View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Oxford Brookes University: RADAR

An integrated personalization framework for SaaS-based cloud services

Haolong Fan[a], Farookh Khadeer Hussain[a], Muhammad Younas[b], Omar Khadeer Hussain[c]

[a]*School of Software*
Faculty of Engineering and Information Technology
University of Technology, Sydney
Ultimo, NSW 2007, Australia

[b]*Computing and Communication Technologies*
Oxford Brookes University
Oxford, UK

[c]*School of Business*
University of New South Wales;
ADFA, Canberra ACT 2600, Australia

Abstract

Software as a Service (SaaS) has recently emerged as one the most popular service delivery models in cloud computing. The number of SaaS services and their users is continuously increasing and new SaaS service providers emerge on a regular basis. As users are exposed to a wide range of SaaS services, they may soon become more demanding when receiving/consuming such services. Similar to the web and/or mobile applications, personalization can play a critical role in modern SaaS-based cloud services. This paper introduces a fully designed, cloud-enabled personalization framework to facilitate the collection of preferences and the delivery of corresponding SaaS services. The approach we adapt in the design and development of the proposed framework is to synthesize various models and techniques in a novel way. The objective is to provide an integrated and structured environment wherein SaaS services can be provisioned with enhanced personalization quality and performance.

Keywords: cloud computing, SaaS, cloud service, personalization, semantic web, dynamic service composition, recommendation engine

1. Introduction

Cloud computing has enormous potential in shaping the IT service industry and in providing new business models and opportunities. A cloud (software) service, made accessible over the Internet, is the fundamental aspect of cloud computing [1]. Cloud services can be published, integrated as well as remotely provided to users through the web/Internet [2, 3]. One of the commonly used cloud service provisioning models is Software as a Service (SaaS). SaaS is hosted and run entirely on cloud servers. Users are able to access and use such services via browser, irrespective of their location or devices [3]. SaaS is generally provided in a multi-tenant fashion where each user has an independent instance of a SaaS service. As the number of SaaS grows, so too do the demands of users in terms of personalization and service quality.

Personalization means adopting a range of service strategies to meet the requirements of different individuals. For example, in business applications, a personalized service plays a key role in supporting enterprise market strategies [4]. Personalization seeks to integrate users' preferences, which are of different types and vary by location, time and other environment contexts. To successfully deliver appropriate personalized content and services, personalization involves gathering the user's profile information during the users interaction with the service [5]. The aim of personalization is to improve the user's experience of the services.

Email addresses: FHL-VP@hotmail.com (Haolong Fan), Farookh.Hussain@uts.edu.au (Farookh Khadeer Hussain), m.younas@brookes.ac.uk (Muhammad Younas), O.Hussain@adfa.edu.au (Omar Khadeer Hussain)

A significant trend in cloud services today is to adopt different services for disparate users' interests and needs. Devising an effective service personalization framework is an active research area in the cloud services domain and the challenge is to find the right content for users' needs. In most cloud systems, users need to furnish fine-grained request information during the interaction process to achieve customized tasks in cloud applications [6]. However, these personalization service providers do not currently offer a convenient and intelligent solution for users. It is therefore vital to design a new framework that: builds on different techniques and technologies which are essential for the design and development of SaaS services; and provides efficiency and scalability in the provisioning and consumption of SaaS-based services. For instance, current server-based personalization approaches are tailored in such a way that the entire processing of personalization services is carried out on the server side, such as user profiles, preferences, security, privacy checks and so on. But this places heavy burden on the cloud servers. In order to improve the efficiency of the personalization systems, some of the tasks can be assigned to the client side.

In addition, a lack of semantics in the personalization framework often leads to poor knowledge discovery about users' preferences and the discovery/selection of desired services. Current recommendation techniques are based on classical information mining techniques in order to carry out recommendation in personalization systems. However, SaaS-based personalization requires multifaceted recommendation techniques that recommend accurate services based on users' preferences. Most traditional personalization frameworks focus mainly on offering personalized services based on users' profiles and historical data. However, other factors (e.g., knowledge of domain experts) are also useful for enabling the framework to make more precise recommendation decisions for users [7]. A better capability to deliver personalized services needs a collaborative recommendation system to be able to offer recommendation items, based on similarities among different users' profiles [8]. Further, in a personalization framework, one dedicated service is rarely able to satisfy all service requests [9]. Enhancing interoperability and collaboration across a set of cloud services to deliver personalized services is a challenging task. Building composite services may be a good approach; however, the dynamic composition of atomic services is a complex problem in service integration [10]. A beneficial framework needs to choose suitable services automatically and to set up corresponding configurations to offer personalized services dynamically.

In this paper, we propose an integrated SaaS-based personalization framework which builds upon current cloud-based architectures such that it can be deployed across different cloud platforms. The objective is to implement efficient personalized services for users using a SaaS-based platform.

The rest of the article is organized as follows: in Section 2, we present related work on the personalization framework. In Section 3, an overall design of the framework is proposed. In Sections 4 - 9, we explain each area of this framework in detail. In Section 10, we describe the implementation of an experimental prototype to evaluate the personalization framework. Implementation aside, this paper will also demonstrate a series of evaluations and comparisons of the prototype, according to a given set of metrics to provide evidence of how this framework improves effectiveness and efficiency compared to existing frameworks. In Section 11, we draw conclusions and identify the plan for future work.

2. Related Works

This section reviews and analyses some of the related work on personalization of cloud services. Hsueh et al. [11] proposed a cloud-enabled platform in order to facilitate the collection and delivery of evidence for personalization in a multi-provider ecosystem environment. This platform provides basic personalization services, smart analytics for active personalization and dynamic provision. This platform design consists of an infrastructure layer, platform layer and application layer. The application layer adopts a single SaaS model developed using open APls of the common services from the platform layer. Though useful this platform does not provide a common integrated design for SaaS.

Guo et al. [12] proposed a framework to support the provisioning of personalized services for individual users. The authors split the framework into two parts. On the client side, a user's profile and activities are recorded, and a user model is also built up to avoid server overhead. On the server side, cloud services fetch user preferences from the client side periodically and update data accordingly on a central cloud storage facility. Similarly, Gopalan et al. [13] proposed a recommendation framework with two parts. On the client side, a context and content information learner runs on a device client while a recommendation service is placed on cloud servers. These client-side designs were built successfully, but most heavy tasks are still handled on the server side, neglecting the capability of the current browser

client, new front-end technologies and the utilization of semantics. In our previous work [14], we created an innovative approach to implement semantic client-side personalization for SaaS. While other personalization services rely extensively on the server side, this solution leverages Rich Client technologies and Semantic Web to ameliorate performance and efficiency. It presents encouraging results in SaaS-based services. However, due to lack of server side design, it does not offer a full framework for personalization on SaaS.

CPRS [15] is a cloud-based recommendation system for TV platforms. In this system, the client-side is used to record user habits and receive programs. On the other side, programs in a cluster will be recommended to users with similar preferences by peer-to-peer connection. CPRS has been used for generating Cloud-based recommendations. The authors have not used CPRS for the personalization of Cloud services. Furthermore, this work does not consider the semantics of data while generating recommendations. In our proposed personalization framework (Figure 1), we enhance the working of the Recommendation Engine by considering the semantics of data. The semantics-aware recommendation engine is used for personalization of Cloud services.

The work in [16] presents an approach to generate semantic user profile data for personalization using various algorithms such as RandomNeighbour, VisitBasedCF, VisitBasedImpact, etc. This approach provides more useful recommendations. An Edge Service Architecture [17] deploys an edge service to handle caching preference, leading to query process and decision making. It also offers a personalization outcome with relative high accuracy. However, these approaches [16, 17] do not account for cloud-oriented design, context-ware preference, multi-dimensional knowledge, and client-side approach.

BooksHPCLab [18] is a mashup application (from different sources of the Web), aims to offer intelligent and scalable personalization services. By introducing semantic rules and taking into account semantic-based mechanisms, this application enhanced the performance. An revised version of this application (BooksHPCLab_revised) implements link data in the mashup design and gathers data to map an OWL ontology. However, this application falls short of rendering personalized Cloud services. Given that these two approaches are closely related to personalization approaches, we compare the performance of these approaches with our proposed methods in Section 10.

3. SaaS Personalization Framework

This section describes the proposed framework called *"SaaS Personalization Framework"*. Figure 1 represents a conceptual diagram of a personalization process. It comprises different phases, including: Data collection, Data preparation and transformation, Data preservation, Recommendation, Personalization and service delivery [14]. These phases require different information sources so as to prepare personalization of services. The information sources include: (i) User profile: a group of personal data associated with a specific user, (ii) User preference: a set of user options defined by a user for SaaS services, (iii) Usage data: a users behavior information on a specific service, (iv) Recommendation rule: a set of rules to define recommendation logic, and (v) Service directory: a group of candidate services that could be offered to users.

Data Collection is a process to gather user profile and activities data via various client-side and server-side methods. As long as raw data has been collected, it will be filtered, processed and transformed by related components in the *Data Preparation and Transformation* phase. In the *Data Preservation* step, the preference data will be stored in a user data repository in a suitable format. Based on data mining techniques, a set of recommendation patterns will be generated in the *Recommendation* phase. These patterns are critical to the framework's ability to perform recommendations and generate recommended preferences. Next, in the *Personalization* phase, personalization configurations are constructed by performing related computing methods with user preference and recommendation patterns. In the last step, the final personalized services are supplied to users in the *Service Delivery* stage.

Based on the conceptual framework, we design the proposed framework. It comprises four fundamental layers as shown in Figure 2. In the *Client Layer*, it deploys a semantic client-side approach [14] with local web storage and semantic web content for SaaS. A *Business Model Layer* on the main tier is responsible for receiving user data, storing data, processing data, generating patterns, building personalization configurations and recommending services. In this tier, an *Edge Component* communicates with client-side components and synchronizes data between the client side and the server side. An *Observer Module* gathers user preferences and environment information and generates concept usage data. By applying data mining techniques, the task of a *Recommendation Engine* is to discover recommendation patterns

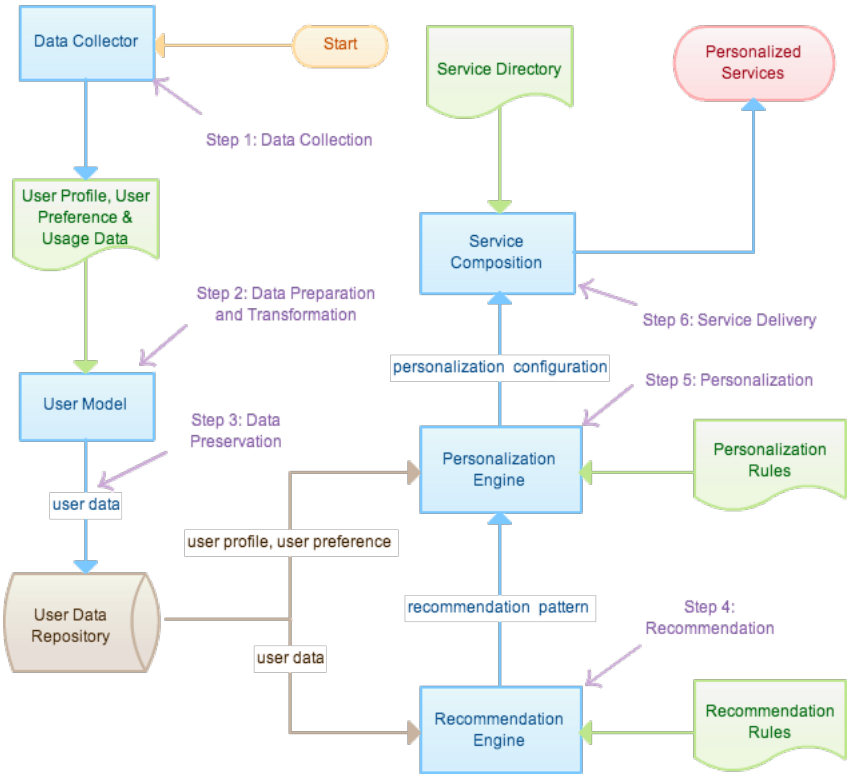

Figure 1: Conceptual Diagram of Personalization Process [14]

by investigating user data to predict user personalization. According to the recommendation patterns offered, a *Personalization Engine* contributes to the creation of personalization configurations and the delivery of service content. In the *Business Model Layer*, a *Dynamic Service Composition* is responsible for choosing and configuring the cloud services that will be allotted to users. A *Data Access Layer* builds up data entities and performs task manipulation for the data system. A *Data Layer* contains database systems and file systems that store data including user profiles, user preferences, user activities, service directory and ontology. The last tier, *Cloud Services*, is located in a remote cloud environment which supplies a group of candidate cloud services. In the following sections we explain each of the components of the proposed framework in more detail.

4. Extended Semantic Client-side Personalization

We extend a semantic client-side personalization approach [14] by integrating Rich Client and Semantic Web for SaaS-based personalization. To improve the efficiency in gathering user profile data and reducing the overhead of server-side computing, parts of the functions and recommendation tasks are migrated from the back end (server-side) to the front end (client-side). The objective is to exploit semantic contents in order to enable direct access to user profiles and activities. First, the user model is established on the browser as a dedicated user source. Second, a mechanism is proposed to provide efficient user data synchronization between the client side and the server side. Third, this approach carries out fundamental data mining and recommendation tasks which are usually deployed on the server side. Lastly, parallel programming can also be initiated on the client side using a MapReduce-like mechanism with Web Workers (CMRW2). With enhanced computing capability, the client-side approach becomes an irreplaceable part of a complete personalization framework. Overall, the rich client solution dramatically improves the performance in the proposed framework.

The architectural design (as in Figure 3) of the proposed approach is separated into two parts. On the client side, web content with semantic elements is presented to users on screens. The semantic web content can be easily detected and tracked by JavaScript event handlers in collecting user profiles and behaviors by a *Dynamic Profile Collector*. User preferences are then stored in a local web storage facility,

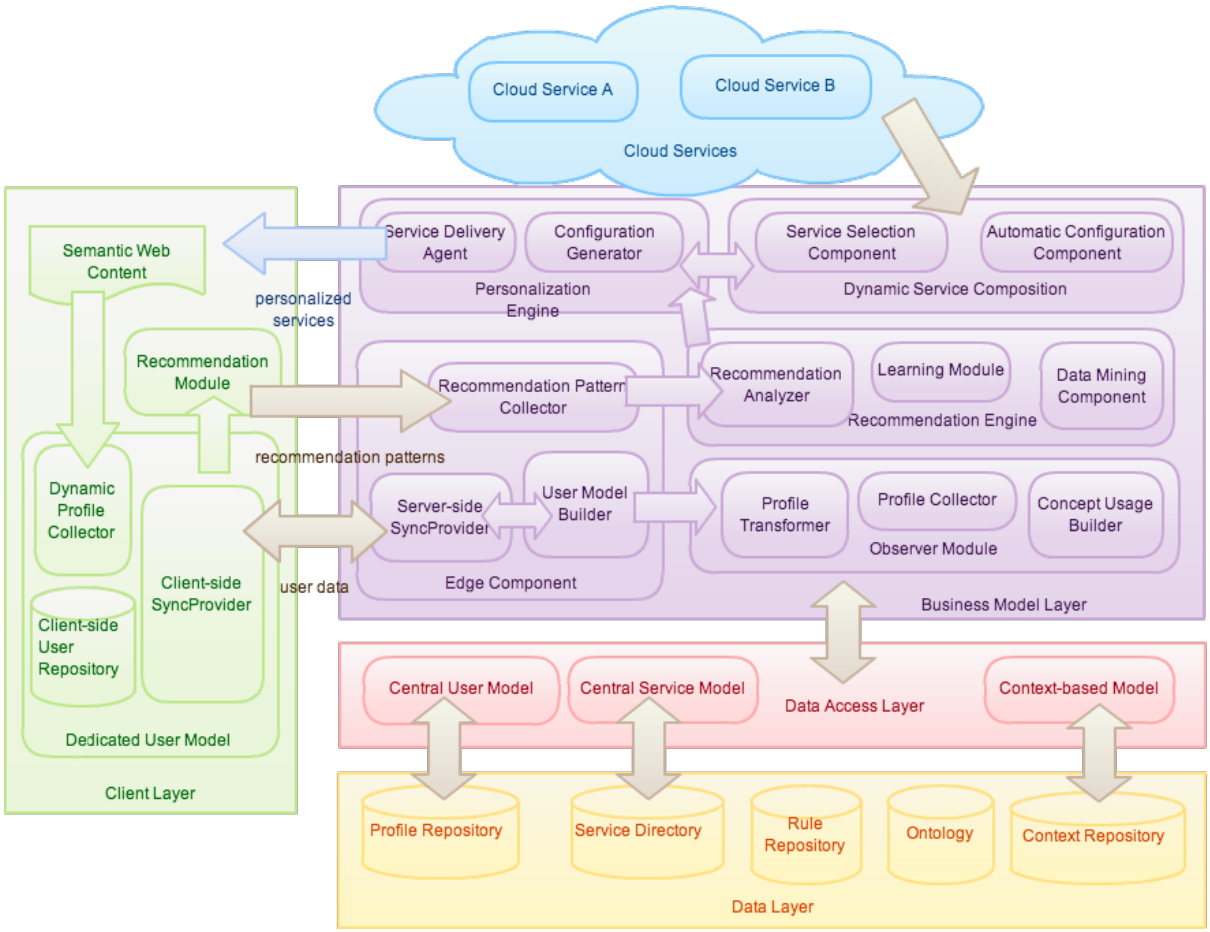

Figure 2: Architecture of Personalization Framework

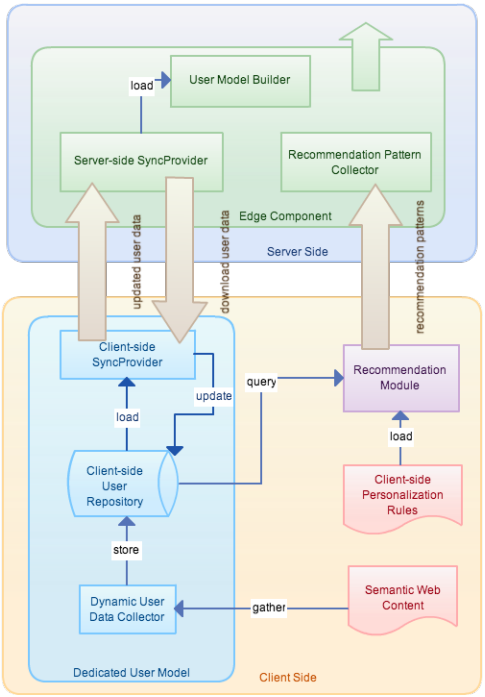

Figure 3: Architecture of the Semantic Client-side Approach [14]

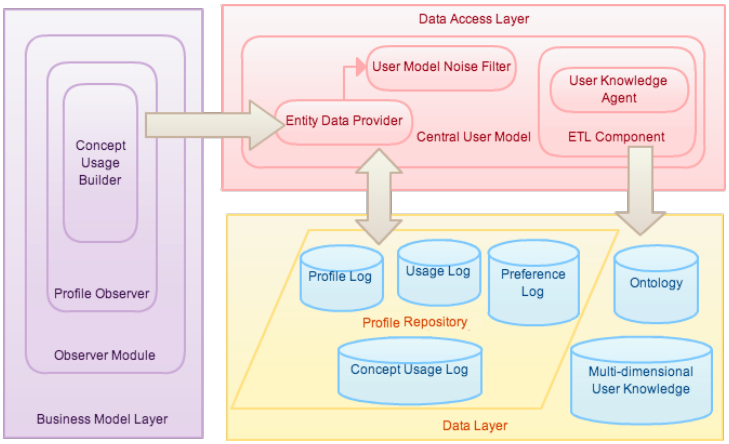

Figure 4: Architecture of the Central User Model

the *Client-side User Repository*. A *Client-side SyncProvider* connects to the server side and synchronizes updated user data with the central data system.

On the client side, a *Recommendation Module* is deployed to process profile data and to generate an initial client-side recommendation pattern. Five data mining methods: Weighted Web Usage Clustering, Simple Content-based Filtering, Decision Rule Technique, Association Rule Discovery and Sequential Pattern Mining, are applied in this module. In this recommendation procedure, a set of recommendation rules with JSON-LD format are imported into the Recommendation Module to implement the Decision Rule Technique. A composite client-side recommendation pattern is then submitted to the server side for further processing.

On the other side, an *Edge Component* that consists of the *Server-side SyncProvider*, *User Model Builder* and *Recommendation Pattern Collector* synchronizes the user data and receives initial recommendation patterns from the client-side. All data is passed to other components in the *Business Model Layer* in this framework (see Figure 3).

5. The Central User Model

A user model presents a collection of personal profile data and web access behaviors. It is important for a personalization framework to employ a well-designed user-interpretable user model to achieve personalization goals [19]. The user model provides the fundamental data for adaptive changes to user preferences and behaviors. It provides data sources for personalization analysis and manipulation. The final recommendation decisions of personalization are dependent on the data stored in the user model.

User modeling is a human-computer interaction that performs operations on building a user model to adapt users behavior more intelligently [20]. To maintain the user model, a key integrated component called the *Central User Model* on the Data Access Layer handles data manipulation for a *Profile Repository* and a *Multi-dimensional User Knowledge* database [21]. It consists of three components - an *Entity Data Provider*, an *ETL (Extract, Transform and Load) Component* and a *User Model Noise Filter*. Components on the three layers participate to implement smooth data transfer and establish an integrated user model. The diagram revealing the whole design architecture for the user model is presented in Figure 4.

5.1. Profile Observer

In the design of SaaS Personalization Framework, a *Profile Observer* in the *Observer Module*, observes user data for building a user model. The Profile Observer loads, manages and transforms user information including user profiles, user preferences and usage records. It is responsible for receiving user preference data from the *Edge Component* and retrieving the state information and behaviors from the surrounding environment [22].

Additionally, a *Concept Usage Builder* in the *Profile Observer* performs the transformation of raw usage data from the *Usage Log* to the Concept Usage Log. An innovative feature in this framework is the

creation of a Usage Log to record user access logs and a Concept Usage Log for the extraction of usage patterns [23]. The Concept Usage Log creation process involves two steps: content classification and log transformation [24]. The content classification process analyzes the semantic usage log and assigns categories to each usage record. The log transformation is performed to transform the concept usage data into a semantic format, which is OWL/RDF in this framework.

5.2. Profile Repository

A user profile can be significantly enriched by semantic data. All the information about a user is expressed and stored in the form of OWL/RDF in this framework. We define a repository, *Profile Repository*, as a container for storing profile data, preference data, usage log and concept usage log. A usage log is a record of a user's behavior information. A concept usage log is a group of classified and optimized usage logs. A usage log may contain inconsistencies such as incorrect or missing entries. Thus, the major difference between them is that a concept usage offers more structured and organized information, avoiding any useless entries. In this repository, all profile information from different end users can be managed in a centralized way. This approach enables the creation and management of extensible and expressive user profiles [25]. By introducing a profile representation structure, the aggregation of profile data from multiple heterogeneous data sources is optimized [25]. Any additional data field can be imported into an appropriate triple in this profile repository without changing the structure of data model. Thus, user profiles can be organized and expressed in a clear way.

An *Entity Data Provider* placed in the Central User Model is deployed to support data manipulation and query for the user data. The main rational tasks for this component include add/query/update user profile and user preference, and add/query usage log and concept usage log.

Additionally, a *User Model Noise Filter*, situated in the Central User Model, filters statistical noise from collected user profiles and behavior logs. The noise takes the form of invalid values, useless information or blank data. This noise is mixed into the user profile and usage data, and it is important to apply algorithms to clean the data or reduce the noise to an acceptable level. Hsueh et al. [11] proposed an algorithm to filter statistical noise from collected user profile and behavior logs. Based on the PWR-wide risk stratification algorithm, and according to concept usage patterns, if one new usage data demonstrates 100% deviation from the pattern, we define it as a noise. The noise data will not then be imported into the Profile Repository.

5.3. Multi-dimensional User Knowledge

A *Multi-dimensional User Knowledge*, an analysis-based data, is designed to describe the characteristics of users and historical usage records. It provides a long-term analysis and prediction for user personalization. Specifically, Multi-dimensional User Knowledge could be a standard data warehouse to store all relevant data for all users. An *ETL Component* is responsible for tracking user logs and processing ETL (Extract, Transform and Load) for the Multi-Dimensional User Knowledge. It dramatically improves long-term recommendations for personalization.

In the user model, the creation and revision of a knowledge base that captures a set of dimensions which correspond to the different conceptual characteristics of a user and a personalization action is proposed [19]. We define five separate dimensions in user modeling: user profile, preference, usage, concept usage and feedback rating, which assist in performing advanced data mining processes within the framework.

The Multi-dimensional User Knowledge database is based on a data warehouse, which combines data from multiple variable sources into one high-efficiency and easily manipulated database. The Multi-dimensional Knowledge can also be refreshed automatically. A *User Knowledge Agent* in the ETL Component is adopted for tracking the User Repository regularly and then dynamically revising Multi-dimensional User Knowledge database with a user's latest behaviors [19]. The responsibility of this agent includes filtering the user model, grouping user data, maintaining knowledge and performing real-time ETL for intelligent analysis [26]. The purpose of adopting an agent-based architecture is to render a set of services that have the capability of maintaining a model of users and offering adaptive services to each user [19].

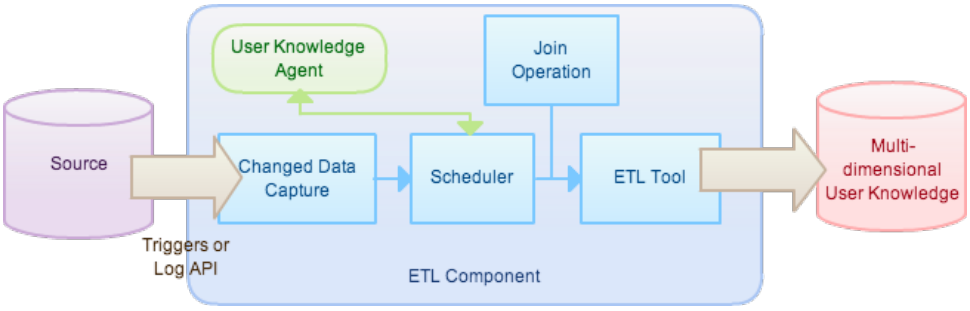

Figure 5: Framework of CDC-based Scheduling Real-time ETL Mechanism

5.4. CDC-based Scheduling Real-time ETL Mechanism (CSR-ETL)

The traditional approach to update data for analysis is to refresh offline the entire data warehouse. ETL is based on time slicing, in which the ETL process is triggered in a fixed time interval which could be up to many hours [27]. Recently, real-time data warehousing has become the means to eliminate long period offline and develop intelligent systems to process continuous real-time ETL [28, 27]. With real time ETL, data is refreshed in minutes rather than hours [28]. As the expectation of efficient recommendation grows, updated user data could be analyzed in real-time, which will dramatically improve efficiency in achieving personalization goals [29].

A real-time ETL mechanism needs to be developed for the proposed SaaS-based personalization framework, given that recommendation tasks must keep working whenever the ETL process is performed while data volumes are exponentially growing. The key to designing a real-time ETL approach includes ETL task triggering and scheduling [27]. This real-time solution processes data as quickly as possible (to meet the constraints of maximum refreshment of data) without imposing additional expense on the sources. Several approaches to optimize the real-time ETL process by changing triggers and the micro-batch process exist. Three operations called change data capture (CDC), scheduling, and join operation play a key role during the entire real-time ETL process, and are especially useful for pre-processing data and consolidating ETL manipulations. We define this mechanism as a CDC-based Scheduling Real-time ETL Mechanism (CSR-ETL) (see Figure 5). The ETL Component with the User Knowledge Agent is adopted to assist the framework to implement this real-time ETL. The key components in this mechanism are explained as follows:

a. Changed Data Capture (CDC)

Real-time ETL should be triggered according to a minimum batch of records, which is integrated from multiple data sources. In this design, CDC identifies, captures and delivers data as it is being inserted, updated, or deleted from the data source [30]. Database triggers, widely supported by most industrial databases, activate the execution of defined procedures when a table change occurs. Some database systems also offer Database Log APIs for handling table changes programmatically. These methods make the changed data available for integration processes [30].

b. Scheduling Algorithm

A new scheduling algorithm is proposed to solve the issue of conflicts when new records are imported, or existing records are refreshed in parallel. Jie, Yubin and Jingang [27] illustrate a flexible scheduling algorithm that estimates system context and distributes resources with a balancing mechanism.

In a real-time ETL, updates and queries are often performed in concurrency. Thus, a scheduling algorithm based on context factors should be considered. These context factors could include the optimized concurrency of the environment, the current primary function being updated or being queried or both, the incoming speed of queries and updates, and the extraction speed of queries and updates[27], all of which are ETL related performance factors. This algorithm is located in a *Scheduler*.

c. Join Operation

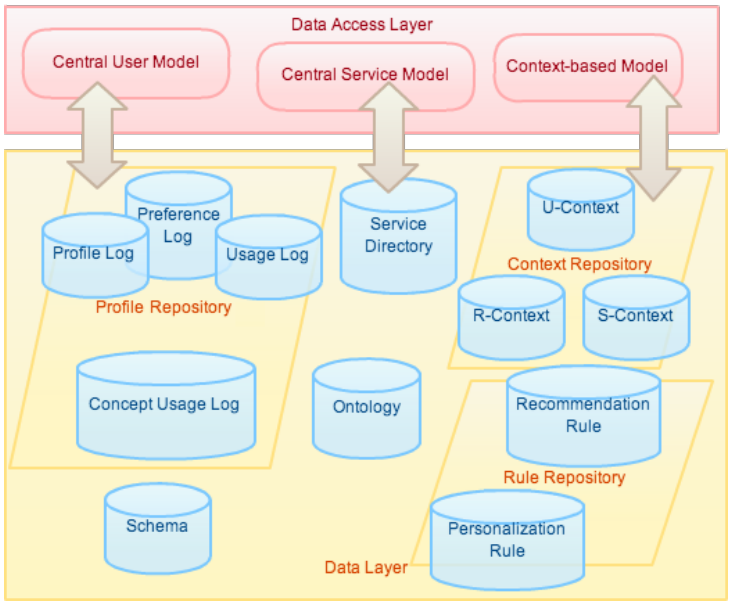

Figure 6: Architecture of the Semantic Data Module

To produce useful information from the raw data retrieved from different data sources, factual and dimensional information need to be joined. Tank et al. [31] emphasize that join operations need to be deployed to combine data from two or more data sources. The dominant sources are primarily from the data stores in the Profile Repository. Using matching keys in join operations is recognized as a powerful data processing strategy because it enables rapidly changing data to be organized [31].

6. Ontology-based Semantic Integration

The semantic approach aims to enrich a huge, dynamic and federated web with machine-processable semantics [32, 33]. To realize personalized services according to users' needs, we take account of a semantic approach in the design of the proposed personalization framework due to several reasons. First, the semantic web provides a way to represent knowledge of information [2]. Second, semantic techniques are useful for retrieving information from the web, which can be viewed as a huge distributed database [2]. Third, the semantic approach assists us to generate personalization outcome accomplished by a set of cloud services.

Semantic integration is a useful tool for performing information integration. It integrates multiple semantic models and semantic data in one framework and enables information or resources to be located easily and quickly on the web [2]. It involves systems with the capability to understand the meaning of each piece of data and its inner relationships which is extremely useful for discovering patterns in relation to a user's needs, interest and preferences. For instance, Solomou [18] proposed an intelligent and scalable personalization service that utilizes linked data to produce metadata-oriented recommendations for building personalized services.

6.1. Semantic Data Module

A *Semantic Data Module* is a group of components and data stores with semantics. It receives OWL/RDF data for user profiles, user preferences, user usage data, concept usage data, service directory and context-based data. A rule repository is a vital data store which contains all rule data. As explained in the semantic client-side approach [14], the recommendation rules defined as JSON-LD are applied to perform rule-based recommendations. On the server side, personalization rules stored as OWL/RDF information are indispensable for the *Personalization Engine*. Additionally, a *Schema* is implemented to preserve schema information for the data model. The complete architecture diagram is illustrated in Figure 6.

A *Central Service Model* is responsible for providing access to manipulate service profiles. In our personalization framework, it uses a semantic annotation to describe each cloud service for dynamic

personalized mapping [9]. A cloud service can be described by a semantic annotation of what it does and how it works. To cope with a user's personalization patterns, a semantic service annotation must be applied to seek appropriate services for the user.

Personalization is not only dependent on a user's profile and activities, it is also dominated by the surrounding environment of the system, including the user's status (online or offline), service status (working or temporarily down) and resource status (available or used up). Based on context-based theory [5], three sub-contexts are built into this architecture to classify context data for the user model: U-context (user context), S-context (service context) and R-context (resource context). The role of U-context in the Central User Modeling System is to keep track of a user's current status and preferences. S-context defines the current service status including its execution constraint. R-context is used to detect the current status of resources. U-context, S-context and R-context collaborate to perform a context-based personalization as an additional enhancement to the personalization framework.

A *Context-based Model* in this framework uses a number of policies to manage the integration of personalization into cloud service composition and provision [5]. Before performing personalized services, this framework checks the status of services and resources using S-context and R-context, which will be executed according to certain context restrictions. Only available services and resources can be used as personalization outputs for users.

6.2. Resource Description with OWL/RDF

According to the W3C recommendation, the Resource Description Framework (RDF) and OWL/XML are the foundation for processing metadata in the Semantic Web [33]. SPARQL (a recursive acronym for SPARQL Protocol and RDF Query Language) query is an RDF query language as well as a data access protocol to manipulate (create/read/delete/update) RDF triples [25, 2]. As an open standard, OWL/RDF is a universal shared knowledge representation language. There are several advantages to embracing the Semantic Web using OWL/RDF: first, for achieving semantic interoperability, OWL/RDF has significant advantages over XML with more independent syntax and better reuse. A semantic unit is given naturally through object-attribute construction. Second, the information is easy to extend when new attributes need to be inserted [25]. Third, because OWL/RDF is assembled by triples, it can be efficiently implemented and stored. It optimizes the capturing process for explicit and implicit user data and retrieving profiles for recommendation [32]. Last but not least, OWL/RDF syntax is layered, thus the encoding operation can be easily processed. A set of well-designed parsing technologies across various platforms can be easily applied with OWL/RDF. In this design, RDF/XML is the major data format for *Profile Repository*, *Service Repository*, *Context Repository*, *Ontology* and *Schema*.

6.3. Ontology Design

An ontology affords a common knowledge that the system can learn from the personalization domain [2]. It significantly extends the capability to undertake large-scale machine processing automatically for recommendation. The whole of the server-side *Semantic Data Module* is organized and interlinked by the ontology. The ontology contains the terms and relationships between these data entities which can be encoded as knowledge for a machine to understand and accumulate personal information on web access behaviors and habits, as well as the emotional influence of the accessed resources [23]. The ontology-base semantic solution is useful for offering a better personalization outcome in this framework by relating recommendation technologies.

In this design, the idea of using an ontology is to build personal knowledge for users. The knowledge focus on the user is at the center of personalization integration [34]. As a people-centric personalization approach, this framework identifies relationships and properties among all the resources that surround people to interconnect them. By continual knowledge expansion, a network of users can be established. Providing the correct mapping for a given user's related data from the personalization domain is critical and helps us to mine useful and connotative user preferences with the ontology-based recommendation technologies.

An ontology data store identifies the knowledge and relationships for all the terms used in this personalization framework using OWL data [2]. The *Schema* maintains all schema and vocabularies which are rendered as RDF Schema (RDFS) for all semantic resources. Let us take a single user ontology as an example. A *Profile* has three properties: *Preference*, *U-Context* and *R-Context*. *Usage* is the action performed by a *Profile* (user). *Concept Usage* is attached with the *Usage* in the ontology design.

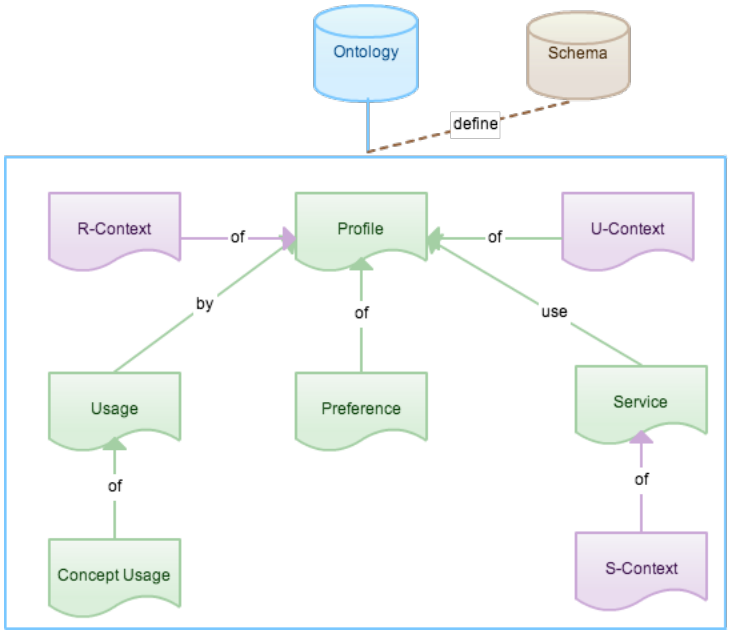

Figure 7: Conceptual Diagram of Ontology Design

A *Service*, which has a property *S-Context* to describe the current service status, is used by a *Profile* (user). A simple overview of the ontology design is illustrated in Figure 7.

6.4. Microdata and JSON-LD for the Client Side

On the client side, the semantic approach plays an important role in information retrieval and representation [14]. Given that a fast and simple solution is preferred for implementation, heavy XML-based formats like OWL or RDF are certainly not suitable for the client side. *Microdata* as a semantic markup is used in HTML pages by adding more attributes to describe the details for each HTML element. To store metadata in local storage, *JSON-LD*, derived from the JavaScript language, is the best choice for explaining the definite meaning of each piece of information with minimized header overload. It inherits dominating features such as being lightweight, providing natural support, having a clear structure and is easily manipulated.

7. Recommendation Engine, Client-side Recommendation Module and Ontology-based Data Mining

Recommendation aims to meet the expectation of personalization with recommended patterns based on user profile, usage log and preference. In order to make recommendations, algorithms are used to discover an appropriate recommendation set for user access, considering the active user profile in conjunction with the discovered usage patterns [35]. The recommendation is usually tied to data mining technologies. Recommendation systems can be broadly categorized into two types: *content-based* and *collaborative filtering* [36]. The best solution is to combine the two methods to generate recommendation patterns. In the proposed framework, data mining and recommendation technologies are widely allocated and distributed to two sides. Some simple and direct technologies are implemented in a *Recommendation Module* on the client side; while other complicated and social-related technologies are deployed in the *Recommendation Engine* on the server side, notably by applying ontology-based mining. The final recommendation pattern is produced by processing all the recommendation patterns via a *Recommendation Analyzer*. In this process, reasonable filtering, merging and analyzing methods have significant effects on the final recommendation patterns. Figure 8 represents the whole architecture for the server-side Recommendation Engine.

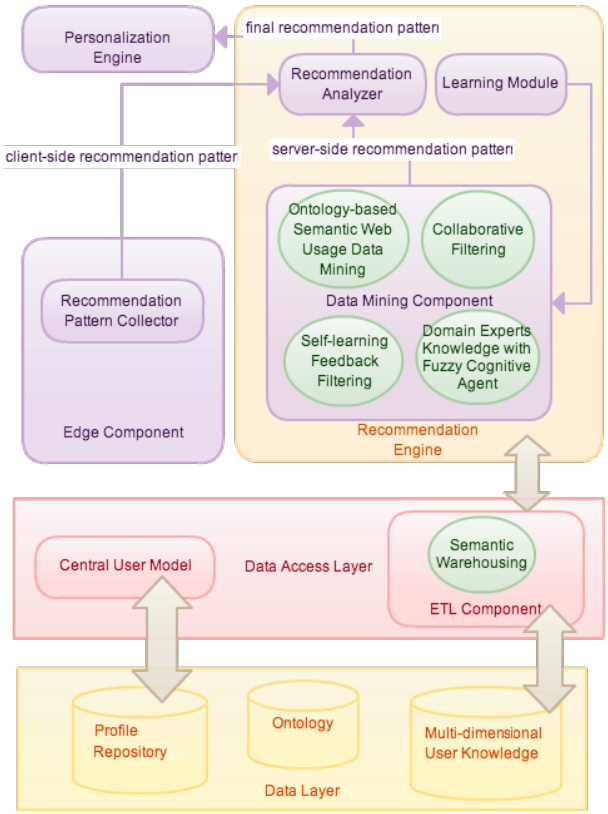

Figure 8: Architecture of the Recommendation Engine

7.1. Recommendation Engine

From the server-side perspective, we deployed four ontology-based technologies in the *Data Mining Component* to enhance expectation and recommendation capability (see Figure 8). They are all built on distributed systems which perform complex computing processes with large data sets. An *Ontology-based Semantic Web Usage Mining* technology, as the basis of the full recommendation process, aims to discover insights and patterns in the usage data based on the ontology view, which plays a dominant role in this Recommendation Engine. Personal usage ontology produced by this algorithm can be used widely for other recommendation methods. *Collaborative Filtering* is applied to gather useful information from user groups with similar characteristics, based on respective visiting patterns. Furthermore, *Self-learning Feedback Filtering* is used to examine and optimize recommendation outputs based on user feedback. A *Learning Module* (as an auxiliary) is installed to assist with the implementation of this learning algorithm. Lastly, expert knowledge is also useful for offering constructive recommendations by a *Domain Expert Knowledge with Fuzzy Cognitive Agent*. In particular, an *Online Analytic Processing (OLAP) Data Cube* technique is applied to capture useful information from a data warehouse to enhance the manipulation of a large set of big data. These technologies are incorporated to construct a powerful recommendation solution.

7.2. Ontology-based Semantic Web Usage Data Mining

Mining web usage data for the Semantic Web has recently become an ongoing approach to associate all web usage logs [23]. Web usage mining aims to discover insights and patterns about the meaning of web resources and their usage [33]. A *Semantic Web Usage Mining* approach in this framework periodically generates patterns for user usage, taking advantage of ontology-based semantic design.

One of the major tasks for web usage mining is to convert the *Web Usage Log* to a *Concept Usage Log* that preserves usage patterns by a *Timelist-Based Hierarchical Concept Usage Generation Algorithm* (see Section 7.3). The *Personal Web Usage Lattice (PWUL)* is generated by an Ontology-based Learning Algorithm (see Section 7.4) using semantic concept web usage logs as inputs. By collecting and grouping all PWUL, a *Global Web Usage Lattice (GWUL)* is established to represent all periodic pattern-based

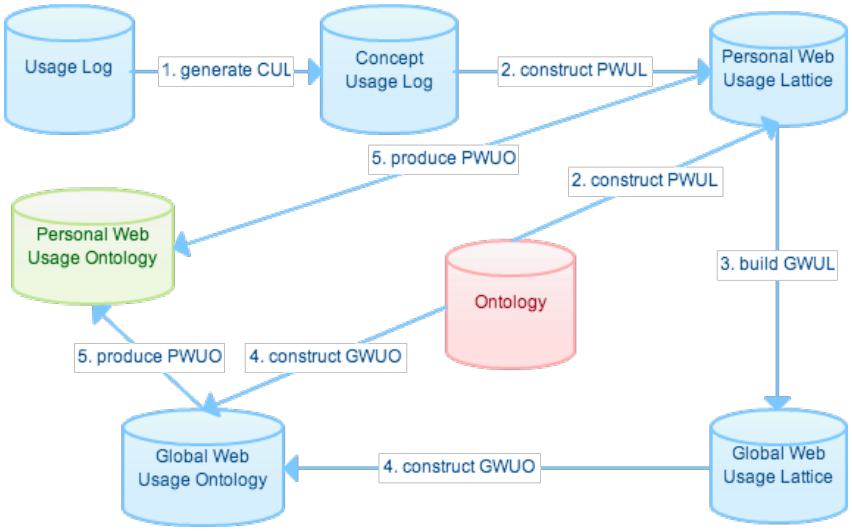

Figure 9: Ontology-based Semantic Web Usage Data Mining

web access activities, and the hierarchical relationships between these activities [23]. OLAP with Multi-dimensional User Knowledge (see Section 7.6) is useful for performing this aggregation process efficiently. The GWUL is then used to generate the *Global Web Usage Ontology (GWUO)* by mapping the global ontology to access activities and hierarchical relationships using an *Ontology Mapping Algorithm* (see Section 7.5). Subsequently, the PWUL maps with the GWUO to produce the *Personal Web Usage Ontology (GWUO)*, which demonstrates connotative relations for all user-performed activities and is basic user knowledge that can be applied to promoting further data mining processes. The steps involved in Ontology-based Semantic Web Usage Data Mining are listed as follows and are illustrated in Figure 9.

1. Transform Usage Log to Concept Usage Log
2. Construct Personal Web Usage Lattice (PWUL) by mapping Concept Usage Log and the Ontology
3. Aggregate all Personal Web Usage Lattice (PWUL) to build a Global Web Usage Lattice (GWUL)
4. Map Global Web Usage Lattice (GWUL) with the ontology to generate a Global Web Usage Ontology (GWUO)
5. Produce Personal Web Usage Ontology (PWUO) by mapping Personal Web Usage Lattice (PWUL) and Global Web Usage Ontology (GWUO)

7.3. Timelist-Based Hierarchical Concept Usage Generation Algorithm

To generate the concept usage, fundamental semantic classification algorithms should be first applied. A generally applicable method to discover the taxonomic of usage classification is *Hierarchical Clustering*, which exploits similarities for usage concepts to generate the hierarchy of items [37]. We need to perform morphological processing in order to extract data from text [37].

Additionally, a set of periodic access activities are discovered by analyzing the Semantic Web usage log. Mining periodic patterns from time relevant usage data is also an important task, and *Progressive Timelist-Based Verification* is a two-phase algorithm to uncover periodic patterns in sequence [38]. The last step is to map the periodic access activities with the hierarchical clustering tree, using the *Ontology Mapping Algorithm* (see Section 7.5). Figure 10 and Algorithm 1 illustrate how to generate Concept Usage in this framework.

Algorithm 1 Timelist-Based Hierarchical Concept Usage Generation Algorithm

```
Require: usage records: TS = {{R_1, d_1}, {R_2, d_2}, ..., {R_n, d_n}}
  ▷ minimum required duration
  minimum duration: d_m
  l = lengthofTS
  clusterResult = null
  outputList = [l]
                                                        ▷ use Hierarchical Clustering
  for t ∈ TS do
                                                        ▷ use term extraction
      terms = termExtraction(t)
      hierarchicalClustering(terms, clusterResult)
  end for
                                                        ▷ for each usage record
  for i = 1 → l do
                                                        ▷ for each usage item t in one usage record
      for t ∈ TS[i − 1] do
          for j = 1 → l do                              ▷ check whether an event contains valid segments
              if PeriodicyCheck(t,l,j) && t.d < d_m then Append t to outputList
              end if
          end for
      end for
  end for
                                                        ▷ invoke Ontology Map function
  return Map(outputList, clusterResult)
  function PeriodicyCheck(t,l,p)
      ifValid = false
                                                        ▷ initalize a data structure seq
      for i = 1 → p do
          seq[i − 1].LP = −1;
          seq[i − 1].SP = −1;
      end for
      for j = 1 → l do
          pos = j%p
          if j − seq[j − 1].LP == p then
              seq[j − 1].LP = j
              continue
          else
              seq[j − 1].LP = j
              seq[j − 1].SP = j
          end if
          if seq[j − 1].LP − seq[j − 1].SP >= p then
              ifValid = true
          end if
      end for
      return ifValid
  end function
```

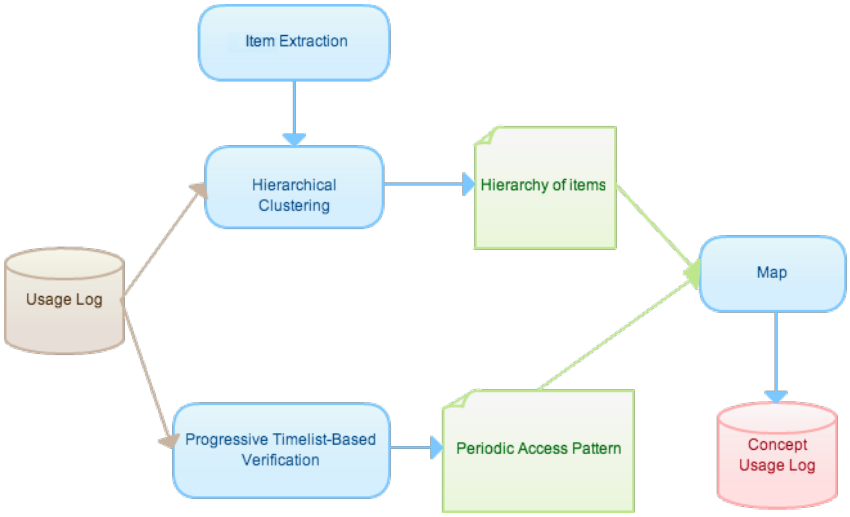

Figure 10: Conceptual Diagram of Timelist-Based Hierarchical Concept Usage Generation Algorithm

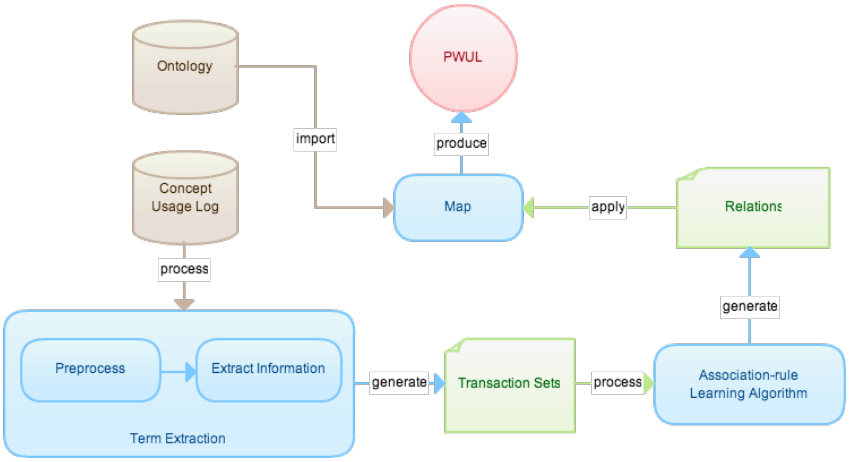

Figure 11: Conceptual Diagram of Ontology-based Learning Algorithm

7.4. Ontology-based Learning Algorithm

The Semantic Web relies heavily on the ontology by performing comprehensive and transportable machine understanding [37]. In the Ontology-based Semantic Web Usage Data Mining process described above, the mapping function is critical for achieving machine learning to construct the PWUL. Several ontology-based learning algorithms can be applied for this. In this framework, we implement multi-strategy learning algorithms to produce results.

The global ontology, as the basic knowledge about the personalization domain, is firstly imported for this ontology-based learning. In the extraction phase, lexical entries are captured, based on the Concept Usage Log. In this step, to contract PWUL, we need to preprocess the Concept Usage Log with data cleaning, user identification, etc. and extract the most relevant information [39]. This process is carried out by eliminating irrelevant items and assigning duration (d) to each concept usage record $UR = \{R_1, R_2, ..., R_n\}$ to generate transaction sets (TS).

$$UR = \{R_1, R_2, ..., R_n\} \Rightarrow TS = \{\{R_1, d_1\}, \{R_2, d_2\}, ..., \{R_n, d_n\}\} \quad (1)$$

The third step is to apply association-rule learning algorithm to discover the relations between transaction sets [37]. Lastly, we map the relations with the ontology using the *Ontology Mapping Algorithm* (see Section 7.5) to produce the PWUL. The whole procedure is illustrated in Figure 11.

7.5. Ontology Mapping Algorithm

An *Ontology Mapping Algorithm* is a fundamental function widely used in *Ontology-based Semantic Web Usage Data Mining*. Since hierarchical taxonomies are common characteristics in ontologies, the purpose of the mapping algorithm is to find one-to-one correspondence between the taxonomies of two given ontologies [3]. Given each concept node in one taxonomy, this algorithm is applied to find the most similar concept (i.e., having the closest meaning) in the other taxonomy. For similarity computing, *joint probability distribution* is considered to be a good measure for this approach.

An ontology which provides the knowledge relations and structure for a mapping outcome is mapper N. Another ontology receives the mapping process called de-mapper D. The solution is to generate a new ontology R that uses the structure of the ontology N to describe the ontology D. Thus, the form of the mapping function is:

$$R = Map(N, D) \tag{2}$$

The pseudo-code for this algorithm is presented in Algorithm 2.

Algorithm 2 Ontology Mapping Algorithm

Require: ontology N, ontology D and acceptable similarity sim_m

```
function MAP(N,D)
    for n ∈ N do
        for d ∈ D do
            if jointProbabilityDistribution(n, d) > sim_m then
                n.nodes.push(d)
            end if
        end for
    end for
    for n ∈ N do
        if n.nodes.length == 0 then
            remove n from N
        end if
    end for
    return N
end function
```

7.6. Semantic Warehousing with Multi-dimensional User Knowledge

Semantic Web has become a new environment for data warehousing. Since the technologies implemented in the *Recommendation Engine* are all ontology-based, how to fetch and manipulate semantic data and ontologies for the Multi-dimensional User Knowledge database is a significant challenge. *OLAP* is usually associated with the traditional data warehouse for mining data using multiple taxonomies [40, 33]. The ontology-based process has now become mature enough to offer different technologies and tools to generate semantic-related data warehousing. Based on the multi-dimensional feature, a key step is to import OWL/RDF data into a fact table or a dimension table in a data warehouse. We can utilize XML data in an existing database system and load all semantic data into the data warehouse. Significantly, this is critical for generating GWUL, which indicates global patterns for all users in the *Ontology-based Semantic Web Usage Data Mining*. Based on semantic warehousing, Nebot and Berlanga [40] advanced a solution that combines data warehousing and OLAP techniques in the semantic area. It could be helpful for this framework to use the Semantic Web to implement ontology-based analysis and recommendation.

7.7. Collaborative Filtering

Collaborative filtering aims to learn user preferences and make personalization recommendations based on community data [36]. Recommender systems based on collaborative filtering have been the most popular practice in recent times [33]. This method (see Algorithm 3) is applied to gather user

groups with similar characteristics from respective visiting patterns [41]. The similarities among different users providing a similar service content can be used to make personalization recommendations [4].

The Artificial Intelligence (AI) algorithm adopted for collaborative filtering is called Intelligent Adaptive-Support Association Rule Mining (IASARM) [8]. The AI techniques in this filtering take advantage of other users behavioral activities by promoting responsible discovery based on intelligent measures of the similarities between them [42]. The significant improvement delivered by this algorithm is that it reduces the running time and achieves good recommendation performance.

Das et al. [36] advanced an efficient collaborative filtering solution for Google News. A mix of memory-based and model-based algorithms are proposed to implement the recommendation. Memory-based algorithms model users and make up ratings predictions for users, based on past news ratings.

To implement collaborative filtering, the cosine similarity of user preferences and recommendation patterns is typically calculated first. Subsequently, if the cosine similarity between two user profiles and rating vertexes is sufficiently high, it is determined that patterns with high-level feedback rating can be used for the current user.

Algorithm 3 Collaborative Filtering

Require: user preference: $U = \{U_1, U_2, ..., U_n\}$,
▷ P_n indicate recommendation pattern and f_n is feedback rating value
recommendation patterns: $P = \{\{P_1, f_1\}, \{P_2, f_2\}, ..., \{P_n, f_n\}\}$
current user preference: U_c
current recommendation pattern: P_c
minimum acceptable cosine similarity for user preference:$minSim$
minimum acceptable weighted similarity:ws

▷ for each user preference
for $UP \in U$ **do**
▷ for each user preference, calculate cosine similarity
$preferenceSim = cosineSimilarity(U_c, UP.U_i)$
if $preferenceSim > minSim$ **then**
▷ for each recommendation pattern, calculate cosine similarity
for $RP \in P$ **do**
$patternSim = cosineSimilarity(P_c, RP.P_i)$
▷ when cosine similarity by feedback score larger than minimum acceptable weighted similarity
if $patternSim \times RP.f_i > ws$ **then**
$MergePattern(P_c, RP.P_i)$
end if
end for
end if
end for

7.8. Self-learning Feedback Filtering

To improve recommendation results, analyzing user feedback and rating a set of provided recommendations is a feasible and novel approach [16] as it allows a whole system to gain the self-learning ability. Most personalization systems represent user feedback as an n-dimensional vector of ratings of each item [16]. The feedback given by users can further optimize corresponding patterns. The *Self-learning Feedback Filter* is responsible for analyzing feedback and updating patterns.

Usually, feedback is scaled into scores. For this framework, we need to collect explicit feedback and implicit feedback. If the user has already made a rating on a recommendation output, the feedback is the rating result. If the user does not provide a rating, we generate a feedback score according to the usage log. In the data system, a matrix is stored with pair-value sets for recommendation pattern and feedback rating.

7.9. Domain Experts Knowledge with Fuzzy Cognitive Agent

Most traditional personalization recommendation systems mainly focus on what an application can extract and recommend general preferences based on a user's historical data [7]. However, this does

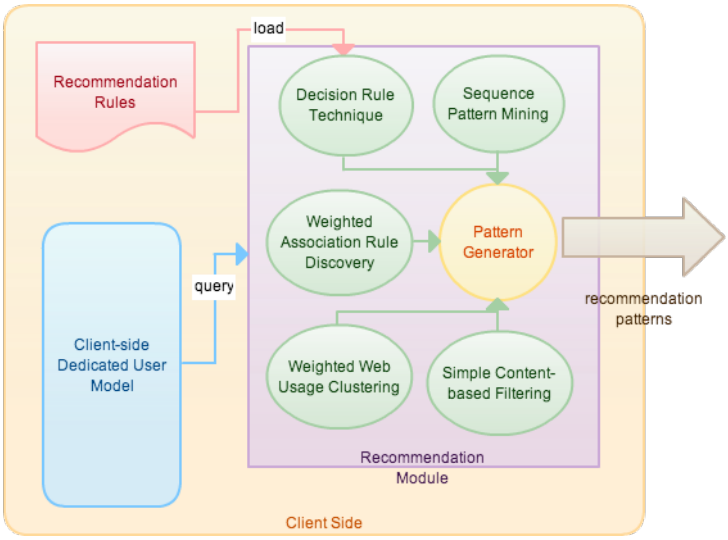

Figure 12: Architecture of the Client-side Recommendation Module

not ensure that there are sufficient recommendations. This system may also rely upon domain experts knowledge which extends the capability of decision making for personalization. Thus, in this framework, a fuzzy cognitive agent is desirable to overcome the impersonal nature of the integrated recommendation system and treat each user individually [7].

The fuzzy cognitive agent is designed to represent personalization knowledge via extended fuzzy cognitive maps. This agent provides recommendation patterns by analyzing a user's current personalization preferences, other user's common preferences and expert knowledge using specific fuzzy cognitive algorithms [7]. The fuzzy cognitive maps learn a user's usage ontology from the most recent cases of other users to help the system to determine the recommendation patterns.

The algorithm model comprises two types of objects: concept and weight. A concept means a node in a personalization map. These concepts are connected by weights, which indicate the effects in relationships between the concepts [7]. This model can build on the PWUO. The directed graph of the model is based on the Fuzzy Cognitive Maps (FCM) theory, which involves case-based reasoning, self-organization map and neural network learning.

7.10. Client-side Recommendation Module

To take full advantage of modern web browsers, the *Client-side Recommendation Module* (see Figure 12) performs local recommendation tasks and generates an initial recommendation pattern by *Content-based Filtering*, *Association Rule Discovery*, *Sequence Pattern Mining* and *Decision Rule Technique* [14]. It is responsible for discovering an appropriate recommendation set with related algorithms and technologies on the client side. First, the *Content-based Filtering* filters frequent usage items from the usage log. Next, the *Association Rule Discovery* finds relationships/connections between resources accessed by users. Then, *Sequential Pattern Mining* processes sequential user data to predict the prospective personalization services. Last, the *Decision Rule Technique* provides a methodology to perform recommendations by adopting a rule-based repository. All four technologies collaborate sequentially to build up a client-side recommendation pattern.

8. Personalization Engine

Since the context of users changes frequently, this framework needs to adapt to provide users with value-added services [43]. A *Personalization Engine* is one major component capable of creating a unique personalized configuration for each user by tailoring cloud services for users. The procedure comprises the following six steps. First, the Personalization Engine retrieves the user profile information and preferences, loads the personalization rules and accepts the recommendation patterns. Second, according to the personalization rules, the system initializes a personalization configuration. Third,

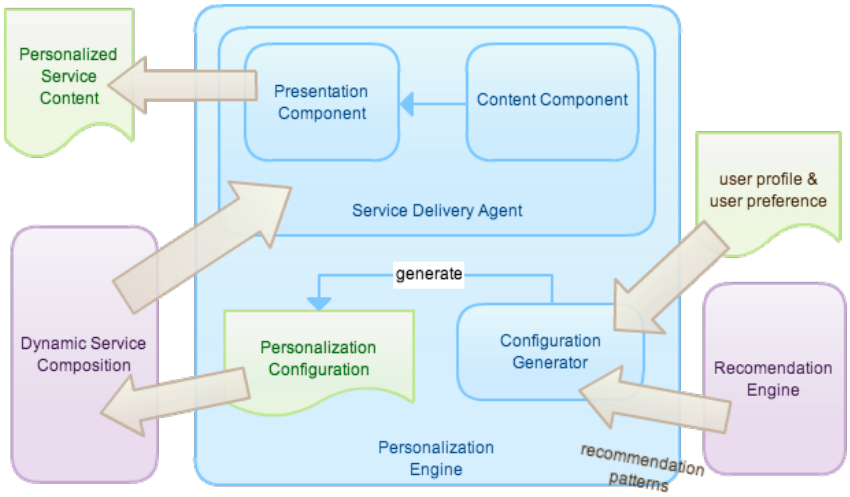

Figure 13: Architecture of the Personalization Engine

with regard to the analysis of the user's profile and preferences, customization fields are incorporated in the configuration, thus constructing the initial personalization configuration. Fourth, taking the recommendation patterns into consideration, defined functions are used to map the current configuration and calculate the weight of each personalization field. A logic adaption is applied in this process, after which the personalization configuration is finalized. A *Configuration Generator* is responsible for constructing the personalization configuration and submitting it to the *Dynamic Service Composition* for selecting services and generating service configurations. Once raw personalized service data are accepted in step five, a *Service Delivery Agent* will decorate the raw service data received from the cloud service and deliver the user-friendly personalized service content to users in step six. The delivery process is performed automatically and independently. Figure 13 gives an overview of the architecture of the Personalization Engine.

8.1. Personalization Configuration

Personalization configuration assists in generating personalized cloud services automatically and dynamically for users, using the *Dynamic Service Composition* component. In this framework, the personalization rules contain a configuration template. A personalization configurations is built upon the template. The configuration consists of the settings of all relevant personalization fields that can be used to generate service parameters. It is a vital source for performing service customization adaption. OWL/RDF semantic data is suitable for rendering the configuration.

8.2. Service Delivery Agent

This agent deals with delivering formatted personalized service content to users. To adjust a presentation to meet user requirements and device possibilities, a service delivery agent is deployed in this framework. This agent is composed of two parts: the *Content Component* and the *Presentation Component*. The Content Component is used to adjust data content to fulfill personalization requests. The role of the Presentation Component is to demonstrate a satisfactory presentation format for delivering content [44].

The major technology in the Presentation Component is based on Extensible Stylesheet Language Transformations Language (XSL-T). This specialized transformer uses XSL templates to transform XML content from the Content Component to a suitable output by using different channel types such as HTML page, JSON data or other media for users [44]. Since a user may use multiple devices for service personalization, a *Template Adaptor* is used to switch different templates for the different devices connected. In this design, a *Template Container* is responsible for storing and managing XSL templates as output patterns for performing XML/XSL transformations [44]. Usually, the service content is transferred by a service-oriented approach or technology to the browsers [43].

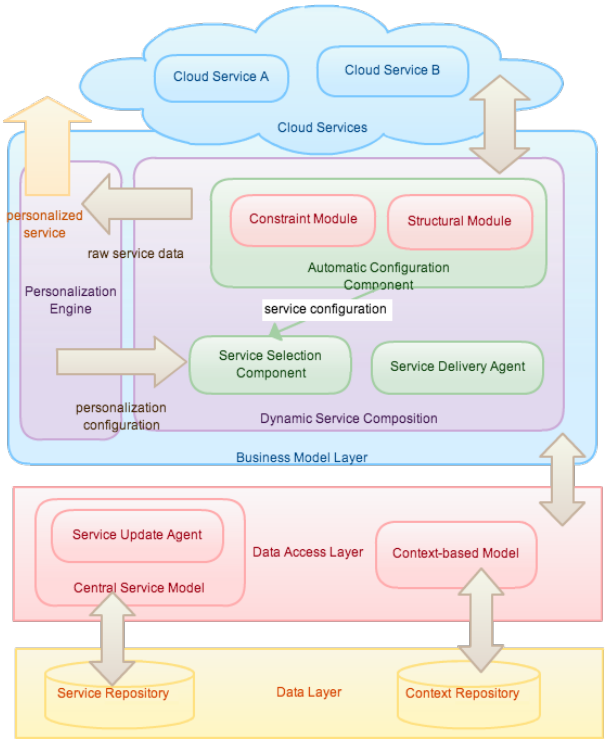

Figure 14: Architecture of the Dynamic Service Composition

9. Dynamic Service Composition

The dynamic composition of atomic cloud services in generating a composite service and selecting cloud services with regard to user preference are two major problems in creating personalized service output [10]. Building composite web services can significantly enhance interoperability and collaboration among users [9], and the seamless composition of web services for this framework has enormous potential for streamlining personalization processes and the integration of the Semantic Web, data mining and cloud services [9].

In setting up a personalization framework, the dynamic and personalized composition of cloud services is a critical element of the architecture design. The composition of cloud services refers to those services that can be customized automatically to meet the needs and preferences of individual users [9]. As stated by Kumanayaka and Ranasinghe, a plug-play service composition scheme can significantly reduce the effort involved in service provisioning for interactive, flexible and collaborative personalized cloud services. It is desirable to seek a combination of existing composite services to fulfill the request of a user's personalization configurations which adopt fields from recommendation patterns and user preferences [9]. The component used to handle mapping personalization configurations with semantic service annotation is called *Dynamic Service Composition* (see Figure 14).

In the Dynamic Service Composition component, the *Service Selection Component* is used to choose appropriate cloud services according to personalization configurations. An *Automatic Configuration Component* is applied to generate the configuration for each recommended service. The request for personalized services will later be submitted to cloud services with selected services and related customized configurations. The raw service data received from the cloud service will be transferred to the *Personalization Engine* for delivery. Finally, personalized services will be supplied to the user by the Personalization Engine.

9.1. Service Directory and Service Update Agent

SaaS-based personalization services dynamically select and compose user-specific services from the pre-structured service templates offered by multiple providers [11]. The service template in this design is called the *Service Directory*, where a set of service profiles are stored in RDF/XML format. In the service

directory, a service profile repository is implemented to advertise and discover services in a rich semantic way. The service profile describes the incorporated functionalities of the service by public interface [10].

In this design, a *Service Update Agent* dynamically and automatically imports and updates service metadata from service providers into the service directory, stored as RDF data. The Service Update Agent automatically obtains the updated service profile through the registration interface for each cloud service. Sending new profile data to the service directory and updating this data is its main task. This agent may also need to resolve ambulation issues from the service directory [26].

9.2. Steps of Automatic Service Provision

Automatic service provision involves four steps. The first step is to determine whether a personalization configuration can be satisfied. In this step, a personalization configuration may need to be decomposed into preferred sub-goals that can be solved by a set of cloud services [45]. The second step is to filter services from a selected service set. When two services can perform the same task, this component needs to determine which service will be a better choice according to other typical parameters, such as execution cost or load time [45]. The Service Selection Component also needs to measure the quality of service by averaging the difference between what a service does and what it advertises [10]. In the third step, the primary task is to set up a service configuration that is automatically processed by an *Automatic Configuration Component*. The last step is to claim a group of services as the best personalized services available to users.

9.3. Automatic Configuration Component

The *Automatic Configuration Component* consists of the defining service parameters from the service directory and confirming the constraint rules by comparing the requests and constraint requirements of the selected services in a *Constraint Module* [46]. To provide a mechanism to satisfy the personalized service requirements automatically, Sam, Boucelma and Hacid [46] adapt a paradigm for web services which enables service providers to avoid the weighty task of building a configuration for each user. The role of this Automatic Configuration Component is to automatically transform services published in a service directory in order to generate suitable configurations to fulfil client's needs [46].

The automatic configuration component is made up of two modules: a structural module and a constraint module [46]. The structural module is defined by the service directory and recommended patterns. The constraint module is used to measure the constraints of input/output and domain. The S-context information stored in the *Context Repository* is invoked to set up the constraint. Service providers publish only one explicit configuration for a given service; the others are dynamically and automatically inferred from the service directory [46]. The structural module and constraint module work together to build each personalized configuration for specified services.

10. Prototype Implementation and Evaluation

In this section, we present a prototype of SaaS Personalization Framework. Based on the basic prototype developed in [14], we implemented a complete prototype application "Personalized Music", to evaluate this framework. This prototype gathers user preferences and recommends music feeds to users based on each user's profile and activities. This is a simple case study, but it is appropriate for evaluating the SaaS Personalization Framework. The implementation works on the Amazon EC2 platform and includes a main web application and a set of cloud-based services.

The scenario can be described as follows. When a user visits this cloud-based web application, the system starts to collect the user's information as a profile. The user identifies their level of interest in music; the higher the level they define, the more possibilities will show on a page. Finally, the prototype produces a set of personalized music lists generated from the cloud services.

10.1. Data Sets and Test Process

We used Last.fm Dataset 1K users [47] as sample data to evaluate the proposed framework. One dataset with access entries of the eight-month period from 1 May 2008 to 31 December 2008 is chosen as training set, which is used to generate recommendation patterns. The remaining group from 1 January 2009 to 4 May 2009 are the test set to evaluate recommendation results by comparing them with individual profiles and usage activities.For validation of our proposed framework, we made use of the following data set:

- 5 User profiles are collected from Last.fm User Dataset.
- 5 User preferences are extracted from Last.fm User Dataset.
- Totally 33,830 usage data are collected and tested from Last.fm Dataset.
- Recommendation rules are identified.
- Service directory is a RDF document to identify personalization services which offer personalized music lists.

The test process for the evaluation was as follows:

i. Five users were selected and their preferences and usage data were extracted

ii. The training sets were extracted from the usage dataset and the user profiles were extracted from the profile dataset. These were then imported into the Multi-dimensional User Knowledge database

iii. The recommendation patterns were generated

iv. An evaluation candidate was extracted from the usage dataset and it was stored in the central repository

v. The prototype was executed and the test results were recorded along the identified benchmarks/evaluation metrics

The amount of data items used in this evaluation is shown as follows:

Table 1: Experiment Data

User#	#Training Set	#Test Set
user_000001	5774	4127
user_000002	8109	2877
user_000003	4304	1018
user_000004	4725	1053
user_000005	1475	368

In our test, we generated recommended songs for each user according to their profile and activities. If one recommended song that has been played by the user in the next 4 months (can find in our test set), we set this recommendation result to be "correct". By means of the evaluation, the statistical records determine how this personalization framework performs in a SaaS environment.

10.2. Performance Measurement

To evaluate the performance of this framework, we conducted various performance measurement experiment and compared this framework with an original mash-up application (BooksHPCLab), a new implementation (BooksHPCLab_revised) [18] and a client-side approach [14]. We use two metrics, Load Time and Reasoning Time, to evaluate the performance of the proposed framework. Load Time identifies how long it takes for the prototype to capture user data and Reasoning Time measures how long it takes the system to generate rule-based personalized recommendations.

Table 2 and Figure 15 illustrate the comparison results between the four approaches for performance measurement. The SaaS Personalization Framework presents relatively low in load time and reasoning time. Over the two semantic recommendation applications, BooksHPCLab and BooksHPCLab_revised, the SaaS Personalization Framework achieved better semantic integration. Comparing with BooksHPCLab and BooksHPCLab revised, the average load time is reduced by 44.8% and 20.4%; and the average reasoning time is reduced by 89.4% and 23.2%. The dramatic improvement in time cost highlights performance evolution. The SaaS Personalization Framework can offer better recommendation outcomes with a strong capability to process a high volume of dataset, incurring relative less load time and reasoning time.

Table 2: Performance Comparison

Approach	Average Load time (millisecond)	Average Reasoning Time (millisecond)
BOOKS@HPCLAB	138.5	79.75
BOOKS@HPCLAB_REVISED	96	11
SaaS Personalization Framework	76.45	8.45

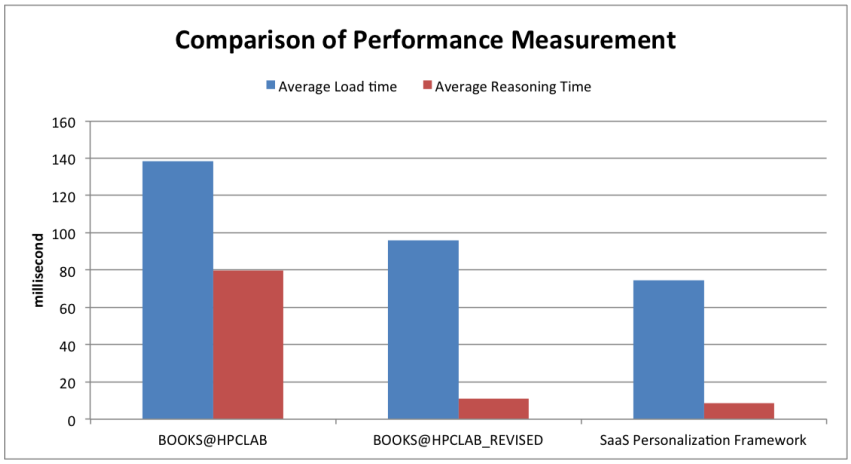

Figure 15: Performance Comparision

10.3. Recommendation Metrics

A set of parameters are used to identify recommendation performance [16].

- Mean absolute error (MAE). The average absolute error is a common accuracy measurement in recommendation systems. It compares forecasts with actual outcomes. We calculate the MAE as:

$$MAE = \frac{\sum_{i=1}^{m_r} \sum_{x_j \in Ri \cap Hi} abs(r_i(x_j) - p_i(x_j))}{\bigcup_{i=1}^{m_r} R_i \ capH_i} \tag{3}$$

- Pearson's correlation coefficient, r. Rather than measuring the error in prediction, this metric provides a measure of the linear correlation degree between the predicted ratings and actual ratings. The higher the value, the better recommendation match. This metric is calculated as follows:

$$MR = \frac{\sum_{i=0}^{m_r} \sum_{x_j \in Ri \cap Hi} (r_i(x_j) - \bar{p})}{\sqrt{\sum_{i=0}^{m_r} \sum_{x_j \in Ri \cap Hi} (r_i(x_j) - \bar{r})^2} \sqrt{\sum_{i=0}^{m_r} \sum_{x_j \in Ri \cap Hi} (p_i(x_j) - \bar{p})^2}} \tag{4}$$

- Mean rank (MR). This metric algorithm generates a group of recommendations which is ranked on the basis of forecasting rating. The higher the MR value, the better the recommendation set. MR is calculated as follows:

$$MR = \frac{\sum_{i=1}^{m_t} \sum_{x_j \in Ri \cap Hi} (100 - rank_i(xj))}{m_t} \tag{5}$$

- Precision (P), recall (R) and F1. To use these measurements, we categorized items into those "liked" and "disliked" by the user. Hence, when an item has been marked as "liked", true positives (TPs) will be defined. These three metrics are calculated as follows:

$$P = \frac{TP}{TP + FP'} \tag{6}$$

$$R = \frac{TP}{TP + FN'} \tag{7}$$

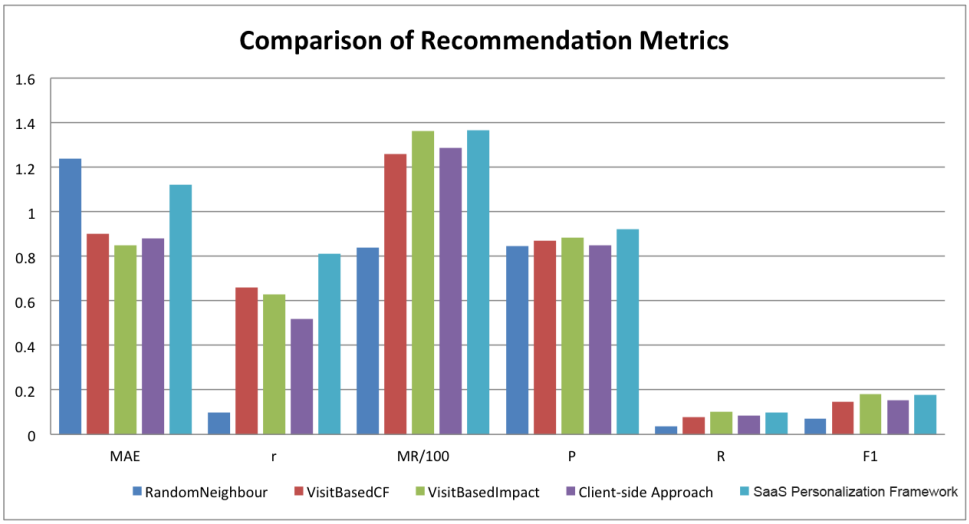

Figure 16: Comparison of Recommendation Metrics

$$F1 = \frac{2PR}{P+R} \tag{8}$$

RandomNeighbour, VisitBasedCF and VisitBasedImpact are three major algorithms highlighted in a personalization solution [16]. They all provide a novel approach to generate user profiles. The test results (Table 3 and Figure 16) demonstrates that the recommendation metrics of our prototype provided significant benefits from SaaS Personalization Framework compared to the RandomNeighbour, the VisitBasedCF, the VisitBasedImpact and the client-side approach. It is clear that, by introducing the semantic model and multi-dimensional user knowledge, the SaaS Personalization Framework generates more precise recommendation patterns and improves the personalization performance.

Table 3: Comparison of Recommendation Metrics

Approach/Framework	MAE	r	MR	P	R	F1
RandomNeighbour	1.24	0.098	83.87	0.845	0.036	0.07
VisitBasedCF	0.9	0.66	125.75	0.87	0.079	0.145
VisitBasedImpact	0.85	0.631	36.37	0.882	0.101	0.181
Client-side Approach	0.88	0.52	128.45	0.85	0.085	0.154
SaaS Personalization Framework	1.12	0.81	136.5	0.92	0.098	0.177

10.4. Relative Entropy Measure

Relative entropy measure is another metric for personalization evaluation. It identifies the performance of a user model with implicit ontologies and explicit ontologies [17]. For measuring performance, given a learned usage pattern, the relative entropy measure defines the extent of the recommendation error is based on a user's profile. The lower the value of relative entropy, the more effective the algorithm applied to usage mining. This is calculated by applying the following relation recursively.

$$D'(r) = D(r) + \sum_{\forall k} p(k|r)D'(k) \tag{9}$$

where r is a common node of the two ontologies, k is an immediate child node of r, D is the entropy of the current node, D' is the relative entropy of the current node, and $p(k|r)$ is calculated by the following equation.

$$p(k|r) = \frac{W_k}{\sum_{W_i}} \tag{10}$$

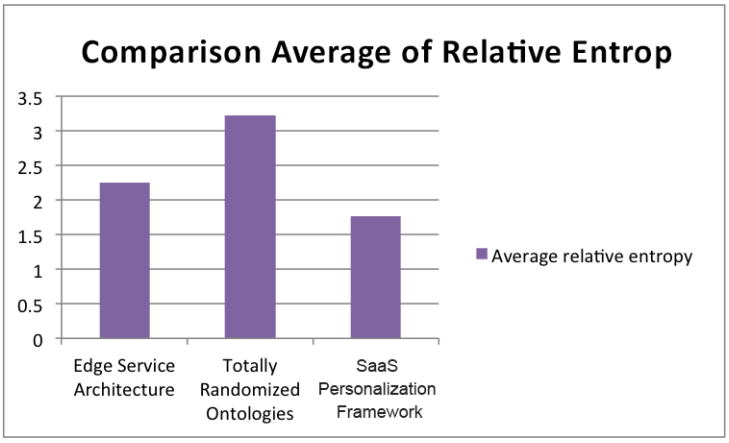

Figure 17: Comparison of Average Relative Entropy

where W_k is the score associated with the child node, W_i is the score with one child node in the common node.

From Table 4 and Figure 17, the average relative entropy in this prototype is 1.76 which illustrates a dramatic improvement in the design of the user model. Totally Randomized Ontologies is a method to generate personalized content with random algorithm, presenting ineffective performance. The Edge Service approach is adopted to use an additional module to handle a caching and learning algorithm to optimize recommendations. Compared to other approaches, including edge service architecture, randomized ontologies and the client-side approach, SaaS Personalization Framework provides a better solution for building a user's preference. It is apparent that multiple techniques in this solution boost recommendation performance. Furthermore, a precise user profile is the cornerstone for accurate personalization services. This test result manifests that an integrated framework could deliver incredible potential.

Table 4: Comparison of Client-side Profile Utility

Approach	Average relative entropy
Edge Service Architecture	2.25
Totally Randomized Ontologies	3.22
SaaS Personalization Framework	1.76

11. Conclusion and Future Works

In this paper, we proposed a complete personalization framework for SaaS-based cloud services. The increasing need for personalization for users has inspired our approach to create an effective, efficient, flexible and integrated personalization framework. The construction of this framework is based on a number of prior research outcomes, including client-side personalization, semantic approach with ontology-based semantic integration, central user modeling, data mining technologies and recommendation engine, and dynamic cloud service composition.

Multiple systems have a range of different requirements, yet require a framework that will support personalized services. It is therefore not possible to build one solution for all systems. This framework does not provide a strict design; rather, it offers a complete, guided and flexible solution for service personalization. The solution can be adapted from different fields in different SaaS projects.

Our future work aims (i) to consider more optimized and flexible solutions, (ii) develop a personalization standard with multiple-level solutions for a personalization framework, (iii) adaptation of this framework for IaaS, and (iv) investigating the security issues of the framework.

References

[1] C. Baun, M. Kunze, J. Nimis, S. Tai, Cloud Computing: Web-Based Dynamic IT Services, Web-Based Dynamic IT Services, Springer, 2011.

[2] L. Yu, A Developer's Guide to the Semantic Web, IT Pro, Springer Berlin Heidelberg, 2011.

[3] A. Doan, J. Madhavan, R. Dhamankar, P. Domingos, A. Halevy, Learning to match ontologies on the semantic web, The VLDB Journal 12 (4) (2003) 303–319.

[4] C. Wei, W. Sen, Z. Yuan, C. Lian-Chang, Algorithm of mining sequential patterns for web personalization services, ACM SIGMIS Database 40 (2) (2009) 57–66.

[5] Z. Maamar, G. AlKhatib, S. K. Mostefaoui, Context-based personalization of web services composition and provisioning, in: Euromicro Conference, 2004. Proceedings. 30th, 2004, pp. 396–403.

[6] M. Rahman, H. Longe, O. Abass, F. Ahmad, Enterprise data integration towards web service personalization, in: Computer and Electrical Engineering, 2008. ICCEE 2008. International Conference on, 2008, pp. 715–720.

[7] C. Miao, Q. Yang, H. Fang, A. Goh, A cognitive approach for agent-based personalized recommendation, Knowledge-Based Systems 20 (4) (2007) 397–405.

[8] W. Lin, S. Alvarez, C. Ruiz, Efficient adaptive-support association rule mining for recommender systems, Data Mining and Knowledge Discovery 6 (1) (2002) 83–105.

[9] D. Zhang, M. Chen, L. Zhou, Dynamic and personalized web services composition in E-business, Information Systems Management 22 (3) (2005) 50–65.

[10] O. Kumanayaka, D. N. Ranasinghe, Personalized web services creation via dynamic composition, in: Industrial and Information Systems, First International Conference on, 2006, pp. 408–413.

[11] P.-Y. Hsueh, R. Lin, M. J. H. Hsiao, L. Zeng, S. Ramakrishnan, H. Chang, Cloud-based platform for personalization in a wellness management ecosystem: Why, what, and how, in: Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2010 6th International Conference on, 2010, pp. 1–8.

[12] H. Guo, J. Chen, W. Wu, W. Wang, Personalization as a service: the architecture and a case study, in: Proceedings of the first international workshop on Cloud data management, CloudDB '09, ACM, New York, NY, USA, 2009, pp. 1–8.

[13] K. S. Gopalan, S. Nathan, C. Teja, A. B. Channa, P. Saraf, G. Shanker, A cloud based service architecture for personalized media recommendations, in: Next Generation Mobile Applications, Services and Technologies (NGMAST), 2011 5th International Conference on, 2011, pp. 19–24.

[14] H. Fan, F. K. Hussain, O. K. Hussain, Semantic client-side approach for web personalization of saas-based cloud services, Concurrency and Computation: Practice and Experience (2014) n/a–n/adoi:10.1002/cpe.3418.
URL http://dx.doi.org/10.1002/cpe.3418

[15] C.-F. Lai, J.-H. Chang, C.-C. Hu, Y.-M. Huang, H.-C. Chao, CPRS: A cloud-based program recommendation system for digital TV platforms, Future Generation Computer Systems 27 (6) (2011) 823–835.

[16] S. S. Anand, P. Kearney, M. Shapcott, Generating semantically enriched user profiles for web personalization, ACM Transactions on Internet Technology 7 (4).

[17] X. Xie, H.-J. Zeng, W.-Y. Ma, Enabling personalization services on the edge, in: Proceedings of the tenth ACM international conference on Multimedia, MULTIMEDIA '02, ACM, New York, NY, USA, 2002, pp. 263–266.

[18] G. D. Solomou, A. K. Kalou, D. A. Koutsomitropoulos, T. S. Papatheodorou, A mashup personalization service based on semantic web rules and linked data, in: Signal-Image Technology and Internet-Based Systems (SITIS), 2011 Seventh International Conference on, 2011, pp. 89–96.

[19] L. Ardissono, L. Console, I. Torre, An adaptive system for the personalized access to news, AI communications 14 (3) (2001) 129–147.

[20] R. Kass, T. Finin, A general user modelling facility, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '88, ACM, New York, NY, USA, 1988, pp. 145–150.

[21] F. Zhang, Z. Song, H. Zhang, Web service based architecture and ontology based user model for cross-system personalization, in: Web Intelligence, 2006. WI 2006. IEEE/WIC/ACM International Conference on, 2006, pp. 849–852.

[22] M. Kleek, H. Shrobe, A practical activity capture framework for personal, lifetime user modeling, in: C. Conati, K. McCoy, G. Paliouras (Eds.), User Modeling 2007, Vol. 4511 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2007, pp. 298–302.

[23] A. Fong, B. Zhou, S. Hui, J. Tang, G. Hong, Generation of Personalized Ontology Based on Consumer Emotion and Behavior Analysis, Affective Computing, IEEE Transactions on 3 (2) (2012) 152–164.

[24] M. Eirinaki, M. Vazirgiannis, I. Varlamis, Sewep: using site semantics and a taxonomy to enhance the web personalization process, in: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '03, ACM, New York, NY, USA, 2003, pp. 99–108.

[25] R. Ghosh, M. Dekhil, Mashups for semantic user profiles, in: Proceedings of the 17th international conference on World Wide Web, WWW '08, ACM, New York, NY, USA, 2008, pp. 1229–1230.

[26] M. Alade Rahman, H. F. Ahmad, H. Suguri, S. Sadik, H. Longe, A. K. Ojo, Supply chain optimization towards personalizing web services, in: Intelligent Systems, 2008. IS '08. 4th International IEEE Conference, Vol. 3, 2008, pp. 19–17–19–22.

[27] J. Song, Y. Bao, J. Shi, A Triggering and Scheduling Approach for ETL in a Real-time Data Warehouse, in: Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on, 2010, pp. 91–98.

[28] P. Vassiliadis, A. Simitsis, New Trends in Data Warehousing and Data Analysis, Vol. 3 of Annals of Information Systems, 3, Springer Science+Business Media, LLC, 2009.

[29] M. A. Naeem, G. Dobbie, G. Webber, An event-based near real-time data integration architecture, in: Enterprise Distributed Object Computing Conference Workshops, 2008 12th, 2008, pp. 401–404.

[30] Oracle, Best Practices for Real-time Data Warehousing (2012) 1–11.

[31] D. M. Tank, A. Ganatra, Y. P. Kosta, C. K. Bhensdadia, Speeding ETL Processing in Data Warehouses Using High-Performance Joins for Changed Data Capture (CDC), in: Advances in Recent Technologies in Communication and Computing (ARTCom), 2010 International Conference on, 2010, pp. 365–368.

[32] N. Henze, D. Krause, Personalized access to web services in the semantic web, in: IN: THE 3RD INTERNATIONAL SEMANTIC WEB USER INTERACTION WORKSHOP (SWUI, COLLOCATED WITH ISWC, Springer, 2006.

[33] B. Berendt, G. Stumme, A. Hotho, Usage mining for and on the semantic web, AAAI/MIT Press, 2004, pp. 461–480.

[34] J. Ng, The personal web: smart internet for me, in: Proceedings of the 2010 Conference of the Center for Advanced Studies on Collaborative Research, CASCON '10, IBM Corp., Riverton, NJ, USA, 2010, pp. 330–344.

[35] B. Mobasher, H. Dai, T. Luo, M. Nakagawa, Effective personalization based on association rule discovery from web usage data, in: Proceedings of the 3rd international workshop on Web information and data management, WIDM '01, ACM, New York, NY, USA, 2001, pp. 9–15.

[36] A. S. Das, M. Datar, A. Garg, S. Rajaram, Google news personalization: scalable online collaborative filtering, in: Proceedings of the 16th international conference on World Wide Web, WWW '07, ACM, New York, NY, USA, 2007, pp. 271–280.

[37] A. Maedche, S. Staab, Ontology learning for the semantic web, Intelligent Systems, IEEE 16 (2) (2001) 72–79.

[38] K.-Y. Huang, C.-H. Chang, Mining periodic patterns in sequence data, in: Y. Kambayashi, M. Mohania, W. W (Eds.), Data Warehousing and Knowledge Discovery, Vol. 3181 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2004, pp. 401–410.

[39] C. Wong, S. Shiu, S. Pal, Mining fuzzy association rules for web access case adaptation, in: Proceedings of the Workshop Program at the Fourth International Conference on Case-Based Reasoning, 2001.

[40] V. Nebot, R. Berlanga, Building data warehouses with semantic data, in: Proceedings of the 2010 EDBT/ICDT Workshops, EDBT '10, ACM, New York, NY, USA, 2010, pp. 9:1–9:8.

[41] M. Zhang, Research of personalization services in e-commerce site based on web data mining, in: Computational and Information Sciences (ICCIS), 2011 International Conference on, IEEE, 2011, pp. 438–441.

[42] M. D. Mulvenna, S. S. Anand, A. G. Büchner, Personalization on the net using web mining: introduction, Communications of the ACM 43 (8) (2000) 122–125.

[43] Y. Yang, M. Williams, L. MacKinnon, R. Pooley, A service-oriented personalization mechanism in pervasive environments, in: Web Intelligence, 2005. Proceedings. The 2005 IEEE/WIC/ACM International Conference on, 2005, pp. 132–135.

[44] J. Rykowski, W. Cellary, Virtual web services: application of software agents to personalization of web services, in: Proceedings of the 6th international conference on Electronic commerce, ICEC '04, ACM, New York, NY, USA, 2004, pp. 409–418.

[45] W.-T. Balke, M. Wagner, Towards personalized selection of web services, WWW (Alternate Paper Tracks) (2003) 20–24.

[46] Y. Sam, O. Boucelma, M. Hacid, Dynamic web services personalization, in: E-Commerce Technology, 2006. The 8th IEEE International Conference on and Enterprise Computing, E-Commerce, and E-Services, The 3rd IEEE International Conference on, 2006, p. 86.

[47] O. Celma, Music Recommendation and Discovery in the Long Tail, Springer, 2010.