4. KNOWLEDGE BASED FILTERING APPROACH FOR DISCOVERING CLOUD RENDERFARM SERVICES

Recommendation Engines recommend the services based on the requirements of the users. The cloud renderfarm services recommendation engine has to match both the functional and the non-functional requirements of the users in order to generate meaningful recommendations. The first stage of the recommendation is to identify the services, which meet the functional requirements of the users. This is done by the Service Discovery Engine component of the Recommendation Engine using the proposed cloud renderfarm services ontology based knowledge filtering techniques.

This chapter explains in detail about the steps involved in acquiring the domain knowledge about the cloud renderfarm services like collecting and pruning the data using appropriate data mining methods, framing a taxonomy for cloud renderfarm services and synthesizing it to a tree structured taxonomy for classifying the cloud renderfarm services and finally developing an ontology for cloud renderfarm services that could be used as the source of domain knowledge for filtering the cloud renderfarm services that matches the functional requirements of the user using the knowledge based filtering techniques.

4.1 DOMAIN KNOWLEDGE ACQUISITION

4.1.1 Towards a taxonomy of cloud renderfarm services

As a first step towards developing the domain knowledge about the cloud based renderfarm various cloud renderfarm services in the current market were analyzed to identify the key characteristics or components of these services and an overall taxonomy of the cloud renderfarm services was developed as shown below in the Figure 4.1.
The overall taxonomy design was further drilled down to a Tree Structured taxonomy design to enable easy classification and identification of services. The tree structured taxonomy is given above in Figure 4.2.

The Seven main characteristics of the cloud renderfarm services identified are:

The tree structure taxonomy, defined has the cloud renderfarm services as the root and is classified into 3 levels, namely:

a) Main taxonomy levels of Cloud renderfarm Services (Li),
b) Common functional characteristic levels (Ii),
c) Specific PaaS level characteristics (Pi).

The Main taxonomy levels of Cloud renderfarm Services (Li), indicate the main characteristics that are common to both the IaaS and the PaaS type of services and are labeled as (Li) where, (i= 1 to 10), for example L1 indicates the Delivery Models (DM), which is a main characteristic of the cloud renderfarm services which is common to both IaaS and PaaS type of services and classified under the taxonomy level (Li).

The functional characteristics that are common for both IaaS and PaaS type of services is categorized under the label (Ii) where, (i= 1 to 3), for example I3 indicates the Virtualization Technology used (VT) which is a common functional characteristic of both IaaS and PaaS type of services.

The service characteristics that are more unique or specific to PaaS type of services are categorized under the label (Pi) where, (i= 1 to 11), for example P1 indicates the Cost Estimation Methods, that is a specific feature offered by PaaS type of services than IaaS service providers.

a) Main taxonomy levels of Cloud Renderfarm Services

The ten main taxonomy levels of cloud renderfarm services were synthesized to tree structured taxonomy as given above in Figure 4.3 and each level identified is discussed in details in this section.

- Cloud render farm services delivery models (L1)

The Cloud computing basics and their technical characteristics have been explored widely in the literature [128]. Cloud renderfarm services are delivered in two service models, namely the IaaS (Infrastructure-as-a-Service) and PaaS (Platform-as-a-Service). Infrastructure as a Service (IaaS) provides the required computing
infrastructure as a service. The number of computing resources required can be scaled up or down dynamically based on the requirement. Example of IaaS type of service is the Amazon EC2.
Figure 4.2 Proposed Tree Structured Taxonomy of Cloud Renderfarm services
Amazon EC2 can also be used to set up public cloud services based Renderfarms. The difference between an IaaS and a PaaS cloud service is that, in a PaaS type of cloud renderfarm services model, the underlying infrastructure required as well as the other requirements the rendering job manager software and the software licenses required, etc. are also provided to the user. Examples of PaaS type of cloud renderfarm service provider includes Renderingfox, Rebusfarm etc. The taxonomy of cloud renderfarm delivery models is given in Figure 4.4.

- **Delivery Model Components and Sub Components (L2a, L2b)**

  In the tree structured taxonomy the DM components (L2a) and Subcomponents (L2b) are given as different levels and they include details about the components of the IaaS and PaaS type of cloud Renderfarm services. In the IaaS type, the component of the service includes Memory, CPU, HDD, Storage, Data Traffic etc. Some of these components can be classified into their subcomponents like the Data Traffic which has two sub components namely, data traffic in and out. The other component is the processing unit which could be a CPU or a GPU.

- **Deployment Models (L3)**

  The deployment models of cloud renderfarm services are the public, private, hybrid and community renderfarms as given in Figure 4.5. In Public cloud renderfarms, the files uploaded are rendered on the cloud renderfarm service provider resources and the rendered files are downloaded from the renderfarm web portal. Examples of public cloud renderfarm services are Renderingfox, Rebusfarm etc. The advantages of these type of public cloud renderfarms is the on demand scalability of resources and the pay
as per use model. Disadvantage is the security issues in a public cloud. A hybrid cloud renderfarm has the benefit of both private and public clouds like the maximum utilization of publicly available resources in private cloud; high security and the advantage of public cloud render farms provided like the elasticity of resources on-demand.

**Figure 4.5 Taxonomy of Cloud Renderfarm Deployment Models**

Community renderfarms are free the users of a community share their computational resources for mutual benefit of rendering their animation files faster. They are popular among students. VSwarm is a popular example of the community-based renderfarm. The compromise of security and privacy involved in using the community-based renderfarms are the major disadvantages.

- **License type (L4)**

  The different types of software license delivery models include free, proprietary software or a leased license from a third party. The software licensing is one of the major challenges faced by the animation studios as they have to get the license for all the possible software combinations as each project may demand different software. The PaaS type of cloud-based rendering helps them to overcome this problem as the service provider arranges for the software license with an additional fee or may include the same in the render node cost. Some also make arrangements with the SaaS-based providers. Examples include the Render Solve, Rebus Render Farm, and Render Core that provide software license and the render node cost includes the license cost.

- **Potential users of cloud renderfarm services (L5)**
Users from a wide category, including the largest animation studios, SME (Small and Medium Enterprises), freelancers, and students have moved to cloud renderfarms since it satisfies both the needs of reduced production cost and high rendering quality at the same time as given in Figure 4.6.

- **Formal Agreements (L6)**

  The two formal agreement signed by the service providers is the Service Level Agreement (SLA) and the Non Disclosure Agreement (NDA). The Non Disclosure Agreement (NDA) is signed by both the IaaS and PaaS type of service providers as it helps to prevent others from copying the animation model designs which are the intellectual properties of the animators. The detailed information about the SLA criteria mentioned in the SLA’s signed by the service providers are not available on their respective websites except for few like the Amazon EC2, however the details provided are also limited.

- **Security Measures (L7)**

  Security is an important consideration of the animators while moving their files to cloud for rendering and many service providers like Amazon EC2 provide security at various levels. The common type of security features provided by the service providers includes FTP (File Transfer Protocol, Encryption, SFTP (Secure File Transfer Protocol), SCP (Secure Copy) and firewall.

- **Cost based Characteristic Levels of Cloud Renderfarm Services**

  The Cost based Characteristic levels of cloud renderfarm services are given below in Figure 4.7. The main components include the payment models, priced components, pricing models, cost estimation methods etc.
I. Payment Types (L8)

Some of the pricing models common among the PaaS as well as the IaaS services include the prepaid packages, subscription, prepaid package, free, dynamic and monthly rental models. The advantages of these payment models are that the animator can choose the payment type according to his budget and rendering time requirement.

II. Priced Components (L9)

Figure 4.7 Overall Taxonomy of Cost

free, dynamic and monthly rental models. The advantages of these payment models are that the animator can choose the payment type according to his budget and rendering time requirement.
The various components to be considered before comparing the render node cost includes, the pre-rendering cost estimation methods, payment models, priced components and the pricing models. Using the pre-rendering cost estimation methods the users can estimate the rendering cost of their files before starting the actual rendering process and gets an idea about the total budget of rendering the complete animation file. The render node unit may be a CPU or a GPU which is charged accordingly. The charges include the activation and the use time. The other component which may also be charged includes the incoming and outgoing traffic and the storage space.

III. Pricing Models (L10)

The other important component to be considered is the pricing model. One of the pricing model common to both PaaS and IaaS services include the configuration based pricing model, where higher the configuration of resources required higher will be the price. Another commonly used model is the priority queue based pricing. Where, higher the priority of the queue in which the rendering job is placed, the higher will be the pricing. The other popular pricing models include prepaid, lease, monthly rental models.

b. Common Functional Characteristic level (ii)

The common functional characteristics of IaaS and PaaS Cloud renderfarm services as given in the figure 6 below, includes the operating system, Development Tools and the virtualization Technology. The operating system that is commonly available to the IaaS service providers includes the Linux, Windows, Mac and OpenSolaris. Whereas, the PaaS service providers more options like the Android, BDS (Berkeley Software Distribution series of Unix variants) and iOS (Original iPhone OS) etc. The IaaS service providers provide SDK (Software Development Kit) and the Command Line tools as the development tools, whereas the PaaS service provides mostly prefer only SDK. Most of the IaaS service providers use Xen or KVM (Kernel based Virtual Machine) as the virtualization technology which is nothing but the creation of a virtual version of any resource like the server or storage etc. by dividing the resource into two or more executable environments.
Unlike the IaaS services in which the resource could be used for any purpose, at the PaaS Level, the characteristics of services defined are more specific to the animation file rendering and includes characteristics like Cost Estimation Methods (CEM), Rendering Job Manager (RJM), RJM Software License Types (JM_L), RJM Pricing Types (JM_P), RJM Deployment Types (JM_D), Rendering Software License (RSL), Render Engine Supported (RES), Software Supported(S/W), Plug-in supported (Plug-in) which are
c. PaaS Characteristic levels (Pi)

![Diagram of PaaS Characteristic levels](image)

**Figure 4.8** Common Functional and PaaS Specific Characteristic of Cloud Renderfarm Services

Specific to the animation file rendering. Since the PaaS service users may not have a good knowledge about the cloud environment and requires a professional help and environment to guide him to use the services. Thus, it includes the characteristics like the Free Support Provided (FSup), Paid Support Offered (PSup) etc as given in Figure 4.8.

### 4.1.2 Comparison of Cloud Renderfarm Services Using the Tree Structure

The proposed tree-path taxonomy help the animators to identify the similarities and differences between two alternatives services clearly and visually as given below in tree path structure of Amazon EC2 and Rebus renderfarm in Figure 4.9 and 4.10.
respectively. The Amazon EC2 is of the IaaS type of renderfarm that offers the infrastructure required as a service to the animators. However, the animators need prior knowledge about installing and working with the virtual machines. Whereas, PaaS services do not require any prior knowledge.

From the tree path the animator can immediately identify the differences in the service components offered by the two cloud renderfarm services and can make decisions quickly depending on his project needs. From the above figures, one can quickly find out that the Amazon EC2 offers only the infrastructure like compute unit, but not the software license whereas the Rebus renderfarm offers software license as well and the details like the render engine software, animation software and the plug-in supported by the Rebus renderfarm details are available to the animator to quickly decide whether to opt for an IaaS or PaaS type of cloud renderfarm service.

Thus the users like the students and the new learners who may not be considered about securing their work could identify the community renderfarm services quickly.
Figure 4.9 Tree Path of Amazon EC2

Figure 4.10 Tree Path of Rebus Renderfarm
4.1.3 Ontology of Cloud Renderfarm Services

An ontology enables the researchers to share information related to a specific domain by defining all the most common vocabulary of the specific domain along with the relationship among them. Hence, ontologies are important for sharing the structure of information and their relationships in a domain for understanding among common people or software agents. It also helps to reuse of domain knowledge acquired earlier and to make the domain assumptions understandable and explicit. As the cloud computing technology bloomed, many worked towards developing an ontology for the cloud computing domain [128-130]. Sim KM, et al [131-133] developed a cloud service system based on their proposed ontology for cloud services, however the ontology they developed were only for the IaaS type of general cloud services and the proposed ontology could not be used to identify the PaaS type of cloud renderfarm services. Hence, in this research work, an ontology specific to cloud renderfarm services was developed using the protégé tool based on the tree structured taxonomy of cloud renderfarm services explained earlier. The data about the functional requirement offering of the various cloud renderfarm service providers are extracted from the web using the appropriate extraction tool like excel which is a simple, easy to use extraction tool and the extracted unstructured data is processed using the data mining techniques like the data cleaning and stemming techniques etc. to obtain the structured data about the functional offerings details of the cloud renderfarm service providers. The figure of the extracted unstructured data and the structured data are given in Appendix 1 and the structured data about the functional offerings details of the cloud renderfarm service providers is given in Appendix 2. The samples of the cloud renderfarm services domain specific ontology developed in this research work is given in Appendix 3. The knowledge based filtering methods were used in the ontology to identify the services that could satisfy the functional requirements of the user.
4.2 KNOWLEDGE BASED DISCOVERY USING SIMILARITY REASONING

The proposed recommendation methodology uses three different reasoning algorithms based on the type of data to be matched. The three reasoning algorithms used are the Concept similarity reasoning, Equivalent reasoning and the Numerical similarity reasoning. The concept similarity reasoning method, measures the degree of commonality between two concepts x and y. Whereas, the equivalent reasoning method, determines the similarity between two sibling concepts by measuring the similarity based on their label values. If the two concepts to be compared for similarity are of numeric type, then the numerical similarity reasoning method is used. The three reasoning methods are discussed in detail below. The similarity score obtained using all the three methods is aggregated using the Aggregate Sim (R) method and the services are ranked based on the aggregated similarity score in decreasing order. The three similarity reasoning methods used for knowledge based discovery are explained in detail below.

4.2.1 Concept Based Similarity Reasoning

The Concept based similarity reasoning method, measures the degree of commonality between two concepts x and y using the Formula 4.1 given below.

\[
\text{Sim}(x, y) = \rho \frac{|\alpha(x) \cap \alpha(y)|}{|\alpha(x)|} + (1 - \rho) \frac{|\alpha(x) \cap \alpha(y)|}{|\alpha(y)|}
\]

\[
\text{.... (4.1 )}
\]

Where,

\(\alpha (x)\) be the set of nodes that can be reached upwards from x including x.

\(\alpha (y)\) be the set of nodes that can be reached upwards from y including y.

\(\rho\) is used to determine the generalizations degree of influence.

\(\alpha (x) \cap \alpha (y)\) is the number of reachable nodes that is shared by both x and y.

For example, let us consider two different cases and calculate the similarity between two different nodes in the ontology of ‘Animation Software’ given in Figure 4.16
below. In case 1, let us compute the similarity between the two nodes of the 3D animation software namely, the 3DsMax and AC3D. In case 2 let us compute the similarity between two other different nodes, namely the 3DsMax and Anim8or, belonging to two different software categories like the 3D animation and 3D modeling software respectively. The steps involved in calculating the similarity between the two nodes are indicated by $\text{Sim}(x,y)$ and is illustrated in detail below.

![Figure 4.11 Ontology of Animation Software](image)

**Case: 1**

Similarity between 3DsMax and AC3D

$\rho = 0.5 \text{ (Assumed)}$

$\alpha(3DsMax) = x = 4$

$\alpha(AC3D) = y = 4$

$\alpha(x) \cap \alpha(y) = 3$

$\text{Sim}(x,y) = \left[ 0.5 \left( \frac{3}{4} \right) + 0.5 \left( \frac{3}{4} \right) \right] = 0.75$
The concept based similarity reasoning method used to calculate the similarity between the two nodes of the 3D animation software, namely the 3DsMax and AC3D is illustrated in the above example given as case 1. From the ontology of the software given above, it is evident that the value of $\alpha$ (3DsMax) which represents the set of nodes that can be reached upwards from $\alpha$ (3DsMax), including $\alpha$ (3DsMax) is 4. Whereas, the value of $\alpha$(AC3D) which represents the set of nodes that can be reached upwards from $\alpha$(AC3D), including $\alpha$(AC3D) is 4. However, the number of reachable nodes that is shared by both 3DsMax and AC3D, which is represented as $\alpha (x) \cap \alpha (y)$ is 3. Hence, the similarity between the two nodes indicated as Sim($x,y$) is calculated using the formulae given above in Formula 4.1 and the value of Sim($x,y$) is calculated as 0.75.

The similarity between two other different nodes, namely the ‘3DsMax’ and ‘Anim8or’, belonging to two different software categories like the 3D animation and 3D modeling software respectively is illustrated in case 2. The value of $\alpha$ (3DsMax) which represents the set of nodes that can be reached upwards from $\alpha$ (3DsMax), including $\alpha$ (3DsMax) is 4. Whereas, the value of $\alpha$(Anim8or) which represents the set of nodes that can be reached upwards from $\alpha$(Anim8or), including $\alpha$(Anim8or) is 4. However, the number of reachable nodes that is shared by both ‘3DsMax’ and ‘Anim8or’, which is represented as $\alpha (x) \cap \alpha (y)$ is 2. Hence, the similarity between the two nodes indicated as Sim($x,y$) is calculated as 0.5 using the formulae given above in Formula 4.1 as given below in case 2.

**Case: 2**

Similarity between 3DsMax and Anim8or

$\alpha$(3DsMax) = $x = 4$

$\alpha$(Anim8or) = $y = 4$

$\alpha(x) \cap \alpha(y) = 2$

Sim($x,y$) = $[ 0.5 \cdot (2/4) + 0.5 \cdot (2/4) ] = 0.5$
Comparing the concept based similarity values obtained in case 1 and 2, it is evident that the similarity between the two nodes in case 1 is more compared to the similarity of the two nodes in case 2. Hence, the ‘3DsMax’ and ‘AC3D’ are identified to be more similar compared to ‘3DsMax’ and ‘Anim8or’ for similarity.

4.2.2 Equivalent reasoning

The Equivalent similarity reasoning, determines the similarity between two sibling concepts by measuring the similarity based on their label values using the Formula 4.2 given below.

\[
\hat{\text{Sim}}(x, y) = \text{Sim}(x, y) + \frac{0.8|c_1 - c_2|}{10} \quad \text{.... (4.2)}
\]

Where,

- \(\text{Sim}(x, y)\) – is the similarity value calculated using the Similarity reasoning method.
- \(C_1\) – represents the label value of concept \(x\).
- \(C_2\) – represents the label value of concept \(y\).

As an example, the steps in the equivalent similarity reasoning method to calculate the similarity of the software versions of the Maya software using the ontology given in Figure 4.12 is illustrated below. As the equivalent similarity reasoning method, determines the similarity between two sibling concepts by measuring the similarity based on their label values, the labels of the different software versions of the Maya software are labelled based on their version. The oldest version Maya 1.0 is labelled as 1 and the latest version of Maya 4.0.2 is labelled as 12. As an example, similarity is calculated between Maya 4.0.2 and Maya 3.0, Maya 4.0.2 and Maya 1.0 as illustrated below.

Let us calculate the similarity between Maya 3.0 and Maya 4.0.2 first. Based on the Formula 4.2 given above to calculate the similarity between the two software versions, the value of \(C_1\) that represents the label value of concept \(x\) (Maya 3.0) is 7 and the value of \(C_2\) that represents the label value of concept \(y\) (Maya 4.0.2) is 12. Let
us assume the Sim(x,y) to be 0.5, applying these values to the formula, equivalent similarity between Maya 3.0 and Maya 4.0.2 is calculated as 0.5327 as given below.

Let Sim(x,y) = 0.5, then

\[
\text{Sim (Maya 3.0, Maya 4.0.2)} = 0.5 + 0.8 \frac{|7-12|}{10} = 0.5327
\]

Similarly, the equivalent similarity between Maya 1.0 and Maya 4.0.2 is calculated. Where, the value of C1 that represents the label value of concept x (Maya 1.0) is 1 and the value of C2 that represents the label value of concept y (Maya 4.0.2) is 12 and the similarity between Maya 1.0 and Maya 4.0.2 is calculated to be 0.5085 as given below.

\[
\text{Sim(Maya 1.0 & Maya 4.0.2)} = 0.5 + 0.8 \frac{|1-12|}{10} = 0.5085
\]

Figure 4.12 Ontology of Maya 3D Animation Software
From the above illustrated example, it could be observed that the Maya 4.0.2 is more similar to Maya 3.0 compared to Maya 1.0.

4.2.3 Numerical reasoning

The similarity between two concepts in the same domain that are of numeric data type can be determined using the numerical reasoning method based on the formula given below:

\[
\text{Sim}(a,b,c) = 1 - \frac{a-b}{\text{Max}_c - \text{Min}_c} 
\]

… (4.3)

Where, ‘a’ is first numeric value & ‘b’ is second numeric value and ‘c’ is concept name. The numerical reasoning has been used to calculate the similarity of the attributes that are of the numeric value like the render node cost as given below. For example, to filter the cloud renderfarms that charges around 3.5$ as the render node cost, the numerical similarity reasoning method is applied as illustrated below. Let, ‘a’ is first numeric value be the desired value of render node cost and ‘b’ indicate the second numeric value that has to be compared and ‘c’ is concept name, which is the render node cost. ‘Max\( _c\)’ and ‘Min\( _c\)’ indicate the maximum and minimum values of the particular concept considered. Then, as an example, let us calculate the similarity between the desired render node cost of 3.5$ and 2.5$. Let us assume the values of ‘Max\( _c\)’ and ‘Min\( _c\)’ as, Max\( _c\) = 6.0 and ‘Min\( _c\)’ = 1.0 . The similarity value is calculated as 0.80.

Sim (3.5,2.5,Render node Cost),

\[
= 1 - \frac{3.5 - 2.5}{6.0 - 1.0} = 0.80
\]

Sim (3.5,5.5,Render node Cost),

\[
= 1 - \frac{3.5 - 5.5}{9.0 - 2.0} = 0.71
\]

Thus, the numerical similarity reasoning method can be used to filter the
concepts of numeric type and the similarity values obtained for all the concepts are aggregated and the services with the highest similarity value is recommended first followed by the others in the descending order of their aggregated similarity values.

4.3 DISCUSSION

The knowledge based filtering approach for discovering the cloud renderfarm services discussed in this chapter uses three different similarity reasoning methods based on the concept or attribute type to identify the services that are similar to the user requirements. In the recommendation engine portal that has been developed, once the user is validated and enters his functional requirements, the recommendation engine calculates the similarity between the nodes in the ontology and identifies the cloud renderfarm services with the functional offerings that are more similar to the user functional requirements and recommends those services in the first phase of recommendation as shown in the Appendix 4. The listed cloud renderfarm services that satisfy only the functional requirements of the users are further ranked based on the QoS requirements of the users using any one of the MCDM methods.

However, if all the cloud renderfarm services filtered in the first phase are new and the QoS data about these services are not available then, these services filtered in the first phase itself is provided to the users as the recommendations to overcome the cold start problem. Once the user starts to use the recommended new services, the QoS data are obtained. The method of recommending services in three different stages helps to overcome the cold start problem in recommending new type of cloud services and enhances the accuracy of the recommendations. The MCDM methods used for ranking, the services based on their Quality of Service and re-ranking based on their QoE rating are discussed in detail in the following chapter 5.