Títol: Xpertum

Volum: 1/1

Alumne: Agustí Sancho Martínez

Director/Ponent: Esteve Almirall Mezquita

Departament: LSI

Data: 12.6.2009

DADES DEL PROJECTE

Títol del Projecte: **Xpertum**

*Nom de l'estudiant:***Agustí Sancho Martínez**

Titulació: **Enginyeria Informàtica**

Crèdits: **37.5**

Director/Ponent: **Esteve Almirall Mezquita**

Departament: **LSI**

MEMBRES DEL TRIBUNAL *(nom i signatura)*

*President:***Miquel Sanchez Marre**

Vocal: **Josep Grané Manlleu**

Secretari: **Esteve Almirall Mezquita**

QUALIFICACIÓ

Qualificació numèrica:

Qualificació descriptiva:

Data:

xpertum

Agustí Sanchó Martínez

Barcelona

June 1, 2009

Table of Content

Table of Figures

Tables

Abbreviations

LL	LinLog
MCL	Markov Cluster Algorithm
XML	Extensible Markup Language
KK	Kamada-Kawai Algorithm

1 Introduction

Improve the innovation process is the main objective of the European project Laboranova. I began to collaborate within the project on September 2009 and my main objective has been to achieve this objective. Xpertum is the result of my work and my contribution to develop a web-based tool that the organizations can use to confront the challenges of the future.

This document has been divided in 2 parts, the research and the prototype documentation. In the research part, I explain all the theoretical concepts that support the design and the functionalities of Xpertum and, in the prototype documentation part, I describe technical documentation, produced in the design tool step, that I have used to develop Xpertum.

Before to proceed with this document, it needs to be clarified the innovation process approach that have been tackled in Laboranova project, that includes the context and the purpose of the application. The innovation process is used to describe basically an array of sources and objectives that culminate in the act of innovation [Gellatly, Peters, 2000]. The innovation takes place inside of knowledge cycle (acquisition, assimilation and development) [Ho 2007]where it leads to creating or discovering new knowledge or technology, thus a new value could be created by applying the new knowledge to actual business or social challenges. There is no doubt that innovation is market-driven since it depends on its commercial value.

The knowledge can be divided into two categories: tacit and explicit, the first one can be easily expressed in words and numbers; the second one is context-specific, highly personal and hard to formalize and includes, among other things, intuitions and experiences [Nonaka 1998]. Nowadays, the organizations deal with metrics that capture the level of knowledge that its employees have in a certain moment, including both types of knowledge. Based on it, the leaders can take decisions to assemble teams, hire new employees or simply be aware of its technical potential. Xpertum offers control over these process to help the organization, to achieves their objectives successfully.

The innovation requires multidisciplinary perspectives to get a high commercial value. Innovation needs a balanced intellectual attitude and level of knowledge as well, in order to find all problems-solutions. The level of knowledge can be represented as a competence. Even though, the concept of competence has different meanings, Xpertum focused on the level of knowledge based on the results of previous actions to enrich the know-how [Ryle 1949] in certain technical data or discipline. That being so, a set of competences could be found among people involved in the innovation tasks.

A social network analysis maps the relationships between people or groups to understand how these relationships could help or being an obstacle to the knowledge flow and human interaction. We can represent a social network in terms graphs (nodes and edges). Nodes represent individuals and edges are the relationships between them. If all nodes in the social network are categorized in the same type, the network is a one-mode network, otherwise is an n-mode network. In social networks represented as graphs, it could find edge and weighted-edge, the first one only represents a relationship between nodes, the second one represents the strength of relationship between the nodes in a measured way. The type of weights will depend on how the social network is built and the kind of relation that it represents.

Visualization of social networks has provided new insights to researchers about structure and shape about it. There are several tools to visualize social networks and its possible links with concepts (such as: tags). However, most of these tools could be complex to manipulate (several visualization with non-precise meaning), support visualizations of their own data or accessrestricted. Moreover, the majority of visualization provided by tools are egocentric (focused on individuals) and it is not useful when relationships between concepts and users have to be highlighted as well.

The approach of Xpertum to fostering innovation is based on augmenting social capacities of innovators, i.e. enhance their social networking and their sharing of concepts and ideas [Andrade, et al. 2007] and offering a friendly user interface.

The objective of my work is provide to organizations a web-tool that offers these possibilities and create a service that requires minimum resources using the software engineering beginning with a prototype application and develop a tool to improve the innovation process.

2 Research

2.1 Concepts

2.1.1 Introduction

Social network theory represents social relationships in terms of nodes and edges. Nodes are the individual actors within the networks, and edges are the relationships between the actors. Then, a edge relates two actors. The social network is a finite set of actors and all possible relations among them. If all actors in the social network are of the same type, the network is a one-mode network, otherwise is an n-mode network.

The shape of the social network helps determine a network's usefulness to its individuals. Smaller or tighter networks can be less useful to their members than networks with lots of loose connections (weak edges) to individuals outside the main network. An important characteristic is that an analogy of social networks and mathematical models can be done. A social network could be represented as a graph.

In social networks, it could find edge and weighted-edge, the first one only represents a relationship between nodes, the second one represents the strength of relationship between the nodes in a measured way. The type of weights will depend on how the social network is built and the kind of relation that it represents.

Visualization of social networks has a rich history, particularly within the social sciences, where node-link depictions of social relations have been employed as an analytical tool since at least the 1930s. Linton Freeman documents the history of social network visualization within sociological research, providing examples of the ways in which: spatial position, color, size, and shape can all be used to encode information (Freeman). For example, networks can be arranged on a map to represent the geographic distribution of a population. Alternatively, the generated layouts have useful spatial properties: a force-directed layout can be quite effective for spatially grouping connected communities, while a radial layout intuitively portrays network distances from a central actor. Different factors such as: color, size, and shape have been used to encode both topological and non-topological properties such as centrality, categorization, and gender.

Xpertum system has the objective to display relationships between persons and concepts. To improve the selection process and consult the staff state process, we use affiliation of knowledge like concepts. These affiliations are organized in areas of knowledge that we mention simply areas.

To display the relationships, Xpertum uses two different types of graphic layout, the graphs and the treemaps.

2.1.2 Elements

As we said, Xpertum displays visualization of relationships between persons and competences. We decided use these types of elements with the objective of improve the research process, because the selection process of a new team (with the objective of make a research project), is critical to obtain the success. The project manager have the responsibility of ensemble his/her team, he needs to respond the following questions “What is the best team to realize the project?”, “Is possible realizes the project with my human resource?”, “Maybe need I external persons to realize the project?”, …. to improve and foresting the innovation process with base on multidisciplinary competences among involved persons and could produce innovative solutions.

For these reasons Xpertum displays the relationships between persons and competences, because, to realize a project, project managers need a human team that can satisfy the competence and knowledge that the project requires.

We decided to group the competences in areas to simplify the task of work with a great number of competences.

Finally, the type elements that Xpertum will offer to the user are:

- **Person**

Attribute	Description
Firstname	First name of the person
Lastname	Last name of the person
Alias	Alias of the person
Email	Email address of the person
Jobrole	Current job role of the person into the organization
Organization	The organization where the person works
Phone	The number phone of the person
Region	The region where the person lives
Postalcode	The postalcode where the person lives
Country	The country where the person lives
Timezone	The timezone of the person's zone
Picture	A picture of the person

Table 1: Attributes of Person elements

- **Competence**

Attribute	Description
Name	The name of the competence
Area	The area of the competence

Table 2: Attributes of Competence elements

- **Connection**
  To display the relationships between these types of elements, we use connections with a strength value, to represent the force of the relation.

2.1.3 Visualization Types

Knowledge representation in visualization tools is always associated a typical graph structure. Where individuals and competences are nodes, and the strength between them are represented by ties. Focusing in the project, we wanted to represent as well, some kind of different representation. We found it in a bipartite mode representation. In one hand individuals, and in the other competences, and ones connected to others, but not within themselves. Next section describes it. In addition, we found another technique for representing data called Treemaps that offers an artistic and pleasant visualizations to display the individuals and competences.

2.1.3.1 Common characteristics

2.1.3.1.1 Colors of the elements

The use of the colors in Xpertum is a critical factor to the realize the functionalities. We decided use the same colors for every visualizations, because it's very important that the user can identify the meaning of the visualizations without external tools.

For this, we associated for the person elements always a colour that we don't use for the competence elements.

For the competences, we decided associate a different colour for every competence area. Thanks this, the user can differentiate between the different areas only viewing the visualizations.

2.1.3.1.2 Size of the elements

The size of the elements in all the visualizations, determines the relevancy of the element in function of its relevance.

In the case of persons, the size of the visual representation is determinate by the number of relationships with a competences are being visualized. Thanks to this, it's so much easy for the user identify the most relevance persons at current visualization.

In other hand, the affiliation size is determined by how many individuals does it have connected. It means, how many individuals relate to this affiliation. The bigger the number of affiliations is, the bigger the circle becomes.

So, this way, it is easier to identify an important affiliation in the network, because it is bigger in size than the other ones.

Another important thing to be taken into account related to this last one is that size is the same in all application for a given affiliation (unless some individual relations related to this the affiliation change during the session). It means that in other visualizations, such as *affiliations* or *treemap* ones, a given affiliation of a certain affiliation type has the same size (in *affiliations* representation).

2.1.3.1.3 Correlation coefficient

Another important concept to be taken into account is the Pearson's correlation coefficient. It has been used for both individuals and affiliations visualizations. First we will expose what is it, after that we will explain how others use collaborative filtering for recommending things. In addition, we will explain why we have chosen Pearson's correlation coefficient and for what have been used it, and we will show where we have applied it as well.

In statistics, the Pearson product-moment correlation coefficient (sometimes known as the PMCC) is a measure of the correlation of two variables *X* and *Y* measured on the same object or organism, that is, a measure of the tendency of the variables to increase or decrease together.

The Pearson coefficient is a statistic which estimates the correlation of the two given random variables. So it is widely used in collaborative filtering applications for recommending products related to one is being consumed, checked, interested on or so. We want to apply it for visualizing relations between individuals or affiliations. So, applying it, we are able to see which individuals are near to others or which affiliations are as well. In knowledge representation it can mean which users may be interested in the same things according what they have defined as affiliations in the application and, vice versa, which affiliations are interesting to the same people.

A very well known recommendation system is the one build by Amazon. It uses item-to-item collaborative filtering. A recommendations service recommends items to individual users based on a set of items that are known to be of interest to the user, such as a set of items previously purchased by the user. In the disclosed embodiments, the service is used to recommend products to users of a merchant's Web site. The service generates the recommendations using a previously-generated table which maps items to lists of "similar" items. The similarities reflected by the table are based on the collective interests of the community of users.

Weighting the relation seems to be a problem in collaborative filtering and when using Pearson's correlation coefficient, but in our case, the relation weight is done by the own user, who exposes his affiliations preferences and rate them from 0 to 100%. So, we only have to calculate the resultant distances matrix for knowing how far or near has an individual or an affiliation to be from other individuals or affiliations respectively.

So in our project, we are use the same technique that Amazon is using, the item-to-item collaborative filtering.

2.1.3.2 Graphs

2.1.3.2.1 Introduction

We decided use a drawing graph to show the relationships between persons and concepts because, for a user that haven't any type of visualization information knowledge, is the most naturally way to understand the information.

Our graphs use circles to represent the nodes (persons or concepts) because closed contours are basic in defining visual objects and for human mind, is a powerful form that pay her attention. Through the circles, Xpertum displays 3 types of a different information, using the size of circle, the colour and a label.

The colour is used to display the type of node, the person nodes have a different colour that affiliation nodes, and these, have a limited colour palette (excluding person colour). Every affiliation colour display to user the area that the affiliation belongs.

The size is used to show the relevance of the node in terms of the importance and quantity of their relationships.

Finally, the label is used to display the name that identifies the node. For persons, Xpertum display the person name, and I case of affiliation, the name of affiliation.

The other component of a graph, the lines that connect nodes, have used to display the relationships between two nodes.

Using drawing graphs Xpertum display information in 3 different layouts, that use different mathematical algorithm to visualize the information and display the relationships since different parameters.

2.1.3.2.2 Using Bipartite Graphs

Previous work with bipartite graphs in social network representations can be found in a paper (Alexander) made by Malcom Alexander where they use bipartite graphs (2-mode social network data) for visualizing and analyse the actor-by-event datasets. They justify its use as follow:

"2-mode social network data is generated when a researcher assembles information that links a population of social actors with an array of collective or corporate entities such as committees, boards, or social clubs (Borgatti and Everett, 1997) (Wasserman and Faust, 1994). The key aspect of 2-mode data is that it does not record direct relations between social actors or direct connections between the collective entities although such direct relations can be derived from this data as 1-mode datasets. Conventionally we label the collective entities as 'events' and speak of an actor-by-event matrix. 2-mode datasets are rectangular (n x k) matrices and the number of actors is nearly always more than the number of events. Conventionally the row headings are the social actors and the columng heading the events. An entry in the aij cell indicates that actori is associated with eventj."

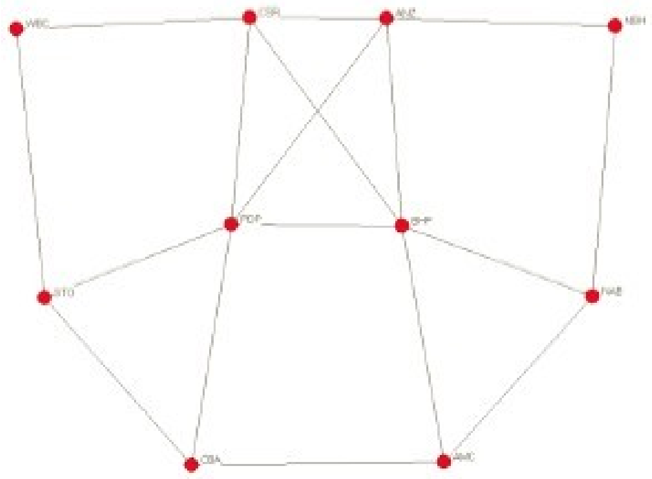


Figure 1: Inter-corporate network

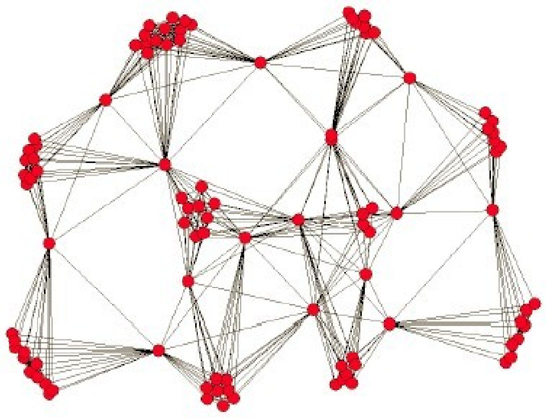


Figure 2: Interpersonal network

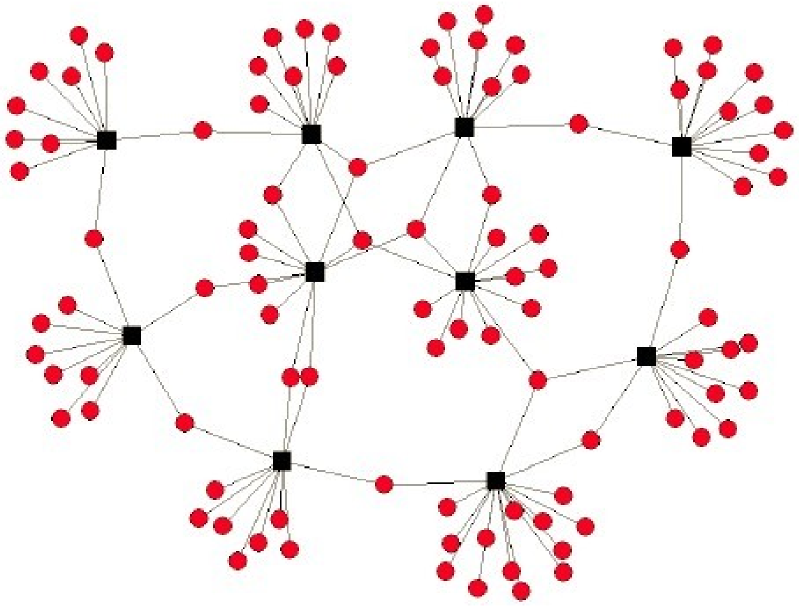


Figure 3: Mode Network

Focusing in our project, this study made use to realize that these three visualizations of the knowledge representation conform a basic set of knowledge. In the first figure an inter-corporate network illustrates the relations within different corporations; In the second one, an interpersonal network represents the social connections between people; and at last, the both of them together shows, in only one sight, a physical structure of the knowledge when relating people to corporations.

Kazuo Misue (Kazou, 2006) also used bipartite graphs to draw Anchored Maps. Supposing that the node set of a bipartite graphs is divided into set A and set B. On an anchored map, as He defined, of the bipartite graph, the nodes in A, which are called "anchor nodes", are arranged on the circumference, and the nodes in B, which are called “free nodes”, are arranged at suitable positions in relation to the adjacent anchor nodes. His paper describes aesthetic criteria that are employed according to the purpose of drawing anchored maps and a method to arrange the anchor nodes so that they satisfy the criteria. But it is not considered in our project, since anchored map is not helping us in the interaction part of our purpose.

2.1.3.2.3 Types

2.1.3.2.3.1 Geometric

With the objective to offer a main visualization to Xpertum, we implemented a customized algorithm to display graphs based on geometric rules and obtain a perfect combination between the importance of show the relationships of the social networks and their elements and offer a characteristic and personal main visualization.

- **Motivation**

  We decided that it's very important for the user receives a constant visualization that at the same situation (nodes and relationships) can display always the same graph to obtain a reference visualization for users. The others algorithms based on force haven't this property, and then we decided design an algorithm to satisfy this requirement.

- **The algorithm**

  Our algorithm displays the graphs of persons and affiliations (competences) in 2 stages.

  The first stage order the affiliation nodes in a ellipse around the centre of the display area.

  The second stage order the person nodes into the ellipse of affiliations. The position for every person was computed in function of the position of their affiliations, that is to say, the affiliations that are connected with the person, and the strength of the relationships between every pair.

  The computational cost of the algorithm is:

  $\Theta(p * e)$

  p = number of person nodes

  e = number of edges that relate persons and affiliations

- **The implementation**

```
function gllayout(Affiliation[] affs, Persons[] persons Edge[] edges, Dimension d) {



        //Align the affiliations in a elipse
        for (int i = 0; i < affs.length; i++) {
                affs[i].pos_x = ((d.getWidth() / 2 - 50) * Math.cos(2 * i *      Math.PI/
                affs.length));
                affs[i].pos_y = ((d.getHeight() / 2 - 50) * Math.sin(2 * i
                                * Math.PI / affs.length));
                affs[i].pos_z = 0;
        }

        //Compute the position respect the affiliations for every person
        int cx = 0;
        int cy = 0;
        for (int i = 0; i < persons.length; i++) {

                double w = 0.0;
                double rx = 0.0;
                double ry = 0.0;

                for (int j = 0; j < persons[i].edges.size(); j++) {

                        Edge edge = persons[i].edges.get(j);

                        if (edge.target_vertex.type.equals("affiliation")) {

                                double x2 = edge.target_vertex.pos_x;
                                double y2 = edge.target_vertex.pos_y;
                                // "a" is the tangent
                                double a = (y2 - cy) / (x2 - cx);
                                // Obtain the angle
                                a = Math.atan(a);
                                //Obtain the hypotenuses
                                double zzx = edge.strength * Math.cos(a);
                                double zzy = edge.strength * Math.sin(a);
                                if (((x2 - cx) < 0 && (y2 - cy) < 0)) {
                                        if ((x2 - cx) < 0) {
                                                zzx = zzx * -1;
                                        }
                                        if ((y2 - cy) < 0) {
                                                zzy = zzy * -1;
                                }
                            } else if (((x2 - cx) >= 0 && (y2 - cy) >= 0)) {
                                if ((x2 - cx) < 0) {
                                    zzx = zzx * -1;
                                }
                                if ((y2 - cy) < 0) {
                                    zzy = zzy * -1;
                                }
                            } else if (((x2 - cx) >= 0 && (y2 - cy) < 0)) {
                                if ((x2 - cx) < 0) {
                                    zzx = zzx * -1;
                                }
                                if ((y2 - cy) < 0) {
                                    zzy = zzy * 1;
                                }
                            } else if (((x2 - cx) < 0 && (y2 - cy) >= 0)) {
                                if ((x2 - cx) < 0) {
                                    zzx = zzx * -1;
                                }
                                if ((y2 - cy) >= 0) {
                                    zzy = zzy * -1;
                                }
                            }

                            rx += zzx;
                            ry += zzy;

                            w += edge.strength;
                        }
                    }

                    // Fixing the person coordinates
                    if (persons[i].edges.size() > 1) {
                        persons[i].pos_x = ((d.width/2 - 125) * rx * (1 / w));
                        persons[i].pos_y = ((d.height/2 - 125) * ry * (1 / w));
                        persons[i].pos_z = 0.0f;
                    } else {
                        persons[i].pos_x = ((d.width/2 - 125) * rx);
                        persons[i].pos_y = ((d.height/2 -125) * ry);
                        persons[i].pos_z = 0.0f;
                    }
                }
                //Centrer the visualization vertexes
                Vertices[] vertices = persons.add(affs);
                for (int i = 0; i < vertices.length; i++) {
                    vertices[i].pos_x += d.width / 2;
                    vertices[i].pos_y += d.height / 2;
                }
            }
```

- **Execution example**

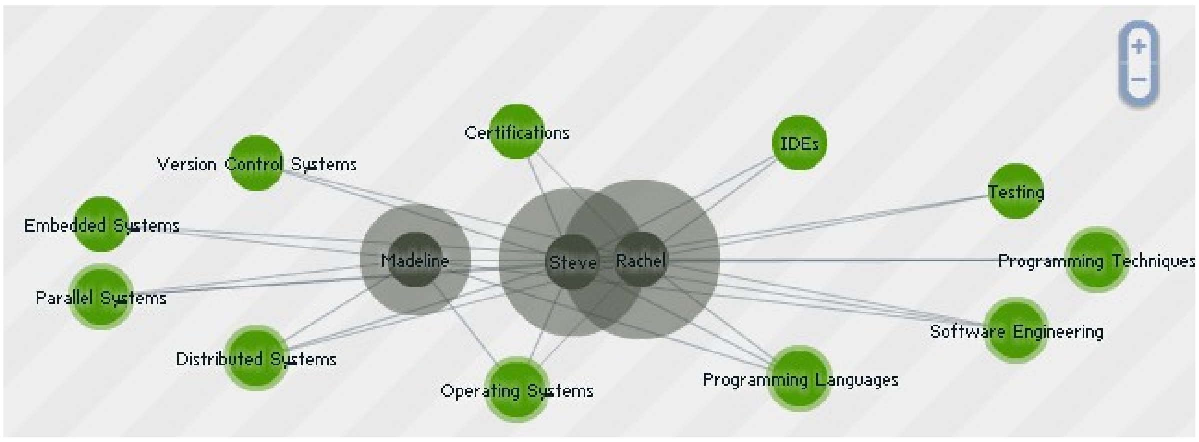


Figure 4: Example of Geometric Layout

2.1.3.2.3.2 Force

Another way to approach the visualization of graph representations of social networks is the one presented by a family of algorithms under the generic name of force-based layout model.

Force-directed algorithms are a class of algorithms for drawing graphs in an aesthetically pleasing way. Their purpose is to position the nodes of a graph in two dimensional or three dimensional space so that all the edges are of more or less equal length and there are as few crossing edges as possible.

The force-directed algorithms achieve this by assigning forces amongst the set of edges and the set of nodes. The forces are applied to the nodes, pulling them closer together or pushing them further apart. This is repeated iteratively until the system comes to an equilibrium state; i.e., their relative positions do not change any more from iteration to the next. At that moment, the graph is drawn. The physical interpretation of this equilibrium state is that all the forces are in mechanical equilibrium.

The major disadvantage is that the simulation can be very slow: order n^2 where n is the number of nodes. This is because in every iteration, all pairs of nodes need to be visited and their mutual repulsive forces computed.

- **Algorithms**
  - **Kamada-Kawai**

  The Kamada-Kawai Algorithm is a force directed layout algorithm. The idea of a force directed layout algorithm is to consider a force between any two nodes. In this algorithm, the nodes are represented by steel rings and the edges are springs between them. The attractive force is analogous to the spring force and the repulsive force is analogous to the electrical force. The basic idea is to minimize the energy of the system by moving the nodes and changing the forces between them.

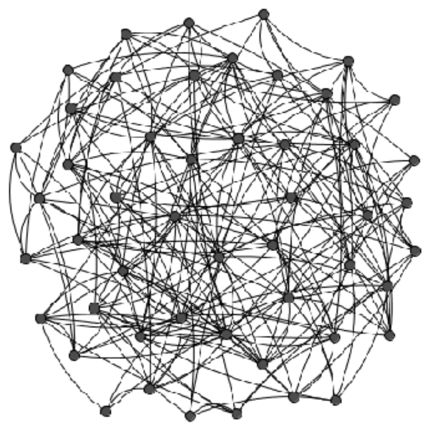


Figure 5: KK algorithm applied to a network

Pros & Cons

The KK algorithm achieves faster convergence and can be used to layout networks of all sizes. However, to obtain an aesthetically pleasing layout it sometimes becomes necessary to use the Fruchterman-Reingold algorithm after the Kamada-Kawai generates an approximate layout.

Applications

The Kamada-Kawai layout can be used to visualize networks of any sizes. It is used in several graph based evolution softwares such as Gevol.

- **Fruchterman-reingold algorithm**

The Fruchterman-Reingold Algorithm is a force-directed layout algorithm as well.

In this algorithm, the sum of the force vectors determines which direction a node should move. The step width, which is a constant, determines how far a node moves in a single step. When the energy of the system is minimized, the nodes stop moving and the system reaches its equilibrium state. The drawback of this is that if we define a constant step width, there is no guarantee that the system will reach equilibrium at all. T.M.J. Fruchterman and E.M. Reingold introduced a "global temperature" that controls the step width of node movements and the algorithm's termination. The step width is proportional to the temperature, so if the temperature is hot, the nodes move faster (i.e, a larger distance in each single step). This temperature is the same for all nodes, and cools down at each iteration. Once the nodes stop moving, the system terminates.

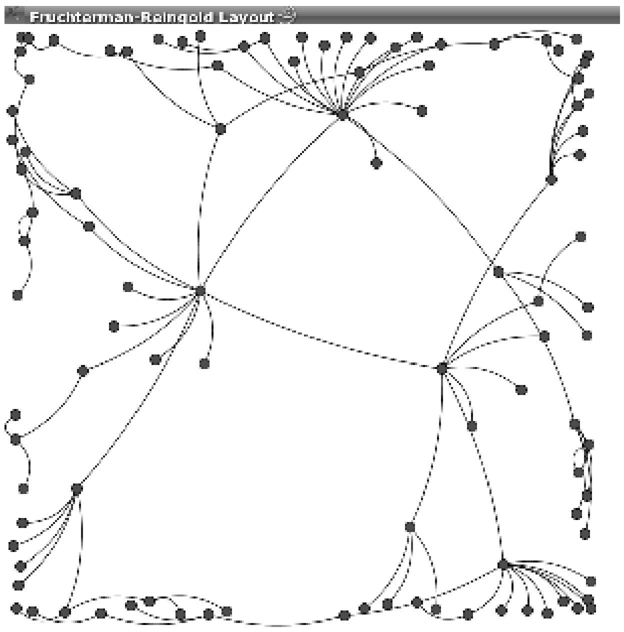


Figure 6: Fruchterman-reingold algorithm applied to a netowork

Pros & Cons

The Fruchterman-Reingold Algorithm is useful for visualizing very large undirected networks. It guarantees that topologically near nodes are placed in the same vicinity, and far nodes are placed far from each other. Using a global temperature creates an overall satisfying layout, however there will be deficiencies in some local areas of the graph. This can be improved by using the "adaptive temperature scheme" based on local temperatures.

Applications

The BioLayout program which is used for visualisation of biological networks is the C-implementation of the Fruchterman-Reingold algorithm. A Java version of the same program is also available.

2.1.3.2.3.3 Clustering

The clustering techniques in graphs are applied to obtain groups of the elements of a them and to obtain visualizations that display the relationships most relevant using the positions of elements and other visual elements (like colour shapes).

We researched the main clustering layouts algorithms for Xpertum visualizations and found two different solutions that are interesting for the project, the algorithms based on force and based on flow.

- **Algorithms**
  - **Energy: LinLog (LL)**

Energy-based methods are a good method to create clustering representation of undirected graphs and it's very appreciated for Xpertum.

The energy algorithms have two parts: an energy model for the graph elements, and an algorithm that searches a state with minimum total energy.

LinLog is an energy model algorithm to clustering graphs. The visualization that obtain LL is a unconnected graph and the relationship between the elements are showed with the dimensional proximity of the elements.

Pross & Cons

The layout offers a very understandable visualizations and the groups that obtain of the graph are clearly distinguished between themselves.

Unfortunately, the search of the minimum total energy is an expensive process, but can be adjust in function of the quality of the result.

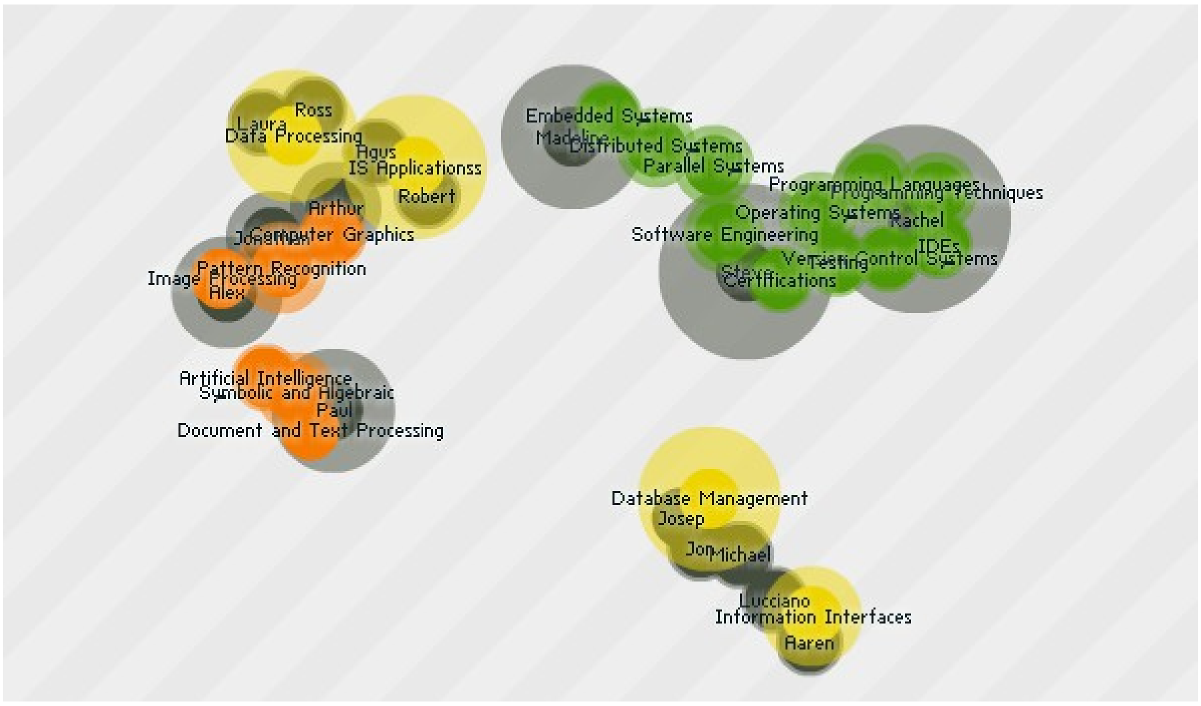


Figure 7: LL applied to a graph of persons and competences

- **Flow: Markov Cluster Algorithm (MCL)**

The Markov Cluster Algorithm is a fast and scalable unsupervised cluster algorithm for graphs based on simulation of (stochastic) flow in graphs. The algorithm was invented by Stijn van Dongen at the Centre for Mathematics and Computer Science in the Netherlands.

The MCL defines a sequence of stochastic matrices by alternation of two operators on a generating matrix. The result of the algorithm's in a connected graph is the same graph with its elements group around, every group, an element.

The MCL is an algorithm of clustering that must be applied in a graph visualization, because doesn't change the positions of elements, only erase edges between nodes to obtain the cluster groups.

The element that contains all the connections of a cluster is the most relevant and helps to obtain the visual group effect.

Pross & Cons

This algorithm is very useful to obtain clusters of graphs that have the same type of nodes, but for bipartite graphs loses the visual references and can be confuse for the users to understand the visualizations.

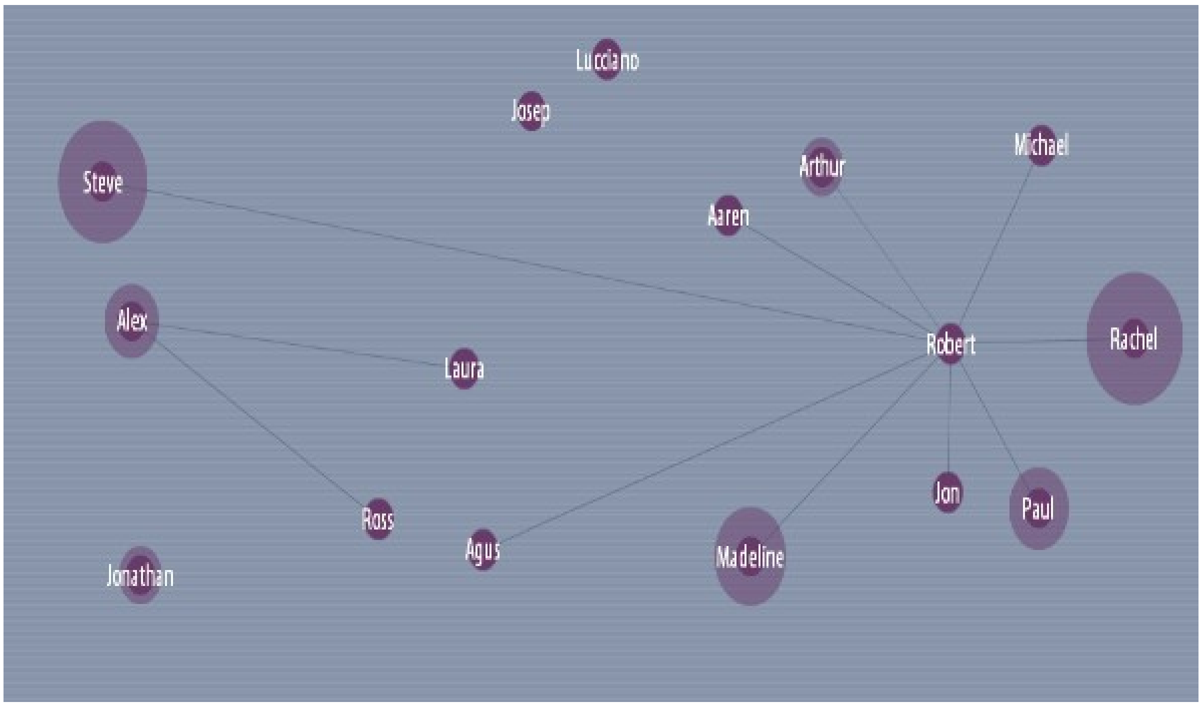


Figure 8: MCL applied to a graph of persons in a KK layout

2.1.3.2.4 Collision System

- **Motivation**

  The visualization graph algorithms consider the nodes like a point, without thickness, without shape and in our circle representation, without radius.

  The visualizations were taken, despite its mathematical precision, might not be visually understandable, so we decided to apply geometric collision techniques to solve the problem.
  With this goal we developed a collision algorithm that subtly modifies the positions nodes without change the general visualization shape. This algorithm's implemented in visualization layer to ensure that the domain's not affected and the different devices can choose how to process the common information that the domain of Xpertum provides.

  This procedure for visualizations has been adjusted to obtain the bests results with the lowest cost. Nevertheless, evidences suggest that from a number of nodes the algorithm cost becomes an unnecessary burden. So, the algorithm only can work on a concrete scene conditioned by number of nodes.
  Despite these facts, we obtain more understandable and natural visualization for the users without cost sacrifices when the number of nodes are appropriated for Xpertum Visualization Layer.

- **The algorithm**

  The collision algorithm makes a path by all the graph nodes and, for each node, check that doesn't collide with the others circles using the center points and the radius. If they collide, then the algorithm change the position of nodes using the repulsion force, that moves the two nodes in opposite direction that obtain with opposite vectors that are in the same straight that includes the two center points of nodes. The repulsion force is calculated in function that the view port size and a range that we defined.

  This operation's realized meanwhile in a iteration doesn't detect collisions, i.e., the visualization is stable or when the number of iterations don't overcome a constant that we defined to avoid an excessive consume of the resources, because the view port is an limited space but the high number of nodes isn't defined.

  After the proves that we did, we decided that the number of iterations of the check detection, the *Niterations,* to obtain an optimal results can't be more greater than 15.

  Less iterations don't modify sufficiently the visualization for the actual view ports and more, isn't good idea, despite the visualization algorithms are fiscally in an other machine, thanks to distributed architecture, because is possibly that doesn't exist any stable situation because is possible that don't fit in the view port within collisions, and the changes since the *Niteraions* can be unnecessary.

  The time computational cost of the algorithm is:

  $\Theta(Niterations * n^2) = \Theta(15 * n^2) = \Theta(n^2)$

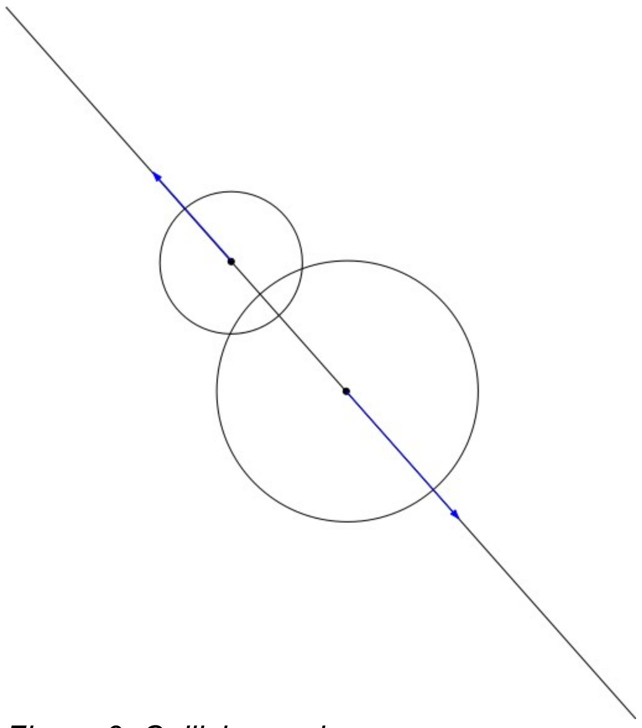


Figure 9: Collision nodes

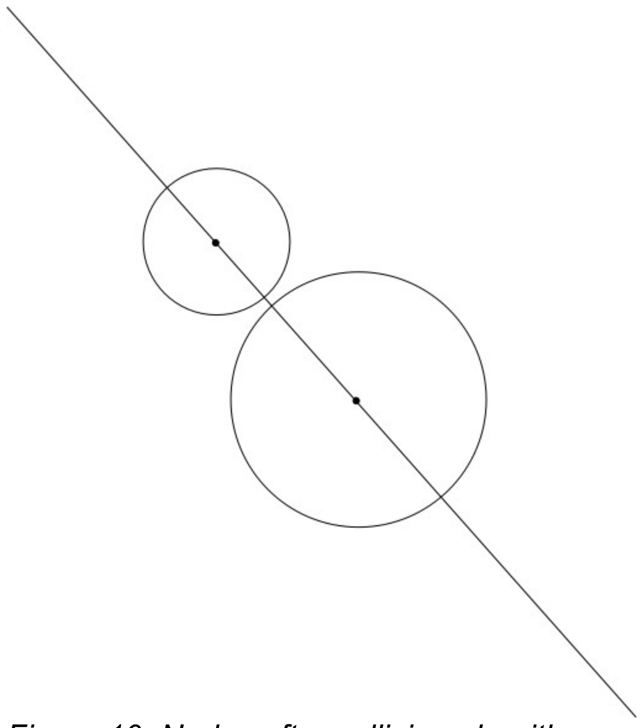


Figure 10: Nodes after collision algorithm

- **The implementation**

```
 function collisionCheck( int viewport_width, int viewport_height , Vector<Node>
nodes){

        Bool stable = false;
        int  i_iteracion = 0;

        //Repulsion Force
        int wh = iewport_width + iewport_height;
        Double repulsionForce = (-(0.097)*wh) + 197;

        //Adjust the repulsion to be efficiently
        if(repulsionForce < 3)            repulsionForce = 3;
        else if(repulsionForce > 100)            repulsionForce = 100;

        //The main loop
        while(!stable && i_iteracion < 15){
                stable = true;
                //For each realtion between nodes
                for(int i = 0 ; i < nodes.size() ; i++)
                        for(int j= i+1 ; j < nodes.size() ; j++){
                                if( collide(nodes[i], nodes[j]) ){
                                        stable = false;
                                        repulse(nodes[i], nodes[j], repulsionForce);
                                }
                        }
                i_iteracion++;
        }
}

function collide(Node n1, Node n2):Boolean{
//The distance between the centers of nodes
```

```
Double dist = Math.sqrt( Math.pow( (n2.x - n1.x),2) + Math.pow( (n2.y - n1.y),2));

//Returns true if the distance is less than 75% of the sum of the radius
return dist <= (n1.radius+n2.radius -(n1.radius +n2.radius)*0.25);
}

function repulse(Node n1 , Node n2 , Double repulsionForce){
        Point vec1, vec2;
        Double mod;

        //If the centers are th same point, we change the position of the first node to obtain a
        distance between centers
        if( n1.x == n2.x && n1.y == n2.y ){
                n1.x--;
                if(n1.x < 0)
                        n1.x = 2;
        }
        //Repulsion direction of n2
        vec1.x = n2.x - n1.x;
        vec1.y = n2.y - n1.y;
        mod = Math.sqrt( Math.pow(vec1.x,2)+Math.pow(vec1.y,2));
        vec1.x /=  mod;
        vec1.y /= mod;

        //Repulsion direction of n1
        vec2.x = n1.x - n2.x;
        vec2.y = n1.y - n2.y;
        mod = Math.sqrt( Math.pow(vec2.x,2)+Math.pow(vec2.y,2));
        vec2.x /=  mod;
        vec2.y /= mod;

        Double distExpulsion = Math.min(n1.getRadius(), n2.getRadius()) / repulsionForce;
        n1.x=  n1.x + vec2.x*distExpulsio;
        n1.y = n1.y + vec2.y*distExpulsion;
```

```
            n2.x = n2.x + vec1.x*distExpulsion;
            n2.y = n2.y + vec1.y*distExpulsion;
    }
```

- **Execution examples**

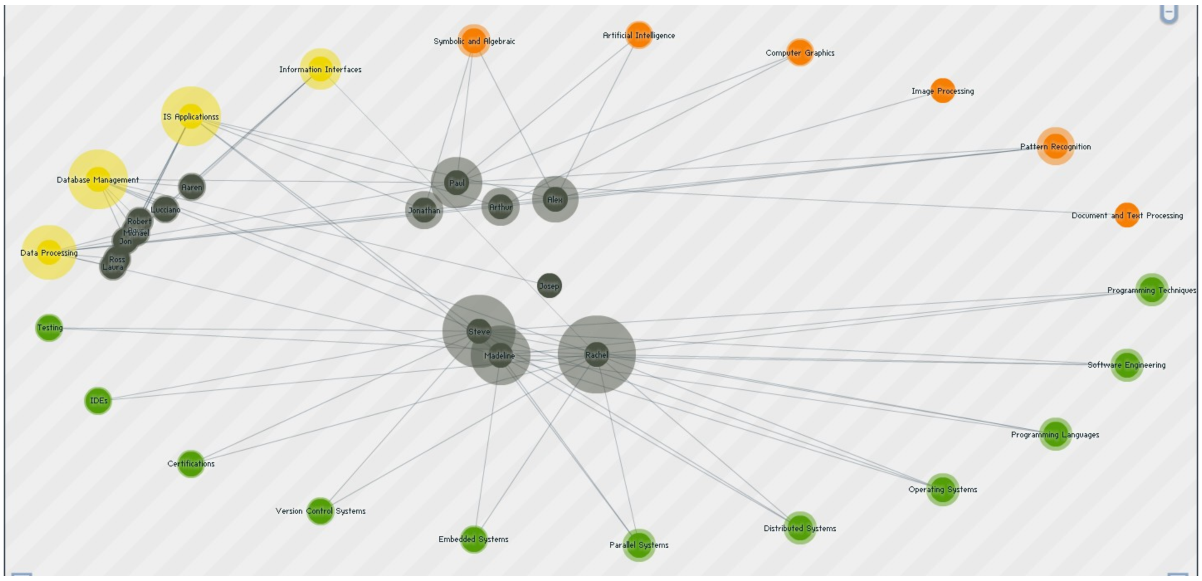


Figure 11: Geometric layout without collision algorithm

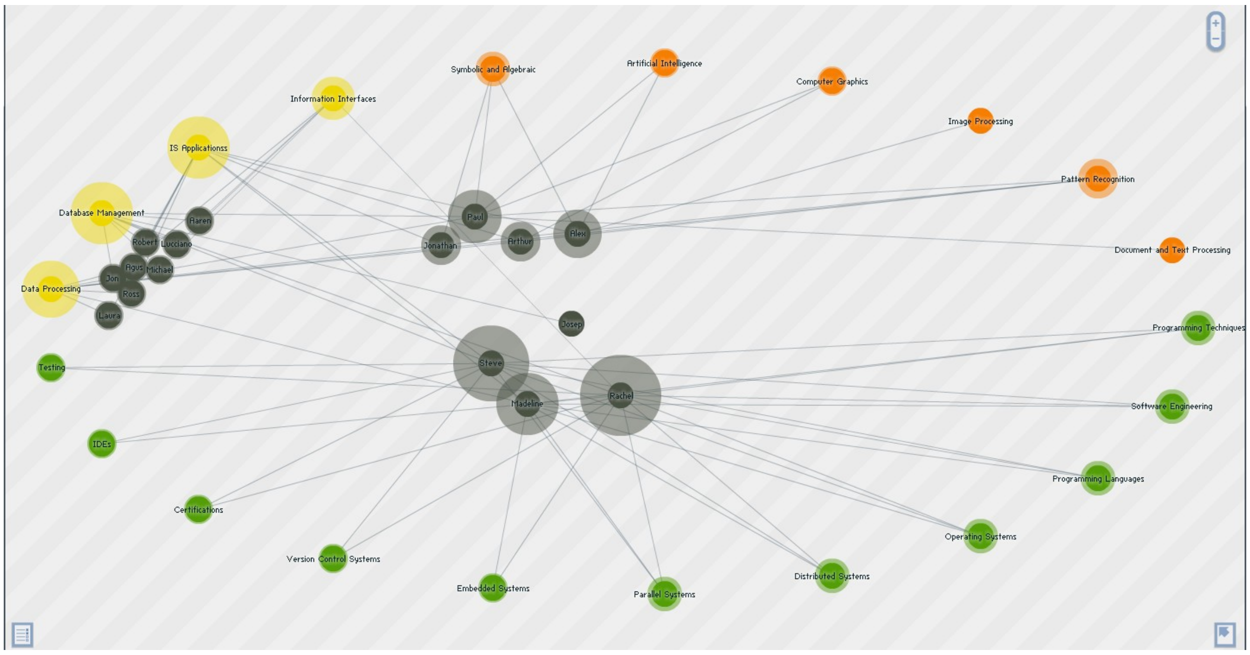


Figure 12: Geometric layout with collision algorithm

2.1.3.3 Hierarchical Visualization

In this section, we broadly describe treemap technology, an hierarchical visualization; first, by explaining and showing a simple treemap image, at-a-glance, then by presenting a more complete synopsis of its distinguishing features. Next, we provide the algorithm options that exist for implementing treemaps, which one have we used, for what and, finally, this section ends with the implementation that we have used.

Treemap technology arranges hierarchically-structured, categorical data, into a rectangular, mosaic image. The treemap image could be largely thought of as an amalgamation of a tree diagram and a pie chart, with additional features; it depicts both the structure and the content of the data hierarchy.

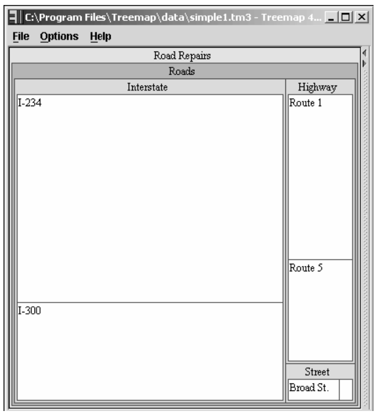


Figure 13: A treemap of a read maintenance record (SHNEIDERMAN & PLAISANT, 2003)

The treemap algorithm partitions the display space into a collection of box-shaped elements, forming a mosaic. Each inner box pertains to an individual data element and is sized (the physical area the box takes in the display space) according to a quantitative value.

The treemap is intended to be used interactively on a computer and displayed on a computer screen, rather than viewed as a static, or printed, display. This allows for effortless exploration of the data hierarchy with concurrent appraisal of the quantitative aspects of the data. In addition, computer screen mouseovers are often built-in to treemap software, which can provide detailed element-specific information to the user simply by placing the computer mouse over the box of the element of interest. These interactive features situate the treemap as an immensely powerful tool for visual exploration of large datasets.

- Characteristic features

  Johnson and Shneiderman (1991) set out four design objectives for the treemap: (a) efficient space utilization, (b) interactivity, (c) rapid comprehension, and (d) pleasing esthetics.

A hierarchical dataset contains two kinds of information: structural and content (Johnson & Shneiderman, 1991). The *structural* component is associated with the hierarchy; usually thought of as the network of multi-level relations, as in a non-cyclic tree structure, with nodes at the terminals. The *content* pertains to content of each node, that when rolled-up its hierarchy and has meaning at the higher, grouped level(s), as well as at the individual node level.

From this perspective, there are three variables that can be represented in a treemap: (a) a quantitative value for each entity, (b) the group membership of an entity, and (c) a relative and ordered relationship of each entity within a group; respectively, these variables are represented by (a) the scaled size of the rectangle representing the entity, (b) the placement of a entity within the larger rectangle, and (c) the placement of an entity's rectangle with the parent rectangle. The placement aspect (item c) is an area that is currently being developed; so far, there seems to be inherent physical limitations to reflecting contextual meaning to the placement of entities within a parent box.

- **Application to Social Network Analysis**

  While traditional, time-honored network visualization tools, e.g., node-link diagrams, bar charts, tree diagrams, etc, remain indispensable, treemaps have direct benefit in specific areas of social network analysis—we emphasize, however, that treemap technology ought to be considered an *adjunct* to the traditional tools and *not* thought of as a replacement. In particular, treemaps likely are a best-fit if used during exploratory data analysis, particularly when examining clustering models—*cluster models* are settings where nodes are partitioned into groups based on specific criteria then the network studied across or within the groups.

- **Treemaps in the Social Network Setting**

  Among the many beneficial traits of treemap technology, five characteristics are directly relatable to social network analysis; treemaps: (a) offer a visual representation of quantitative features of a clustering model, (b) show the hierarchical aspects of a clustering model, (c) provide for interactive exploration of the hierarchy of a clustering model, (d) make full use of available display space, and (e) are unaffected by the size of the underlying network. On there own, these characteristics are not unique, but a treemap uniquely combines all of these benefits into a single tool, thus its attractiveness to analysts.

  While these five characteristics of treemaps are not unique to treemaps, only treemaps combine all of these characteristics into a single tool. These characteristics are particular beneficial when exploring subgroups in a network.

  Even having not so large social networks to visualize in our project, we have chosen squarified treemap for representing our data. Let's take a look to what Mark Bruls, Kees Huizing, and Jarke J. vanWijk have done on its paper: *Squarified Treemaps.*

- **Squarification**

  Howcan we tesselate a rectangle recursively into rectangles, such that their aspect-ratios (e.g. max(*height/width*; *width/height*)) approach 1 as close as possible? The number of all possible tesselations is very large. This problem falls in the category of NP-hard problems.

  However, for our application we do not need the optimal solution, a good solution that can be computed in short time is required.

  Mark Bruls, Kees Huizing, and Jarke J. vanWijk have experimented with many different algorithms. In this section we present their method that (empirically) turned out to give the best results. The key idea is based on two notions. First, they do not consider the subdivision for all levels simultaneously. This leads to an explosion in computation time. Instead, they strive to produce square-like rectangles for a set of siblings, given the rectangle where they have to fit in, and apply the same method recursively. The startpoint for a next level will then be a square-like rectangle, which gives good opportunities for a good subdivision. Second, we replace the straigthforward subdivision process for a set of siblings of the standard treemap technique (width or height is given, rectangle is subdivided in one direction) by a process that is similar to the hierarchical subdivision process of the standard treemap. They present their method first with an example, followed by a description of the complete algorithm.

- **Example**

  Suppose they have a rectangle with width 6 and height 4, and furthermore suppose that this rectangle must be subdivided in seven rectangles with areas 6, 6, 4, 3, 2, 2, and 1 (*Figure III.31*). The standard treemap algorithm uses a simple approach: The rectangle is subdivided either horizontally or vertically. Thin rectangles emerge, with aspect ratios of 16 and 36, respectively.

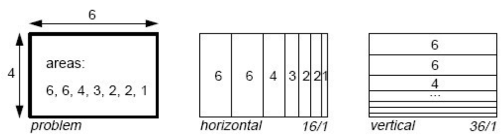


Figure 14: Subdivision problem

  The first step of their algorithm is to split the initial rectangle. They choose for a horizontal subdivision, because the original rectangle is wider than high. They next fill the left half. First they add a single rectangle (*Figure III.32*). The aspect ratio of this first rectangle is 8/3. Next we add a second rectangle, above the first. The aspect ratios improve to 3/2. However, if they add the next (area 4) above these original rectangles, the aspect ratio of this rectangle is 4/1. Therefore, they decide that they have reached an optimum for the left half in step two, and start processing the right half.

  The initial subdivision they choose here is vertical, because the rectangle is higher than wide. In step 4 they add the rectangle with area 4, followed by the rectangle with area 3 in step 5. The aspect ratio decreases. Addition of the next (area 2) however does not improve the result, so they accept the result of step 5, and start to fill the right top partition.

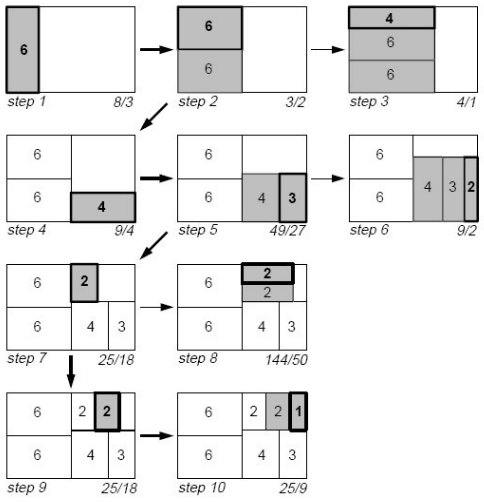


Figure 15: Subdivision algorithm

These steps are repeated until all rectangles have been processed. Again, an optimal result can not be guaranteed, and counterexamples can be set up. The order in which the rectangles are processed is important. They found that a decreasing order usually gives the best results. The initially large rectangle is then filled in first with the larger sub-rectangles.

- **Treemap of Xpertum**

  In our project, we apply the squarified treemaps technique over our affiliations and our persons using 2 steps, the areas and the elements.

  At the first step, we apply the treemap algorithm to areas (including person area if it's necessary) to show how relevant is the group of elements that are included.

  For the second step, for each treemap element that we obtained at the first step, we apply the treemap algorithm for each element to show how relevant is an affiliation or a person.

  To widen the use of the algorithm we have used it in two different variants.

(a) Getting the element that have connectors.

(b) Getting the elements that have only connectors that connect to the element and to other elements. So, basically, this option only selects elements which are related to connector elements.

In addition to the rectangles shown in the layout, when passing over a rectangle we show the relevant data associated to it. It means, in (a), it shows which elements are connected to it, and in (b), in addition to elements connected, it shows, for each elements which others they are related with.

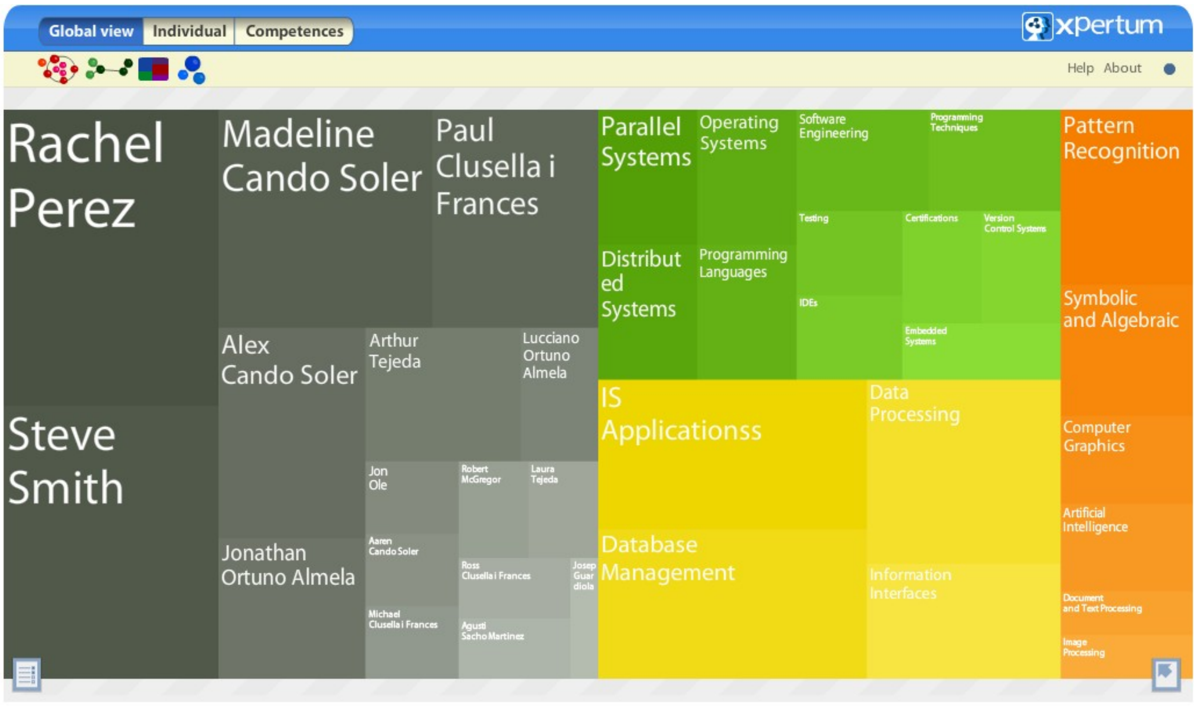


Figure 16: Xpertum treemap

2.2 Architecture

The task of display visualization of a social network involves the power of compute to display useful graph visualisations to the users.

Actually, the world of computers is experimenting a transformation. The people use nettops, netbooks, PDAs, mobiles, etc. like a computer to access the net get all the information that require to work everywhere.

These devices have different specifications between them, and the great majority haven't enough process power to execute the complex algorithms that requires the graph visualizations. Moreover, the screens of these devices offers different sizes and someone offers too new technologies to interact with them like touching screens.

On the other hand, the access to the net is very common and every the bandwidth is being improved.

For that reason we decided use a distributed architecture to design Xpertum.

2.2.1 Distributed System

A distributed system is a collection of computers physically separated and connects by a communication net. Every machine have its hardware and its software and the users, when use these have to perceive like a unique system.

The distributed systems born to tackle complex computational problems for a single computer and offer a multi-user environment.

Thanks to this, we will can release the user device of the graph computes and take this opportunity of offer a user interface appropriate to the screen device. In this way, using a server to compute the visualizations and the net like the channel between the devices and this one, we will obtain a visualization system with the opportunity to arrive to every user with net access.

2.2.2 User side

Thanks to distributed system philosophy the user side application can be implemented in any device and technology with net access.

The finally objective of the Xpertum is offers a mashup to embed the users to website and the possibility of use the application in his browsers.

For that reason, we decided use flash technology and implements the mashup in Actionscript 3, because offers a object oriented language, advanced visualizations techniques and the plugin required to display it in a web browser is in the great majority of the computers.

2.2.3 Server Side

For the server side of the application, we need a flexibility environment to implement the heart of Xpertum. For that reason, the java sevlet technology is very appropriate.

Thanks to this, we can use the Jung framework to compute the graph visualizations. Jung is an opensource project implemented in Java that groups graph visualizations algorithms and tools to work with graph.

Actually, it's the unique framework to offers these services and includes all the tools that we need to satisfy the requirements of the project.

2.2.4 Communication

For the communication between user side and server side, we decided use XML, concretely, the standard GraphML.

XML (Extensible Markup Language) is an extensible metalanguage developed by World Wide Web Consortium (W3C) and offers a comprehensible environment for the human mind to work with graph data without the help of another programs.

GraphML is a comprehensive and easy-to-use file format for graphs based on XML. Thanks to this, we can define all graphs in easy structure. Easy not only for humans, for software too and thanks to this, we can customize the content of files to reduce the size to improve the communications.

3 Prototype Documentation

3.1 Introduction

In this chapter, we'll define the design of the Xpertum system, mention the requirements of the system that Xpertum will achieve. Next, using these requirements we will present the use cases and then, the sequential diagrams that describe the functionality of these use cases.

At the end, you will can see a complete system walkthrough to enjoy our application and a evaluation that we realized to try the application with users.

This chapter is the guide to implement the Xpertum application. For that reason, we use Unified Modeller Language (UML) that offers all the notations that we need to design the system and because has been accepted as a standard through the industry and the university.

3.2 Requirements

3.2.1 Purpose of Xpertum

Xpertum will be used as a supporting tool for evaluate social networks between persons and knowledge areas. The objective of the application is improve the selection teams process in the organizations and an easy way to see the knowledge resource of the organization through visualizations.

Xpertum provides all functionalities to visualize the social network with different layouts and filter the source data like a mashup or like an full screen application. All information is accessible using a simple and minimalist user interface based on mouse events.

3.2.2 Functional Requirements

In this section we provide the list of functional requirements of Xpertum application.

3.2.2.1 Display Visualizations using Geometric Layouts

The Xpertum application should provides to users the possibility of display a relationship visualization between persons and affiliations using an algorithm based on geometric rules to distribute the elements in the view port.

3.2.2.2 Display Visualizations using Force Layouts

The Xpertum application should provides to users the possibility of display a relationship visualization between persons and affiliations using an algorithm based on force rules to distribute the elements in the view port.

3.2.2.3 Display Visualizations using Clustering Layouts

The Xpertum application should provides to users the possibility of display a relationship visualization between persons and affiliations using a clustering algorithm to distribute and group the elements in the view port.

3.2.2.4 Display Visualizations using Treemap Layouts

The Xpertum application should provides to users the possibility of display a relationship visualization between persons and affiliations using an hierarchical visualization, the treemap.

3.2.2.5 Filter Visualizations by elements types

The users should can change the elements types that are shown by the application. They should can try to show only persons, only affiliations or both at the same time.

3.2.2.6 Filter Visualizations by knowledge areas

The users should can change the knowledge areas that are shown by the application. They should can select anything number of areas (at least one) and then, the affiliations that Xpertum display will belong of the area selection and the persons without relationships between these areas will not show.

3.2.2.7 Customize resolution

Xpertum should can be showed on net browser with any type of view port resolution, to adjust the size of application with their functional context (mashup, net application, ..) and the user can turn on/off a full screen.

3.2.2.8 External Data Source

The users should can use external data to be shown in Xpertum. The data should are sent using URLs that containing XML files in GraphML format.

3.2.2.9 Visualization size

Xpertum should can display any type of his visualization with 500 elements at most.

3.2.3 Non-Functional Requirements

For the categorization of non-functional requirements we follow the FURPS+ model used by the Unified Software Development Process. The FURPS+ model makes a distinction between non-functional requirements that are quality requirements and those that are constraints.

3.2.3.1 Quality Requirements

3.2.3.1.1 Performance

One major requirement is to efficiently display visualizations. Xpertum is a distributed system in net, therefore the server must serve efficiently the visualization data because the user mustn't perceive any retard by Xpertum to realize operations and should not face any performance problems while navigating in the application.

3.2.3.1.2 Portability

Another requirement of the application is that should be easily deployed on standard servers without difficult to configure the software.

3.2.3.1.3 Extensibility

Xpertum user interface provides to user only 4 algorithms to display visualizations but the Xpertum domain can offer 8 algorithms and the possibility of, in future research, add new algorithms to display graph visualizations without change the communication between the user side and the server side and without difficult to realize the improvement.

3.2.3.1.4 Adaptability

The Xpertum application has to show high adaptability with regards to modifications and extensions of software engineering domain concepts and structures. We use standards to data communication and data source for guarantee an easily develop of communication interfaces between Xpertum and external databases.

3.2.3.1.5 Flexibility

Xpertum should have the ability to change the behaviour against various environment interactions and it should work with different conceptual representations.

3.2.3.2 Constraints

3.2.3.2.1 Interface Requirements

Xpertum should be easily configured to match the corporate look and integrated at any website with any resolution.

3.2.3.2.2 Implementation

All users should be able to access Xpertum with a web browser supporting Flash 9.

3.3 Functional Architecture

3.3.1 Use cases

Thanks to the previous chapter, we can describe in this section the use cases that Xpertum should instantiate. The main use case diagram is illustrated in figure 17:

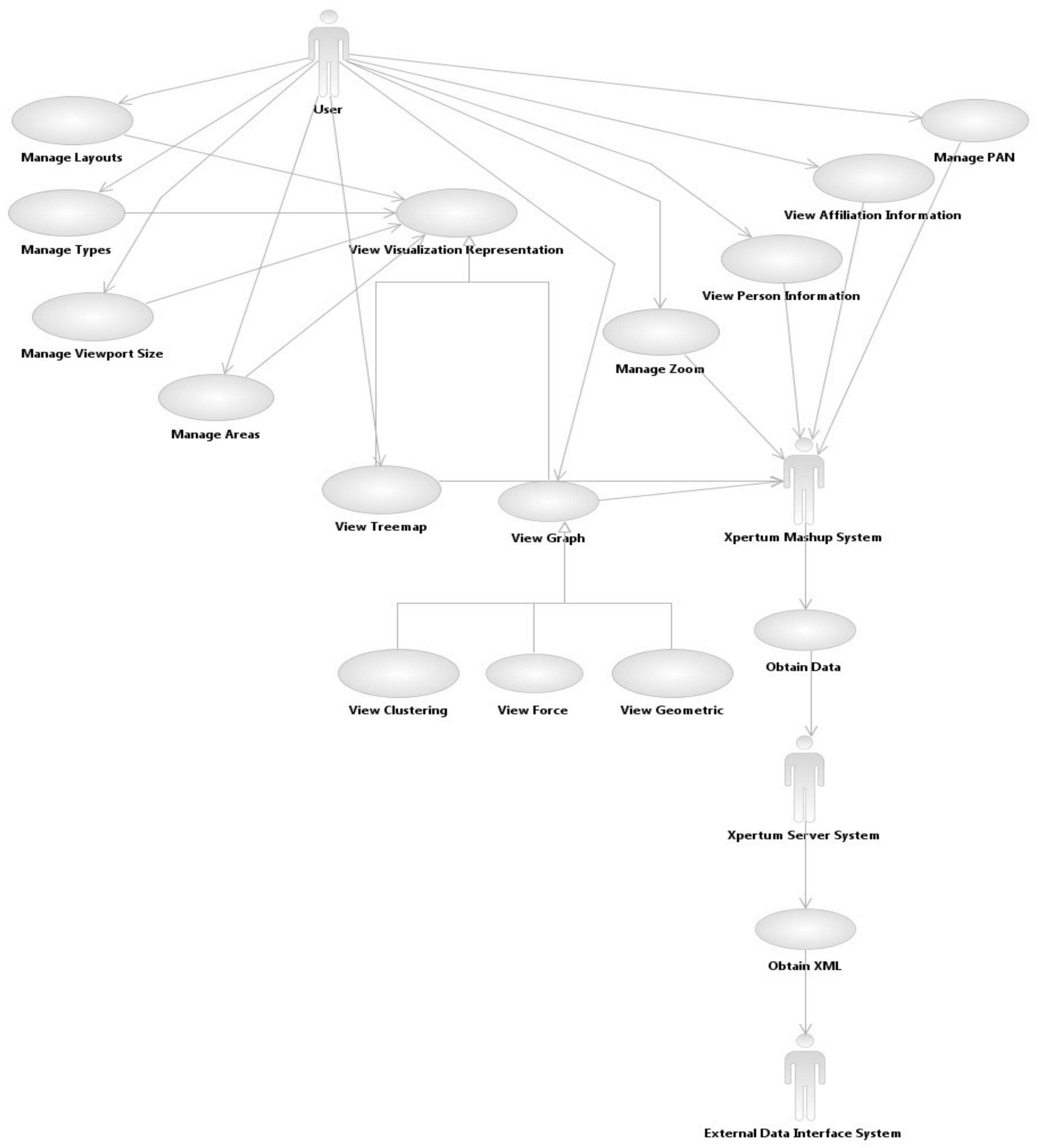


Figure 17: Main Use case diagram

3.3.1.1 Manage PAN

If the visualization area is better than the view port, the user can scroll the visualization making a PAN. When the interaction between the user and the application is activated the user make the PAN using the mouse device. The application recalculate the visualization coordinates in real-time and displays the new visualization area.

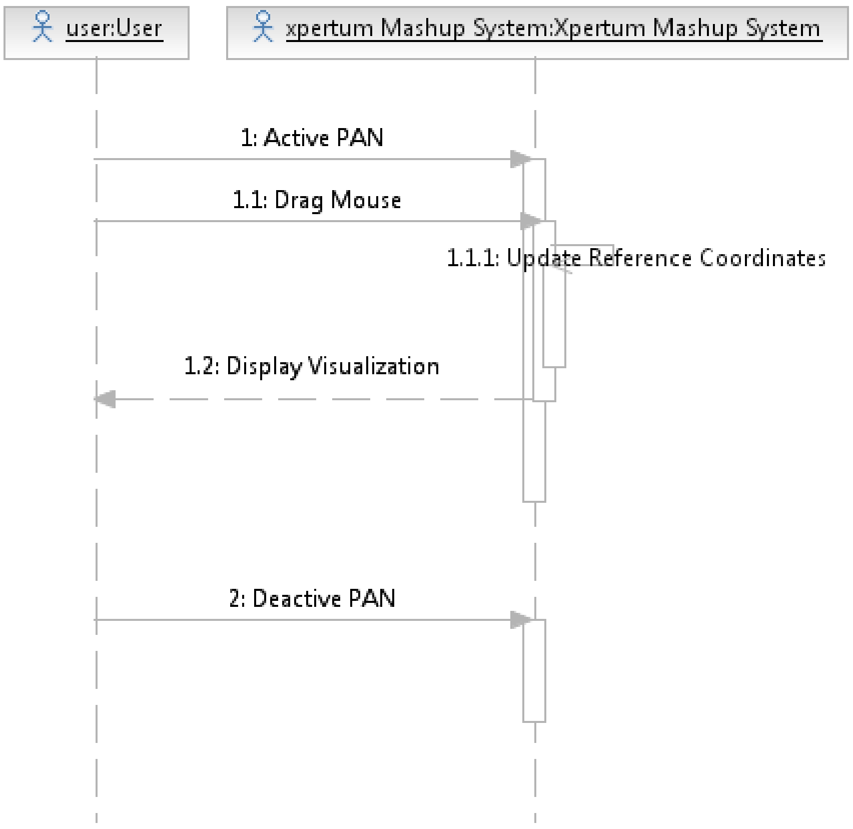

user:User
xpertum Mashup System:Xpertum Mashup System
1: Active PAN
1.1: Drag Mouse
1.1.1: Update Reference Coordinates
1.2: Display Visualization
2: Deactive PAN

Figure 18: Manage Pan basic flow

3.3.1.2 View treemap

The user request a treemap visualization and then Xpertum Mashup using the current parameters configuration of the visualization, compute the treemap visualization and display it to user.

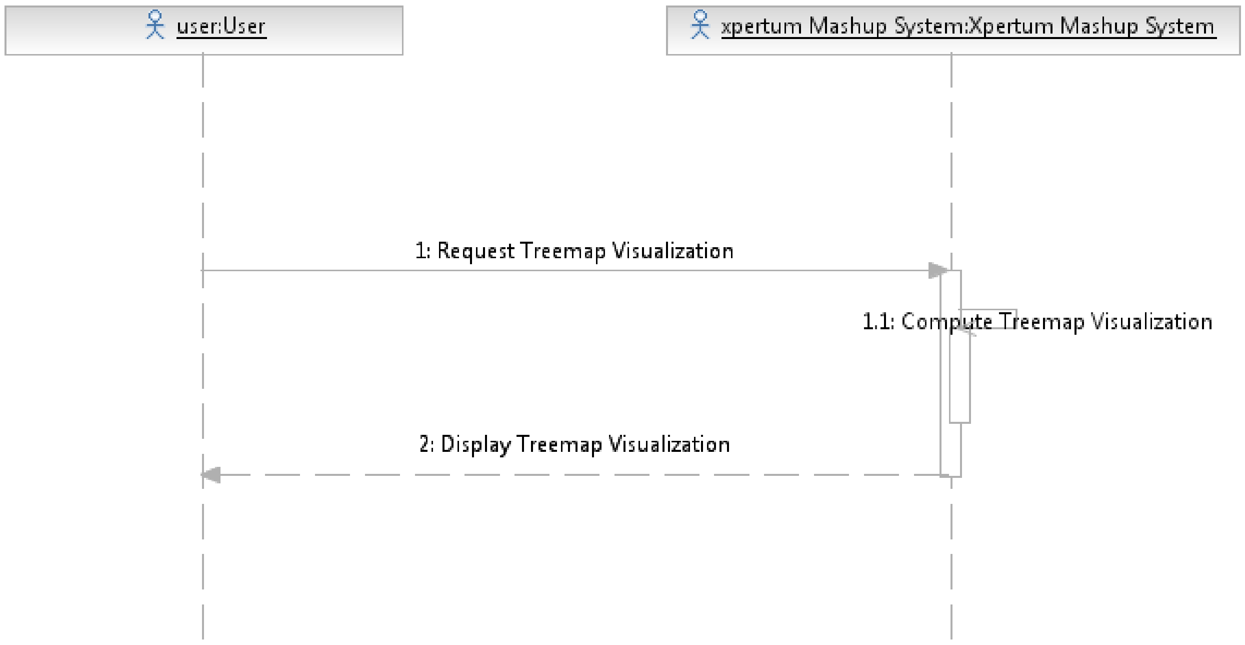

user:User
xpertum Mashup System:Xpertum Mashup System
1: Request Treemap Visualization
1.1: Compute Treemap Visualization
2: Display Treemap Visualization

Figure 19: View treemap basic flow

3.3.1.3 View graph

The user request a graph visualization, that can be based on geometric, force or clustering algorithms. Then Xpertum Mashup receive the user's request and repeats to Xpertum Server, that is the responsible of the compute process with the current parameters state.

The Xpertum Server computes the new layout with the request parameters and then write the response in a XML, using the standard GraphML. When Xpertum Mashup finished downloading the information, displays to user the new visualization that was required.

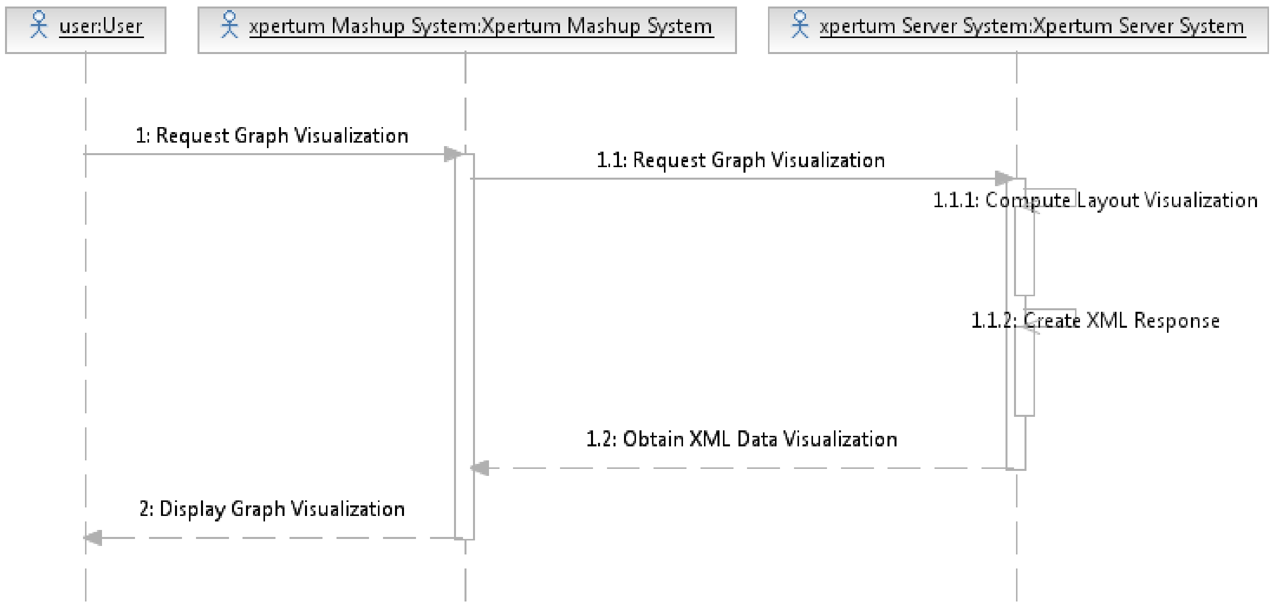

user:User
xpertum Mashup System:Xpertum Mashup System
xpertum Server System:Xpertum Server System
1: Request Graph Visualization
1.1: Request Graph Visualization
1.1.1: Compute Layout Visualization
1.1.2: Create XML Response
1.2: Obtain XML Data Visualization
2: Display Graph Visualization

Figure 20: View graph basic flow

3.3.1.4 View Person Information

The user request a person element information to Xpertum Mashup. The persons in Xpertum Mashup are represented by circles for graph visualizations and for rectangles in treemap visualization. The users click using the mouse device over a person element and then, the Xpertum Mashup shows to user a pop-up with all the information of the person.

This pop-up contains:

- complete name
- e-mail
- job role
- list of affiliations related with the person (for each affiliation appears too the strength force of the relationship)

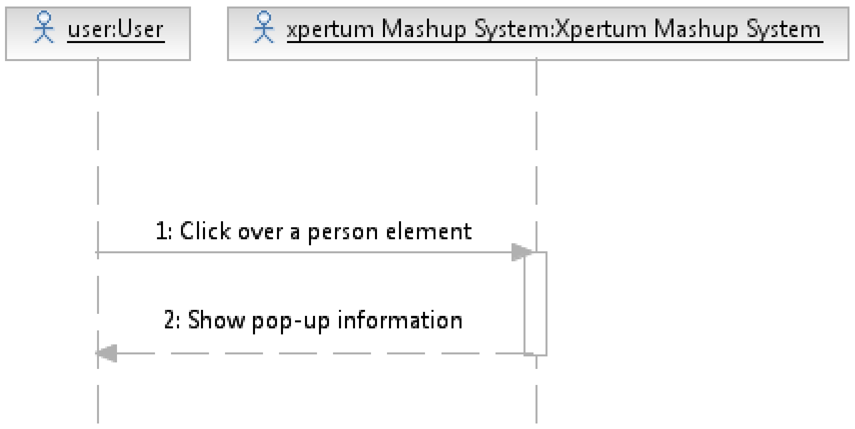

user:User
xpertum Mashup System:Xpertum Mashup System
1: Click over a person element
2: Show pop-up information

Figure 21: View person information basic flow

3.3.1.5 View Affiliation Information

The user request an affiliation element information to Xpertum Mashup. The affiliations in Xpertum Mashup are represented by circles for graph visualizations and for rectangles in treemap visualization. The users click using the mouse device over an affiliation element and then, the Xpertum Mashup shows to user a pop-up with all the information of the affiliation.

This pop-up contains:

- name
- list of persons related with the affiliation (for each person appears too the strength force of the relationship)

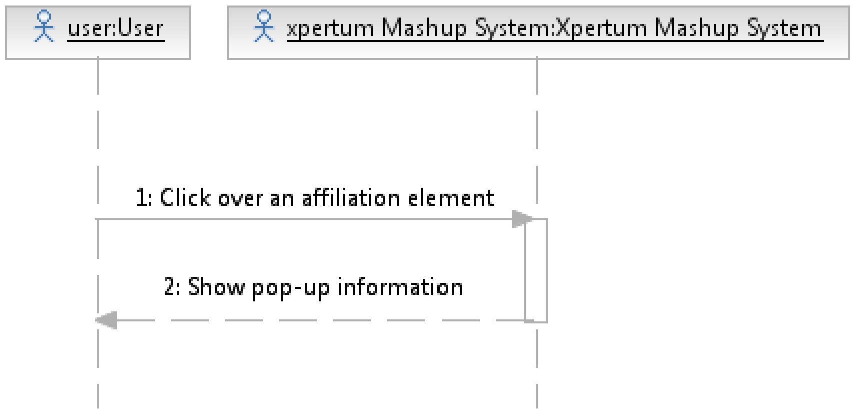

user:User
xpertum Mashup System:Xpertum Mashup System
1: Click over an affiliation element
2: Show pop-up information

Figure 22: View affiliation information basic flow

3.3.1.6 Manage Zoom

The user request make a zoom to approach or move away the visualization. When Xpertum Mashup receives one of these petitions, increase or decrease the visualization and then, update the reference coordinates of the visualization to display the same area with the change of size.

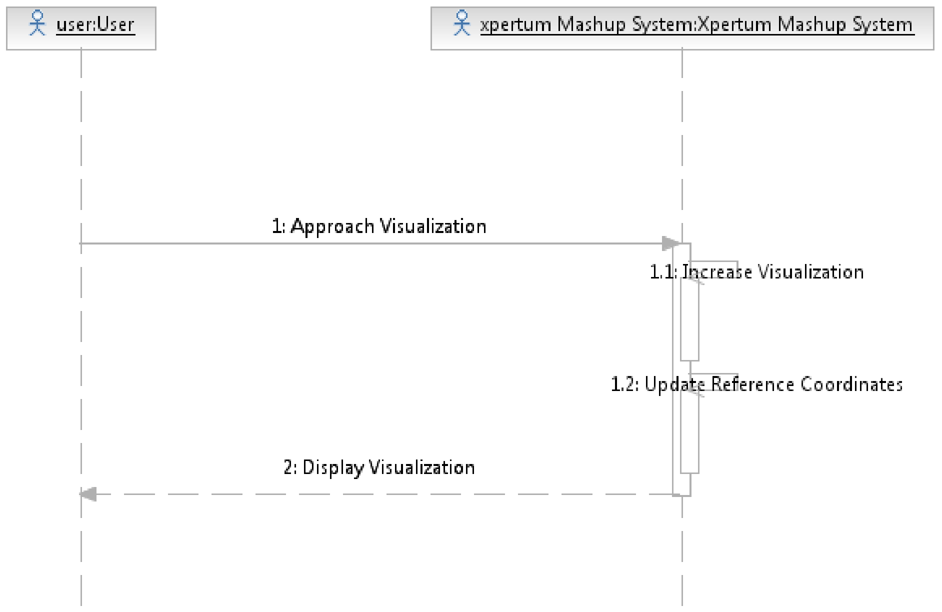

user:User
xpertum Mashup System:Xpertum Mashup System
1: Approach Visualization
1.1: Increase Visualization
1.2: Update Reference Coordinates
2: Display Visualization

Figure 23: Manage Zoom In basic flow

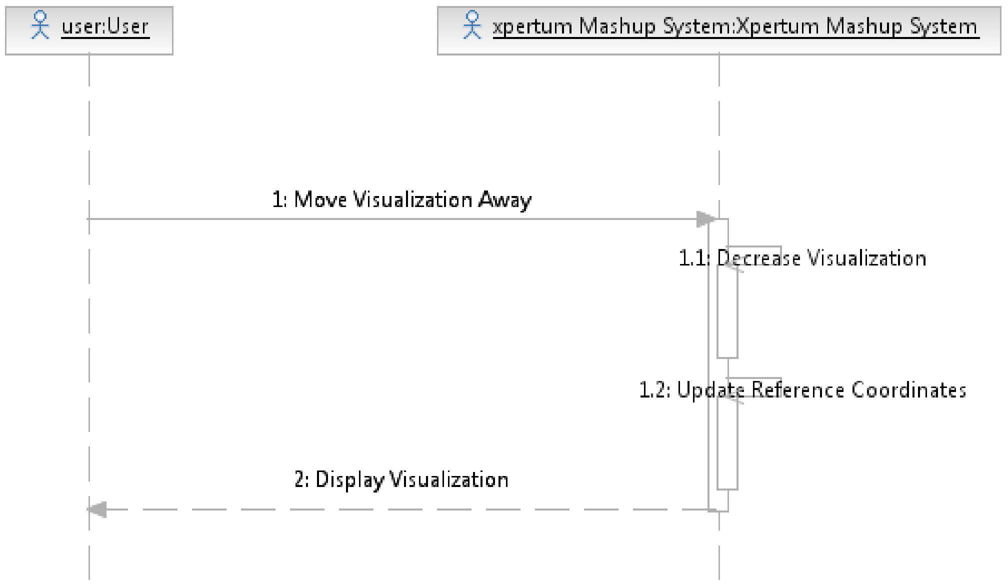

user:User
xpertum Mashup System:Xpertum Mashup System
1: Move Visualization Away
1.1: Decrease Visualization
1.2: Update Reference Coordinates
2: Display Visualization

Figure 24: Manage Zoom Out basic flow

3.3.1.7 Obtain Data

When Xpertum Mashup is initialized requests the data to the Xpertum Server. Xpertum Server receive an URL (***U**niform **R**esource **L**ocator),* that was provided by the Xpertum Mashup or by the user, that contains the data in GraphML format.

Once received, Xpertum Server download the data and store it a copy for the future visualization requests and for avoid that, if the source are an interface data connected to a database, if the information changes in the source, don't break the consistency with the user Xpertum session.

Then, when the copy has completed, the server system leaks the necessary data to Xpertum Mashup and writes the XML response that he will read.

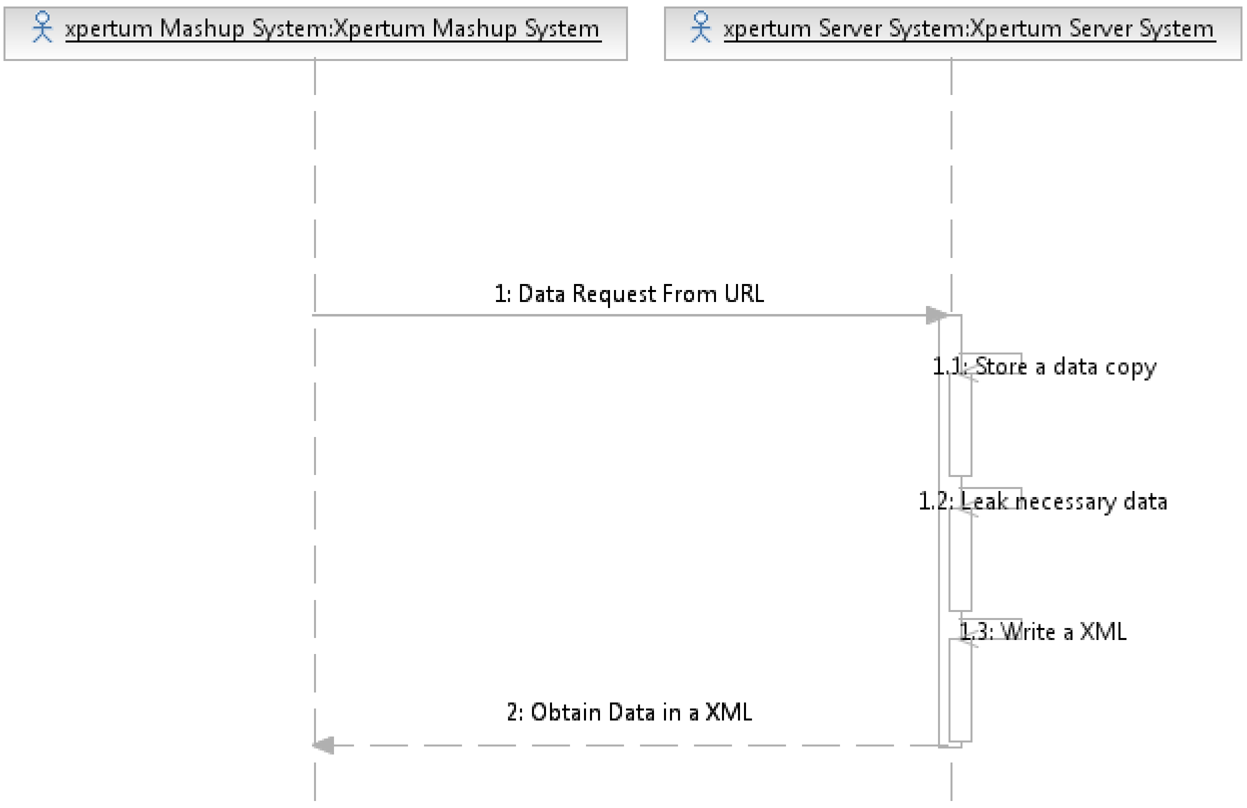


Figure 25: Obtain data basic flow

3.3.1.8 Obtain XML

The Xpertum Server requests the XML file (in GraphML format) to an external Data Interface using a URL. These interface provides to Xpertum Server the XML with the data for begin the session.

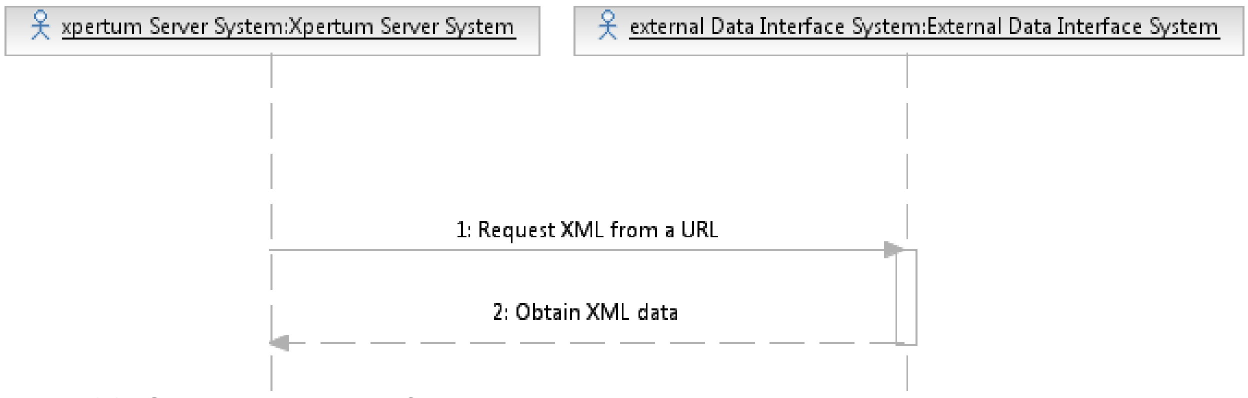

xpertum Server System:Xpertum Server System
external Data Interface System:External Data Interface System
1: Request XML from a URL
2: Obtain XML data

Figure 26: Obtain XML basic flow

3.3.1.9 Manage Areas

The users can change the areas that Xpertum Mashup visualizes. The user sends a selection of areas to Xpertum Mashup and these request a new visualization representation. When is ready, displays the new visualization of the selection areas to the user.

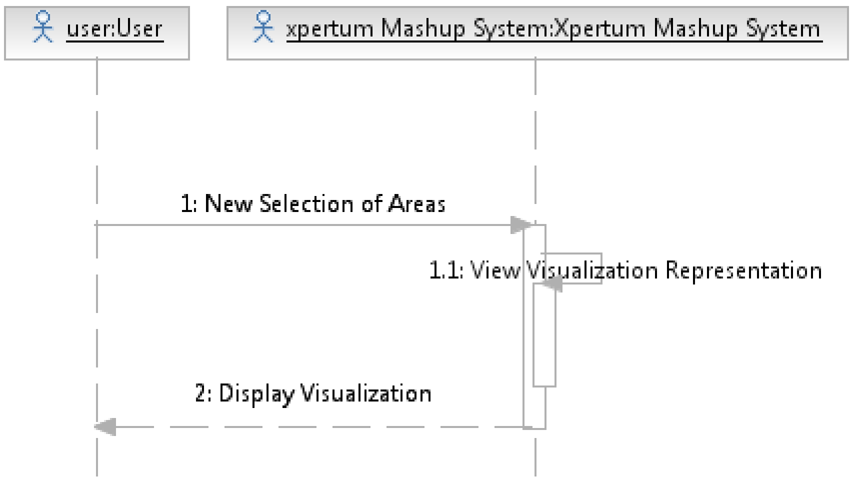

user:User
xpertum Mashup System:Xpertum Mashup System
1: New Selection of Areas
1.1: View Visualization Representation
2: Display Visualization

Figure 27: Manage Areas basic flow

3.3.1.10 Manage Types

The users can change the type that Xpertum Mashup visualizes, specifically users can select Individual type (only persons), Competences type (only affiliations) or Complete type (persons and affiliations). The user selects one of these types and Xpertum Mashup sends a request of a new visualization representation. When is ready, displays the new visualization of the new type.

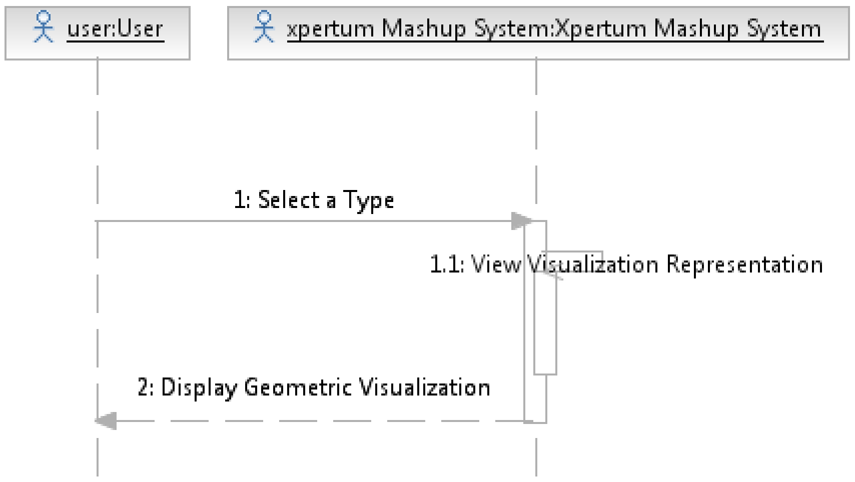

user:User
xpertum Mashup System:Xpertum Mashup System
1: Select a Type
1.1: View Visualization Representation
2: Display Geometric Visualization

Figure 28: Manage Types basic flow

3.3.1.11 Manage Layouts

The users can change the layout that Xpertum Mashup visualizes, specifically users can select Geometric layout (based in geometric algorithms), Force layout (based on force layout), Treemap layout (based in hierarchical visualization) or Clustering layout(based on clustering algorithms). The user selects one of these layouts and Xpertum Mashup sends a request of a new visualization representation. When is ready, displays the selected layout visualization.

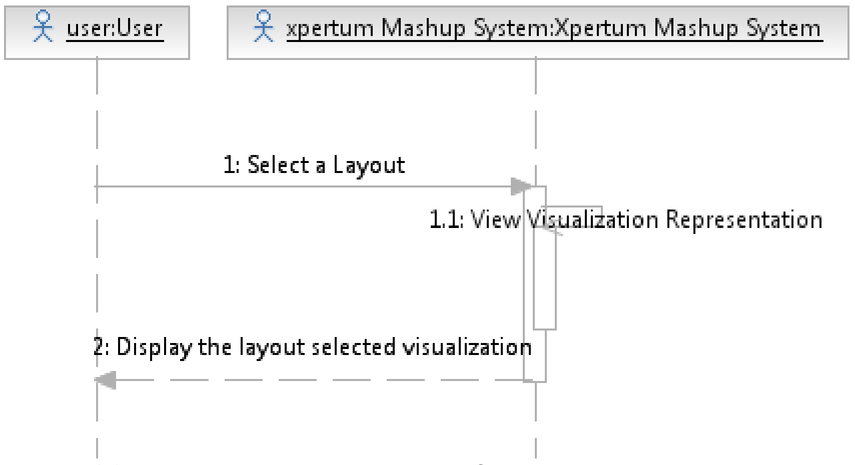

user:User
xpertum Mashup System:Xpertum Mashup System
1: Select a Layout
1.1: View Visualization Representation
2: Display the layout selected visualization

Figure 29: Manage Layouts basic flow

3.3.1.12 Manage View port size

The user changes the view port size and sends a request to Xpertum Mashup. Then, the application sends a new request to obtain a new visualization representation. When is ready, displays the re-sized visualization to user.

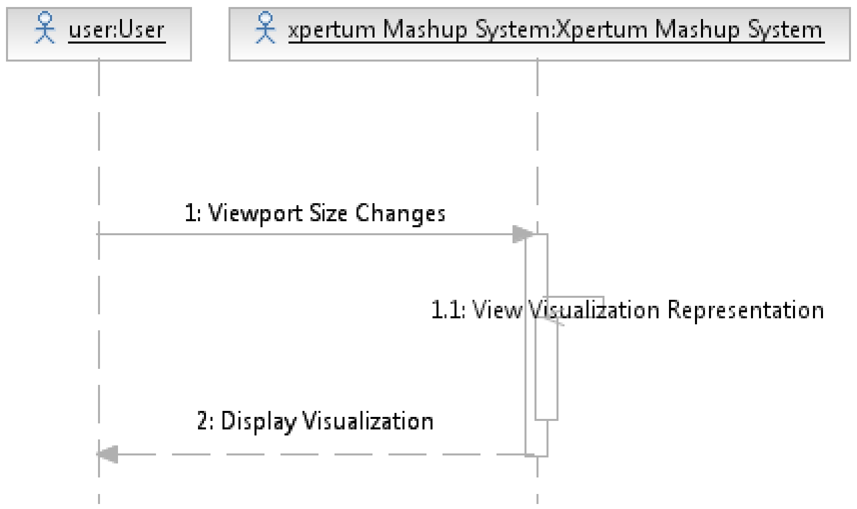

user:User
xpertum Mashup System:Xpertum Mashup System
1: Viewport Size Changes
1.1: View Visualization Representation
2: Display Visualization

Figure 30: Manage View port size basic flow

3.4 Technical Architecture

Xpertum is an application with distributed architecture in 3 independent layers.

Initially, the Xpertum project, that provides a visual representation of relationships between competence areas and people, consisted solely in a flash-based application. Thanks to this technology, we had a framework to develop our system and brought the user experience to a web browser.

But flash has inconveniences. Xpertum has become more complex and requires more versatility and to reduce the processing charge in the client side, obtaining thus more performance and a better user experience.

Under this philosophy, the team has created a distributed architecture, obtaining three independent layers: visualization, domain and data.

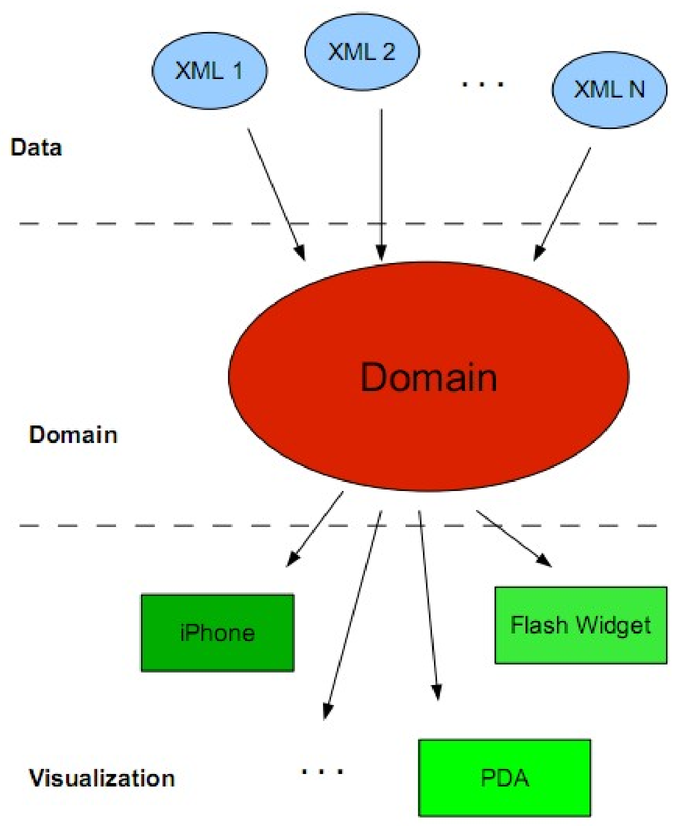


Figure 31: Architecture in 3 layers for Xpertum

With this architecture, the domain layer is responsible of the data processing from data sources that can be independent of the application and provides the required visualization. This was currently being performed entirely by a flash application, but thanks to the new architecture, the series of data sources that can provide the required data for Xpertum is unlimited, as is the environment or the devices that can act as the client of the visualization, ranging from flash widgets to iPhone apps.

The main advantage of this new architecture is the versatility and transparency of data processing while increasing the performance and resource efficiency in the user side. Another collateral advantage is that building Xpertum in separate modules offers the development team to increase the number of algorithms and visualizations available transparently to the user.

Finally, to obtain data from any external databases, we added to the architecture a new component, the Data Interface, that is the responsible of translates the external data to a XML format that the Data Manager of Xpertum can understand.

At figure 32 you can see the definitely software architecture of Xpertum.

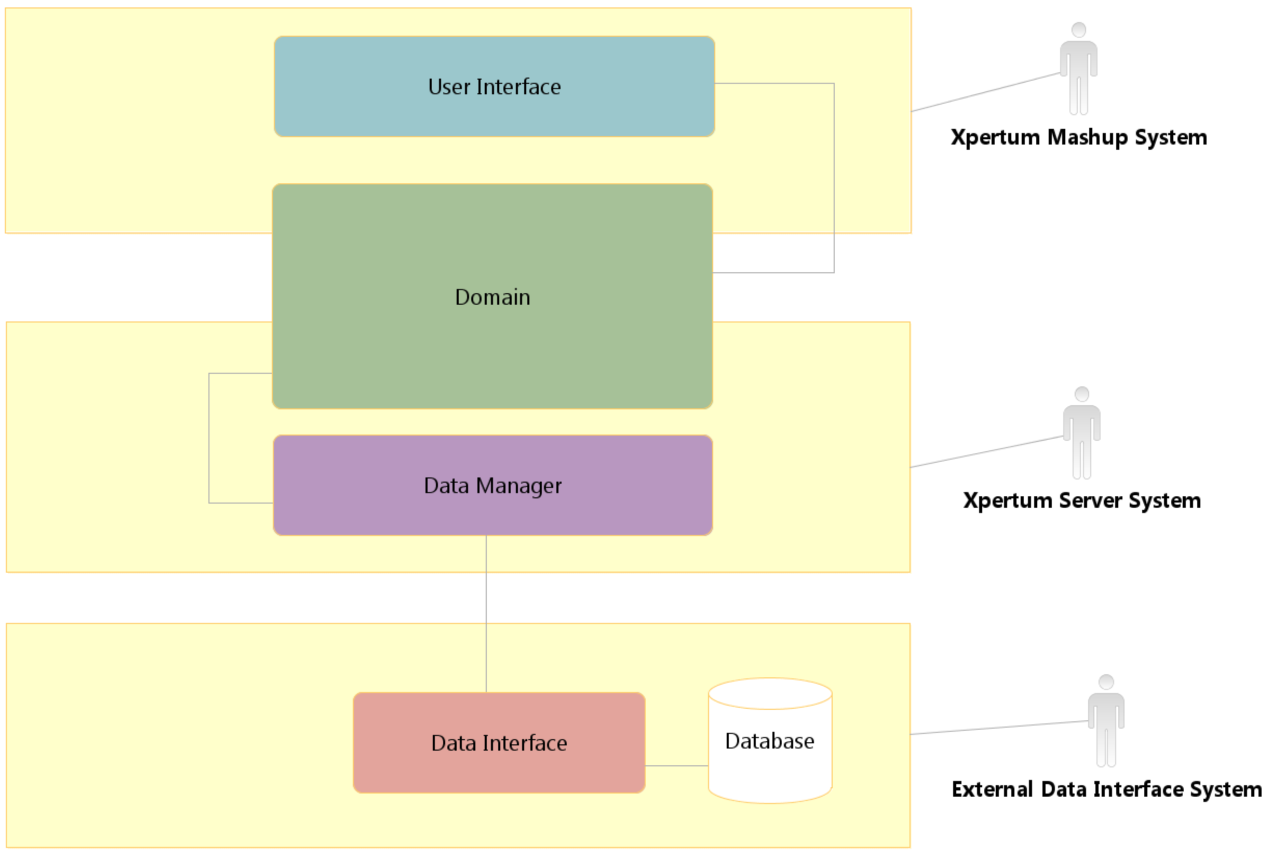


Figure 32: Xpertum Architecture

User Interface

This layer provides the user interactions with the applications and is responsible of display the visualizations to the users. The requests of new petitions of visualizations are send by this layer to the Domain.

Domain

This layer receives the petitions of User Interface layer and communicates with Data Manager when needs access to data.

Its mission is compute the visualizations that User Interfaces requires and send to it. The Graph visualizations are computed in Xpertum Server and the treemap visualizations, that not require so much computational cost, are computed in Xpertum Mashup.

Data Manager

This layer is the responsible of manage the data and obtains the XML data from external sources, files or Data Interfaces and send to Domain the data that requires.

Data Interface

It's responsible of translate the content of database petition to a XML that Data Manager can understand.

3.5 System Walkthrough

3.5.1 Manage Zoom

The Manage Zoom option allows access to 2 functionalities:

- Zoom in

  The user push the button "+" (look the red rectangle at figures 33 and 34) and then, if the zoom level doesn't arrive to the limit, the visualizations increases.

- Zoom out

  The user push the button "-" (look the red rectangle at figures 33 and 34) and then, if the zoom level doesn't arrive to the limit (like the figure 33) , the visualizations decreases.

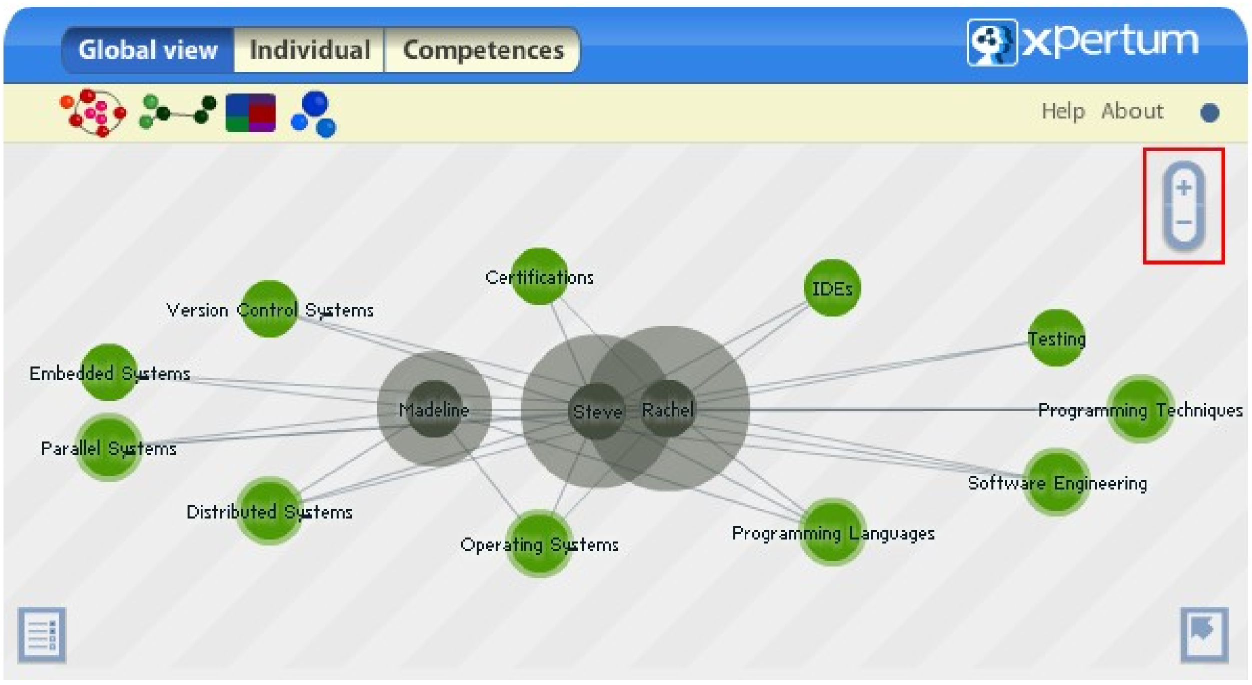


Figure 33: Zoom controls

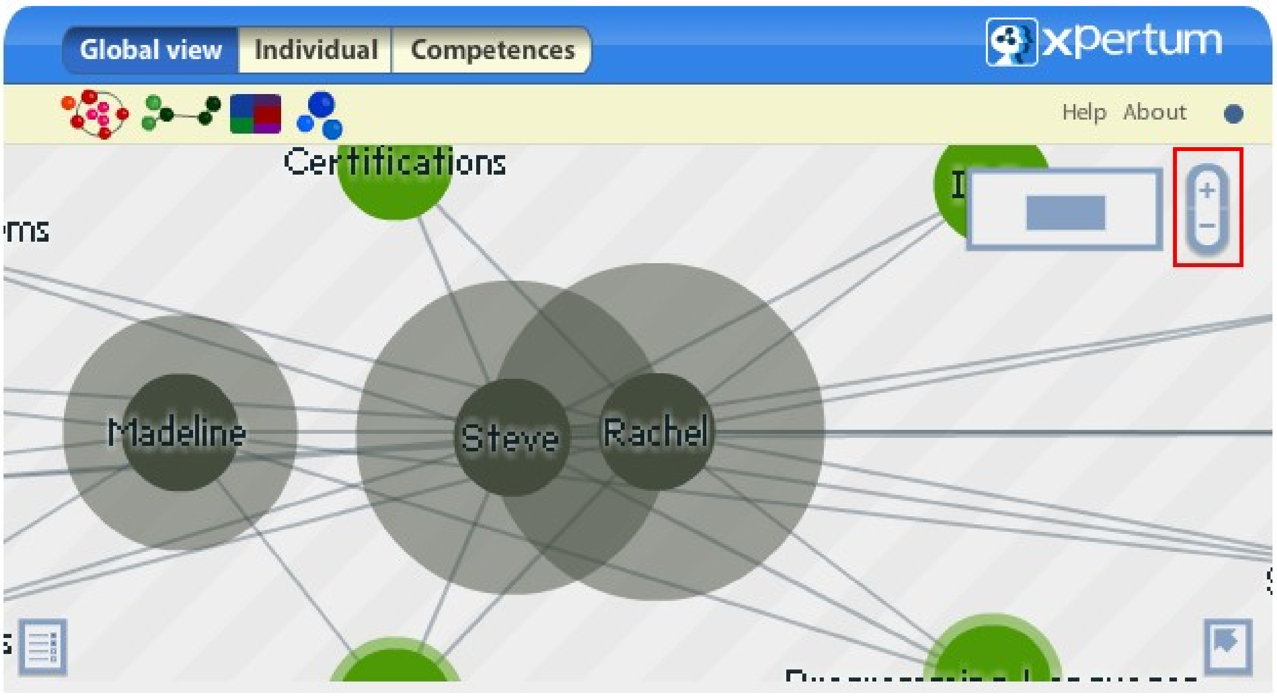


Figure 34: Zoom controls – Zoom realized

3.5.2 Manage PAN

The Manage PAN option allows to the user the possibility of move the visualization area when doesn't fit in the view port.

When this situation occurs, the PAN control appears (look figure 35) and the users can scroll the rectangle that it's into the control panel clicking with mouse device and dragging it. The movements of the little rectangle will be realized by the visualization area.

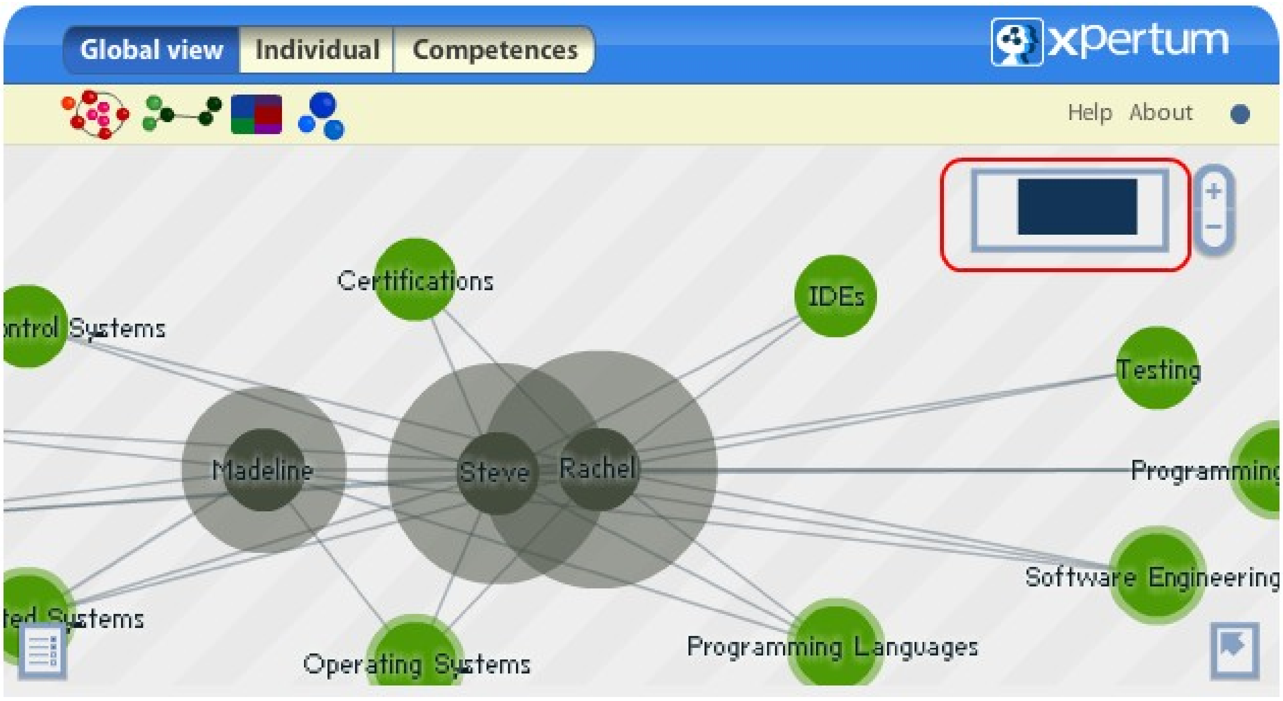


Figure 35: Pan controls

3.5.3 Full screen

The users can turn on/off the full screen mode clicking over the button that you can see at figure 36. When is active, Xpertum take up all the view port, like a desktop application.

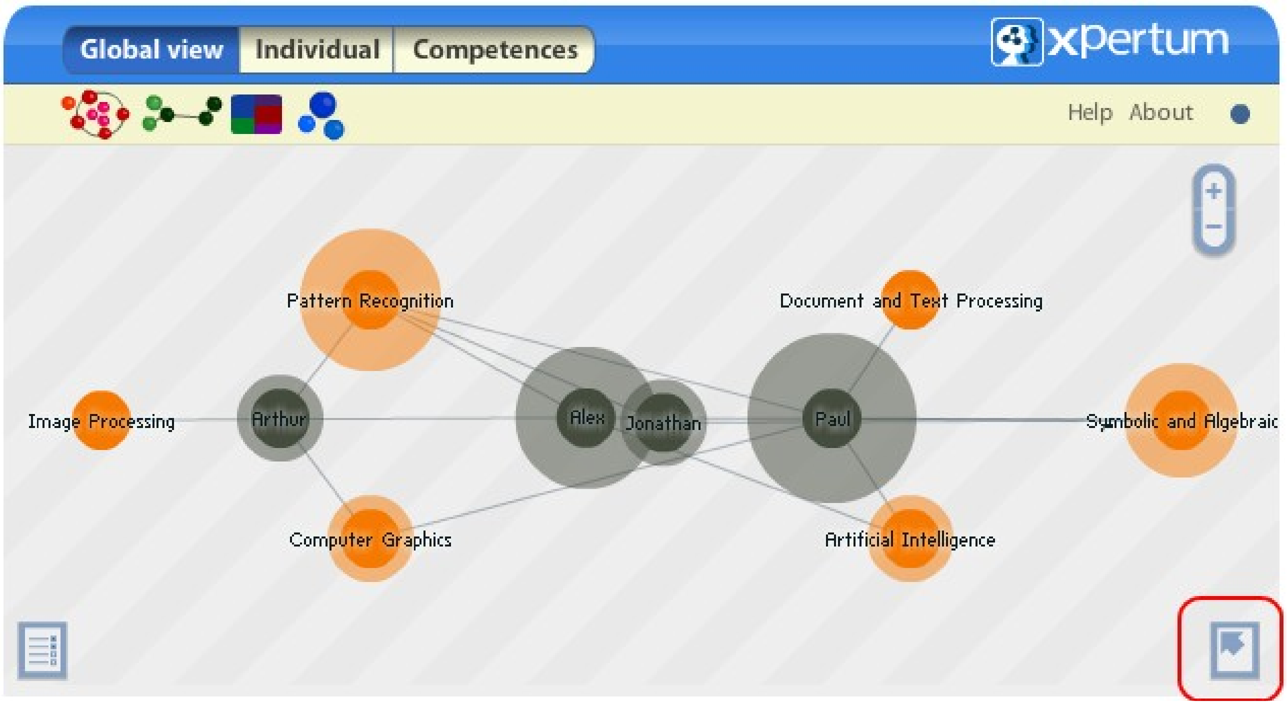


Figure 36: Full screen control

3.5.4 Manage types

The Manage Type option allows to user the functionalities to change the type of visualization to display between 3 possibilities:

- Global View: visualizations with persons and affiliations
- Individual: visualizations only with persons
- Competences: visualizations inly with affiliations

To change the type, the user must click over the buttons of figure 37. The button of the current type will change the colour for the blue. The others buttons in grey colour aren't active.

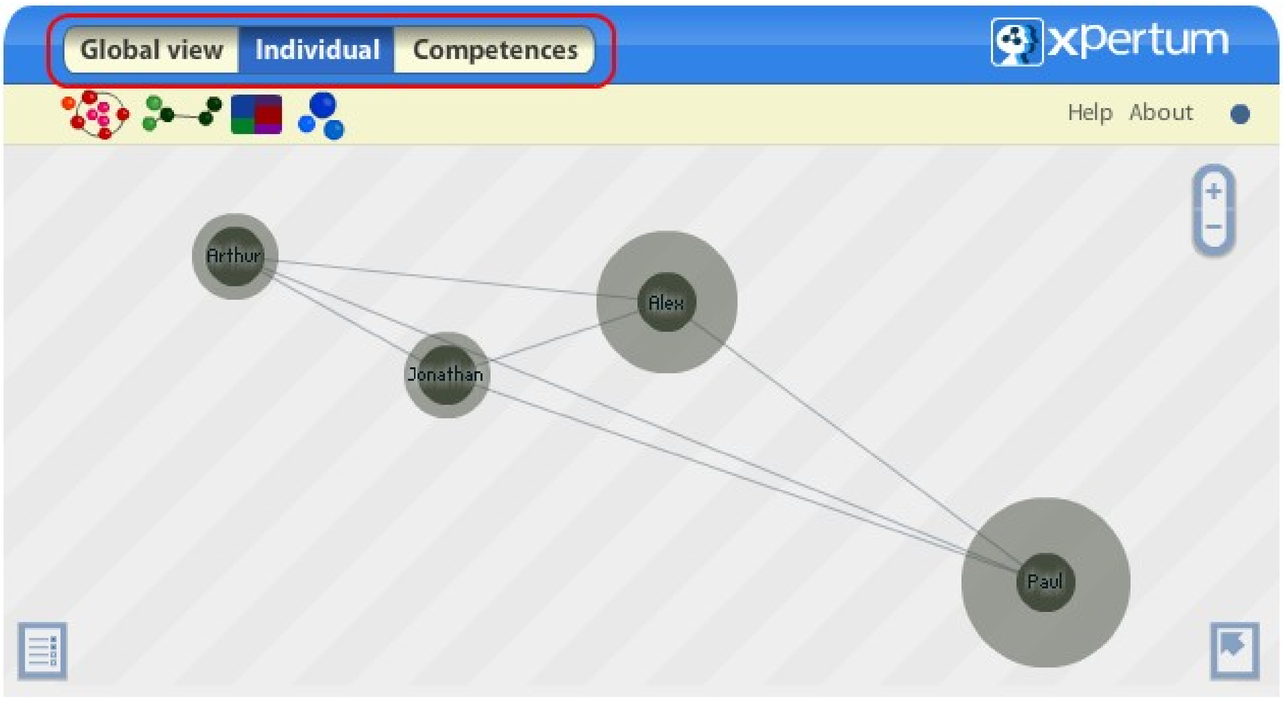


Figure 37: Manage types

3.5.5 Manage Areas

The Manage Areas option allows to the user the possibility of select the areas that the user wants to display.

This action is realized by a control panel (figure 39 and 40) that allows select All areas at the same time clicking over the button “All“ or add/delete to selection areas. To add/delete an area, the user must click over the little rectangle that is at the right of the area name. If the rectangle has the colour of the area, means that the area is selected. To select only an area to display (and unselect the others) the user must click over the area name.

To show the control panel to manage areas, the user must put the mouse cursor over the button that you can see at figure 38.

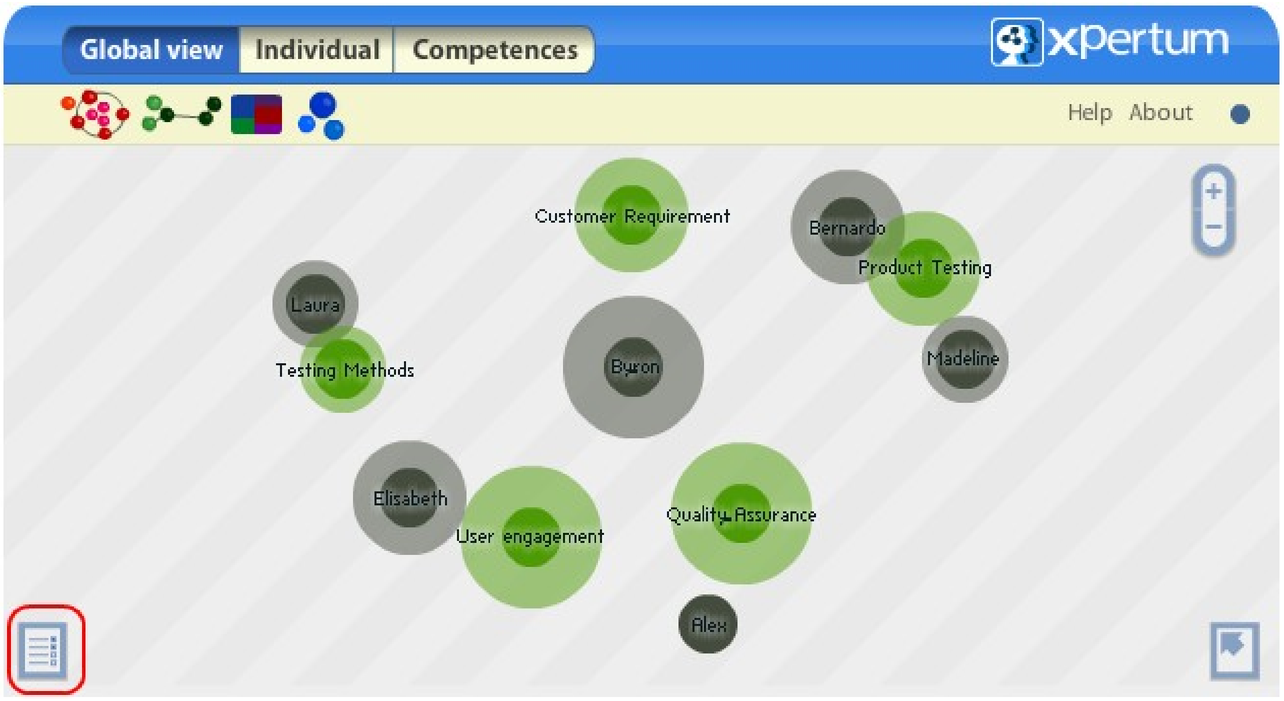


Figure 38: Button to Manage Areas

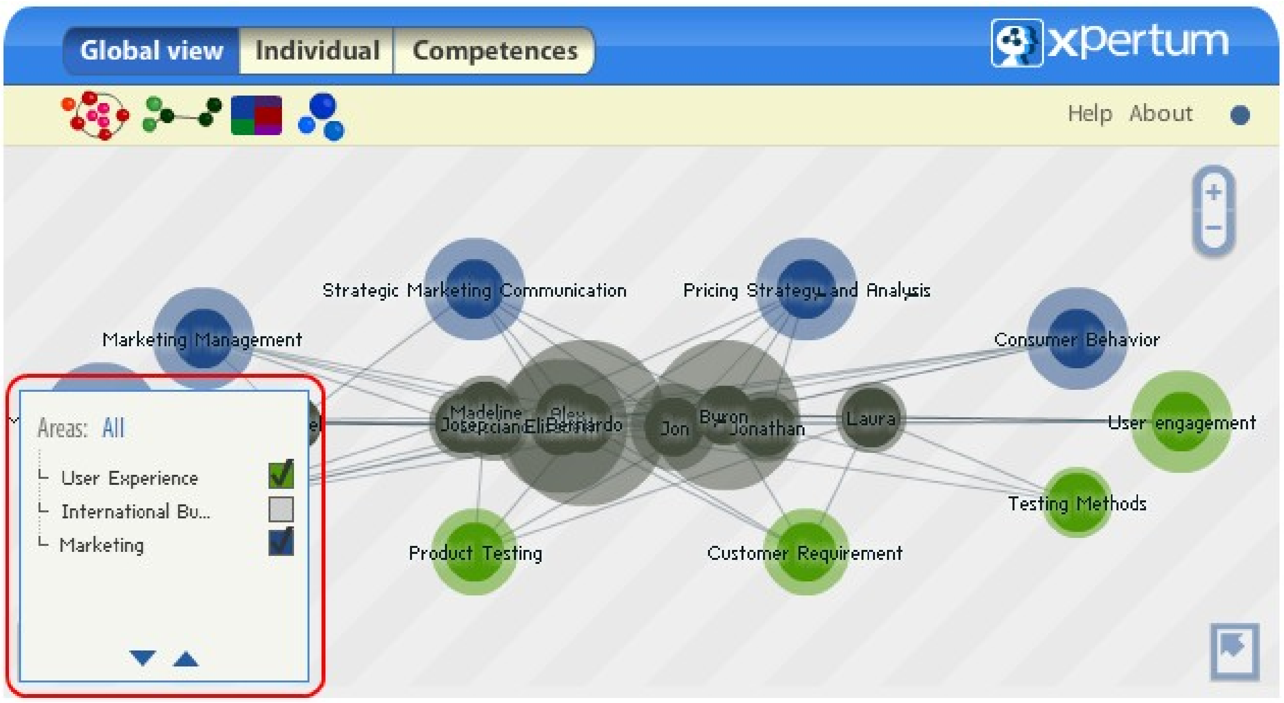


Figure 39: Control panel to Manage Areas

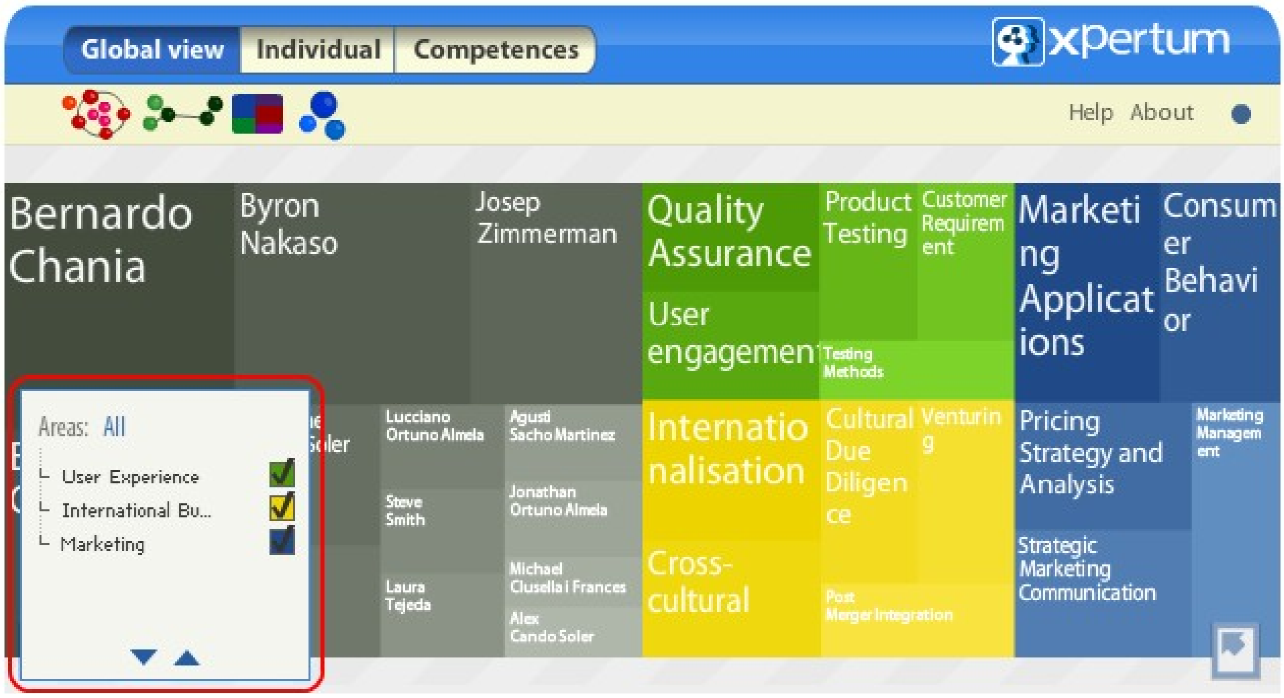


Figure 40: Control panel to Manage Areas with all areas selected

3.5.6 Manage Layouts

The Manage Layout option allows to the user the possibility of applies the 4 different visualizations algorithms that Xpertum Mashup offers to the user.

To select a new visualization using an other layout, the user only have to click over one of the buttons:

- Galaxy View: graph visualization using a geometric algorithm (figure 41)
- Big Bang View: graph visualization using a force algorithm (figure 42)
- Map View: treemap visualization (figure 43)
- Clustering: graph visualization using a clustering algorithm (figure 44)

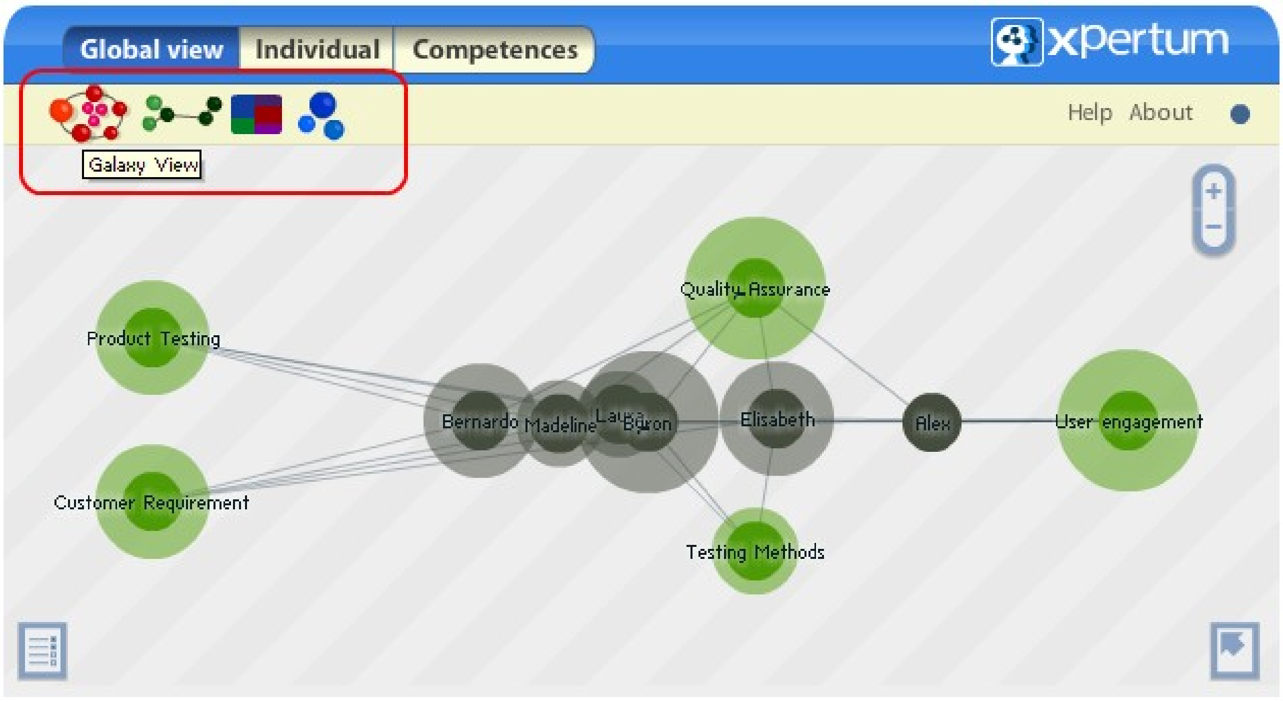


Figure 41: Galaxy view

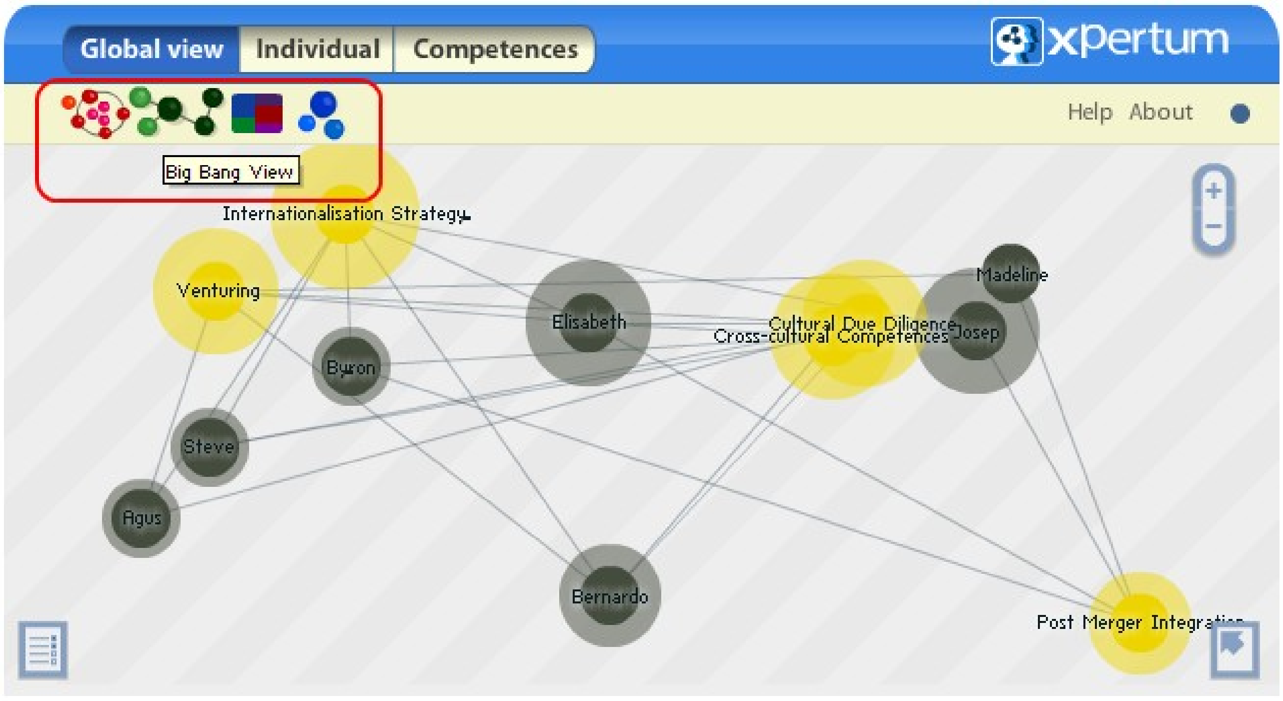


Figure 42: Big Bang view

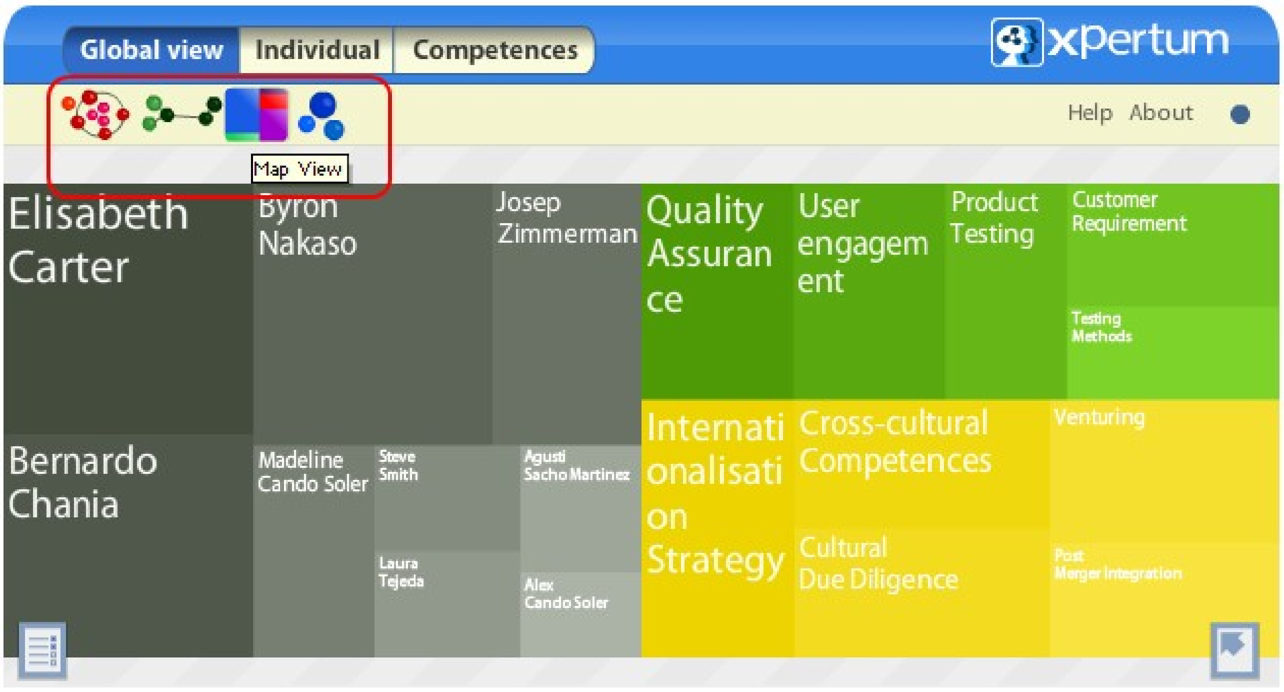


Figure 43: Map view

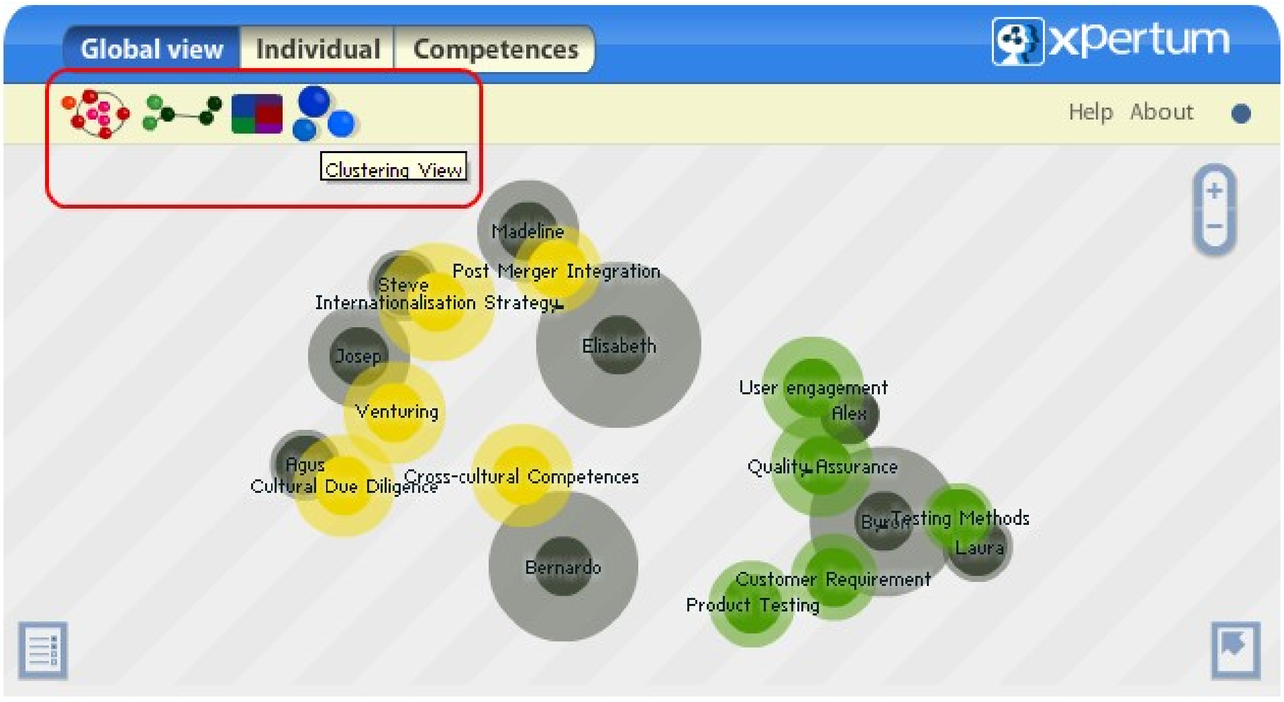


Figure 44: Clustering view

3.5.7 Download Status

Xpertum is a distributed system and Xpertum Mashup and Xpertum server exchanges a lot of data, but in small packets (is in function of the graph size). In spite of this, sometimes the data charge causes a little lag between the user request and the display of the visualization or the data download.

To show the current status to the user, the user interface of Xpertum Mashup offers a little icon that plays an animation when the application is downloading data (figure 45).

On the other hand, when the application is been downloading to the browser, the applications displays the figure 46.

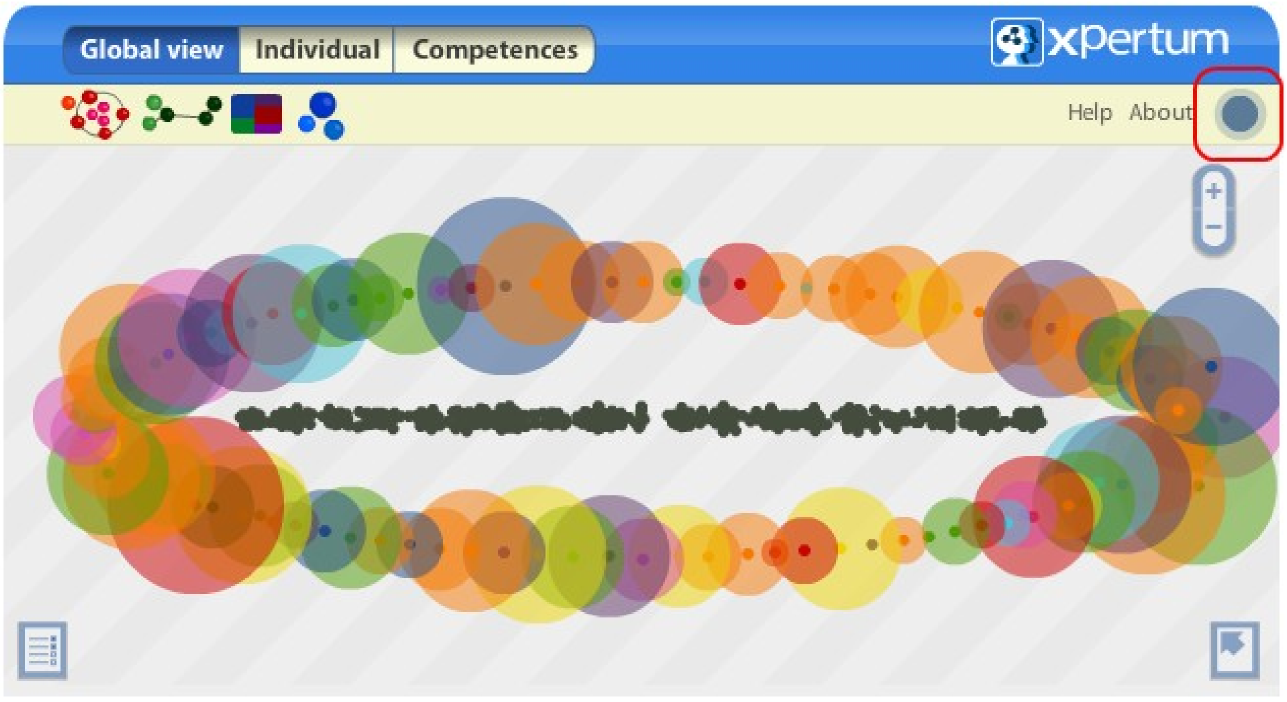


Figure 45: Downloading a new visualization

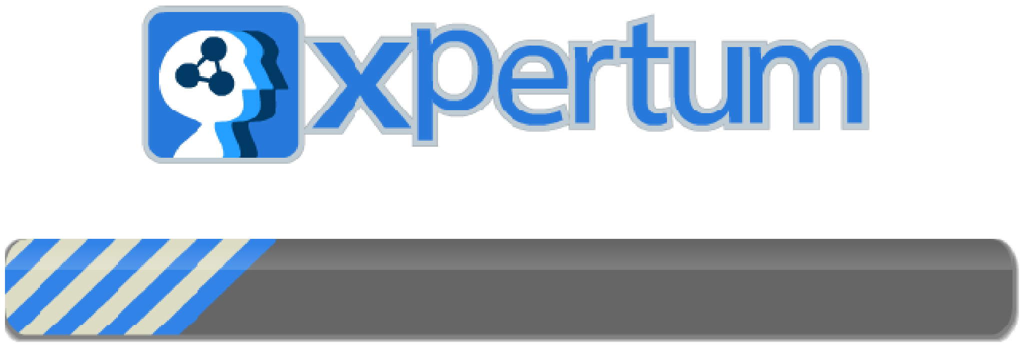


Figure 46: Downloading Xpertum Mashup

3.5.8 Provide Data Source

When the user wants to change the data source, he/she has to add to the URL at the net browser the word"?object=" and the URL of the new data source (figure 47).

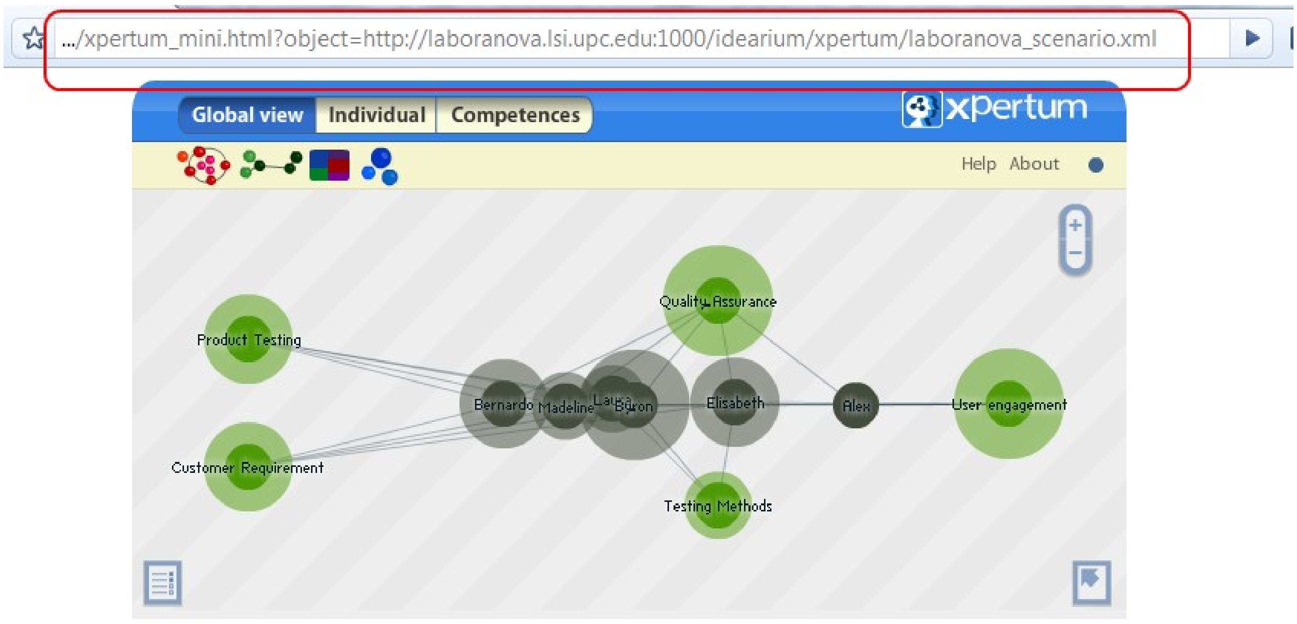


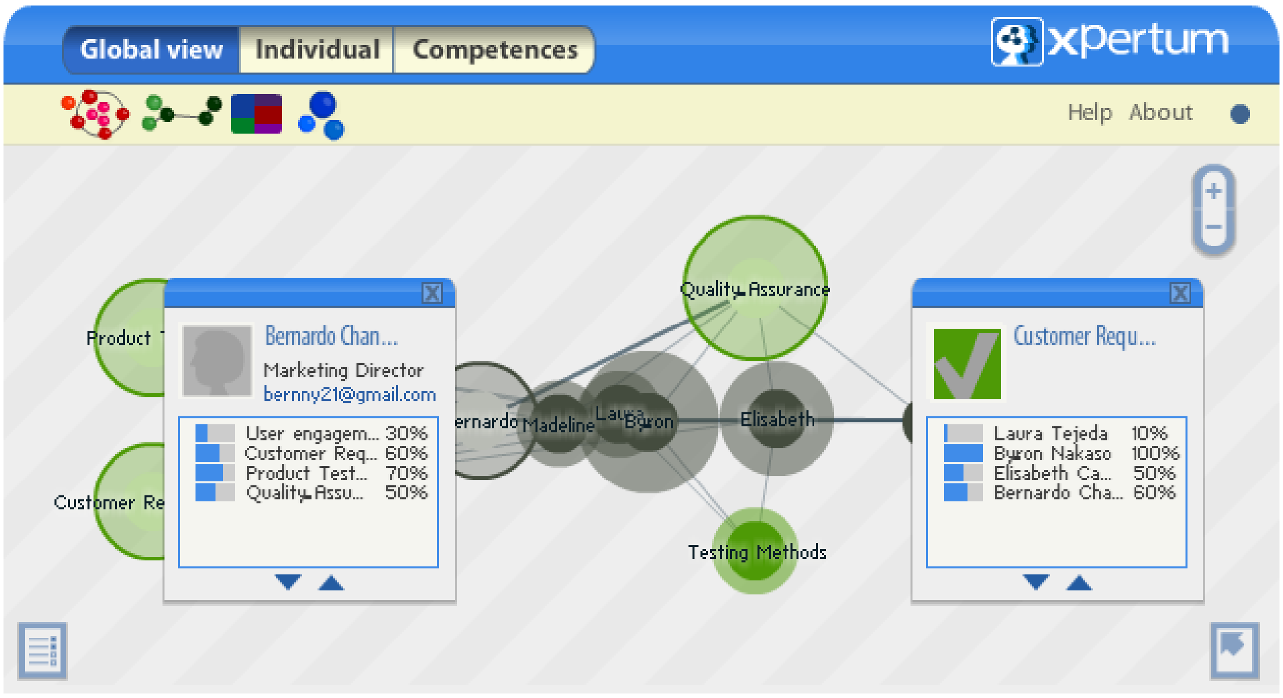
Figure 47: Setting the data source

3.5.9 Display Element Information

The Display Element Information option allows to the user the possibility of display the information of person or of an affiliation (figures 48 and 49) clicking over a circle (for graph visualizations) or over a rectangle (for treemap visualizations).

The information is displayed in a pop-up and the user can have 5 pop-up displayed at the same time. When the user wants close the pop-up, he has to click over the icon "x" and to scroll the list of relationships, he has to click over the two arrows.

Pop-ups can scroll around the visualization area too, the user has to click over the upper bar and drag the pop-up using the mouse device.



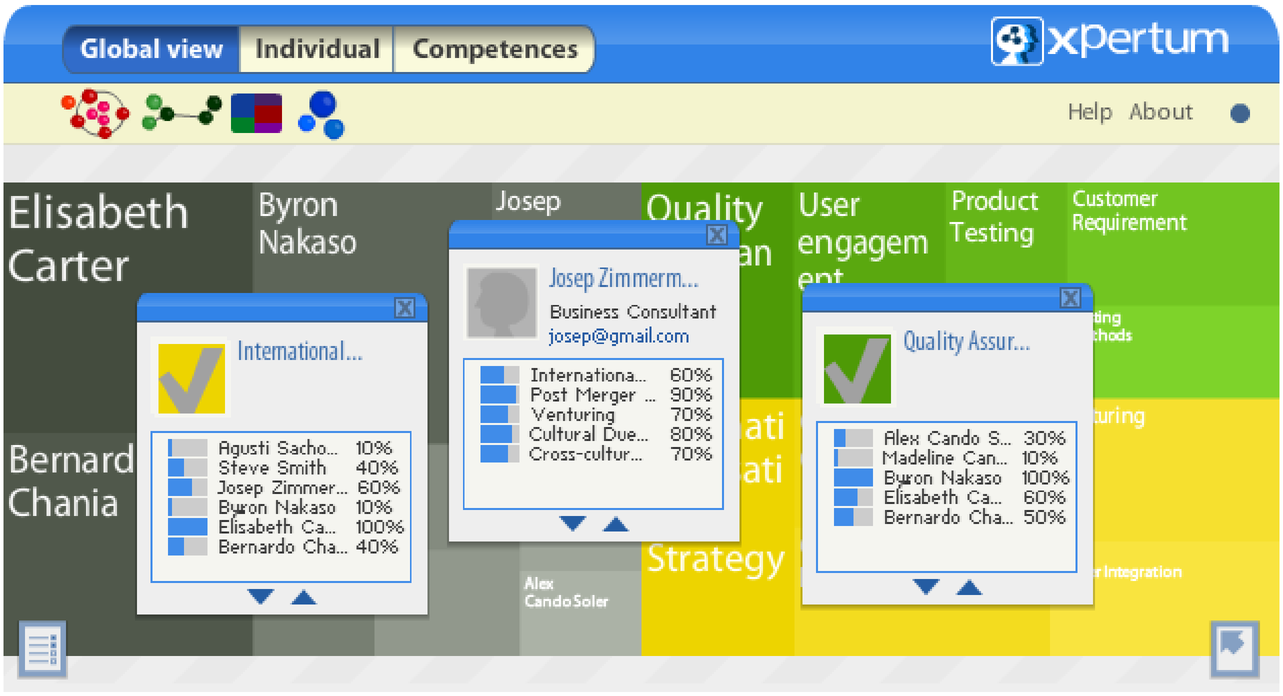
Bernardo Chan...
Marketing Director
bernny21@gmail.com

User engagem...	30%
Customer Req...	60%
Product Test...	70%
Quality Assu...	50%

Customer Requ...

Laura Tejeda	10%
Byron Nakaso	100%
Elisabeth Ca...	50%
Bernardo Cha...	60%

Figure 48: Displaying element information with pop-ups in a graph visualization



International...

Agusti Sacho...	10%
Steve Smith	40%
Josep Zimmer...	60%
Byron Nakaso	10%
Elisabeth Ca...	100%
Bernardo Cha...	40%

Josep Zimmerm...
Business Consultant
josep@gmail.com

Internationa...	60%
Post Merger ...	90%
Venturing	70%
Cultural Due...	80%
Cross-cultur...	70%

Quality Assur...

Alex Cando S...	30%
Madeline Can...	10%
Byron Nakaso	100%
Elisabeth Ca...	60%
Bernardo Cha...	50%

Figure 49: Displaying element information with pop-ups in a treemap visualization

3.5.10 Help & About

The users has access to consult the help or about information putting the mouse cursor over the “Help“ and “About“ (figure 50).

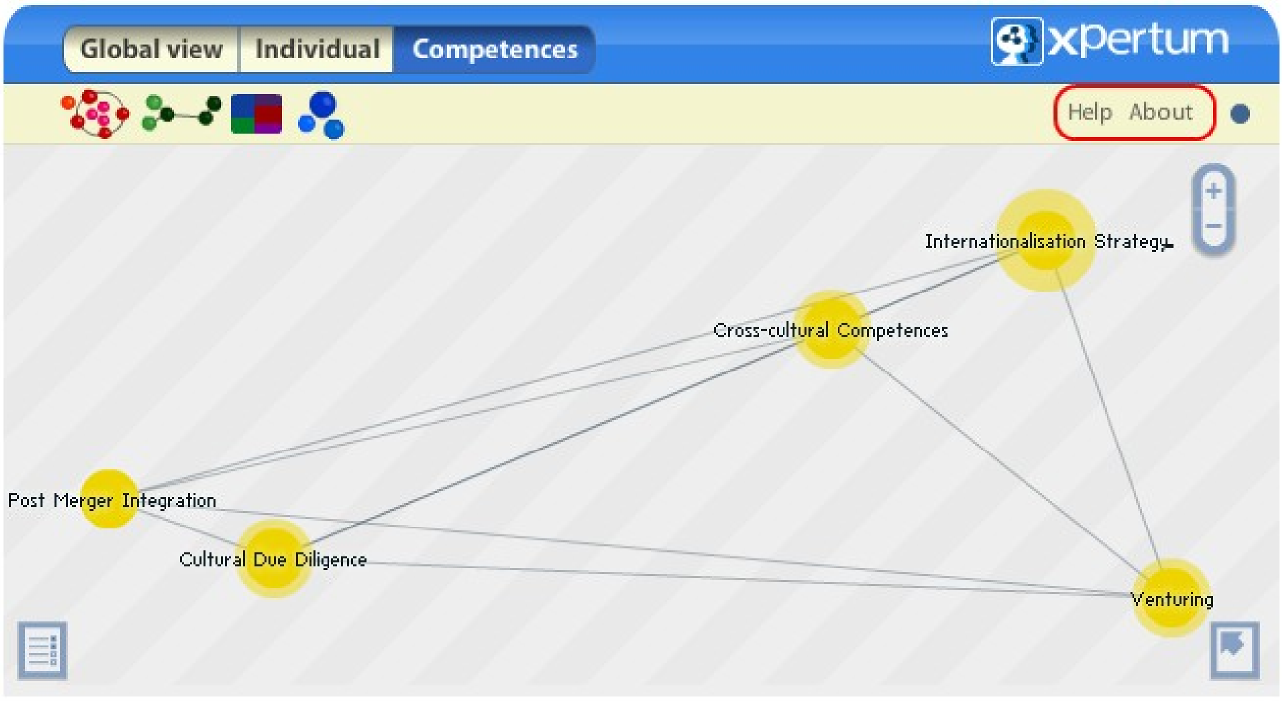


Figure 50: Help and About controls

3.6 Evaluation

3.6.1 Introduction

This evaluation chapter is a summary extracted from a workshop that was realized to evaluate the Laboranova tools in Dauphine between more than 50 organizations that participated.

We decided include its to complete the information that we considered relevant to design Xpertum.

3.6.2 Questionnaires and feedbacks

To obtain the feedback, the participants responded these questionnaires to evaluate the innovation of the tool, the usefulness and the ease of use.

The questionnaire was solely composed of closed questions. The authors used likert scale and likert item ranging from 1 to 5, 1 being the lowest, 5 being the highest to get a finer appreciation on the audience feedback on tools. The first question on innovativeness is an exception as respondents were invited to rank (and not rate) the different tools.

They gathered a total of 20 fully fulfilled questionnaires. Because this number is low, the following results should be interpreted with caution. The sample being too small, no results are presented in percentage. Nor are they presented according to the type of organizations (SMEs, Public authority etc.), the sample for each of them being even smaller.

However, despite the low number of collected questionnaires, participants were qualified people with strong expertise. Their impression is then still highly relevant for the project and tool developers.

3.6.2.1 Innovation

Average rank	**Most frequent rank (*Number of answers)**	**Times ranked in 1st position**	**Times ranked in 5th position**
3,47	1st (6*)	6	2

Table 3: Innovation questionnaire results

3.6.2.2 Usefulness

Average position	**Most Frequent rating (number of occurrences)**	**Number of “Very useful ”**	**Number of “Almost useless”**
3,21	4 (7), 3 (6)	0	0

Table 4: Usefulness questionnaire results

3.6.2.3 Easy of use

Average position	**Most Frequent rating (number of occurrences)**	**Number of “Very useful ”**	**Number of “Almost useless”**
3,21	4 (7), 3 (6)	0	0

Table 5: Easy of use questionnaire results

3.7 Conclusions of prototype

In this document we introduced the conceptual model and specifically of Xpertum, an application to display visualizations of social networks that show the relationships between competences grouped in areas and persons to improve the creation team process of the organizations.

We designed a distributed system to separate the hard computational operations and the user interface and provide the possibility of develop in future a visualization tool for any devices of the market with net access. This is possible too thanks to the architecture in 3 layers that we used: user interface, domain and data manager.

The user interface executes at the user system like a mashup for net browsers. The domain is responsible of compute the visualizations that user interface request thought the net and it's located at a server. The data manager is responsible of receive and interpret the data from external sources using the GraphML format based on XML technology.

Thanks to this design, we haven't only a simple application, we have, thanks to the distributed architecture, the potentially possibility of take the visualizations to every device without the restrictions of hardware specifications and improving the Xpertum experience using the characteristics of the user devices.

We designed like demonstrates the evaluation that we designed an application easy to use and becomes a help to user and not a hitches.

Thanks to the results of the evaluation, we know that our application innovates the possibility of display the current human resource state is very usefully for the organisations and Xpertum will be a perfect complement to enrich the websites of the organization to offer to their employees a tool to improve the productivity of their current and future projects.

4 Plan and economic costs

4.1 Project Plan

The development of Xpertum project has been done during 10 months, from September 2008 until June of 2009 with a time loading equivalent to part-time work (4 hours per day). The development process was divided in several stages that I describe in the following part:

1. **Initial study:** In this stage previous states of art and works are researched
2. **Define project stages**: In this stage, the project objectives and stages are defined.
3. **Analyse requirements**: In this stage, meetings with director are done to define the system requirements and the functionalities of Xpertum project.
4. **Technology evaluation**: In this stage, the objective is evaluated several software technologies to achieve and discard the functionalities.
5. Technical design
    1. **Architecture design**: Design of the main architecture
    2. **Use cases and functionalities**: Definition of the use cases and functionalities
    3. **User interface design**: In this stage, the objective is designed a commercial user interface
6. Implementation
    1. **Xpertum Server system**
    2. **Xpertum Mashup**
    3. **Support tools**: Several little tools to test and debug and offer support data are realized.
7. **Test and debug**
8. **Documentation**

The project started the September of 2008 and finished June of 2009. I define the time dedicated for this project along this period in the following table:

Stage	**Hours**	**Hours(real)**
Initial study	60	60
Define project stages	40	40
Analyse requirements	40	40
Technology evaluation	80	80
Technical design		
Architecture design	20	20
Use cases and functionalities	40	40
User interface design	60	110
Implementation		
Xpertum Server System	80	120
Xpertum Mashup System	200	240
Support Tools	20	20
Test and Debug	100	140
Documentation	140	140
	880	1050

Table 6: Distribution of time

4.2 Project costs

Taking into account the real dedicated hours on each stage – shown in the last section tables – and the hardware and software necessary for the development, we calculate, in economic terms, the project costs.

To create Xpertum, 2 profiles for human resources are necessary, an analyst and a programmer. For calculating the costs, I have been supposed the following prices:

• Analyst:	45 € / hour
• Programmer:	25 € / hour

Based on the last section table and the real hours for every task, I have been assumed the total hours for every human profile. The following table shows the total cost of the project.

Human resource	
290 hours of analyst	13.050,00 €
760 hours of programmer	19.000,00 €
Software	
Free distributed OS	Free
PHP	Free
MySQL	Free
Java	Free
Adobe Flash CS3 Professional	810,84 €
Eclipse	Free
Hardware	
UPC resources	Free
Total cost:	**32.860,84 €**

Tabla 7: Project cost

5 Conclusions

5.1 Objectives achieved

I achieve all the objectives of the Xpertum project. I've designed and implemented an application that provides the possibility of display social networks to analyse the knowledge resource of the members of an organization, using distributed software technologies to distribute the computational charge and using web-based technologies to provide to users an environment to improve the innovation process.

Thanks to the design stage, I implemented a friendly user interface to make the interaction easy, that provides all the functionalities of visualizations without previous education.

5.2 Future work

My current intention is extend the user interface functionalities of Xpertum Mashup (the web-tool based on flash technology) to offer the possibility of make groups, selecting person elements, and the possibility of share the output data of these groups with others applications.

5.3 Personal interest

Thanks to this project, I improve my personal knowledge and skills about software architecture, web based technologies like AS3 and Java/Servlets.

Before the beginning of this project, I didn't know the importance of the graph visualizations and the complexity that requires display a social network using graphic elements. Now, I know that this process isn't trivial, in other words, it's the opposite.

Annex I: Xpertum Mashup Logical View

Xpertum Mashup have several classes which are responsible for handling the user interactive actions, compute the hierarchy visualizations and communicates with the distributed domain. In the following we provide a short description of the most important classes and operations of the Xpertum Mashup application. A full listing of them is depicted in figure 51.

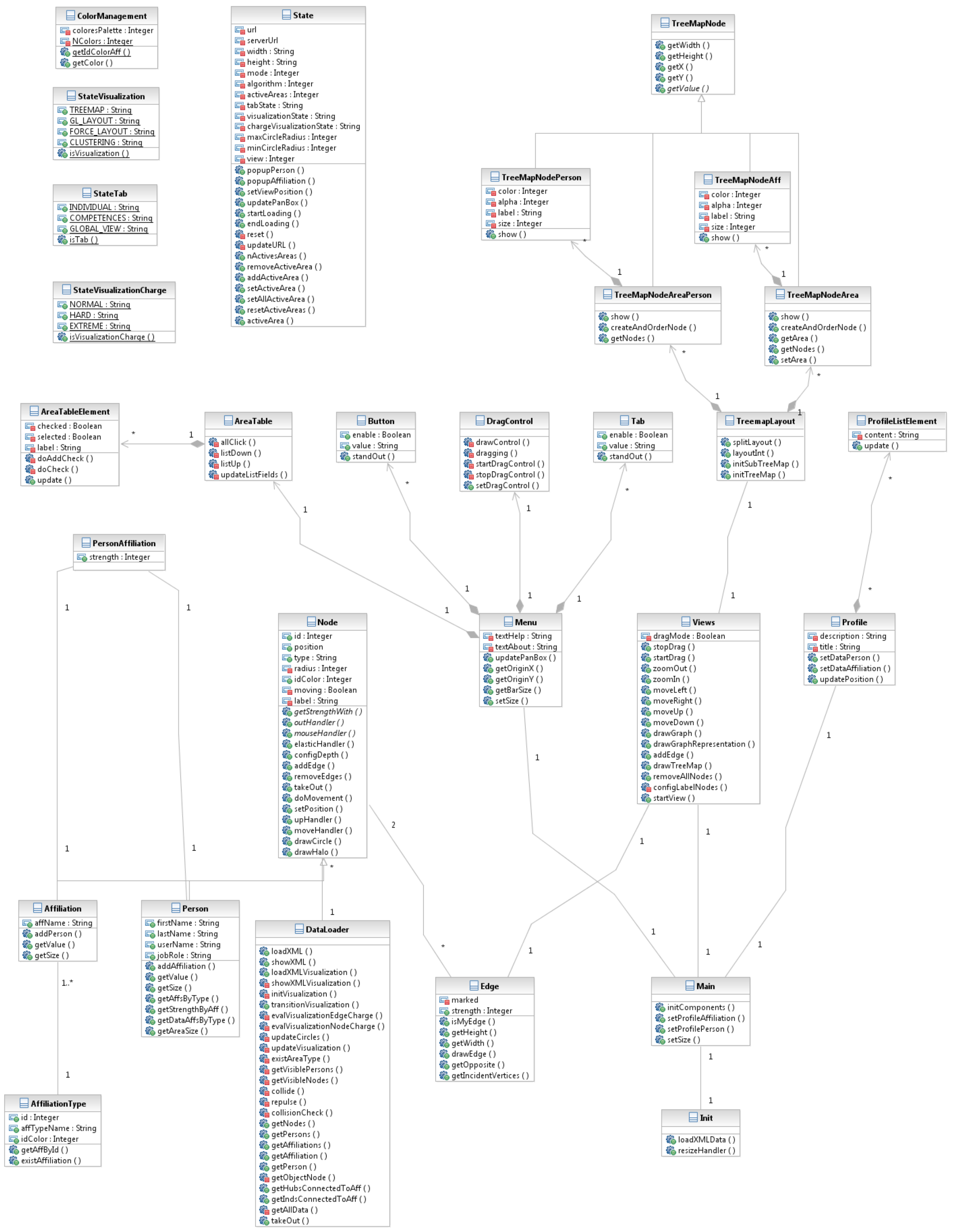


Figure 51: Xpertum Mashup class diagram

The State class

This class is a singleton class and includes all the attributes that represents the current state of the user interface application and all the operations that are needed to manage the changes in global state.

The Node class

This class is a graphic representation of a node, an element of the visualizations. “Node” includes all attributes that defines the visual representation like the radius, the label of the node, the colour and the position into the current visualization. The operations of this class allow the interaction between the element and the user across the mouse device, the manage of the relationships between nodes using edges and the graphic operations to draw the component at the visualization. The following operations are implemented:

getStrengthWith: This operation receives as input a node and returns the strength of the relationships between them.

doMovement: This operation receives as input a new position and realizes a movement animation between the current position and the new position.

drawCircle: This operation draws the node at the current position.

drawHalo: This operation draws a halo at the current position.

elasticHandler: This operation realizes a elastic movement between the current position and the original position.

addEdge: This operations receives an object Edge and add this to connections list of the nodes.

The Person class

This class is a graphic representation of a Person, an element of the visualizations and inherits of the Node class. The following operations are implemented:

addAfilliation: This operation receives as input an Affiliation and adds the relationship between the affiliation and the person.

getValue: This operation returns the value of the relevance of the person.

getSize: This operation receives as input a identifier of an area and returns the number of affiliations that are connected with the person that are from the input area.

getAffsByType: This operation receives as input a identifier of an area and returns a list of affiliations that are connected with the person that are from the input area.

getStrengthByAffi: This operation receives as input a identifier of an affiliation and returns a the value of the strength of and edge between the affiliation and the person that's displayed.

The Affiliation class

This class is a graphic representation of an Affiliation, an element of the visualizations and inherits of the Node class. The following operations are implemented:

addPerson: This operation receives as input a Person and adds the relationship between the affiliation and the person.

getValue: This operation returns the value of the relevance of the affiliation.

getSize: This operation returns the number of persons that are connected with the affiliation.

The AffiliationType class

This class represents an area forms by affiliations.

The Edge class

This class is a graphic representation of a connection between two nodes. The following operations are implemented:

isMyEdge: This operation receives as input a Node and checks if the node are in the edge.

drawEdge: This operation draws an edge between the two nodes that connects.

getOpposite: This operation receives a node of the edge and returns the other node.

getIncidentVertices: This operation returns the list of nodes that the edge connects.

The DatLoader class

This class includes all the operations that communicates with Xpertum Server to request data and graph visualizations. The following operations are implemented:

loadXML: This operation receives as input an URL with the data source and initializes the download data process from Xpertum Server.

loadXMLVisualization: This operation initializes the download graph visualization process from Xpertum Server.

transitionVisualization: This operation updates the elements of the visualizations to receive the new visualization.

evalVisualizationNodeCharge: This operation evaluates the visualization node charge to configures the current visualization state.

evalVisualizationEdgeCharge: This operation evaluates the visualization edge charge to configures the current visualization state.

updateCircles: This operation updates the size of the nodes for the new visualization.

updateVisualization: This operation is called when the download visualization process is ended. The operation loads the new visualization to display, evaluates the visualization charge, updates the circles and realizes the collision check algorithm.

collisionCheck: This operation applies over the nodes the collision algorithm (see Collision System at page 17).

getAllData: This operation is called when the download data process is ended. The operation loads the data for the session.

The Menu class

This class is a graphic representation of the menu that allows the interaction between the user and the Xpertum Mashup and change the current state. “Menu” class implements the PAN controls to move the visualization, the Zoom controls, the controls to select areas to visualize and the buttons to change the type and the algorithm to request.

The Views class

This class is a graphic representation of the visualizations and includes all operations to display the visualizations that using the data offered by DataLoader class. The following operations are implemented:

drawGraphRepresentation: This operation draws a visualization using the current state.

drawGraph: This operation draws a graph visualization using the current state.

drawTreemap: This operation draws a treemap visualization using the current state.

startView: This operation initializes the view port area to display visualizations.

zoomOut, zoomIn: These operations realizes the zoom in/out action over the visualization.

moveLeft, moveRight, moveDown, moveUp: These operations realizes the PAN actions over the visualization.

The TreemapLayout class

This class includes all the operations that generate a treemap visualization elements. The following operations are implemented:

layoutInt: This operation receives as input a rectangle and creates a list of treemap elements into the rectangle using the treemap algorithm.

splitLayout: This operation receives as input a rectangle and creates a list of treemap elements into the rectangle using vertical the treemap algorithm.

The TreemapNode class

This class represents an element of a treemap visualization and is used to implement the subclasses TreemapNodeArea and TreemapNodeAreaPerson that group the graphic representation of treemap elements TreemapNodeAff and TreemapNodePerson.

The Init class

This class includes all the operations to initialize the Xpertum Mashup program. The following operations are implemented:

loadXMLData: This operation receives as input an URL with the data source and call to DataLoader to obtain from Xpertum Server the session data.

resizeHandler: This handler controls the resize actions over the view port and update the visualization to adjust to the new size.

Annex II: Xpertum Server System Logical View

Xpertum Server have several classes which are responsible for handling compute of graph visualizations and manage data. These classes accept Xpertum Mashup requests process them and generate graph visualizations and data for the Xpertum session which are sent to the User Interface layer. In the following we provide a short description of the most important classes and operations of the Xpertum Server application. A full listing of them is depicted in figure 52.

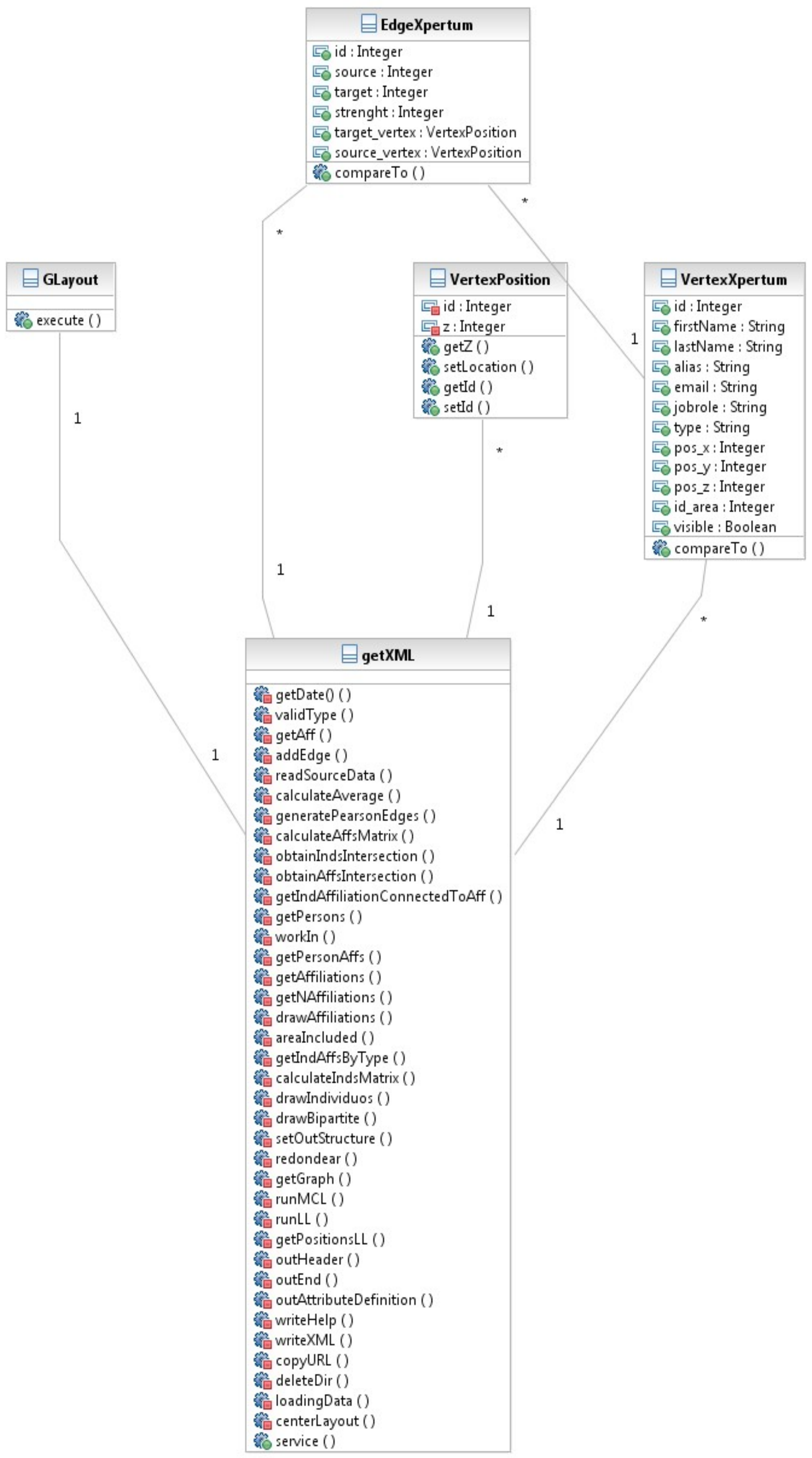


Figure 52: Xpertum Server class diagram

The graphXML class

This class includes all the operations that generate the graph visualizations and the data using GraphML format. It's the responsible to obtain the data from the external source using a URL that Xpertum Mashup provides it. The following operations are implemented:

readSourceData: This operation receives as input the source data and load the information into the local data stuctures (VertexXpertum and EdgeXpertum class) for the domain layer.

obtainAffsIntersection: This operation receives as input two list of affiliations (VertexXpertum) with the edges connections (EdgeXpertum) and compute their intersection.

obtainIndsIntersection: This operation receives as input two list of persons (VertexXpertum) with the edges connections and compute their intersection.

calculateAffsMatrix: This operation receives as input a list of affiliations (VertexXpertum) and returns a matrix with the values of relations between the affiliations.

generatePersonEdges: This operation receives as input a list of persons (VertexXpertum) and a matrix with the values of the relationships between the persons and returns a list of edges (EdgeXpertum) that relations the persons.

getIndAffsByType: This operation receives as input a person(VertexXpertum) and a list of identifiers of areas. The operation returns a list of edges (EdgeXpertum) of the person that are included in the area.

calculateIndsMatrix: This operation receives as input a list of persons (VertexXpertum) and returns a matrix with the values of relations between the persons.

drawAffiliations: This operation receives as input the dimension of view port, the identifier of the algorithm to apply and the list of identifiers of areas. The operation generates a graph visualization of Competences type that will send to Xpertum Mashup.

drawIndividuos: This operation receives as input the dimension of view port, the identifier of the algorithm to apply and the list of identifiers of areas. The operation generates a graph visualization of Individual type that will send to Xpertum Mashup.

drawBipartite: This operation receives as input the dimension of view port, the identifier of the algorithm to apply and the list of identifiers of areas. The operation generates a graph visualization of Complete type that will send to Xpertum Mashup.

getGraph: This operation receives as input a list of affiliations and persons (VertexXpertum) and edges (EdgeXpertum) and returns a graph.

runMCL: This operation receives as input a graph and applies on it the MCL clustering algorithm.

runLL: This operation receives as input a graph and applies on it the LL clustering algorithm.

writeXML: This operation write the response to Xpertum Mashup request in GraphML format.

copyURL: This operation copies the data source at local disk to guarantee that the data doesn't corrupt while the session exists.

service: This operation is the main function of the Xpertum Server and receives the requests from Xpertum Mashup and responses it.

The GLayout class

This class is the responsible of apply the geometric algorithm to a graph using the dimension of a view port and the vertexs and edges to make a visualization.

Annex III: XML Format

In order to use Xpertum as visualization tool, the organization has to store a cross domain file, called “crossdomain.xml” in the root of webserver path to allow Laboranova server reading the XML file to be visualized.

The URL of the crossdomain should be such as the following URL:
http://applicationserver.com/crossdomain.xml

This cross-domain file must contain information about access permissions. In this case, the Laboranova server should have permissions to access (read) the information from XML file that you provided.

The content of the cross domain file has to be defined as follows:

```
<?xml version="1.0"?>
<!DOCTYPE cross-domain-policy
SYSTEM "http://www.macromedia.com/xml/dtds/cross-domain-policy.dtd">
<cross-domain-policy>
<allow-access-from domain="laboranova.lsi.upc.edu" />
</cross-domain-policy>
```

If you want visualize your XML, for example, *test.xml*, you must upload the file to your server and copy the URL, for example *http://www.mydomain.org/xpertum_data/test.xml, and then, you can use Xpertum at your net browser using this URL:*

http://laboranova.lsi.upc.edu:1000/idearium/xpertum/mini0209/xpertum_mini.html?
object=http://www.mydomain.org/xpertum_data/test.xml

xpertum_url?object=xml_url

Sumary of XML Structure

- graphml
  - attributes list
  - graph
    - nodes list
    - edges list

XML Structure Details

- **graphml**
  - **Attributes Definition List** (for nodes and connections between nodes) with format: *<key id="ed-strength" for="edge" attr.name="strength" attr.type="double"/>*
    - id: identifier attribute
    - for: type of element that have the atribute, edge (connection between two nodes) or node
    - attr.name: Attribute name
    - attr.type: Attribute data type
  - **graph**
    - **List of nodes** (in this list are two types of nodes, persons or affiliation, the order is not important)
      - **Example of Affiliation node:**

```
<node id="205">
        <data key="nd-pos_x">0</data>
        <data key="nd-pos_y">0</data>
        <data key="nd-pos_z">0</data>
        <data key="nd-type">affiliation</data>
        <data key="nd-area">International Business Development</data>
        <data key="nd-id_area">2</data>
        <data key="nd-name">Cross-cultural Competences</data>
</node>
```

Attributes definitions

- id: unique node identifier
  - nd-pos_x: always 0, not necessary, only for visualization
  - nd-pos_y: always 0, not necessary, only for visualization
  - nd-pos_z: always 0, not necessary, only for visualization
  - nd-type: type of node, in that case “affiliation”
  - nd-area: Area name of this affiliation
  - nd-id_area: Identifier of the area of this affiliation
  - nd-name: Affiliation name

- **Example of Person node:**

```
<node id="1007">
        <data key="nd-firstname">Madeline</data>
        <data key="nd-lastname">Cando Soler</data>
        <data key="nd-alias">Madeline</data>
        <data key="nd-email">madeline@gmail.com</data>
        <data key="nd-jobrole">Musician</data>
        <data key="nd-organization">Toy Store</data>
        <data key="nd-phone">967834526</data>
        <data key="nd-region">Ourense</data>
        <data key="nd-postalcode">05671</data>
        <data key="nd-country">Spain</data>
        <data key="nd-timezone">GTM(+01)</data>
        <data key="nd-pictname">8_noa.jpg</data>
        <data key="nd-pos_x">0</data>
        <data key="nd-pos_y">0</data>
        <data key="nd-pos_z">0</data>
        <data key="nd-type">person</data>
</node>
```

Attributes definitions

- id: unique node identifier
  - nd-firstname: First-name of person
  - nd-lastname: Last-name of person
  - nd-alias: Nickname of person
  - nd-email: Email of person
  - nd-jobrole: Jobrole of person
  - nd-organization: Organization where person works
  - nd-phone: Telephone number of person
  - nd-region: Region where person lives (for example Melbourne, Paris,...)
  - nd-postalcode: Postal code where person lives
  - nd-country: Country where person lives
  - nd-timezone: GTM code for the place where person lives
  - nd-pictname: url of person photo
  - nd-pos_x: always 0, not necessary, only for visualization
  - nd-pos_y: always 0, not necessary, only for visualization
  - nd-pos_z: always 0, not necessary, only for visualization

- **List of edges**

  Example:

```
<edge id="101" source="2001" target="101">
        <data key="ed-strength">0.9</data>
</edge>
```

Attributes definitions

- id: unique edge identifier (it's a different list, so you can have a node and an edge with the same identifier, no problem)
- source: node identifier, that connects with target node
- target: node identifier, that connects with source node

obviously the source nd the target must be different!

- *Ed-strength: the strength of this connection*

Annex IV: Visualization capabilities

We developed Xpertum with the objective of display at the most with 500 nodes. This number supposed a problem for two reasons:

- **The view port size**

  This number of nodes have to be displayed in a little solution for mashup view (600x350 pixels) and for full screen view.

- **The time to transmission the data through the net**

  Xpertum uses two types of transmissions packages between mashup and server, the complete data, with persons profiles and competences data, and the visualization package, with the relationships to display and the positions of nodes and the size of these packages increase with the number of nodes

To solve the first problem, we developed an evaluation process that defines the charge of the current session and, in consequence, changes the appearances of the visualizations graphs. You can see the differences for two types that are applied to the visualizations in the next table:

	Normal Mode	Hard Mode
Connections	All connections are showed	Only the most relevance edges
Node labels	All labels are showed	Only for the nodes that user put the pointer mouse over
Collision detection	Yes	No (to improve the performance)
Movements between visualizations	Yes	No (to improve the performance)
Exmaple	Figure 53	Figure 54

Table 8: Characteristics of display modes

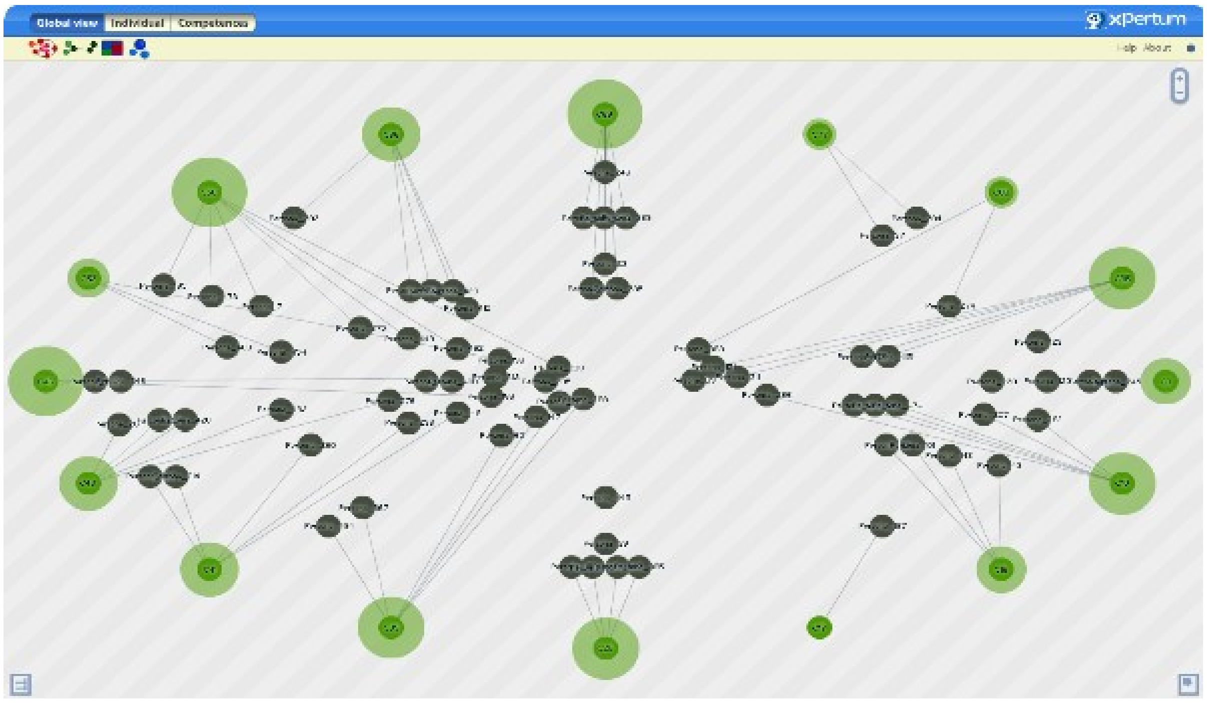


Figure 53: Xpertum Visualization in normal mode

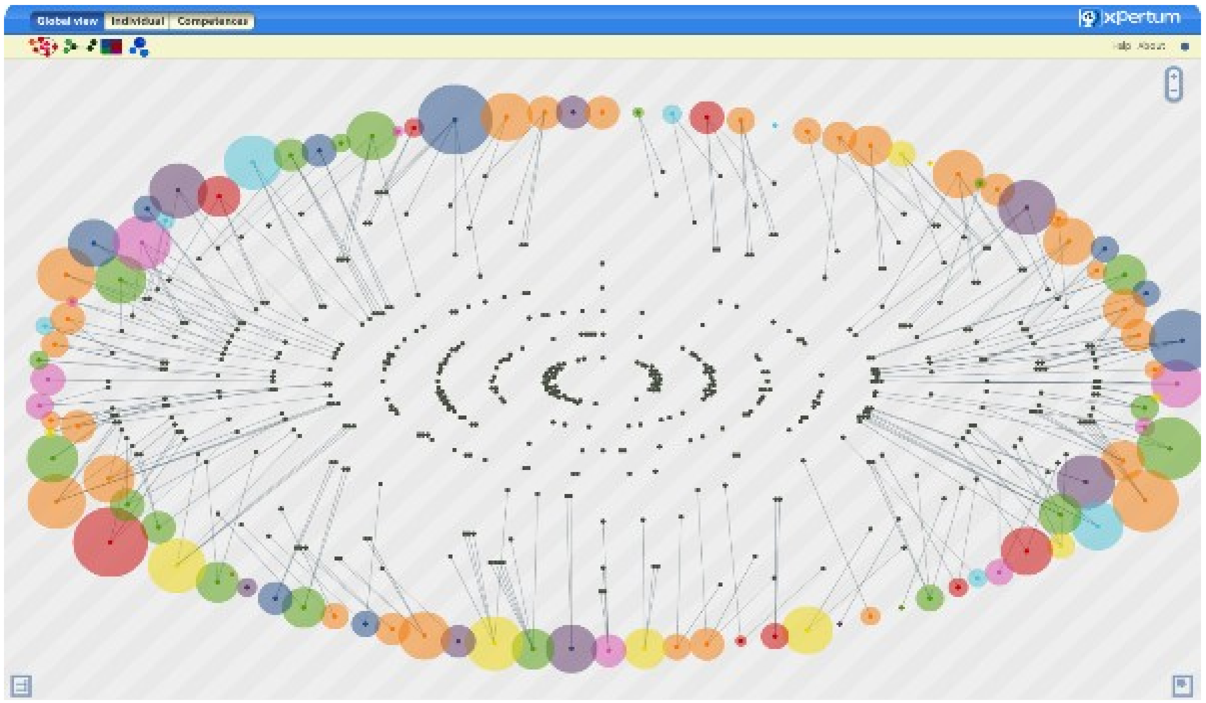


Figure 54: Xpertum Visualization in hard mode with 500 nodes

To solve the second problem, the net transmission time, we reduced the number of attributes of the nodes to only the necessary for Xpertum Mashup and developed a system in Xpertum server to identify the Xpertum user interface that requires the data (in that case, Xpertum Mashup) and provides to them in GraphML format only the data that needs.

To evaluate the Xpertum Mashup functionality, we created XML files to prove the Xpertum Mashup in a real scenery, using an ADSL connection of 1MB of bandwidth and realizing 10 times the prove for each number of nodes to obtain the medium. We present the results in the next table:

Number of nodes	**Time for download data (seconds)**	**Time for download the complete graph visualization (seconds)**
50	<1	<1
100	<1	<1
150	<1	<1
200	1	1
250	1	2
300	2	2
350	2	3
400	3	4
450	3	5
500	4	6

Table 9: Results of communication proves between Xpertum Mashup and Xpertum Server

Bibliography

1. Miles, R. E., Miles, G., Snow, C.: Collaborative Entrepreneurship: How Communities of Networked Firms Use Continuous Innovation to Create Economic Wealth, Stanford University Press, Stanford, CA (2005)
2. Gellatly, Guy; Peters Valerie. (2000) Understanding the innovation process: Innovation in Dynamic Service Industries. Analytical Studies Branch, Research Paper Series. Statistics Canada, Micro- Economic Analysis Division, No 11F0019MPE, No 127.
3. Colin Ware: Information Visulization: Perception for design, Second EditionMorgan Kaufmann, Elseivier (2004)
4. Ho, D. (2007) Research, Innovation and Knowledge Management: the ICT Factor. UNESCO, Commissioned Paper Series.
5. Joseph O'Rourke: Computational Geometry in C, Second Edition, Cambridge University Press (1998)
6. Ho, D. (2007) Research, Innovation and Knowledge Management: the ICT Factor. UNESCO, Commissioned Paper Series.
7. William Sanders and Chandima Cumaranatunge: Actionscript 3.0 Design Patterns, O'Reilly(2007)
8. Jung, http://jung.sourceforge.net/
9. Jacobson, G.Booch, and J.Rumbaugh: The Unified Software Development Process, Addison-Wesley, Reading, MA, (1999)
10. Andrew S. Tanenbaum and Maarten van Steen: Distributed Systems: Principles and Paradigms, Second Edition,Prentice Hall (2006)
11. Jean Dollimore,Tim Kindberg and George Coulouris: Distributed Systems: Concepts and Design, Addison Wesley (2005)
12. GraphML, http://graphml.graphdrawing.org/
13. Jean-Loup Guillaume and Matthieu Latapy, *Bipartite Graphs as Models of Complex Networks.* LIAFA – CNRS - Université Paris 7
14. Ryle, G. (1949) The Concept of Mind. In: Gracia, J; Reichberg, G. M; Schumacher, B.N. (Eds): The Classics of Western Philosophy, a Reader's Guide. Blackwell Publishing, 2003.
15. Fruchterman, T. M. J., & Reingold, E. M. (1991). *Graph Drawing by Force-Directed Placement*. Software: Practice and Experience, 21(11).
16. Alexander, Malcom. Using the bipartite line graph to visualize 2-mode social networks. Griffith University, Australia.
17. Andrade, Z; Tejeda, A; Almirall, E (2007) Application-oriented research on tools for augmenting social capacities. Laboranova, Deliverable D4.2.1, 2007.
18. Borgatti, S.P. and M.G. Everett (1997). *Network analysis of 2-mode data. Social Networks* 19: 243- 269.

19. Kazou, M. *Drawing Bipartite Graphs as Anchored Maps*. University of Tsukaba, Japan. 2006.
20. Freeman, L. Visualizing Social Networks. *Journal of Social Structure*, 1, 2000.
21. *(HCIL TR 91-03)* Shneiderman, B. (March 1991). *Tree visualization with treemaps: a 2-d space-filling approach*. *ACM Transactions on Graphics,* vol. 11, 1 (Jan. 1992) 92-99. HCIL-91-03, CS-TR-2645, CAR-TR-548
22. *(Tree1992)* Tree visualization with tree-maps: 2-d space-filling approach
23. *(HCIL TR 92-06)* ftp://ftp.cs.umd.edu/pub/hcil/Reports-Abstracts-Bibliography/92-06html/92-06.html
24. *(Treemap2001)* ftp://ftp.cs.umd.edu/pub/hcil/Reports-Abstracts-Bibliography/2001-06html/2001-06.htm
25. Johnson, B.& Shneiderman, B. (1991). *Tree-maps: a space filling approach to the visualization of hierarchical information structures*.
26. Stijn van Dongen, *Graph Clustering by Flow Simulation*. PhD thesis, University of Utrecht, May 2000.
27. Arturo Tejeda, *Xpertum: Competences and Social Netwoking, Universitat Politècnica de Catalunya, Research paper*