Applying clustering algorithms to web-based adaptive virtual environments

George Lepouras
Department of Computer Science and Technology, University of Peloponnisos
E-mail: G.Lepouras@di.uoa.gr

Abstract. The emergence of the World Wide Web during the past few years has provided a medium for communicating information faster and to more people than before. The technologies used allow for the development of personalised, adaptive to the users’ needs, information systems. So far, the complexity of the design and implementation of Virtual Environments has restricted their usage in locally executed, stand-alone applications. In this paper we propose an architecture that permits and facilitates the dynamic, on-the-fly creation of Virtual Environments on the Web that adapt to the users’ preferences and profiles. We focus on the algorithms available for creating an efficient virtual environment generation engine. We illustrate the proposed architecture with examples from a case study of a Virtual Museum.

Keywords: Adaptive virtual environments, rule-based dynamic creation, virtual museums, clustering algorithms

1. Introduction

The emergence of the World Wide Web during the past few years has provided a medium for communicating information faster and to more people than before. For web-based systems it is very important to be able to adjust to the users’ needs: different users with different profiles, preferences or objectives may visit the same system and they may look for different information. To this end, it is highly desirable for the system to be customisable and adaptive. A system is called customisable or adaptable when it allows users to modify parameters of the system’s functionality in order to meet their preferences. When the system automatically modifies its execution according to user’s preferences, it is called adaptive. The technologies used for Web applications, offer the possibility of creating customisable, adaptive to the users’ requirements systems.

Such systems are dynamic in their nature, because it is not efficient to a-priory create multiple instances of the same system, one for each user or user profile that may visit the web site. The dynamic character of the system increases the complexity of the design and implementation. Model-based approaches as described in [1,2] assist in the specification of content and navigation aspects of adaptable web-based systems.

Virtual environments enable the user’s immersion in a synthetic world and can provide a vivid, life-like experience. Virtual environments have found applications in a number of different areas: Strickland et al. [3] describe a virtual environment for the cure of phobias, Alison et al. a virtual environment that lets students assume the persona of an adolescent gorilla and interact as part of a gorilla family unit [4], while Charitos et al. [5] describe an environment used for aiding the organisation of autistic children behaviour in everyday tasks.
There exist a number of languages and applications that afford the implementation of virtual environments on the web. Of them VRML [6], which stands for Virtual Reality Mark-up Language has become the standard for presenting virtual reality content on the web. Using VRML one can create static as well as animated dynamic 3D and multimedia objects, and even link them to media such as text, sounds, movies, and images, creating a rich information space.

Developing an adaptive virtual environment is a far more complex task than that of implementing an equivalent hypermedia application. This can be anticipated, since both the creation of the content and the interaction methods can be more complex in 3D than in 2D. However, a number of reasons may justify the effort needed for the development of an adaptive virtual environment on the web:

- **Support of diverse target groups.** Web application users form in most cases quite diverse groups, as they originate from a variety of educational, cultural and economical backgrounds and have different interests, educational needs, available time and cognitive abilities. The content and the interaction method of the web application should be able to cater for as great a part of their needs as possible. Multilingual and multicultural issues should also be addressed. An adaptive virtual environment will be able to better meet the needs of a diverse target group.

- **Offer a vivid presentation.** A virtual environment can offer its visitors an enhanced experience. The user is able to interact with presented information, manipulate objects and navigate in the information space in a natural way. The virtual environment can employed for both educational and entertainment purposes, for visualising information or for recreating lost or difficult to visit worlds.

A virtual environment on the web offers the advantages of any web application coupled with the rich presentation facilities of the virtual environment. However, restricted bandwidth may create obstacles in delivering this content to the user. These impediments have to be taken into consideration when developing a web-based virtual environment.

In the rest of the paper the architecture and the computational algorithms for a system that creates adaptive virtual environments will be described. In more detail, the next section outlines the general architecture for implementing adaptive virtual environments on the web, the third section gives an overview of partitioning algorithms and describes a suitable algorithm for clustering virtual environment objects, while the fourth section presents a case study of implementing adaptive, web-based virtual museums. The final section concludes with guidelines and plans for future research.

2. General architecture

The presented architecture follows the general approach of Chittaro and Ranon [7]. The architecture proposed in this paper elaborates and extends this general architecture. In our approach we adopt a holistic view of the system that generates adaptive virtual worlds. For such a system, two are the main user groups involved: designers and end-users. Designers can be domain experts or they work closely with domain experts in order to prepare usage scenarios for the virtual environment. They specify object categories and index objects and their corresponding resources (3D, images, sounds, texts, etc), they define target groups, assign usage scenarios to them and define the usage data to be collected. An approach for planning the content in a distance-learning course through the Web can be found in [8].

The designer can create predefined collections of objects; assign the order of presentation either for each object individually or by means of adding category precedence, define different resources according to the target group profile. For example, the designer may assign simple, small texts as the default descriptive texts for generic profile users, detailed texts for scientists and simple texts for children.
In the proposed approach, usability of the resulting virtual environment is closely coupled with the designer’s ability to easily control aspects of the produced world. To this end, the designer can use a number of tools in order to define and modify features of the user groups, content database and personalisation engine. The next figure provides an overview of the complete system, with all its components and tools for its administration.

We will provide a brief description of each of the architecture components.

– **Usage Data Recorder**. In order to implement usage data recording one may use the event model handling mechanisms of the VRML. With VRML it is possible to implement event hooks that will be activated whenever the user performs certain tasks. The designer can select which events she wants to monitor and create a function that reports back to the system the actions carried out by the user. For example, the designer may elect to monitor whenever the user touches or moves close to certain objects like doors or exhibits and to perform tasks such as preparing the next virtual space, or to update the user’s profile.

– **Domain Content Database**. The domain content database stores virtual reality content pertinent to the context of the application. For example, in the case of a virtual museum the database will hold artefacts.

– **User Model Database**. The database of user models stores information about the user profile, age, educational background, native language, preferences in the type of media, style of virtual environment and navigation style, Internet connection type, etc. Some of these data can be adapted dynamically based on the user’s interaction or through user’s explicit selections. The usage data recorder can update information stored in the user model database. For example, the database can hold the number of times a user examined an exhibit in a virtual museum or the amount of time spent in front of an exhibit, data that can consequently be used in forming better user profiles.

– **Template Database**. The template database holds virtual spaces and interaction methods necessary to create a new virtual environment. These are preconfigured spaces or interaction methods with variables set in modifiable aspects that will be instantiated by the virtual environment generator component, when a new environment is being created. For example, the designer may store in the template database a rectangular space with exhibit stands in certain spots, special areas for descriptive texts or audio, modifiable wall colouring, etc. When the space is instantiated in the virtual environment generator the stand spots, text and audio placeholders and wall colours will be assigned with actual values.

– **Virtual Environment Generator**. The virtual environment generator interprets user requests and retrieves data from the domain content database, user model database and template database in order to construct the database according to the user profile and preferences. To realise this target the virtual environment generator uses a set of constraints defined by the designer during the creation of user profiles as well as the user’s preferences. Of them the first can be regarded as soft or optional, while the latter are hard or mandatory. For example, if a user is connecting through a low bandwidth line the designer may set a limitation in the details of the virtual environment to decrease download time, however, if the user sets a preference for high-detail objects, then this overrides the limitation set by the designer. A key issue during the generation of the virtual environment is that of dynamically defining the positioning of domain content. Since, it would be inefficient in terms of system resources and download time to create a large virtual environment, the virtual environment generator has to be able to divide the virtual environment into smaller spaces. Furthermore, the generator has to group objects of the virtual environment into ‘sensible’ collections, in order to provide meaningful information and to aid the user’s navigation. Theory offers a number of algorithms for creating
partitions of objects according to predefined criteria. These algorithms will be presented in the next section in order to help discover the most efficient and effective for the needs of the virtual environment generator.

It has to be noted that the generic approach this architecture is based upon, can actually cater for the development of a number of adaptive web-based systems and not only 3D systems. An adaptive, web-based system would require a method for monitoring users, a data store for keeping content and user profiles and a component for creating the personalised content to be sent to the user. However, in applying this generic approach alternative architecture designs may be selected. On the one hand templates, user models and domain content can reside in the same data store. On the other hand the client part of the system can be enhanced to perform a more thorough processing of the user-generated data. A third alternative would be to employ a proxy-server between the dissemination server and the client system to speed the delivery of content, by keeping a copy of most frequently requested material.

For the proposed architecture, templates, user models and domain content are depicted in separate data stores to denote that there does not exist a tight coupling between them; in the actual implementation though, they could reside in the same data store. For the second alternative, the architecture proposes a ‘thin’ client instead of an enhanced one. Although an enhanced client may reduce the amount of data that have to be communicated to the adaptive content creation system, it increases the requirements set for the client system, which in turn can limit the number of possible users. Finally, in regard to the introduction of a proxy server, such an alternative may improve the overall performance of the system especially if the virtual environment generator creates often the same content for different user requests. Nevertheless, implementation of a proxy server for dynamically created web pages can be cumbersome [9,10], and for the needs of this approach can be omitted.

3. Partitioning algorithms

Breaking up the virtual environment into smaller spaces is not always easy, if some type of consistency and rational distribution of objects both in the same space and between spaces has to be kept. Pattern recognition algorithms may be employed to accomplish these objectives. Pattern recognition is a broad discipline with applications not only in computer science but to other sciences as well. Pattern recognition aims to classify objects (or as they are called patterns) into different sets according to a number of criteria. To this end, each pattern can be characterised by a set of features $x_i, i = 1, 2, \ldots, n,$ forming a feature vector $x = [x_1, x_2, \ldots, x_n],$ which identifies it uniquely. Generating features, selection of best features and design of a suitable classifier are some of the basic steps necessary for the creation of a classification system. Clustering algorithms aim to classify objects into corresponding categories, but unlike other types of algorithms that require some initial classification as a training feature vector, clustering algorithms fall into the unsupervised pattern recognition.

In order to examine the similarity or proximity of two property vectors one may use a similarity measure. A similarity measure (SM) $s$ on a data set $X$ is defined as [11, p. 359]:

$s : X \times X \rightarrow R$

such that:

$\exists s_0 \in R : -\infty < s(x, y) \leq s_0 < +\infty$

$s(x, x) = s_0, \forall x \in X$
and
\[ s(x, y) = s(y, x), \forall x, y \in X \]

If in addition
\[ s(x, y) = s_0 \iff x = y \]

and
\[ s(x, y)s(y, z) \leq [s(x, y) + s(y, z)]s(x, z), \forall x, y, z \in X \]

\( s \) is called a metric SM.

A dissimilarity measure has a symmetrical definition and quantifies the difference between two property vectors. By employing a similarity measure one can see how ‘close’ are two property vectors. This can be employed in a clustering algorithm to test if two objects are near enough to belong to the same cluster.

Once a similarity (or a dissimilarity) measure has been chosen, one has to select an algorithm that will be able to produce sets of objects that share common properties according to the similarity measure. Although this may appear to be a simple task, it is can be a process intensive procedure. If we consider the case of virtual environment representing a supermarket with hundreds of different items, it does not exist only one classification of these items into ‘sensible’ categories. For example, one may put shampoos and soaps close to detergents, while in some other classification these items can be put close to deodorants and fragrances. The number of all possible clusterings of 100 feature vectors into 5 groups is approximately \( 10^{68} \) [12]. If a computer evaluates each clustering in \( 10^{-12} \) seconds it will require \( 10^{48} \) years to find the ‘best’ clustering.

Clustering algorithms try to overcome this obstacle by examining only a fraction of the set of all possible groupings. Clustering algorithms can be organised [11] into the following categories:

- **Sequential algorithms.** They are relatively fast methods that produce as a result a single clustering. The result is however dependent on the order in which vectors are processed.
- **Hierarchical clustering algorithms.** Methods that produce a sequence of clusterings either in decreasing or in increasing number of clusters.
- **Clustering algorithms based on cost function optimisation.** These methods employ a cost function that measures how ‘sensible’ is the clustering (usually of a predetermined number of clusters), aiming to optimise the result.
- **Other types.** This category includes special clustering methods that cannot be assigned to any of the previous categories, like bounding detection algorithms, binary morphology clustering, branch and bound clustering, and competitive learning algorithms.

### 3.1. Selection of a suitable algorithm

The need for efficient and fast generation of the virtual environment sets a number of requirements for the partitioning algorithm to be used. The main priority for such an algorithm is execution time. As previously described, in order to create a successful virtual environment generator for web, response time has to be minimised. However, the algorithm need not produce a complete clustering. Since only one cluster has to be sent to the user, execution time can be reduced by generating only clusters that appear to be more promising. Furthermore, even if a complete clustering was produced, the user’s preferences as monitored and updated by the usage data recorder component, may change before the user visits the
next virtual hall, changing thus the definition of a ‘sensible’ cluster. For example, depending on the user’s navigation, the system may record that the user prefers to spend time on a particular category of items ignoring other categories. It is ‘sensible’ to increase the weight of this category, selecting a cluster that better fits this preference.

From the categories of clustering algorithms previously presented the category of hierarchical clustering algorithms fits better to the requirements set for the clustering algorithm. Although sequential algorithms are fast, they also produce a complete clustering and more importantly the final result depends on the order in which property vectors are being processed. Additionally the result of this type of algorithms is affected by the choice of the threshold of similarity or dissimilarity. Poor choice of threshold may lead to the creation of too many or too few clusters. On the other hand, hierarchical clustering algorithms and especially divisive clustering algorithms, start with a first clustering containing a single set and divide into two clusters, selecting the optimum according to some predefined criterion. This procedure is then applied iteratively to each of the two sets, until we reach the final clustering that consists of \( N \) clusters.

Therefore, a Generalised Divisive Scheme starts with a single cluster containing the entire set \( X \) and divides it in stages, until a final clustering is reached which will contain \( N \) clusters, with one sample in each one [13]. The Generalised Divisive Scheme (GDS) is depicted in the following algorithm [11, p. 432]:

1. Initialization
   1.1. Choose \( \mathcal{R}_0 = \{X\} \) as the initial clustering
   1.2. \( t = 0 \)

2. Repeat
   2.1. \( t = t + 1 \)
   2.2. For \( i = 1 \) to \( t \)
     2.2.1 Among all possible pairs of clusters \( (C_r, C_s) \) that form a partition of \( C_{t-1,i} \), find the pair \( (C_{1_{t-1,i}}, C_{2_{t-1,i}}) \) that gives the minimum (if we use a dissimilarity function then it should be the maximum) value for \( g \).
     Next \( i \)
   2.3. From the \( t \) pairs defined in the previous step choose the one that minimises \( g \). Suppose that this is \( (C_{1_{t-1,j}}, C_{2_{t-1,j}}) \).
   2.4. The new clustering is \( \mathcal{R}_t = (\mathcal{R}_{t-1} - \{C_{1_{t-1,j}}\}) \cup \{C_{1_{t-1,j}}, C_{2_{t-1,j}}\} \)
   2.5. Relabel the clusters of \( \mathcal{R}_t \)

Until each vector lies in a single distinct cluster.

The generalised divisive scheme presented above, works as follows (\( C_{t,j} \) denotes the \( j \)th cluster of the \( t \)th clustering). Initially the algorithm starts with clustering \( \mathcal{R}_0 = \{X\} \), which is partitioned into two partitions. From all the possible pairs of clusters we select the pair that minimises \( g \) and the next clustering \( \mathcal{R}_1 = \{C_{11}, C_{12}\} \) is produced. Then, the same procedure is repeated first for \( C_{11} \) and next for \( C_{12} \). The algorithm stops when each vector lies in a single cluster.

This generalised scheme is computationally intensive. However, a number of simplifications can be made to produce a faster algorithm. The first can be based on the fact that for the requirements of partitioning the space of the virtual environment only one cluster has to be computed, the one that will be
sent to the user. Therefore, the generalised algorithm can be modified to check clusterings of only $C_{i1}$.

It has to be stressed at this point that even if a complete clustering was calculated, that could become invalid based on the user’s interaction in the virtual environment.

Secondly, there is no need to terminate when each vector lies in a distinct cluster. Since the space that is going to be sent to the user should not be either minimal or too large the corresponding cluster should not be either very small or very large, with the exact size varying according to the application. Therefore, the algorithm can terminate when a cluster of size inside the thresholds set by the application, is found.

4. Case study – Virtual museums

The generic architecture previously described can be used to implement user adaptable virtual environments on the web. A prominent application area is that of museums. The Internet has offered museums the chance to reach an even wider audience through their web sites. Museum web sites such as the one of the Tower of Pisa [14], the museum of Louvre [15] or of the Hermitage Museum [16] offer their visitors the possibility of controlling interactively the viewing of individual 3D objects, panoramic views or static stereo images of 3D models. Adaptability to the user’s profile and preferences can be an essential feature for a successful museum web site. The substantial amount of exhibits and information accumulated especially in large museums and the diversity of users visiting through the web can easily justify such an approach.

In the next sections we will elaborate on the specifications of the architecture components for an adaptive virtual museum on the web, with special focus on the design issues of the virtual environment generator.

4.1. Domain content database

Of major importance for an efficient system operation is the database that will store the domain content, that is the artefacts and their corresponding resources. As described in the general architecture, it is of the essence that the database is able to store resources in various levels of details. It is also necessary to be able to associate artefacts with an arbitrary number of properties and retrieve them through these properties, as well as to assign a variety of resources and link these to different groups and even individual visitors. The schema must be optimised so that retrieval operations are executed efficiently. [17,18] present a schema and the requirements of a multimedia, multilingual database for virtual museums. In that schema, multimedia resources are stored as links to external files, facilitating the maintenance of the database content.

In the case of the virtual museums database, special care has to be taken for the digitisation process of the exhibits, as it directly affects the resulting virtual environment and the user’s experience. For the digitisation process one of the following techniques can be used: 3D photography, 3D scanning or 3D modelling, with distinct advantages and disadvantages. The technique of 3D photography can produce good quality representation of the original artefact with a reasonable amount of effort. On the one hand, 3D scanning can produce a better quality and a more precise reproduction model, but on the other hand it will also require more effort from 3D photography and can easily produce too large files with too much information. Finally, 3D modelling can be employed when the original artefact comprises of geometrical forms (as is usually the case with machines). 3D modelling offers the possibility of creating interactive objects that the user will be able to manipulate. For example, by creating the 3D model of a scientific tool, the designer may allow the user to disassemble it in order to view its inner workings.
4.2. User model database

The user model database should be able to store user profiles for both user groups and individual users. Depending on the museum’s content the visitor groups can be researchers, students (undergraduate, high school or primary school), or general public. The designer can assign general preferences according to the group’s profile. For example, researchers prefer highly detailed objects that they can study better and search for specific features of the artefact. On the other hand, a primary school student may be presented with a lower detail object and with simpler descriptive texts.

4.3. Template database

The template database of the virtual museums can hold the museum halls and the stands for presenting artefacts, categorised on their size, style, number of exits, etc. Based on the user’s preferences and on the number and type of objects the virtual environment generator will select a space and adequate number and style of stands for displaying artefacts. The template database can store for example, spaces resembling that of a modern or of a more classic architectural form, providing different styles of spaces for different content and users’ profiles.

4.4. Usage data recorder

In the case of adaptive virtual museums the usage data recorder can send valuable data regarding the user’s navigation in the system, adapting thus better the environment to the user needs. If the user shows more of an interest for some of the exhibits than for others, then this information can be recorded and utilised by the virtual environment generator during the creation of new spaces. For example, if the user shows interest for urns from Crete and does not show any interest for urns from mainland Greece, this information should be used by the virtual environment generator to increase the weight for such artefacts and create a cluster that meets better the user’s preferences.

4.5. Virtual environment generator

The Virtual Environment Generator for the virtual museums has to retrieve artefacts from the domain content database according to the user’s preferences and profile, and to consequently create the exhibition hall to be sent to the user. The divisive algorithm presented, will, whenever necessary, partition the set of artefacts that correspond to the user’s preferences into smaller groups. For the algorithm to be implemented, a proximity measure has to be defined that will be used to measure the proximity of two objects.

Two exhibits can be thought to be similar if they have been indexed into the same categories and have been assigned the same values for these categories. For virtual museum exhibits one can identify categories such as construction period, place of origin, construction material, etc. A museum exhibit can belong to one or more of these categories. If a category has a set of values \( x_i, i = 1, \ldots, n \), and an exhibit has the \( i \)th value, this can be represented by a binary vector of size \( n \), with the \( i \)th bit set to 1 and the rest of the bits set to 0. Let \( v^x \) denote a category vector for an exhibit \( x \), \( V = \{v_1, v_2, \ldots, v_k\} \) the set of category vectors and \( v^x_{i,j} \) the \( i \)th value of the \( j \)th category of exhibit \( x \).

We define a proximity measure as follows: \( s(x, y) = \sum_{m=1}^{k} \left[ w_m \left( \sum_{i=1}^{n} v^x_{i,m} v^y_{i,m} \right) \right] \), where \( w_m \) the weight of the category \( m \). The weight of the category can vary depending on the user’s criteria and profile, in order to facilitate grouping together of objects according to the user’s preferences.
We can show that \( s \) is a similarity measure. Since \( v^x_{i,m} \) and \( v^y_{i,m} \) are binary vectors their product is defined as: 
\[
v^x_{i,m} v^y_{i,m} = \begin{cases} 1, & v^x_{i,m} = v^y_{i,m} = 1 \\ 0, & \text{otherwise} \end{cases}
\]
Therefore, the maximum value of \( s(x, y) \) happens when all vector values coincide. One such case is when \( x = y \). This means that \( 0 \leq s(x, y) \leq s(x, x) = s_0 < +\infty \).

Additionally \( s(x, y) = \sum_{m=1}^{k} \left[ w_m \left( \sum_{l=1}^{n} v^x_{i,m} v^y_{i,m} \right) \right] = \sum_{m=1}^{k} \left[ w_m \left( \sum_{l=1}^{n} v^y_{i,m} v^x_{i,m} \right) \right] = s(y, x) \), proving that \( s \) is a similarity measure. However, \( s \) is not a metric similarity measure. Since two objects can belong to the same categories and have the same values for these categories \( s(x, y) = s_0 \) does not necessarily mean that \( x = y \). In order to illustrate the application of such a similarity measure we will present two examples. The first of a simple situation where the user requests exhibits from two distinct categories, and a second, more complicated one, where exhibits belong to more categories.

### 4.5.1. First clustering example

For a first example we can assume the existence of a virtual museum of natural history. Such a museum could comprise the following categories: habitat with attributes marine, coastal, wetland and forest; collections with attributes botany, entomology, zoology, palaeontology and mineralogy; systematic with attributes mammal, bird, amphibian, insect, fish; and location with attributes of Europe, Asia, Africa, North America, South America and Oceania. We can assume that the user requests all mammals or birds from North America. We can set a weight of 1 for all categories and the following exhibits that satisfy the user request: a moose (forest, zoology, mammal, North America), a bear (forest, zoology, mammal, North America), a woodpecker (forest, zooology, bird, North America) and a squirrel (forest, zoology, mammal, North America). A ‘sensible’ clustering would be to group all mammals together and all birds together. We will check whether the application of the algorithm will produce such a result.

Starting with a clustering of all exhibits \{moose, bear, squirrel, woodpecker\} the possible clusterings with two sets are:

\[
\begin{align*}
&\{\{\text{bear, squirrel, woodpecker}\}, \{\text{moose}\}\}, \\
&\{\{\text{moose, bear, squirrel}\}, \{\text{woodpecker}\}\}, \\
&\{\{\text{moose, bear, woodpecker}\}, \{\text{squirrel}\}\}, \\
&\{\{\text{moose, squirrel, woodpecker}\}, \{\text{bear}\}\}, \\
&\{\{\text{squirrel, woodpecker}\}, \{\text{moose, bear}\}\}, \\
&\{\{\text{bear, woodpecker}\}, \{\text{moose, squirrel}\}\}, \\
&\{\{\text{bear, squirrel}\}, \{\text{moose, woodpecker}\}\}.
\end{align*}
\]

If we apply the similarity measure previously described we get that:

\[
\begin{align*}
s(\text{moose, bear}) &= 4 \\
s(\text{moose, woodpecker}) &= 3 \\
s(\text{moose, squirrel}) &= 4 \\
s(\text{bear, woodpecker}) &= 3 \\
s(\text{bear, squirrel}) &= 4 \\
s(\text{woodpecker, squirrel}) &= 3
\end{align*}
\]

Therefore, by applying the similarity measure, the algorithm will actually produce the anticipated ‘sensible’ clustering of \{moose, bear, squirrel\} and \{woodpecker\} since this clustering minimises the value of \( s \) for any pair of clusterings.
4.5.2. Second clustering example

For a second example we can assume a virtual museum of archaeology, with three categories: era, location and manufacturing material with attributes first, second and third for era, mainland and islands for location, and clay, wood, marble, and metal for manufacturing material. We can also assume a weight of 1 for all categories except for user-selected categories, which should have a weight of 2. If the user made a query for objects made of clay, wood or marble then this category will have a weight of 2. Finally, we can assume three exhibits: exhibit $a$ of the first era, from mainland and made of clay, exhibit $b$ of the second era, from the islands made of wood and exhibit $c$ of the third era, from the islands, made of clay. Then, the exhibit vectors are the following:

$$v_a^1 = (1, 0, 0), \quad v_a^2 = (1, 0), \quad v_a^3 = (1, 0, 0)$$

$$v_b^1 = (0, 1, 0), \quad v_b^2 = (0, 1), \quad v_b^3 = (0, 1, 0)$$

$$v_c^1 = (0, 0, 1), \quad v_c^2 = (0, 1), \quad v_c^3 = (1, 0, 0)$$

and the similarity measure gives the results:

$$s(a, b) = 1 \cdot (1 \cdot 0 + 0 \cdot 1 + 0 \cdot 0) + 1 \cdot (1 \cdot 0 + 0 \cdot 1) + 2 \cdot (1 \cdot 0 + 0 \cdot 1 + 0 \cdot 0 + 0 \cdot 0) = 1 \cdot 0 + 1 \cdot 0 + 2 \cdot 1 = 2$$

$$s(a, c) = 1 \cdot (1 \cdot 0 + 0 \cdot 0 + 0 \cdot 1) + 1 \cdot (1 \cdot 0 + 0 \cdot 1) + 2 \cdot (1 \cdot 1 + 0 \cdot 0 + 0 \cdot 0 + 0 \cdot 0) = 1 \cdot 0 + 1 \cdot 0 + 2 \cdot 1 = 2$$

According to these results exhibit $a$ is ‘closer’ to $c$ than to exhibit $b$ and exhibit $b$ is ‘closer’ to exhibit $c$ than to exhibit $a$. By applying the divisive algorithm and similarity measure $s$ the virtual environment generator can group museum artefacts in ‘sensible’ clusters. In this example a possible clustering would be $\{a, c\}$ and $\{b\}$ since it minimises the value of $s$ for any pair of clusterings.

Upon producing the clustering the virtual environment generator component will perform the following actions: retrieve a suitable virtual space from the template database, retrieve the corresponding exhibits from the first clustering, parse the template and substitute the placeholders with the actual VRML objects. Since these objects may vary in size the virtual environment generator can apply a simple rule for displaying objects: for objects of smaller sizes a stand can be inserted to compensate for the difference in the size. Otherwise, the graphic designer may create template spaces with different placeholders according to the size of the exhibit. In such a case the virtual environment generator will select a template space and fill as many placeholders with exhibits of the first clustering as possible. The virtual environment generator will also incorporate the usage data recorder, a small fragment of code, into the virtual environment code. Finally, the produced environment will be sent through the dissemination server to the end user.

5. Conclusions

In this paper we described the general architecture of a system that generates web-based, adaptive virtual environments, giving emphasis on the design of the virtual environment generator component. The selection of a suitable partitioning algorithm is of vital importance for the implementation of a
successful system. The virtual environment generator has on the one hand to respond fast and on the other hand to create an interactive, intuitive, user-friendly environment that will realise the full potential of the virtual museum. A ‘sensible’ grouping of artefacts will help communicate efficiently the museum’s significance and meaning to the visitor, will aid the user in discovering the information she seeks, to view and understand the exhibits’ message while at the same time entertaining her.

References