Simulation Repository Visualisation and Exploration

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

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

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

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

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

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

I. Introduction

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

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

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

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

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

II. Related Works

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

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

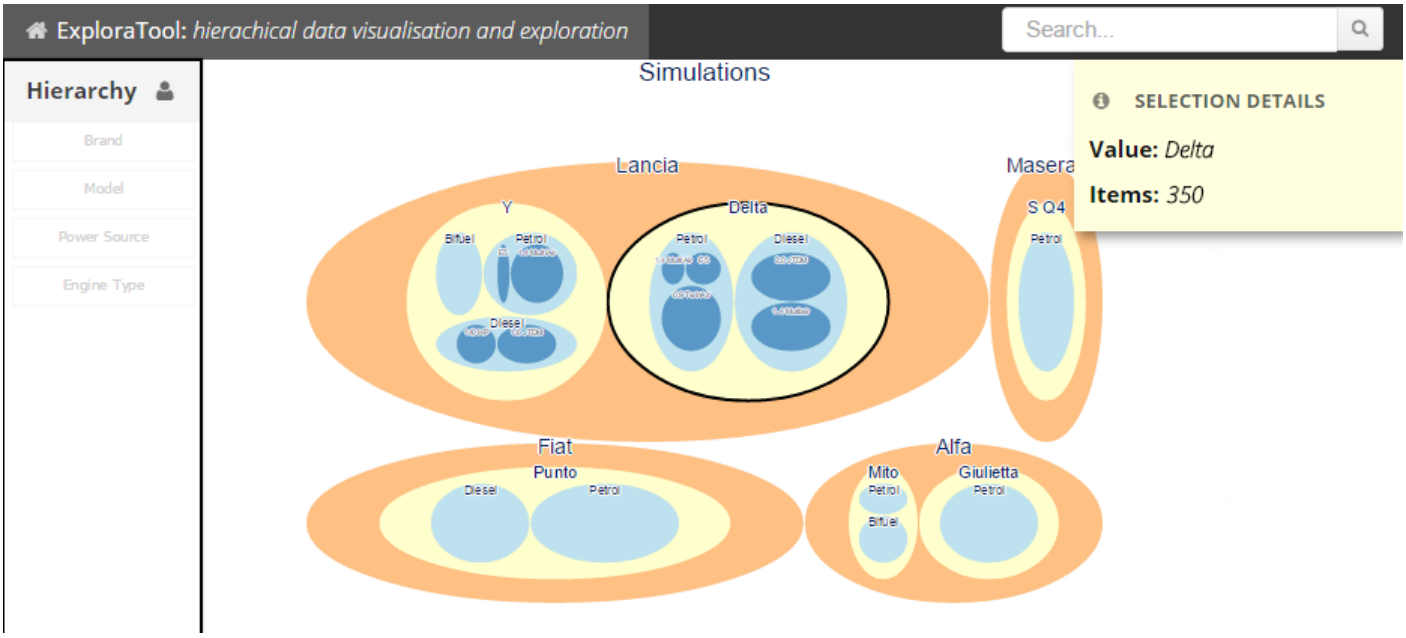

Fig. 1. The ExploraTool's Graphical User Interface. It shows an overview of the simulation repository through an initial hierarchy made by the following simulations' attributes: brand, project model, power source and engine type. The attributes' order is shown in the navigation bar on the left. Instead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

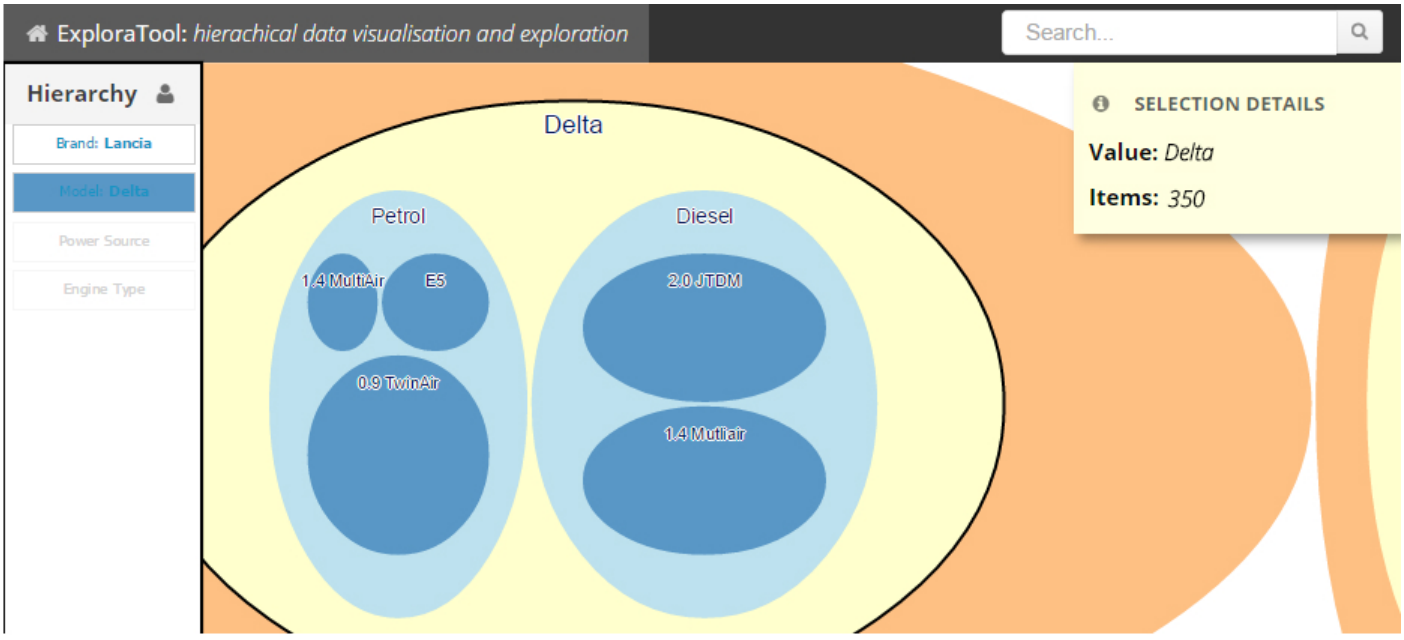

Fig. 2. The picture shows the result of the drill-down operation performed on the Delta category. Starting from Fig. 1, the analyst clicks on the ellipse "Delta". The ExploraTool smoothly enlarges the selected group, rendering a fast transition to the new view. If the user desires to go return back to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

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

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

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

III. EXPLORATOOL FEATURES

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

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

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

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

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

A. Data Exploration: in-depth navigation

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

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

B. Hierarchy Attributes Rearrangement

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

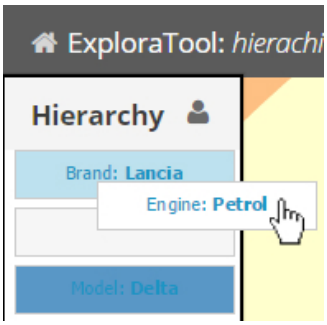

Fig. 3. Hierarchy rearrangement operation performed through the drag-and-drop of an attribute (facet) from its original position to a new slot. In this way, the user changes the order of the attributes, thereby updating the hierarchy.

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

C. ExploraTool Tasks

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

IV. ExploraTool Software Architecture

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

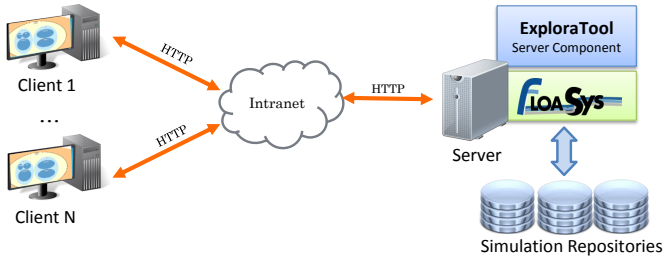

Fig. 4. ExploraTool prototype client/server architecture. ExploraTool is web-based, so analysts use the web-browser (clients on the left) to access to the server. On the Server-side (right part of the picture), ExploraTool accesses to the existing simulation repositories to retrieve the data and build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

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

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

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

A. Hierarchy building process

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

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

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

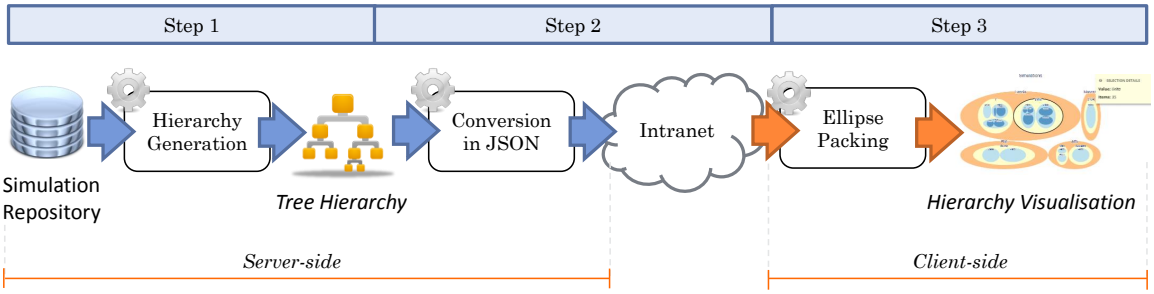

Fig. 5. Pipeline of transformation from the simulation repository to the visualisation on the client Web-browser. In order, the steps are: (1) reading of simulation data and creation of a tree data structure, (2) conversion of the tree data structure in JSON format, and transferring of the JSON data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

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

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

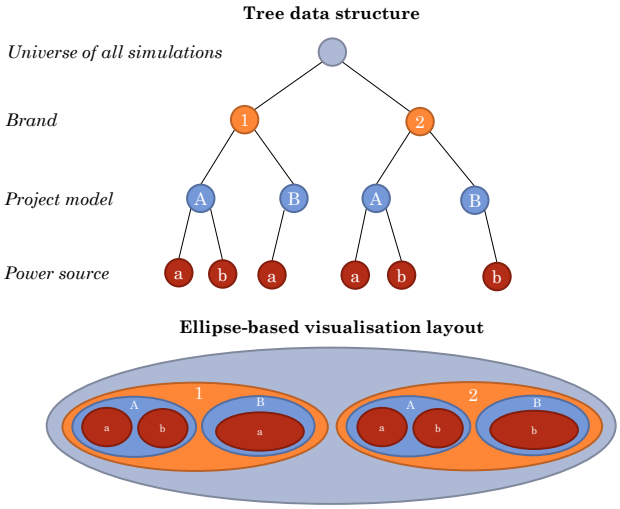

Fig. 6. The tree data structure built by the ExploraTool and the subsequent ellimap rendering. The ellipse layout is based directly on the tree data structure, in a manner so that each level in the tree is a layer of the ellimap, and each node of the tree is an ellipse.

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

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

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

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

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

V. Conclusions and Future Works

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

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

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

References

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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