UNIVERSITÀ DEGLI STUDI DI SALERNO

Department of Computer Science

PhD in Computer Science

XIII Ciclo - Nuova Serie

PhD Thesis in Computer Science

Teamwork Collaboration around Simulation Data in an Industrial Context

Author

Donato Pirozzi

PhD Program Chair
Prof. Giuseppe Persiano

Supervisor
Prof. Vittorio Scarano

Academic Year 2013-2014

Supervisor
Prof. Vittorio Scarano, Dip. di Informatica,
Università degli Studi di Salerno, Italy

Graduate Group Chair
Prof. Giuseppe Persiano, Dip. di Informatica,
Università degli Studi di Salerno, Italy

Dean
Prof. Vincenzo Loia, Dip. di Informatica,
Università degli Studi di Salerno, Italy

Date of defense
April 22th, 2015

Donato Pirozzi
ISISLab, Dip. di Informatica
Università degli Studi di Salerno
Via Giovanni Paolo II, 84084 Fisciano (Salerno), Italy
E-mail: `dpirozzi@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 ten years since my first day in Lancusi on September 2004. In particular, I would like to thank professor Vittorio Scarano and ISISLab to be involved in an active and stimulating research group for my bachelor, master and this PhD thesis.

I have to thank Fiat Chrysler Automobiles (FCA) for this great opportunity to do a multidisciplinary PhD and be involved at same time in both academic and industrial contexts. In particular, I thank Claudio Gargiulo for the interesting and stimulating long discussions about very different technical and non-technical topics.

I'd like to thank University of Brighton to host me over three months providing all the facilities to do research in an international environment with a mix of different cultures. Especially, I want thank Andrew Fish for his continuous support.

A great good luck to everyone I met during my PhD.

And ... of course a special thanks to my parents and Teresa :)

All software licenses (e.g., CD-adapco™ STAR-CCM+®, BETACAE™ ANSA®, modeFrontier®), software documentation and training materials to make this thesis have been provided by Fiat Chrysler Automobiles (FCA) and Università degli Studi di Salerno (Unisa).

ISISLab FCA FIAT CHRYSLER AUTOMOBILES

Abstract

Nowadays even more small, medium and large enterprises are world-wide and compete on a global market. In order to face the new challenges, industries have multiple co-located and geographically dispersed teams that work across time, space, and organisational boundaries. A virtual team or a dispersed team is a group of geographically, organisationally and/or time dispersed knowledge workers who coordinate their work using electronic technologies to accomplish a common goal. The advent of Internet and Computer Supported Cooperative Work (CSCW) technologies can reduce the distances between these teams and are used to support the collaboration among them. The topic of this thesis concerns the engineering dispersed teams and their collaboration within enterprises. In this context, the contributions of this thesis are the following: I was able to (1) identify the key collaborative requirements analysing a real use case of two engineering dispersed teams within Fiat Chrysler Automobiles; (2) address each of them with an integrated, extensible and modular architecture; (3) implement a working industrial prototype called Floasys to collect, centralise, search, and share simulations as well as automate repetitive, error-prone and time-consuming tasks like the document generation; (4) design a tool called ExploraTool to visually explore a repository of simulations provided by Floasys, and (5) identify the possible extensions of this work to other contexts (like aeronautic, rail and naval sectors).

The first research aim of this work is the analysis of the key collaborative requirements within a real industrial use case of geographically dispersed teams. In order to gather these requirements, I worked closely with two geographically separated engineering teams in Fiat Chrysler Automobiles (FCA): one team located in Pomigliano D'Arco (Italy) and the other one in Torino (Italy). Both teams use computer numerical Computational Fluid Dynamic (CFD) simulations to design vehicle products simulating physical phenomenons, such as vehicle aerodynamic and its drag coefficient, or the internal flow for the passengers thermal comfort. The applied methodology to collect the collaborative and engineering requirements is based on an extensive literature review, on site directly observations, stakeholders' interviews and an user survey. The identified key collaborative requirements as actions to perform to improve the collaboration among dispersed teams are [1]: centralise simulation data, provide metadata over simulation data, provide search facility, simulation data versioning, and data sharing. Engineers, in the analysed field, use multiple simulator software, so in order to centralise simulation data, it is fundamental to collect data from multiple heterogeneous sources avoiding

the Vendor Lock-In Anti-Pattern [2]. In according to the gathered collaborative and engineering requirements, a working real prototype called Floasys has been developed. The Floasys target customers are industries who use CFD simulators to design their products. Floasys [1, 3] collects, centralises, and stores simulation data in open format (e.g., XML). Floasys provides additional services over collected data like a simulator independent SearchTool, that is very useful to get simulations performed by different engineers, and compare the found simulations performances over multiple design revisions. In addition, the system allows the data sharing through URLs exchange.

From architectural point of view, Floasys meets the extensibility and modularity Non-Functional requirements to be tailored to the customers needs, accommodate future requirements and be used in other departments. In order to meet the extensibility and modularity Non-Functional Requirements, the architecture is based on the concept of plug-in [3]. In this way, each module of the architecture is a plug-in that can be replaced with another equivalent implementation. The Floasys uses a three layers approach. It integrates on the bottom layer the data source wrappers (e.g., simulators software that generates simulation data, or test-beds that generates experimental data). The middle layer abstracts and manages the data source heterogeneities. On the top layer there are the tools that provide collaborative and engineering functionalities to users.

Although this research activity concerns an automotive use case, issues faced within this sector seem to be very common issues also in other sectors as highlighted in the existing literature [4,5], especially for the list of gathered requirements. Therefore, many of the considerations and design decisions described through this work could be used also for other type of simulations and experiments in other contexts (i.e., aeronautic, rail and naval sectors).

Collecting a huge amount of data from multiple sources and centralising them in open format, introduces the need to have an overview, and at same time, explore and query the dataset to get the desired data. Therefore, another contribution of this work is an interactive tool to visualise, explore and query simulations repositories called briefly EsploraTool that is a Floasys tool. Although the tool idea was born in the context of simulation and experimental data, it is enough general to be used with any dataset so that it has been tested with the amazon.co.uk clothing catalogue to demonstrate its generality. The tool is based on the Euler Venn diagrams. The tool represents the data items in a hierarchical way. The user can explore the dataset through drill-down and roll-up operations to get more or less dataset details. Going down through the hierarchy the user is filtering items within the dataset and making a graphic query. Using the experimental data, the idea is that engineers explore the available experiments and get two or more experiments to compare together. After the tool design and implementation stages, now the tool is under testing with real users [6] to perform a User Experience and Performance test as well as Usability test. In addition, the tool its self can be the test-bed for other features, such as, the opportunity to merge two ellipses performing a logic AND operation between two properties of the dataset.

Abstract (Italiano)

Oggigiorno le piccole, medie e grandi imprese operano e competono su un mercato globale e mondiale. Per affrontare le nuove sfide ed essere competitive sul mercato, le industrie hanno più sedi e team geograficamente lontani. In questo contesto, membri dello stesso team o afferenti a team differenti si ritrovano a lavorare assieme indipendentemente dal fuso orario, dal posto in cui si trovano e delle strutture aziendali. Un team virtuale è costituito da gruppi di persone geograficamente lontane che coordinano il loro lavoro usando le nuove technolgie per raggiungere un obiettivo comune. L'avvento di Internet e le nuove tecnologie a supporto del lavoro cooperativo (CSCW) possono ridurre le distanze tra i team geograficamente lontani e possono supportare la collaborazione tra essi. Il tema principale di questa tesi è relativo ai team virtuali di ingegneri e al modo in cui collaborano all'interno dell'industria automotive. In tale contesto, i contributi di questo lavoro di tesi possono essere sintetizzati nel seguente modo: (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 simulationi all'interno di Floasys (5) sono state identificate le possibili estensioni della piattaforma in altri contesti quali l'aeronautico, il ferroviario ed il navale.

L'obiettivo di questo lavoro è la raccolta dei requisiti collaborativi e delle relative necessità che sopravvengono nel momento in cui differenti team geograficamente lontani (virtual teams) si ritrovano a collaborare per perseguire un risultato comune. Per cui si è considerato un caso d'uso industriale e reale di team geograficamente lontani, lavorando a stretto contatto con due team di ingegneri di Fiat Chrysler Automobiles (FCA): un team presso la sede di Pomigliano D'Arco (Italia) e l'altro team presso la sede di Torino. Entrambi i team di analisti utilizzano le simulazioni al calcolatore (Computational Fluid Dynamic simulations) per progettare automobili simulando fenomeni fisici, quali ad esempio l'aerodinamica esterna dell'autoveicolo. La metodologia applicata per la raccolta dei requisiti si basa su osservazioni dirette sul campo, interviste agli utenti, la somministrazione di un questionario on-line ed il confronto con la letteratura esistente. I requisiti collaborativi identificati come azioni da compiere per supportare la collaborazione tra team geograficamente lontani sono: centralizzare i dati delle simulazioni, fornire la possi-

bilità 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 accordo ai requisiti individuati è stato sviluppato un prototipo chiamato Floasys che è il secondo contributo di questo lavoro. I clienti finali di Floasys sono tutte le industrie che utilizzano le simulazioni di CFD per progettare i loro prodotti, quindi, le industrie automotive, aeronautiche e navali. Floasys colleziona i dati delle simulazioni, li memorizza in formato aperto XML e li centralizza in un repository condiviso. Floasys fornisce servizi aggiuntivi sui dati raccolti e memorizzati in formato aperto, ad esempio la possibiltà di annotare i file oppure di cercare all'interno del repository delle simulationi indipendentemente dal simulatore con cui sono stati generati i file. È molto utile ottenere le simulazioni effettuate da altri membri dello stesso team o di team diversi, questo è particolarmente utile quando si vogliono confrontare le prestazioni di più prodotti relativamente a più revisioni di progetto. Infine, Floasys offre la possibilità di condividere le simulazioni tramite lo scambio di URL univoche.

Nel cercare di fornire concretamente questi servizi differenti sfide vanno considerate: 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 concreato 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 stessa dai simulatori stessi.

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 [4]. 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 abbastanza generico per essere utilizzato con qualsiasi dataset. Il tool è basato 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, ora il tool è in fase di testing con utenti reali allo scopo di ottenere dati sulla sua usabilità [6].

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.

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

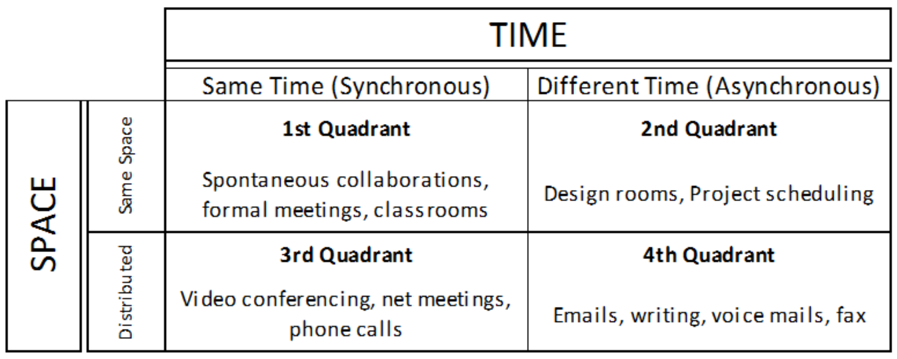
Nowadays small and medium enterprises (SMEs) as well as large industries are worldwide and work in a global market. Their organisation is often spread over multiple nations and companies face time, space and cultural barriers. In a high competitive world market and economy, companies face the need for a fast time-to-market, low cost and rapid product development [7]. The introduction of Internet, new communication technologies and the variety of computer-mediated communication systems 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 [8]. 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* [8]. Computer

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 [8]).

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 according to Bruegge at al [9], through this thesis the participants are all the persons involved in a project. In team-based organisations, participants are grouped into teams (Fig. 1.2). 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 [9, 10]. 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[1] means that people will try to cooperate, using their individual skills providing constructive feedback, despite any personal conflict between individuals.

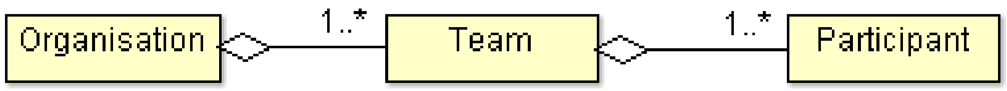


Figure 1.2: A team-based organisation consists of teams, which consist of participants (*adapted from Bruegge at al. [9]*).

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 or dispersed teams. In literature, different definitions of virtual teams exist. For instance, the authors of the paper "*Virtual teams: a literature review*" [7] identified a list of at least five existing definitions. Based on the existing definitions, 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 [11], such as e-mail, video-conferencing, telephone, etc.

Of course, today complex products like automotive products are designed in collaboration with the suppliers[2] directly involved in the design process. Hence, also teams belong 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, and
- the team members are knowledge workers.

The knowledge workers are the employees who have their main capital in the knowledge whose job is to "*think for a living*" [12]. Examples of knowledge workers are: engineers, doctors, architects, scientists, academics, etc.
Summarising these characteristics, the definition of virtual team is:

[1]Teamwork source definition: `http://www.businessdictionary.com`
[2]Supplier: a party that supplies goods or services to the company. For instance, a supplier that provides components to an automotive industry. [Source: `http://www.businessdictionary.com`]

Virtual Team definition *[7]: (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.*

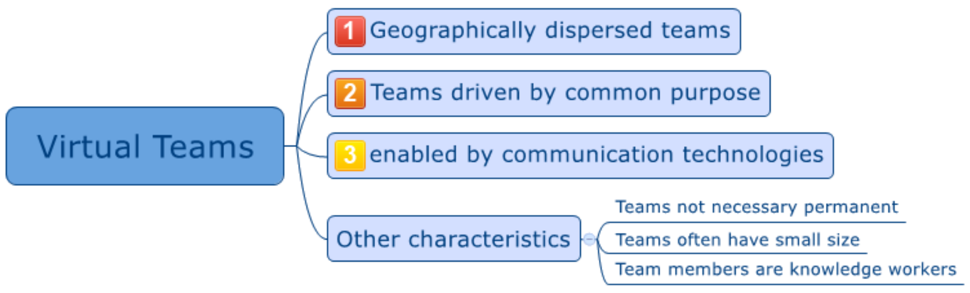

Virtual Teams
1 Geographically dispersed teams
2 Teams driven by common purpose
3 enabled by communication technologies
Other characteristics
Teams not necessary permanent
Teams often have small size
Team members are knowledge workers

Figure 1.3: Virtual teams characteristics.

An important keyword of the Virtual Team Definition 1.3 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 [7]. 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 [13], 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. One difficult is the initial social relationships development among team members, and it is possible to overcome it organising face-to-face meetings. Another difficult is the need to reinforce project objectives because the geographically separated members can inadvertently change or misunderstand the project objectives. When team members are in the same office, it is very easy to ask directly to a colleague and have a fast answer. Finally, it is evident from literature that another drawback of Virtual Teams is the increase of workload in communications because the team members tend to exchange formal e-mails for everything.

1.1.2 Enterprise 2.0

The term Web 2.0 was coined in 2004 to describe the internet's capability to allow people to connect together and contribute with content. Examples of emergent social software platform are Facebook, Twitter, YouTube, and Wikipedia. Enterprise 2.0, coined by Professor McAfee, is the use of Web 2.0 technologies within enterprises.

The **Enterprise 2.0** known also as E2.0 is the use of *emergent social software platforms* within an organisation to pursue its goals [14]. 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 predefine 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 and when does everyone. On the opposite side E2.0 are completely free without predefine 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 installed it.

1.1.3 Trialogic Learning

In the **Trialogic Learning** [15] 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:

1. the subjects or learners,
2. the collaboratively development, transformation or object creation
3. for later use or other users.

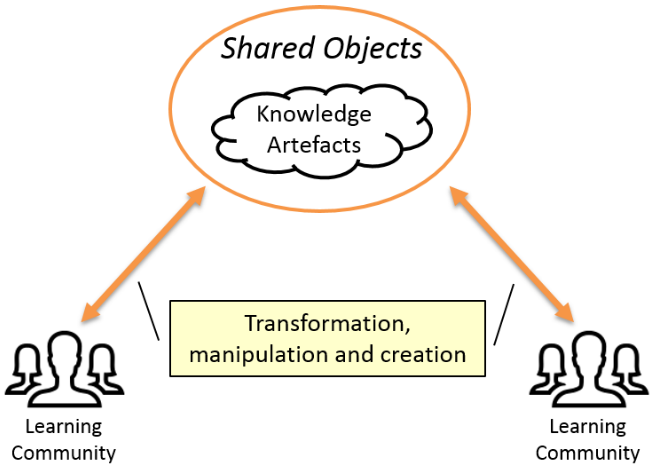


Figure 1.4: Trialogic Learning elements.

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. The manipulation of knowledge objects differentiates the Trialogic Learning from the Monologic and Dialogic learning as shown in Figure 1.5.

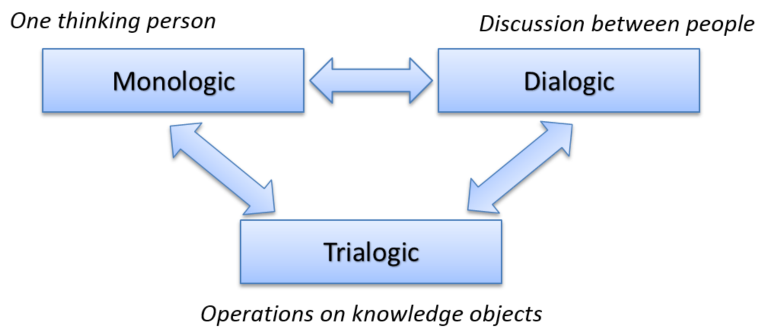


Figure 1.5: Monologic, Dialogic and Trialogic Learning.

Specifically, **shared objects** [15] 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 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, but at same time introduces different

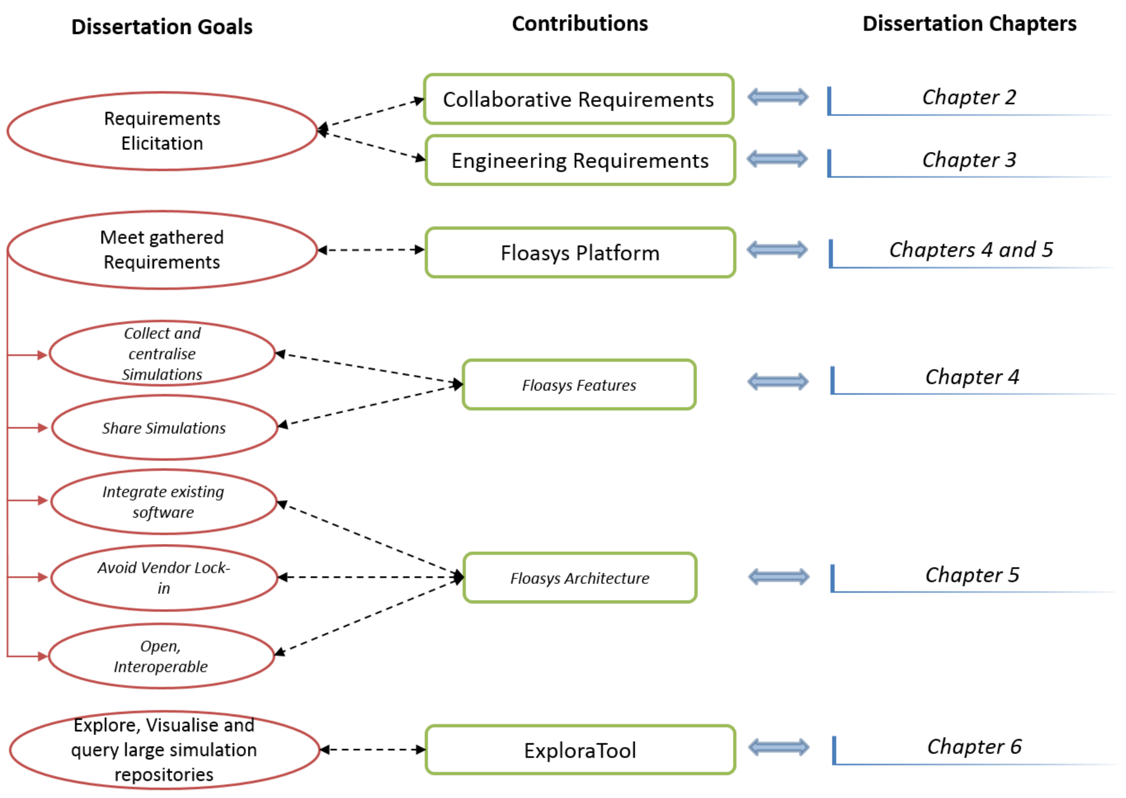


Figure 1.6: Dissertation Goals, Contributions and Chapters.

issues to face, such as the multidisciplinary and the need to get insight into an engineering field that initially seems to be far from the computer science.

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:

1. Collection, identification and analysis of the key collaborative requirements of dispersed teams within a real industrial use case (FCA use case);
2. Design of an integrated, extensible and modular software architecture;
3. Design, implementation and testing of a real working prototype called Floasys with its collaborative and engineering functionalities;
4. Design, implement and test a generic tool called ExploraTool to visually explore large datasets in hierarchical way;
5. Identify the possible extensions to different contexts (like aeronautic, rail and naval sectors).

Figure 1.6 shows the mapping among the dissertation goals, contributions and chapters. The picture reports on the left the achieved goals. From each goal, the picture tracks the relative contribution and the dissertation chapter in which the topic has been described.

1.3 Dissertation Roadmap

This thesis has been organised logically starting from the requirements elicitation activity and the gathered collaborative and engineerig 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. The last chapter discusses an interactive tool to visualise, explore and query repository of simulations and experiments.

The overall dissertation has been organised as follows:

- Chapters 2 and 3 introduce, describe and analyse respectively the collaborative and engineering requirements gathered through on site observations, stakeholders interviews and an on-line survey.
- Chapter 4 introduce Floasys, its features and its graphical user interface (GUI). Of course, the functionalities have been divided in two parts: collaborative and engineering features.
- Chapter 5 describes the Floasys integrated, extensible and modular architecture and maps the architecture features with the provided functionalities and the gathered requirements.

- Chapter 6 introduces the problem of large dataset navigation and exploration like simulation and experimental repositories and describes the general idea behind the ExploraTool.
- The last chapter 7 concludes the thesis 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 (see Appendix A).

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.

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).

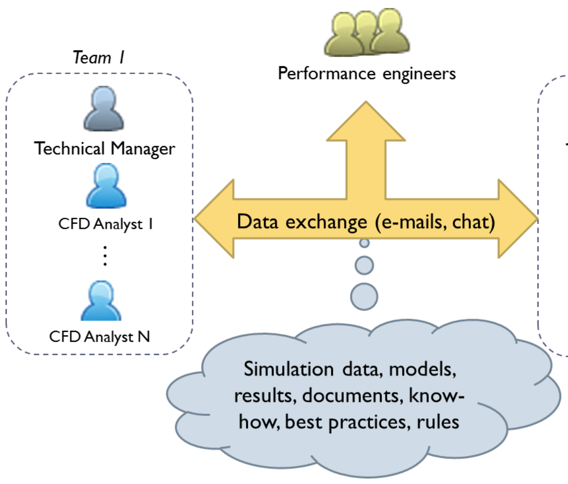


Figure 2.1: Collaboration among geographically distributed teams.

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 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 [16] 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 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).

	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[1]. 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*" [4] 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. 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 [5], 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

As already known in literature, the requirements elicitation is not an easy activity. 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:

1. stakeholders identification;
2. domain understanding;

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

3. 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:

- direct on field observations;
- open discussions and informal meetings with a small group of simulation analysts;
- requirements elicitation survey;
- discussions 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 [9]) **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 [17] as explained by Dix et al. [6] 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 [6].

One could ask why it is important to identify all the stakeholders. In the book "*Human-Computer Interaction*" [6], the authors described an example of organisation 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. One must be really careful to not disrupt these communication schemes, like the hierarchy, introducing a new system.

2.3.2 Domain understanding

Software Engineers often need to deal with new domains that are far from their field of studies. Therefore, they need to learn a new technical language and identify the correct keywords to establish an efficient communication with the identified stakeholders. It is exactly the case described here, stakeholders are Computational Fluid Dynamics simulation domain experts. The first step is to learn how they work everyday. This has iteratively done through: (1) workday observation, (2) open discussions, (3) stakeholders interviews, (4) meetings, and (5) a survey [6]. To get a deep insight into the domain, all the previous activities must be performed. In addition, these activities involve the stakeholders from the beginning so they are aware about the design decisions.

It is fundamental to identify the purpose of the research study. There are four types of purposes [18] although sometimes it is really difficult to effectively classify a study:

- **Exploratory**: to find out what is happening and seek new insights as well as generate new ideas;
- **Descriptive**: to describe a phenomenon;
- **Explanatory**: to seek an explanation of a situation or a problem
- **Improving**: to improve an aspect of the studied phenomenon.

In this work, 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 [18]. 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 [18], the research methodologies are depicted in Table 2.2.

2.3.3 Impact of the a new system in the organisation

One reason for which the introduction of new systems fails is due to the mismatch between information systems and organisational and social factors [6]. Another consideration is the impact of the technology introduction within the organisation. The impact should be assessed and evaluated before its introduction [6] 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 grater then costs.

2.4 Collaborative Requirements Survey

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 the 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[2]. 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.

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.

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

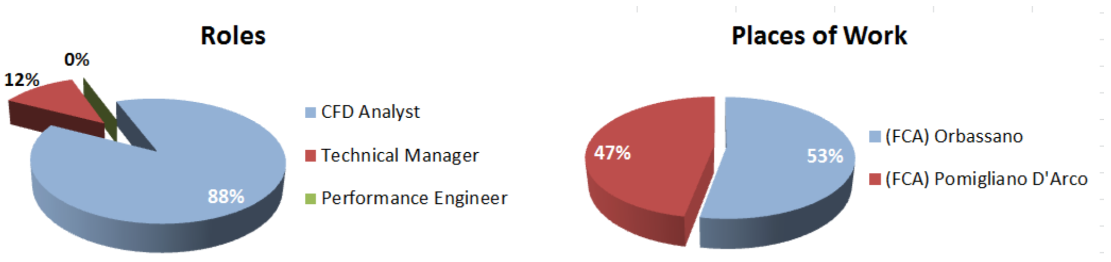


Figure 2.2: Participants' roles and their working place (*questions Q1 and Q2*).

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.

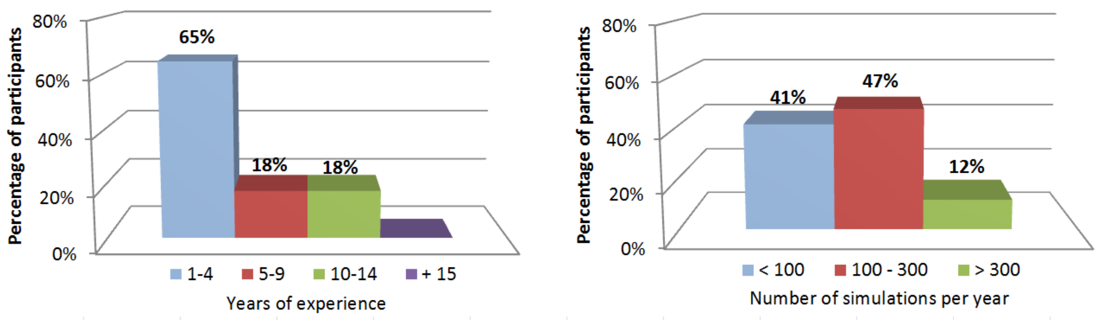


Figure 2.3: Participants' experience.

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

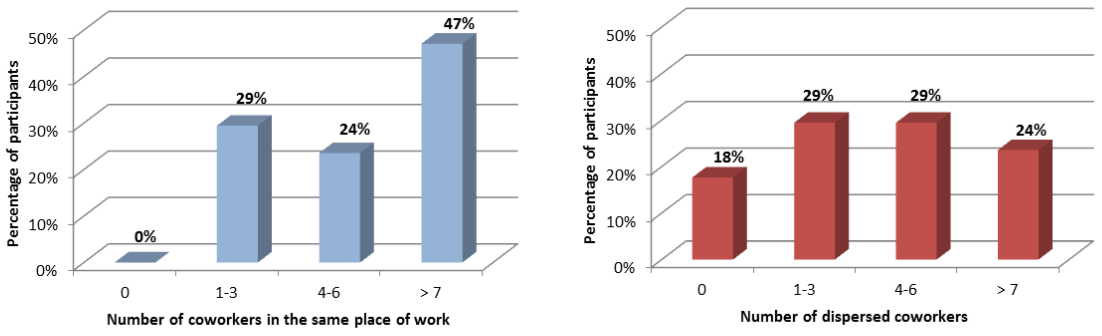


Figure 2.4: Co-located (*question Q8*) vs distributed workers (*question Q9*).

them in an open format.

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*" [4], in order to improve the company competitiveness, they should centralise 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.

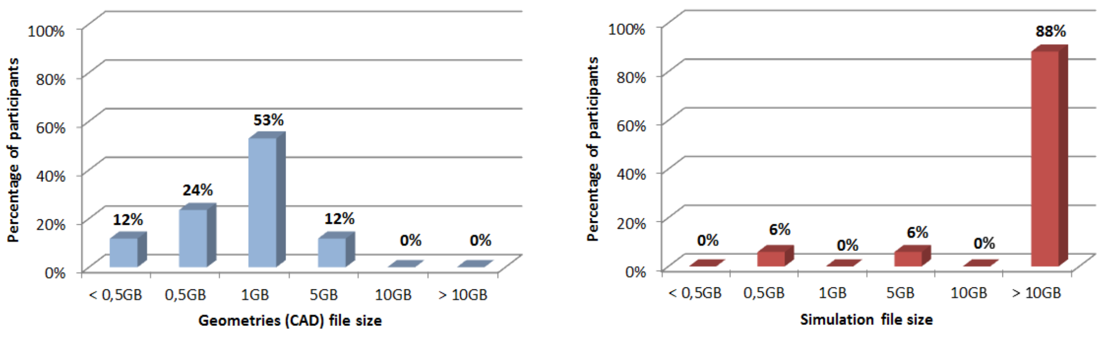


Figure 2.5: Geometries and simulations file size (*question Q10 and Q11*).

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 (questions **Q8** and Q**9** in the Survey in Appendix A). 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 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 the next time to find the data.

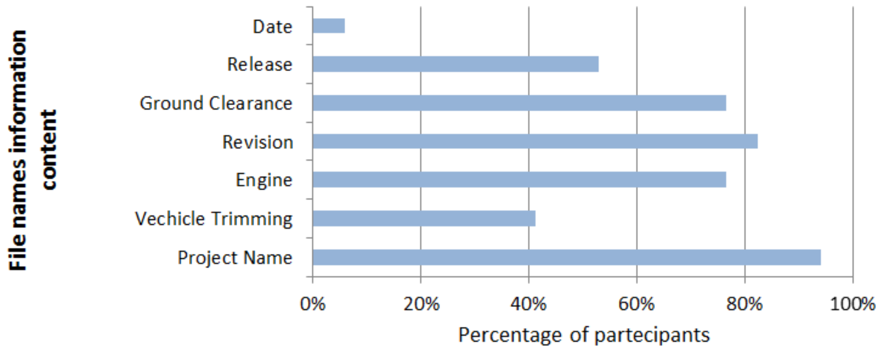


Figure 2.6: Information stored in the file names (*question Q20*).

As shown in Fig. 2.6, the main information inserted in the file name are (*questions Q21-Q28*): 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 can not be used for next retrievals.

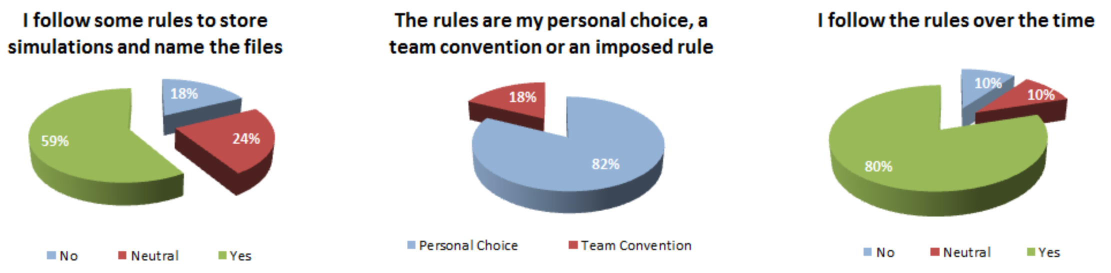


Figure 2.7: Rules to store files (*questions Q19, Q21, Q22*).

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 can not use the existing search tools (e.g., the Windows Serach 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.

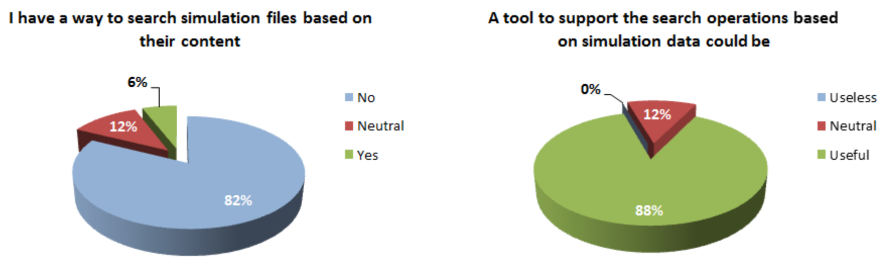


Figure 2.8: Search based on file content (*questions Q27 and Q28*).

To further explore the search of files, we asked to stakeholders (question Q27) 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 (question Q28); 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 and the Unix and Windows find tools.

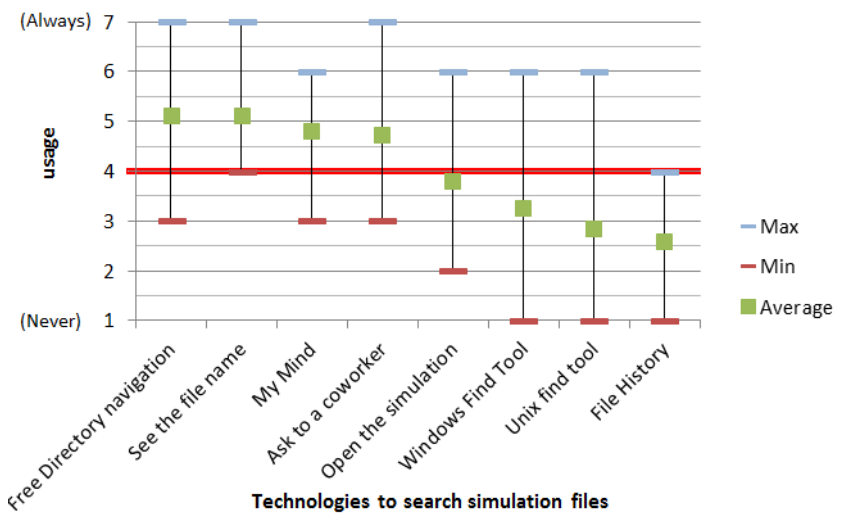


Figure 2.9: Technologies to search simulations (*question Q31*).

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 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. 2.6). Obviously, the file name can not 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 (question Q25). All analysts (100%) desire a system to link other information to the files, such as the file tagging.

2.8 Simulation data versioning

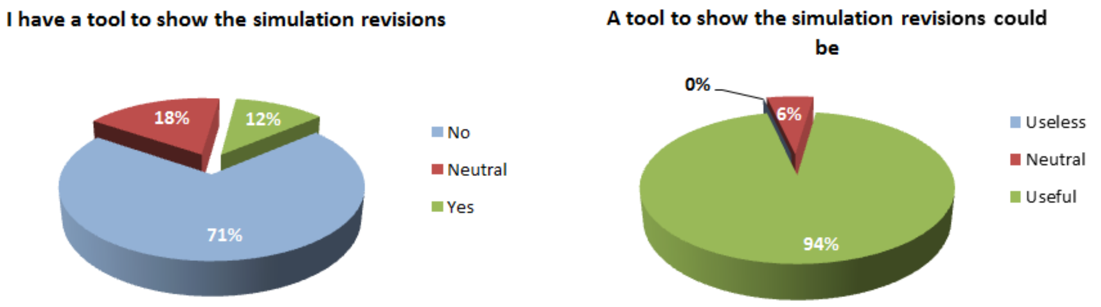


Figure 2.10: Version control (*question Q30*).

As reported in the Aberdeen Group market research [4], 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 can not 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 (multiple choices question Q31) 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. This is a key difference with other platforms (i.e., e-Science) that often integrate simplified or in-house developed solvers [19].

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 [20] otherwise simulation data are locked in the vendor software. Vendor Lock-In is a well-known Anti-Pattern [2] [21] [22]: the phenomenon that causes customer dependency on given vendor about a specific good or service [23] with high switching costs [24]. 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 [25]. 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 [26] and is an obstacle to cloud computing [27]. 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 [28]. A design solution useful to mitigate the Vendor Lock-In is to design the system with an additional layer called isolation layer [2].

2.11 Extensibility and modularity

The combination of modularity and extensibility [29, 30] 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 customers needs. Extensibility is the ability of a software system to allow and accept significant extension of its capabilities without major rewriting of code [29] [30]. 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 [31].
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*" [29]. 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 (EPL[3] 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[4]). 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.

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,

[3]EPL license web site `http://www.eclipse.org/legal/epl-v10.html`
[4]Elgg official web site: `http://elgg.org/`

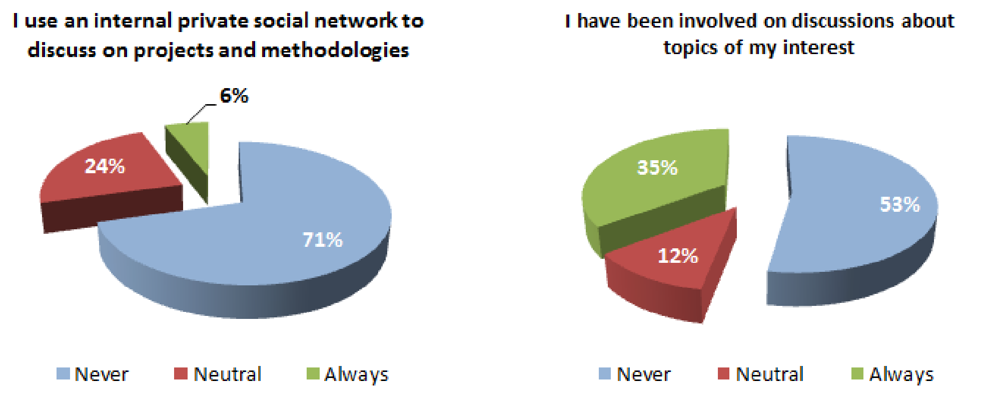


Figure 2.11: Actual use of private social networks (*Questions Q31 and Q32*).

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).

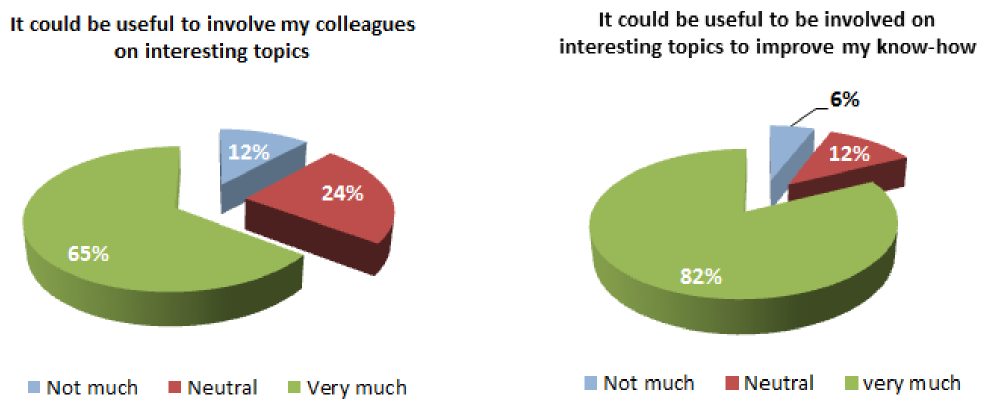


Figure 2.12: Introduction of social networks (*Questions Q33 and Q34*).

About the use of social networks to dicuss on simulation data, through the survey (question Q35) 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 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. In addition, it opens further investigation towards the use of social networks.

Chapter 3

Engineering Requirements

Contents

This chapter introduces, analyses and discusses the engineering Functional and Non-Functional requirements to support daily engineers simulation activities automating repetitive, time consuming and error prone tasks. These engineering requirements have been gathered through interviews with stakeholders, daily work observations and an on-line survey. In the considered use case in Fiat Chrysler Automobiles, engineers design vehicles using computer simulation following a standard simulation workflow called

Computational Fluid Dynamic (CFD) Workflow. Through the extensive interviews activities with stakeholders, the observations, a survey and the existing literature [4], the main gathered engineering functional requirements can be divided in the following main points:

- provide industrial templates and wizards for less experienced users and conform the process followed by the engineers to perform simulations;
- provide tools to automatise, support and improve manual, repetitive and error-prone engineering daily tasks;
- integrate industrial software, HPC resources and industrial know-how.

These features must be provided directly within the simulation workflow and their achievement represents the basic foundation to provide a service that automatically performs the Multi-Disciplinary and Multi-Objective simulations.

This chapter is organised as follows. It introduces briefly the engineering context and the simulation workflow. Then, 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.

3.1 Introduction

Through the closely work with engineers to gather the Collaborative Requirements, stakeholders highlighted additional engineering requirements to improve their daily work. Practically during the discussions with the engineers it is very difficult to get what they desire in terms of collaborative features but instead they are more likely to discuss about their daily work and the issues that practically they face. And, of course, collaborative requirements within an engineering context can not be completely separated from the engineering needs. Based on these motivations, this chapter introduces the engineering context and the engineering requirements that are related mainly to tools to automate repetitive, time consuming and error-prone tasks. Of course, the aim is not to replace existing simulator software and functionalities, but instead to integrate them together to provide new services.

3.1.1 Introduction to the Simulation Workflow

Briefly the Simulation Workflow is made by three steps (Figure 3.6): 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.

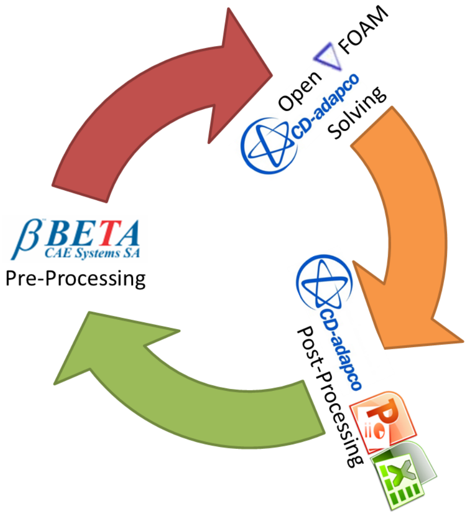


Figure 3.1: CFD Workflow: software used by analysts.

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®[1] developed by BETA CAE Systems S.A. Another motivation to use a pre-processor is the need to clean up the geometry and create both the surface and volume mash 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 3.3. 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

[1]The ANSA official web site is `http://www.beta-cae.gr/`.

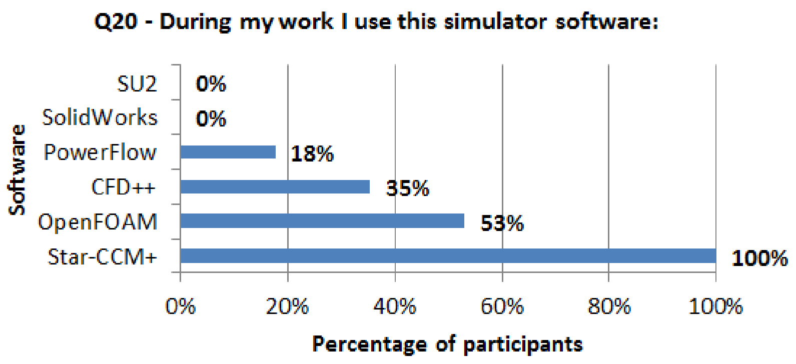


Figure 3.2: Simulation software used to simulate vehicles.

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.

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 3.3, 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.

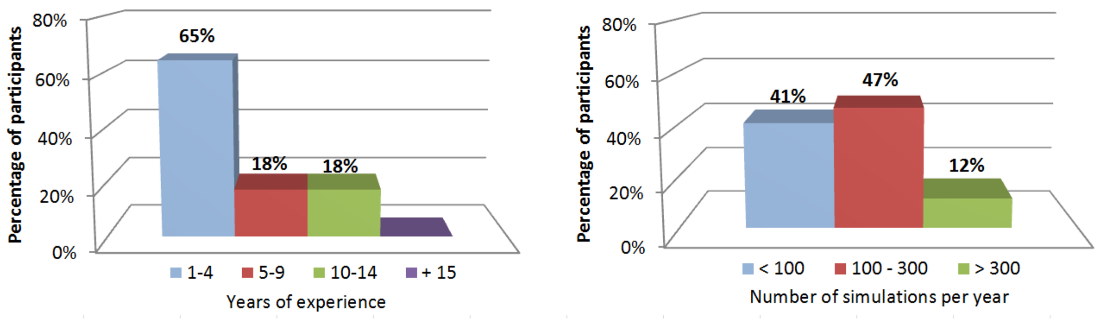


Figure 3.3: Participants' experience.

3.1.2 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 every one in the team works in the same manner.

The "*Getting Product Design Right the First Time with CFD*" [4] 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 [4], 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.

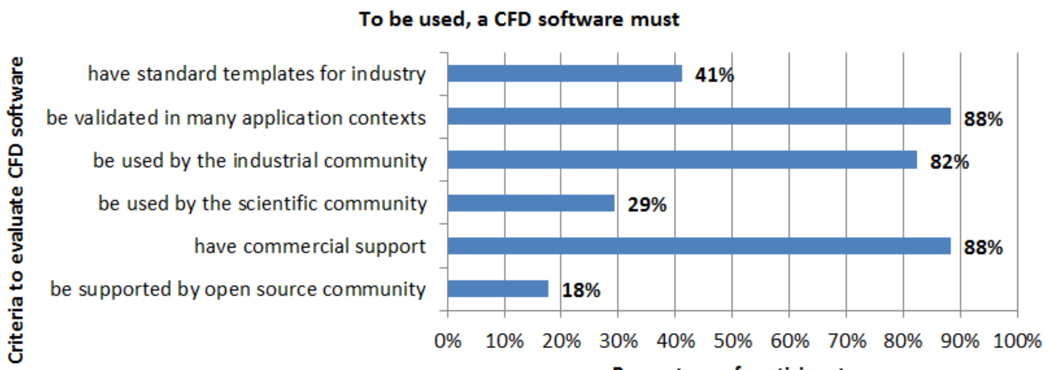


Figure 3.4: Criteria used to evaluate a CFD solver (*Question* Q32).

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. 3.5. 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*" [4], 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 introduced in the Chapter 2 and performed within FCA, 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.

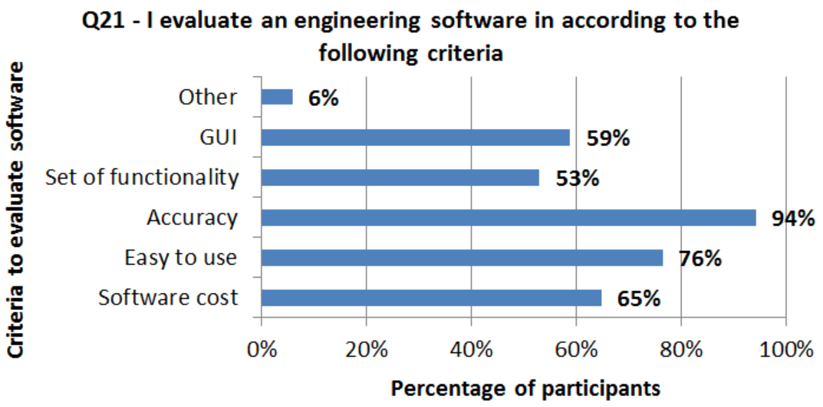


Figure 3.5: Criteria used to evaluate an engineering software (*Question* Q33).

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.

3.2 Requirements Overview

Requirements have been divided in two main categories [9]: Functional and Non-Functional requirements. Functional requirements define the system functions so they concern mainly the system behaviour 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 3.1 lists briefly the main functional requirements gathered within the use case, instead the table 3.2 lists the Non-Functional requirements.

The table 3.2 of Non-Functional requirements, contains also additional constraints to follow that come to light during the discussions with engineers and stakeholders. Engi-

	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 3.1: Stakeholders' key engineering Functional Requirements.

neers 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). 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 (Re. 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 3.2: Stakeholders' key engineering Non-Functional Requirements.

3.3 Functional Requirements

This section describes briefly the Functional requirements gathered in the Fiat Chrysler Automobiles spa 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 3.6 shows above the pre-processing phase the Run simulation, a wizard of multiple pages to information like the cluster to user

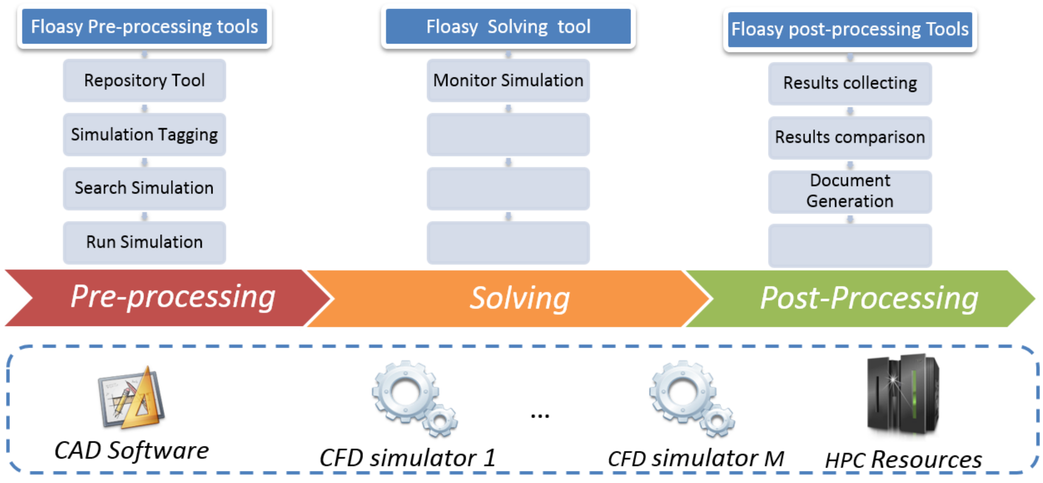


Figure 3.6: CFD Workflow and Floasys Tools

and the number of processors. The provided wizard is not only a direct replacement of the command line but provides additional pages that depend on the simulation to run.

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 [32], 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.

3.4 Non-Functional Requirements

This section introduces and describes the Non-Functional requirements.

3.4.1 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. 3.2): 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.

3.4.2 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.

3.4.3 Concurrent simulators use

Analysts use many instances of the same simulator software opening multiple files and running many solving jobs. The platform must support consider two cases: the support of multiple different simulators concurrently and multiple instances for the same simulator.

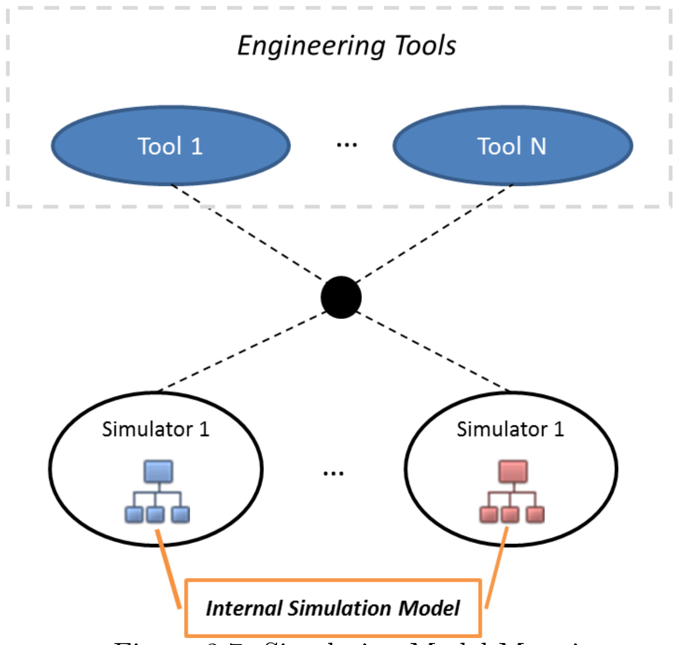


Figure 3.7: Simulation Model Mapping.

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.

Concurrent simulators use

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.

3.4.4 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 [33] 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[2], 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.

[2]Esteco modeFrontier Official Web site: `http://www.esteco.com/modefrontier`

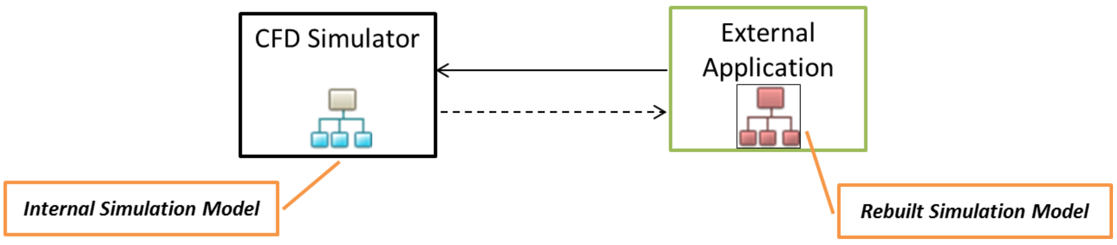


Figure 3.8: Simulator black-box interaction.

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.

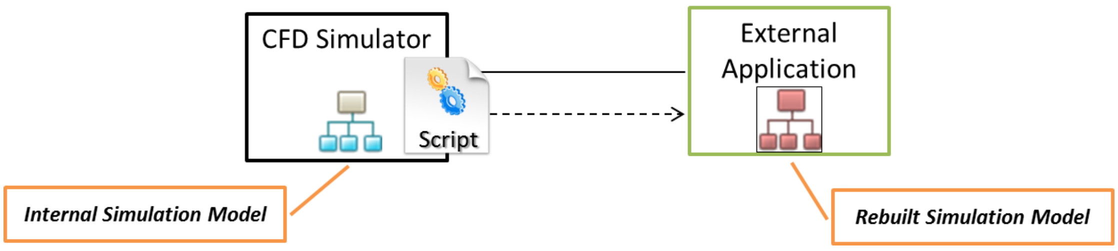


Figure 3.9: Interaction with a simulator through a macro.

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 can not 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 [2] [21] [22]: the phenomenon that causes customer dependency on given vendor about a specific good or service [23] with high switching costs [24]. 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 system architecture is product-dependent [25] 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 [26] and is an obstacle to cloud computing [27]. 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 [28].

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 [29] [30]. Extensibility is an important Non-Functional requirement during the system development because systems are constantly under change and, adopting the Continuous Delivery [31], 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*" [29].

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.

3.5 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.

3.5.1 New Product Development Process

Manufacturers aim is to bring products on market quickly within the budget and performance constraints [4]. The New Product Development Process (PDP) describes the process adopted to design, develop and bring products on the market [34] and involves a continuous information exchange among many tasks. As shown in Figure 3.10 it is made by the following steps: concept, design, prototyping, manufacturing.

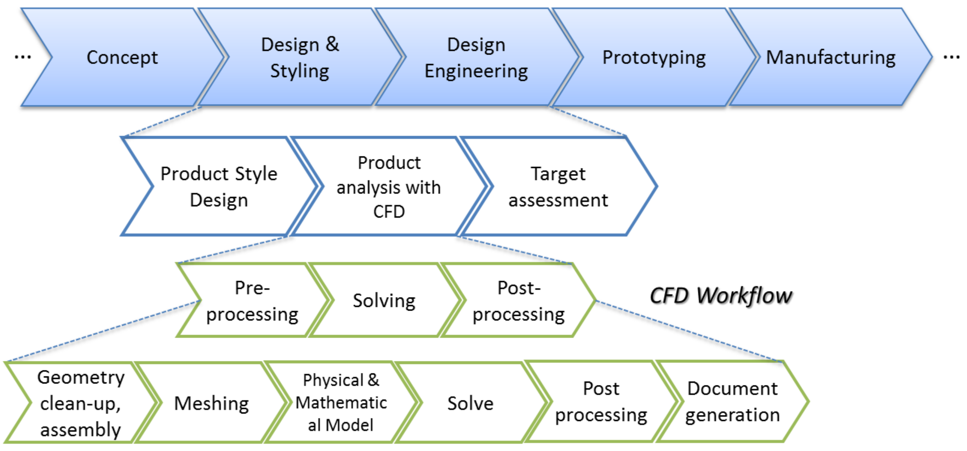


Figure 3.10: Product Development Process (PDP) and CFD Workflow.

Nowadays, in a world wide and highly competitive market, enterprises face short time to market (TTM), continuous innovations, global collaborations and complex risk management [35]. 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 externalises 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 [35] 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 created so its 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 [36]. PDM systems were born in the engineering field and they used mainly to store information like geometric models, Bill of materials (BOM) and FEA 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 designed for engineers to manage engineering data. Later, in the 1990s, with the introduction of Internet and WWW, vendors adopted these new technologies to the 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 its self 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.

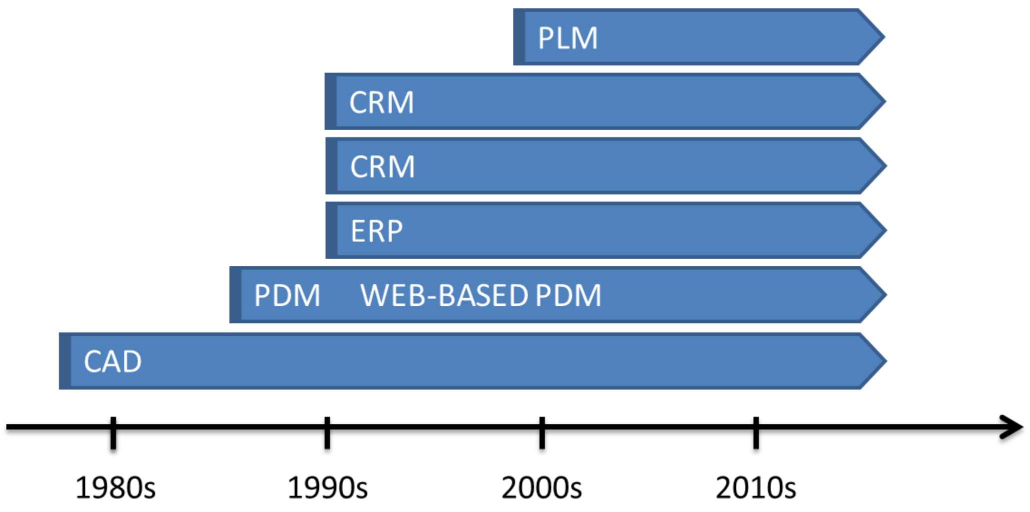


Figure 3.11: Product Lifecycle Management Evolution

From historical point of view, the evolution of PLM and PDM is tracked roughly in the Figure 3.11. 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 [36], 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[3] an open source product, but many other product exist.

3.5.2 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

[3]Aras official web-site `http://www.aras.com/`

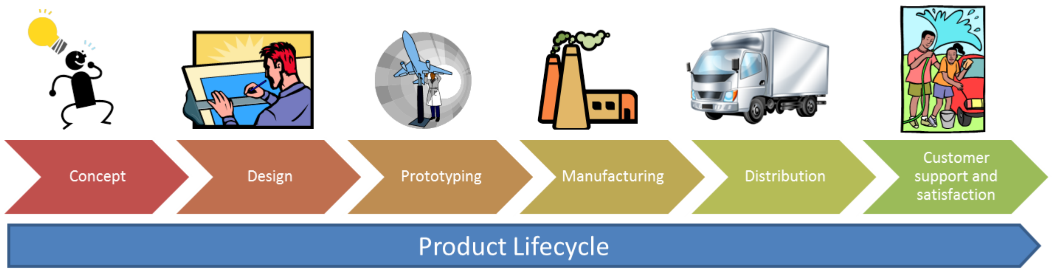


Figure 3.12: Product Lifecycle

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 behaviour. 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. 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*" [4], 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. 3.10): 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.

3.5.3 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 iterate 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

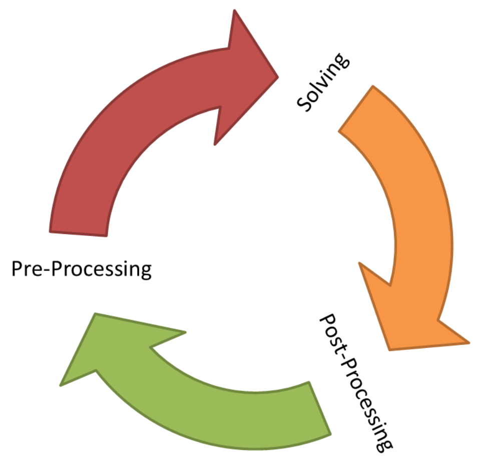


Figure 3.13: CFD Simulation Workflow.

engineers. 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. 3.14) 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[4]). 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 may 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 [37].

[4]ANSA BETA CAE Systems SA website `http://www.beta-cae.gr/ansa.htm`.

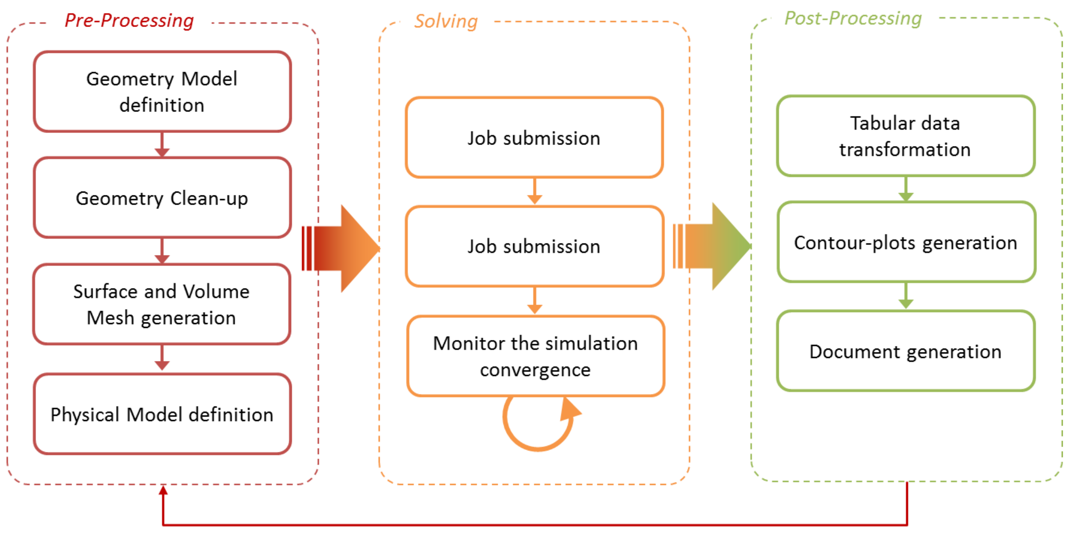

Pre-Processing
Geometry Model definition
Geometry Clean-up
Surface and Volume Mesh generation
Physical Model definition
Solving
Job submission
Job submission
Monitor the simulation convergence
Post-Processing
Tabular data transformation
Contour-plots generation
Document generation

Figure 3.14: CFD Simulation Workflow detailed phases.

3.5.4 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) [19] is an educational Web portal developed in Korea to help students to understand the aerodynamic simulation process [38]. EDISON_CFD [39] is the e-AIRS improvement in terms of stability, faster data response time and waiting time [40, 41]. 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 [19]. 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 [42]. It allows the "*dispatching and controlling of long-running simulations*" [42]. 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 [43] 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 [43] developers are working on the integration of OpenFoam®, an open source CFD software package.

3.6 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 3.15): 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 behaviour. For more details about the Simulation Workflow phases see the Section 3.5.3.

The Simulation Workflow is iterative. Actually, each iteration takes hours to be completed (sometimes 24hours) 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[5] (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

[5]Ride height explanation `http://en.wikipedia.org/wiki/Ride_height`

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 (Fig. 3.15). 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 3.15.

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 form 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

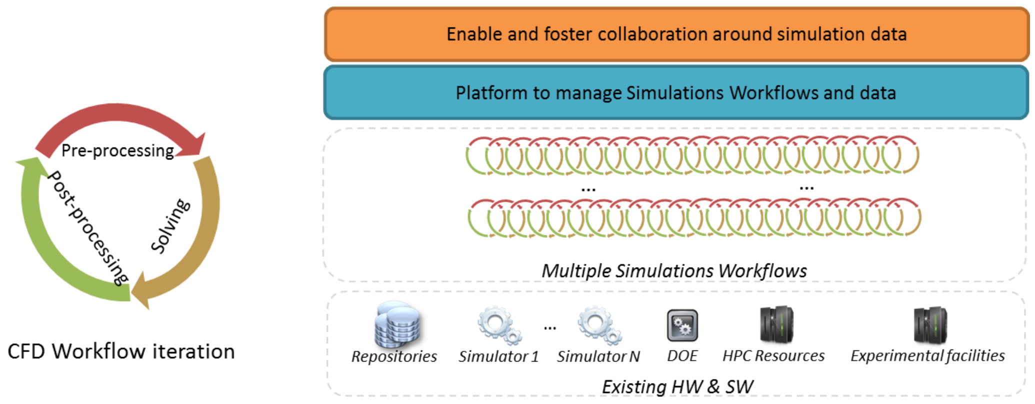


Figure 3.15: Multiple Workflows Management.

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 [2].

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 can not 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.

3.6.1 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 can not 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).

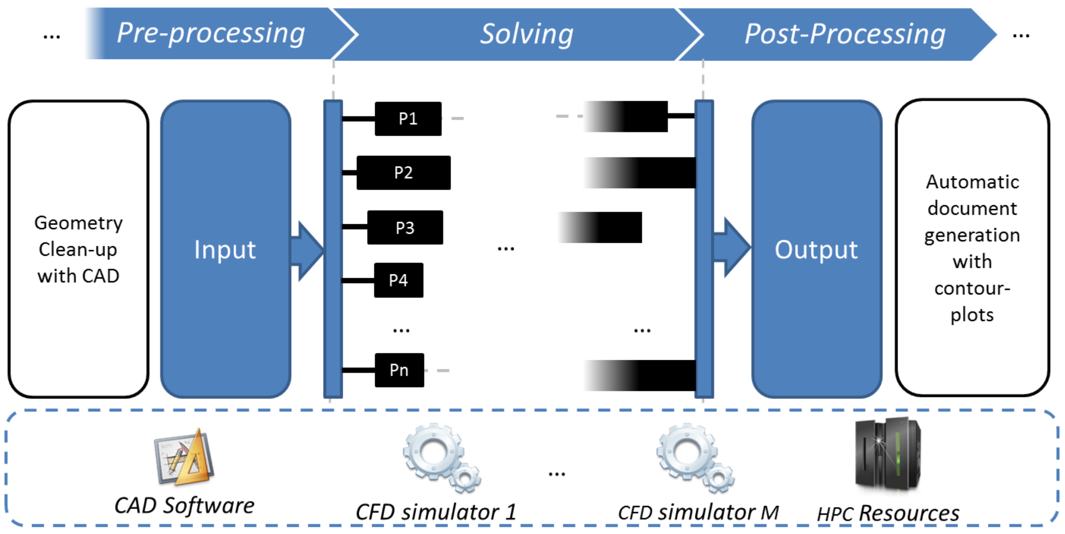


Figure 3.16: Multidisciplinar, mutliobjects and automatic CFD workflows.

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.

3.6.2 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.

3.7 Conclusions

This 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. The requirements described here have been implemented in a prototype (described in the next Chapter 4) used within the Aerothermal CFD of Pomigliano D'Arco (NA).

Chapter 4

Floasys

Contents

4.1 Introduction

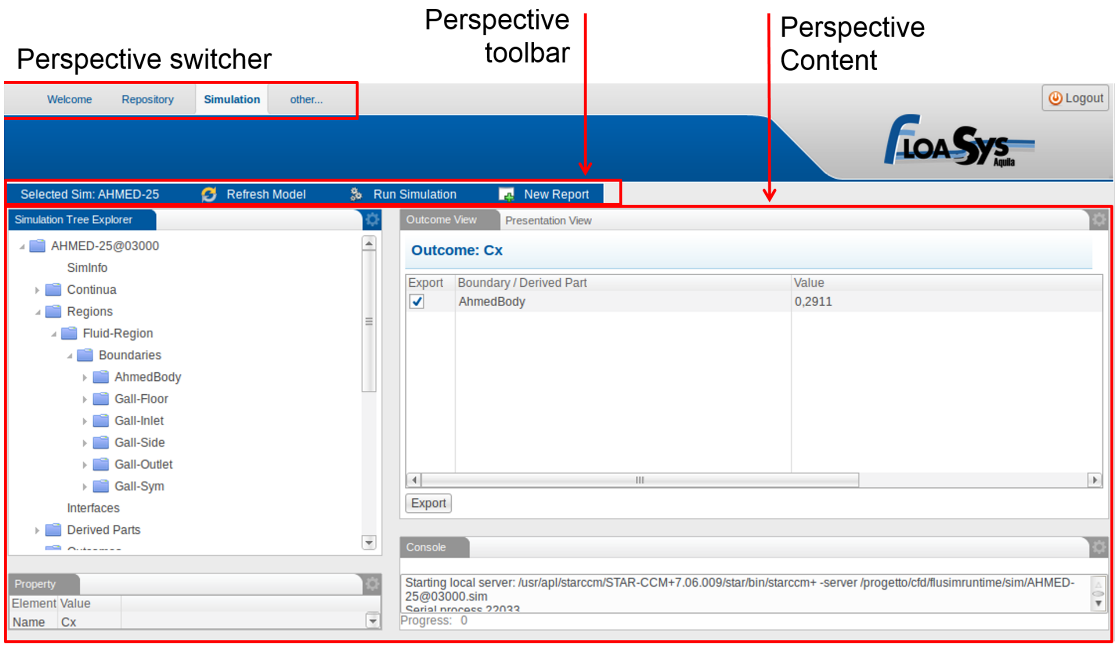
This chapter describes the Floasys functionalities and shows its graphical user interface (GUI). Floasys functions have been divided in two groups: collaborative and engineering functionalities.

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.

The chapter is mainly organised in two parts to reflect the Floasys functionalities subdivision and to describe them in a homogeneous way. Both parts describe a typical workflow and then the Floasys graphical user interface to support it. For instance, the Section 4.3 describes the typical collaborative workflow supported by Floasys.

4.2 Floasys Graphical User Interface



Element	Value
Name	Cx

Outcome: Cx

Export	Boundary / Derived Part	Value
✓	AhmedBody	0,2911

Starting local server: /usr/apl/starccm/STAR-CCM+7.06.009/star/bin/starccm+ -server /progetto/cfd/flusimruntime/sim/AHMED-25@03000.sim
Serial process 22033
Progress: 0

Figure 4.1: Floasys Graphical User Interface.

Floasys provides a re-configurable GUI based on Perspectives and Views concepts provided by Eclipse Remote Application Platform (RAP) [44]. 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 organised in multiple views. Each view effectively contains the data using the available widgets. For instance Figure 4.1 shows the "Simulation Controller Perspective" with four views: Simulation Tree Explorer, Property, Outcome View and Console.

4.3 Collaborative Features

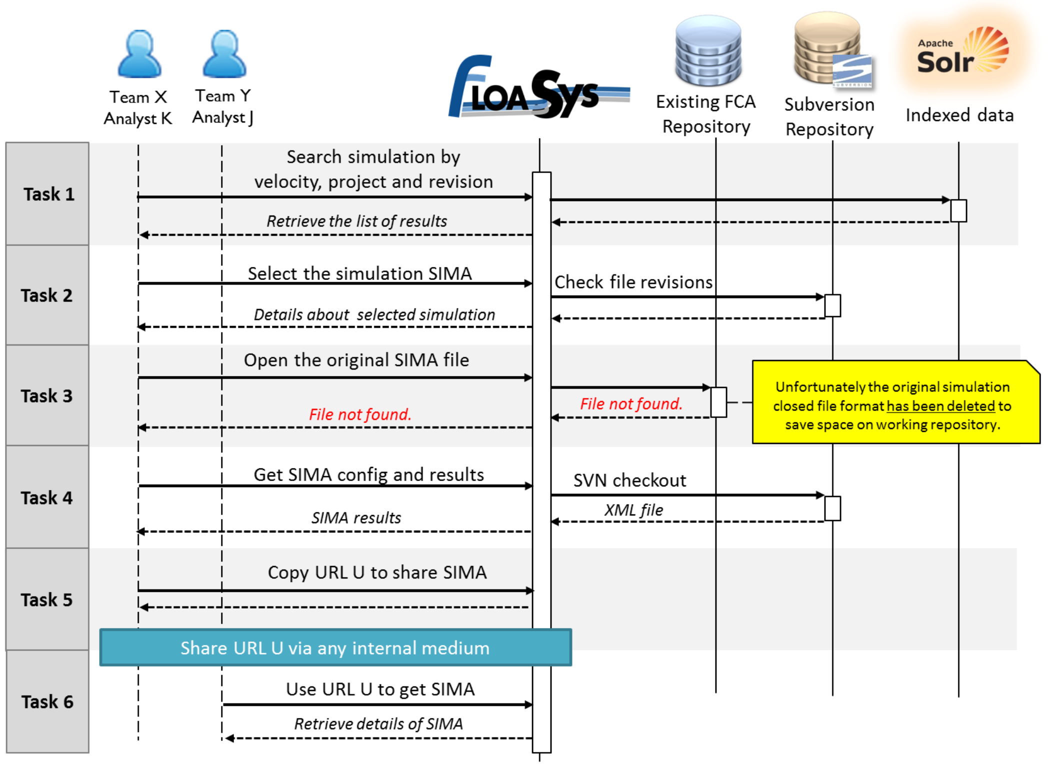
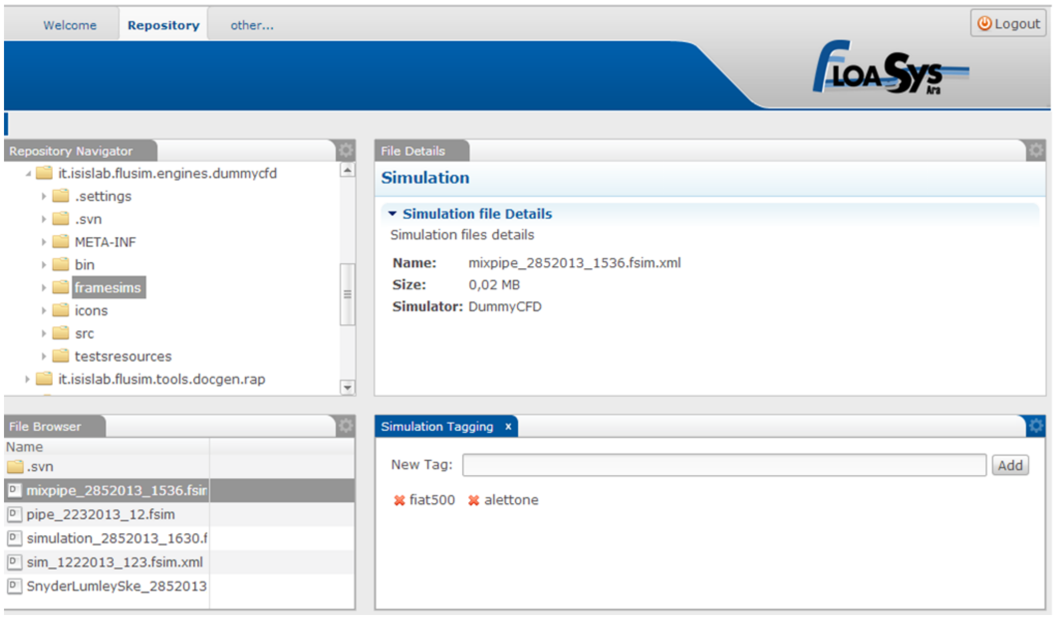

Team X
Analyst K
Team Y
Analyst J
FLOASYS
Existing FCA Repository
Subversion Repository
Apache Solr
Indexed data
Task 1
Search simulation by velocity, project and revision
Retrieve the list of results
Task 2
Select the simulation SIMA
Details about selected simulation
Check file revisions
Task 3
Open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

Figure 4.2: Example of a typical workflow supported by Floasys.

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 4.2 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 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 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.

4.3.1 Repository Tool and Simulations Tagging



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

Figure 4.3: Repository tool to navigate and tag a simulation repository.

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 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. 4.3). Tags are both unstructured with free tags and structured inserted filling out standard forms like in Figure 4.4.

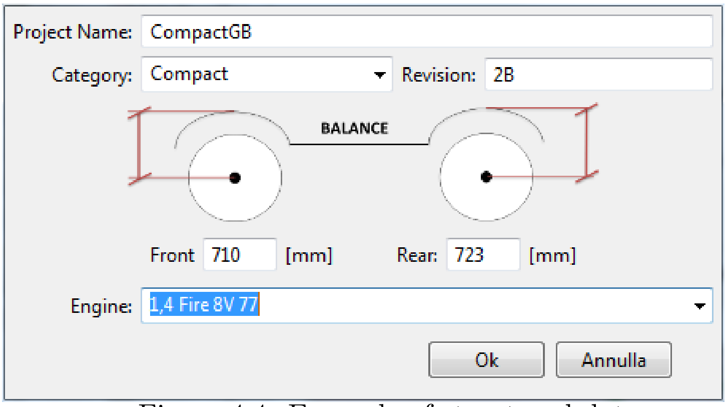

Project Name: CompactGB
Category: Compact
Revision: 2B
Front 710 [mm]
Rear: 723 [mm]
Engine: 1,4 Fire 8V 77

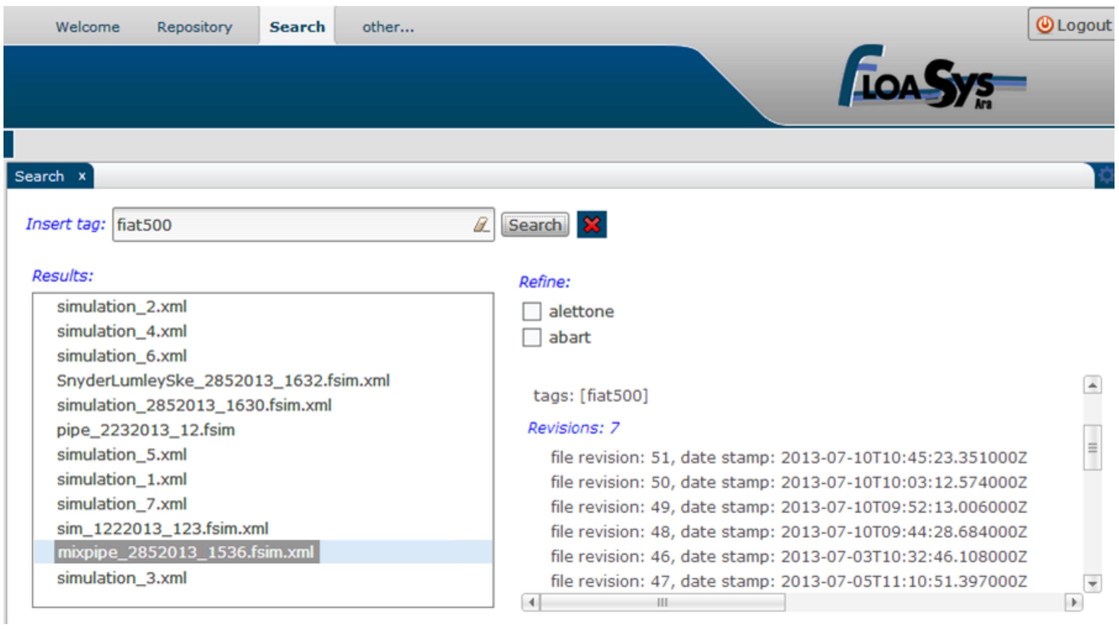
Figure 4.4: Example of structured data.

4.3.2 Search tool and Data sharing

The Search tool (Fig. 4.5) 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. 4.5). 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



Insert tag: fiat500

Results:

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

Refine:

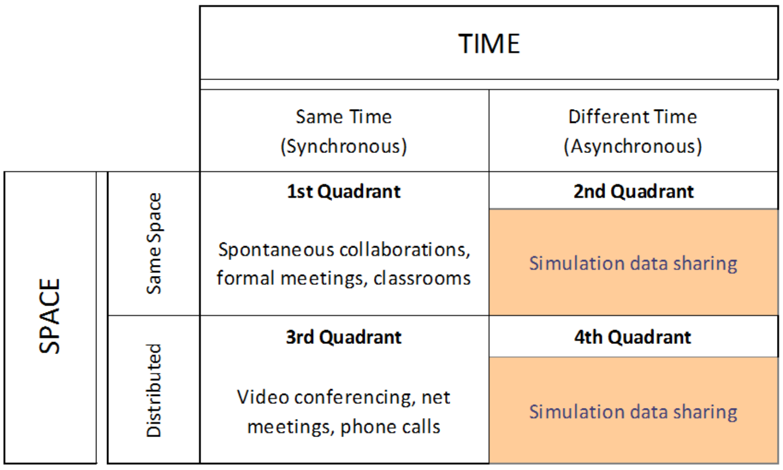
☐ alettone
☐ abart

tags: [fiat500]

Revisions: 7

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

Figure 4.5: 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 4.6: CSCW Quadrants.

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 [45], Floasys supports the *asynchronous* data sharing for both co-located and distributed teams.

4.3.3 Web-based 3D Model Visualisation

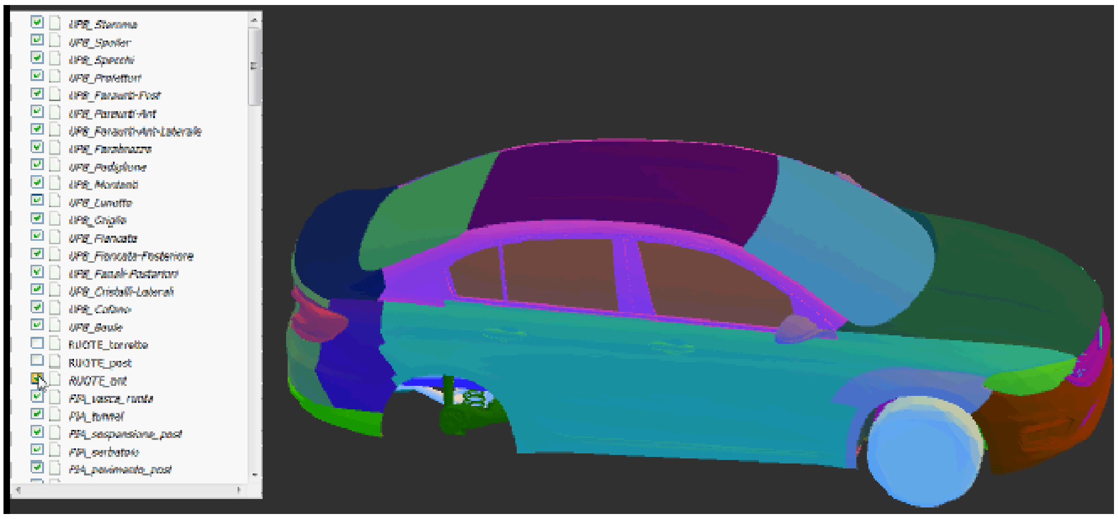


Figure 4.7: Floasys 3D model visualisation.

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. 4.7). The user can activate or deactivate some parts and can perform the basic zoom and pan operations. Figure 4.7 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 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.

4.4 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 4.8 shows the CFD Workflow and for each step shows the functionality supported by Floasys.

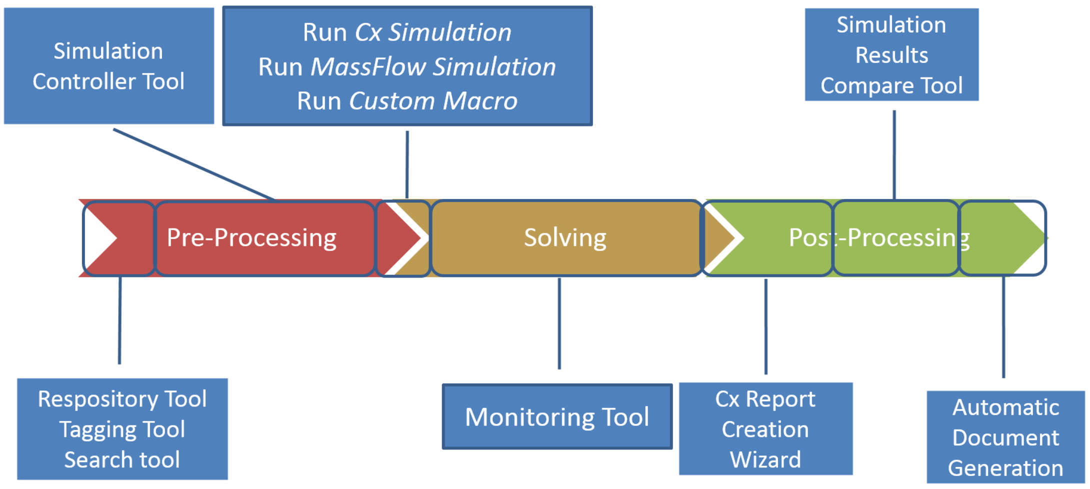

Simulation Controller Tool
Run *Cx Simulation*
Run *MassFlow Simulation*
Run *Custom Macro*
Simulation Results Compare Tool
Pre-Processing
Solving
Post-Processing
Respository Tool
Tagging Tool
Search tool
Monitoring Tool
Cx Report Creation Wizard
Automatic Document Generation

Figure 4.8: CFD Workflow and tools provided by Floasys.

In order to understand how the users interact with the system, the Figure 4.9 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. 4.9). Then, the "*Simulation Controller Tool"* opens the simulation and shows its content and details (Task C, Fig. 4.9).

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. 4.9) 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. 4.9) and the tool to generate the documentation automatically (Task E, Fig. 4.9).

4.4.1 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

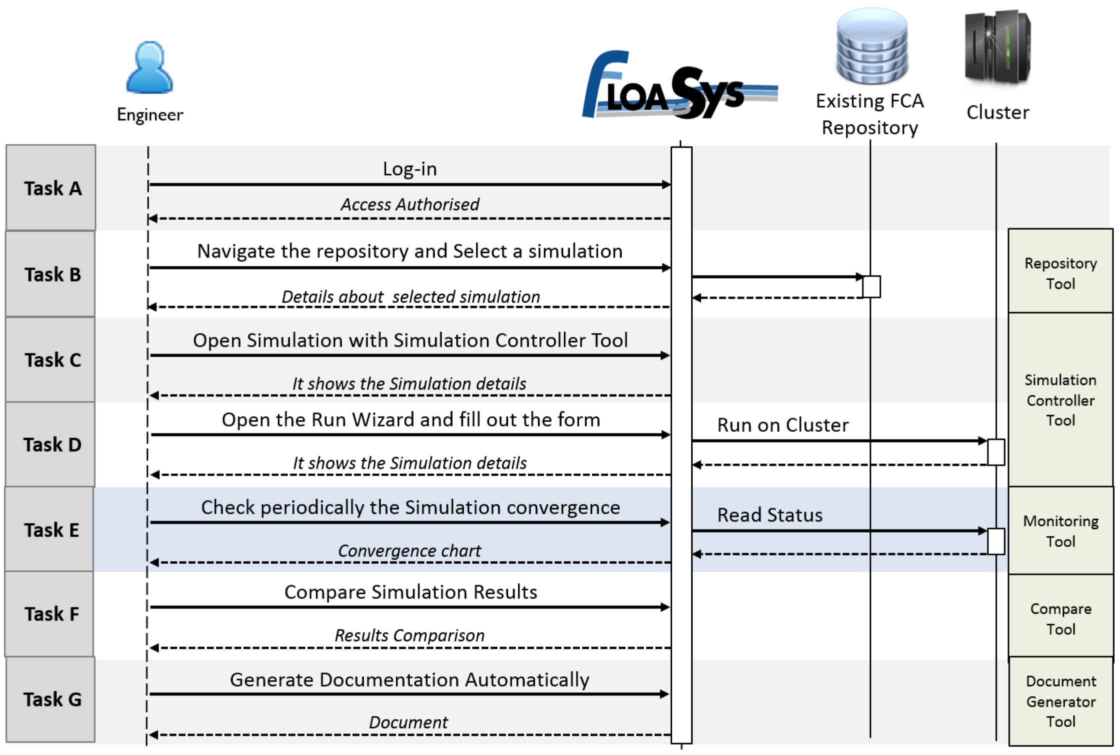
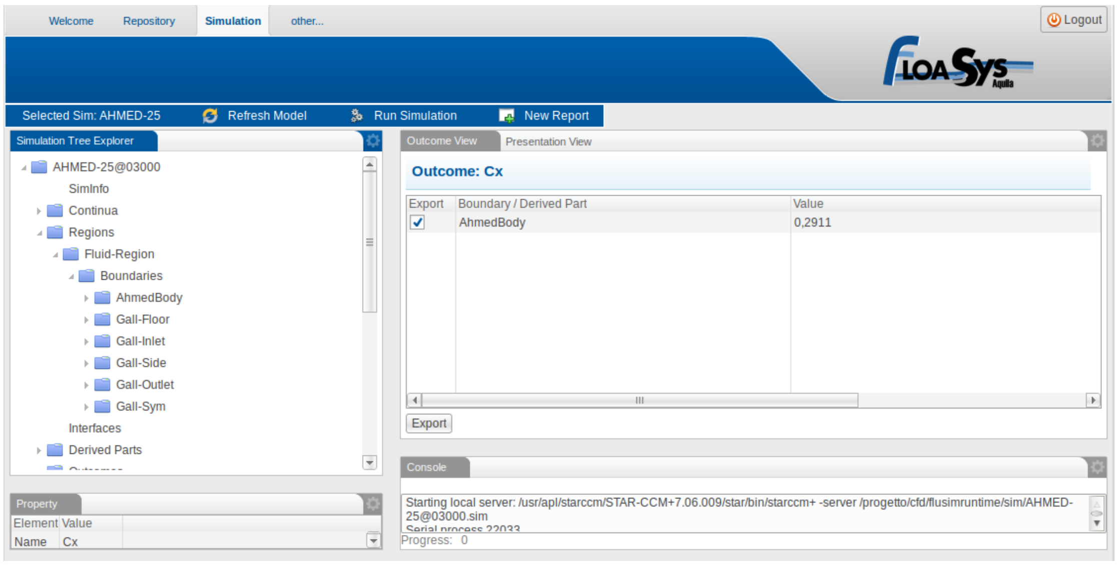

Engineer
FLOASYS
Existing FCA Repository
Cluster
Task A
Log-in
Access Authorised
Task B
Navigate the repository and Select a simulation
Details about selected simulation
Repository Tool
Task C
Open Simulation with Simulation Controller Tool
It shows the Simulation details
Simulation Controller Tool
Task D
Open the Run Wizard and fill out the form
Run on Cluster
It shows the Simulation details
Task E
Check periodically the Simulation convergence
Read Status
Convergence chart
Monitoring Tool
Task F
Compare Simulation Results
Results Comparison
Compare Tool
Task G
Generate Documentation Automatically
Document
Document Generator Tool

Figure 4.9: A typical engineering workflow.

the "*Repository Tool*". The screenshot in Figure 4.10 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.

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.



Outcome: Cx

Export	Boundary / Derived Part	Value
✓	AhmedBody	0,2911

Export

```
Starting local server: /usr/apl/starccm/STAR-CCM+7.06.009/star/bin/starccm+ -server /progetto/cfd/flusimruntime/sim/AHMED-
25@03000.sim
Serial process 22033
Progress: 0
```

Element	Value
Name	Cx

Figure 4.10: Simulation Controller Tool.

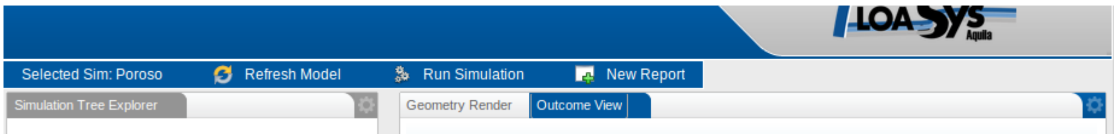


Figure 4.11: Simulation Tool Toolbar

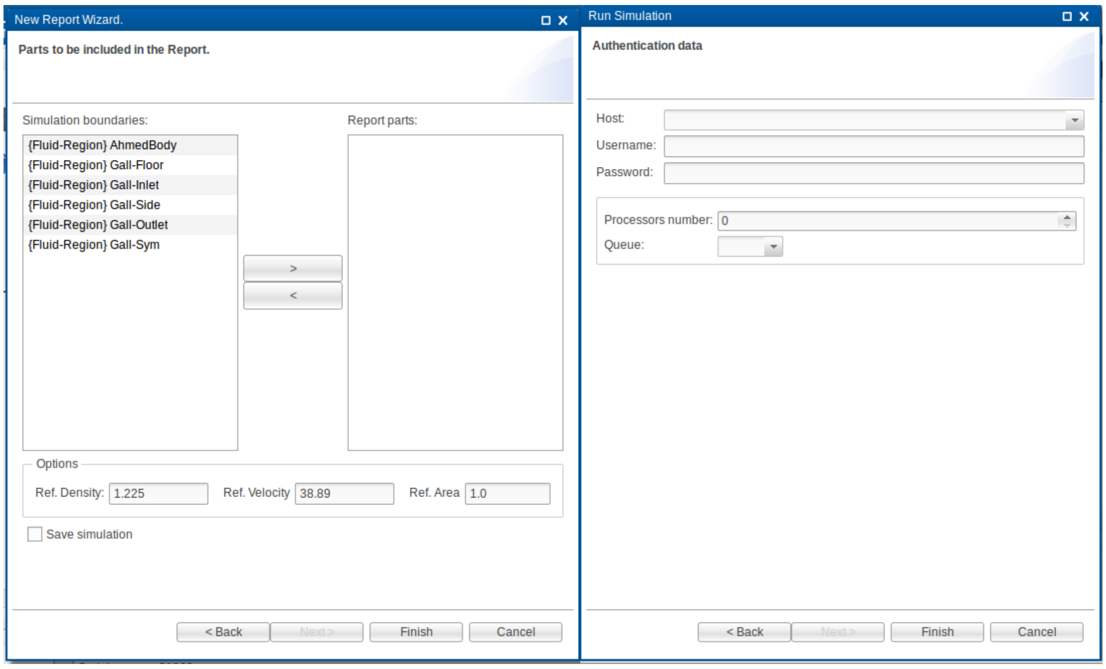
Figure 4.11 shows the Simulation Controller Tool toolbar: the main entry to other functionality of the Selected Simulation. This toolbar continuously evolves providing new features, currently it contains the following functionalities:

1. **Selected Sim**: it shows the selected simulation name, if the simulation is stored in a file than the selected simulation is the file name;

2. **Refresh model**: to refresh data shown, it extracts the simulation data and shows them in the simulation tree under the toolbar;

3. **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. 4.12 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 4.12 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,


New Report Wizard.

Parts to be included in the Report.

Simulation boundaries:

{Fluid-Region} AhmedBody
{Fluid-Region} Gall-Floor
{Fluid-Region} Gall-Inlet
{Fluid-Region} Gall-Side
{Fluid-Region} Gall-Outlet
{Fluid-Region} Gall-Sym

Report parts:

Options

Ref. Density:	1.225	Ref. Velocity	38.89	Ref. Area	1.0

☐ Save simulation

Run Simulation

Authentication data

Host:	
Username:	
Password:	
Processors number:	0
Queue:	

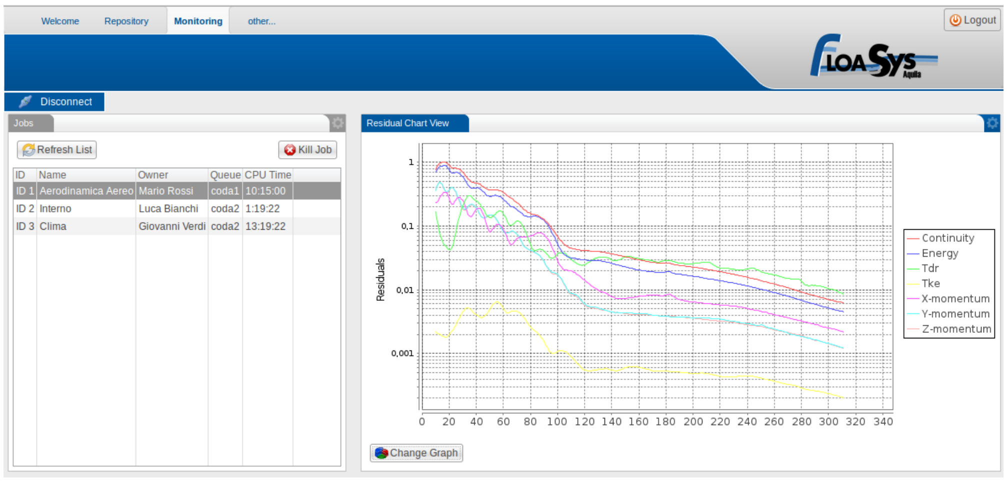
Figure 4.12: Screenshots from the Run Simulation Wizard.

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.

4.4.2 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 4.13 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. The X axis usually shows the number of iterations.



ID	Name	Owner	Queue	CPU Time
ID 1	Aerodinamica Aereo	Mario Rossi	coda1	10:15:00
ID 2	Interno	Luca Bianchi	coda2	1:19:22
ID 3	Clima	Giovanni Verdi	coda2	13:19:22

Figure 4.13: MonitoringTool to check the simulation convergence.

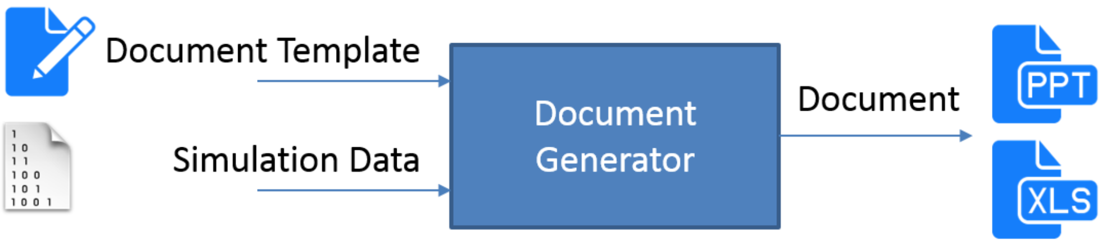
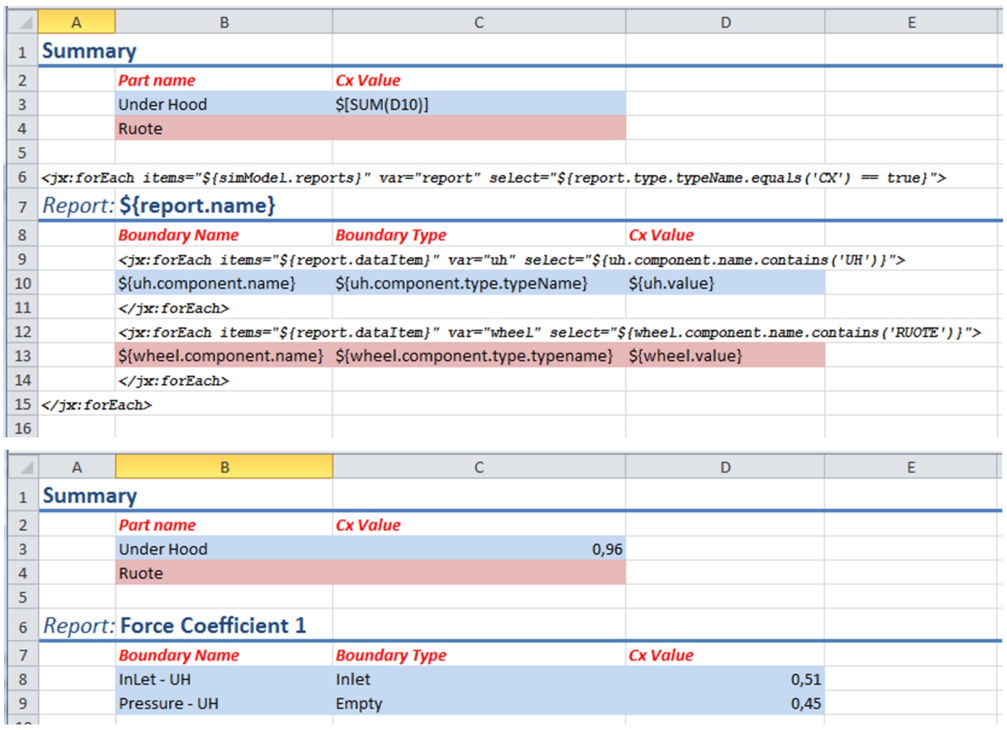


Figure 4.14: Generation of documents from simulation data.

4.4.3 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 can not 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 4.14 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 can not perform.

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



	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 4.15: Example of a template and a generated document.

Java template engines. Figure 4.15 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.

4.4.4 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[1] and Dakota[2]. 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 V_x is the set of values for the variable x. The set of experiments is $E \subseteq (V_{x_1} x ... x V_{x_n})$. 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 4.16 shows a pipe with two inlet

[1]Esteco modeFrontier official web-site: `http://www.esteco.com/modefrontier`

[2]Dakota official web-site: `https://dakota.sandia.gov/`

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

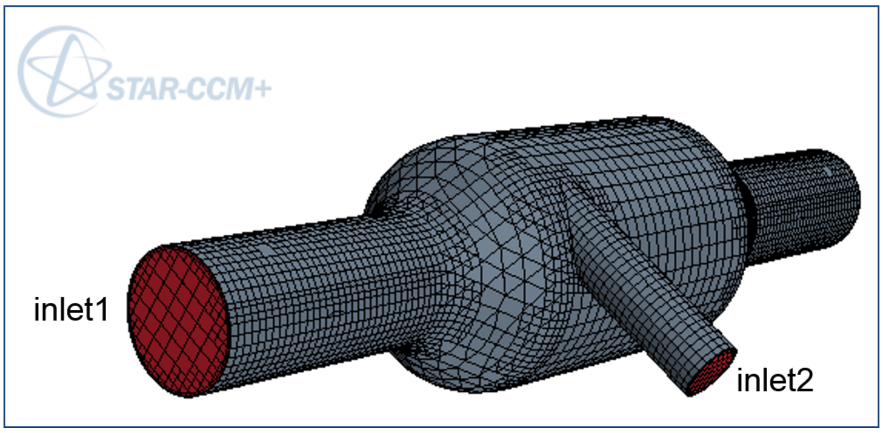


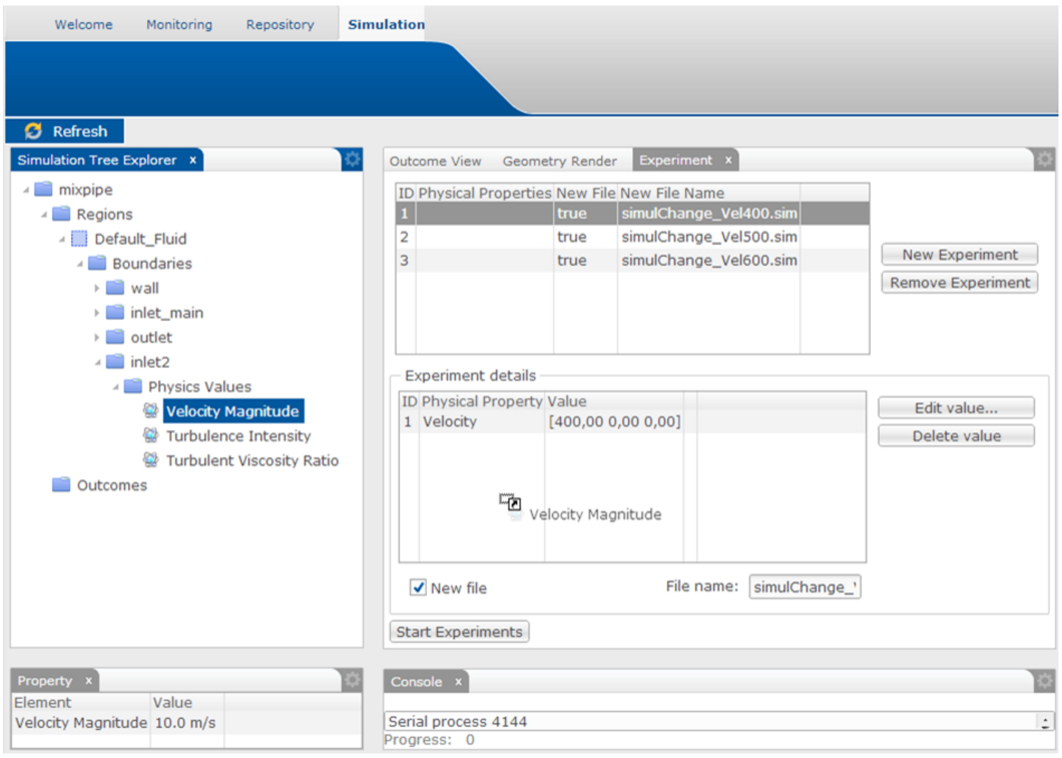
Figure 4.16: An example of Parametric Study.

fluids and one outlet fluid. In order to design this pipe multiple experiments must be performed changing for instance the inlet velocity. The table shown in Figure 4.16 shows four experiments with the relative inlet velocities.

The Parametric Study Tool (Fig. 4.17) 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.

4.5 Conclusions

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 centralise 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



mixpipe
- Regions
  - Default_Fluid
    - Boundaries
      - wall
      - inlet_main
      - outlet
      - inlet2
        - Physics Values
          - Velocity Magnitude
          - Turbulence Intensity
          - Turbulent Viscosity Ratio
- Outcomes

ID	Physical Properties	New File	New File Name
1		true	simulChange_Vel400.sim
2		true	simulChange_Vel500.sim
3		true	simulChange_Vel600.sim

Experiment details

ID	Physical Property	Value
1	Velocity	[400,00 0,00 0,00]

Velocity Magnitude

New file [✓] File name: simulChange_'

Element	Value
Velocity Magnitude	10.0 m/s

```
Serial process 4144
Progress:   0
```

Figure 4.17: Parametric Exploration Tool

the outcomes. And its 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.

Chapter 5

Floasys Platform Architecture

Contents

This chapter describes the Floasys architectural solution [46] 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), trasforms, idexes (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 higest pressure.

This 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 5.2 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.

5.1 Introduction

The idea is to collect both simulation and experimental data, and store them in central repositories as shown in Figure 5.1. 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 5.1 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 sectors), 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 [47] to query the data, a restful service [48] 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 5.1. 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 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 [49]. 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).

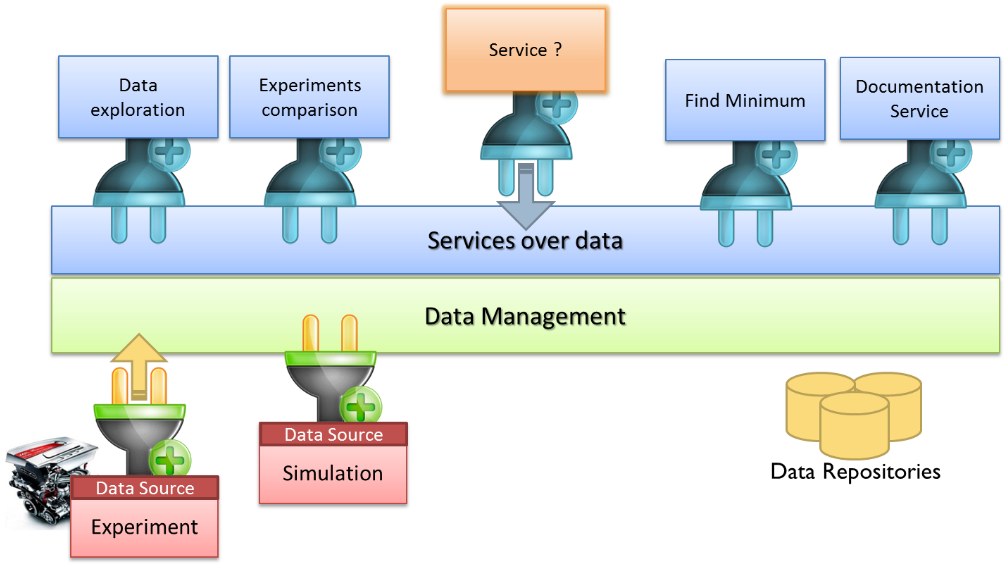


Figure 5.1: Floasys Architectural Solution General Idea

5.2 Floasys Architecture Overview

Floasys is based on a Client/Server architecture (Fig. 5.2) developed using Eclipse Remote Application Platform (RAP) [44]. 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 [50] over the HTTP protocol. Servers tend to interact with user browsers using the JSON exchange format [51] because it is easily parsed in client-side JavaScript language [50]. 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 can not 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 [52–54]. The server can access also to simula-

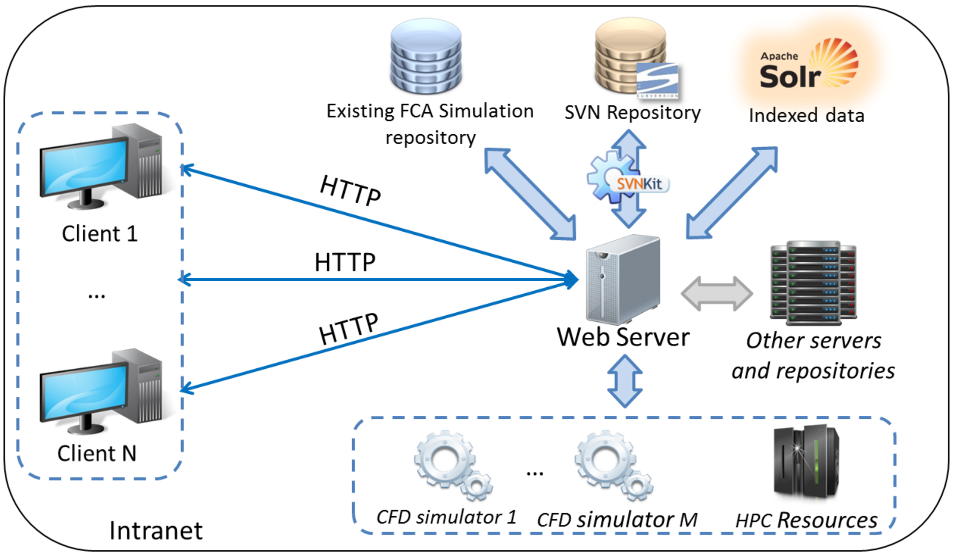


Figure 5.2: Floasys Client/Server Architecture.

tor 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 [55, 56] 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 [57]. 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[1]) 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 in a way to react immediately to the incoming high priority request.

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

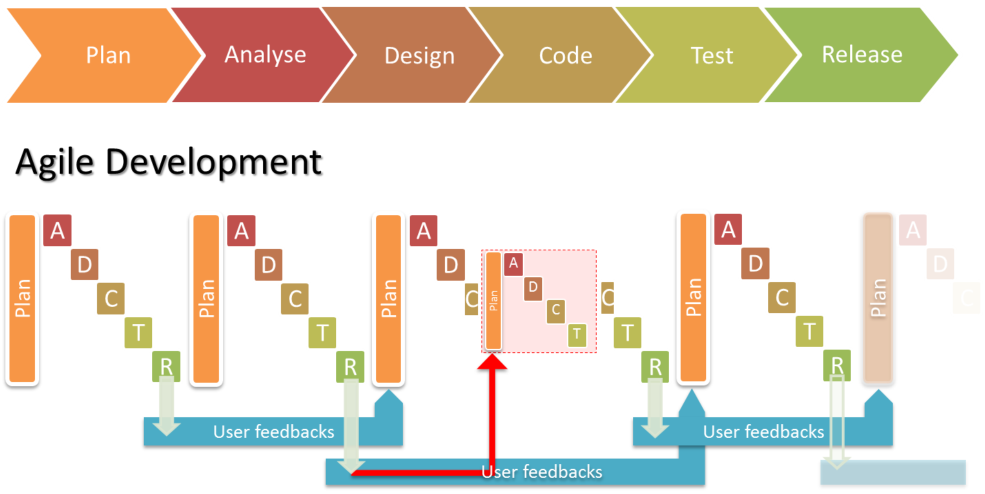


Figure 5.3: Floasys Agile Development and blocking bugs.

5.3 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. 5.7). 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 [2] 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.

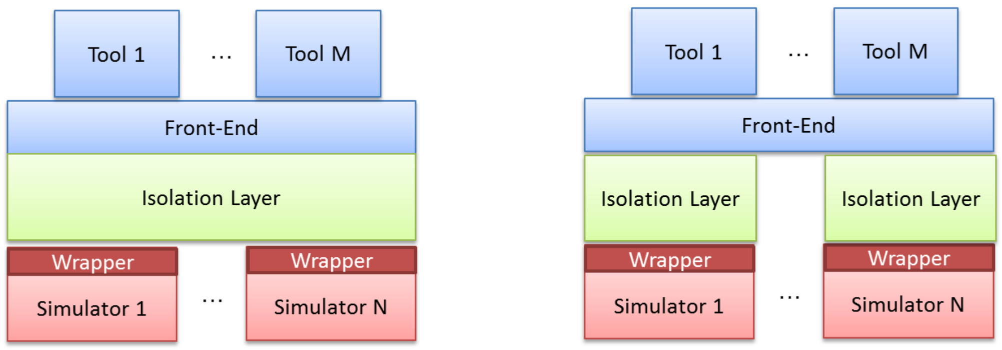


Figure 5.4: Alternative architectural solution comparison.

An alternative solution to the previously described architecture could be the introduction of a separate isolation layer for each vendor software. Figure 5.4 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 [33] 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 5.5 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 5.5: Execution of the Macro to extract data from a simulation file.

Figure 5.6 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. 5.6), the **simulator wrapper** and the **shared folder** (top side of Fig. 5.6). 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.

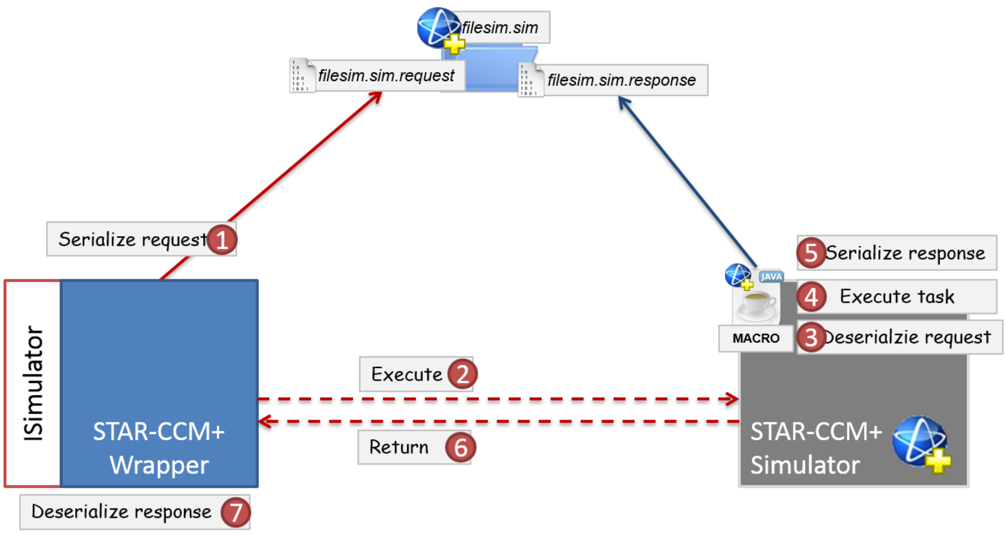


Figure 5.6: How the STAR-CCM+® wrapper extracts simulation data.

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 5.5, 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.

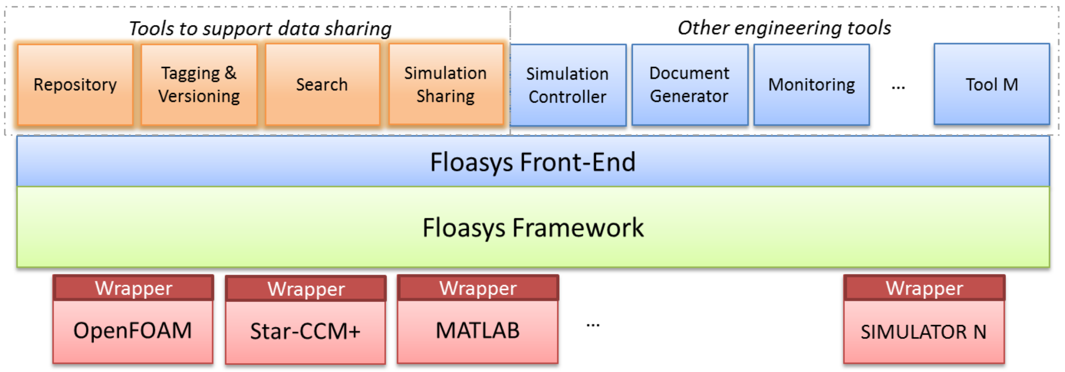


Figure 5.7: Floasys Server-side architecture.

In order to meet extensibility and modularity requirements (Req. 7), the server is based on a pure plug-in architecture [58]. 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 run-time. The Floasys core provides two extensions points to extend its functionalities:

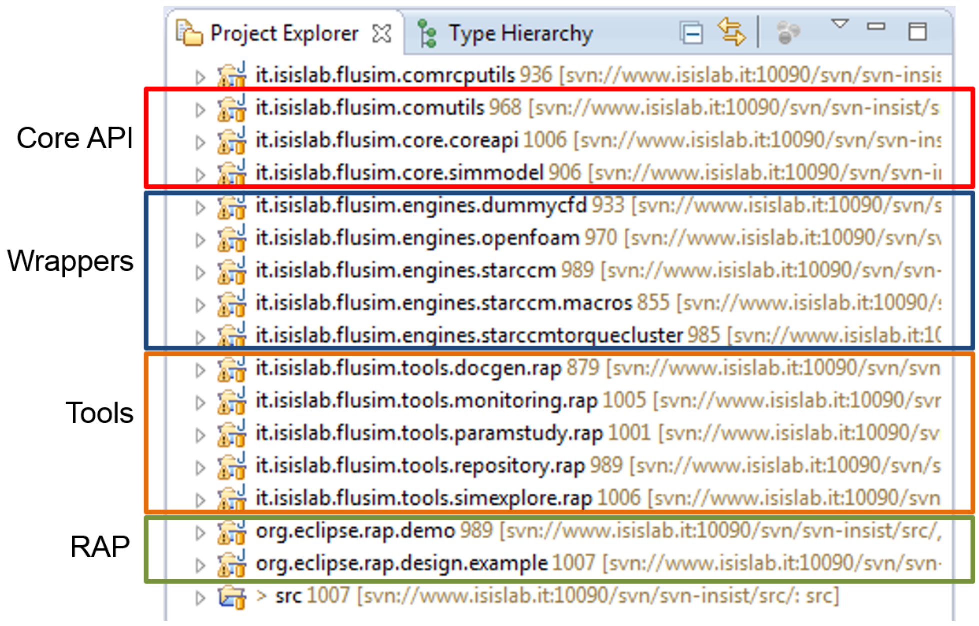
- one hook point to introduce new tools in the upper layer;
- another hook point for new wrappers.

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 [59] (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. 5.7). 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.



```
it.isislab.flusim.comrcputils 936 [svn://www.isislab.it:10090/svn/svn-insis
it.isislab.flusim.comutils 968 [svn://www.isislab.it:10090/svn/svn-insist/s
it.isislab.flusim.core.coreapi 1006 [svn://www.isislab.it:10090/svn/svn-ins
it.isislab.flusim.core.simmodel 906 [svn://www.isislab.it:10090/svn/svn-ii
it.isislab.flusim.engines.dummycfd 933 [svn://www.isislab.it:10090/svn/s
it.isislab.flusim.engines.openfoam 970 [svn://www.isislab.it:10090/svn/sv
it.isislab.flusim.engines.starccm 989 [svn://www.isislab.it:10090/svn/svn-
it.isislab.flusim.engines.starccm.macros 855 [svn://www.isislab.it:10090/:
it.isislab.flusim.engines.starccmtorquecluster 985 [svn://www.isislab.it:1(
it.isislab.flusim.tools.docgen.rap 879 [svn://www.isislab.it:10090/svn/svn
it.isislab.flusim.tools.monitoring.rap 1005 [svn://www.isislab.it:10090/svr
it.isislab.flusim.tools.paramstudy.rap 1001 [svn://www.isislab.it:10090/sv
it.isislab.flusim.tools.repository.rap 989 [svn://www.isislab.it:10090/svn/s
it.isislab.flusim.tools.simexplore.rap 1006 [svn://www.isislab.it:10090/svn
org.eclipse.rap.demo 989 [svn://www.isislab.it:10090/svn/svn-insist/src/,
org.eclipse.rap.design.example 1007 [svn://www.isislab.it:10090/svn/svn-
> src 1007 [svn://www.isislab.it:10090/svn/svn-insist/src/: src]
```

Figure 5.8: Floasys projects within the Eclipse IDE.

Figure 5.7 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 5.8:

- 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*" [44]. 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[2] but any other Linux distribution can be used.

5.4 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 trough command line. The architecture has an isolation layer (Floasys Framework in Fig. 5.7) 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*" [2]. Floasys introduces an intermediate common representation for simulation data called **Simulation Data-Model**. It is based on a tree-like data structure as CGNS [60] 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 [61]. Figure 5.9 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. 5.7) 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 [62] 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

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

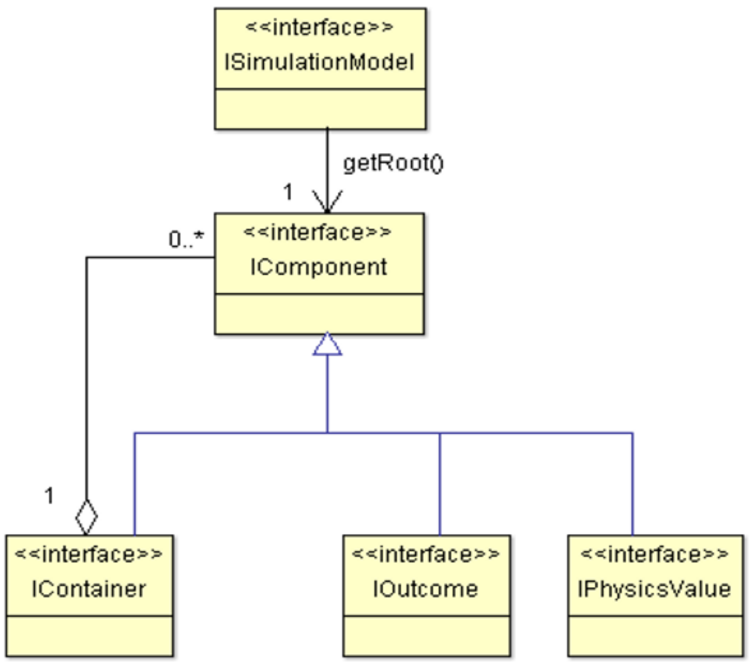


Figure 5.9: Simulation Model Class Diagram

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 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 and for tge 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.

5.5 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 [63] (or shallow checkout) to checkout a portion of the working directory with the freedom to get more files and directories later [63]. 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 [64]. Apache Solr provides extensions, configuration, infrastructure and programming languages bindings around Apache Lucene. In according to the official documentation [64], 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 4.3.3) 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.

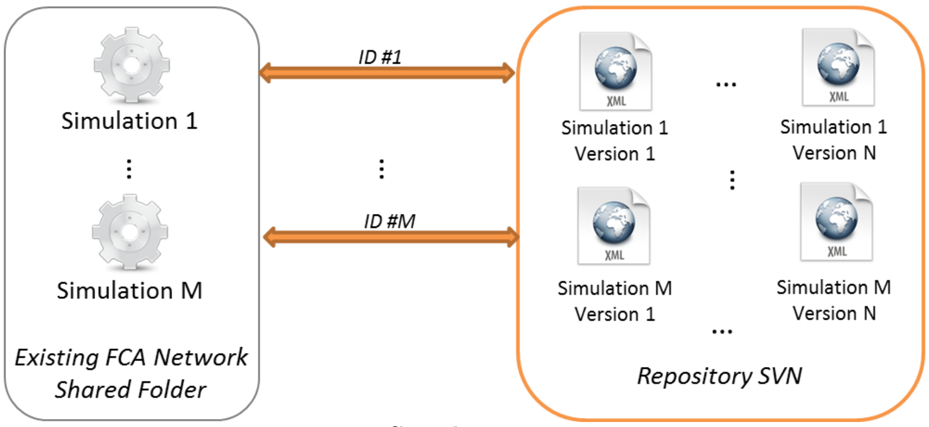


Figure 5.10: Simulation Data versioning.

For each simulation file (left-side of Fig. 5.10) stored in closed file format, an XML file exists in the SVN repository (right-side of Fig. 5.10) that contains extracted simulation data and metadata in open format. In addition, each XML file is indexed using Apache Solr [64]. 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:

- 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 (Fig. 5.2): 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 can not store metadata within the simulation files, also because files are in closed file format.

5.6 Collaborative Requirements Traceability

During the design of a system it is essential to track the requirements through all the design steps. The Figure 5.11 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 5.11 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 world wide 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.

5.7 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.

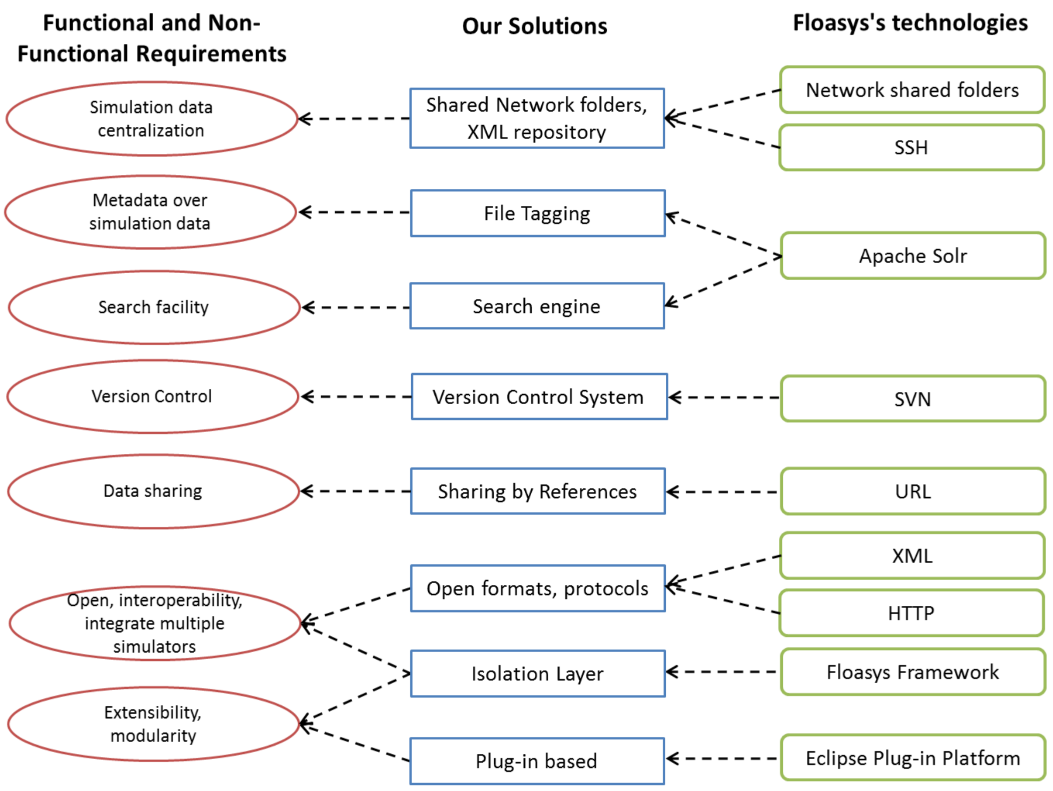


Figure 5.11: Mapping of requirements, solutions and technologies.

5.7.1 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 5.12.

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 (see Section 4.3.1) 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.

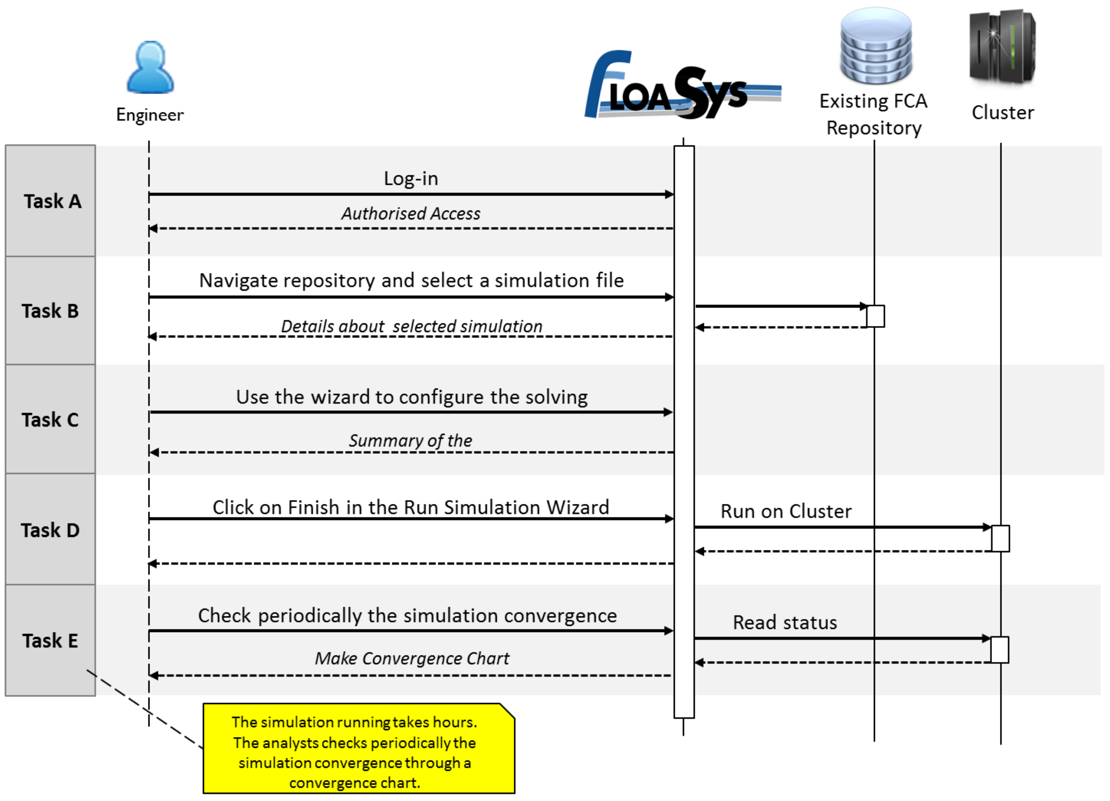

Engineer
FLOASys
Existing FCA Repository
Cluster
Task A
Task B
Task C
Task D
Task E
Log-in
Authorised Access
Navigate repository and select a simulation file
Details about selected simulation
Use the wizard to configure the solving
Summary of the
Click on Finish in the Run Simulation Wizard
Run on Cluster
Check periodically the simulation convergence
Make Convergence Chart
Read status
The simulation running takes hours. The analysts checks periodically the simulation convergence through a convergence chart.

Figure 5.12: Run Simulation Workflow supported by Floasys

In the Task B, Since Floasys manages multiple types of simulator, it recognises which one is able to manage the selected simulation file. For instance, lets 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 recognises 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 [61].

Figure 5.13 shows a piece of source code with in 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 5.15a, 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

```
 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);
```

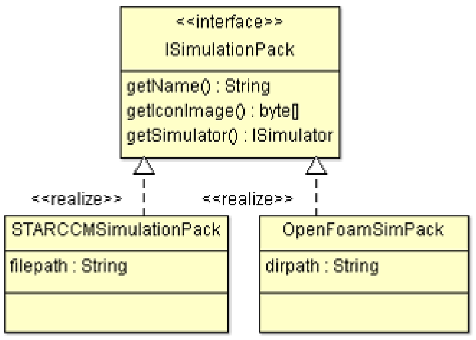
Figure 5.13: A code snippet: how solve a simulation within Floasys.

a directory has a folder path. 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`.

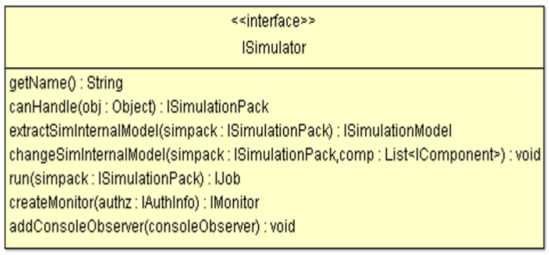
```
File file = ...; //file contains a reference to simulation path.

#foreach ISimulator simulator in simulators
    ISimulationPack simpack = simulator.canHandle(sim);
    #if (simpack != null)
        return simpack;
return null;

//Post-condition: the simulation package instance or null.
```

Figure 5.14: Search a simulator able to handle the simulation file.

Figure 5.14 shows the Floasys framwork pseudocode 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. 5.15b) to understand whether the simulator is able to recognise and handle an object. Therefore, the code in Figure 5.14 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. 5.13) calls this piece of code.



(a) SimulationPack.


<<interface>>
ISimulator
getName() : String
canHandle(obj : Object) : ISimulationPack
extractSimInternalModel(simpack : ISimulationPack) : ISimulationModel
changeSimInternalModel(simpack : ISimulationPack,comp : List<IComponent>) : void
run(simpack : ISimulationPack) : IJob
createMonitor(authz : IAuthInfo) : IMonitor
addConsoleObserver(consoleObserver) : void

(b) Simulator.

Figure 5.15: Floasys Framework: (a) SimulationPack and (b) Simulator.

5.7.2 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. 5.19).

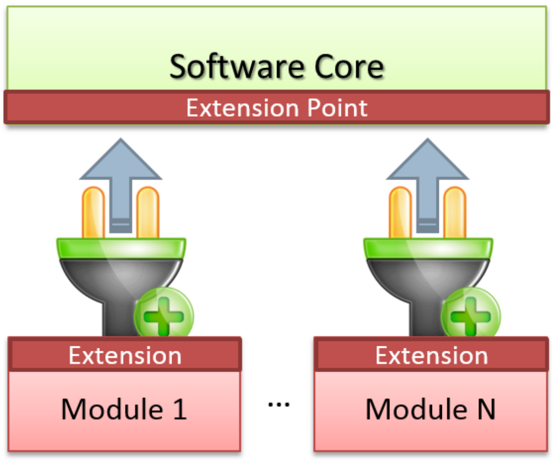
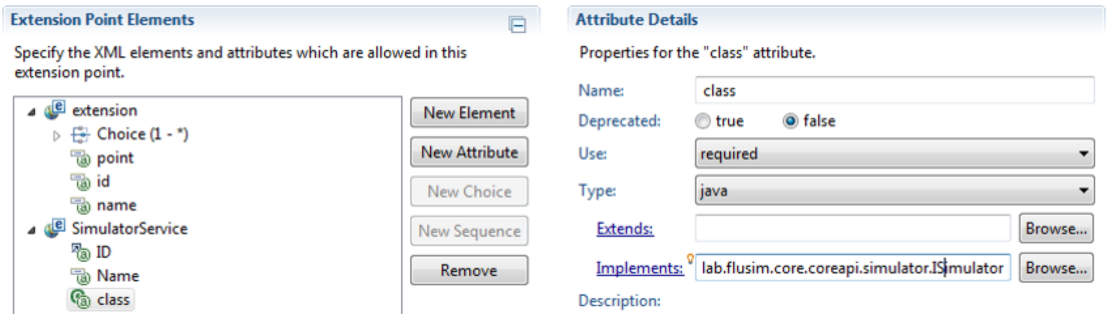


Figure 5.16: Eclipse Extension Point and Extension concepts.

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.


Extension Point Elements

Specify the XML elements and attributes which are allowed in this extension point.

- extension
  - Choice (1 - *)
  - point
  - id
  - name
- SimulatorService
  - ID
  - Name
  - class

Attribute Details

Properties for the "class" attribute.

Name:	class
Deprecated:	( ) true [✓ false]
Use:	required
Type:	java
Extends:	
Implements:	lab.flusim.core.coreapi.simulator.ISimulator
Description:	

Figure 5.17: How to define an extension point within Eclipse.

Figure 5.17 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. 5.15). All data have shown on the left side of Figure 5.17.

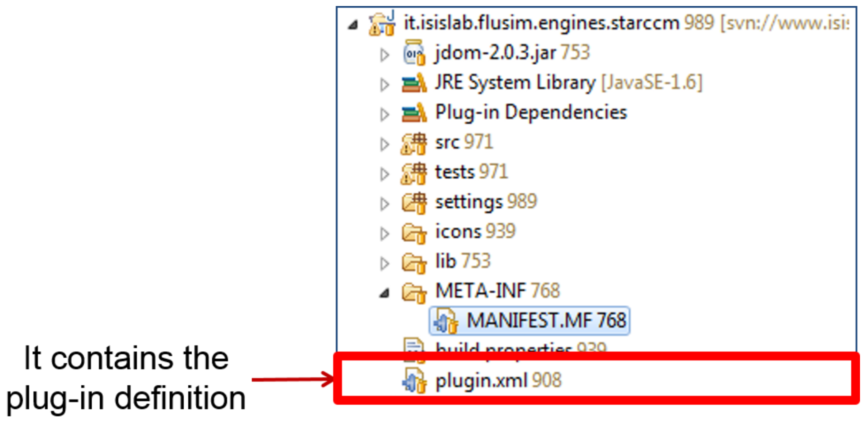
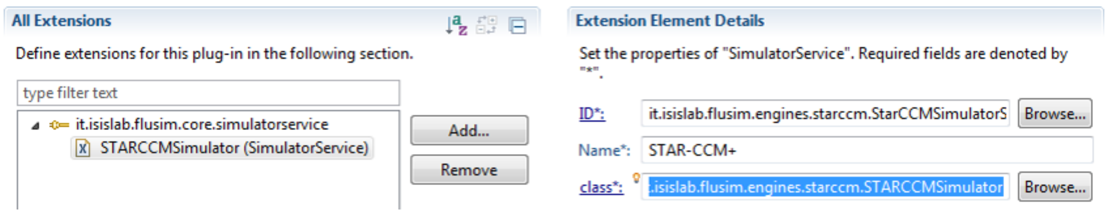

- it.isislab.flusim.engines.starccm 989 [svn://www.isi
  - jdom-2.0.3.jar 753
  - JRE System Library [JavaSE-1.6]
  - Plug-in Dependencies
  - src 971
  - tests 971
  - settings 989
  - icons 939
  - lib 753
  - META-INF 768
    - MANIFEST.MF 768
  - build.properties 939
  - plugin.xml 908

Figure 5.18: Floasys Simulator Wrapper Plug-In Eclipse Project.

Figure 5.18 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 order the following: 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.


All Extensions

Define extensions for this plug-in in the following section.

type filter text

it.isislab.flusim.core.simulatorservice
STARCCMSimulator (SimulatorService)

Add...
Remove

Extension Element Details

Set the properties of "SimulatorService". Required fields are denoted by "*".

ID*:	it.isislab.flusim.engines.starccm.StarCCMSimulatorS	Browse...
Name*:	STAR-CCM+	
class*:	.isislab.flusim.engines.starccm.STARCCMSimulator	Browse...

Figure 5.19: Plug-in Simulator Wrapper Extension Definition.

5.8 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 (Fig. X) 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. As described in Section 5.7.2, Floasys strongly relies on this concept so that it has a pure plug-in architecture.

The RAP architecture overview is shown in Figure 5.20. 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

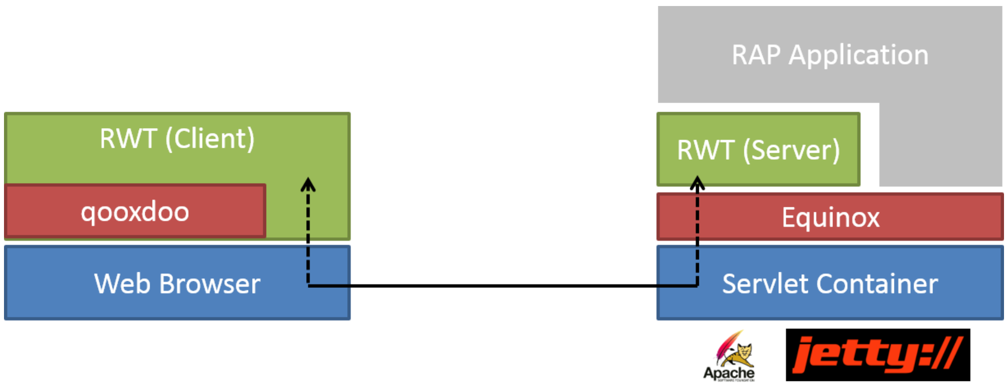


Figure 5.20: Eclipse RAP Client-Server Architecture.

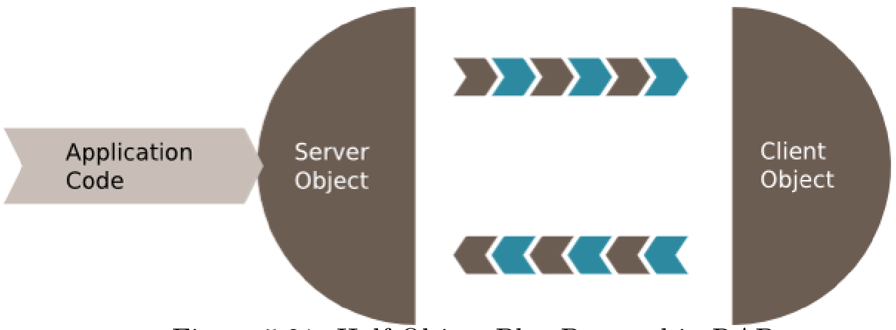


Figure 5.21: Half Object Plus Protocol in RAP.

5.21 shows a conceptual view of the Half Object Plus Protocol (HOPP), the circle that represent a 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.

5.9 Conclusions

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.

Chapter 6

Floasys Simulation Repository Exploration

Contents

This 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.

6.1 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 [65], industries perform computer numerical simulations to design their product facing time-to-market, high quality and cost down pressures [65]. 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 [1], 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 [1]. 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 [16].

Based on my experience working closely a team of aerothermal CFD within Fiat Chrysler Automobiles, I identified the following three main requirements: (1) data collection, centralisation [65], and sharing [1] (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.

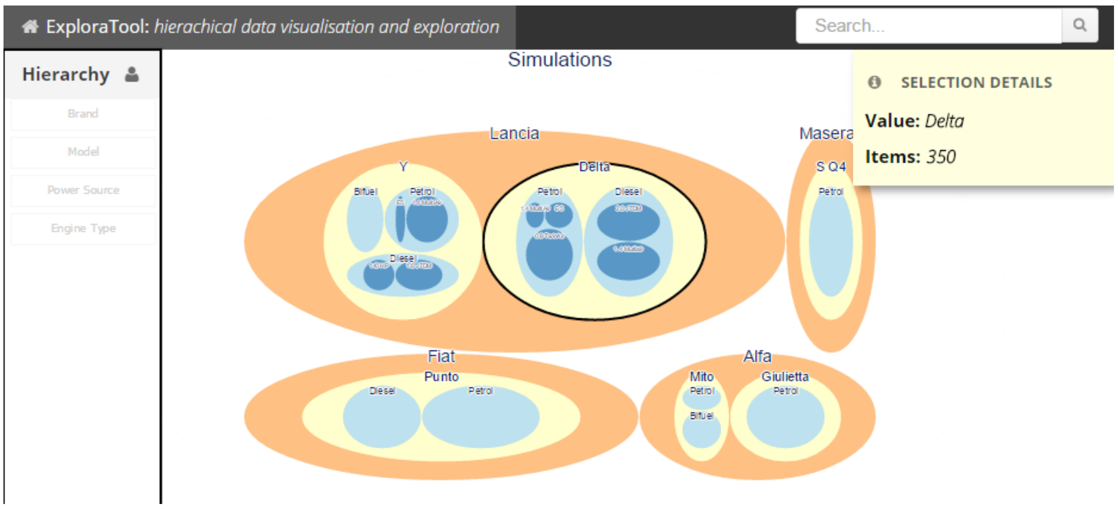


Figure 6.1: The ExploraTool's Graphical User Interface. It shows an overview of the simulation repository through an initial hierarchy made by the following simulations' attributes: brand, project model, power source and engine type.

6.2 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 [66] 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 visualise different hierarchical data, such as inherently hierarchical organisation structures [67],

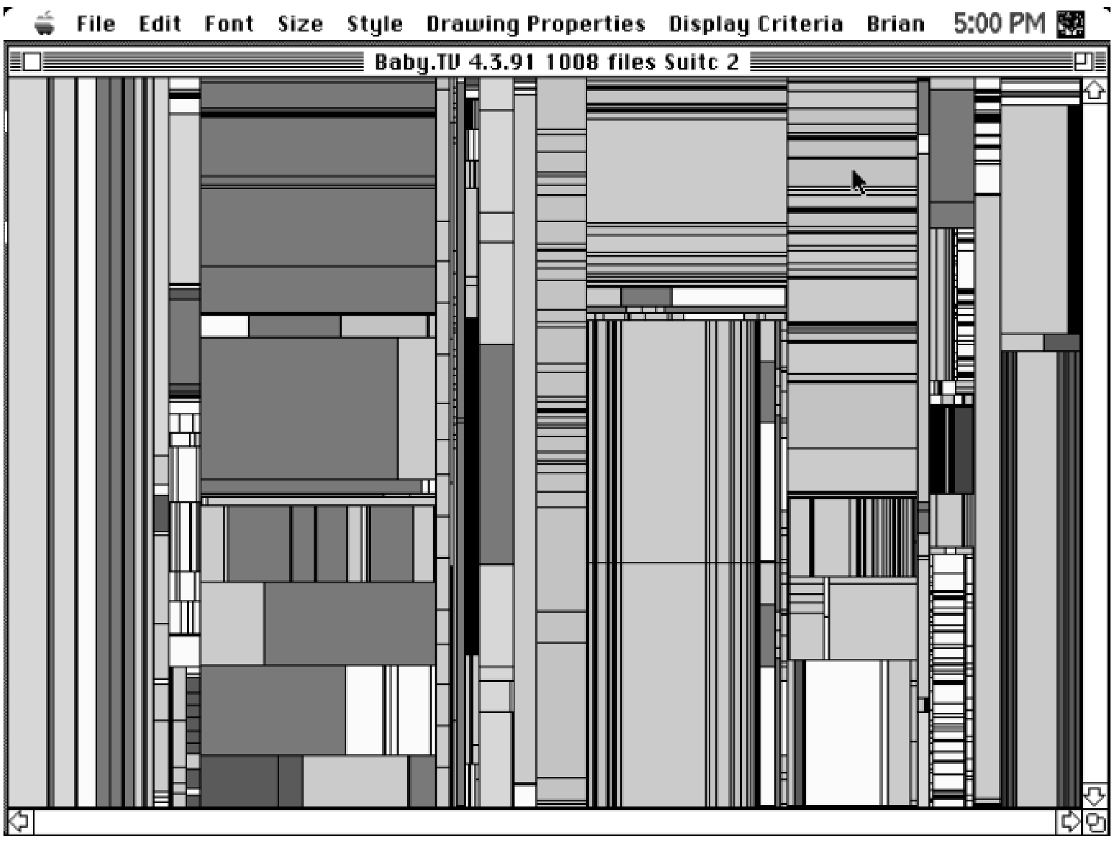

File Edit Font Size Style Drawing Properties Display Criteria Brian 5:00 PM
Baby.TU 4.3.91 1008 files Suitc 2

Figure 6.2: Original treemap visualisation introduced by Shneiderman during 1990.

file systems [68], Usenet newsgroup [69] and so on. Well-known treemap drawbacks are the hierarchy discernment [70] and the fact that *the position of the mouse pointer designates an entire branch of the tree* [71] because *each point belongs to a single leaf node but also to all its ancestors* [71]. Of course, one of their advantages is the use of the all available 2D space.

Ellimap [70] 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. [72], the use of ellipses with their extra space improves the hierarchy discernment compared to the visualisation based on rectangles.

ExploraTool 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 [70]. 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.

6.3 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. 6.1) has a central view to show graphically the simulations available within the repository. The tool shows data using the ellimap [70] visualisation technique, a 2D space-filling approach that uses ellipses as basic shapes to represent sets of simulations. As shown in Figure 6.1, 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 [1].

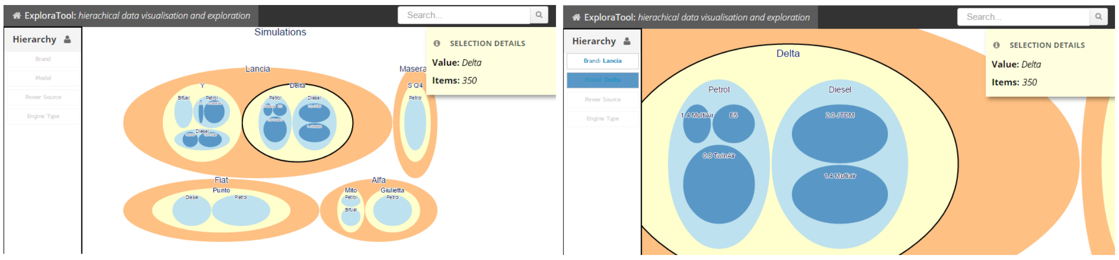
The user can have additional details on each group of simulations (ellipse) just by hovering the mouse cursor over it. The tool shows the additional information, such as, the number of items in a yellow box on the top-right (see Figure 6.1). This space can be used in the future to provide aggregated 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 [73], 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.

6.3.1 Data Exploration: in-depth navigation

The user can further explore the simulation repository through the *in-depth* navigation [71] 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. When



SELECTION DETAILS

Value: *Delta*

Items: *350*

Figure 6.3: 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.

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 [74] 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 6.3) 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* [71] because *each point belongs to a single leaf node but also to all its ancestors* [71]. Finally, in the ExploraTool, also to obtain the list of simulations within a specific ellipse the user can click directly on the target ellipse.

6.3.2 How should I use the colours?

The use of the colours is really important within a visualisation tool. ExploraTool uses the colours described in the following work [75]. Table 6.1 lists the colours used within the ExploraTool and visible through the Figure 6.1 and Figure 6.3.

Colour	HTML Colour
🟥	E04A4D
🟧	FEC083
🟨	FFFFCD
🟦	BDE1EE
🟦	5997C6

Table 6.1: Colours used to draw the ellipses within the ExploraTool.

6.4 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. 6.4). 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. 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.

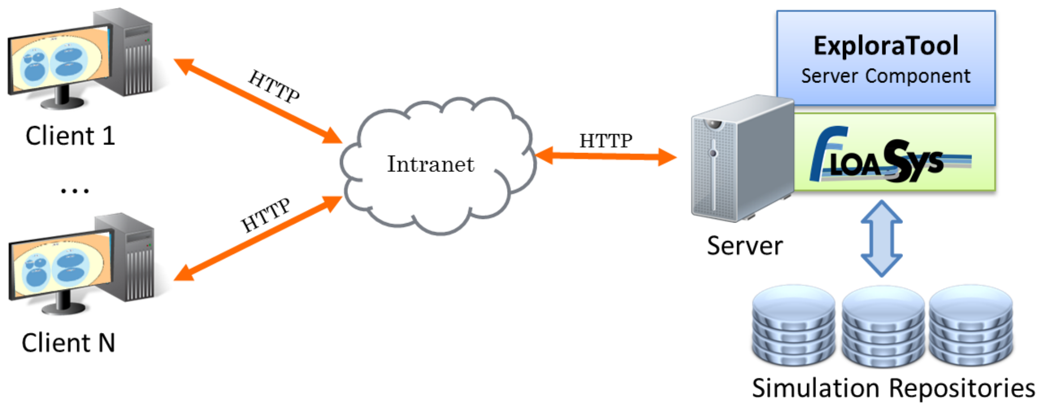


Figure 6.4: ExploraTool prototype client/server architecture.

On the server-side there are one or multiple simulation repositories. The Floasys Framework [16] 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 6.5. The simulation data in tabular format are the input for the *Hierarchy generation* phase performed by the algorithm described in the next Section 6.6. 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.

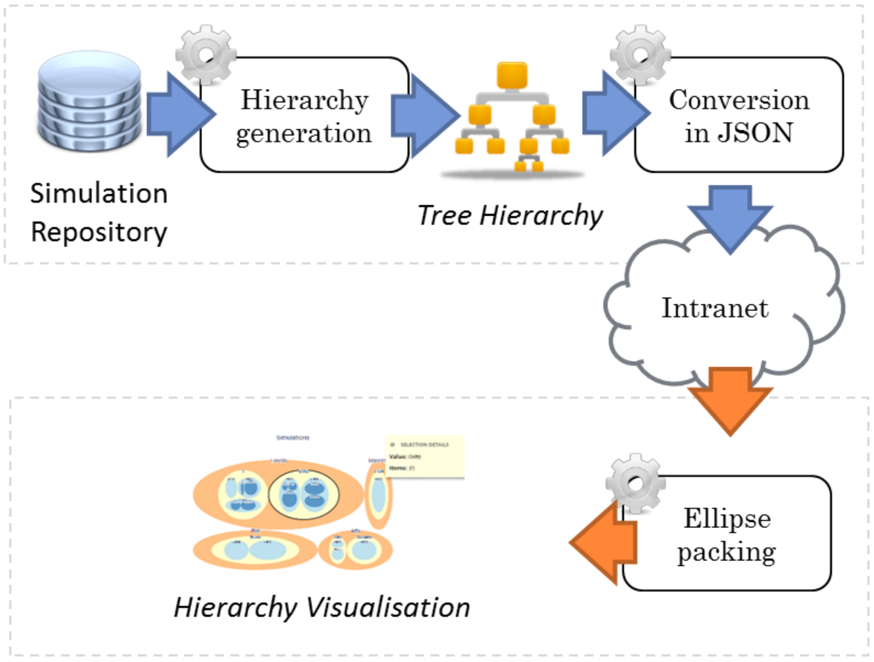


Figure 6.5: Pipeline of transformation from the simulation repository to the visualisation on the client Web-browser.

From technological point of view, ExploraTool leverage from mainstream technologies. Clients exchange data with the server in JSON text format [50, 51] using standard Web protocols (e.g., HTTP). Clients are implemented using the open source JavaScript library *D3 Data-Driven Documents*[1] [76] and SVG. The server has been implemented using Java.

6.5 Simulation Repository Model

A repository R is a collection of n items $R = \{s_1, ..., s_n\}$. Through this chapter obviously the items are simulations (Fig. 6.6). Each item $s \in R$ is described through attributes attached to it. Therefore, this work introduces a set $A = \{a_1, a_2, ..., a_m\}$ that contains the labels/names of the attributes useful to describe the items. Therefore, each attribute a_i is a label. For example, in the use case, the labels for the attributes are $A = \{brand, projectmodel, powersource, enginetype\}$. These attributes are also known as facets [77, 78].

For each attribute a_i exists a set of valid values that we call domain $D_i = dom(a_i)$. For instance, in the simulation context, the attribute *power source* has the domain

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

$D_{powersource} = dom(powersource) = \{Bifuel, Petrol, Diesel\}$. In Fiat Chrysler Automobiles, the attribute *brand* has the following valid values $D_{brand} = dom(brand) = \{Fiat, Lancia, Alfa, Chrysler, Maserati\}$. The domain collection $D = \{D_1, D_2, ..., D_m\}$ contains a domain set D_i for each attribute a_i, so that $\forall i \in \{1, ..., m\}$, $D_i = dom(a_i)$.

Each simulation $s \in R$ is an ordered list of m values $s =< v_1, ..., v_m >$, where $\forall i \in \{1, ..., m\}$, $v_i \in dom(a_i)$. According to [79], in order to easily access the values of the simulations attribute, for each simulation s in the repository R, its useful to introduce the notation $s[i] \in D_i$ to obtain the value of the i-th attribute of the ordered list s. Therefore, s is a list of values and they are drawn one from each domain set, so that the i-th value $s[i]$ is from the domain D_i. For simplicity, the assumption here is that all the values are single-valued [77] as opposed to multi-valued (e.g., colour attribute for a flag with multiple values, such as red and green).

Therefore, the formal model for a repository of simulations (Fig. 6.6) is a triple (A, D, R).

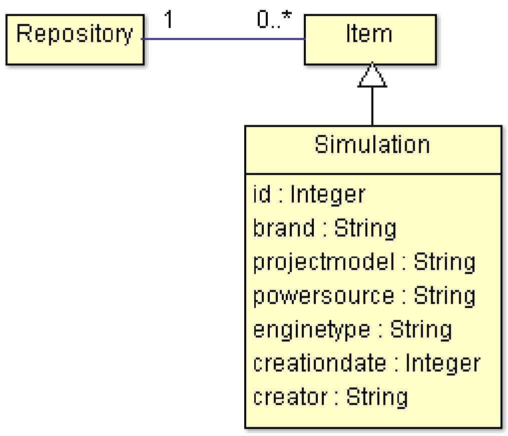


Figure 6.6: Class Diagram to describe a typical repository of simulations.

6.6 Hierarchical representation for tabular data

The ExploraTool presents a hierarchical view of the large dataset, and permits navigation through drill-down and roll-up operations. This is fully compliant to the Shneiderman guideline of "*Overview first, zoom and filter, then details-on-demand*" [80].

As depicted in Figure 6.7, the ExploraTool allows the graphic navigation of the dataset through a hierarchy (tree) built over the repository content. Every time the user decides to drill down, he/she clicks on an ellipse that identifies a branch in the tree. As shown in Figure 6.7, the tree and the visualisation have a level for each attribute a_i in the repository. At each level of the tree, that corresponds to the attribute a_i, there are the nodes with the values is in D_i. Some nodes corresponds to zero items in the original dataset, 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 the constructed tree because there are not items in the original dataset with the values $s[1] = 1 \wedge s[2] = B \wedge s[3] = b$.

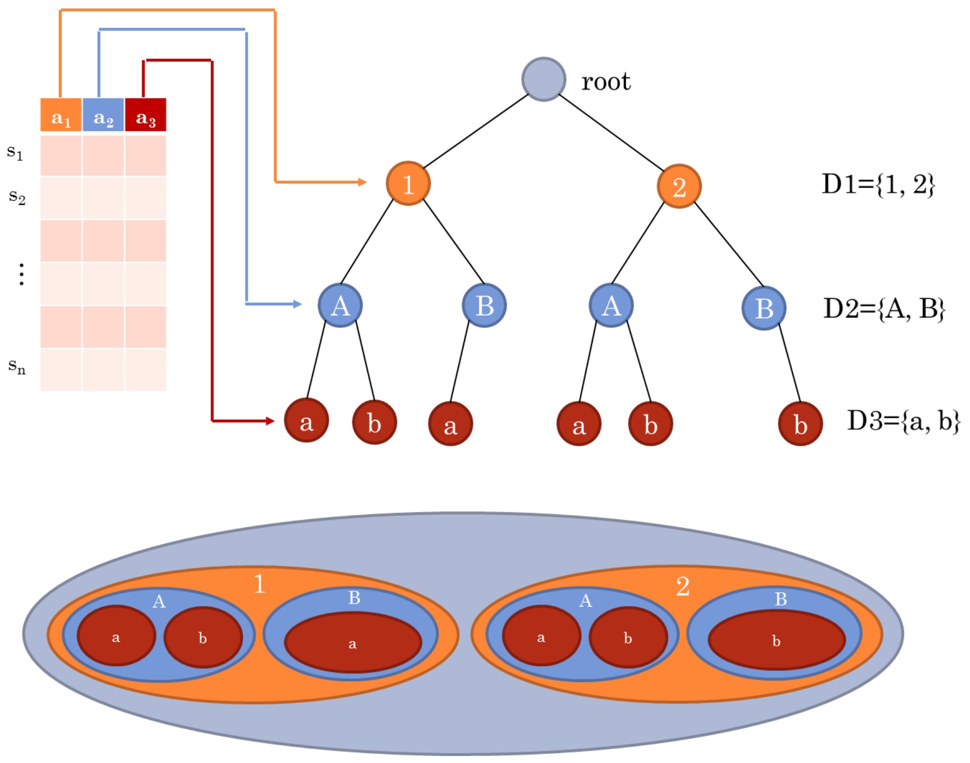


Figure 6.7: Example of hierarchy extraction from a table. The table on the left has tree attributes $A = \{a_1, a_2, a_3\}$ and n simulations. The tree on the right has a level for each domain D_i and at each level i there are the nodes with the labels in D_i. ExploraTool builds the visualisation shown on the bottom of the figure, starting from the tree data structure.

It is clear and evident that the visualisation through the ellipses is equivalent to the tree representation, and actually ExploraTool builds the visualisation from the tree. Therefore, the tool strictly depends on the creation of a hierarchy from the repository of simulations. This section describes a generic algorithm called `BuildHierarchy` implemented in the ExploraTool server-side component, to get a hierarchical view from any kind of dataset, not only a repository of simulations.

The algorithm `BuildHierarchy` runs in two 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 users perform a drill-down operation, they are 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.

6.6.1 BuildHierarchy Algorithm Overview

Generally speaking, the algorithm reads the data from an input dataset (e.g., simulation repository) described as (A, D, R), and builds a tree data structure $T = (V, E)$ that will be rendered using nested ellipses (Fig. 6.7). In order to limit the amount of data to show within the ExploraTool, we introduce the number levels to display $r \in \{1, 2, ..., m\}$ which is exactly the height of tree data structure to build (the longest downward path between the root and a leaf [81]). In this way, the process depicted in Figure 6.5 will retrieve only a part of the dataset.

In the overall process depicted in Figure 6.5 only the part of dataset corresponding to the r attributes is retrieved from the simulation repository, and the ExploraTool shows only the first r levels of the hierarchy to the user. The loading of further data is triggered when the user navigates the hierarchy with a drill-down operation. In this way, the server provides chunks of subtrees made by r levels.

The **algorithm input** is a 5-tuple (A, D, R, r, f_S).

The f_S function specifies a sorting of the domains that will have direct impact upon the resulting hierarchy. Given m domains, $m!$ permutations exist, so $m!$ different tree hierarchies exist. Therefore, the algorithm needs an initial permutation of the domains and this can be decided at tool configuration time. Here, a domain permutation is just a function $f_S : \{1, 2, \cdots, m\} \rightarrow \{1, 2, \cdots, m\}$ that maps the given sort with the domains. The function f_S defines the initial order for the attributes as shown in the navigation bar on the left side of the ExploraTool GUI (Fig. 6.1).

The **algorithm output** is a tree $T = (V, E)$ that will be displayed as nested ellipses starting from the root. In according to [81], $\forall u \in V$ we use the following notations: $Children[u]$ is the list of all the nodes v such that exists an edge $(u, v) \in E$; $Count[u] \in \mathbb{N}$ is the counter to track the number of simulations represented by this node u that will determine the area of the relative ellipse; $Label[u]$ is the label for the node u that will be displayed on top of the ellipse.

6.6.2 BuildHierarchy Algorithm Description

Initially the tree T has only one node that is the *root* and no edges. The algorithm 1 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). Obviously the number of attributes to scan can not be greater than $min(r, m)$, where r is the number of levels to display and m is the total number of available attributes.

The Algorithm 1 takes each simulation and navigates the tree from the root to a leaf node and goes one level down for each domain. Initially, the root is the selected node (Alg. 1 lines 3 and 18). For each attribute's value $s[i]$, the algorithm at line 9 tries to find a node with the label $s[i]$ if it already is present but also it sets the current node `curnode` to `null` if there is no such node already present. In this way, if there is not a child in the tree with value $s[i]$, at line 10, the algorithm creates a new node with that attribute value.

Table 6.2: Notations used to describe the BuildHierarchy Algorithm.

Notation	**Description**
R	Repository of simulations.
n	Number of simulations, $\|R\| = n$.
s	Simulation $s \in R$.
A	Simulation attributes as a set of labels/names.
m	Number of attributes $\|A\| = m = \|D\|$.
a_i	i-th attribute of A, $a_i \in A$.
D_i	Set of valid values for the attribute a_i.
$s[i]$	Value of i-th attribute of simulation s.
V	Set of nodes for the tree data structure.
E	Set of directed edges for the tree data structure.
v, u	Two nodes $u, v \in V$.
$Label[u]$	Label attached to the node u.
$Count[u]$	A positive integer number to track the number of simulations.
$Children[u]$	A list of all nodes v that $(u, v) \in E$.

In the tree data structure each node $\forall u \in V$ has a size specified by the $Count[u]$ that is the number of items in the dataset with the specific attribute values. All the nodes have size greater than zero, so that during the exploration of the hierarchy, the user will never encounter an empty set.

The `BuildHierarchy` algorithm can be used to provide the **rearrangement feature**. 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 levels in the hierarchy or enable/disable some other levels. In this way, the user is sorting the n domains by choosing a permutation. By changing the input function f_S to Algorithm 1 one may obtain the different hierarchical views corresponding to the different sortings of the domains.

The algorithm to build the hierarchy can be executed by using the map/reduce paradigm on large datasets by slicing the dataset in q groups of rows. Each group is mapped to a computing node. Each computing node can execute the algorithm on a specific set of rows and compute the partial tree data structure. Finally, the resulting tree data structure can be merged together to get the final hierarchy. The main problem to face is to use a real time paradigm especially because the web-based ExploraTool is interactive and requires high response time.

6.6.3 Analysis of BuildHierarchy algorithm

The algorithm input is a 5-tuple (A, D, R, r, f_S). The algorithm output is a dataset hierarchical view, a tree data structure (V, E). We calculate the number of the nodes $|V|$ in the output tree and compute the wrost case running time of the `BuildHierarchy`. We adopt the asymptotic notation $f(n) = \Theta(g(n))$ to indicate that "$g(n)$ *is an asymp-*

Algorithm 1: BuildHierarchy(A, D, R, r, f_S)

```
Input: labels $A$ for the attributes, domains $D$, set of simulations $R$, number of
       levels $r$ to load and a domains ordering function $f_S$
Output: $T = (V, E)$
/* Initialisation                                                          */
1  $root \leftarrow$ CreateNewNode();
2  $V \leftarrow \{root\}, E \leftarrow \emptyset$;
3  $prevnode \leftarrow root$;
4  $curnode \leftarrow null$;
/* It iterates along all simulations in the repository $R$                */
5  foreach $s \in R$ do
6      for $j \leftarrow 1$ to $min(r, m)$ do
7          $i \leftarrow f_S(j)$ ;                   /* $i \in \{1, \cdots, m\}$ is an index.  */
8          $value \leftarrow s[i]$ is the value of i-th attribute of $s$;
9          $curnode \leftarrow$ FindChild($V, E, prevnode, value$);
10         if $curnode$ is null then
11             $curnode \leftarrow$ CreateNewNode();
12             $Count[curnode] \leftarrow 0$;
13             $Label[curnode] \leftarrow value$;
14             $V = V \cup curnode$;
15             $E = E \cup \{(prevnode, curnode)\}$;
16         $Count[curnode] + +$;
17         $prevnode \leftarrow curnode$;
18     $prevnode \leftarrow root$;
19 return $(V, E)$;
```

Algorithm 2: FindChild(V, E, u, nodelabel)

```
Input: tree data structure $(V, E)$, a node $u \in V$ and a $nodelabel$
Output: a node $v \in V$ with $(u, v) \in E$, or $null$
1  foreach $v \in Children[u]$ do
2      if $Label[v] == nodelabel$ then
3          return $u$
4  return $null$
```

totically tight bound for $f(n)$" [81] and $f(n) = O(g(n))$ to indicate that "*$g(n)$ is an asymptotic upper bound for $f(n)$*" [81].

For each domain $D_i \in \{D_1, \cdots, D_m\}$, D_i is a set of values. We denote with $|D_i|$ the number of items in the set D_i. Therefore, in according to Equation 6.1, p is the maximum number of items in D_i where $i \in \{1, \cdots, m\}$. Equation 6.1 is an upper bound for the domains' sizes.

$$p = \max_{i \in \{1, \cdots, m\}} |D_i| \tag{6.1}$$

Hierarchy analysis

The aim is to calculate $|V|$ (the size of the set V) which is the number of nodes in the generated tree. The algorithm `BuildHierarchy` generates in the worst case a full, complete and balanced tree with height r as requested by input to the algorithm. Therefore, the total number of nodes is $O(p^r)$ with $p > 1$ and $1 \leq r \leq m$. The worst case occurs when all domains have exactly size p.

The algorithm 1 generates a tree with a level for each domain D_i and the maximum number of levels is r.

At level 1, the algorithm inserts a new node for each value in the domain $D_{f_S(1)}$ that is maximum p. At level 2, the algorithm inserts a new node for each value in the domain $D_{f_S(2)}$, and it does this for each node at the previous level (i.e. level 1). Therefore, the number of total nodes in the tree is just the product of domains' sizes (Eq. 6.2). Equation 6.1 provides an upper bound on the domains' sizes that can be used in the following equation to obtain an upper bound for $\prod_{i=1}^{m} |D_i|$.

$$|V| \leq |D_1| \times |D_2| \times \cdots \times |D_r| = \prod_{i=1}^{m} |D_i| \leq \prod_{i=1}^{r} p = O(p^r) \tag{6.2}$$

The final tree data structure (the dataset hierarchical view) in the worst case, has a number of nodes that grows exponentially with the parameter r (the number of levels). Therefore, the number of nodes in the tree $|V|$ is $O(p^r)$ (Eq. 6.1 and Eq. 6.2). In order to keep the hierarchy clear during the visualisation we decide to not show more than five levels at each time ($r \leq 5$), so the number of the nodes in the tree will in the worst case $O(p^5)$.

Algorithm running time analysis

The `BuildHierarchy` algorithm scans all the dataset items (the table rows) at line 5. Assuming that the dataset has n rows ($|R| = n$), the `foreach` iterative statement at line 5 is $\Theta(n)$. For each row in the dataset, algorithm 1 scans exactly r columns (the domains) of the dataset at line 6 in the order denoted by the sorting function f_S. Therefore, the running time upper bound for the line 6 is $O(r)$. Finally, there is a call to the algorithm `FindChild` (Alg. 2) whose running time is proportional to the maximum number of children for each parent node in the tree which is $O(p)$ (Eq. 6.1). Here, we are assuming that each parent node in the tree has a linear list of children. Therefore, the algorithm `BuildHierarchy` **running time is**: $\Theta(n) \cdot O(r) \cdot O(p) \leq O(n \cdot r \cdot p)$.

The algorithm has the r parameter to decide the number of levels to show in the hierarchy. The same parameter is useful to render the hierarchy on-demand. In this way, two interesting features can be provided: (1) when the user drills-down in the hierarchy, it is asking for details on-demand and so the algorithm loads the next r levels; (2) from computational point of view, the algorithm computes only the first r levels and so the running time becomes $O(n \cdot p)$.

6.7 Conclusions and Future Works

This 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 [82,83]. 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 [78], 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 7

Conclusions & Future Works

Contents

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

7.1 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.

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 analysing 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 are depicted in the Figure 5.11.

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 [82, 83]. The user acceptance of the software will be investigated as well [84].

7.2 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 every thing 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 [85]. The research aim could to understand and evaluate the benefits of using the social in the field of industrial CFD simulations.

Bibliography

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

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

[3] Claudio Gargiulo, Donato Pirozzi, and Vittorio Scarano. An architecture for CFD Workflow management. In *INDIN*, pages 352–357, 2013.

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

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

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

[7] 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.

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

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

[10] Marina Mendonça 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.

[11] 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.

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

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

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

[15] 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.

[16] 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.

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

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

[19] 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.

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

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

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

[23] 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.

[24] 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.

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

[26] 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.

[27] 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.

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

[29] 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.

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

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

[32] 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.

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

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

[35] 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.

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

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

[38] 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.

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

[40] 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.

[41] 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.

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

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

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

[45] 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.

[46] 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.

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

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

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

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

[51] Guanhua 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*, pages 182–185. IEEE, 2011.

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

[53] Rafa\l Kuć. *Apache Solr 4 Cookbook.* Packt Publishing Ltd, 2013.

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

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

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

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

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

[59] Eclipse Public License.

[60] 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.

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

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

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

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

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

[66] 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.

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

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

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

[70] Benoît Otjacques, Maël Cornil, and Fernand Feltz. Visualizing cooperative activities with ellimaps: the case of Wikipedia. In *Cooperative Design, Visualization, and Engineering*, pages 44–51. Springer, 2009.

[71] 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.

[72] Benoît Otjacques, Maël 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.

[73] 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.

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

[75] 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.

[76] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. D\$ˆ3\$ data-driven documents. *Visualization and Computer Graphics, IEEE Transactions on*, 17(12):2301–2309, 2011.

[77] 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.

[78] 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.

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

[80] 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.

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

[82] Computer System Usability Questionnaire, December 2014.

[83] Questionnaire for User Interface Satisfaction.

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

[85] 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 Conference on*, pages 305–314. IEEE, 2013.

Appendices

Appendix A

Requirements elicitation Survey

This Appendix reports the survey used to collect the key collaborative requirements.

Collaborate around simulation data

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.

1. Nickname: ______________________________ *(choose a nickname)*

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. I am mainly

- Academic
- Industrial

Q4. How many years you spent working in the CFD field?

- ________________ *(write the number of years)*

Q5. How many simulations do you perform per year?

- ________________ *(write the number of simulations per year)*

Q6. Mainly, I perform the following simulation types

- Aerodynamics
- Aeroacustic
- Underhood cooling
- Cabin climatisation
- Other

Q7. Every year I perform a number of simulations equal to:

- $<$ 50 per year
- between 50 and 100 per year
- between 100 and 150 per year
- between 150 and 200 per year
- between 200 and 250 per year
- between 250 and 300 per year
- between 300 and 350 per year
- between 350 and 400 per year
- $>$ 400 per year

Collaboration among analysts and data sharing

Q8. In my office I daily work with a number of analysts equal to

- ________________ *(write the number of analysts)*

Q9. I daily work with a number of analysts in a different place equal to

- ________________ *(write the number of analysts)*

Q10. On average, the geometries file size in average is

- ________________ *(write the geometry file size)*

Q11. On average, the simulations file size in average is

- ________________ *(write the simulation file size)*

Q12. In order to perform assigned tasks, I usually exchange files with other engineers who are located in **my same office**

CAD files	(Never)	1	2	3	4	5	6	7	(Always)
Simulations file	(Never)	1	2	3	4	5	6	7	(Always)

Q13. In order to perform assigned tasks, I usually exchange files with other engineers who are located in **another location**

CAD files	(Never)	1	2	3	4	5	6	7	(Always)
Simulations file	(Never)	1	2	3	4	5	6	7	(Always)

Q14. 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)

Q15. 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)

Q16. During my working activity, I changed the development platform *For instance, I changed the used simulators software e.g., Fluent, Star-CCM+, PowerFlow, CFD++ etc.*

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

Q17. During my working activity, I changed the development platform and the impact on my work activities had:

0 I had no experience (No effect) 1 2 3 4 5 6 7 (Serious effects)

Q18. In future, the change of the actual used working platform (e.g., simulator software), I think that I will have an impact on my work

(No effect) 1 2 3 4 5 6 7 (Serious effects)

Project data accessibility

Q19. The access to the project performance is *(e.g., data about performance, revisions, change to the simulation models*:

(Difficult) 1 2 3 4 5 6 7 (Easy)

Q20. I have access to data about the project performances (e.g., performance data about different project revisions)

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

Q21. I think that the access to the global project data performance is

(Not very useful) 1 2 3 4 5 6 7 (Very useful)

Q22. Project data are distributed over different place of work:

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

Q23. I have an automatic system to generate statistics about simulations performed on the same project

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

Q24. I have an automatic system to generate statistics about simulations performed over multiple projects

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

Q25. An automatic system to generate statistics about simulations performed over multiple projects could be

(Not very useful) 1 2 3 4 5 6 7 (Very useful)

Q26. I have a system to generate automatic statistics about data coming from different simulator software:

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

Q27. A system to generate automatic statistics about data coming from different simulator software could be:

(Not very useful) 1 2 3 4 5 6 7 (Very useful)

Q28. I have a system to generate automatic statistics and compare data about numerical simulations and tests in the wind tunnel:

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

Q29. A system to generate automatic statistics and compare data about numerical simulations and tests in the wind tunnel could be:

(Not very useful) 1 2 3 4 5 6 7 (Very useful)

Q30. I have a system to visually compare the output (3D scene) from multiple simulations:

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

Q31. A system to visually compare the output (3D scene) from multiple simulations:

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

Data centralization and simulation data search

Q19. I follow some rules to store simulations and assign the name to their corresponding files

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

Q20. The information that I store in the simulation file name are *(Multiple choice)*

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

Q21. The rules are:

- *Personal Choice*
- *Team Conventions*
- *Fixed imposed rules*

Q22. I follow the rules over time
(Never) 1 2 3 4 5 6 7 (Always)

Q23. I am able to find simulations, geometries and results in useful time:
(Never) 1 2 3 4 5 6 7 (Always)

Q24. It happens that I am not able to find simulations
(Never) 1 2 3 4 5 6 7 (Always)

Q25. The opportunity to link other information (e.g., tags) to files could be:
(Useless) 1 2 3 4 5 6 7 (Useful)

Q26. 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)

Q27. I have a tool to search simulations according to their content
(Never) 1 2 3 4 5 6 7 (Always)

Q28. 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

Q29. I have a tool to show the simulation's modification over time
(Never) 1 2 3 4 5 6 7 (Always)

Q30. A tool to show the simulation revisions could be
(Never) 1 2 3 4 5 6 7 (Always)

Social network

Q31. I use an internal private social network to discuss on projects and methodologies:
(Never) 1 2 3 4 5 6 7 (Always)

Q32. I have been involved on discussions about topics of my interest:
(Never) 1 2 3 4 5 6 7 (Always)

Q33. It could be useful to involve my colleagues on interesting topics :
(Not much) 1 2 3 4 5 6 7 (Very much)

Q34. It could be useful to be involved on interesting topics to improve my know-how:
(Not much) 1 2 3 4 5 6 7 (Very much)

Q35. Do you think that a private social network could be useful?

____

____

____

Engineering Software

Q31. During my work I use these simulator software (multiple choices):

- *Star-CCM+*
- *OpenFOAM*
- *SolidWorks*
- *PowerFlow*
- *CFD++*
- *SU2*

Q32. To be used in the industrial field a software must

- have the support from open source community
- have the commercial support
- be widely adopted by the scientific community
- be widely adopted by the industrial community
- be validate for multiple applications
- have templates for industrial use case
- Other: ____________________________

Q33. The criteria that I use to evaluate an engineering software are:

- software license cost
- how it is easy to use
- how it is accurate
- the available functionalities
- Graphical User Interface
- how much it is used by the competitors
- commercial support
- Other: ____________________________