examples are modeFrontier and Dakota. Formally, in a design of experiment we have *X* that is the set of input variables to explore. Of course, for each variable only a set of values is valid. The set *Vx* is the set of values for the variable *x*. The set of experiments is *E*. The number of experiment usually is very high and it is impossible to perform all of them within the budget and time constraints. Therefore, engineers need to choose a subset of experiment to perform. For instance, Figure 3.16 shows a pipe with two inlet fluids and one outlet fluid.

	Experiment A	Experiment B	Experiment C	Experiment D
Initial velocity [m/s]	2	10	20	20
Inlet1 velocity [m/s]	2	10	20	39

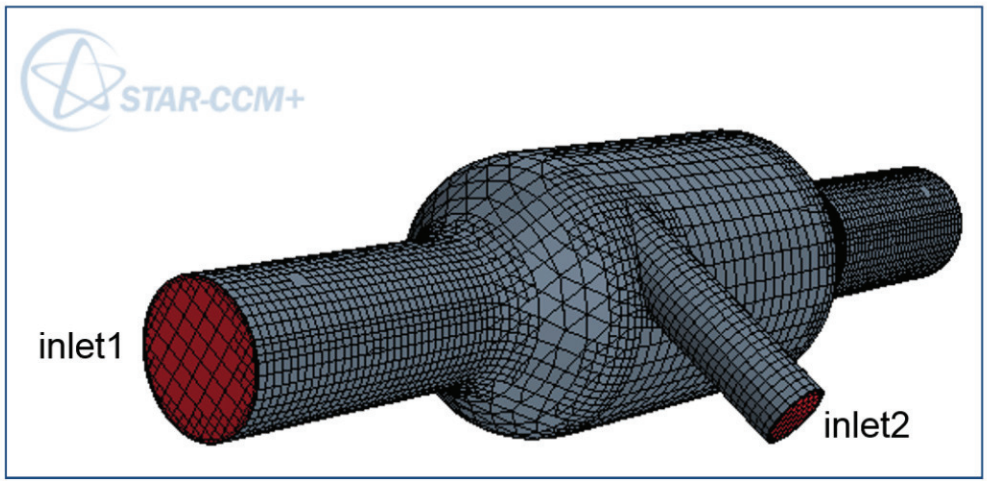


Figure 3.16: An example of Parametric Study.

In order to design this pipe, multiple experiments must be performed changing for instance the inlet velocity. The table shown in Figure 3.16 shows four experiments with the relative inlet velocities.

The Parametric Study Tool (Fig. 3.17) allows the design of experiments. It shows the simulation data with its parameters on the left and the experiments on the right. The user can choose and drag a parameter from the simulation tree on the left and drop it on the right and set the values. In this way the tool knows which parameters must have each simulation. Floasys runs all the simulations on the High Performance Computing resources and collects the results that can be stored within documents.

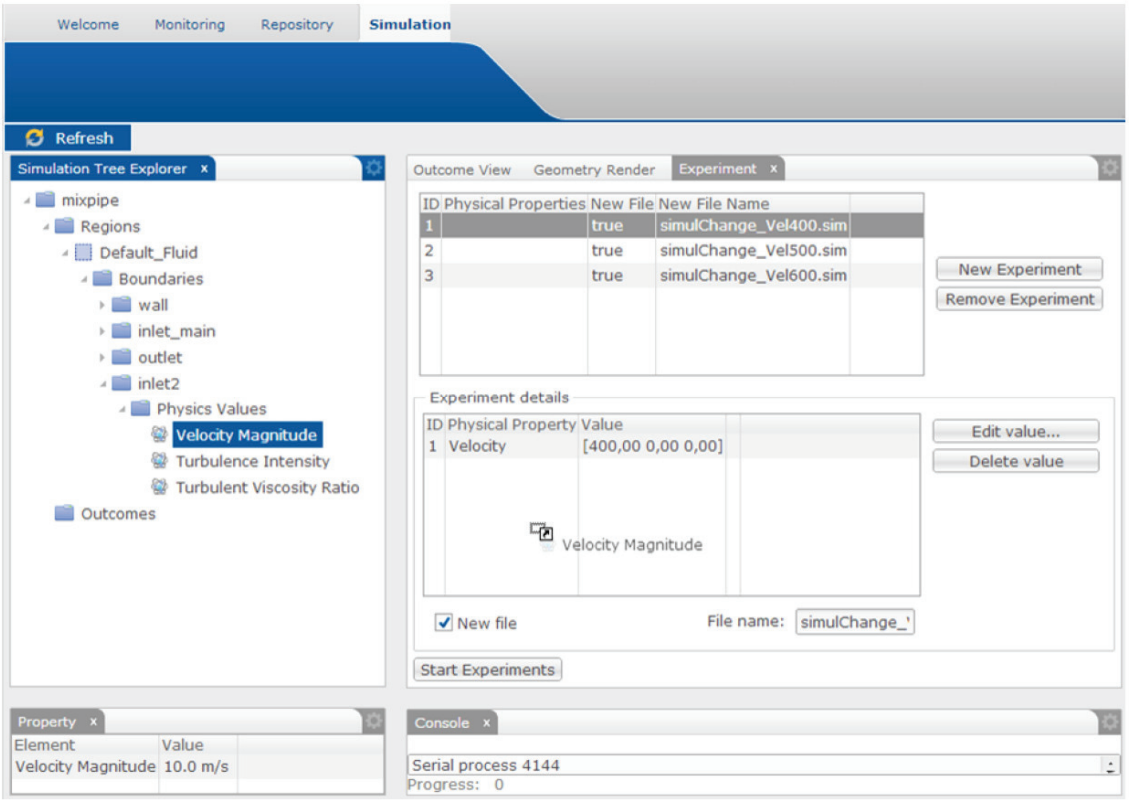


Figure 3.17: Parametric Exploration Tool

3.5 Floasys Architecture Overview

The idea is to collect both simulation and experimental data, and store them in central repositories as shown in Figure 3.18. Then, the architecture provides additional services over collected data. Example of services are the collaborative features to share the data among distributed teams of analysts and the engineering services to manage the simulation life-cycle on the High Performance Computing resources like clusters.

Figure 3.18 shows the Floasys abstract architecture design to introduce its ideas, components, and features. In this way its idea can be potentially adopted in other sectors, fields and contexts like aeronautic, rail and naval), and it can be replicated with different technologies.

Generally speaking, the architecture is based on three layers. In the bottom layer there are the data sources; in the use case the experimental and simulation data. The data management layer is responsible for the data source management. To get simulation data, the architecture can read the data directly from a source file or through the simulator software that generates the data. For

experimental data, the architecture usually for security reason does not connect to the test-beds to get data but the test-bed itself generates textual data (e.g., Common Separated Values, CSV files) that are read by the data management layer. From technological point of view, a data source can be a web services [45] to query the data, a restful service [46] or a piece of software to integrate within Floasys. The data management layer must handle heterogeneities among data sources. Therefore, it has a common open model to represent data.

Floasys provides services over collected data. For instance the opportunity to share simulation data exchanging a Uniform Resource Locator (URL). Another examples are the tools to get insight into data like data exploration, filtering and querying. Engineers often ask for the experiment with the highest pressure value or the ones for which a particular event occurs. On the top layer there is the Graphical User Interface (GUI) can be divided in three main layers as depicted in Figure 3.18. The central layer called **data management** is the core. In order to meet the extensibility and modularity Non-Functional requirements the system relies on the concept of **pluggable software modules**. Each electrical device has a power cable with a plug at its end that can be plugged in a socket of the same shape, type and size. This analogy has been used in software engineering for long time. A module A provides a socket with specific characteristics that can be used by a module B to extend the module A functionalities. Over years the pluggable modules have been implemented with different software technologies, such as OSGi [47]. The Figure 5.1 uses multiple times the pluggable modules concept depicted as a power plug. In order to meet extensibility, it provides two types of extension point. One supports the extension to introduce new data sources (bottom layer) and another type of extension point to provide additional services over the data (top layer).

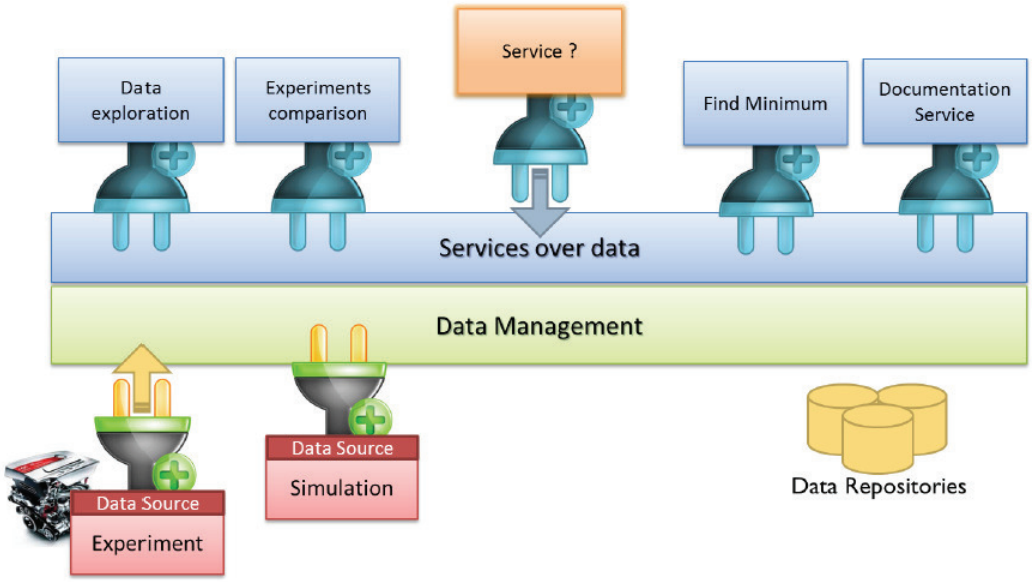


Figure 3.18: Floasys Architectural Solution General Idea

Floasys is based on a Client/Server architecture (Fig. 3.19) developed using Eclipse Remote Application Platform (RAP) [42]. 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 [48] over the HTTP protocol. Servers tend to interact with user browsers using the JSON exchange format [49] because it is easily parsed in client-side JavaScript language [48]. 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. It is an important asset for the industry, so every solution cannot change it to not change drastically how engineers work every day. 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. Floasys server accesses to these existing repository through SSH connections to an existing industrial server. Floasys opens a SSH connection towards the network folder for each connected client. In this way, the SSH connection is initiated with the end-user credentials and he can access only to his authorized files and directories. Therefore, Floasys implicitly inherits the existing files authorizations that have been decided by the central ICT administration.

Obviously, 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 relaying on a well-established search engine technology like Apache Solr [50, 52]. 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 cannot be override.

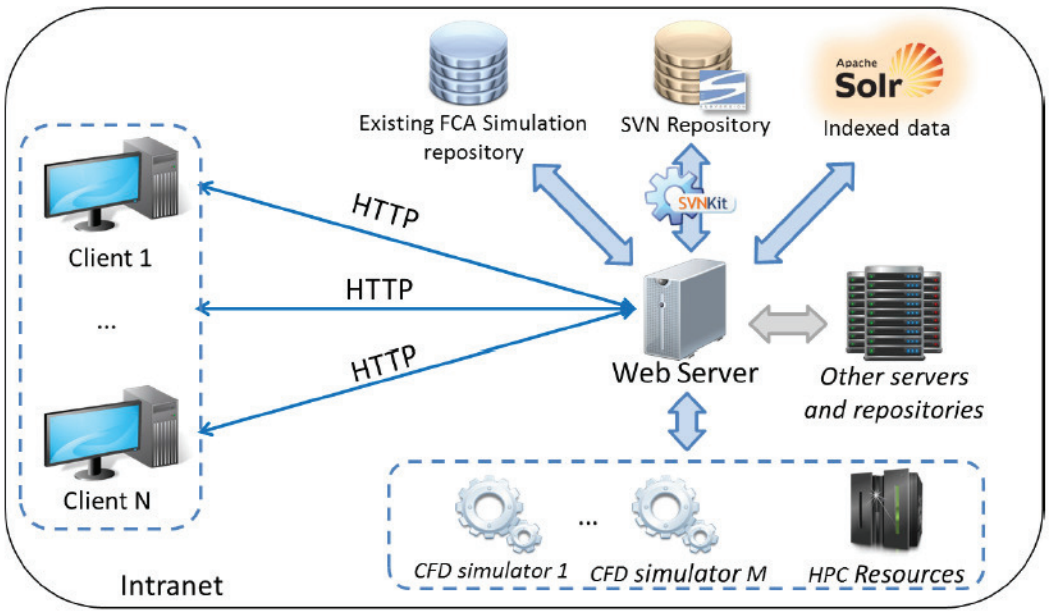


Figure 3.19: Floasys Client/Server Architecture.

To provide authentication and to manage both users and groups, Floasys can rely on existing industrial internal Lightweight Directory Access Protocol (LDAP) servers [53, 54] 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 [55]. 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) and scheduled for the next plans in according to the issue/enhancement priority. Of course, we received the user feedback during the development of the current planned release. Sometimes we received blocking issues that unfortunately did not have been discovered during the planned functionalities testing phase. The blocking issues were planned in according to the Figure 3.20 in a way to react immediately to the incoming high priority request.

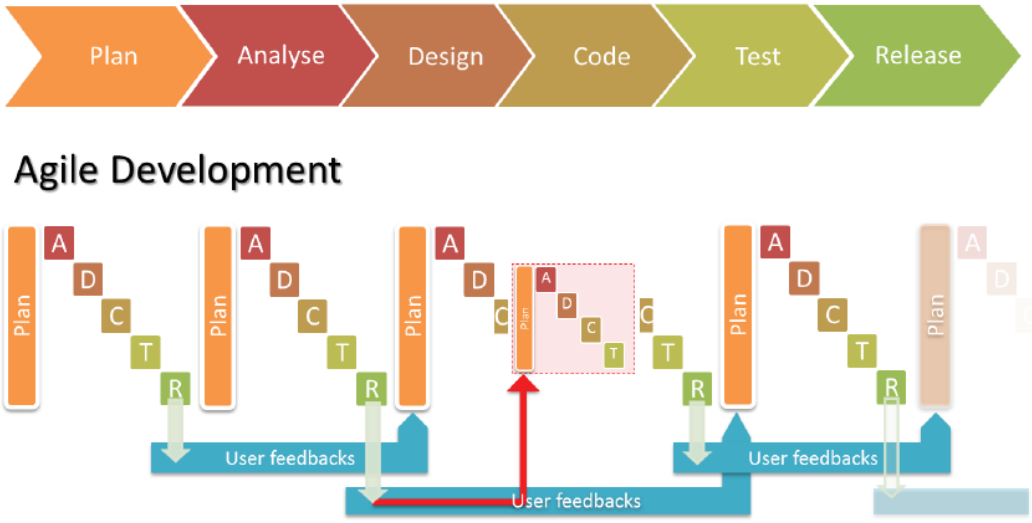


Figure 3.20: Floasys Agile Development and blocking bugs.

3.6 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. 3.24). 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 has the follows characteristics:

- it provides a common APIs to the front-end tools;
- it provides a common unified data representation called Simulation Model for data coming from different vendor systems;
- it is an isolation layer [18] to decouple the front-end from vendor-specific simulator wrappers;
- 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 use of a common isolation layer does not exclude that each wrapper itself is designed with an isolation layer using a proxy pattern. 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.

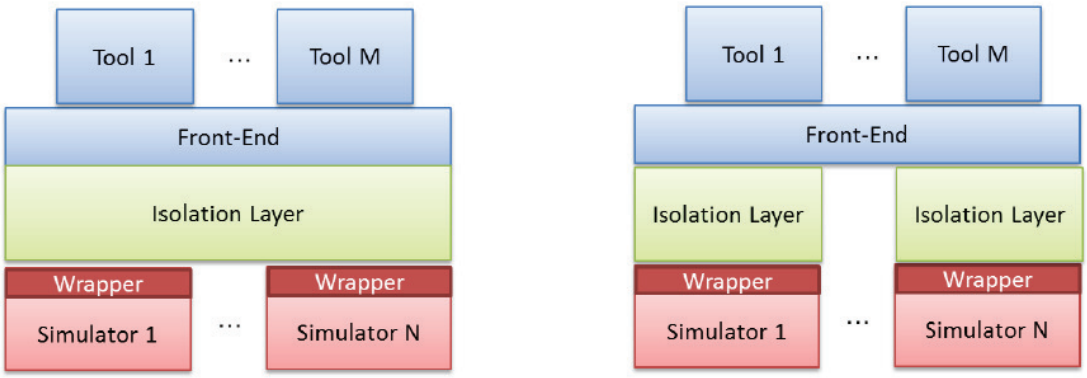


Figure 3.21: Alternative architectural solution comparison.

An alternative solution to the previously described architecture could be the introduction of a separate isolation layer for each vendor software. Figure 3.21 compares the Floasys's architecture on the left with the alternative solution on the right that use an independent isolation layer for each simulator wrapper. The *support of multiple replaceable* vendor products and the *simulators selection process* requirements impose the introduction of a common isolation layer. The alternative solution has the following drawbacks:

- the selection process is performed in the application layer;
- separate isolation layers means also different APIs, differences that must be handled in the application layer.

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. The idea is to interact with the simulator taking advantage of its specific features. Specifically, CFD simulators have an interesting bult-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. Figure 3.22 shows an

example on how to run the extraction macro from the command line. 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.

```
> starccm+ simfile.sim -batch MacroExtraction.java
```

Figure 3.22: Execution of the Macro to extract data from a simulation file.

Figure 3.23 shows the sequence of events and actions performed to extract simulation data from closed file format. The components of the system are: the CD-adapco STAR-CCM+ **Simulator** (right side of Fig. 3.23), the **simulator wrapper** and the **shared folder** (top side of Fig. 3.23). The wrapper interacts with the simulator through the command line. As previously described, the central work to extract simulation data is performed by a Java Macro. The Simulator executes this macro. All the parameter to execute the macro and the data response are serialised and deserialised in files within the shared folder.

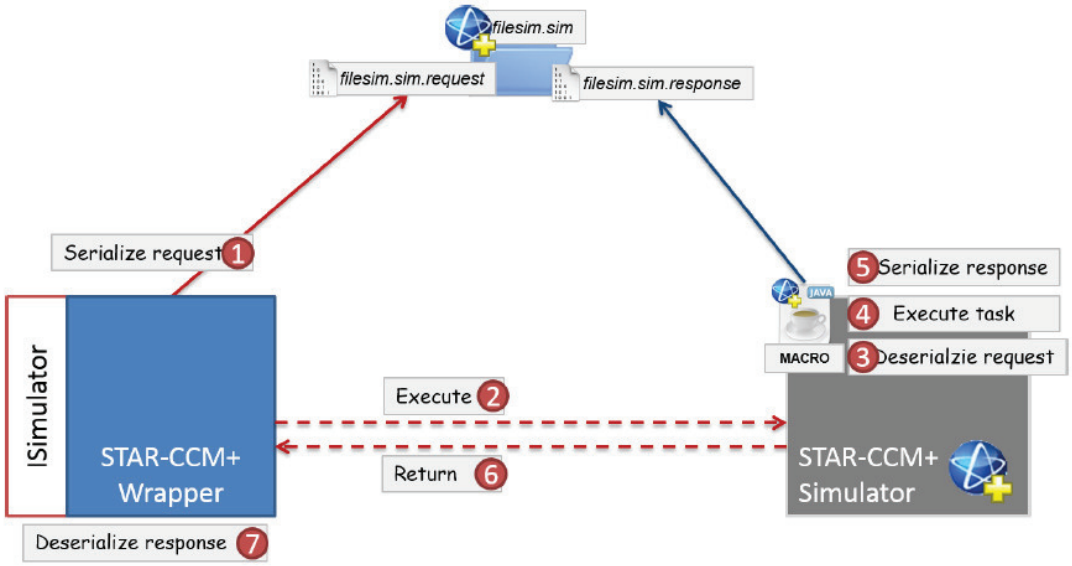


Figure 3.23: How the STAR-CCM+ wrapper extracts simulation data.

The sequence of steps is the following:

1. **Request Serialisation** the wrapper serialises all the parameters to make the request in a file on file system (file with extension *.sim.request), one of the parameter describes the task to perform, for instance the extraction of all simulation data;

2. **Simulator Running** the wrapper runs the simulator and its macro as shown in Figure 3.22, in addition, the wrapper blocks until the simulator execution is completed;

3. **Request Deserialization** the macro deserialises the request file and gets the parameters;

4. **Task Execution** the macro based on the input parameters executes the task;

5. **Response Serialisation** the macro serialises the simulation data and the task results within a file on file system (file with extension *.sim.response);

6. **Simulator Running terminated** the wrapper that was waiting until the simulator running completion and it recognises that the task has finished;

7. **Response Deserialisation** the wrapper deserialises the file response and gets all the data.

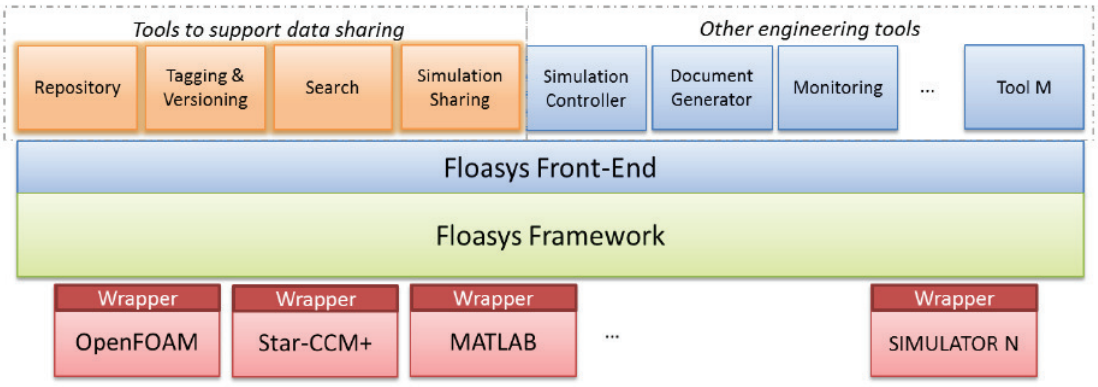


Figure 3.24: Floasys Server-side architecture.

In order to meet extensibility and modularity requirements (Req. 7), the server is based on a pure plug-in architecture [56]. 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 an extension point. In addition, a module can be replaced with another equivalent implementation also at runtime. The Floasys core provides two extensions points to extend its functionalities:

- one hook point to introduce new tools in the upper layer;
- another hook point for new wrappers.

In this way, the following opportunities exist for the final customers:

- multiple Floasys instances can be deployed choosing which modules will compose the overall architecture in according to the industrial needs;
- the industry can identify exactly which modules contain their specific know-how;

- each company can decide to invest money for the development of its own internal modules to customise Floasys and meet specific internal requirements;
- in according to Eclipse Public License [57] (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. 3.24). An interesting Floasys extension planned for the future is to develop a wrapper that collects experimental data (e.g., wind tunnel experimental data, engine testbed). 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.

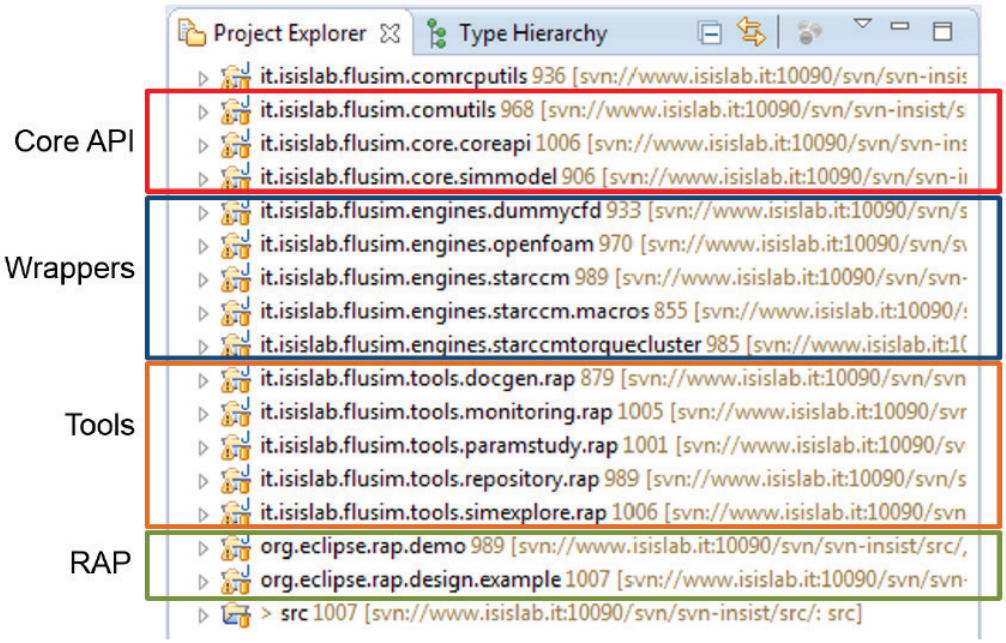


Figure 3.25: Floasys projects within the Eclipse IDE.

Figure 3.24 shows the Floasys architecture with different layers. This software architecture reflects also in the source code organisation. In the Floasys Eclipse IDE there are the following group of projects as depicted in Figure 3.25:

- Core API is the Floasys Framework and contains the simulation model and the interfaces to abstract wrappers and tools concepts;
- Wrappers are the simulator wrappers that know how to interact with the simulator software, for example Floasys has a wrapper for the OpenFoam simulator;
- Tools contain the implementation of the front-end and the user functionalities, for instance the document generator;
- RAP dependencies are the Eclipse Remote Application Platform used to develop Floasys.

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" [42]. 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 but any other Linux distribution can be used.

3.7 Simulation Model

Floasys aims to collect data from multiple different simulators that often use closed file formats. A lack of interoperability among CFD software exists 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 through command line. The architecture has an isolation layer (Floasys Framework in Fig. 3.24) 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*" [18]. Floasys introduces an intermediate common representation for simulation data called **Simulation Data-Model**. It is based on a tree-like data structure as CGNS [58] format. In order to be reusable, it consists mainly of interfaces and abstract classes. In addition, Floasys provides a basic implementation based on the composite design pattern [59]. Figure 3.26 shows part (for simplicity) of Simulation Model Class Diagram. The interface *IComponent* is the abstraction of all components within the model. *IContainer* is a set of components, they represent an intermediate node within the tree-data structure like a folder in the file system. Each component has its specialisation to store specific data types. For instance, there is a node to store the outcome or a physical value. The adding of metadata to this structure is very easy, it is just the adding of a new node to the Simulation Model structure.

In Floasys, each wrapper (architecture bottom layer Fig. 3.24) 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 [60] 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.

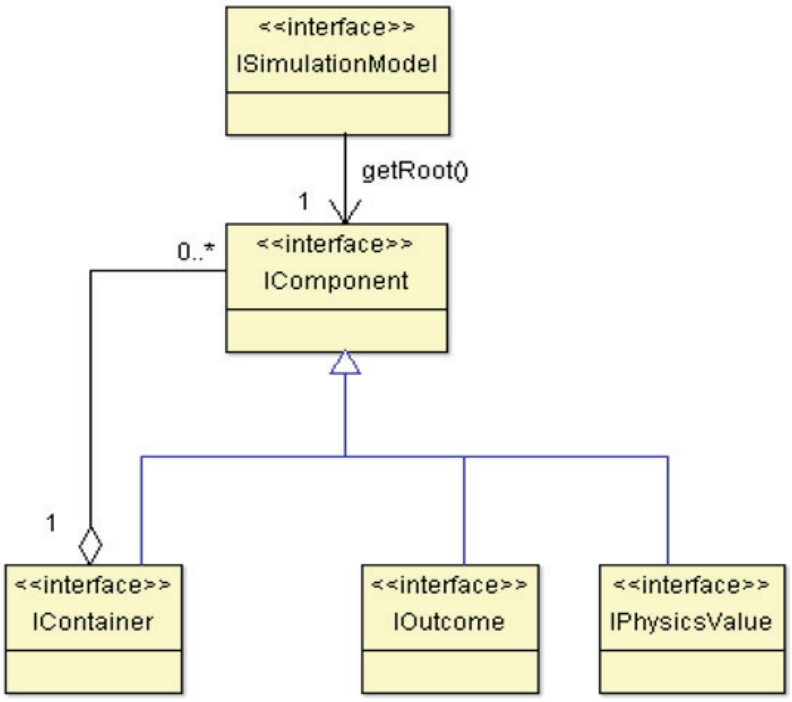


Figure 3.26: Simulation Model Class Diagram

The advantages of the intermediate Simulation Data-Model representation are:

- metadata over data adding custom nodes;
- serialisation in open format such as XML;
- decoupling of wrappers from tools so it is possible to replace a wrapper limiting changes to upper layers;
- 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. 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.

3.8 Simulation Data Management: 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 [61] (or shallow checkout) to checkout a portion of the working directory with the freedom to get more files and directories later [61]. 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 [62]. Apache Solr provides extensions, configuration, infrastructure and programming languages bindings around Apache Lucene. In according to the official documentation [62], 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) 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 visualization 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. For each simulation file (left-side of Fig. 3.27) stored in closed file format, an XML file exists in the SVN repository (right-side of Fig. 3.27) that contains extracted simulation data and metadata in open format. In addition, each XML file is indexed using Apache Solr [62]. 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.

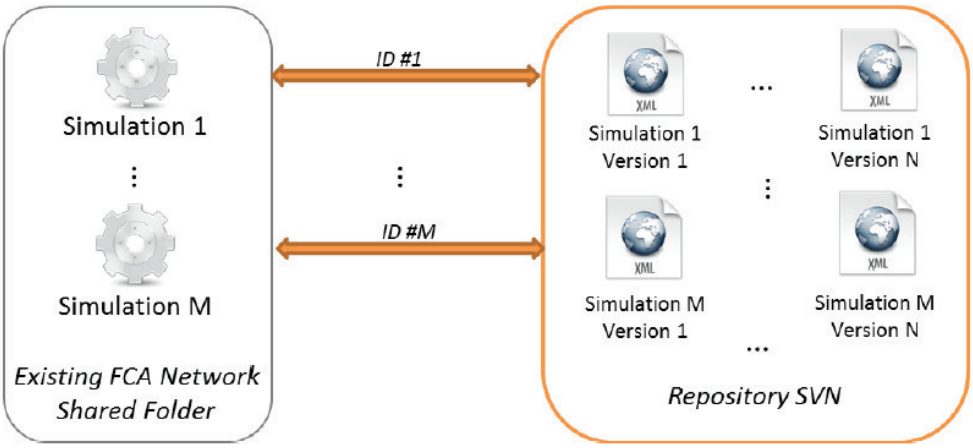


Figure 3.27: Simulation Data versioning.

This solution has been discarded because has the following drawbacks:

- it is difficult to find available and unused fields in the simulation files;
- the simulation files are still stored in closed file format, so the solution is vendor software specific;
- the metadata management requires the access to files through the vendor software using HPC resources due the geometry data;
- 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: SolrJ to interact with the Solr Server and SVNKit to commit and update data to Subversion repository. The 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 cannot store metadata within the simulation files, also because files are in closed file format.

3.9 Collaborative Requirements Traceability

During the design of a system it is essential to track the requirements through all the design steps. The Figure 3.28 depicts graphically the mapping among the stakeholders' requirements, the solutions and the used technologies. Therefore, it shows three columns: the first one lists the collaborative requirements identified and described in the chapter 2, the second column lists the solutions and the last one lists the specific technologies. In addition, the Figure 3.28 has arrows to track and map for each technology and solution, the related stakeholder requirement.

The simulation data centralisation has achieved using central repositories, such as network shared folder. Stakeholders aim to add metadata over simulation data, and Floasys provides a file tagging feature. In addition, engineers want retrieve simulations based on the file name and its content. Unfortunately simulations are closed file format. Floasys extracts the simulation data, stores them in

open format and indexes all the data through Apache Solr, a scalable search engine widely adopted by big digital firms. The worldwide and the dispersed teams have triggered the requirements to share simulations. Floasys supports the sharing of data through the exchange of URLs, a standard technique to share resources over Internet. Floasys has a plug-in based architecture and a central layer called isolation layer to meet the extensibility and modularity Non-Functional requirements. Within the isolation layer it uses a common unified Simulation Model that handles the simulations heterogeneities.

3.10 Code Snippets

This section describes two code snippets took from Floasys: how to run a simulation and how to extend Floasys. The main aim is to show with two practical examples the concepts described in the previous sections like the extension by plug-in.

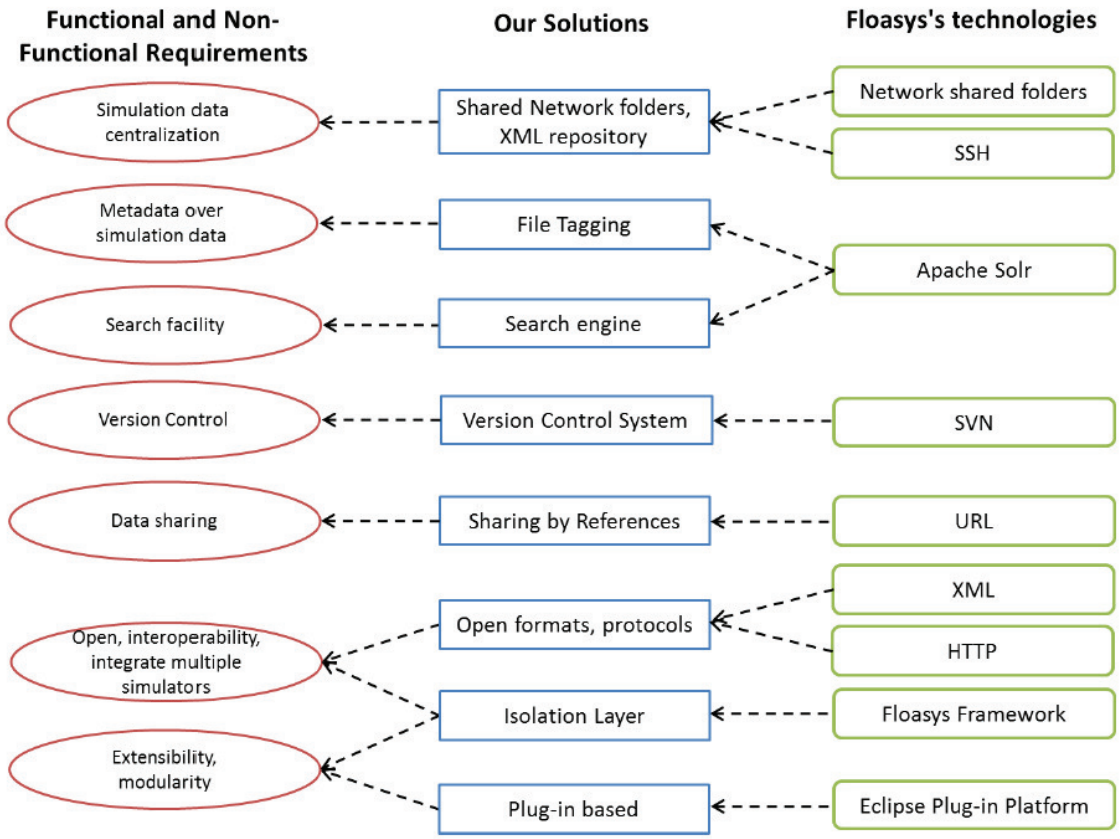


Figure 3.28: Mapping of requirements, solutions and technologies.

3.10.1 How to run a simulation

This section describes a practical example of the Floasys Framework use. Floasys supports both collaborative and engineering tasks. Here, this section presents a typical engineering workflow supported by Floasys called *run a simulation*. An engineer selects a simulation file from the

repository and solve it using the available High Performance Computing resources. This workflow and its relative tasks are depicted in Figure 3.29.

Task A the actor uses his credentials (a pair user name and password) to be authenticated within Floasys;

Task B the user selects a simulation file through the repository tool with the goal to solve it;

Task C the user through the wizard inserts all the parameters to run a simulation (e.g., number of processors);

Task D the user clicks on finish and Floasys runs the simulation on the cluster;

Task E the run of a simulation takes hours so the engineer uses the Monitor Tool that shows a chart to monitor the simulation convergence.

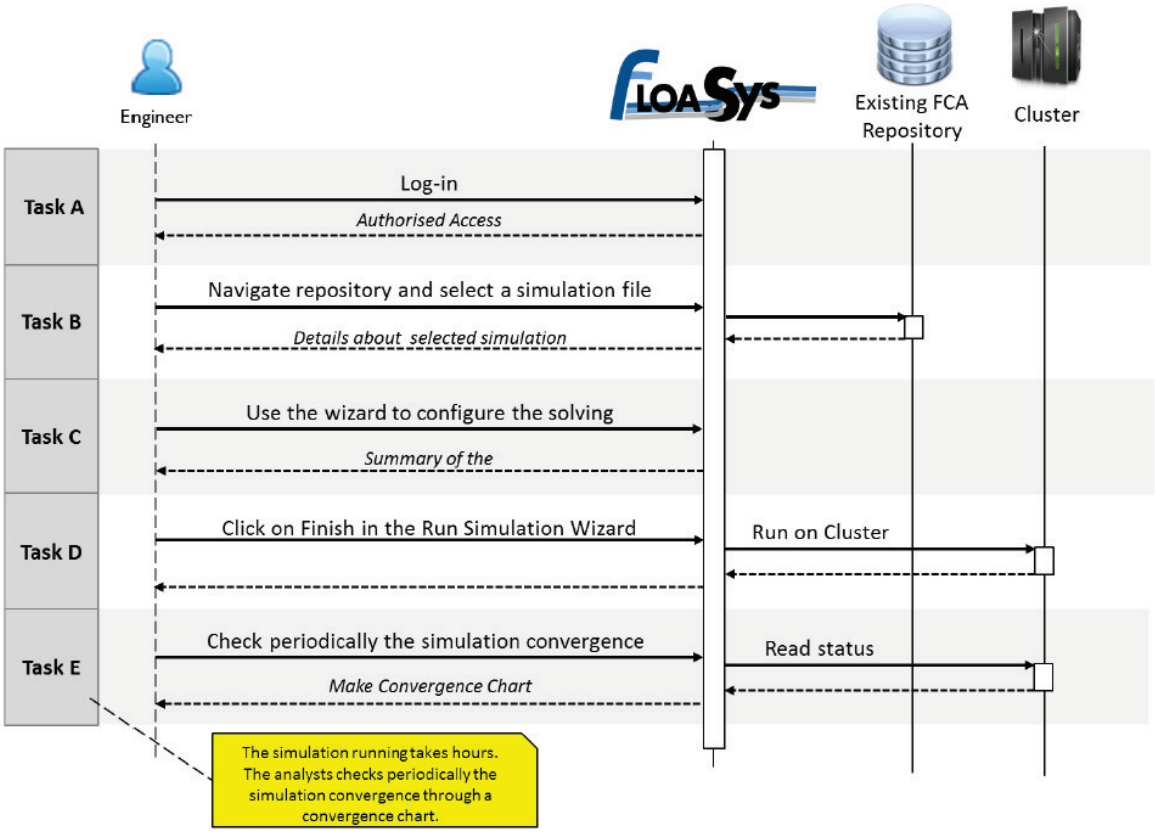


Figure 3.29: Run Simulation Workflow supported by Floasys

In the Task B, Since Floasys manages multiple types of simulator, it recognizes which one is able to manage the selected simulation file. For instance, let's assume for simplicity that Floasys has configured to manage two types of simulator called A (e.g., OpenFoam) and B (e.g., CD-Adapco STAR-CCM+). When the user selects a file of type A, Floasys recognizes this file and knows that

the simulator to use is the type A. This feature has been implemented using the **Chain of Responsibility** design pattern [59].

Figure 3.30 shows a piece of source code within the wizard to run the simulation when the user clicks on finish, at the end of Task D. At line 2, the object file contains the path of the selected simulation file. Lines 5-6 use the Floasys Framework to understand whether in the system there is at least one simulator able to manage that file (resource generically), and if at least one exist then the *simpack* contains a reference to a **Resource Descriptor** with details on the simulation file as well as the simulator able to solve the simulation (line 7). In Figure 3.32a, the SimulationPack class diagram has two subclasses one for each supported simulator, in the example STAR-CCM+ and OpenFoam. Each subclasses can decide how to represent a resource, in according on how the simulator stores the data. STAR-CCM+ uses a single file so it has a file path as instance variable, instead OpenFoam that stores a simulation on multiple file within a directory has a folder path.

```
1  //node is the selected file.
2  File file = ((FSNode) node).getFile();
3
4  //It finds the simulator able to manage the selected file.
5  ISimulationPack simpack =
6      FloasysPlatform.getInstance().createSimulationPack(file);
7  ISimulator simulator = simpack.getSimulator();
8
9  //Options to run the simulation.
10 RunSolverOptions options = new RunSolverOptions();
11 options.NumProc = 32;
12 options.queue = "cfd";
13
14 //It solves the simulation (non blocking).
15 simulator.getRunSolverService().runSolver(simpack, auth, options);
```

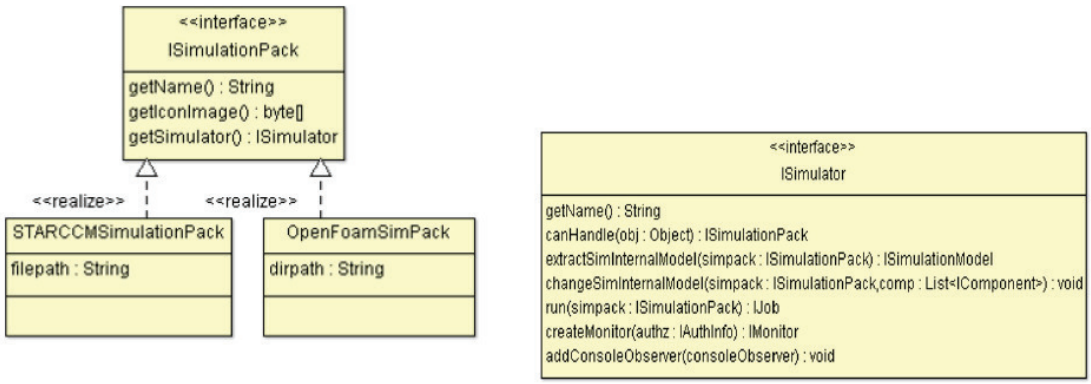
Figure 3.30: A code snippet: how solve a simulation within Floasys.

The example is just to show that Floasys can handle any kind of simulation because for each simulation there will be a relative SimulationPack that will be used by its proper simulator wrapper within Floasys. Lines 10-12 read the parameters to solve the simulation from the wizard GUI and create an object to carry them. Finally, at line 15 the simulator runs the simulation described by the resource descriptor simpack using the parameters within options and the authentication info within auth.

```
1  File file = ...; //file contains a reference to simulation path.
2
3  #foreach ISimulator simulator in simulators
4      ISimulationPack simpack = simulator.canHandle(sim);
5      #if (simpack != null)
6          return simpack;
7  return null;
8
9  //Post-condition: the simulation package instance or null.
```

Figure 3.31: Search a simulator able to handle the simulation file.

Figure 3.31 shows the Floasys framework pseudo-code used to search a simulator able to read (handle) the simulation file selected by the user. Floasys integrates multiple simulators and all their instances are stored within the collection simulator. Each simulator wrapper has a method *canHandle* (Fig. 3.32b) to understand whether the simulator is able to recognize and handle an object. Therefore, the code in Figure 3.31 simply calls this method on all simulators. The first one that returns a reference to a SimulationPack object declares that it is able to handle the object, will accept all requests and will provide the required services. The method *createSimulationPack* (line 6 of Fig. 3.30) calls this piece of code.



(a) SimulationPack. (b) Simulator.

Figure 3.32: Floasys Framework: (a) SimulationPack and (b) Simulator.

3.10.2 Extension by plug-in

Floasys has an extensible and modular architecture based on the plug-in concept inherited from the Eclipse Remote Application Platform. In this way Floasys integrates dynamically the simulators and the front-end modules. Technically, the plug-in architecture has designed around the *extension point* and *extensions* concepts (Fig. 3.33).

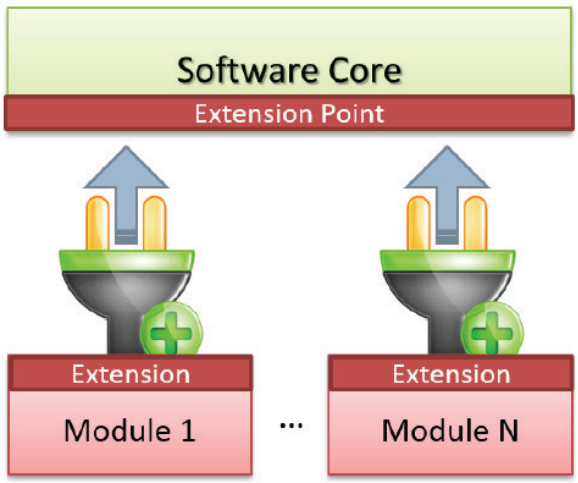


Figure 3.33: Eclipse Extension Point and Extension concepts.

In the plug-in mechanism there are two components: a module to be extended and at least one extender module. The module to be extended defines an extension point like a power socket. It defines formally the rules to extend it, mainly syntactic rules. An extender module defines an extension compliant to the extension point. Definitively, between the extension point and the extension exists a contract. Using this mechanism it is possible to simply copy the extender module within the software plugins folder and the new functionalities will be available within the software. Technically, an Eclipse plug-in is a Java Archive (zip file with jar extension), with a plugin.xml file inside that contains all the information to execute the plug-in. Obviously, Java Runtime is not able directly to read and execute this kind of jar, but the Eclipse platform with its characteristics can read and load this file as a plug-in. In addition to the XML file, a plug-in has also an activator that manages its lifecycle.

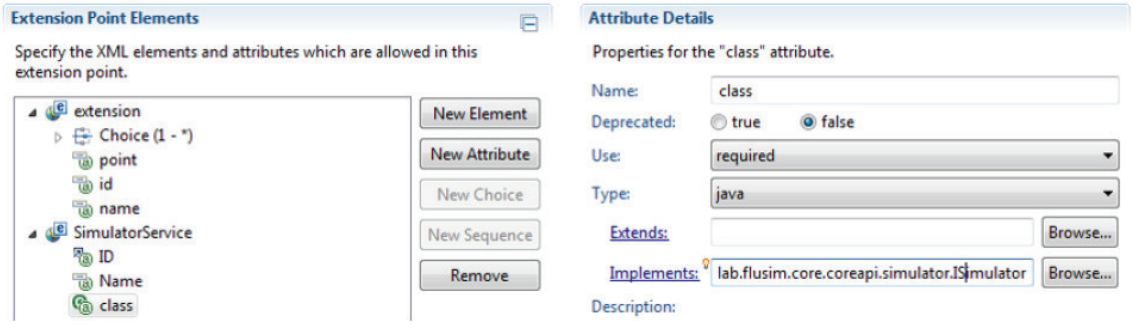


Figure 3.34: How to define an extension point within Eclipse.

Figure 3.34 shows the Eclipse window used to define an extension point. In particular it defines how a simulator can be integrated within Floasys. A simulator plug-in is essentially a Java class that implements the interface *ISimulator* (Fig. 3.32). All data have shown on the left side of Figure 3.34.

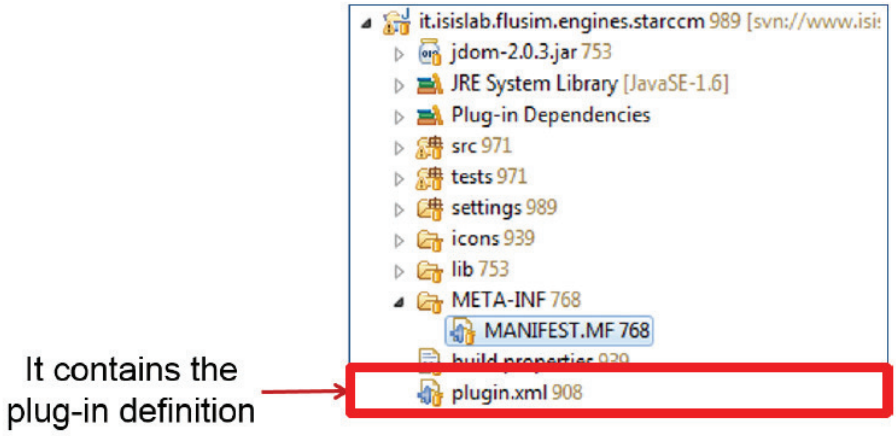


Figure 3.35: Floasys Simulator Wrapper Plug-In Eclipse Project.

Figure 3.35 shows the extender plug-in (the simulator wrapper). The project has a XML file called plugin.xml that defines the wrapper for STAR-CCM+. The extension details are in the following order: a unique ID to identify the plug-in within the platform, a user readable name for the simulator and the Java class that implements the *ISimulator* interface.

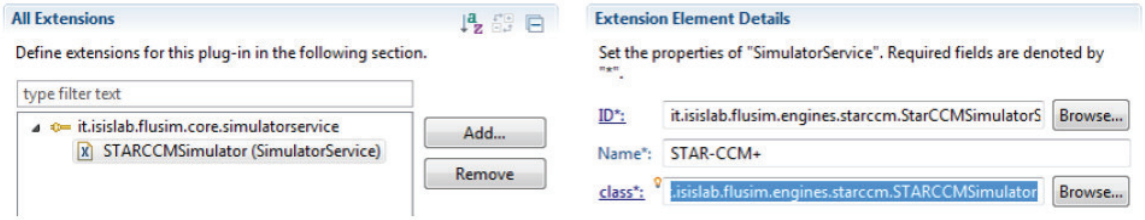


Figure 3.36: Plug-in Simulator Wrapper Extension Definition.

3.11 Remote Application Platform

The Integrated Development Environments (IDE) are software to aid the developers to design, implement and test software systems. The most known IDEs are Microsoft Visual Studio, Apple Xcode, NetBeans and Eclipse. Eclipse IDE is cross-platform and has a plug-in based architecture. Eclipse IDE is gaining even more success because it supports multiple programming languages (e.g., C, C++, PHP, HTML, Javascript, CSS, etc.) and its environment can change dynamically in according to the programming language and the performed tasks. For instance, in the software development lifecycle there are the programming and debugging steps; the Eclipse IDE has two user interfaces configurations called perspectives that contain the needed tool in each phase.

Eclipse is more than an IDE, under the hood there is a full stack platform to develop standalone and web-based application that will have the Eclipse Style. Therefore, applications could have a graphical user interface based on the perspective concept with multiple views. In addition, the developed applications can use all the features already developed for the Eclipse IDE, such as the source code file parsing. One of the interesting feature that can be reused is the plug-in architecture. In this way, the application can be structured in a central core and additional modules to plug in the software. Floasys strongly relies on this concept so that it has a pure plug-in architecture.

The RAP architecture overview is shown in Figure 3.37. It has a client-server architecture. The server is just a servlet container like Apache Tomcat or Jetty. The client is the browser installed on the clients workstations. A RAP Client shows the graphical user interface based on HTML and Javascript. RAP is based on the Half Object Plus Protocol so a widget has two parts: the widget graphical user interface and its logic like the event handlers. These two parts are divided between the client and the server. The client just visualises the widget and gets the user interactions. All the events generated by the user on the clients are managed by the server. For instance, when the user performs a double click on an item of a table, this event is sent to the server that has a listener for it and manage it. Figure 3.38 shows a conceptual view of the Half Object Plus Protocol (HOPP), the circle that represent an object is split in two parts: one is the client object and the other one the server object. These two objects become separate and run on different hardware. A communication layer is placed between the two objects. RAP uses the HTTP protocol to exchange data between the two half objects. In particular the objects exchange messages in JSON format. The main drawback of this protocol is that any event on the client side triggers a message from the client to the server to handle the event and reply. These drawback is evident when the user scrolls a widget like a list of items. In this case any time the user scrolls the list, an event and a message is sent to the server adding a communication delay to exchange the message. A solution to this drawback is to split the object asymmetrically. Instead to send any event to the server, the client can handle the events directly in the browser avoiding the communication with the server. Of course, other events that require the server to be performed trigger a message from the client to the server. Java is the programming language to develop RAP applications, and Java is used to develop both the client and the server side features. One of the drawback of use the same language to program both client and

server, especially as happens in RAP is that sometimes the developer loose the knowledge on which part of the system he is programming. RAP directly decides where a piece of code is executed with the rule that the graphical user interface (everything within the graphical thread) is on the clients and all the codes to manage the events is on the server.

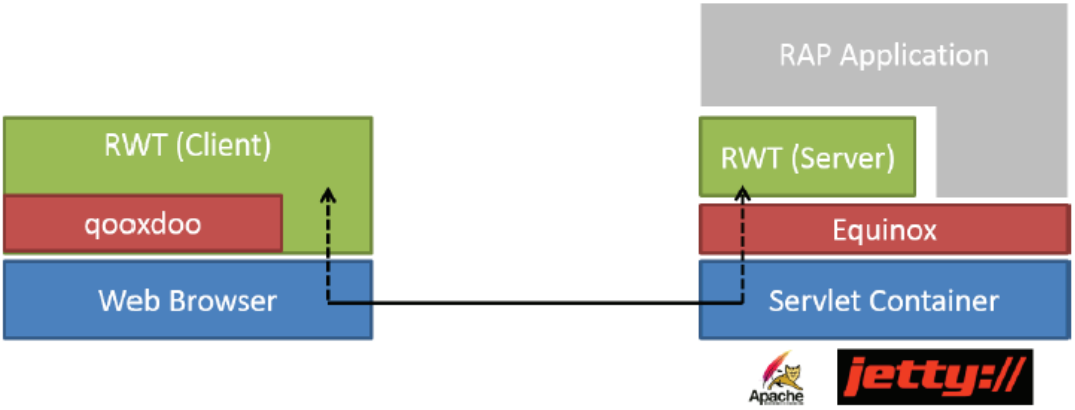


Figure 3.37: Eclipse RAP Client-Server Architecture.

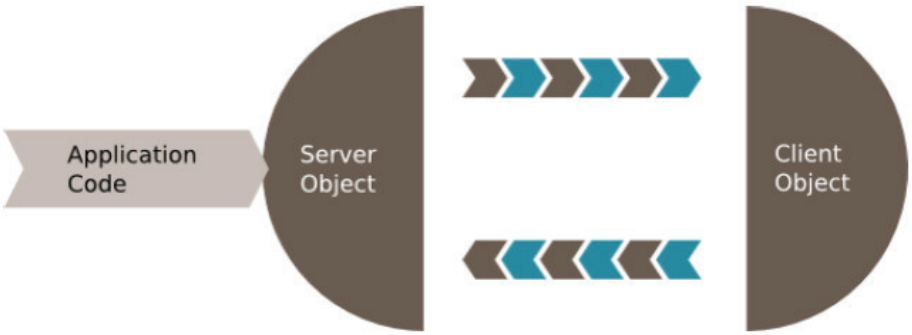


Figure 3.38: Half Object Plus Protocol in RAP.

3.12 Visualisation Tool

Nowadays, industries and researchers extensively run simulations and experiments to design their products. In the automotive, industrial equipment, high-tech, aerospace and defence sectors [63], industries perform computer numerical simulations to design their product facing time-to-market, high quality and cost down pressures [63]. For example, automotive industries use Computational Fluid Dynamic (CFD) simulations to design the external vehicle aerodynamics or the internal air-conditioning. Another example comes from the engine design: researchers and industries have real engine test-beds that run for hours collecting sensor data like pressure, temperature and torque forces.

Simulation repositories usually store huge amounts of data for years. For instance, in large manufactures like Fiat Chrysler Automobiles, each analyst performs at least one hundred simulations per year [64], and there are many analysts working over years. This has generated a

large, valuable repository of assets. In addition, analysts typically deal with simulations that are at least ten gigabytes each [64]. This gives an idea of the large quantity of data to manage within these repositories and the difficulty in having a clear idea of what they contain. Simulation Analysts, as well as Experiment Analysts, need to clean, analyse and compare the collected results as well as get insight into the data repository. Sometimes, specific phenomenons need to be understood. For instance, if a particular event in an engine experiment run occurs sporadically, found through the analysis of huge amounts of experimental data, then the analyst need to extract the input conditions for which such an event occurs (e.g., for which pressure values). For this reason there is a demand for software platforms able to collect, centralise, and get insight into information in a data repository, as well as to analyse and share results [10].

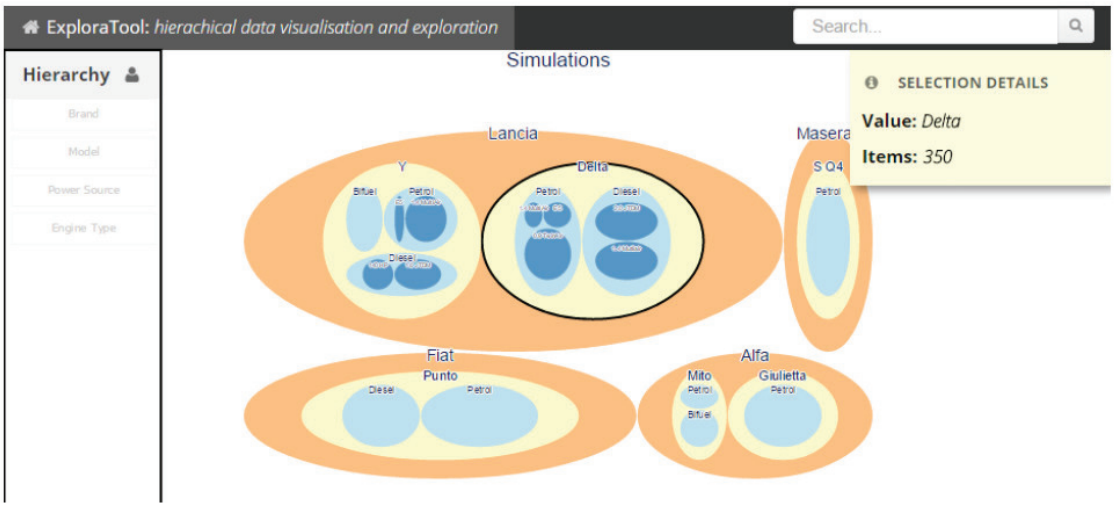


Figure 3.39: The ExploraTool's Graphical User Interface. It shows an overview of the simulation repository through an initial hierarchy made by the following simulations' attributes: brand, project model, power source and engine type

I identified the following three main requirements: (1) data collection, centralisation [63], and sharing [64] (2) data heterogeneity management, and (3) repository visualisation and exploration. The requirements one and two have been extensively described in the previous chapters, here this chapter will focus on the simulation repository visualisation and exploration.
This chapter focuses on the visualisation, exploration, and query of large repository of simulations. The idea is to provide a graphical tool called ExploraTool to (1) get an overview of the repository content, (2) navigate the repository of simulations based on their properties, and (3) select and extract a set of simulations in order to compare their performance. The tool is actually usable for generic data exploration, thereby being usable to also explore repositories of experimental data, or any other big data sets.

3.13 Related Works

The visualisation of large datasets has become really important because the classical list based widgets are not able to manage the large number of items, and also because it is practically impossible to show all the data available within a dataset. In this context, the 2D space-filling visualisation techniques aim to exploit all the available screen-space supporting the overview of the datasets, the opportunity to navigate the dataset and get more details on request. Generically speaking, the 2D space-filling approaches divide the available screen space recursively using a basic shape (e.g., rectangle, circle). In this way parent-child relationships are represented as nested shapes, and sibling nodes are represented as closest shapes at same depth.

Treemap was introduced by Shneiderman during 1990 to have a compact file system visualisation and be able to identify at a glance the directories that take up the most of the space on the hard drive. Then, treemap [65] has been extensively used to present intrinsically hierarchical data, providing an overview of an entire dataset at a glance. In treemap, every node in the hierarchy is represented as a rectangle with an area proportional to the node size. Parent-child nodes are represented as nested rectangles. Usually the navigation within the hierarchy is based on a drill-down with a left mouse click to go down in the hierarchy and a roll-up with a right mouse click to go up in the hierarchy. Over years, the treemap visualisation approach has been used to visualize different hierarchical data, such as inherently hierarchical organisation structures [66], file systems [67], Usenet newsgroup [68] and so on. Well-known treemap drawbacks are the hierarchy discernment [69] and the fact that *the position of the mouse pointer designates an entire branch of the tree* [70] because *each point belongs to a single leaf node but also to all its ancestors* [70]. Of course, one of their advantages is the use of the all available 2D space.

Ellimap [69] is another type of 2D space-filling visualisation approach. It uses ellipses instead of rectangles to represent the nodes. In this way, there is always space between ellipses, both nested ellipses and adjacent ellipses (i.e., sibling nodes in the hierarchy). According to Otjacques at al. [71], the use of ellipses with their extra space improves the hierarchy discernment compared to the visualisation based on rectangles.

ExploraTool exploits the ellimap visualisation technique to explore large repository of simulations within Fiat Chrysler Automobiles (FCA). Until now, the ellimap has always been used coupled with other classical visualisation widgets like tree widget [69]. Here, this chapter explores the repository of simulations directly through the ellimap, integrating a vertical navigation bar to track the user position in the hierarchy during the navigation. In addition, this work exploits the natural extra space between the ellipses in order to provide a hierarchy navigation facility in which the user points directly to the target shape and interacts with the left mouse click.

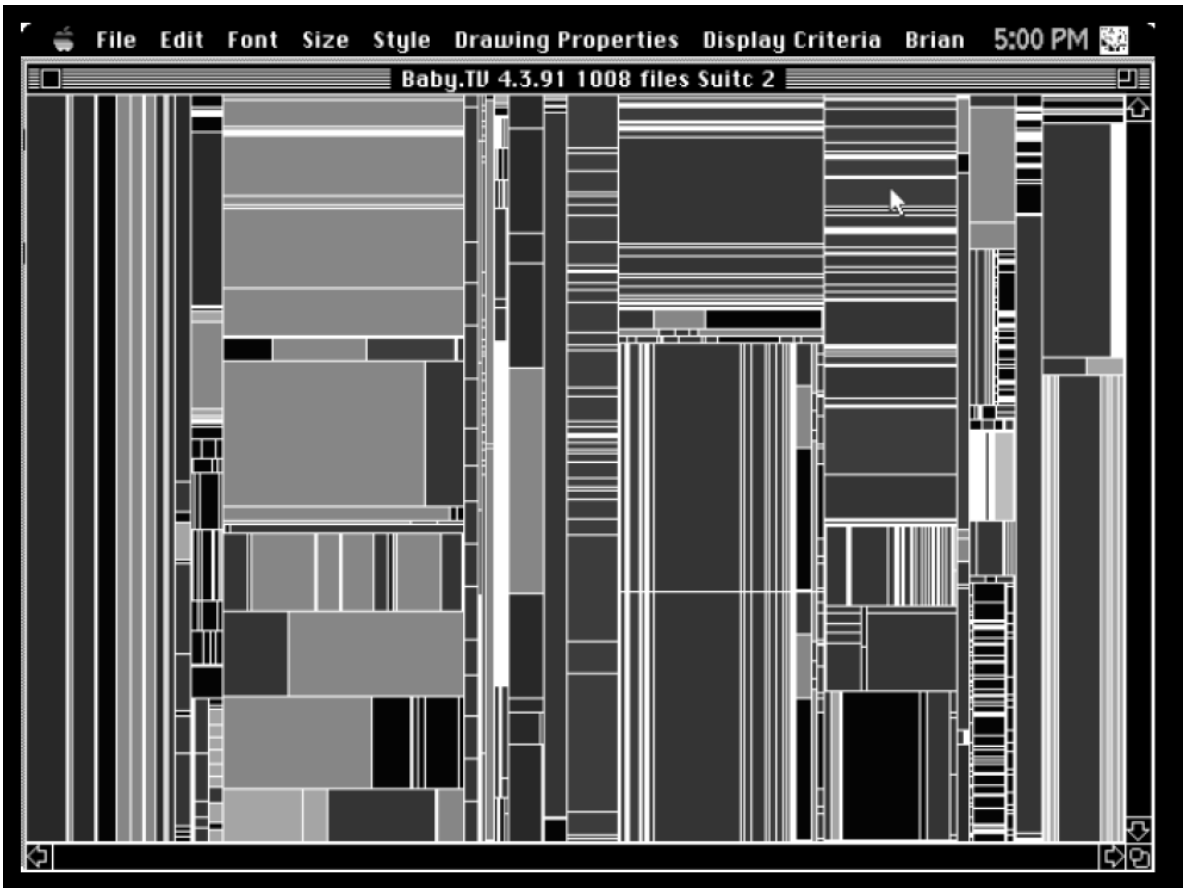


Figure 3.40: Original treemap visualisation introduced by Shneiderman during 1990.

3.14 ExploraTool features

This section describes ExploraTool and its features. Instead of starting from scratch with an empty screen without results, the tool shows an initial overview of the dataset filling all the 2D screen available space. Starting from this initial view, the user can navigate the simulation repository through an hierarchical structure made by nested groups of simulations. The tool's graphical user interface (Fig. 3.39) has a central view to show graphically the simulations available within the repository. The tool shows data using the ellimap [69] visualisation technique, a 2D space-filling approach that uses ellipses as basic shapes to represent sets of simulations. As shown in Figure 3.39, the external white space is the universe that represent the set of all simulations within the repository. The universe of simulations is further divided in subsets represented as ellipses. Each ellipse area is proportional to the number of items that it represents. The ExploraTool shows an initial overview of the dataset displaying the simulations by brand, project model, power source and engine type. This default initial sequence of attributes is based on the feedback provided by analysts in Fiat Chrysler Automobiles [64].

The user can have additional details on each group of simulations (ellipse) just by hovering the mouse cursor over it. The tool shows the additional information, such as, the number of items in a yellow box on the top-right (see Figure 3.39). This space can be used in the future to provide aggregated statistics about the shown group of simulations. The user can navigate the hierarchy

through an *in-depth navigation* based on the drill-down and roll-up operations. On the left, the tool has a vertical navigation hierarchy bar that has multiple aims: (1) it gives an overview of the hierarchy, (2) it shows the current depth during the simulation repository navigation supporting the user orientation [72], and (3) it allows hierarchy rearrangement by swapping the levels.

The tool shows exactly *r* levels of the hierarchy. Actually, the default value for this parameter *r* is decided at configuration time and it can be changed via the user preference functions. Of course, the trade-off is between the amount of data categories displayed on the screen-space and the computational efficiency to extract the relevant hierarchy from the repository of simulations.

3.14.1 Data Exploration: in-depth navigation

The user can further explore the simulation repository through the *in-depth* navigation [70] based on two basic operations: drill-down and roll-up. **Drill-down** occurs when a user has identified a potential interesting group of simulations and he/she wishes to explore further details of this group, and so he/she clicks on an ellipse to obtain more details. Every time the user drills down in the hierarchy by one level, ExploraTool loads further data showing more nested ellipses. ExploraTool shows multiple nested ellipses, so the user can drill-down one level at time or multiple-levels in one step just clicking on the most internal nested ellipses. **Roll-up** is the operation opposite to the drill-down.

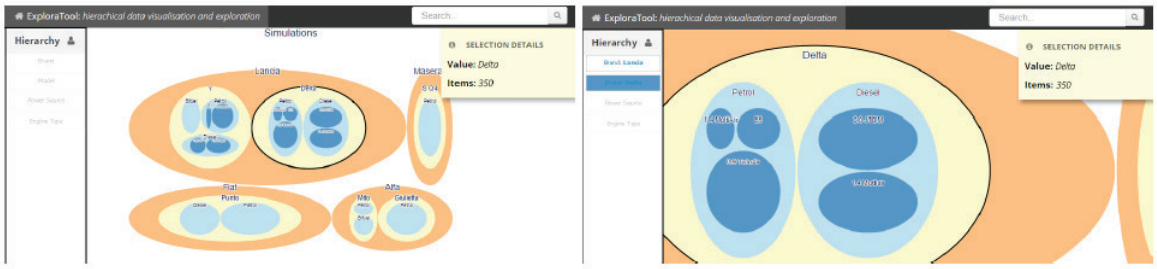


Figure 3.41: The ExploraTool (on the left) shows an initial overview of the repository with all the simulations progressively grouped by brand, vehicle model, power source and engine type, as shown by the vertical navigation bar. In addition, the analyst has moved the mouse pointer on the project *Delta* highlighting the relevant ellipse contour and showing additional details within the *selection details* yellow tooltip box. In order to focus on this group, the user can drill-down by directly clicking on the ellipse with the label Delta. The ExploraTool smoothly enlarges the selected group (right side of the figure) rendering a fast transition. When the user desires to go back to see less details, he can directly click on the universe white space to perform a roll-up operation returning to the initial view shown on the left.

When the user wants to have a global dataset view he/she goes up in the hierarchy clicking on the container ellipse. Every time the user drills down in the hierarchy, he/she is effectively performing a refinement of the query, filtering all of the simulations in the repository.

All the operations provided by the ExploraTool rely on the direct manipulation [73] principle introduced by Shniderman. It concerns the direct interaction and manipulation of the rendered objects. The use of ellipses as basic shapes guarantee that there will be always space between sibling ellipses at same level and among nested ellipses. In this way every operation performed by the user involves exactly the target shape. For instance, in order to drill down in the hierarchy, the user points and clicks exactly on the nested ellipse. In order to roll-up the user points and clicks exactly on the parent shape utilising the space between the parent and child ellipses (Figure 3.41) which is always present. It is not the same for other 2D space-filling techniques. For instance, in the

treemap visualisation technique both nested rectangles and adjacent rectangles have no space among them, so *the position of the mouse pointer designates a branch of the tree* [70] because *each point belongs to a single leaf node but also to all its ancestors* [70].
Finally, in the ExploraTool, also to obtain the list of simulations within a specific ellipse the user can click directly on the target ellipse.

3.14.2 Shapes and Colours

The use of the colours is really important within a visualisation tool. ExploraTool uses the colours described in the following work [74]. Table 3.1 lists the colours used within the ExploraTool and visible through the Figure 3.39 and Figure 3.41.

Colour	HTML Colour
	E04A4D
	FEC083
	FFFFCD
	BDE1EE
	5997C6

Table 3.1: Colours used to draw the ellipses within the ExploraTool.

3.15 ExploraTool Software Architecture

This section describes the ExploraTool architecture and the technologies used for its implementation. The tool is based on a Client/Server architecture (Fig. 3.42). Nevertheless in the industrial contexts for confidentiality reasons software systems usually are used within the industries boundaries (Intranet-based), the overall architecture is designed using standard protocols to work properly both on the Intranet and Internet.

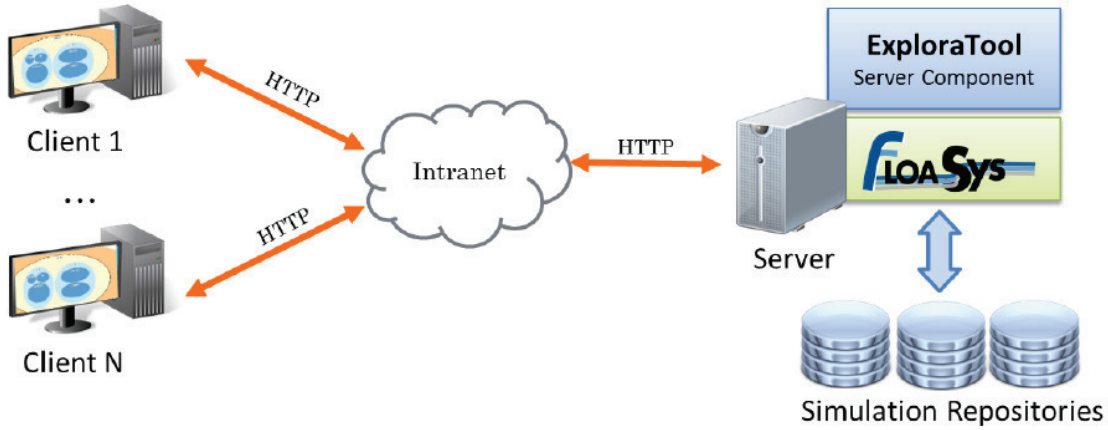


Figure 3.42: ExploraTool prototype client/server architecture.

In order to explore the repository, analysts can just open one of the web-browsers (e.g., Mozilla Firefox) installed on their workstations targeting to a specific Intranet URL. This allows zero-configuration on the client-side.

On the server-side there are one or multiple simulation repositories. The Floasys Framework [10] reads the data from the simulation repository, transforms them in open format and indexes them to improve their retrieval. ExploraTool on the server-side uses the Floasys Framework API to retrieve the simulations stored within the repository. The overall process with detailed steps has been depicted in Figure 3.43. The simulation data in tabular format are the input for the Hierarchy generation phase performed by the algorithm. The output is a tree data structure converted in JSON text format and sent to the Web-Browser. The browser gets the hierarchy structure, generates an ellipse for each node in the hierarchy and packs all the ellipses.

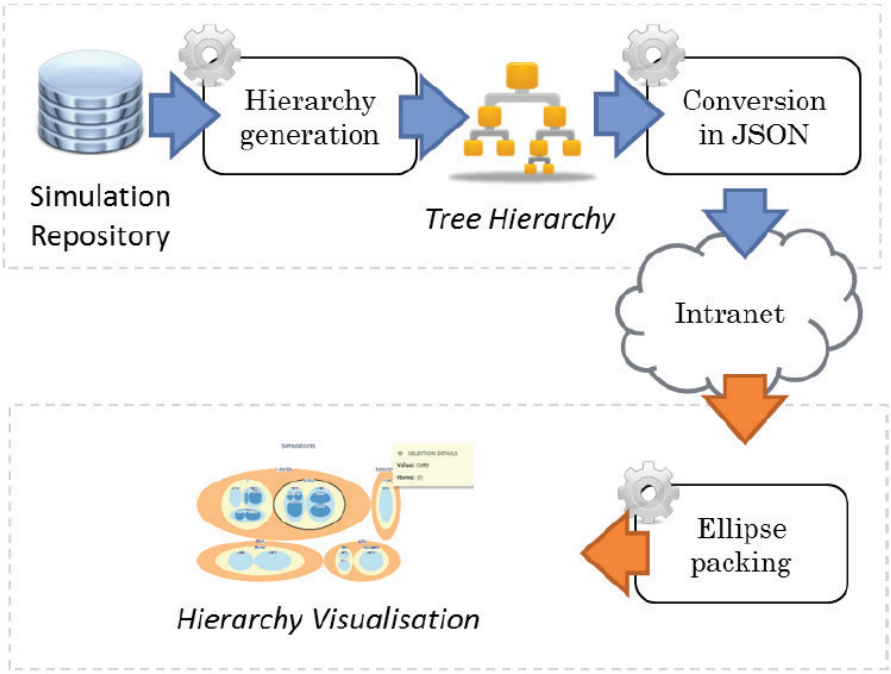


Figure 3.43: Pipeline of transformation from the simulation repository to the visualization on the client Web-browser.

From technological point of view, ExploraTool leverage from mainstream technologies. Clients exchange data with the server in JSON text format [48, 49] using standard Web protocols (e.g., HTTP). Clients are implemented using the open source JavaScript library *D3 Data-Driven Documents*1 [75] and SVG. The server has been implemented using Java.

3.16 Conclusions and Future Works

This chapter has introduced two main typical Floasys workflows and described some of the Floasys Functionalities showing its main screenshots. The collaborative functionalities are generic and can be applied to any other field. For instance, the idea to centralize data, tag them adding metadata over data and provide a search tool as well as the data sharing exchanging the URLs can be used for simulation data, experimental data and any other data. The Floasys engineering functionalities are specific of the CFD engineering sector, but they can be used in any other sector that uses the CFD

simulations to design their products, such as the aeronautics, rail and naval sectors. Finally, the idea to generate automatic documentation from the data repository is a key feature for the industries because they standardise how the engineers work with less effort to make the outcomes. And it's an important thing considering that companies assess the value of a technology based on the saved time, and saved money or earn money; and not how cool is a technology.

Floasys's architecture is extensible so that it supports the adding of new services over collected data. A lot of research work can be done for the creation of new services. For instance, one useful service is the comparison among experiments in terms of mathematical functions to identify the ones that have the same trend or the ones with the same physical phenomena. This service involves the study of time-series algorithms.

In the final part, the chapter described a tool called ExploraTool to visualise, explore and query large repositories of simulations. Large industries like FCA have large repository of simulation data and they must be sure that analysts have access to previous generated data. ExploraTool provides an overview of the repository content fostering its exploration selecting the key attributes to limit the space of results to find previous simulations. In addition, ExploraTool is immediately useful to answer questions like *how many simulations we performed for the vehicle X?*, and *why for the vehicle Y with the engine type Z there are few simulations?* ExploraTool gives a further advantage for the technical managers who can periodically check the working in progress on a specific vehicle model. The idea behind the tool is generic and can be easily used with the repository of experiments as well as other type of big data sets. In order to do this, it is necessary to identify the common and interesting data categories, and build the relative hierarchy that ExploraTool will render.

As future works on the ExploraTool, it is important to improve the layout algorithm to avoid thin ellipses, thereby improving the overall visualisation aesthetic. Of course, the residual space among nested ellipses can be reduced, but this could impact upon user hierarchy perception and discernment. In addition, an evaluation study is essential to analyse the tool usability and the user satisfaction when interacting with it, by utilizing a well-known questionnaire [81, 82]. Furthermore, will be interesting to generalise the tool and use it on a generic repository like a catalogue of products and compare how users will perform with it as compared to using different types of visualisation techniques, like a classical list of results, Treemap, FacetMap [77], etc. Within the industrial context an interesting issue to explore is the data authorisation problem, where a user may only have access to a specific subset of simulations within the repository.

Chapter 4

Floasys Extension and Fields of Application

Contents

This chapter illustrates the Floasys multi-disciplinary applications, not limited just to the Computational Fluid Dynamics field. It begins with a brief description of other filed of applications: Structural, Thermal, Multibody, Fatigue and Coupled Analysis. One of the main goal of the present work is to guarantee collaboration in these complex scenario.

In the 2nd paragraph it's described the way to extend Floasys to other Industrial Environment (Aeronautical, Rail, Naval, etc.) through the plug-ins architecture.

4.1 Multi-Disciplinary Analysis

A complex industrial project involves several departments that have to optimize the performances to achieve the prefixed target. Each department has its rules and best practices to reach this goal.
It follows a brief description of the main areas of analysis with a focus on the collaborative topics.

4.1.1 Structural Analysis

Structural analysis is the determination of the effects of loads on physical structures and their components. Structures subject to this type of analysis include all that must withstand loads, such as buildings, bridges, vehicles, machinery, furniture, attire, soil strata, prostheses and biological tissue. Structural analysis employs the fields of applied mechanics, materials science and applied mathematics to compute a structure's deformations, internal forces, stresses, support reactions, accelerations, and stability. The results of the analysis are used to verify a structure's fitness for use, often precluding physical tests. Structural analysis is thus a key part of the engineering design of structures.

To perform an accurate analysis a structural engineer must determine such information as structural loads, geometry, support conditions, and materials properties. The results of such an analysis typically include support reactions, stresses and displacements. This information is then compared to criteria that indicate the conditions of failure. Advanced structural analysis may examine dynamic response, stability and non-linear behavior. There are three approaches to the analysis: the mechanics of materials approach (also known as strength of materials), the elasticity theory approach (which is actually a special case of the more general field of continuum mechanics), and the finite element approach. The first two make use of analytical formulations which apply mostly to simple linear elastic models, lead to closed-form solutions, and can often be solved by hand. The finite element approach is actually a numerical method for solving differential equations generated by theories of mechanics such as elasticity theory and strength of materials. However, the finite-element method depends heavily on the processing power of computers and is more applicable to structures of arbitrary size and complexity.

Regardless of approach, the formulation is based on the same three fundamental relations: equilibrium, constitutive, and compatibility. The solutions are approximate when any of these relations are only approximately satisfied, or only an approximation of reality.

Each method has noteworthy limitations. The method of mechanics of materials is limited to very simple structural elements under relatively simple loading conditions. The structural elements and loading conditions allowed, however, are sufficient to solve many useful engineering problems. The theory of elasticity allows the solution of structural elements of general geometry under general loading conditions, in principle. Analytical solution, however, is limited to relatively simple cases. The solution of elasticity problems also requires the solution of a system of partial differential equations, which is considerably more mathematically demanding than the solution of mechanics of materials problems, which require at most the solution of an ordinary differential equation. The finite element method is perhaps the most restrictive and most useful at the same time. This method itself relies upon other structural theories (such as the other two discussed here) for equations to solve. It does, however, make it generally possible to solve these equations, even with highly complex geometry and loading conditions, with the restriction that there is always some numerical error. Effective and reliable use of this method requires a solid understanding of its limitations.

It is common practice to use approximate solutions of differential equations as the basis for structural analysis. This is usually done using numerical approximation techniques. The most commonly used numerical approximation in structural analysis is the Finite Element Method.

The finite element method approximates a structure as an assembly of elements or components with various forms of connection between them and each element of which has an associated stiffness. Thus, a continuous system such as a plate or shell is modeled as a discrete system with a finite number of elements interconnected at finite number of nodes and the overall stiffness is the result of the addition of the stiffness of the various elements. The behaviour of individual elements is characterized by the element's stiffness (or flexibility) relation. The assemblage of the various

stiffness's into a master stiffness matrix that represents the entire structure leads to the system's stiffness or flexibility relation. To establish the stiffness (or flexibility) of a particular element, we can use the mechanics of materials approach for simple one-dimensional bar elements, and the elasticity approach for more complex two- and three-dimensional elements. The analytical and computational development are best effected throughout by means of matrix algebra, solving partial differential equations.

Early applications of matrix methods were applied to articulated frameworks with truss, beam and column elements; later and more advanced matrix methods, referred to as "finite element analysis", model an entire structure with one-, two-, and three-dimensional elements and can be used for articulated systems together with continuous systems such as a pressure vessel, plates, shells, and three-dimensional solids. Commercial computer software for structural analysis typically uses matrix finite-element analysis, which can be further classified into two main approaches: the displacement or stiffness method and the force or flexibility method. The stiffness method is the most popular by far thanks to its ease of implementation as well as of formulation for advanced applications. The finite-element technology is now sophisticated enough to handle just about any system as long as sufficient computing power is available. Its applicability includes, but is not limited to, linear and non-linear analysis, solid and fluid interactions, materials that are isotropic, orthotropic, or anisotropic, and external effects that are static, dynamic, and environmental factors. This, however, does not imply that the computed solution will automatically be reliable because much depends on the model and the reliability of the data input.

4.1.2 Thermal Analysis

Thermal analysis is a branch of materials science where the properties of materials are studied as they change with temperature. Several methods are commonly used – these are distinguished from one another by the property which is measured:

- Dielectric thermal analysis (DEA): dielectric permittivity and loss factor
- Thermal Analysis (DTA): temperature difference versus temperature or time
- Differential Scanning Calorimetry (DSC): heat flow changes versus temperature or time
- Dilatometry (DIL): volume changes with temperature change
- Dynamic Mechanical Analysis (DMA or DMTA) : measures storage modulus (stiffness) and loss modulus (damping) versus temperature, time and frequency
- Evolved Gas Analysis (EGA) : analysis of gases evolved during heating of a material, usually decomposition products
- Laser flash analysis (LFA): thermal diffusivity and thermal conductivity
- Thermogravimetric Analysis (TGA): mass change versus temperature or time
- Thermomechanical analysis (TMA): dimensional changes versus temperature or time
- Thermo-optical analysis (TOA): optical properties
- Derivatography: A complex method in thermal analysis

Simultaneous Thermal Analysis (STA) generally refers to the simultaneous application of Thermogravimetry (TGA) and differential scanning calorimetry (DSC) to one and the same sample in a single instrument. The test conditions are perfectly identical for the TGA and DSC signals (same atmosphere, gas flow rate, vapor pressure of the sample, heating rate, thermal contact to the sample crucible and sensor, radiation effect, etc.). The information gathered can even be enhanced

by coupling the STA instrument to an Evolved Gas Analyzer (EGA) like Fourier transform infrared spectroscopy (FTIR) or mass spectrometry (MS).

Other, less common, methods measure the sound or light emission from a sample, or the electrical discharge from a dielectric material, or the mechanical relaxation in a stressed specimen. The essence of all these techniques is that the sample's response is recorded as a function of temperature (and time). It is usual to control the temperature in a predetermined way - either by a continuous increase or decrease in temperature at a constant rate (linear heating/cooling) or by carrying out a series of determinations at different temperatures (stepwise isothermal measurements). More advanced temperature profiles have been developed which use an oscillating (usually sine or square wave) heating rate (Modulated Temperature Thermal Analysis) or modify the heating rate in response to changes in the system's properties (Sample Controlled Thermal Analysis).

In addition to controlling the temperature of the sample, it is also important to control its environment (e.g. atmosphere). Measurements may be carried out in air or under an inert gas (e.g. nitrogen or helium). Reducing or reactive atmospheres have also been used and measurements are even carried out with the sample surrounded by water or other liquids. Inverse gas chromatography is a technique which studies the interaction of gases and vapours with a surface - measurements are often made at different temperatures so that these experiments can be considered to come under the auspices of Thermal Analysis. Atomic force microscopy uses a fine stylus to map the topography and mechanical properties of surfaces to high spatial resolution. By controlling the temperature of the heated tip and/or the sample a form of spatially resolved thermal analysis can be carried out.

Thermal analysis is also often used as a term for the study of heat transfer through structures. Many of the basic engineering data for modelling such systems comes from measurements of heat capacity and thermal conductivity.

4.1.3 Multibody Analysis

Multibody analysis is the study of the dynamic behavior of interconnected rigid or flexible bodies, each of which may undergo large translational and rotational displacements.

The systematic treatment of the dynamic behavior of interconnected bodies has led to a large number of important multibody formalisms in the field of mechanics. The simplest bodies or elements of a multibody system were treated by Newton (free particle) and Euler (rigid body). Euler introduced reaction forces between bodies. Later, a series of formalisms were derived, only to mention Lagrange's formalisms based on minimal coordinates and a second formulation that introduces constraints.

Basically, the motion of bodies is described by their kinematic behavior. The dynamic behavior results from the equilibrium of applied forces and the rate of change of momentum. Nowadays, the term multibody system is related to a large number of engineering fields of research, especially in robotics and vehicle dynamics. As an important feature, multibody system formalisms usually offer an algorithmic, computer-aided way to model, analyze, simulate and optimize the arbitrary motion of possibly thousands of interconnected bodies.

While single bodies or parts of a mechanical system are studied in detail with finite element methods, the behavior of the whole multibody system is usually studied with multibody system methods within the following areas:

- Aerospace engineering (helicopter, landing gears, behavior of machines under different gravity conditions)
- Biomechanics
- Combustion engine, gears and transmissions, chain drive, belt drive

- Dynamic simulation, Vehicle simulation (vehicle dynamics, rapid prototyping of vehicles, improvement of stability, comfort optimization, improvement of efficiency, ...)
- Hoist, conveyor, paper mill
- Military applications
- Particle simulation (granular media, sand, molecules)
- Physics engine
- Robotics

4.1.4 Fatigue Analysis

In materials science, fatigue is the weakening of a material caused by repeatedly applied loads. It is the progressive and localized structural damage that occurs when a material is subjected to cyclic loading. The nominal maximum stress values that cause such damage may be much less than the strength of the material typically quoted as the ultimate tensile stress limit, or the yield stress limit. Fatigue occurs when a material is subjected to repeated loading and unloading. If the loads are above a certain threshold, microscopic cracks will begin to form at the stress concentrators such as the surface, persistent slip bands (PSBs), and grain interfaces. Eventually a crack will reach a critical size, the crack will propagate suddenly, and the structure will fracture. The shape of the structure will significantly affect the fatigue life; square holes or sharp corners will lead to elevated local stresses where fatigue cracks can initiate. Round holes and smooth transitions or fillets will therefore increase the fatigue strength of the structure.

4.1.5 Multiphysics Analysis

Multiphysics treats simulations that involve multiple physical models or multiple simultaneous physical phenomena. For example, combining chemical kinetics and fluid mechanics or combining finite elements with molecular dynamics. Multiphysics typically involves solving coupled systems of partial differential equations. Many physical simulations involve coupled systems, such as electric and magnetic fields for electromagnetism, pressure and velocity for sound, or the real and the imaginary part of the quantum mechanical wave function. Another case is the mean field approximation for the electronic structure of atoms, where the electric field and the electron wave functions are coupled.

Single Discretization Methods mainly rely on the Finite Element Method or similar commonplace numerical methods for simulating coupled physics: thermal stress, electromechanical interaction, fluid structure interaction (FSI), fluid flow with heat transport and chemical reactions, electromagnetic fluids (magnetohydrodynamics or plasma), electromagnetically induced heating. In many cases, to get accurate results, it is important to include mutual dependencies where the material properties significant for one field (such as the electric field) vary with the value of another field (such as temperature) and vice versa.

In Multiple Discretization Methods each subset of partial differential equations has different mathematical behavior, for example when compressible fluid flow is coupled with structural analysis or heat transfer. To perform an optimal simulation in those cases, a different discretization procedure must be applied to each subset. For example, the compressible flow is discretized with a finite volume method and the conjugate heat transfer with a finite element analysis. Another example is the use of electromagnetic or electrostatic Particle-in-cell (PIC, EMPIC, ESPIC)

methods combined with Direct simulation Monte Carlo, where the particles may interact with an electromagetic (EM) field or other fields, with each other, and with fluids evolved by finite volume or other methods. The particles interact with the EM fields through the charges and currents they create and by being accelerated by the EM field. Particles collide with each other, and they collide with fluids.

4.1.6 Collaborative Use Cases

In this complex scenario Floasys comes in handy. Obviously different roles in the company have different needs; in every engineering department we have at least these roles:

- CAE Analyst
- Performance Engineering
- Project Manager

The CAE Analyst has the task to do the CAE Analysis, the Performance Engineering is responsible for a specific project performance and the Project Manager is responsible for the whole project. So the Multi-Disciplinary issues presented in this chapter are prevalently oriented to the Project Manager view of the project. In Figure 4.1 is illustrate a typical hierarchy in the company organization.

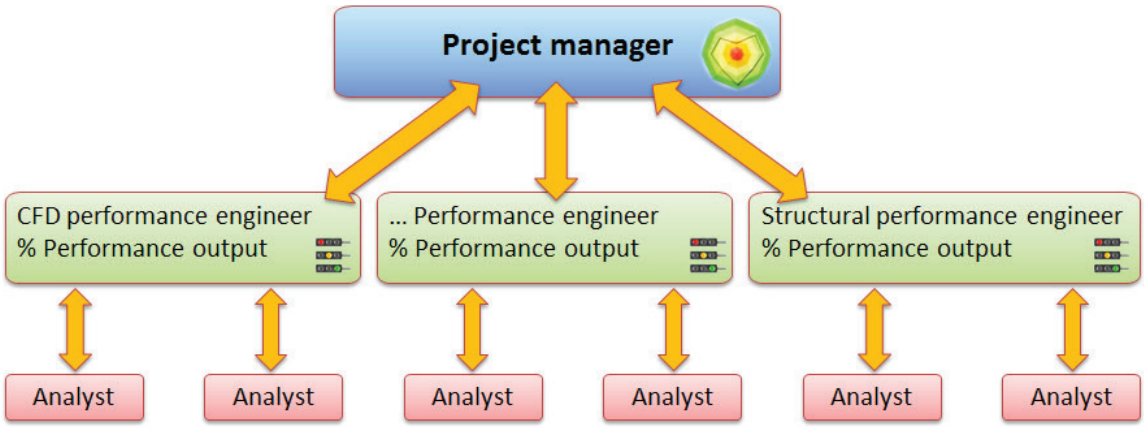
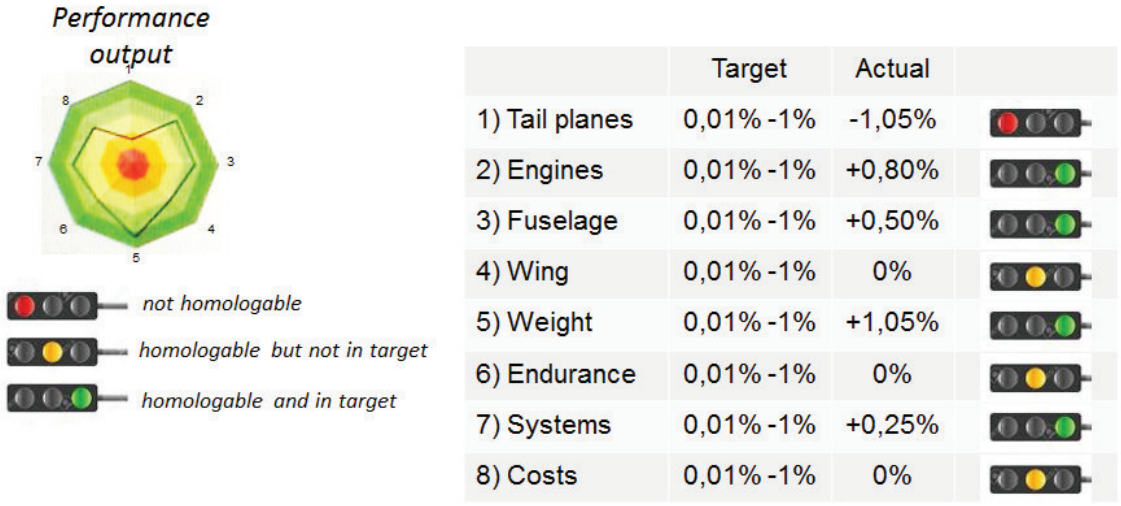


Figure 4.1: Hierarchy in the performance target achieving.

To constantly monitor project performances is best-practice to make use of a Radar chart as depicted in Figure 4.2.



	Target	Actual	
1) Tail planes	0,01% -1%	-1,05%	
2) Engines	0,01% -1%	+0,80%	
3) Fuselage	0,01% -1%	+0,50%	
4) Wing	0,01% -1%	0%	
5) Weight	0,01% -1%	+1,05%	
6) Endurance	0,01% -1%	0%	
7) Systems	0,01% -1%	+0,25%	
8) Costs	0,01% -1%	0%	

Figure 4.2: Radar chart to monitor Project status.

This tool gives a fast overview of the status of the project and helps companies to recognize critical issues and put on the track of recovery actions.
So Floasys guarantees collaborative needs, around the various simulation environments, through all the services described in the previous chapters as shown in the Use Case in Figure 4.3.

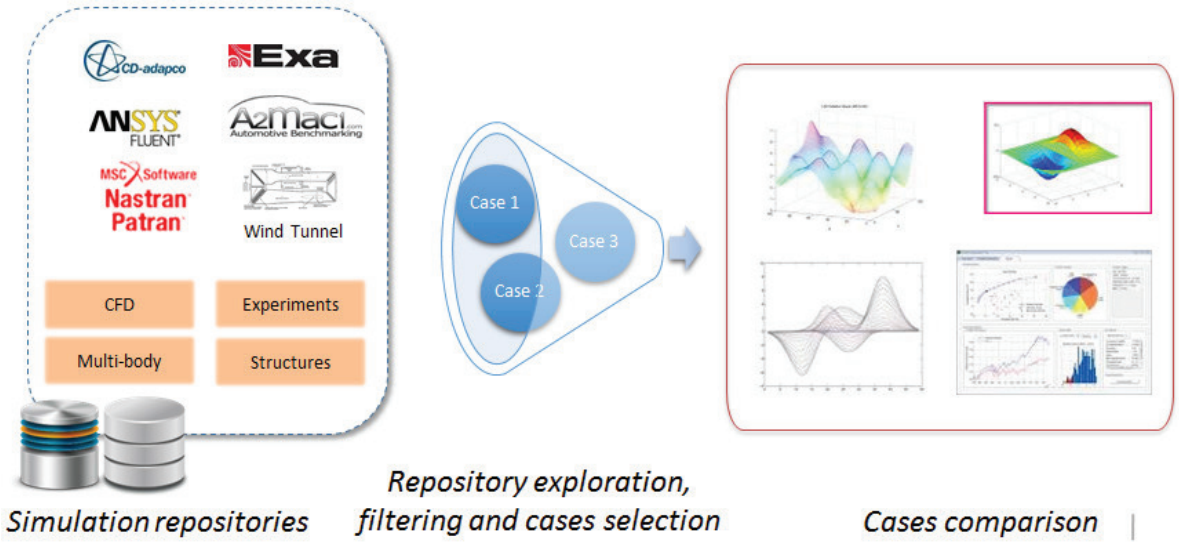


Figure 4.3: Typical Use Case involving several CAE Environments.

4.2 Intersectorial Extension through Industrial Environment plug-ins

The extensibility features of Floasys through plug-ins architecture gives opportunity to adapt the framework to several industrial sectors (rail, naval, automotive, aeronautical, etc.). Indeed each

sector has its own regulations, configurations and data sets. It's enough, in terms of implementation, to add another layer to the Floasys Architecture as shown in Figure 4.4.

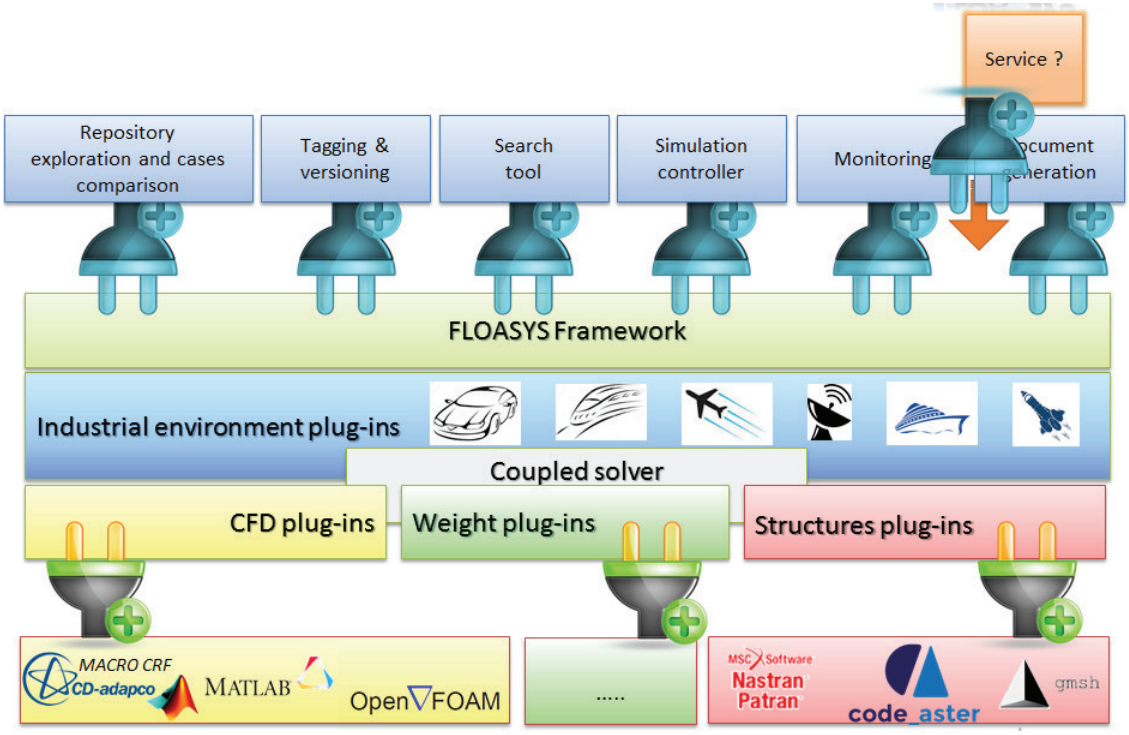


Figure 4.4: Floasys Architecture extension through Industrial Environment Plug-ins.

4.2.1 Regulations

It's well knowns that each sector has its specific regulations to observe. The Target Deployments depends on the main rules that the final project has to satisfy. The regulations plug-ins take into account these aspect and fix the performance to monitor and to achieve at every stage of the project.

4.2.2 Configurations

In the same way, the simulation case settings are different for each industrial sector. A structural analysis for naval industrial is completely different from a structural analysis for automotive industrial. The configurations plug-ins allows to decouple the diversity from the main framework.

4.2.3 Data Sets

Finally, the input/output data to exchange through engineering environment may be completely different in term of Data Sets. This is another issue to solve through the layer of Industrial Environment Plug-ins.

4.2.4 Collaborative Use Cases

In terms of collaboration, the Industrial Environment Plug-ins Layer guarantees all the functionality described in the previous chapter. The Figure 4.5 depicts the scenario.

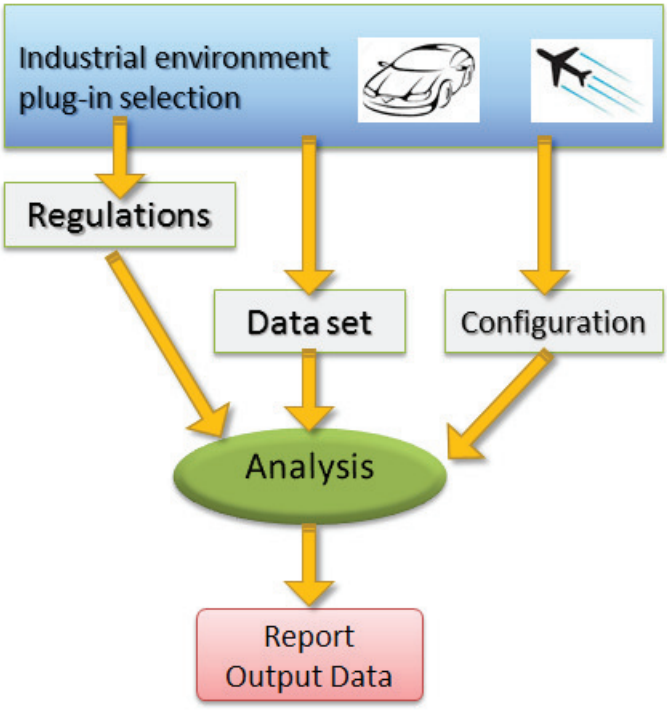


Figure 4.5: Industrial Environment plug-in functionality

4.3 Conclusions

This chapter described the need to adapt the platform to each industrial environment. The question was solved through the Floasys architectural nature that allow the implementation of several industrial context with the addition of another layer that manage the differentiation in terms of Regulations, Configurations and Data Sets.

Chapter 5

Conclusions & Future Works

Contents

This concluding chapter provides an overview of the entire dissertation describing the main achieved results.

5.1 Summary

Nowadays enterprises are world-wide and compete on a global market. They have multiple locations around the world over multiple nations and often over different time zones. In this context modern Internet technologies can support the communication, coordination and the sharing of resources among workers. Starting with the main objectives that a CSCW system is intended to achieve, I have played a direct field experience, I found that these elements emerge as customer needs and I have implemented them in the platform prototype developed.

The aim of this dissertation is to analyse a real use case provided by Fiat Chrysler Automobiles to understand the needs of engineering teams that work far from each other, and explore the use of modern technologies, such as the collaborative systems to support their work.

In this dissertation, I was able to identify the key collaborative requirements analyzing a real use case of two teams within FCA, through the use of stakeholders interviews, on-site observations and an on-line user survey. In addition, I was able to address these requirements with an integrated, extensible and modular architecture to collect, centralise and store simulations in open format independently by the original simulator software. Over the basic platform there are other tools and services like simulation tagging, simulation searching and engineering features. The provided Floasys architecture is modular and extensible so industries can customise it designing, implementing and testing new modules to plug in the architecture.

In this way, the dissertation provides solutions and technologies able to address the collaborative requirements. Furthermore, requirements, solutions and technologies are tracked through the dissertation and their links.

Floasys is Web-based platform designed and developed to meet the collaborative and engineering requirements. It is an 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:

- simulators wrappers that communicate with the simulator software to get the simulation data and transform them in XML open format;
- the version control repository for the XML files (e.g., SVN);
- an enterprise search engine to index, cache and search the XML documents (e.g., Apache Solr), and
- the central web server that provides the Web content (e.g., JBoss servlet container).

From scalability point of view, Apache Solr has been choose because it can scale using SolrCloud. To guarantee the data versioning, Floasys relies on Subversion technologies and it supports multiple SVN repositories and a mainstream container. In this way the data to put under versioning can be spread over multiple servers splitting data by folder.

Of course, as next steps, different tests have been planned: controlled benchmark tests to quantitatively assess and evaluate the Floasys performance, reliability and robustness. Also the evaluation of the graphical user interface is interesting so the plan is to conduct an evaluation study to analyse the usability of the Floasys user interface, and the user satisfaction when interacting with it [81, 82]. The user acceptance of the software will be investigated as well [83].

5.2 Future Works

Through this dissertation many other ideas come to light that could be explored. This section discusses the main ways to further explore the topics.

The main research avenue that could be further explored are:

- **Visually Repository Exploration.** In a working context analysts perform a lot of simulations and experiments per year, so the repository are really huge and contains the products history of many years. One of the main observed difficult concern the repository data visualisation and navigation. Analysts need to get insight into the overall repository getting first its overview. The questions that usually arise are: *how many simulations we have for the project Y? How the vehicle performances have evolved during the years?*

- **Experimental data heterogeneity** Engineers run test-bed engines for hours collecting a huge amount of data generating a high valuable repository of assets over years. The main issue here is the heterogeneity among the data representations because engineers use different engine test-beds. One idea to manage this heterogeneity is to use the technologies form the semantic web to create a common representation storing everything in a central repository. At there is a performance issue because the same repository must be able to manage queries to filter and compare data.

- **Social networks coupled with simulation repository.** An interesting future work is the opportunity to link the subversion repository (SVN) that contains simulation data in open format with an internal private social network enabling the discussions on artefacts [84]. The research aim could to understand and evaluate the benefits of using the social in the field of industrial CFD simulations.

Bibliography

[1] Nader Ale Ebrahim, Shamsuddin Ahmed, and Zahari Taha. Virtual teams: a literature review. *Australian Journal of Basic and Applied Sciences*, 3(3):2653-2669, 2009.

[2] Ilze Zigurs and Bjorn Erik Munkvold. Collaboration technologies, tasks, and contexts. *Human-computer interaction and management information systems: Applications*, pages 143-169, 2006.

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

[4] Marina Mendonfica Natalino Zenun, Geilson Loureiro, and Claudiano Sales Araujo. The Effects of Teams' Co-location on Project Performance. In *Complex Systems Concurrent Engineering*, pages 717-726. Springer, 2007.

[5] Guido Hertel, Susanne Geister, and Udo Konradt. Managing virtual teams: A review of current empirical research. *Human Resource Management Review*, 15(1):69-95, 2005.

[6] Thomas H Davenport. *Thinking for a living: how to get better performances and results from knowledge workers*. Harvard Business Press, 2013.

[7] Paul S Chinowsky and Eddy M Rojas. Virtual teams: Guide to successful implementation. *Journal of management in engineering*, 19(3):98-106, 2003.

[8] Andrew P McAfee. Shattering the myths about Enterprise 2.0. *IT Management Select*, 15(4):28, 2009.

[9] Kai Hakkarainen and Sami Paavola. From monological and dialogical to trialogical approaches to learning. In *A paper at an international workshop" Guided Construction of Knowledge in Classrooms*, 2007.

[10] Claudio Gargiulo, Donato Pirozzi, Vittorio Scarano, and Giuseppe Valentino. A Platform to Collaborate around CFD Simulations. In *2014 IEEE 23rd International WETICE Conference, WETICE 2014, Parma, Italy, 23-25 June, 2014*, pages 205-210, 2014.

[11] Colin Kelly-Rand Michelle Boucher. Getting Product Design Right the First Time with CFD, 2011.

[12] Stephan Onggo, Simon Taylor, and Arman Tulegenov. The need for cloud-based simulation from the perspective of simulation practitioners. 2014.

[13] M Kirby. Custom Manual. Technical report, Technical Report DPO/STD/1.0, HCI Research Centre, University of Huddersfield, 1991.

[14] A Dix, J. Finlay, G. Abowd, and R. Beale. *Human-Computer Interaction (3rd edition)*. 2003.

[15] Per Runeson and Martin Host. Guidelines for conducting and reporting case study research in software engineering. *Empirical software engineering*, 14(2):131-164, 2009.

[16] Jongbae Moon, Chongam Kim, Yoonhee Kim, and Kum Won Cho. CFD Cyber Education Service using Cyberinfrastructure for e-Science. In *Networked Computing and Advanced Information Management, 2008. NCM'08. Fourth International Conference on*, volume 2, pages 306-313. IEEE, 2008.

[17] Volker Bertram and Patrick Couser. Aspects of Selecting the Appropriate CAD and CFD Software. *9th Conf. Computer and IT Applications in the Maritime Industries (COMPIT). Gubbio.*, 2010.

[18] William H Brown, Raphael C Malveau, and Thomas J Mowbray. AntiPatterns: refactoring software, architectures, and projects in crisis. 1998.

[19] Cunningham and Cunningham Inc. Anti-Pattern. Last checked on: 2014-09-08.

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

[21] Mark Perry and Thomas Margoni. Floss for the canadian public sector: open democracy. In *Digital Society, 2010. ICDS'10. Fourth International Conference on*, pages 294-300. IEEE, 2010.

[22] Rajiv Shah, Jay Kesan, and Andrew 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.

[23] Vasudeva Varma and Vasudeva VarmaWrite. *Software Architecture: A Case Based Approach*. Pearson Education India, 2009.

[24] Sherif Sakr, Anna Liu, Daniel M Batista, and Mohammad Alomari. A survey of large scale data management approaches in cloud environments. *Communications Surveys & Tutorials, IEEE*, 13(3):311-336, 2011.

[25] Chia-Wei Chang, Pangfeng Liu, and Jan-Jan Wu. Probability-based cloud storage providers selection algorithms with maximum availability. In *Parallel Processing (ICPP), 2012 41st International Conference on*, pages 199-208. IEEE, 2012.

[26] Brian W Fitzpatrick and JJ Lueck. The case against data lock-in. *Queue*, 8(10):20, 2010.

[27] Anne Geraci, Freny Katki, Louise McMonegal, Bennett Meyer, John Lane, Paul Wilson, Jane Radatz, Mary Yee, Hugh Porteous, and Fredrick Springsteel. *IEEE standard computer dictionary: Compilation of IEEE standard computer glossaries*. IEEE Press, 1991.

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

[29] Jez Humble and David Farley. *Continuous delivery: reliable software releases through build, test, and deployment automation*. Pearson Education, 2010.

[30] Peter Sempolinski, Douglas Thain, Daniel Wei, and Ahsan Kareem. A system for management of computational fluid dynamics simulations for civil engineering. In *E-Science (e-Science), 2012 IEEE 8th International Conference on*, pages 1-8. IEEE, 2012.

[31] OpenFOAM GUIs. "http://openfoamwiki.net/index.php/GUI". Last checked on 08/09/2014.

[32] Julian Weber. *Automotive Development Process*. Springer, 2009.

[33] SG Lee, Y-S Ma, GL Thimm, and J Verstraeten. Product lifecycle management in aviation maintenance, repair and overhaul. *Computers in Industry*, 59(2):296-303, 2008.

[34] Farhad Ameri and Deba Dutta. Product lifecycle management: closing the knowledge loops. *Computer-Aided Design and Applications*, 2(5):577-590, 2005.

[35] Juha Kortelainen. *Semantic Data Model for Multibody System Modelling*. VTT, 2011.

[36] Jongbae Moon, Kum Won Cho, Soon-Heum Ko, Jin-Ho Kim, Chongam Kim, and Yoonhee Kim. A Cyber Environment for Engineering Cyber Education. In *eScience, 2008. eScience'08. IEEE Fourth International Conference on*, pages 532-539. IEEE, 2008.

[37] Seonguk Lee and Chongam Kim. Development and Utilization of Online Computational Environment for Education and Research in Fluid Engineering. 2013.

[38] Y Jung, Jongbae Moon, D Jin, B Ahn, JH Seo, H Ryu, O Byeon, and JR Lee. Web simulation service improvement on EDISON cfd. *Computer Science and Technology (CST 2012)*, 2012.

[39] Young Jin Jung, Jongbae Moon, Du-Seok Jin, Bu-Young Ahn, Jerry Hyeon Seo, Hoon Ryu, Ok-Hwan Byeon, and JongSuk Ruth Lee. Performance Improvement for Web based Simulation Service on EDISON cfd. *International Journal of Software Engineering and Its Applications, to be accepted*, 2013.

[40] Seonguk Lee and Chongam Kim. Development and utilization of online computational environment for education and research in fluid engineering. 2013.

[41] Simantics platform. "https://www.simantics.org/simantics/about-simantics/simantics-platform/". Last checked on 28/07/2014.

[42] Eclipse rap remote application platform. "http://eclipse.org/rap/". Last checked on 01/09/2014.

[43] Jiten Rama and Judith 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.

[44] Claudio Gargiulo, Donato Pirozzi, and Vittorio Scarano. An architecture for CFD Workflow management. In *Industrial Informatics (INDIN), 2013 11th IEEE International Conference on*, pages 352-357. IEEE, 2013.

[45] Eric Newcomer and Greg Lomow. *Understanding SOA with web services (independent technology guides)*. Addison-Wesley Professional, 2004.

[46] Leonard Richardson and Sam Ruby. *RESTful web services*. " O'Reilly Media, Inc.", 2008.

[47] Richard Hall, Karl Pauls, Stuart McCulloch, and David Savage. *OSGi in action: Creating modular applications in Java*. Manning Publications Co., 2011.

[48] Xihui Chen and Kay Kasemir. Bringing Control System User Interfaces to the Web. *TUPPC078, ICALEPCS*, 13.

[49] GuanhuaWang. Improving data transmission in web applications via the translation between XML and JSON. In *Communications and Mobile Computing (CMC), 2011 Third International Conference on*, pages 182-185. IEEE, 2011.

[50] Apache Solr. Apache Software Foundation Solr, 2014.

[51] Rafal Kuc. *Apache Solr 4 Cookbook*. Packt Publishing Ltd, 2013.

[52] David Smiley and David Eric Pugh. *Apache Solr 3 Enterprise Search Server*. Packt Publishing Ltd, 2011.

[53] Timothy A Howes, Mark C Smith, and Gordon S Good. *Understanding and deploying LDAP directory services*. Addison-Wesley Longman Publishing Co., Inc., 2003.

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

[55] Petar Tahchiev, Felipe Leme, Vincent Massol, and Gary Gregory. *JUnit in action*. Manning Publications Co., 2010.

[56] Dorian Birsan. On plug-ins and extensible architectures. *Queue*, 3(2):40-46, March 2005.

[57] Eclipse Public License.

[58] Thomas Hauser Christopher L. Rumsey, Bruce Wedan and Marc Poinot. Recent Updates to the CFD General Notation System (CGNS). *50th AIAA Aerospace Sciences Meeting*, 2012.

[59] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design patterns: elements of reusable object-oriented software*. Pearson Education, 1994.

[60] Joe Walnes, J Schaible, M Talevi, G Silveira, et al. XStream. *"URL:http://xstream. codehaus. org "*, 2011. Last checked on 08/09/2014.

[61] C Michael Pilato, Ben Collins-Sussman, and Brian W Fitzpatrick. *Version control with subversion*. " O'Reilly Media, Inc.", 2008.

[62] Solr. Apache Solr. "https://lucene.apache.org/solr/". Last checked on 09/04/2014.

[63] Michelle Boucher and Colin Kelly-Rand. Getting Product Design Right the First Time with CFD. *Aberdeen Group: May*, 2011.

[64] Claudio Gargiulo, Delfina Malandrino, Donato Pirozzi, and Vittorio Scarano. Simulation data sharing to foster teamwork collaboration. *Scalable Computing: Practice and Experience*, 15(4), 2015.

[65] Brian Johnson and Ben Shneiderman. Tree-maps: A space-filling approach to the visualization of hierarchical information structures. In *Visualization, 1991. Visualization'91, Proceedings., IEEE Conference on*, pages 284-291. IEEE, 1991.

[66] Peter Demian and Renate Fruchter. Finding and understanding reusable designs from large hierarchical repositories. *Information Visualization*, 5(1):28-46, 2006.

[67] Ben Shneiderman. Tree visualization with tree-maps: 2-d space-filling approach. *ACM Transactions on graphics (TOG)*, 11(1):92-99, 1992.

[68] Andrew Fiore and Marc A Smith. Treemap visualizations of Newsgroups. *Technical Report, Microsoft Research, Microsoft Corporation: Redmond, WA*, 2001.

[69] Benoit Otjacques, Mael Cornil, and Fernand Feltz. Visualizing cooperative activities with ellimaps: the case of Wikipedia. In *Cooperative Design, Visualization, and Engineering*, pages 44-51. Springer, 2009.

[70] Renaud Blanch and Eric Lecolinet. Browsing zoomable treemaps: structure-aware multi-scale navigation techniques. *Visualization and Computer Graphics, IEEE Transactions on*, 13(6):1248-1253, 2007.

[71] Benoit Otjacques, Mael Cornil, Monique Noirhomme, and Fernand Feltz. CGD{A new algorithm to optimize space occupation in ellimaps. In *Human-Computer Interaction INTERACT 2009*, pages 805-818. Springer, 2009.

[72] Carla M. Dal, Sasso Freitas, Paulo R. G. Luzzardi, Ricardo A. Cava, Marco A. A.Winckler, Marcelo S. Pimenta, and Luciana P. Nedel. Evaluating Usability of Information Visualization Techniques. In *Brazilian Symposium on Human Factors in Computing Systems*, 2002.

[73] Ben Shneiderman. Direct manipulation: A step beyond programming languages. In *ACM SIGSOC Bulletin*, volume 13, page 143. ACM, 1981.

[74] Andrew Blake, Gem Stapleton, Peter Rodgers, and John Howse. How Should We Use Colour in Euler Diagrams? In *Proceedings of the 7th International Symposium on Visual Information Communication and Interaction*, page 149. ACM, 2014.

[75] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. D3 data-driven documents. *Visualization and Computer Graphics, IEEE Transactions on*, 17(12):2301-2309, 2011.

[76] Ka-Ping Yee, Kirsten Swearingen, Kevin Li, and Marti Hearst. Faceted metadata for image search and browsing. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 401-408. ACM, 2003.

[77] Greg Smith, Mary Czerwinski, B Robbins Meyers, G Robertson, and DS Tan. FacetMap: A scalable search and browse visualization. *Visualization and Computer Graphics, IEEE Transactions on*, 12(5):797-804, 2006.

[78] Ramez Elmasri. *Fundamentals of database systems*, volume 2. Pearson Education India, 2007.

[79] Ben Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Visual Languages, 1996. Proceedings., IEEE Symposium on*, pages 336-343. IEEE, 1996.

[80] Charles E Leiserson, Ronald L Rivest, Clifford Stein, and Thomas H Cormen. *Introduction to algorithms*. MIT press, 2001.

[81] Computer System Usability Questionnaire, December 2014.

[82] Questionnaire for User Interface Satisfaction.

[83] Fred D Davis. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS quarterly*, pages 319-340, 1989.

[84] Delfina Malandrino, Ilaria Manno, Alberto Negro, Andrea Petta, Vittorio Scarano, and Luigi Serra. Social team awareness. In *Collaborative Computing: Networking, Applications and Worksharing (Collaboratecom), 2013 9th International Conference on*, pages 305-314. IEEE, 2013.

An architecture for CFD Workflow Management

Claudio Gargiulo
R&D - Aerothermal CFD
Fiat Group 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

***Abstract*—Nowadays, to design product impacted by fluid flow, industries use Computational Fluid Dynamics (CFD) to get a better insight into product behaviour. In this paper, we present a system architecture design for CFD Workflow management.**
***Index Terms*—Architecture, CFD, CFD Workflow, management**

I. Introduction

Computational Fluid Dynamics (CFD) is computer numerical simulation able to solve and analyse problems that involve fluid flow, heat transfers and related physical phenomena. Actually, it is used in many industrial sectors such as automotive, aerospace, high tech, oil/gas and so on. Today, industries understand the advantages of using CFD simulations during the product development process especially when their products are impacted by fluid flow, heating, cooling and so on [1]. The manufacturers' goal is to bring high quality products to the market quickly, keeping the costs down. The top external pressure factors that lead the product development are: time-to-market, quality and cost [2]. These factors are often in conflict together [3]. Products are becoming more complex. They have a high number of components and configurations. Brands offer many options and the customers can combine the options together to get their configuration [4]. Engineers need to understand how the components interact together and to assess the performance of each configuration under many physics conditions. The prototypes and the infrastructure that are required to asses the prototype performances (e.g. the wind tunnel for the automotive sector) are expensive. So, it is often expensive and time-consuming to assess the product behaviour for every physics condition within the budget constraints, especially for the extreme conditions.

A benefit of using computer simulations is a reduction of the number of the physical prototypes that leads to a cost reduction. According to the market research "*Engineering Evolved*" [3] three or more different types of simulations are able to reduce the number of prototypes by 37% and CFD is one type of simulation [2]. CFD simulations are able to reduce the number of physical prototypes [3], to manage the overall product system complexity and to get a better insight into product behaviour since the initial development process stages [2].

Standard CFD Workflow phases are shown diagrammatically in Fig. 1: pre-processing, solving and post-processing phases. Industries are looking for an increasingly integration of the CFD Workflow into the business process [5]. This integration heavily depends on internal team organization, industry's best practices, the product development plan and industry's policies. Of course, CFD software are designed so that they focus mainly on the CFD simulations; business process integration is not their main goal and, it is often completely ignored. The main benefit of integrating CFD Workflow into processes is to promote collaboration between CFD engineers, design engineers and analysts [2]. In an industrial context, CFD simulations are just a part of a more wide and complex product development process, so engineers perform additional tasks to standard CFD Workflow. For some products, design and CFD tasks can sometimes be done by the same team, using integrated products (like SolidWorks and CATIA). In the automotive industry (and in other fields like aerospace) the teams are different and therefore the CFD workflow is distinct from design. Besides, in the automotive industry CFD engineers perform different types of CFD simulations on the same product (i.e. external aerodynamics, aeroacustics, air conditioning and engine thermal analysis), so they need to use different CFD software because each one is validated for its own area of application. We observed that many engineers' tasks are repetitive, error-prone and time-consuming (e.g. find and compare simulations results, document generation, parametric studies) and they can be automated. Another issue is the simulation data and results accessibility. All engineers, both CFD and design engineers, must have full and user-friendly access to centrally managed simulation results.

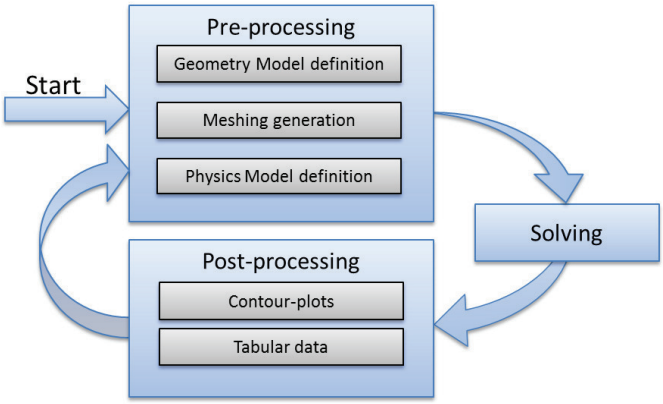


Fig. 1. CFD Workflow.

Our paper presents and validates a flexible architecture to manage the CFD Workflow. Our idea is to design and develop an architecture that enables the creation of tools to use within CFD Workflow. Architecture is the foundation for all future services upon existing CFD software.

The paper is organized as follows. Section II presents both functional and non-functional requirements for the system to design and it also reports some aspects of engineers' tasks to automate. Section III describes designed architecture to meet non-functional requirements. Section IV presents Floasys the prototype of the architecture that we are developing and some built tools.

II. Managing CFD Workflow

During the last year we have worked closely with a team of professionals that extensively use CFD, analysing their needs and supporting many of the findings and suggestions from literature and recent survey in the field. The professional team is composed by four highly skilled engineers with a strong CFD experience. They work in the automotive industry and they are not computer scientists. The issues that we are facing within automotive sector seem to be very common issues also in other sectors. Aberdeen Group market research [2], through a survey and interviews, has studied the experiences of 704 companies about the use of CFD to design products. Companies belong to industrial equipment, automotive, aerospace and defense, high tech, oil/gas and military/public sectors. In the analysed automotive context we observed issues that are almost similar to what has been outlined at the end of Aberdeen Group study. So, we hope that our analysis, that is based on a limited context (i.e. automotive), can be successfully translated into other similar contexts, as well.

Our initial approach is based on an empirical study through direct observations about how engineers work on simulations and discussions with them. At beginning we have tried to identify error-prone, repetitive and time-consuming tasks. So, we have focused on single team member tasks and we have identified initial functional requirements. In this step we have realized that many tasks (e.g. simulation finding, document generation) are not covered by CFD software, because they depend on internal organization and we have pointed out the need for simulation data sharing.

Our claim is that industries require many services upon and correlated to CFD simulations. These services are important in order to reduce costs, time and to improve engineers' tasks. Besides, there is a strong need to increase the integration of CFD Workflow into the overall product development process. So, our idea is to design an architecture for CFD Workflow management that supports the creation of services within the CFD Workflow, independently by and upon CFD software. This section describes some gathered functional requirements and it covers non-functional requirements that have led the architecture design.

A. Functional Requirements

The following are the identified actions to perform in order to improve industrial daily CFD engineers' work:

- centrally manage simulation data;
- identify and automate tasks through software tools;
- take advantage of wizards and templates.

1) Centrally manage simulation data: The aim is to improve the simulation data accessibility and to promote knowledge sharing among engineers.

We observed that this action is already performed by storing all the simulations data in a proprietary file format within one or more shared network folders. Engineers usually need to find the old simulation data stored in the repository to compare the results. It is interesting for the engineers to search the simulations stored in the central repository by the simulation revision name, the physic properties and other tags defined by the end-users. The simulation files are often binary files, so the engineers can not use the search tools based on text content.

We think that simulation data tagging, linking and searching are interesting features. The file system has a hierarchical structure and it is not enough to meet these described criteria.

According to Aberdeen Group study [2], in order to improve their performances, companies should "*Centrally manage previous simulation results*". It is not useful for engineers to store centrally only the simulation results: they need both the simulation results and the configuration case data. So, we decided to centrally manage the configuration case with all settings (e.g. boundary conditions, physical properties, the max number of iterations), the solving logs, the convergence charts and results (i.e. contour-plots and tabular data). Over the years industries perform a high quantity of simulations and each simulation file takes up over ten gigabytes due the geometry model details. Engineers usually need to find simulation data to compare the results.

We think that a central repository for all simulation data provides an historical view for a given project and a strategic competitive advantage for the future. Simulation central repository is the knowledge base on which apply metrics, perform statistics and make strategic decisions.

In order to centrally manage simulation data, we have identified the following functional requirements: (1) *centrally manage configuration case, solving logs, convergence charts and results*; (2) *simulation data tagging*; (3) *search based on simulation data and simulation tags*; (4) *automatic documents generation from simulation data* and (5) *version control*.

2) Tasks automation: Here we report some examples of tasks to automate, considering that this paper focus on the architecture design rather than on a particular tool. CFD Workflow needs a better integration in the product development process. For example, engineers create many documents based on the simulation experiments and results. They always use the same document structure but with different data. So, the first task to be automate is the automatic document generation. The first time engineers will prepare n document templates, each one with its basic structure, and will store them in the system. In order to generate documents, the engineer then selects one or more simulations and chooses a document template; the system merges the simulations data and results with the chosen template. The CFD team is highly skilled and has a lot of experience with command line tools, but as reported in literature this requires high training costs. A common CFD engineer task is to run simulations using the HPC systems.

Usually, engineers use a small set of commands through an ssh connection to submit/kill a job and to monitor the running jobs. Engineers need to do some other operations in order to monitor the simulation convergence data. Our idea is to create a Monitoring Tool: a workbench that provides in one view the job queues and the convergence charts for the running simulations.

3) Take advantage of wizards and templates: Wizards and templates guarantee that all team members work in the same way. Besides, wizards incorporate the best practices and support less experienced users. Both wizards and templates are a good way to support engineers in the repetitive and error-prone tasks.

B. Non-Functional Requirements

Our goals are to identify leading non-functional requirements and to find a trade-off among them. Non-functional requirements gathered from our analysis are: *extensibility, modularity, open, portability, Intranet-based* and *deployment*.

1) Extensibility: Extensibility is the ability of a software system to allow and accept significant extension of its capabilities without major rewriting of code [6]. Extensibility answers to the question: how easy is it to add new functionality to the system [7]? In order to add future functionalities that depend by industry's product development process and by particular CFD team, the system should be open for future extensions. Industries want functionalities tailored to their needs and they want to develop their in-house tools. Industries initially use the base set of functionalities, then in order to accommodate future needs they can develop new functionalities. For example, each industry uses its internal document template; the system must be extensible in order to support future format and future changes in document templates.

2) Modularity: 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 [6]. A module is a logically separable part of a program [6]. The software system is made by modules that fit together to create the overall system. Modularity advantages are: module replacement, creation of new modules and a well-structured system. With modularity, it is possible to identify which services are provided by each module. The system is tailored to customer needs: each customer can compose a system loading just the need modules.

3) Open: The system must use open protocols and open formats to avoid vendor lock-in and to be interoperable with other systems. Vendor lock-in is the phenomenon that causes customer dependency on given vendor with regard to specific good or service [8]. We observed a potential CFD software vendor lock-in data format that occurs when end-user stores data in a proprietary format and the proprietary software does not have import/export functions to an open format. CFD Workflow requires the use of many different software: CAD/CAE, CFD and post-processing software. Before choosing a software, industries estimate the benefits in using open formats. Nowadays, industries use open formats to store and to exchange geometry data between CAD/CAE software and pre-processing tools. IGES and STEP are well-established open formats for geometry data exchange. Industries usually do not store CFD simulation data in an open format. CFD General Notation System (CGNS) is a standard for CFD input and output, grid, flow solution and boundary conditions [9]. CGNS concerns with CFD data representation and can be used also for data exchange [10]. A software must have import and export options [11] to open formats, but not all software vendors offer these functionalities. In our case study, simulation data such as geometry mesh, physical model, convergence data and results are stored into proprietary file format and the proprietary software does not have the export option to CGNS format. This practice is due to proprietary software adoption. Only simulation results are exported in a neutral format.

4) Portability: Portability is how easily a system or component can be transferred from one hardware or software environment to another [6]. Our goal is to build tools and automated procedures that run independently by CFD software. We can say that tools are portable across CFD simulators.

Many CFD software provide a script language which enables engineers to extend software functionalities. Over the years, engineers have written their own scripts to automate the execution of tasks. Scripts are very important for the manufacturers because they contains the experience of the engineers, the internal procedures and the business process practices. Scripts can increase productivity because they automate tasks, but they can also create a potential vendor lock-in problem: scripts are fitted on particular CFD software features and are not portable across CFD software. Our aim is to create automated procedures that run over any CFD software. A solution to vendor lock-in anti-pattern is to design the system with an isolation layer [12].

5) Intranet-Based: Engineers access to services, tools and resources within the company Intranet. In using the system, engineers expect to have Intranet performances, such as high bandwidth, low latency and good response time.

6) Deployment: Deployment into an existing industry can be expensive especially when it must be done on each client workstation. System design must take in account the future deployment costs. The system should reduce costs and time for deploying and updating the system on all clients.

III. System Architecture Design

This section reports the design decisions made to achieve the requirements presented in Section II. Here, we intentionally do not mention any particular software technology, and we speak about patterns, architectures and protocols to guarantee a future reproducibility of our architecture. In the first part we describe a client/server system architecture taking in account both hardware and software, then we give more details about the server-side software architecture.

A. System architecture

Our system has a client/server architecture (Fig. 2). The clients are web-based, so the end-users use their web browser

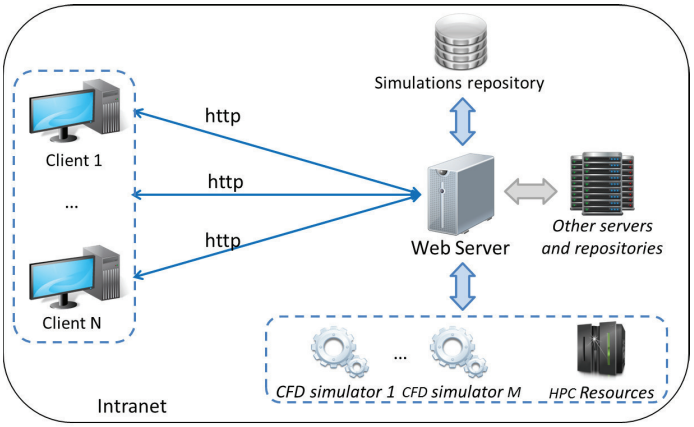


Fig. 2. System Architecture.

to access to the simulation data, HPC resources and CFD functionalities provided by the central web server. The server exchanges data with the simulation repository that provides the persistence service. The simulation repository stores the simulation data: an important asset for the industry over the years. It stores data in an open format (e.g. CGNS) and guarantees version control. We aim to store and to put under version control geometry model, physics model, boundary conditions, charts, simulation results, contour plots and generated documents. The server does CRUD (create, read, update and delete) operations on the repository. However, nowadays each simulation file takes up gigabytes of disk space and over the years the number of simulations grows incredibly: it is not practical to store all data for all simulations. Manufactures (based on their needs) should choose which data must be stored. Often, engineers perform parametric studies on the same geometry model changing only the physic values. In these cases, it is convenient to store the geometry only one time and correlate simulation results, physical model and the configuration case to the same geometry model. Sometimes, it is convenient to not apply version control to the geometry model. In our analysis, CFD engineers do not run and solve an old simulation, they only need data for results compare. Engineers can access to configuration case made by physical model, charts, tabular results and contour plots without run CFD software again. We add metadata and services (like versioning) to existing simulation data.

Industries use CFD simulators that only work with proprietary file format not allowing the import/export in an open format. In this situation it is still feasible to have also a vendor format independent central repository that stores simulation data in a neutral format. Our solution is to convert proprietary format data into an open format and to adopt an isolation layer [12] between CFD simulators and tools (the solution is detailed exposed in the next section III-B).

The central web server interacts with other servers and repositories. For example, companies usually already have internal servers that provide security and authentication services, so our architecture does not directly provide them but relies on other existing Intranet servers. The web server directly

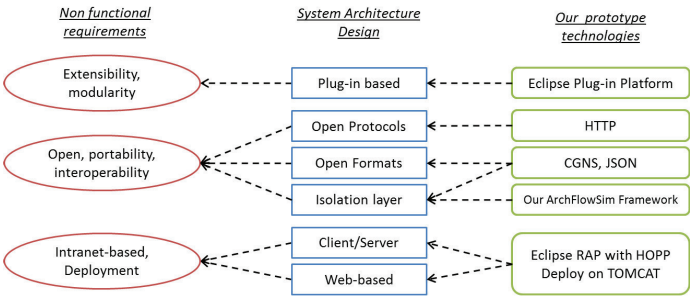


Fig. 3. System Requirements Mapping.

connects to HPC resources and it makes them available to end-users. Job submission, job kill, simulation monitoring and real-time convergence charts are all services available to non-HPC system experts.

B. Server-side software architecture

The server-side software architecture (Fig. 4) consists of three layers. The tools are in the top layer. A CFD tool is an end-user GUI that provides one or more useful functionalities to engineers. The middle layer is the isolation layer and finally, the bottom layer consists of CFD software wrappers.

Each tool performs a well-defined engineering task. Some of the tools depend on the internal team organization and by the business process. For example, document generator tool accepts document templates and creates documents from them. The server-side architecture is based on a pure plug-in architecture in which everything is a plug-in [13]. Each box of the software architecture (Fig. 4) 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.

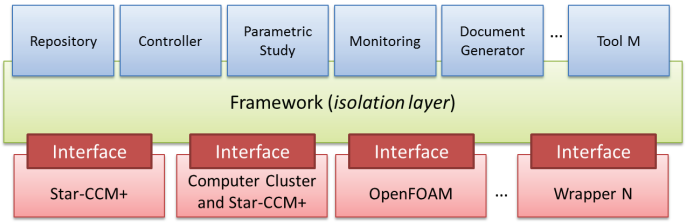


Fig. 4. Server-Side software architecture.

Our architecture supports the concurrent use of multiple CFD software for companies who have decided to employ, for example, both commercial CFD software (e.g. CCM+ developed by CD-adapco) and OpenFOAM as reported in [11].

Plug-in based architecture meets the extensibility and modularity non-functional requirements. Open data formats and isolation layer meet the open and portability non-functional requirements. Open protocols and open formats guarantee the future interoperability with other software. The client/server and the web-based architecture avoid the installation of the software on each workstation. Worries about responsiveness

and bad tolerance of networks outages, typical disadvantages of these architecture, are mitigated by the Intranet setting.

IV. Floasys Prototype Architecture

Our aim is to design and implement an architecture that provides software reusable-building boxes to create new CFD tools upon any CFD simulator. This section presents the prototype based on the architecture described in Section III: it is currently under development and on site testing. It has a pure plug-in architecture [13] based on the Eclipse Platform [14]. Floasys's plug-ins can be arranged in three layers as shown in Fig 5: (1) The top layer consists of CFD tool plug-ins. Each tool can add a new perspective or can add a new view to an existing perspective. Usually, tool design is based on the MVC pattern. (2) The middle layer is the isolation layer. It provides the core API and the common simulation model. Plug-ins in the middle layer provide services to up layer tools by abstracting CFD simulators on bottom layer. (3) In the bottom layer, simulator wrappers communicate and exchange data with CFD simulators.

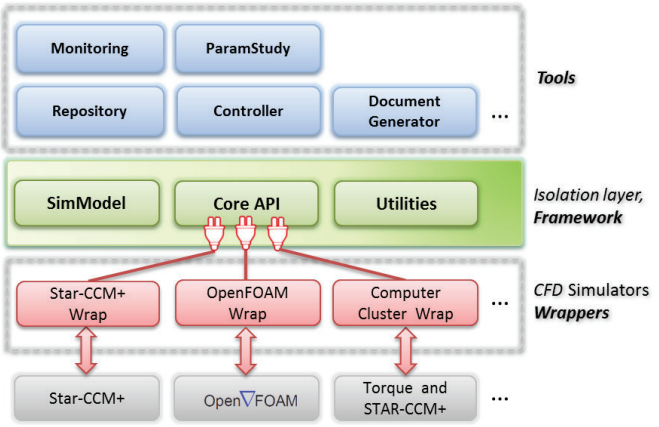


Fig. 5. Floasys Architecture.

Floasys is a web-based client/server architecture (Fig. 6). We use the Eclipse Remote Application Platform (Eclipse RAP) to develop the web application. RAP core implements [15] the Half Object Plus Protocol pattern [16]. Our assumption is that the system runs on the industry Intranet infrastructure, so the use of HOPP pattern is not in contrast with the end-to-end principle [17] because Intranet provides high bandwidth, high availability and low latency compared to Internet connection. The server-side software needs a JEE servlet container (e.g. Apache Tomcat) and a framework for the plug-in life cycle management (e.g. Eclipse Equinox OSGi).

A. Simulation Model

The Simulation Model (SimModel) stores the simulation data. Its aim is to store everything about the simulation not only the geometry, so it stores the configuration case with all settings and results. For example, it stores the boundaries, the physical properties, the stopping criteria, the log files, the outcomes, the charts, the tabular data, the contour plots and the generated documents. SimModel is a tree-like data structure

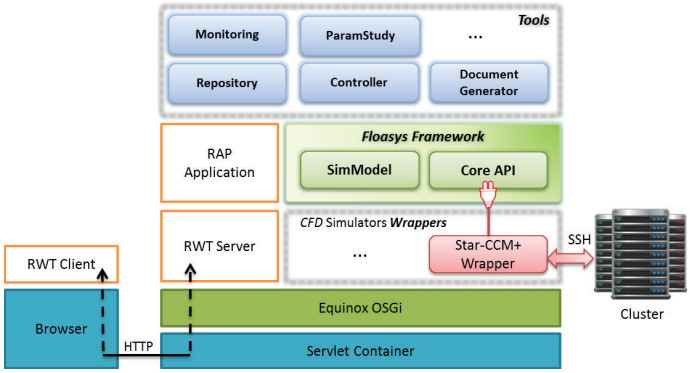


Fig. 6. Client/Server architecture.

as in CGNS standard [9], so it can store each information about the simulation with a tree node. The system can also link other metadata (e.g. the tag names) to the simulation by adding a new tree node. CFD tools should be independent from any particular CFD simulator and should be used across many CFD simulators. To obtain such simulator independence, we have developed the SimModel plug-in showed in the middle layer of figure 5. SimModel is the only joint point between CFD simulators on bottom layer and CFD tools on top layer. Tools implementation is based only on the simulation model and they interact only with simulation model.

B. Core API

From our framework point of view, CFD simulators are black boxes which we can extract model information from and interact with, in order to change some properties. Each simulator has its own internal data model; it consists of geometry model, regions, boundaries, physical properties and so on. Framework API allows to extract simulation data stored inside a proprietary CFD simulator to obtain the framework simulation model for a given simulation. Framework relies on CFD wrapper simulator implementation to extract or change simulation data. At the end of the day we are interested in collecting simulation data and building the simulation model. We store the built simulation model in an open source format (i.e. xml) to avoid the vendor lock-in issue and to provide other services such as the version control and the simulation finding by its data. Potentially, each CFD tool works independently by the CFD engine technology (e.g. operating system, programming language, proprietary and open source software) because each tool interacts directly with the simulation model.

C. CFD Wrappers

CFD wrappers embed CFD software features and provide them through a common interface defined by the Core API. We have categorized CFD simulators by the source code availability and open protocols implementation. CFD software wrappers currently available in our prototype are: Star-CCM+ on Torque cluster and OpenFOAM CFD codes. Star-CCM+ is a proprietary software developed by CD-adapco while OpenFOAM is open source; both CFD software do not directly implement an open format (e.g. CGNS). For

OpenFOAM it exists a converter called *foamToCGNS* available into OpenFOAM Extend Project. From the technical point of view, the interaction between CFD simulators can be based on simulator's API invoking, command line simulator execution or network based interaction. The implementation of a wrapper is a non trivial task and is widely studied; literature describes many techniques and automate wrapper generators [18] (e.g. CORBA wrappers generators). Star-CCM+ Wrapper on bottom layer (Fig. 6) interacts with a computer cluster through SSH to submit jobs, to handle job queue, to monitor the running simulations, to modify simulation file and to extract information from simulation.

V. CFD Tool examples

This section describes briefly some CFD tools actually developed and based on the above architecture. A CFD tool is an end-user GUI that provides one or more useful functionalities to engineers. Currently available tools are: the repository tool, the simulation controller tool, the parametric study tool, the simulation monitoring tool and the document generator tool.

Floasys application GUI is based on the perspective and view concepts, a traditional GUI organization in Eclipse [19]. Each perspective is a visual container for a set of views. Perspectives support the task oriented interaction [20], the CFD engineer will use a different perspective depending on the task to perform. Each tool can contribute with a new set of views arranged in one or more perspectives or it can contribute to an existing perspective with a new view.

The Simulation controller perspective shows data about the selected simulation; it shows simulation data in a tree-like structure (Fig. 7). This perspective is the main perspective because it allows to access to other tools such as document generator tool and parametric study tool. For example, from the simulation tree the user can drag-and-drop simulation items in the parametric study tool. In an industrial context, CFD engineers perform several simulations on the same model by changing the set of parameters values. Engineers run many simulations about the same vehicle model; each simulation runs with a different inlet velocity value and produces a different result. Finally, all results are compared together. Parameter study aim is to assess how a variation of a parameter value affects simulation solution and which impact has on design. Parameter study can be laborious, tedious, repetitive and error-prone task without an automated tool [21].

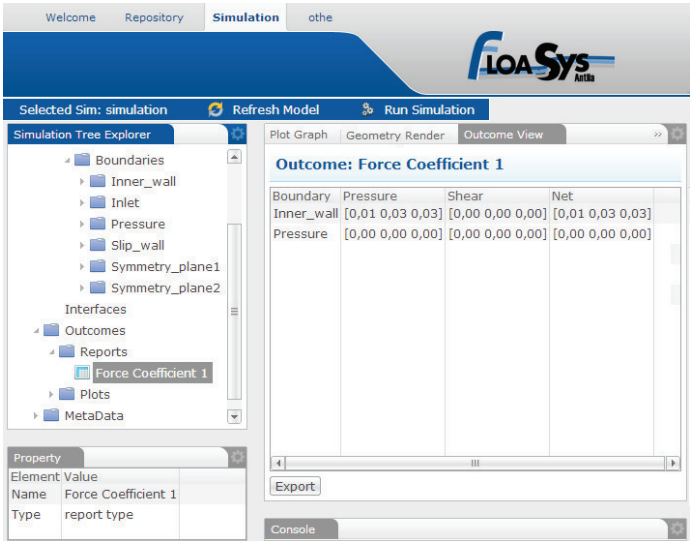


Fig. 7. Simulation controller perspective.

VI. Acknowledgments

The authors gratefully acknowledge the help and the support of Comprensorio CRF Elasis Pomigliano (Fiat Group Automobilies) and, the interactions with Aerothermal CFD team. In particular, the work has been significantly improved by the interesting and stimulating discussions with Antonio Cucca and Ugo Riccio. The authors also thank the anonymous reviewers for useful comments.

References

[1] M. Boucher, "The ROI of Cuncuttent Design with CFD," 2011.
[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] E. Sindhu, A. Lee, and S. M. Salim, "Coves: an e-business case study in the engineering domain," *Business Process Management Journal*, vol. 10, no. 1, pp. 115–125, 2004.
[6] 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," 1991.
[7] B. Bruegge and A. H. Dutoit, *Object-Oriented Software Engineering Using UML, Patterns and Java-(Required)*. Prentice Hall, 2004.
[8] 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, pp. 141–150.
[9] 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.
[10] M. Poinot, M. Costes, and B. Cantaloube, "Application of cgns software components for helicopter blade fluid-structure strong coupling. 31st european rotorcraft forum," *Florence, Sept*, 2005.
[11] V. Bertram and P. Couser, "Aspects of Selecting the Appropriate CAD and CFD Software," *9th Conf. Computer and IT Applications int the Maritime Industries (COMPIT). Gubbio.*, 2010.
[12] W. H. Brown, R. C. Malveau, and T. J. Mowbray, "Antipatterns: refactoring software, architectures, and projects in crisis," 1998.
[13] D. Birsan, "On plug-ins and extensible architectures," *Queue*, vol. 3, no. 2, pp. 40–46, Mar. 2005.
[14] J. Des Rivières and J. Wiegand, "Eclipse: a platform for integrating development tools," *IBM Syst. J.*, vol. 43, pp. 371–383, 2004.
[15] Last checked on May 24, 2013. [Online]. Available: http://eclipsesource.com/blogs/2013/02/01/rap-2-0-countdown-15/
[16] J. O. Coplien and D. C. Schmidt, Eds., *Pattern languages of program design*. NY, USA: ACM Press/Addison-Wesley Publishing Co., 1995.
[17] J. H. Saltzer, D. P. Reed, and D. D. Clark, "End-to-end arguments in system design," *ACM Trans. Comput. Syst.*, pp. 277–288, 1984.
[18] S. J. Eric Wohlstadter and P. Devanbu, "Generating Wrappers for Command Line Programs: The Cal-Aggie Wrap-O-Matic Project," *Proc. of the 23rd Inter. Conf. on Software Engineering*, 2001.
[19] D. Rubel, "The Heart of Eclipse," *Queue*, 2006.
[20] D. Springgay, "Using perspectives in the eclipse ui," *Eclipse Corner Article, Object Technology International, Inc*, 2001.
[21] M. Yarrow, K. M. McCann, R. Biswas, and R. F. V. d. Wijngaart, "An Advanced User Interface Approach for Complex Parameter Study Process Specification on the Information Power Grid," in *Proc. of the 1st IEEE/ACM Inter. Workshop on Grid Computing*, 2000, pp. 146–157.

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

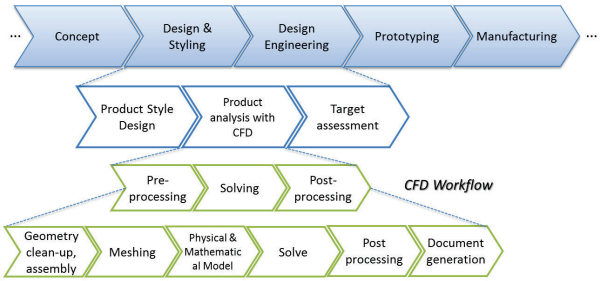


Fig. 1: Product Development Process and CFD Workflow.

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.

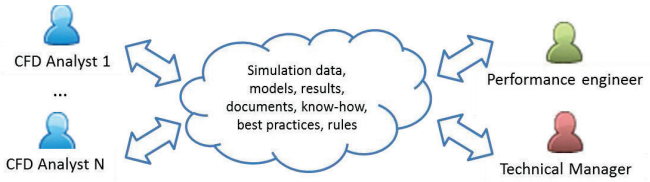


Fig. 2: CFD functional area stakeholders.

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.

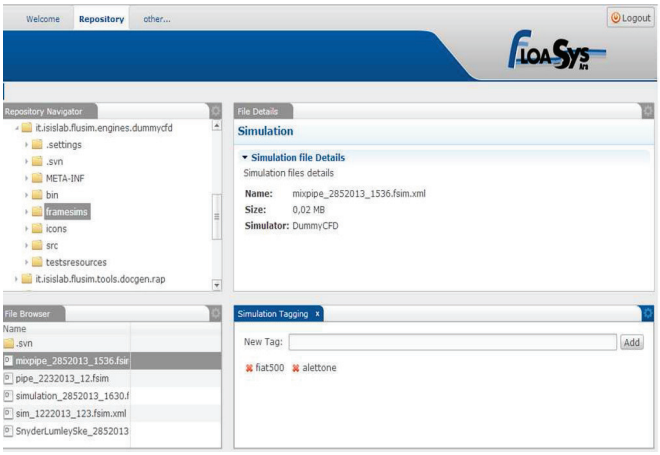


Fig. 3: Repository tool with the tagging file operation.

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.

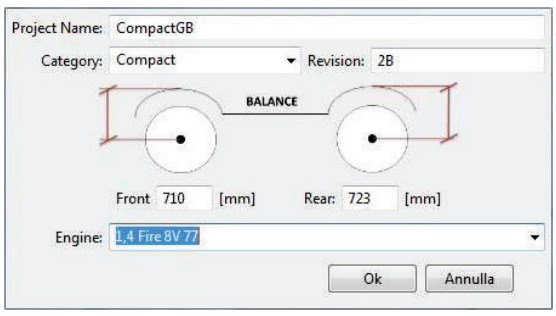


Fig. 4: An example of structured data to store.

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.

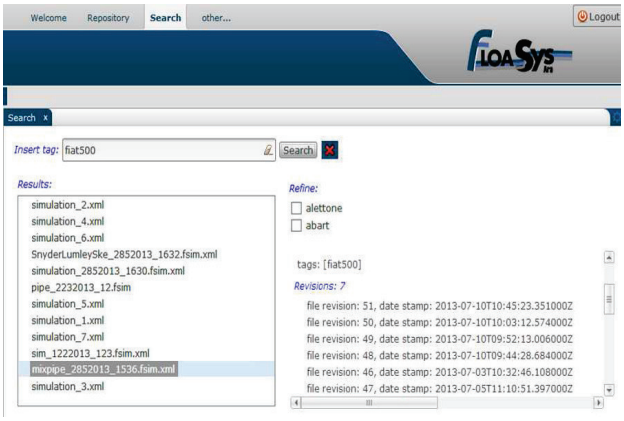


Fig. 5: Search Tool

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 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.

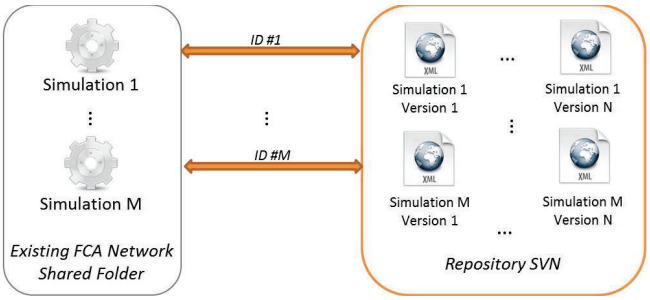


Fig. 6: Simulation Data versioning.

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

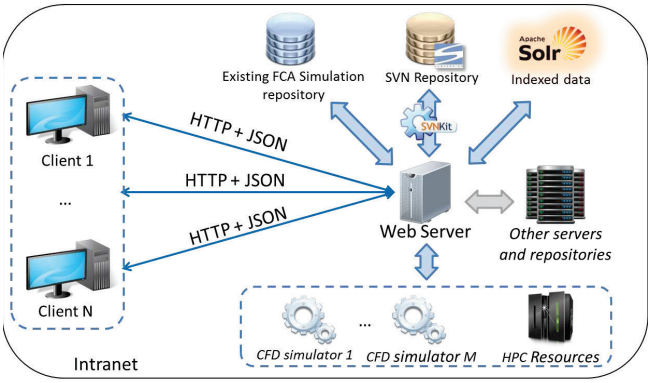


Fig. 7: Floasys Client/Server Architecture.

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.

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

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).

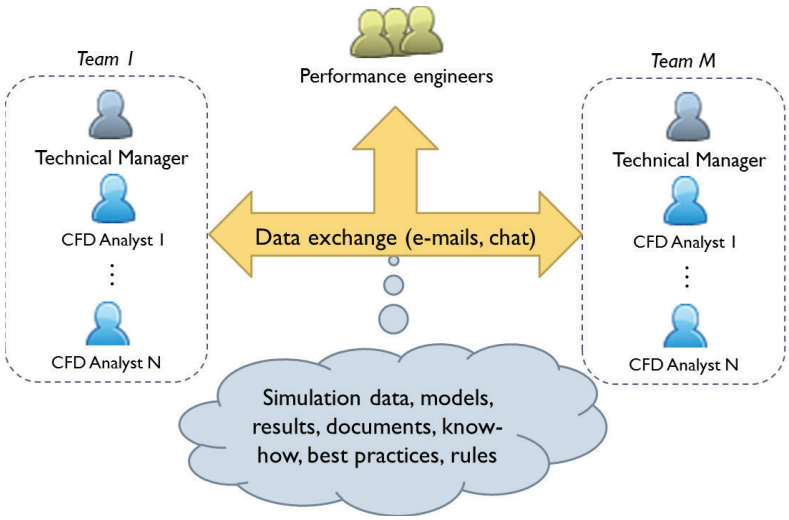


FIG. 2.1. *Geographically distributed teams that collaborate together in asynchronous way.*

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.

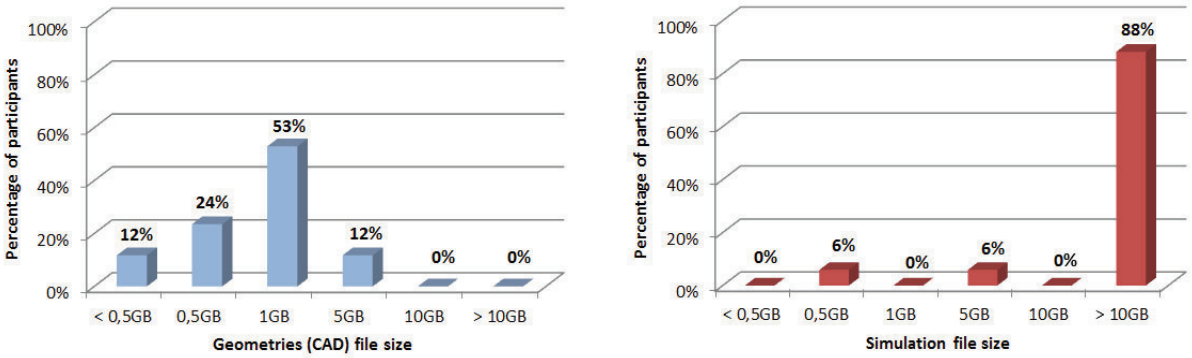


FIG. 2.2. *Geometries (question Q7) and simulations file size (question Q8).*

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.

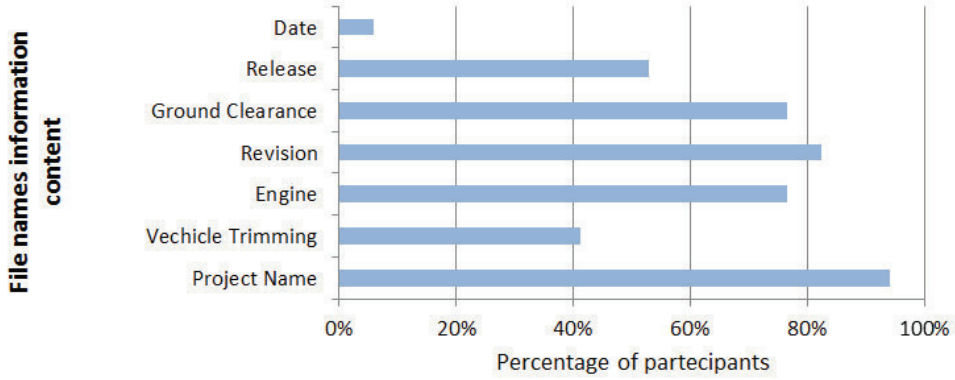


FIG. 2.3. *Information inserted in the simulation file names (multiple choices question Q12).*

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.

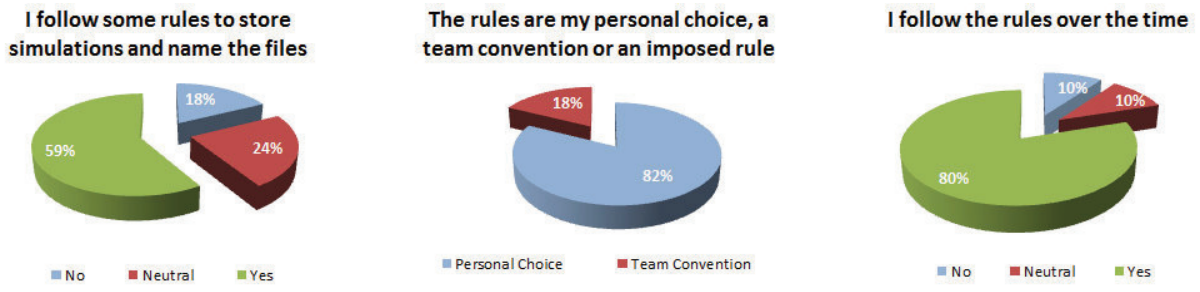


FIG. 2.4. *Rules to store the files on shared network folder (questions Q11, Q13, Q14).*

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.

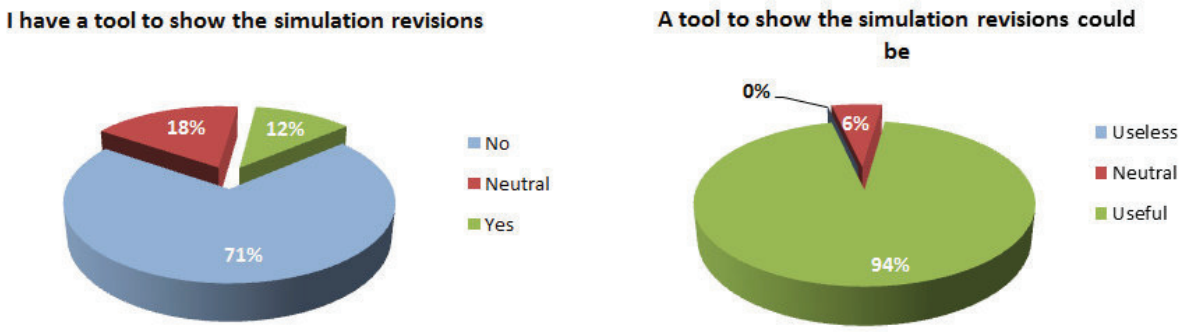


FIG. 2.5. *Version control (question Q20).*

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

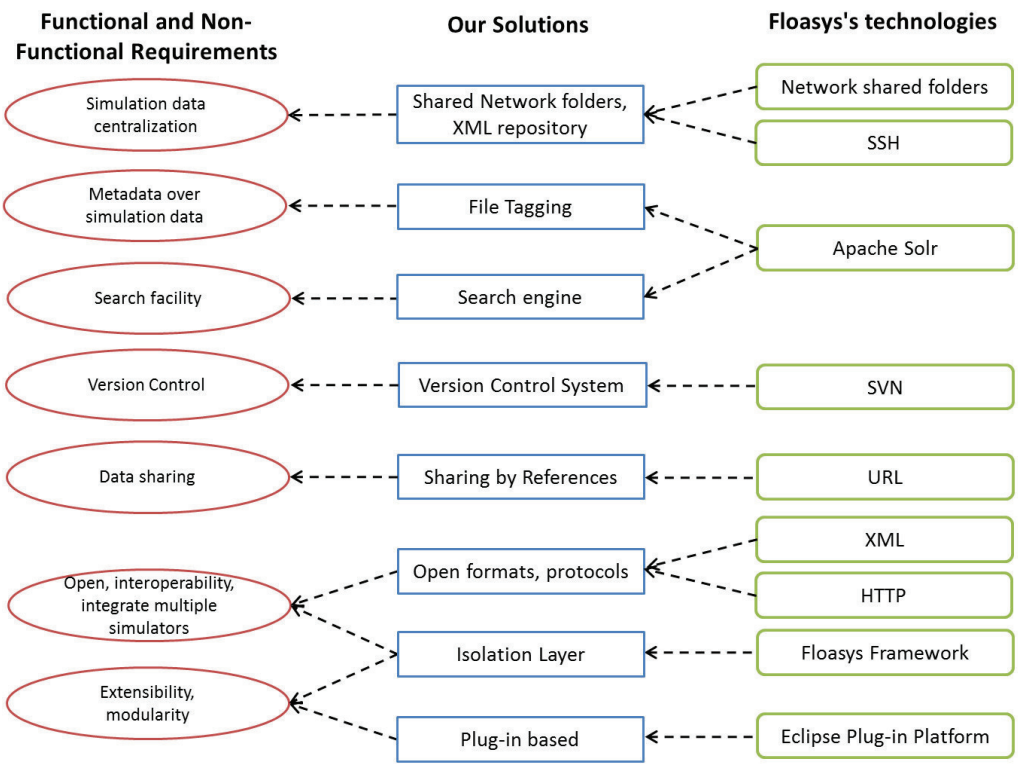


FIG. 2.6. *Mapping among Stakeholders' requirements, solutions and prototype technologies.*

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

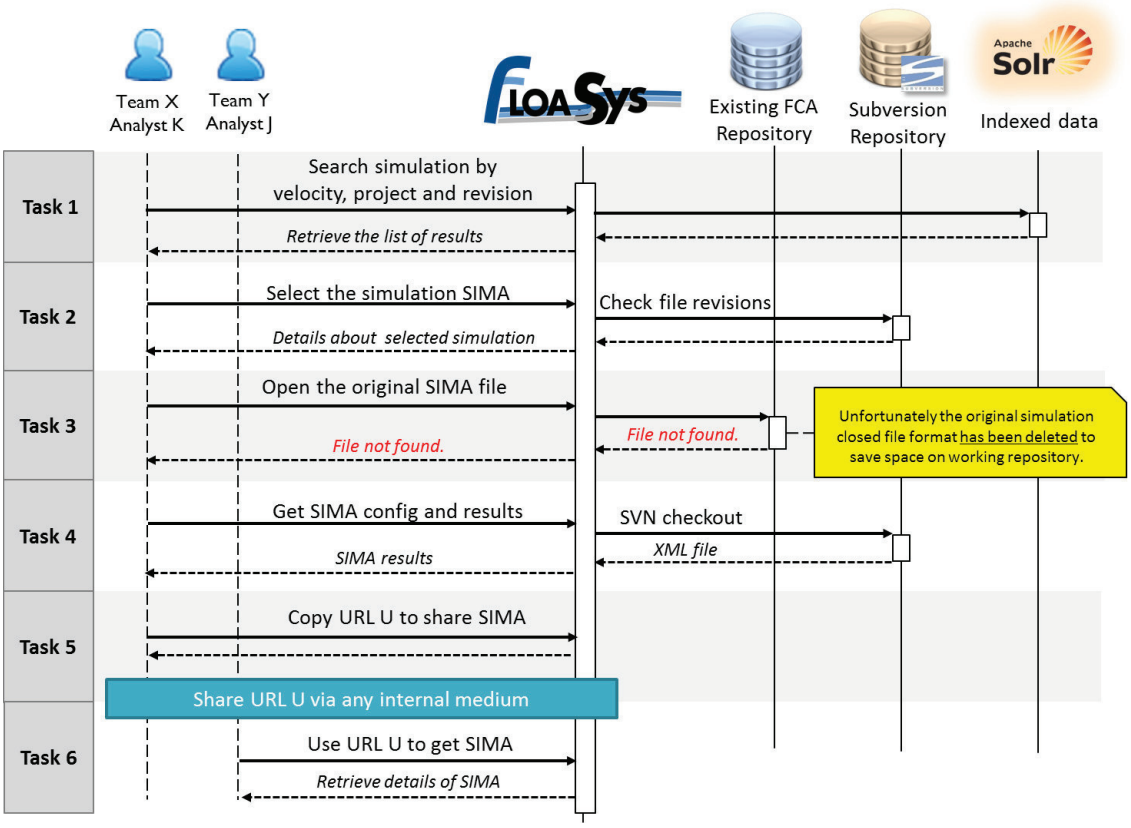


FIG. 3.1. *Example of a typical workflow supported by Floasys.*

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

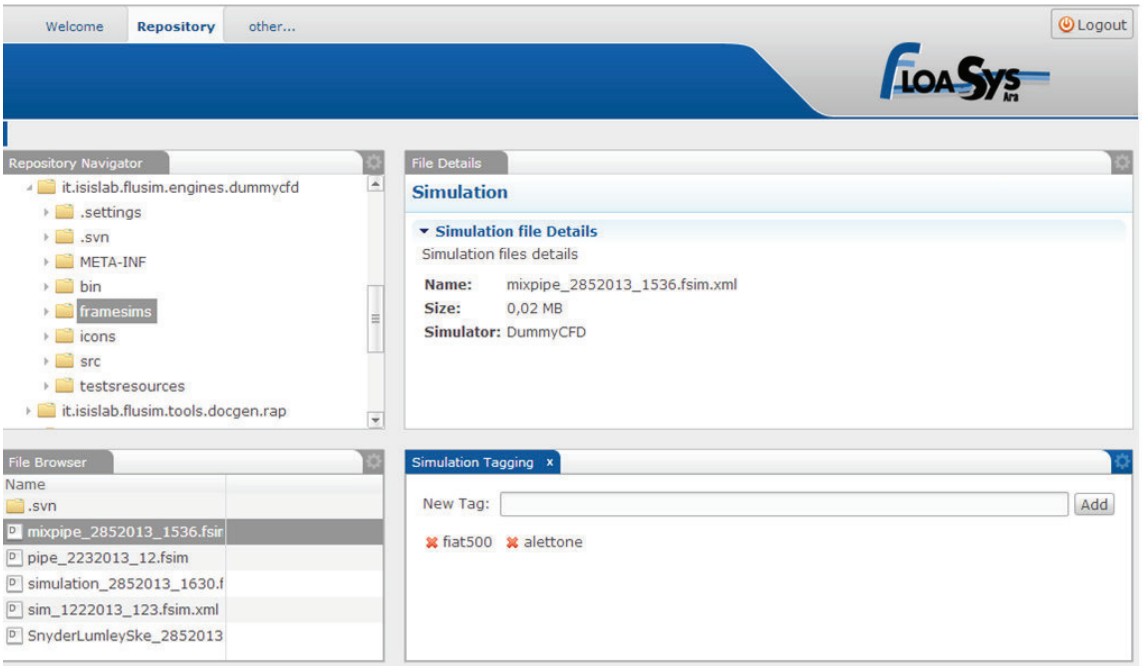


FIG. 3.2. *Repository tool to navigate a simulation repository and tag the resources (e.g., files).*

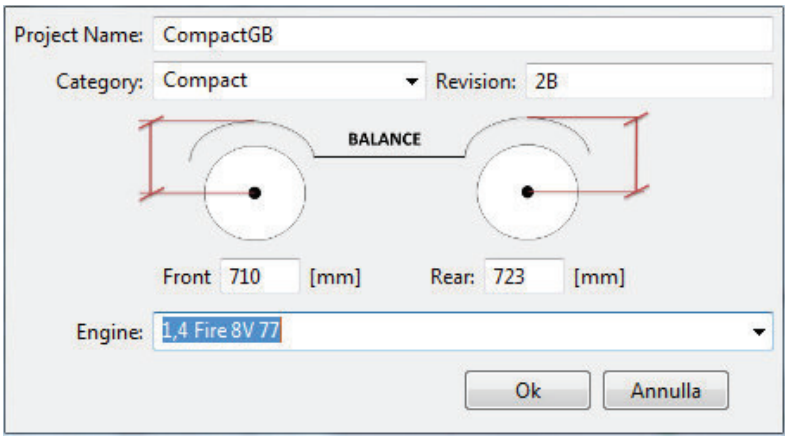


FIG. 3.3. *An example of structured data to store.*

(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

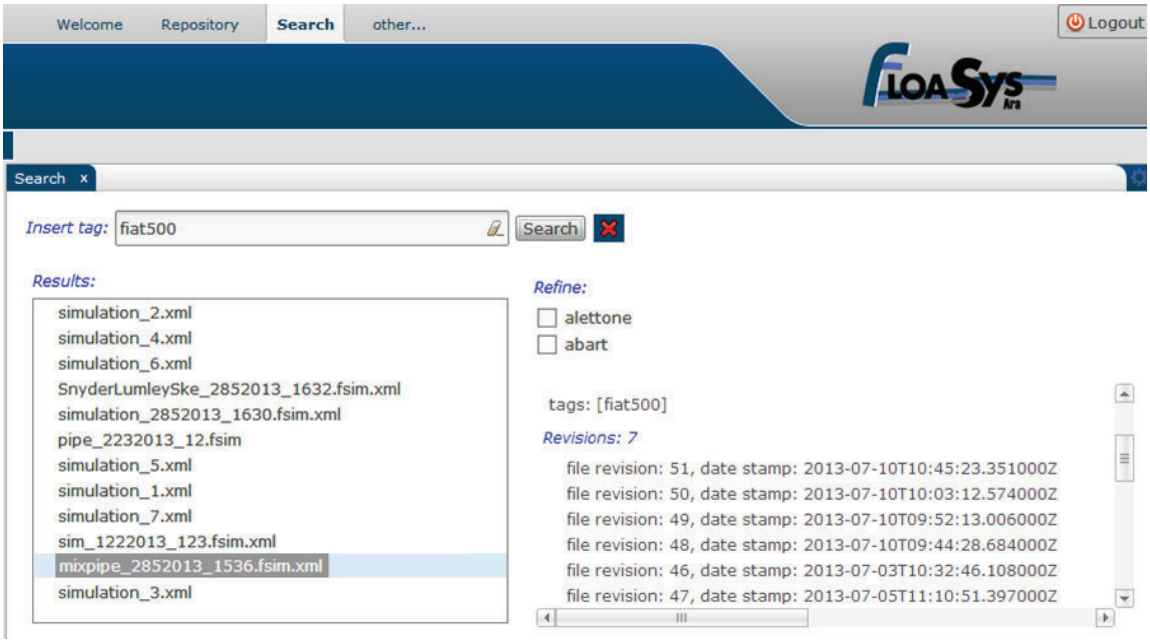


FIG. 3.4. *Search Tool*

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

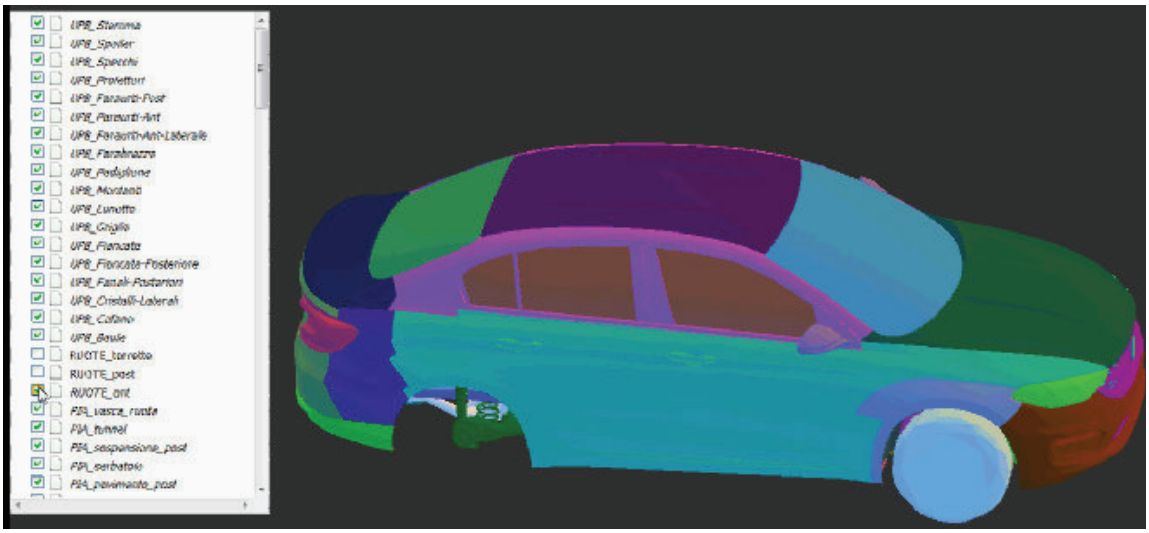


FIG. 3.5. *Floasys 3D model visualisation.*

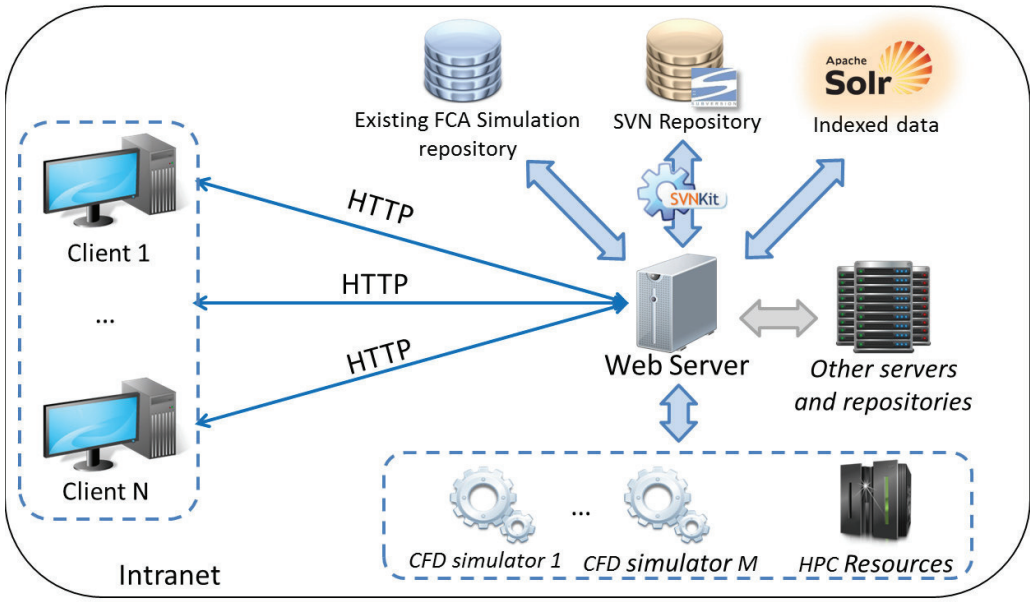


FIG. 4.1. *Floasys Client/Server Architecture.*

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/`

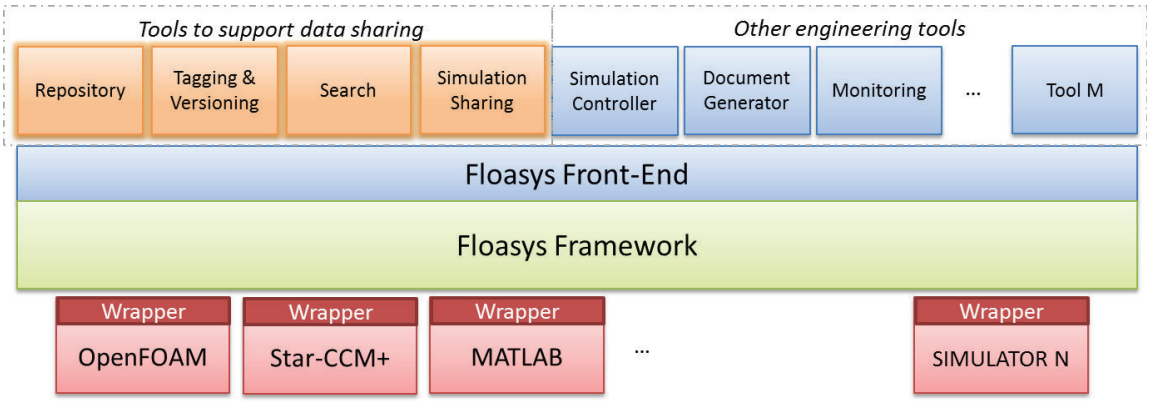


FIG. 4.2. *Floasys Server-side architecture.*

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.

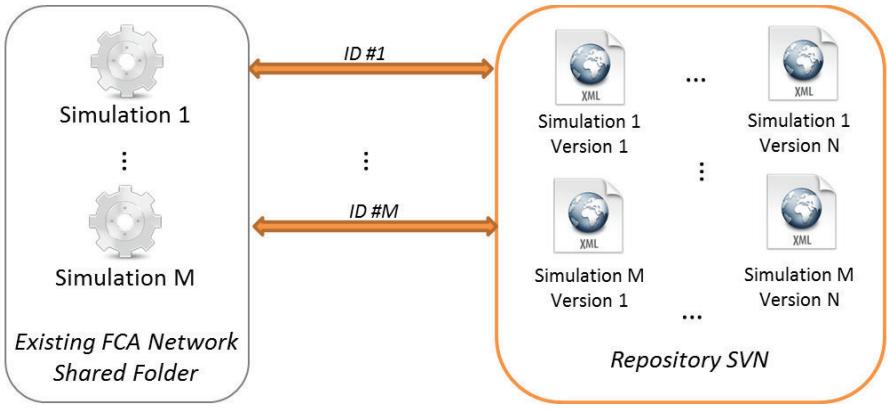


FIG. 4.3. *Simulation data versioning.*

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

Simulation Repository Visualisation and Exploration

Andrew Fish
School of Computing,
Engineering and Mathematics
University of Brighton, UK
Andrew.Fish@brighton.ac.uk

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

Donato Pirozzi
Dipartimento di Informatica
Università di Salerno, Italy
dpirozzi@unisa.it

Vittorio Scarano
Dipartimento di Informatica
Università di Salerno, Italy
vitsca@dia.unisa.it

***Abstract*—This paper describes a tool called ExploraTool to visualise, explore and graphically query large repositories of simulations. Instead of starting with the empty list, ExploraTool provides an initial overview of the repository content, progressively grouping the simulations by their main attributes, such as brand, vehicle model, power source, engine type and so on. Users can interactively navigate the repository view through drill-down, roll-up and rearrangement operations. In this way, using the ExploraTool, simulation analysts can visualise, explore and filter large repository of simulations as well as select groups of simulations to compare their performances.**

***Keywords*—Data Visualisation, Data Exploration, Simulation**

I. Introduction

Nowadays, industries and researchers extensively run simulations and experiments to design their products. In the automotive, industrial equipment, high-tech, aerospace and defence sectors [1], industries perform computer numerical simulations to design their product facing time-to-market, high quality and cost down pressures [1]. For example, automotive industries use Computational Fluid Dynamic (CFD) simulations to design the external vehicle aerodynamics or the internal air-conditioning. Another example comes from the engine design: researchers and industries have real engine test-beds that run for hours collecting sensor data like pressures, temperatures, and torque forces.

Simulation repositories usually store huge amounts of data for years. For instance, in large manufactures like Fiat Chrysler Automobiles, each analyst performs at least one hundred simulations per year [2], and there are many analysts working over years. This has generated a large, valuable repository of assets. In addition, analysts typically deal with simulations that are at least ten gigabytes each [2]. This gives an idea of the large quantity of data to manage within these repositories and the difficulty in having a clear idea of what they contain. Simulation Analysts, as well as Experiment Analysts, need to clean, analyse and compare the collected results as well as get insight into the data repository. Sometimes, specific phenomenons need to be understood. For instance, if a particular event in an engine experiment run occurs sporadically, then the analysts need to extract the input conditions for which such an event occurs (e.g., for which pressure values). For this reason there is a demand for software platforms able to collect, centralise, and get insight into information in a data repository, as well as to analyse and share results [3].

Based on our experience working closely a team of aerothermal CFD within Fiat Chrysler Automobiles, we identified the following three main requirements: (1) data collection, centralisation [1], and sharing [2] (2) data heterogeneity management, and (3) repository visualisation and exploration.

This paper focuses on the visualisation, exploration, and query of a large repository of simulations. The idea is to provide a graphical tool called ExploraTool to (1) get an overview of the repository content, (2) navigate the repository of simulations based on their properties, and (3) select and extract a set of simulations in order to compare their performance. The tool is actually usable for generic data exploration, thereby being usable to also explore repositories of experimental data, or any other big data sets.

This paper is organised as follows. Section II presents the state of the art on 2D space-filling visualisation techniques and existing tools that rely on them. Section III describes the ExploraTool features. Section IV describes the ExploraTool's architecture and the process used to transform the data read from the simulation repository to an interactive visualisation. The last Section V summarises the paper results, reporting the known tool limitations and the planned future works.

II. Related Works

The visualisation of large datasets has become really important because the classical list based widgets are not able to manage the large number of items, and also because it is practically impossible to show all the data available within a dataset. In this context, the 2D space-filling visualisation techniques aim to exploit all the available screen-space supporting the overview of the datasets, the opportunity to navigate the dataset and get more details on request. Generically speaking, the 2D space-filling approaches divide the available screen space recursively using a basic shape (e.g., rectangle, circle). In this way parent-child relationships are represented as nested shapes, and sibling nodes are represented as closest shapes at same depth.

Treemap was introduced by Shneiderman during 1990 to have a compact file system visualisation and be able to identify at a glance the directories that take up the most of the space on the hard drive. Then, treemap [4] has been extensively used to present intrinsically hierarchical data, providing an overview of an entire dataset at a glance. In treemap, every node in the hierarchy is represented as a rectangle with an area proportional to the node size. Parent-child nodes are represented as nested rectangles. Usually the navigation within the hierarchy is based on a drill-down with a left mouse click to go down in the hierarchy and a roll-up with a right mouse click to go up in the hierarchy. Over years, the treemap visualisation

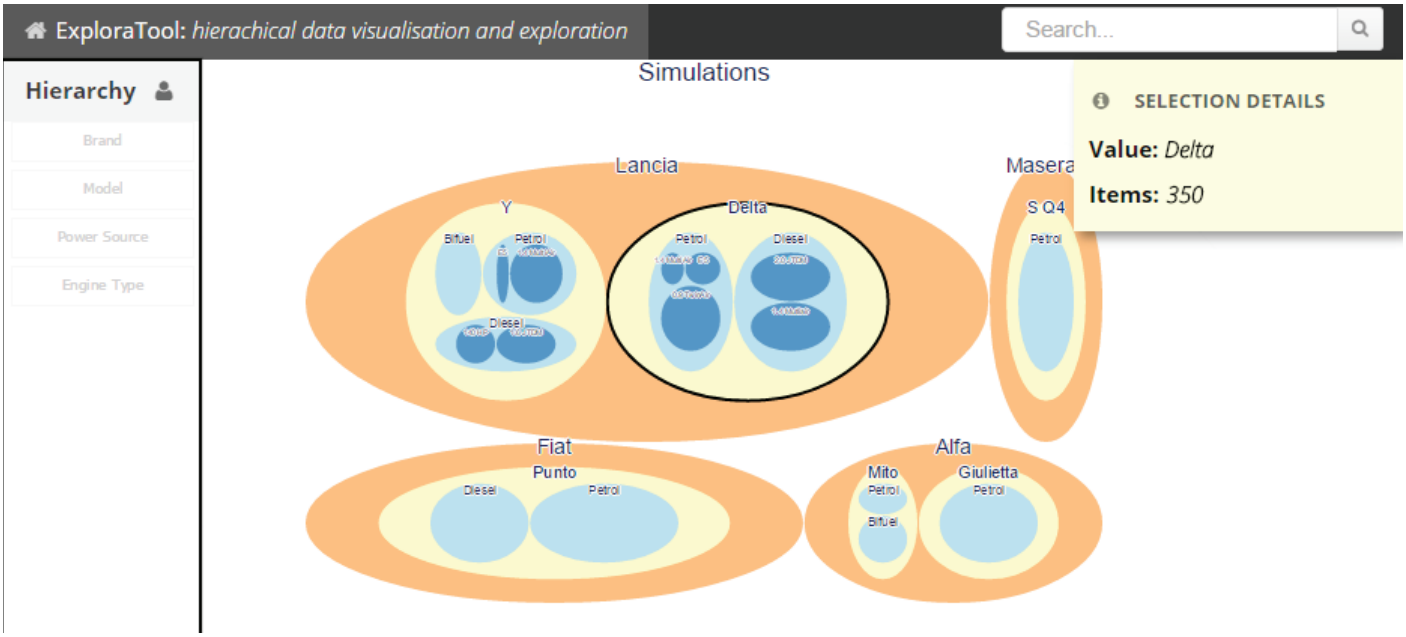


Fig. 1. The ExploraTool's Graphical User Interface. It shows an overview of the simulation repository through an initial hierarchy made by the following simulations' attributes: brand, project model, power source and engine type. The attributes' order is shown in the navigation bar on the left. Instead of starting from scratch the tool shows an initial overview, progressively grouping simulations by their main properties. For instance the picture groups simulations first by the brand (Lancia, Maserati, Fiat, and Alfa), then it further groups simulations by the vehicle model. The user can drill-down by directly clicking on any ellipse.

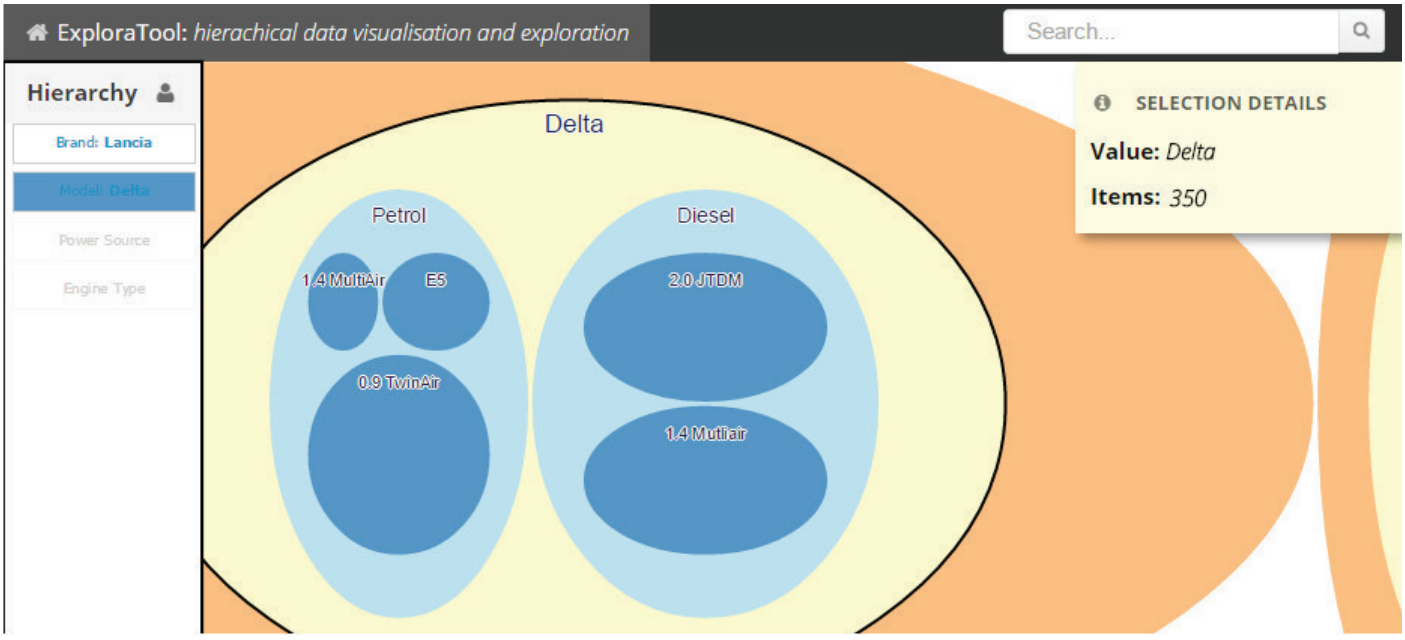


Fig. 2. The picture shows the result of the drill-down operation performed on the Delta category. Starting from Fig. 1, the analyst clicks on the ellipse "Delta". The ExploraTool smoothly enlarges the selected group, rendering a fast transition to the new view. If the user desires to go return back to the less detailed view, he/she can click directly on the external "universe" white space in order to perform a roll-up operation, thereby returning to the initial view as shown in Fig. 1.

approach has been used to visualise different hierarchical data, such as inherently hierarchical organisation structures [5], file systems [6], Usenet newsgroup [7] and so on. Well-known treemap drawbacks are the hierarchy discernment [8] and the fact that *the position of the mouse pointer designates an entire branch of the tree* [9] because *each point belongs to a single leaf node but also to all its ancestors* [9]. Of course, one of their advantages is the use of the all available 2D space.

Ellimap [8] is another type of 2D space-filling visualisation approach. It uses ellipses instead of rectangles to represent the nodes. In this way, there is always space between ellipses, both nested ellipses and adjacent ellipses (i.e., sibling nodes in the hierarchy). According to Otjacques at al. [10], the use of ellipses with their extra space improves the hierarchy discernment compared to the visualisation based on rectangles.

This paper exploits the ellipmap visualisation technique to explore large repository of simulations within Fiat Chrysler Automobiles (FCA). Until now, the ellimap has always been used coupled with other classical visualisation widgets like tree widget [8]. Here, we explore the repository of simulations directly through the ellimap, integrating a vertical navigation bar to track the user position in the hierarchy during the navigation. In addition, in this work we exploit the natural extra space between the ellipses in order to provide a hierarchy navigation facility in which the user points directly to the target shape and interacts with the left mouse click.

III. EXPLORATOOL FEATURES

This section describes ExploraTool and its features. Instead of starting from scratch with an empty screen without results, the tool shows an initial overview of the dataset filling all the 2D screen available space. Starting from this initial view, the user can navigate the simulation repository through an hierarchical structure made by nested groups of simulations. The hierarchical structure is created by grouping simulations by their attributes. The tool's graphical user interface (Fig. 1 and 2) has a central view to show graphically the simulations available within the repository. The tool shows data using the ellimap [8] visualisation technique, a 2D space-filling approach that uses ellipses as basic shapes to represent sets of simulations. As shown in Fig. 1, the external white space is the universe that represents the set of all simulations within the repository. The universe of simulations is further divided into subsets represented as ellipses. Each ellipse area is proportional to the number of items that it represents. The ExploraTool shows an initial overview of the dataset displaying the simulations by brand, project model, power source and engine type. This default initial sequence of attributes is based on the feedback provided by analysts in Fiat Chrysler Automobiles [2].

The user can obtain additional details on each group of simulations (ellipse) by hovering the mouse cursor over it. The tool shows the additional information, such as the number of items in a yellow box on the top-right (see Fig. 1). This space can be used in the future to provide aggregated statistics about the shown group of simulations.

The user can navigate the hierarchy through an *in-depth navigation* based on the drill-down and roll-up operations. On the left, the tool has a vertical navigation hierarchy bar that has multiple aims: (1) it gives an overview of the hierarchy, (2) it shows the current depth during the simulation repository navigation supporting the user orientation [11], and (3) it allows hierarchy rearrangement by swapping the levels.

The tool shows exactly r levels of the hierarchy. Actually, the default value for this parameter r is decided at configuration time and it can be changed changed via the user preference functions. Of course, the trade-off is between the amount of data categories displayed on the screen-space and the computational efficiency to extract the relevant hierarchy from the repository of simulations.

The ExploraTool renders the hierarchical data in a *range traversal* [12] manner: each time only r levels of the hierarchy are rendered on the screen. This allows one to have a clean visualisation without displaying too many shapes on the screen. The number of levels displayed can be changed at configuration time. When the number of levels to show is exactly equal to one ($r = 1$) the render is called *level traversal* [12] giving an overview of the nodes at a specific level. When the number of levels is greater then one ($r > 1$) the tool gives an overview of the data at a specific level plus additional details about the lower levels in the hierarchy.

A. Data Exploration: in-depth navigation

The user can further explore the simulation repository through the *in-depth* navigation [9] based on two basic operations: drill-down and roll-up. **Drill-down** occurs when a user has identified a potentially interesting group of simulations and he/she wishes to explore further details of this group, and so he/she clicks on an ellipse to obtain more details. Every time the user drills down in the hierarchy by one level, ExploraTool loads further data showing more nested ellipses. ExploraTool shows multiple nested ellipses, so the user can drill-down one level at time or multiple-levels in one step by clicking on the internal nested ellipses. **Roll-up** is the opposite operation to drill-down. When the user wants to have a global dataset view he/she goes up in the hierarchy by clicking on the container ellipse. Every time the user drills down in the hierarchy, he/she is effectively performing a refinement of the query, filtering all of the simulations in the repository.

All the operations provided by the ExploraTool rely on the direct manipulation [13] principle introduced by Shniderman. It concerns the direct interaction and manipulation of the rendered objects. The use of ellipses as basic shapes guarantee that there will be always space between sibling ellipses at same level and among nested ellipses. In this way every operation is performed by the user involves exactly the target shape. For instance, in order to drill down in the hierarchy, the user points and clicks exactly on the nested ellipse. In order to roll-up the user points and clicks exactly on the parent shape utilising the space between the parent and child ellipses (Fig. 2), which is always present. It is not the same for other 2D space-filling techniques. For instance, in the treemap visualisation technique both nested rectangles and adjacent rectangles have no space between them, so *the position of the mouse pointer designates a branch of the tree* [9] because *each point belongs to a single leaf node but also to all its ancestors* [9]. Finally, in the ExploraTool, to obtain the list of simulations within a specific ellipse the user can click directly on the target ellipse.

B. Hierarchy Attributes Rearrangement

ExploraTool starts with an initial hierarchy built on a default ordering of the attributes. The initial ordering is shown in the navigation bar (left-side of Fig. 1). This initial attribute ordering has been defined by end-users and this is useful in order to have an initial hierarchy displayed on the screen. In our use case, the attribute ordering is $A = \{brand, project\ model, power\ source, engine\ type\}$ where generically speaking A is the notation for a set of attributes.

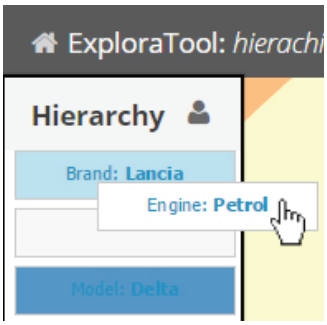


Fig. 3. Hierarchy rearrangement operation performed through the drag-and-drop of an attribute (facet) from its original position to a new slot. In this way, the user changes the order of the attributes, thereby updating the hierarchy.

The user can define an ordering of the attributes by interacting with the navigation bar. The user can drag and drop an attribute label (facet label) to move it from one slot to another one. By swapping two attributes that are on different levels in the navigation bar, the hierarchy updates showing the simulations in a different way. In this way, the ordering of the attributes is selected by the user according to his/her query.

C. ExploraTool Tasks

ExploraTool supports the exploration of simulations data sets, enabling the analyst to easily answer questions such as: *how many simulations were performed for the vehicle Delta with engine Diesel 1.4 Multiair?* or *which simulations have been performed for the vehicle Alfa Giulietta?*. By using ExploraTool the analyst can query the data set through drill-down and roll-up operations; often they will select common simulation attributes, such as the vehicle brand, model, and engine in order to provide information about the simulations with those target features. Since the ellipse layout is area proportional, the user is provided with an immediate perception of the size of groups during their exploration. By hovering the mouse pointer over an ellipse, the user is also provided with the exact number of simulations for that group. Finally, the user can easily obtain the list of all simulations for that group.

IV. ExploraTool Software Architecture

This section describes the ExploraTool architecture and the technologies used for its implementation. The tool is based on a Client/Server architecture (Fig. 4). In order to explore the repository, analysts just open any of the web-browsers (e.g., Mozilla® Firefox®) installed on their workstations, targeting a specific Intranet URL. This allows zero-configuration on the client-side. Enterprises, for confidentiality reasons, prefer to run the system within the industry's boundaries. Therefore, the prototype has been deployed within the company Intranet.

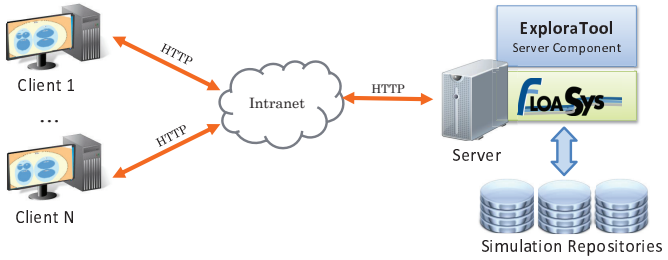


Fig. 4. ExploraTool prototype client/server architecture. ExploraTool is web-based, so analysts use the web-browser (clients on the left) to access to the server. On the Server-side (right part of the picture), ExploraTool accesses to the existing simulation repositories to retrieve the data and build the hierarchy chosen by the user.

However the overall architecture is designed using standard protocols (e.g., HTTP) to work properly both on Intranet and Internet settings.

On the server-side there are one or more simulation repositories. ExploraTool reads the data from the simulation repository, transforms them in open format and indexes them to improve their retrieval. In order to create the hierarchy, ExploraTool needs to access the items and the values of the simulation attributes, which are also called facets [14], [15]. A repository R of simulations is a collection of n items $R = \{s_1, ..., s_n\}$. ExploraTool reads the simulations' data and groups the attribute values together.

From a technological point of view, ExploraTool leverages from mainstream technologies. Clients exchange data with the server in JSON text format [16], [17] using standard Web protocols (e.g., HTTP). Clients are implemented using the open source JavaScript library *D3 Data-Driven Documents*[1] [18] and SVG. The server has been implemented using Java and uses the Floasys Framework API [3], [19] to retrieve the simulations stored within the repository.

A. Hierarchy building process

Figure 5 depicts the process used to extract data from a repository of simulations, build, and render the hierarchical visualisation using the elliptical-based 2D space filling approach. The ExploraTool back-end builds a partial tree data structure sent to the client that will transform it in an ellipse visualisation layout. The detailed steps are the following:

1) Tree data structure build: The ExploraTool back-end reads the data from the simulation repository to build a tree data structure with a level for each attribute. The tree building is an intermediate step and the ellimap layout is based directly on it. Each tree level will be a layer in the visualisation tool. Each node will be an ellipse in the visualisation. Fig. 6 shows an example of tree data structure and the resulting ellimap. In the example (Fig. 6) we assume that the ordering of the attributes is $< brand, project\ model, power\ source, engine\ type >$. The layer Brand in the tree has exactly two nodes (in orange colour) that have been represented by two orange ellipses labelled 1

[1]D3JS documention as well as the library download is available on the official web page http://d3js.org/

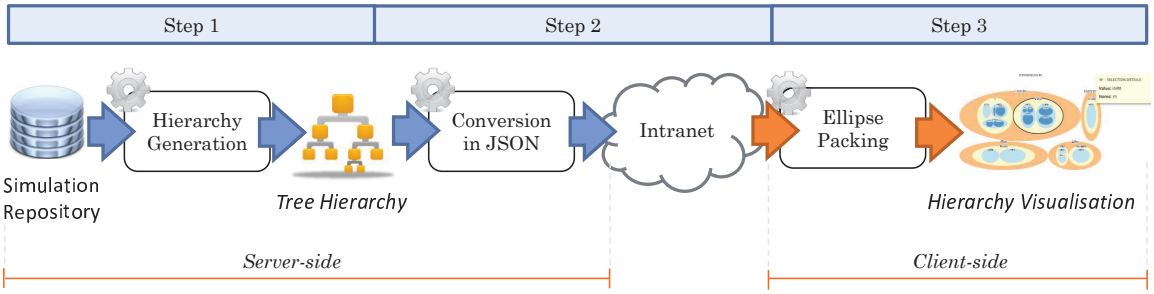


Fig. 5. Pipeline of transformation from the simulation repository to the visualisation on the client Web-browser. In order, the steps are: (1) reading of simulation data and creation of a tree data structure, (2) conversion of the tree data structure in JSON format, and transferring of the JSON data from the server-side to the client-side, (3) reading of the JSON data and packing of the ellipses within the available 2D space.

and 2 in the visualisation layout. A parent-child relationship between two nodes of the tree data structure will be two nested ellipses in the visualisation layout. Two sibling nodes will be two separated ellipses on the same level of the visualisation layout. The resulting tree will have at most r levels, mainly to limit the overall required computation time to load the dataset and build the tree data structure. ExploraTool progressively groups together the simulations with the same attribute value. This grouping is equivalent to a database SQL query that uses the `group by` statement [20]. In the example shown in Fig. 6, ExploraTool firstly groups the simulations by brand, then by project model, and then by power source.

The resulting tree data structure has a root that represents the universe of all items, in our case the simulations. Each level in the tree is an attribute like brand (Fig. 6). Each node is a value for the given attribute. For instance, at Brand level (Fig. 6) there will be the nodes with the values Lancia, Alfa, Maserati and so on. Each tree node has additional metadata, such as the number of simulations in the repository that have that value for the specific attribute.

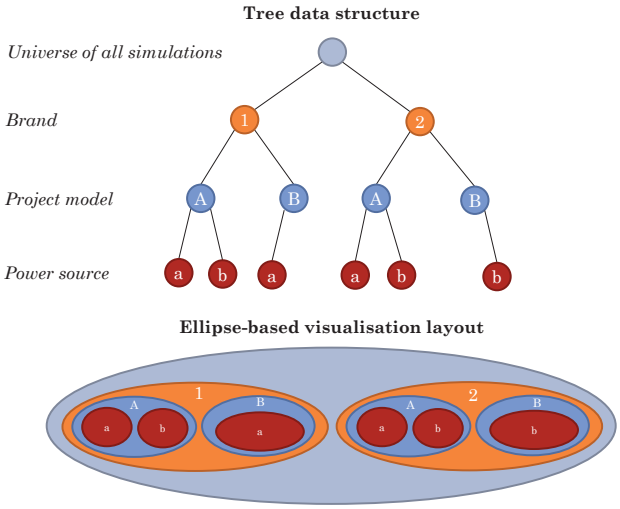


Fig. 6. The tree data structure built by the ExploraTool and the subsequent ellimap rendering. The ellipse layout is based directly on the tree data structure, in a manner so that each level in the tree is a layer of the ellimap, and each node of the tree is an ellipse.

2) Transferring of JSON data from the server to the client: The tree data structure is transformed in JSON format and sent to the client. Fig. 7 shows a partial example of JSON source code. The first part contains the available categories shown to the end-user in the navigation bar. The second part contains the simulation data grouped together. In order to be concise, Fig. 7 shows only the first layer.

```
{
    "categories": [
        {
            "name": "Brand", "id": 0
        },
        {
            "name": "Model", "id": 1
        },
        {
            "name": "Power Source", "id": 2
        },
        {
            "name": "Engine", "id": 3
        }
    ],
    "root": {
        "name": "simulations", "size": 1100,
        "children": [
            {
                "name": "Lancia", "size": 600,
                "children": []
            },
            {
                "name": "Fiat", "size": 250,
                "children": []
            },
            {
                "name": "Alfa", "size": 150,
                "children": []
            },
            {
                "name": "Maserati", "size": 100,
                "children": []
            }
        ]
    }
}
```

Fig. 7. A partial example of JSON data format transferred from the server to the client. In the first part, it contains the categories. The second part is the hierarchy. The size is the number of simulations below the specific branch in the hierarchy and is used to generate the area proportional ellipses.

3) Ellipse Packing for Data Visualisation: The client receives the tree data structure in JSON format and transforms it into the visualisation layout based on ellipses. Over the years

many algorithms have been proposed to pack rectangles for treemap. Some of them are: the slice-and-dice, cluster treemap, squarified [21], ordered [22], and strip [23] algorithms. In ExploraTool the layout algorithm is a modified version of the strip treemap. We consider firstly rectangles and then we replace rectangles with ellipses. In order to pack nested ellipses (children) within an existing ellipse (parent), our algorithm circumscribes a rectangle in the parent ellipse and recursively applies the strip packing algorithm for the children rectangles, that will be replaced by ellipses.

V. Conclusions and Future Works

In this paper we described a tool called ExploraTool to visualise, explore and query large repositories of simulations. Large companies like FCA have large repositories of simulation data and they must be sure that analysts have access to previously generated data. ExploraTool provides an overview of the repository content, fostering its exploration. The analysts can visually and interactively query the data set view through drill-down, roll-up and rearrangement operations. The idea behind our tool is generic and can be easily used with a repository of experiments as well as other types of data sets. In order to do this, it is necessary to identify the common and interesting data categories, and build the relative hierarchy that ExploraTool will render.

As future work on the ExploraTool we wish to improve the layout algorithm to avoid thin ellipses, thereby improving the visualisation overall aesthetic. Of course, the residual space among nested ellipses can be reduced, but this could impact upon user hierarchy perception and discernment. In addition, we are planning an evaluation study [24] to analyse the tool usability and user satisfaction when interacting with it, by utilizing a well-known questionnaire [25], [26]. Furthermore, it will be interesting to generalise the tool and use it on a generic repository like a catalogue of products and compare how users will perform with it as compared to using different types of visualisation techniques, like a classical list of results, Treemap, FacetMap [15], etc. Within the industrial context an interesting issue to explore is the data authorisation problem, where a user may only have access to a specific subset of simulations within the repository.

Acknowledgements. The authors gratefully acknowledge the help and the support of Comprensorio CRF Elasis Pomigliano (Fiat Chrysler Automobiles) and the interactions with Aerothermal CFD team. The authors also gratefully acknowledge the support of the CEREEV (Combustion Engine for Range-Extended Electric Vehicle), a European territorial cooperation project (grant number 4224), part-funded by the European Regional Development Fund (ERDF) through the INTERREG IVA France (Channel) England Programme.

References

[1] M. Boucher and C. Kelly-Rand, "Getting Product Design Right the First Time with CFD," *Aberdeen Group: May*, 2011.

[2] C. Gargiulo, D. Malandrino, D. Pirozzi, and V. Scarano, "Simulation data sharing to foster teamwork collaboration," *Scalable Computing: Practice and Experience*, vol. 15, no. 4, pp. 309–329, 2014.

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

[4] B. Johnson and B. Shneiderman, "Tree-maps: A space-filling approach to the visualization of hierarchical information structures," in *Proceedings of the IEEE Conference on Visualization*, 1991, pp. 284–291.

[5] P. Demian and R. Fruchter, "Finding and understanding reusable designs from large hierarchical repositories," *Information Visualization*, vol. 5, no. 1, pp. 28–46, 2006.

[6] B. Shneiderman, "Tree visualization with tree-maps: 2-d space-filling approach," *ACM Transactions on graphics (TOG)*, vol. 11, no. 1, pp. 92–99, 1992.

[7] A. Fiore and M. A. Smith, "Treemap visualizations of Newsgroups," *Technical Report, Microsoft Research, Microsoft Corporation: Redmond, WA*, 2001.

[8] B. Otjacques, M. Cornil, and F. Feltz, "Visualizing cooperative activities with ellimaps: the case of Wikipedia," in *Cooperative Design, Visualization, and Engineering*. Springer, 2009, pp. 44–51.

[9] R. Blanch and E. Lecolinet, "Browsing zoomable treemaps: structure-aware multi-scale navigation techniques," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 13, no. 6, pp. 1248–1253, 2007.

[10] B. Otjacques, M. Cornil, M. Noirhomme, and F. Feltz, "CGDA new algorithm to optimize space occupation in ellimaps," in *Human-Computer InteractionINTERACT 2009*. Springer, 2009, pp. 805–818.

[11] C. M. Dal, S. Freitas, P. R. G. Luzzardi, R. A. Cava, M. A. A. Winckler, M. S. Pimenta, and L. P. Nedel, "Evaluating Usability of Information Visualization Techniques," in *Brazilian Symposium on Human Factors in Computing Systems*, 2002.

[12] N. Elmqvist and J.-D. Fekete, "Hierarchical aggregation for information visualization: Overview, techniques, and design guidelines," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 16, no. 3, pp. 439–454, 2010.

[13] B. Shneiderman, "Direct manipulation: A step beyond programming languages," in *ACM SIGSOC Bulletin*, vol. 13. ACM, 1981, p. 143.

[14] K.-P. Yee, K. Swearingen, K. Li, and M. Hearst, "Faceted metadata for image search and browsing," in *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 2003, pp. 401–408.

[15] G. Smith, M. Czerwinski, B. R. Meyers, G. Robertson, and D. Tan, "FacetMap: A scalable search and browse visualization," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 12, no. 5, pp. 797–804, 2006.

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

[17] X. Chen and K. Kasemir, "Bringing Control System User Interfaces to the Web," *TUPPC078, ICALEPCS*, vol. 13.

[18] M. Bostock, V. Ogievetsky, and J. Heer, "D3 data-driven documents," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 17, no. 12, pp. 2301–2309, 2011.

[19] C. Gargiulo, D. Pirozzi, and V. Scarano, "An architecture for CFD workflow management," in *Proceedings of the 11th IEEE International Conference on Industrial Informatics (INDIN), Bochum, Germany, July 29-31*, 2013, pp. 352–357.

[20] R. Elmasri, *Fundamentals of database systems*. Pearson Education India, 2007, vol. 2.

[21] M. Bruls, K. Huizing, and J. van Wijk, "Squarified Treemaps," in *Data Visualization 2000*, ser. Eurographics, W. de Leeuw and R. van Liere, Eds. Springer Vienna, 2000, pp. 33–42. [Online]. Available: http://dx.doi.org/10.1007/978-3-7091-6783-0_4

[22] B. Shneiderman and M. Wattenberg, "Ordered treemap layouts," in *Information Visualization, IEEE Symposium on*. IEEE Computer Society, 2001, pp. 73–73.

[23] B. B. Bederson, B. Shneiderman, and M. Wattenberg, "Ordered and quantum treemaps: Making effective use of 2d space to display hierarchies," *AcM Transactions on Graphics (TOG)*, vol. 21, no. 4, pp. 833–854, 2002.

[24] J. Lazar, J. H. Feng, and H. Hochheiser, *Research methods in human-computer interaction*. John Wiley & Sons, 2010.

[25] "Computer System Usability Questionnaire," Dec. 2014. [Online]. Available: http://oldwww.acm.org/perlman/question.cgi?form=CSUQ

[26] "Questionnaire for User Interface Satisfaction." [Online]. Available: http://oldwww.acm.org/perlman/question.cgi?form=QUIS

Visual Exploration System in an Industrial Context

Andrew Fish *Member, IEEE*, Claudio Gargiulo, Delfina Malandrino, Donato Pirozzi, and Vittorio Scarano

***Abstract*—This paper describes ExploraTool a new interactive tool to visually explore data from multiple repositories. The tool has been applied in a real setting to explore CFD simulation data and obtain new insights into the space of simulations. The inclusion of free exploration, filtering operations and chart generation provides a quick method for performance comparisons. The paper proposes an algorithmic means of processing input in the form of tabular data sets, generating a plausible hierarchical structure over metadata categories which is used to initialize the visualisation together with interactions methods to explore, select and compare sets of simulation data. This paper also reports on the Evaluation Study performed involving 24 engineers over two distinct locations from a large automotive manufacturer, to evaluate the usability and the overall user satisfaction with the tool. Participants rated the tool as intuitive, useful and effective.**

***Index Terms*—Exploratory Search System, Information Retrieval, Data Visualisation, User Interfaces, Multirun Simulations, Industrial User Evaluation Study, Dataset processing.**

I. Introduction

With the increased availability of computing power and storage capacity, medium and large enterprises can continuously collect data along the product design process (PDP). Enterprises have large, or even big, data repositories of potentially valuable and strategic assets. In order to boost their competitiveness it is becoming vital to obtain insights into the repositories, exploring their content and extracting new knowledge. Exploring repositories to find valuable information is difficult [1] because data management systems use traditional list based widgets, displaying only a small data portion compared to the repository size, so researchers are exploring the use of other visualisations (e.g., treemap). Without an adequate exploring system, data remains in the repositories without exploitation.

Automotive manufacturers have multiple repositories in which to store: simulation data generated by different simulator software, experimental data continuously collected from the Wind Tunnel infrastructure, and competitors' product data performances accessible through subscription to third part services (e.g., A2Mac1). These repositories are independent, and in order to compare the various data sets across repositories, analysts often have to manually access each of them. Due to this scenario, and market competition, an increasing desire to facilitate easy exploration of repositories, select groups of data, aggregate them, and perform comparisons over the data via an intuitive interface is evident. In contrast, software engineering companies often focus only on the simulation functionality (e.g., Computer Aided Engineering functionality), not addressing the actual industrial need to explore their repositories, and compare data with other simulators' results.

This paper introduces a web-based tool, called ExploraTool that enables the visual and interactive exploration of data sets by item properties. Using ExploraTool, analysts can select multiple groups of simulations or single simulations, and compare their relevant simulations' performances.

ExploraTool has multiple benefits: 1) it provides a visual overview of the repository content, grouping items together by their properties (facets) and visualising them using nested ellipses covering all the available 2D space screen; 2) compared to the traditional list-based results, during the navigation it gives a clue on the overall repositories content, which would be non-trivial with traditional file systems organised by directories and not by item properties; 3) the visualisation based on facets helps the user to explore and discover item properties to further investigate and filter the items by selecting ellipses; 4) ExploraTool reads data from multiple repositories (i.e., multiple network file systems, external sources), allowing the Analysts to select simulations from different sources; 5) it is extensible to be able to also explore experimental data and competitors' performance data, integrating together different data sources to have an all-in-one workbench.

The paper is organised as follows. Section II discusses the related work in terms of Exploratory Search Systems and visualisation approaches. Then, the paper evolves and combines four aspects that are the main paper contributions: 1) the ExploraTool idea, its graphical user interface and features described in Section III; 2) the mathematical background and an algorithm to process as input any kind of tabular data set and impose on it a plausible hierarchical structure over metadata categories that ExploraTool is able to visualise, enabling data set exploration, described in Section IV; 3) a field study performed within a large automotive manufacturer, which involved 24 engineers, to evaluate the tool usability and overall user satisfaction, which is essential for industrial adoption described in Section V; 4) an overview the future extensions obtained through both the field evaluation and the stakeholders' interviews, along with lessons learnt from the process and the potential usage of the tool and approach in other industrial sectors, described in Section VI.

II. Related Work

This paper focuses on the exploration of industrial repositories that store thousands of simulations performed over years.

Manuscript received July 1, 2015; revised October 9, 2015 and December 9, 2015; accepted January, 14, 2016.

A. Fish is with the School of Computing, Engineering and Mathematics, University of Brighton, UK (e-mail: Andrew.Fish@brighton.ac.uk)

C. Gargiulo is with FCA (e-mail: claudio.gargiulo@fcagroup.com)

D. Malandrino, D. Pirozzi and V. Scarano are with the Department of Computer Science, University of Salerno, IT (e-mails: dmalandrino@unisa.it, dpirozzi@unisa.it, vitsca@dia.unisa.it)

It aims to assist the industrial analysts with their need to gain insight, understand the content of the repositories and filter data to help perform comparisons. Instead of displaying a list of results, this work exploits the use of interactive visualisation to provide a visual indication of the repositories' contents, whilst supporting data filtering and selection tasks. Hence, exploratory search systems, visualisation approaches, and interaction techniques are related.

Classical *lookup search* [2], [3] is a query-response paradigm where the user poses a textual query, the system performs the retrieval and shows the results in a list-based widget. Their main drawback [3] is the difficult for users to memorise and master filtering and operators syntax (e.g., "and", "or" operator, etc.). *Exploratory search* [3] considers multiple iterations involving learning and investigation activities with higher-level goals (e.g., comparison, analysis, synthesis and evaluation). *Exploratory Search Systems* [2] aim to involve and engage users actively into the search process by providing human control over the seeking process. They aim to provide features to (re)formulate queries, giving information on the search space and clues for further possible search directions [4], allowing the constant exploration and filtering of retrieved results [3]. The presentation of query results is imperative to engage the user in the search process [5].

One popular technique to help users in deciding what to do next is the grouping of results [6]. Two main approaches exist: clusters and facets. The clustering approach [7] groups, often automatically, the query results according to similarity metrics. In the faceted approach, meaningful items' feature types are identified in advance, mostly manually, and labelled. Thus, *facets* are categories to characterise items in a collection. Each item is automatically enriched with multiple facets' labels. The query results can be grouped together based on these labels which form a categorisation, and can be used to further explore the space of results. According to White and Roth [8], exploratory search tools should "support querying and rapid query refinement" and "offer facets and metadata-based result filtering". The use of clusters or facets aids searchers with the query formulation that "significantly improves results" [5]. Flamenco [6] is a web-based system where the navigation is performed through the selection of hyper-links containing facets' labels. Relation Browser [9] is another example, it has two views, a list of facets and a cloud of facets labels; the user can filter result by choosing the facets. The system mSpace [10] uses a multi-column faceted browser for multimedia data. Carrot2 [11] is a web search engine that supports the navigation of web results through the selection of cluster hyper-links showed in a tree-based widget alongside of the list of results. In a testing with real users using Carrot2 [5], the clusters were useful in providing other relevant keywords to narrow the search and for serendipity search. Other systems that relies on the clusters of results have been introduces, like Vivisimo and SnakeT [12]. AcquaBrowser Library [13] does not show the cluster names in a listed way, but introduces a fluid, attractive and interactive word cloud visualisation, clicking on a word the user can refine his/her search. With the introduction of touch mobile devices, classical list-based result presentation poses challenges for the interaction with Exploratory Search Systems to refine searches [4].

From the GUI point of view, exploration systems are now exploiting visualisations to graphically display groups of items to provide an initial overview, permitting interactions to formulate queries and update the visualisation. This interaction "*overview first, zoom and filter, then details-on-demand*" is known as information seeking mantra [14]. Interactions can be grouped into: overview, navigation, and manipulation operations [15], and optionally, based on the application domain, interactions to compare the results of the query. FacetMap [16] is based on the facet concept and exploits a 2D visualisation to support dataset exploration. Facets are represented by ellipses and navigation is performed by clicking on the ellipses. ExploraTool presented in this paper is also based on the facets concept, but in order to scale on the number of facets, it organises them hierarchically. As example of a manipulation operation, the "*hierarchy manipulation*" term refers to a set of interactive operations performed on a hierarchical visualisation to directly and interactively change, re-order, move or copy its items; for example re-ordering through the drag-and-drop of hierarchical items. Lutz et al. [15] described many types of hierarchy manipulations by diagrammatically depicting their use and effects. ExploraTool has a hierarchical manipulation operation to change the order of the visualised facets.

In the visualisation field, many alternatives have been proposed to overcome the limits of classical list-based widgets of items, due to the impossibility of showing all items of a large dataset. One popular approach is the 2D space-filling visualisation technique that aims to exploit all of the available screen-space to show the dataset content. This technique divides the available screen space recursively using a basic shape (e.g., rectangle, circle). In this way parent-child relationships are represented as nested shapes, and sibling nodes are represented as neighbouring shapes at same depth. The most popular 2D space-filling visualisation is Treemap, introduced by Shneiderman during 1990 to provide a compact file system visualisation and to be able to identify at a glance the directories that take up the most of the space on the hard drive. Treemap has been extensively used to visualise intrinsically hierarchical data [17], [18], providing an overview of an entire dataset at a glance. Ellimap [19] is another type of 2D space-filling visualisation approach, which uses ellipses instead of rectangles to represent the nodes. Usually, shapes of 2D space filling visualisations are area proportional to a given metric (e.g., the number of items), visually giving an overview on this value. ExploraTool, presented in this paper, is a visual exploration system that exploits the use of ellimap layout with additional interaction functionalities (e.g., drill-down, roll-up), and provides dataset overview through the visualisation of facets, which are organised hierarchically on multiple levels. In this way, ExploraTool supports the constant exploration and filtering through selection of facets [8].

ExploraTool needs to interoperate with multiple repositories that can store data in different formats. In the CFD field most simulator software strategically use closed and proprietary data formats to make it expensive for customers to change their software products [20], [21]. ExploraTool relies on the plug-in based Floasys Architecture [22], [23] to be able to com-

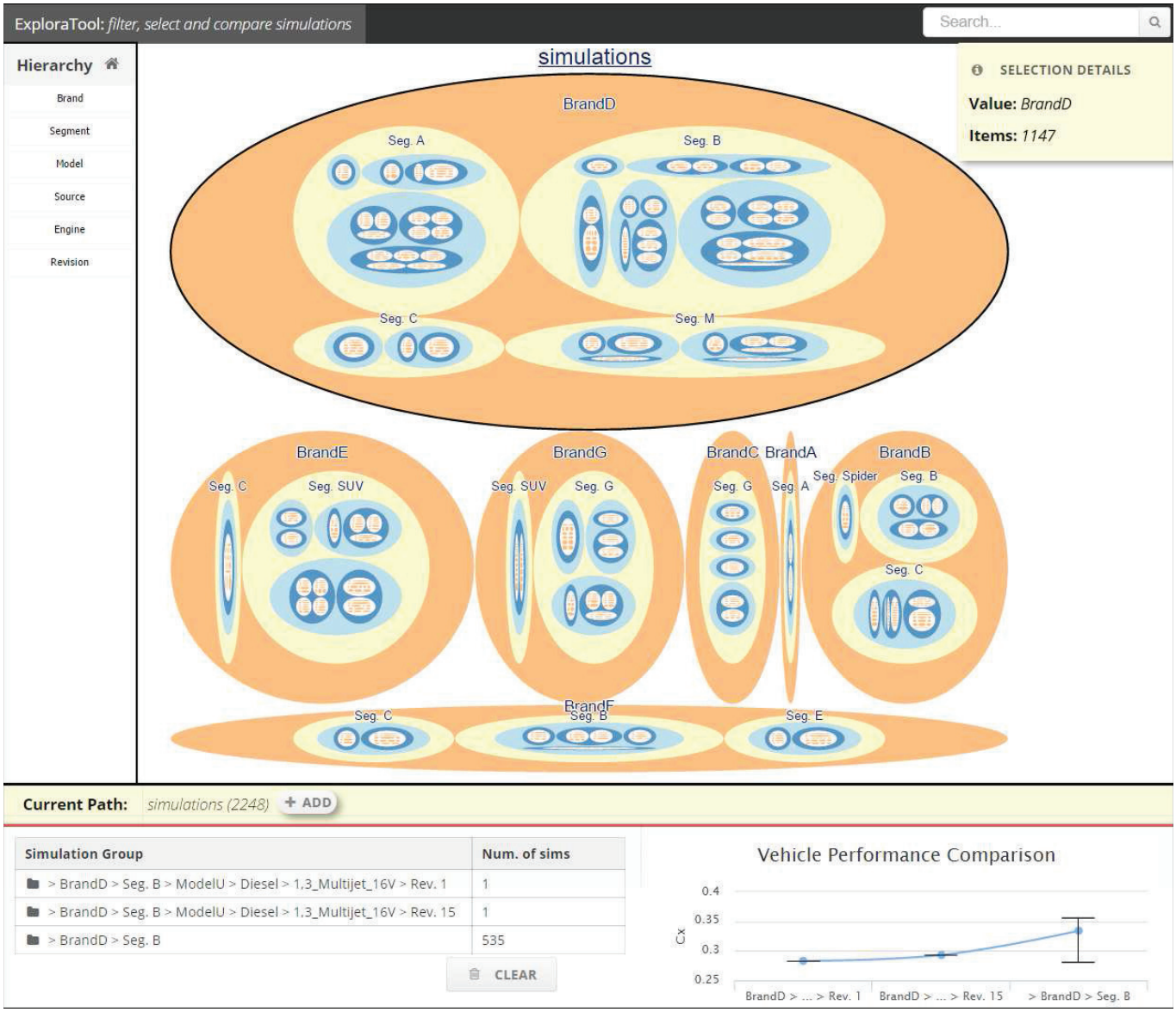


Fig. 1. ExploraTool's Graphical User Interface (GUI). The central overview shows the simulation data set grouped together via their relevant attributes: *brands, segments, models, power sources, engines, and revisions*. The ellipses depict groups of simulations and the prioritised order of the attributes is indicated in the hierarchy column on the left. The user can filter the data set by clicking on any group of simulations (drill-down). If the user desires to return to the previous view, he/she can click on the container ellipse (roll-up). The navigation bar (underneath the main view) shows the path followed during the exploration; in the example, the path is the universe overview, labelled as "*Current Path: simulations*". If user desires a comparison of the performances of one group he/she can click on the "+ *Add*" button which adds the current group to the table below and the chart automatically updates accordingly.

municate with many software and data sources. Floasys has an API interface which acts as isolation layer, and a common data format to communicate with external client applications. In order to perform data integration he2014integration the API interface is implemented by other specifically designed software modules which read data from closed and open sources. Floasys architecture processes data from interoperable standards and protocols like CFD General Notation System (CGNS), and can be extended to other interoperable standards.

III. ExploraTool Features

This Section introduces the ExploraTool's features: dataset overview, exploration and vehicles' performances comparisons. The user task is to select a single simulation, or groups of simulations, in order to compare the vehicles' performances (e.g., aerodynamic performances) via an appropriate chart.

A. Data set Overview

The ExploraTool's GUI (Fig. 1) has a central view which graphically depicts the simulations available within the repository, and a chart on the bottom to compare the selected vehicles' performances (e.g., aerodynamic performances). ExploraTool starts with an initial overview of the data set where the items are progressively grouped together by their main relevant attributes. The screenshot shows the simulations progressively grouped by Brands, Segments, Models, Power Sources, Engines, and Revisions. Brands and model names shown in Fig. 1 and throughout the paper have been anonymised,

replacing them with artificial names. The hierarchy on the left shows the default initial ordering of attributes, based on the feedback provided by analysts in a large automotive manufacturer [22], but it can be changed any time by the user. Additional details on each group are requested by hovering the mouse cursor over the corresponding ellipse, causing additional information to be presented in a yellow tool tip box (see the top-right of Fig. 1). The same tool tip box could be used to show additional statistical information (such as the average, minimum and maximum drag-coefficient Cx values), as suggested by users during the Usability Study (Section V). ExploraTool tries to exploit all of the available screen space using ellimap [19]. Each group of simulations is a set depicted by an ellipse. In Fig. 1, the external white space represents the universe of all simulations within the repository. This is divided into subsets, depicted as ellipses, generating a nested-ellipse layout. For instance, the ellipse labelled *"BrandD"* in Fig. 1 represents the set of all simulations enriched with facet BrandD. The ellipse labelled *"Seg. A"* nested in the ellipse *"BrandD"* represents the set of all simulations of Seg. A with BrandD. In order to reduce cluttering, the tool shows only labels for two levels at a time. Each ellipse's area is chosen to be proportional to the number of simulations in that group, so that a user can obtain a quick perceptual overview of the spread of simulations. Alternatively, one could assign other measures to the area, such as the vehicles' performances for that group.

TABLE I
TYPICAL EXPLORATOOL TASKS.

Task #	Example
Task 1	*Selection of a Group* BrandF, Seg. C, ModelY
Task 2	*Selection of a Case* BrandB, Seg. C, ModelB, Fuel Petrol, 1.8TBI 16V, Rev. 5
Task 3	*Comparison Case vs. Group* *Case*: BrandD, Seg. B, ModelU, Diesel, 1.3 Multijet 16V, Rev. 2 *Group*: BrandD, Seg. B
Task 4	*Comparison Case vs. Case* *Case 1*: BrandB, Seg. C, ModelB, Petrol, 1.8 TBI 16V, Rev. 3 *Case 2*: BrandB, Seg. C, ModelB, Petrol, 1.8 TBI 16V, Rev. 6 *Case 3*: BrandB, Seg. C, ModelB, Petrol, 1.8 TBI 16V, Rev. 8
Task 5	*Comparison Group vs. Group* *Group 1*: BrandD, Seg. B *Group 2*: BrandF, Seg. B

B. Data set Exploration

The users can interactively explore the data set through an *in-depth navigation* [24] performing drill-down and roll-up operations. **Drill-down** occurs when a user has identified a potentially interesting group of simulations and he/she wishes to further explore the group, so he/she can click on the ellipse to obtain more details. ExploraTool shows multiple nested ellipses, so the user can drill-down one level at time or multiple-levels in one step by clicking on the internal nested ellipses (Fig. 2). Every time the user drills down in the hierarchy, he/she is effectively performing a refinement of the query, filtering all of the simulations in the repository. For instance, when the user selects in sequence the ellipses $BrandF$ -> $Seg.\ C$ -> $ModelY$ (Task 1 in Table I), he/she is performing a query to retrieve from the repository all the items with exactly these values (ellipse labels). For each click, ExploraTool smoothly enlarges the selected group, rendering a fast transition to the new view. **Roll-up** operation is performed when the user wants to have a global data set view, he/she traverses up the hierarchy by clicking on the container ellipse.

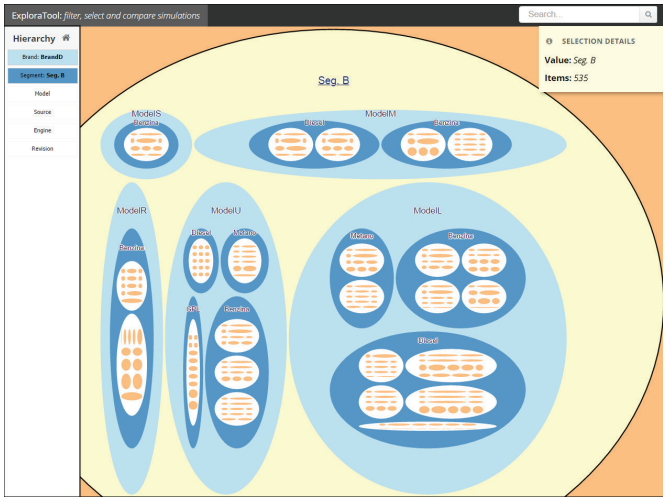


Fig. 2. This effect of the drill-down operation by directly clicking on the ellipse with the label "Seg. B" from the main view of Fig. 1 is shown. The ExploraTool smoothly enlarges the selected group, rendering a fast transition between views. If the user desires to go back and see less details, he/she can click directly on the external space to perform a roll-up operation returning back by one level at a time. In order to return to the initial overview, as in Fig. 1, the user can click on the "Home" icon shown on the top-left.

As shown in Fig. 1, the tool shows a vertical navigation hierarchy bar on the left, which: (1) gives an overview of the hierarchy [14]; (2) shows the current depth during the simulation repository navigation, supporting user orientation; (3) permits the hierarchy re-arrangement by interactively swapping the facet's levels, showing visual cues to indicate for instance what happens if the mouse is released [15]. In addition, during interactive data set exploration, ExploraTool shows the navigation path in the *navigation bar*, indicating the total number of filtered items, and updates the hierarchy bar (displaying only the remainder of the hierarchy from the current position). Fig. 1 shows the whole universe and the total number of simulations (2248) within the repository.

The operations provided by the ExploraTool rely on the direct manipulation [15] principle, which concerns the direct interaction and manipulation of the rendered objects (e.g., directly clicking on the target object). The use of ellipses as basic shapes guarantees there will be space between sibling ellipses at same level and amongst nested ellipses. This extra space improves the hierarchy discernment [19] and every operation involves exactly the target shape. For instance, in order to drill down in the hierarchy, the user points and clicks

on exactly the nested ellipse required. In order to roll-up the user points and clicks on exactly the parent shape utilising the space between the parent and child ellipses, which is always present. In other 2D space-filling techniques, such as the Treemap visualisation technique, both nested rectangles and adjacent rectangles have no space between them, so *the position of the mouse pointer designates a branch of the tree* [24] because *each point belongs to a single leaf node but also to all its ancestors* [24].

C. Vehicles' Performance Comparison

When the user finds an interesting data set he/she can add it to basket of simulations to compare by clicking on the "+ Add" button (Fig. 1). The user explores the repository and adds groups of items to the comparison bar (bottom of Fig. 1). Every time a selection is added to the basket, ExploraTool updates the chart on the right, showing the mean, maximum and minimum value for each selection, facilitating the comparison. Hovering the mouse cursor on the chart displays the exact values from the chart.

IV. Algorithm to process Dataset

Interactive exploration through ExploraTool is performed by clicking on ellipses (drill-down and roll-up operations). Nested ellipses form a hierarchy (tree data structure). Every time the user drills down, he/she clicks on ellipse that identifies a branch in the hierarchy. This Section introduces an algorithm called `BuildHierarchy` to process any tabular dataset to generate one of the possible tree data structures to be visualised with any kind of 2D space-filling visualisation techniques (e.g., Treemap, ellimap, etc.). The algorithm has been used within ExploraTool to process data from vehicle simulation repositories and organise the identified facet categories (i.e., *brand, project model, power source, engine type*) in an initial hierarchical ordering (decided in advance by the analysts) but can be altered any time through a re-arrangement operation that triggers the hierarchy recomputing. The algorithm can process any other dataset to be explored through ExploraTool, such as a catalogue of parts. In order to process other datasets, the requirement is to identify the facets (attributes) of the items and their values. The `BuildHierarchy` algorithm is implemented in the back-end of ExploraTool, which makes use of the generated tree data structure to build the ellipse-based visualisation. It runs in three cases: (1) when the user opens ExploraTool for the first time, the algorithm builds the initial default partial tree made of the first r levels; (2) every time the user performs a drill-down operation, he/she is navigating to a specific branch of the entire tree and so the algorithm builds a new subtree with r levels rooted at the selected node (the server provides chunks of subtrees made by r levels); (3) every time the user performs the re-arrangement operation, by sorting the facets in a different order.

A. BuildHierarchy Algorithm Description

Fig. 3 shows an abstract example of the `BuildHierarchy` algorithm (Alg. 1). It takes a dataset as input, and to be independent of the specific technology to store simulations, it has been transformed as a tabular dataset where each row records the relevant metadata for an individual simulation and the columns are their attributes. So, the **input** is a collection of n items $R = \{s_1, ..., s_n\}$. The dataset R (left side of Fig. 3) has a row for each individual item $s \in R$ (i.e., a simulation) that is described through attributes attached to it. The set $A = \{a_1, a_2, ..., a_m\}$ contains the labels/names of the attributes used to describe the items. For example, in the simulations use case, the labels for the facets [16] are $A = \{brand, projectmodel, powersource, enginetype\}$. The dataset has these facets (attributes) on the columns.

The **algorithm output** is a tree data structure $T = (V, E)$ (right side of Fig. 3), where V is the set of nodes and E are the edges. For each attribute $a_i \in A$, there will be a level in the tree T (the height of tree is exactly the number of facets). Each attribute a_i has a set of valid values called domain $D_i = dom(a_i)$. For instance, in the simulation context, the attribute *power source* has the domain $D_{powersource} = dom(powersource) = \{Bifuel, Petrol, Diesel\}$. In the example, the attribute *brand* has the following valid values $D_{brand} = dom(brand) = \{BrandA, BrandB, BrandC, BrandD, BrandF\}$. At the level of the tree corresponding to attribute a_i, there are the nodes with the values in D_i. Some nodes corresponds to zero items in the original data set, so they are not present in the hierarchical view. For instance, the path $(root, 1, B, b)$ that would be present in the full tree is not present in our constructed tree because there are no items in the original data set with the values $s[1] = 1 \wedge s[2] = B \wedge s[3] = b$.

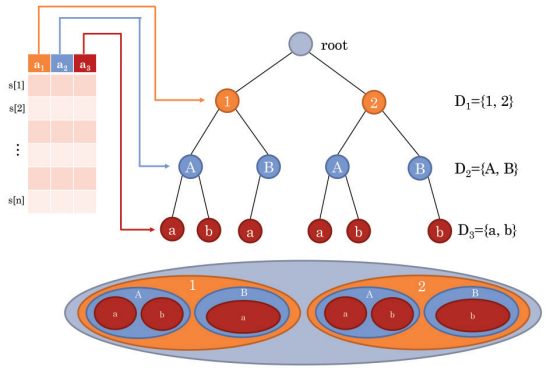


Fig. 3. Example of hierarchy extraction from a table. The table on the left has tree attributes $A = \{a_1, a_2, a_3\}$ and n simulations. The tree on the right has a level for each domain D_i and at each level i there are the nodes with the labels in D_i. ExploraTool builds the visualisation shown on the bottom of the figure, starting from the tree data structure.

The tree T will be displayed as nested ellipses starting from the root (Fig. 3 bottom side). For each node, the algorithm stores the label to be displayed on the ellipses, and calculates the number of simulations ($Count[u] \in \mathbb{N}, \forall u \in V$) that will determine the area of the relevant ellipse.

Algorithm 1 shows the algorithm pseudo-code to process the dataset as input and generate the tree data structure. Initially the tree T has the $root$ node and no edges ($E = \emptyset$). The algorithm scans all the simulations in the repository $s \in R$ exactly one time (Alg. 1 line 5). For each simulation s,

the algorithm scans the simulations' attributes in the order specified by the function f_S (Alg. 1 line 6). So, the algorithm scans the dataset row by row, and for each row the columns.

Algorithm 1 takes each simulation and traverses down the tree from the root to a leaf, one level at time, through the nodes with simulation attribute values. On line 7, the algorithm tries to find the node at level i with value $attrvalue$. If the node does not exist, the algorithm creates it at lines 8-11. At line 12, the algorithm counts the number of simulations; this value will be used to calculate the ellipse's areas.

Algorithm 1: BuildHierarchy(R, r, f_S)

Input: Set of simulations R, number of levels r to load and an attributes ordering function f_S
Output: $T = (V, E)$

```
1  root ← create the hierarchy root;
2  V ← {root}, E ← ∅;
3  prevnode ← root;
4  curnode ← null;
5  foreach simulation s in the repository R (s ∈ R) do
6      foreach attribute value "attrval" in s,
7      ordered by f_S and limited to the first r values do
8          curnode ← Find the child with value attrval;
9          if curnode is null then
10             curnode ← create a new node;
11             Label[curnode] ← attrval;
12             Count[curnode] ← 0;
13             V = V ∪ curnode;
14             E = E ∪ {(prevnode, curnode)};
15         Count[curnode] + +;
16         prevnode ← curnode;
17     prevnode ← root;
18 return (V, E);
```

The f_S function in input specifies a sorting of the facets (sorting of domains $f_S : \{1, 2, \cdots, m\} \rightarrow \{1, 2, \cdots, m\}$), which has a direct impact upon the resulting hierarchy. The algorithm creates an initial hierarchy based on an initial domain sorting function f_S specified at configuration time. Then, the tree is rendered using a 2D space-filling visualisation approach. The user can swap the facets through a rearrangement operation by dragging levels in the hierarchy, choosing a different permutation of the facets that changes the input function f_S to obtain a different hierarchical views.

B. BuildHierarchy Algorithm Running time discussion

The algorithm generates a tree data structure with height r, where each level is a facet (attribute). At each level the tree has a node for each attribute value. For instance, in the hierarchy at the level of the segment attribute, there is a node for each segment value (e.g., Seg. A, Seg. B, Seg. C, etc.). In the worst case all attributes have exactly p values (p nodes at each tree level). Therefore, in the worst case the total number of nodes in the tree is $O(p^r)$ with $p > 1$ and $1 \leq r \leq m$. Introducing the parameter r, two interesting features can be provided: (1) details on-demand when the user drills-down in the hierarchy, it is asking for the next r levels; (2) from a computational point of view, the algorithm computes only r levels at time. In order to keep the hierarchy clear during the visualisation, ExploraTool loads five facets at time; it does not show more than five levels at each time ($r \leq 5$), so the number of the nodes in the tree is in the worst case $O(p^5)$.

In order to build the tree data structure, the algorithm scans all the dataset items ($\Theta(n)$ where $|R| = n$) and exactly r columns ($O(r)$, the chosen facets to visualise). Columns are scanned in the order defined by the sorting function f_S, which is chosen by the user via the GUI. Given the value in the intersection of the selected row (simulation) and column (attribute), the algorithm checks whether a node with that value is in the tree at level r. Thus, the algorithm `BuildHierarchy` **running time is**: $\Theta(n) \cdot O(r) \cdot O(p) \leq O(n \cdot r \cdot p)$.

Furthermore, the algorithm to process any tabular dataset and build a hierarchy can be executed by using the map/reduce paradigm on large data sets in a distributed environment using for instance Apache Spark[1]. The idea is to slice the tabular dataset in q groups of rows. Each group of row will be processed by a computing node, executing the algorithm and generating a tree data structure. The trees generated by all computing nodes can be merged together to obtain the final hierarchy displayed by ExploraTool.

V. User Evaluation Study

ExploraTool adopts a novel visual and interactive user interface for the engineering field to explore multiple repositories of simulations. This section reports on the fundamental activity to evaluate the tool's effectiveness and usability in a real setting involving industrial experts from a large automotive manufacturer. The question to answer is how users perceive the system, evaluating its effectiveness, in terms of tasks completion as well as the usability of the interface and the overall satisfaction, acquiring user feedback.

A. Methodology

The study lasted thirty minutes for each participant and consisted of four phases as described in the following.

1) Preliminary Survey: The participant fills out a questionnaire[2] to collect demographic information, particularly as related to their experience in the simulation field and their existing procedures to compare vehicle performances using simulation repositories. ExploraTool uses colours in its user interface, so it is fundamental to examine its usability by people with colour deficiencies, especially the effectiveness of the chosen colour blindness colour palette. Therefore, a quick Colour Blindness test has been performed (Ishihara test).

2) Training Phase: It demonstrates the functionality to the participant, using *training material* with standard basic tasks to ensure consistency of the explanation among the participants.

3) Testing Phase: Users execute five tasks (Table I) using ExploraTool in which he/she should find and select single simulations and/or groups of simulations to compare their performances. At the end of each task, the user answers questions to evaluate whether it was successfully completed,

[1]Apache Spark web-site http://spark.apache.org/

[2]ExploraTool Usability Study Questionnaires are available on-line at http://floasysorg.github.io/Floasys/usabilitystudy/exptoolv1.html

rate how ease and quick it was to perform the task (standard questions from the After Scenario Questionnaire[3]).

4) Summary Survey: The questionnaire concludes with the assessment of the overall ExploraTool perceived usefulness and user satisfaction. This part of the questionnaire is based on a standard TAM model, which is extensively used in usability studies to explain and/or predict users' behavioural intentions when accessing a new technology or system as well as to test the user acceptance.

B. Evaluation Results Analysis and Discussion

1) Participants' Demographics: Recruits comprised 24 engineers of a large automotive manufacturer. The sample was mostly male (87.5%) with mean age of 34 (std. dev. 7.5). The test has been performed in two locations in Italy: one day in Pomigliano D'Arco (Naples) involving 8 participants and almost two days in Orbassano (Torino) involving 16 participants. The conditions were kept the same (same hardware and software) in both locations, using an isolated room containing only one participant at time, where he/she could concentrate on the test without distractions.

ExploraTool has been designed primarily for simulation Analysts but different company roles performed the usability test, all of whom involved in the CFD simulation field: 16 CFD Analysts, 3 Performance Engineers (PEs) and 5 Technical Managers (TMs) with, on average, 4, 8, and 12 years of experience in the field, respectively. Technical Managers and Performance Engineers usually perform few simulations per year (mean 85, std. dev. 69) as compared to the analysts (mean 151, std. dev. 123), because TMs are responsible for the internal team organisation, resource monitoring and their allocations, whilst PEs work on big picture projects and are responsible for design choices. There was only one colour blind participant; whilst he was not able to distinguish colours, he successfully completed all the tasks, identifying and discerning correctly the ellipses. ExploraTool uses a space filling visualisation approach using ellipses to depict groups of simulations. As such, it is important to understand which users had previous experience with such visualisations. Participants had high experience with standard charts (bar, pie, chart and surface charts) used for the everyday work, but much less had experience with Treemap (4%).

For all participants it would be useful in their role to be able to automatically extract simulation data from the repositories and to obtain comparisons of related statistics among different releases of the same project or different projects releases (agreement of 100%). Notwithstanding, all participants declared that they do not have an automatic tool to perform these tasks. Instead the common procedure is to export data from the simulator software in comma separated value format and to analyse them via Microsoft Excel.

2) Tasks Execution Results: The users executed tasks corresponding to those in Table I. In order to assess the effectiveness of the tool the error rate has been measured for each task. As a result, all participants completed tasks 1, 2, 4, and 5 without errors, whilst task 3 had an error rate of 2.8%. A simulation analyst wrongly selected an alternative group of simulations instead of the group $BrandD$ -> $Seg.B$ requested in Task 3 (Table I). The responses to the ASQ questionnaire indicated that participants were highly satisfied with the ease of the tasks and the amount of time required to complete them; across all of the tasks there was a mean of 6.9 (std. dev. 0.1) for the easiness and 6.7 (std. dev. 0.1) for the time, on a 7-point scale. Furthermore, grouping the participants by their main role (CFD Analyst, Technical Manager, Performance Engineer) in the company yields no statistical difference with regard to the above metrics (Kruskal-Wallis test).

3) Perceived Usefulness and overall Acceptance Results: At the end of the Testing Phase participants responded to the TAM questionnaire, whose Cronbach's Alpha value was 0.92.

TABLE II
SPEARMAN'S CORRELATION COEFFICIENTS BETWEEN SUBSCALES: PU, PERCEIVED USEFULNESS; EOU, PERCEIVED EASE OF USE; ATT, ATTITUDE TOWARD USE; BI, BEHAVIOURAL INTENTION TO USE.

Subscale	PU	EOU	ATT	BI
PU	1.0			
EOU	.194	1.0		
ATT	$.604^{**}$	$.457^{*}$	1.0	
BI	$.530^{**}$	.384	$.557^{**}$	1.0

$^{**}p < .01$, $^{*}p < .05$

Table II reports the Spearman's correlation coefficients among the subscales with the corresponding significance levels (indicated by the $*$ and the p value). In the table the highest correlation is between ATT and PU (.604, with p value $< .01$). BI is positively correlated with PU and ATT with high significant level ($p < .01$ for both metrics). Furthermore, analysing the TAM answers' rates (7-point Likert scale) on the PU, EOU, ATT, BI subscales, results were highly positive for all these metrics. The highest rate was for the question "*Using the system would enable me to accomplish tasks more quickly*" in the Perceived Usefulness questionnaire section (mean 6.8, std. dev. 0.4). In addition, a regression analysis was carried out in order to identify which variables (PU, EOU, and ATT) influenced the use of ExploraTool (BI). The model yielded an adjusted R^2 value of .606. Based on the analysis of the attitude results, participants think that ExploraTool's idea is wise, smart and interesting. As result, ATT is a significant variable in increasing the software's acceptance. When asked to express the positive tool aspects, the participants indicated: easiness (71%), quickness (58%), intuitiveness (25%), usefulness (17%) and effectiveness (13%). The negative aspects concerned mainly the visualisation (29%), in particular the partial overlapping of some labels and some thin ellipses, pointing out the need for improvements of the visualisation overall aesthetic. In addition users reported future tool improvements, like additional aggregated statistical data on mouse hovering (13%) and search by keywords (4%). In summary, despite the participants' lack of knowledge of space filling visualisations like Treemap, they were able to complete the tasks, and expressed high satisfaction in terms of its usefulness, usability, and simplicity. Furthermore, users would use the tool on regular basis and recommend other to use it (questions D18 and D19 of the questionnaire).

[3]ASQ Questionnaire http://garyperlman.com/quest/quest.cgi?form=ASQ

VI. Conclusions and Future Work

This paper presented ExploraTool to visually and interactively explore multiple repositories of simulations through a novel user interface for the engineering field. As practical industrial application, the tool has been used to select individual and/or groups of simulations to compare their performances (i.e., comparison of aerodynamic simulations by the drag-coefficient, called Cx). The utility of ExploraTool has been evaluated in a real setting with expert industrial users. The users found the tool useful, usable and simple to use, and users were interested in the novel ExploraTool interactive user interface. The ExploraTool concept generalises beyond the simulation context to any other context in which the goal is to select items by their properties and perform comparisons. Filling out the questionnaires, analysts indicated the need to select and compare not only simulations, but also wind tunnel experiments from a cleaned repository and competitors' products performance data, extending the applicability of the tool. As possible extension to other industries, the tool can provide a visual overview of catalogues of parts, supporting the interactive finding of parts by their properties.

Reflections on the process adopted to design ExploraTool may serve to provide good practice suggestions for the design of novel interactive user interfaces for the engineering field. In order to identify appropriate tasks and type of interactions, an essential element was the close interaction with the end-users, adopting an agile development methodology consisting of two week periods for the plan, design, develop, and internal testing phases. For ExploraTool, the crucial stimulation of discussions and user engagement was a small proof of concept tool permitting the basic interactions. In this case the users were willing to pro-actively participate in discussions and support its development.

As future work, users already provided interesting requests for new features during the evaluation test: the facility to bookmark the exploration of results; to introduce a specific preference section; to filter the repository items by typing a keyword within a search bar that updates the visualisation with the filtered items; the ability to export the comparisons in Excel and PowerPoint in a manner which is compliant to the internal industrial templates. In addition, improvements to the layout algorithm are needed to avoid thin ellipses, thereby improving the overall visualisation aesthetic, whilst also improving the label positions.

Acknowledgment

The authors gratefully thank all participants who took part to the ExploraTool Usability Study.

References

[1] D. A. Keim, "Visual exploration of large data sets," *Communications of the ACM*, vol. 44, no. 8, pp. 38–44, 2001.

[2] G. Marchionini, "Exploratory search: from finding to understanding," *Communications of the ACM*, vol. 49, no. 4, pp. 41–46, 2006.

[3] E. F. Duarte, E. Oliveira Jr, F. R. Côgo, and R. Pereira, "Dico: A conceptual model to support the design and evaluation of advanced search features for exploratory search," in *Human-Computer Interaction–INTERACT 2015*. Springer, 2015, pp. 87–104.

[4] K. Klouche, T. Ruotsalo, D. Cabral, S. Andolina, A. Bellucci, and G. Jacucci, "Designing for exploratory search on touch devices," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, 2015, pp. 4189–4198.

[5] M. Burt and C. Li Liew, "Searching with clustering: An investigation into the effects on users' search experience and satisfaction," *Online Information Review*, vol. 36, no. 2, pp. 278–298, 2012.

[6] M. Hearst, "Clustering versus faceted categories for information exploration," *Communications of the ACM*, vol. 49, no. 4, pp. 59–61, 2006.

[7] A. L. Kaczmarek, "Interactive query expansion with the use of clustering-by-directions algorithm," *IEEE Transactions on Industrial Electronics*, vol. 58, no. 8, pp. 3168–3173, 2011.

[8] R. W. White and R. A. Roth, "Exploratory search: Beyond the query-response paradigm," *Synthesis Lectures on Information Concepts, Retrieval, and Services*, vol. 1, no. 1, pp. 1–98, 2009.

[9] R. G. Capra and G. Marchionini, "The relation browser tool for faceted exploratory search," in *Proceedings of the 8th ACM/IEEE-CS joint conference on Digital libraries*. ACM, 2008, pp. 420–420.

[10] M. Wilson, A. Russell, D. A. Smith *et al.*, "mspace: improving information access to multimedia domains with multimodal exploratory search," *Communications of the ACM*, vol. 49, no. 4, pp. 47–49, 2006.

[11] C. Carpineto, S. Osiński, G. Romano, and D. Weiss, "A survey of web clustering engines," *ACM Computing Surveys (CSUR)*, vol. 41, no. 3, p. 17, 2009.

[12] P. Ferragina and A. Gulli, "A personalized search engine based on web-snippet hierarchical clustering," *Software: Practice and Experience*, vol. 38, no. 2, pp. 189–225, 2008.

[13] J. Kaizer and A. Hodge, "Aquabrowser library: search, discover, refine," *Library Hi Tech News*, vol. 22, no. 10, pp. 9–12, 2005.

[14] B. Shneiderman, "The eyes have it: A task by data type taxonomy for information visualizations," in *IEEE Symposium on Visual Languages,*, 1996, pp. 336–343.

[15] R. Lutz, D. Rausch, F. Beck, and S. Diehl, "Get your directories right: From hierarchy visualization to hierarchy manipulation," ser. VL/HCC, Melbourne, VIC, 2014, pp. 25–32.

[16] G. Smith, M. Czerwinski, B. R. Meyers, G. Robertson, and D. Tan, "FacetMap: A scalable search and browse visualization," *IEEE Transactions on Visualization and Computer Graphics,*, vol. 12, no. 5, pp. 797–804, 2006.

[17] P. Demian and R. Fruchter, "Finding and understanding reusable designs from large hierarchical repositories," *Information Visualization*, vol. 5, no. 1, pp. 28–46, 2006.

[18] J. Bauder and E. Lange, "Exploratory subject searching in library catalogs: Reclaiming the vision," *Information Technology and Libraries*, vol. 34, no. 2, pp. 92–102, 2015.

[19] B. Otjacques, M. Cornil, and F. Feltz, "Visualizing cooperative activities with ellimaps: the case of Wikipedia," in *Cooperative Design, Visualization, and Engineering*. Springer, 2009, pp. 44–51.

[20] C. Gargiulo, D. Pirozzi, and V. Scarano, "An architecture for CFD workflow management," in *Proceedings of the 11th IEEE International Conference on Industrial Informatics (INDIN), Bochum, Germany, July 29-31*, 2013, pp. 352–357.

[21] C. Gargiulo, D. Pirozzi, V. Scarano, and G. Valentino, "A platform to collaborate around CFD simulations," in *Proceedings of the 23rd IEEE International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, 2014, pp. 205–210.

[22] C. Gargiulo, D. Malandrino, D. Pirozzi, and V. Scarano, "Simulation data sharing to foster teamwork collaboration," *Scalable Computing: Practice and Experience*, vol. 15, no. 4, pp. 309–329, 2014.

[23] A. Fish, C. Gargiulo, D. Pirozzi, and V. Scarano, "Simulation repository visualisation and exploration," in *13th IEEE International Conference on Industrial Informatics (INDIN)*, 2015, pp. 832–837.

[24] R. Blanch and E. Lecolinet, "Browsing zoomable treemaps: structure-aware multi-scale navigation techniques," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 6, pp. 1248–1253, 2007.