Deliverable 5.2

Project Title	Next-Generation Hybrid Broadcast Broadband
Project Acronym	HBB-NEXT
Call Identifier	FP7-ICT-2011-7
Starting Date	01.10.2011
End Date	31.03.2014
Contract no.	287848
Deliverable no.	5.2
Deliverable Name	DESIGN AND PROTOCOL: Multimodal Interface and Context Aware Recommendation Engine
Work package	5
Nature	Report
Dissemination	Public
Author	Joost de Wit, Oskar van Deventer (TNO); Mark Gülbahar (IRT); Marcel Raner, Christian Überall (THM); Sebastian Schumann (ST); Gregor Rozinaj, Ivan Minarik (STUBA)
Contributors	Daniele Abbadessa (NEC)
Due Date	30.09.12
Actual Delivery Date	05.10.12

Table of Contents

Executive Summary

This document is the second deliverable of WP 5 of the HBB-NEXT project. It presents designs and first prototype implementations related to personalisation of next-generation hybrid-broadcast-broadband television. It provides a current (September 2012) view of the status of the designs, prototypes, integrations and demonstrations towards a central use case.

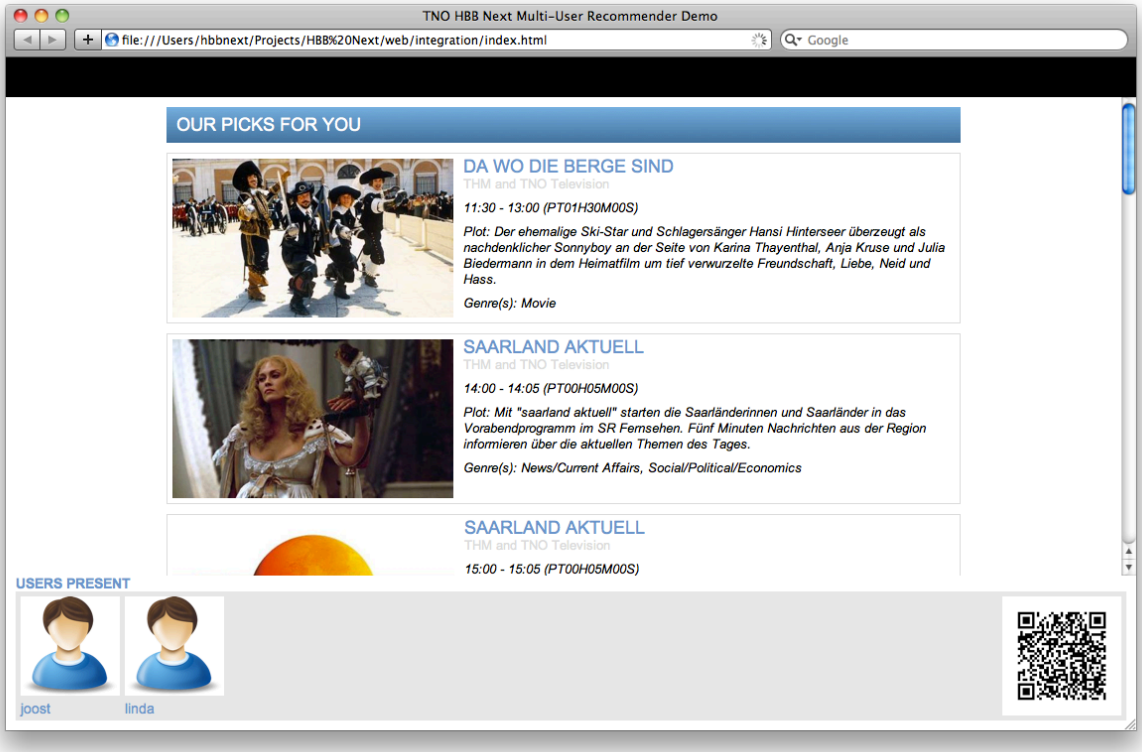
Use case: The central use case of WP5 is described as follows. One person watches television and retrieves a list of content recommendations. Then a second person enters the room, and he is also identified by the system. Subsequently, the recommendation list changes, tailored to the taste of the two persons together.

Design: A preliminary system design is presented, based on the following components: algorithm implementations to calculate recommendations; a metadata enrichment system to obtain multi-sourced metadata for both the calculation of content recommendation and the presentation of recommendations to the users; a coordinating recommendation engine framework; multi-modal and other solutions for identifying the users; a personalisation framework to collect and use personalisation information; a notification framework to support asynchronous interactions between the different components; and Java and RESTful APIs specifying the communication messages between the components.

Prototypes: The following initial prototypes have been realised: a novel and highly efficient collaborative filtering algorithm that calculates the "preference neighbourhood" of a user based on genre preference; a metadata enrichment system that aggregates metadata from various internet sources and outputs it in TV-Anytime format; the coordinating recommendation engine framework (PREF), including a first multi-user algorithm based on the "least misery" strategy; a voice-recognition-based system for identifying users; a QR-code-based user identification system; and a graphical user interface for the user to receive the recommendations, to watch the metadata and to select the content.

Integration: A first integration workshop resulted in successful (still hard-coded) integration of the following components: the filtering algorithm; an enriched set of metadata, ingested as XML file; the recommendation framework; the multi-user recommendation algorithm; the QR-code-based user identification; and the graphical user interface.

Demonstration: Based on the first integration results, the following demonstration was realised. A user opens a browser and selects the HBB-NEXT recommendations URL. The browser displays a generic set of content recommendations and a QR code, inviting the user to identify himself. The user scans the QR code with his mobile device, upon which the browser presents an updated recommendation list, tailored to the personal taste of that user. Then a second user enters, repeats the identification process, and receives an updated recommendation list, tailored to the personal tastes of the two users together. The picture below shows a screenshot of the result. You can also watch a video of the demonstration on the HBB-NEXT YouTube channel: http://www.youtube.com/user/hbbnext.



1. Introduction

This deliverable provides a first system design of a context aware group recommendation engine with a multimodal interface as designed and implemented in work package 5 of the HBB Next project. The "Multi-modal interface for Multi-user Service Personalisation Engine" enabler consists of different software modules. These modules are shown in Figure 1. The modules must work together to realize the use case that is currently central to WP5: One person watches television and retrieves a list of content recommendations. When a second person enters the room and is identified by the system, the recommendation list changes, tailored to the taste of the two persons together.

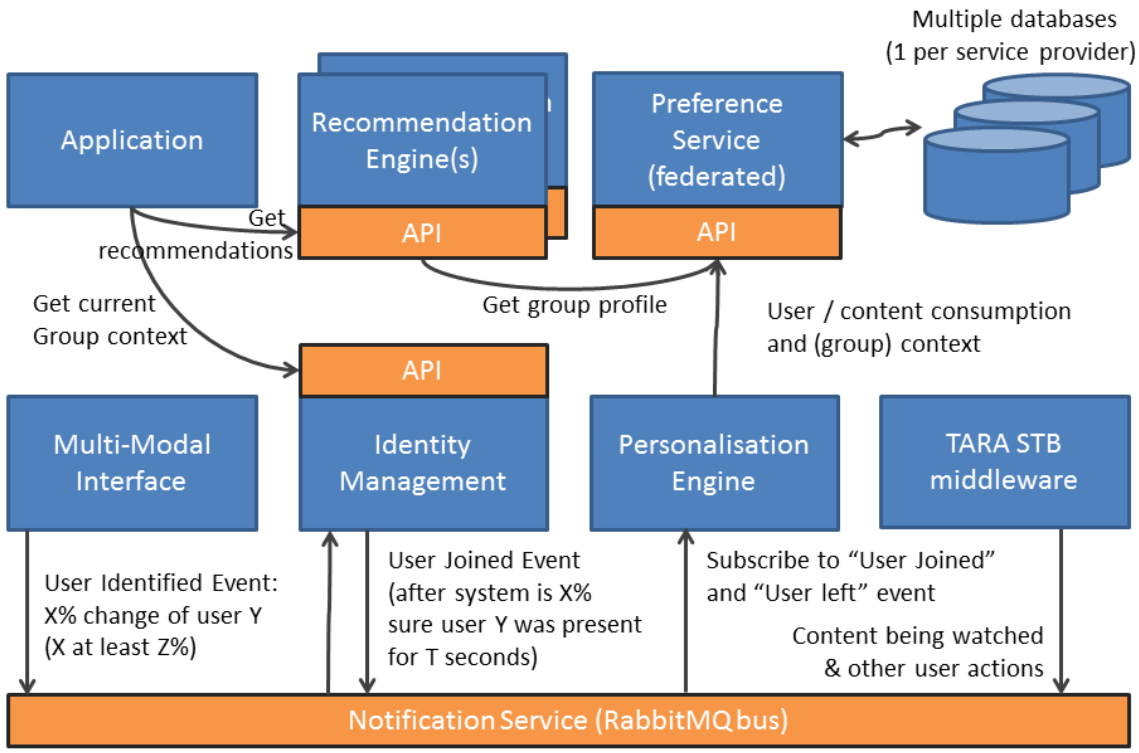


Figure 1: Relationships between WP5 modules

The recommendations that are generated by the Recommendation Engine, are provided to the user(s) through the Application. The identification of the users that are in front of the TV and their interaction with the Application is handled by the Multi modal interface module. Information on the presence of specific users and the content they are consuming is combined by the Personalisation Engine and translated into preferences that are stored by the Preference Service. The Preference Service is the source of user and group

preferences on which the Recommendation Engine is basing its recommendations. The modules communicate with each other through the Notification Service. Note that the Identity Management module, as well as the TARA set-top box middleware are not part of WP5 and are out of this document's scope.

The remainder of this document is structured with respect to the modules presented in Figure 1 and the task they are part of. Chapter 2 describes the modules related to task 5.2: the personalisation engine (section 2.1), notification service (section 2.2) and multi-modal interface (section 2.3). The activities related to task 5.3 (Context-Aware and Multi-User Recommendation Engine) are described in chapter 3. It elaborates on the metadata related modules (sections 3.4-3.6), the first recommendation engine implementation (section 3.7), the preference service (section 3.8) and demo and test applications (sections 3.9-3.11).

2. Multimodal Interface for User/Group-Aware Personalisation in a Multi-User Environment

This chapter subsequently describes the personalisation engine (section 2.1), notification service (section 2.2) and multi-modal interface (section 2.3).

2.1. Personalisation Engine

This chapter describes the personalization engine (PE), an enabler that should allow applications to easily provide context-specific group data on the basis of single user profile information.

The enabler expects applications to request information for certain contexts, e.g., the number of persons in a room, their age etc. The application can determine the context through the Identity Managemant (IdM) enabler (via user/device relation) if it does not know its ID. It will interact with the IdM enabler to retrieve the active users in this context. Through the Profile Management (PM), the personalization engine retrieves the profiles of the active users. In its first version, the PE will provide aggregated information for basic profile parameters. In the future, the API should be extended with the ability to receive more complex algorithms directly in the request and provide personalized information using them.

2.1.1. Design

According to the brief description in Deliverable D6.1.1 [20] the PE shall perform the following functions:

- Retrieve and modify single user profile in the system (via PM)
- Get and set group profile for the active context

The function “Retrieve and modify single user profile in the system (via PM)” is handled via the API of the PM module, which is described in D3.2 [21]. Section 2.1.2 provides the description of the API for managing the group profile.

The group context is an object the personalization engine provides, that contains:

- A context ID (for easy reference in subsequent requests) that contains parameterized information about the group, potential aliases
- All users of a context

The group context data model is shown in Table 1.

Level 1	Level 2	Level 3	JSON Type	JSON Object (preliminary)	Obligation	Ocurrences
Identifier (unique)			object		M	1
	ID		string	id	M	1
	Alias		array	alias	O	n
Users			array	users	M	1
Parameters					O	1
	Amount of users		integer	persons	O	1
	Average Age		number	avg_age	O	1
	Type of group		string	tog	O	1
API version:	1					
Date:	8.9.12					

Table 1: Group context data model

The data model may be extended for future API versions. Especially the parameters showcase in the initial version only the aspired capability of the enabler. In a future version, sample services (incl. service profiles and user parameters) are created and the group profile is extended with service specific personalization.

2.1.2. Protocol

This subsection describes the API of the personalization engine. It provides sample sequence diagrams as well.

Function (Ref. D6.1.1):	Get and set group profile for the active context		
Function:	Get group profile		
Method:	GET	**Resource:**	*/contexts/{contextid}*
Parameters:			

contextid	The id of the group context for which the personalized profile that shall be retrieved.

Function (Ref. D6.1.1):	Get and set group profile for the active context		
Function:	Set group profile		
Method:	PUT	**Resource:**	*/contexts/{contextid}*
Parameters:			
contextid	The id of the group context for which the personalized profile that shall be set.		

Note: Setting the group profile makes only sense once the API is extended. The values that are currently provided, e.g., age, are per-user values. The function makes sense if group profiles are used to set certain values for all users. Instead of setting them per user, the personalization engine receives the group feedback and provisions it to its user profiles.

Function (Ref. D6.1.1):	n/a		
Function:	Retrieve users in a group		
Method:	GET	**Resource:**	*/contexts/{contextid}/users*
Parameters:			
contextid	The id of the group context for which the active users shall be retrieved.		

The response shall contain a body with the users in the group.

The final API description will be made in line with the IdM/PM API described in deliverable D3.2 [21] from WP3.

Note: With regard to context, it shall be clearly understood that the personalization engine in WP5 handles the user profiles in the active context. The IdM from WP3 has a context element as well; it is used however to manage the usual and actual context of a user. While the context assignment in WP3 is rather static and only its activity changes, the context assignment in scope of this WP is dynamic. Further study will clarify this and provide implemented samples to showcase the difference.

Deliverable D3.2 [21] contains a sequence diagram that covers also the personalization engine from this WP.

2.2. Notification Service

The notification service is based on a message queue server to send and subscribe for events of the HBB-NEXT framework. Events in the HBB-NEXT context are to be understood as information describing changes in the context of both system (e.g. “the HBB-NEXT device has tuned to channel RBB”) and users (e.g. “Anna has taken seat in front of the TV”).

2.2.1. Design

Since software implementations of messaging systems exist for a long time, there is no need to design a new / HBB-NEXT specific solution. Instead, an analysis of these existing solutions with respect to usability within the scope of HBB-NEXT use cases and requirements took place. The most well-known solutions are XMPP [16] and RabbitMQ [17]. Since the latter is also used within the Openstack IaaS Cloud Systems [19] and also Javascript libraries for RabbitMQ exist [18], the choice fell onto RabbitMQ.

For first system integration, a RabbitMQ Server [17] was installed on a PC within the home network (e.g. the DLNA Home Gateway). To this Server, all clients (HBB-NEXT STB, 2nd screen devices etc.) will connect in order to subscribe to events, and to receive publications of events. In later stages of the project, deployment of a RabbitMQ Server on the TARA STB will be investigated.

2.2.2. Protocol

The protocol used by RabbitMQ is AMQP, an open and flexible implementation. RabbitMQ extends the AQMP specification, providing several protocol extensions. A detailed description of the RabbitMQ protocol can be found on the official website [17].

2.3. Multi Modal Interface

This section describes the four modularities that are currently part of the multi-modal interface:

- Multi voice identification
- Speech & voice command recognition
- Gesture Navigation
- Eye Navigation

Each subsection explains the algorithms used and the approach taken.

2.3.1. Multi voice identification

Multi-speaker identification aims to identify possibly more speakers based on a recorded signal that may contain utterances of more individuals. This general task can be separated into several categories based on additional refinements. If the speakers who may appear in given conversation are known in prior, i.e. they were present in some sort of training phase, the task resembles a single speaker identification problem, even though additional algorithms must be applied, tuned and enhanced. However when the set of possible speakers is unknown then the techniques of speaker segmentation and clustering (diarization) must be used. In doing so it is expected that there is no serious overlap of utterances of different speakers. In this scenario and setting the detection of changing points for different speakers is the main focus instead of knowing their identity (which can't be done as no prior information exist).

The aim of the designed application is to continuously run and "listen" to an incoming stream of PCM samples (sound waves), detect voice activity (VAD), silence and background noises, and if substantially long voice period is caught then identify the speaker with certain confidence measure. Optionally, system detecting overlapped speech can be employed as well. As it can be seen the main focus here is not in speaker clustering and segmentation thus in following additional and extended algorithms for speaker identification (multi-user) will be outlined.

The two main parts of speaker identification system are speech features methods and classification algorithms. As they are in this scenario similar to the single speaker identification, please see the basic principle of MFCC construction and KNN classification in single user system description. Here the modification to improve the speed and accuracy in the case of less training samples are given.

2.3.1.1. K-D trees and K Nearest Neighbour

As it was stated KNN is a simple but in certain conditions very accurate method, however, its main drawback is in the recognition phase, which takes huge computational load. To alleviate this data is stored in k dimensional binary trees. Each node has a key (dimension upon which it performs a space division) and a dividing value, that should be for a given level and dimension the most separating. Even thought to find a proper element takes log2 searches the situation is more complicated as more individuals must be found (k neighbours). Thus also different leaves are usually needed to be searched as well. To do so two functions must be implemented: within_bonds and overlaped_region. The first one check if the worst neighbour can be outperformed by another vector stored in remaining leafs. The second one determines whether it is necessary to searches also "other" leaf, not the original which the tested vector belonged to. Having these functions the tree is recursively searched and returns required neighbours.

2.3.1.2. GMM

GMM is parameter based classification which presents the feature space by a number of Gaussian mixtures. It is beneficial if there is less data and is computationally faster than KNN in the recognition phase; however it requires more advanced training phase.

It is assumed that a linear combination of Gaussian densities can approximate with arbitrary small error any continuous statistical distribution. A multidimensional GMM is thus given as:

$$pdf(\mathbf{x}) = \sum_{i=1}^{N} \pi_i \frac{1}{\sqrt{(2\pi)^d \det(\mathbf{\Sigma}_i)}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)\mathbf{\Sigma}_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)^T},$$

where x is the observation vector, μi is the mean of i-th mixture, Σi is the covariance matrix of i-th mixture, πi is the weight of i-th mixture, and N is the number of mixtures. Thus the main problem is to estimate the mean vectors, covariance matrices and weights of all Gaussian distribution. It is done by the well know EM algorithm that is a maximum likelihood (ML) estimator. Again only a local maximum of ML is guaranteed. Then the recognition may follow the optimal Bayes classification rule where each speaker (its distribution) is represented by GMM and via its output required likelihoods P(x/classi) can be obtained.

2.3.1.3. VAD

Voice activity detection is usually done based on simple features like energy, intensity, log energy, zero-crossings, periodicity, etc. Its main aim is to separate voices and background noises. In the presence of high SNR the task is rather simple and purely energy-threshold based. However in most situations it is not the case and various combinations of different types of features must be used and adapted in real time. In addition to energy pitch period or voicing measure can be used in some time interval and evaluated. Pitch can be estimated via xcorr or amdf functions as follows:

$$AMDF(k) = \sum_{n=0}^{n<N} \left| x(n) - x(n+k) \right| \quad k \in < T_{0\min}, T_{0\max} >$$

In case of a period and multiplications of its length it exhibits low values (in ideal case zeros). Finding potential pitch a voicing feature can be simply estimated as the measure of resemblance between the original and shifted signal introducing some kind of normalization.

2.3.1.4. Overlapped speech (two or more speaker speak at the same time)

Overlapped speech can be detected directly using acoustical features like pitch period which in case of two competing speeches is not unique and usually the confidence of its detection is rather low (voicing feature). It can be detected indirectly using classification algorithms where in the intervals of competed speeches rather unstable identifications with low confidences are observed. Even in such situation usable speech for classification can be derived from voiced/unvoiced intervals where speeches have different energy levels.

2.3.1.5. Conclusions

Faster classification methods will be implemented and tested so the recognition process would be able to run for an infinite time interval (KD_KNN and GMM). Both acoustical and prosodic features (pitch period, voicing, etc.) will be used to improve speaker recognition and VAD accuracy. A VAD algorithm will be designed and implemented to classify speech and non-speech event with a high accuracy and low computational load. New decision taking algorithms will be suggested to meet multi speaker particularities with a possibility of detecting unknown speaker (not being recorded in the database) by employing some confidence criteria or by creating general speaker model.

2.3.2. Speech & voice command recognition

Speech signal is produced by human speech organs and is originally represented by air waves. It contains among other information lexical part that is crucial for speech recognition (speaker specific information, additional and convolutional noises as well). Every language contains a huge vocabulary, usually of several hundreds of thousands of words. To meet different settings many systems have evolved and thus some basic classifications are used: small, medium or large vocabulary systems, speaker dependent or speaker independent system, phoneme or word based system (sub-phoneme or phrases are also possible), continuous or isolated word (dictation) systems, etc.

In the case of command recognition the systems belonging to the group of isolated word recognition would be an option. The most successful and used ones are those based on HMM statistical speech modelling, especially those using tight context dependent phonemes as a basic modelling unit. In case of a fixed set of commands and abundance of test samples whole word models can be used achieving potentially higher accuracies (better capturing co- articulation effects).

2.3.2.1. Hidden Markov Model

The Hidden Markov Model is a statistical modelling method for speech and more precisely for its parts (words, syllables, phonemes, sub phonemes, etc.). It is based on concept of Markov chain that makes it computationally very effective even though not reflecting time evolution of genuine speech. Thus each model must be estimated using usually very large set of training examples that contain multiple recordings of the same word (its different realizations). HMM is defined by a priory probabilities (π) of being in particular states at the begging, transition matrix (transition probabilities between states, aij) and probability distributions (likelihoods) of generating observation vectors in a given state, P(x/si). These distributions are not known in prior but the most widely used ones are mixtures of multidimensional normal distributions (GMM).

Then the probability of observing string of feature vectors at the model λ is given by:

$$P(\mathbf{x}_1 \ldots \mathbf{x}_T \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i) \text{ where } \alpha_t(j) = [\sum_{i=1}^{N} \alpha_{t-1}(i) * a_{ij}] * P(\mathbf{x}_t / S_j) \quad j = 1, \ldots N$$
$$\text{and } \alpha_1(j) = \pi_j P(\mathbf{x}_1 / S_j)$$

Then the recognition is done be choosing the HMM model (λ) with the highest probability. Great advantage of HMM models is the possibility of concatenating several models into a string thus utterance of any length can be built up using set of basic models. In practical situation only the path with maximal probability is calculated e.g. P(x1,St1,x2,St2,…xTStT), and via backtracking the sequence of hidden states is determined (states can be related to sequence of words) using Viterbi algorithm.

The main problem with HMM is the estimation of their parameters. Usually it is done by maximizing the maximum likelihood P(x1,x2,…xT/ λ) which leads to well-known method called Baum-Welche algorithm. However newer strategies like maximal mutual information (MMI), large margin HMM, and corrective training may be beneficial in some cases, especially when the HMM model constrains do not fit the underlying physical model. The MMI model estimation criterion is defined as follows:

$$\log(P(\mathbf{x}_1,...\mathbf{x}_T / \lambda_i)) - \sum_{j=1}^{P} \log(P(\mathbf{x}_1,...\mathbf{x}_T / \lambda_j)),$$

where λi is the correct HMM model according to the known transcription of the observation (x1,x2,…xT) and λj are all available models. This criterion aims to increase the overall separation gap among models.

Except training strategies the structure of HMM models is also important. Usually for speech units left - right models are used and the most common is probably the Bakis structure. It is commonly accepted that each acoustical state is modelled by 3 HMM states in order to capture beginning, middle and ending part of it. A basic HMM model can model phonemes, syllables, words or even whole phrases. It is a trade-off between accuracy of modelling and computational feasibility and availability of the training data. Some frequent usually short words can be modelled by a single HMM model. A good balance is achieved by so called tied triphones models which are used by most of the employed systems. Whatever training criterion is used there are no close formulas for optimal parameters thus several iteration cycles must be applied in a controlled way. Unfortunately only local maxima are guaranteed to be found. Following the classification theory HMM models are also prone to be overfitted, thus a validation set must be used to detect and prevent this phenomenon. The model complexity, modelled speech units and training strategies depend on the particular application, and are mutually adjusted in order to achieve best trade-off between accuracy and robustness.

Except speech modelling there is still the task of speech extraction. The extracted features should capture the lexical information while suppressing the other ones (noises and supports inter and intra speaker variability). The most successful techniques are MFCC and PLP and their modifications, but still some experimental ones are emerging. As MFCC was briefly described in single speaker identification module please refer to it. Additional features may include signal dynamic observed via delta and acceleration coefficients constructed over acoustic features showing their time evolution as well. They are defined as:

$$\Delta(n) = m\sum_{k=-L}^{L} kc(n+k) \quad and \quad \Delta\Delta(n) = m\sum_{k=-L}^{L} k\Delta(n+k)$$

Where m is a normalizing constant and L is the time span. More sophisticated systems use the notion of super vectors that are reduced by a linear transform usually via dimension reducing methods while preserving essential information, most common are: PCA, LDA and HLDA.

2.3.2.2. Conclusions

For a given voice command based application HMM technology with context dependent triphones will be used with an option for modelling very frequent and short words as a whole. The task will be to design proper model structure (number of gauss mixtures, states and transitions, structures for background models), to use proper training strategy (ML, MMI, MCE, etc.) and to optimise the training process itself (iterations, stages of the training etc.). Another course of research would be represented by selecting and modifying extracted features that would fit the environment as well as the HMM models and training strategies.

2.3.3. Gesture Navigation

Gesture recognition can be conducted in two manners. Either a data glove is used which transforms the body flexions into movement information, or vision-based approach is applied where a camera serves as a human eye to record body positions which are then extracted using image processing. It is clear that while the first method might bring precise results, it is rather uncomfortable in terms of user convenience.

Also, equipment needed to employ the method would be unacceptably costly for most of standard customers making it only suitable for special use. The latter approach, on the other hand, has no other equipment requirements for the end user (except for the camera), making it suitable for general applications. The drawback, however, lays in algorithmic complexity where considerable amount of time and computing power is required to extract body movements.

There are various algorithms available which focus on different aspects of the gesturing person (and take different assumptions). Generally, they can be divided into two categories, appearance- and 3D model-based approach. The 3D model-based approach compares the input parameters of a limb with 2D projection of a 3D limb model. The appearance-based approach uses image features to model the visual appearance of a limb and compare it with extracted image features from the video input.

When focusing on the latter approach, the results depend on the capabilities of the capturing device. If an RGB camera is used the methods focus on tracking the skin colour or shape of the gesturing body part. The approach, however, depends highly on the lighting conditions, as well as stability of foreground and background of the tracked subject. Also, no other skin-coloured or limb-shaped objects can appear in the examined area as they would trick the algorithm. An infra-red light depth camera uses its own IR light emitter and is thus much more resilient to lighting conditions of the scene. Moreover, the camera is capable of providing a depth map, a pseudo 3D image of the scene which can be very useful when tracking gesturing body parts, i.e. hands.

2.3.3.1. Algorithm

The Kinect device was introduced to the market in November 2010 by Microsoft. Primarily, it was designed for the XBOX360 gaming console where it serves as a contactless controller. Later Microsoft responded to the needs of the market and launched a second version of the device, the Kinect for Windows, in February 2012, which is primarily designed for commercial purposes.

Kinect sensor returns the distance as depth data for each captured pixel with the depth camera resolution of 640x480 pixels. Figure 2 shows the depth data rendered as an RGB image after normalization and conversion of distances to RGB elements of the moving scene. Character shooting with depth camera has considerable advantages over using RGB camera, because the depth camera does not depend on light conditions. The only limiting factor for the shooting by depth camera is direct sunlight because it affects the IR sensor by interference with IR light from the sensor's emitter. When using an RGB camera it is necessary to modify the brightness, contrast and other parameters that affect image quality. Using the depth camera these corrections are not needed.

Figure 2: Depth camera image

Finding the contours of the hand

For faster processing of data, we cannot process the whole image area. After we automatically detect the hand we allocate the area around the hand which is then only processed. When we convert obtained distances into the RGB image, we get the contour of our hand on a black background (Figure 3). This aspect also should be taken into consideration for the best calculating. It is mainly due to the fact that only the used area was processed and not the whole picture.

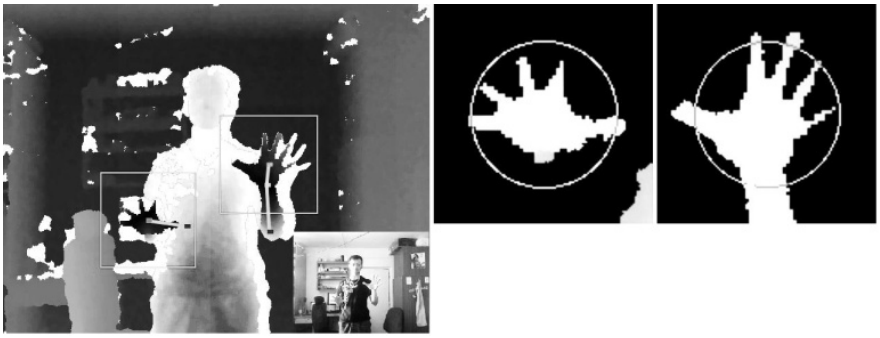


Figure 3: Detected hands are segmented from the main picture in order to minimize image processing complexity

Finding the middle point of the hand

Finding the coordinates of the hand using implemented algorithm in Kinect is relatively inaccurate because large differences occur between the opening and closing palm. Therefore, another method was used for finding the middle point of the palm. We found the hand contour in the previous step. Now, this knowledge will help us find the centre contours - which are ultimately the centre of the palm, no matter whether it is an open or closed hand.

Calculating convexity defects

To find and calculate the convexity defects we used the GetConvexityDefacts method of emguCV. Figure 4 shows convexity defects (white colour). There are two methods to find convexity defects: clockwise method and counter clockwise method. They return the coordinates of three points (START_POINT, DEPTH_POINT and END_POINT, representing the starting point, the deepest point and end point) where the deepest point is understood as maximum distance between the hull and hand contour. Figure 4 shows one of the convexity defects, however, the method computes all defects.

Figure 4: START_POINT (left), DEPTH_POINT (bottom) and END_POINT (top)

Convexity defects method is not primarily designed for recognition of the number of fingers on hand but it can be applied for any object, if the method is applied properly. Therefore it is necessary to adapt its functionality so that it recognizes only the fingers and it does not consider the whole area of hand. When we use the method without modifications, the algorithm finds many defects and cannot recognize if the defect represents a finger or not. Additionally, this raw method finds more defects on the hand with closed palm than when the palm is opened.

Finding the point on the finger that is the farthest from the centre of hand

For finding the point on the finger that is the farthest from the centre of hand we reverse the order of the whole contour points since the contour points are placed in a list. The algorithm starts in point START_POINT and the MAX_POINT has the same value as START_POINT at the beginning. Then we sequentially move to the next coordinate until it finds a coordinate that is the farthest from the centre of hand. The distance from middle point of hand to MAX_POINT is compared at each step, while there is farther point from the centre point than in previous point.

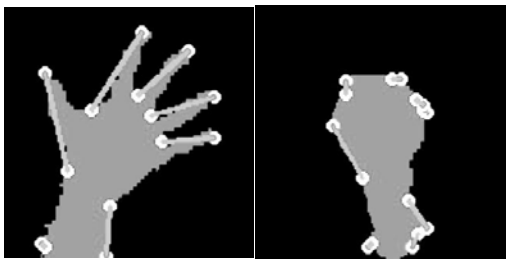


Figure 5: Finding the maximum distances between contour and depth points

Removal of defects whose triangle height is less than the specified

Since we found many defects on the hands, including those which do not represent fingers and thus are not necessary, we must remove all these defects. The first step is to remove all defects whose height is less than a specified value. This value is dynamically changing according to the size of area where the hand is detected.

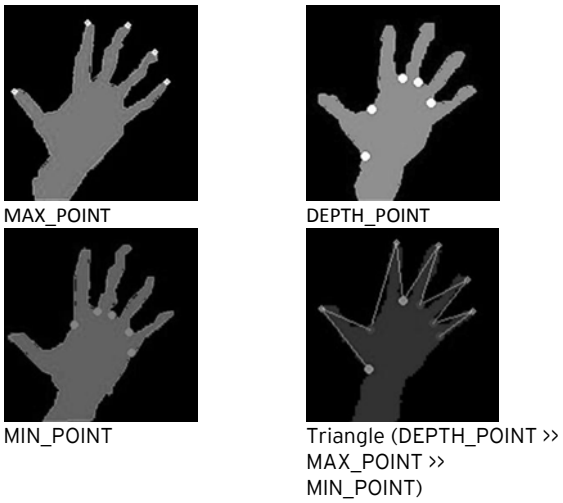


Figure 6: To illustrate the logic behind using triangles to eliminate unnecessary defects

Removing of defects that have the distance MAX_POINT – START_POINT longer than the specified

The second step of filtering is to remove the defects that have a distance between START_POINT and MAX_POINT more than a specified value, which is also dynamically changing (if two points are too far, it means that they cannot represent a finger).

Removing of defects that have the distance START_POINT – DEPTH_POINT less than the specified

The third step of filtering is to remove also the defects that have a distance between START_POINT and DEPTH_POINT less than a specified value. This value changes dynamically according to the size of the region where the hand is detected.

Remove points that are below the wrist

The last step is to remove all the defects that occur below the wrist. We will find a way to join together two points, HAND and WRIST (provided by the Kinect API). This straight line is rotated 90 degrees left or right, eventually. All defects that arise on the side which does not contain coordinate of HAND will be removed.

2.3.3.2. Conclusion

We have developed an application where a user is able to change the drawing tool in real time. A method was added into the API which returns information whether the hand performed a “click” gesture, and the number of fingers. The user is able to draw pictures of various sizes directly on the screen.

The number of fingers on the left hand determines which picture will be displayed. The size of the picture is determined by the "click" gesture of the right hand. If a hand is marked with a red circle, its number of fingers will not be recognized. Whole image can be erased when both hands are positioned in front of the body in non-recognizable position. Figure 7 depicts the simple interface of the created application in practice.

Figure 7: User interface of the test application

The recognition works well up to the distance of 2 metres from sensor. Greater distances provide hand image with limited pixel count due to native (VGA) resolution. The application is currently able to recognize the number of shown fingers in real time, 30 frames per second, while processing both hands.

Methods based on RGB camera image processing depend highly on lighting conditions and background colour and dynamics. Thus, most of the solutions only work as long as the specified conditions are met almost exactly. This is, however, impossible to achieve in a normal user environment. Our method has proved its independence from background and illumination when only strong direct sunlight has forced the sensor not to respond. Moreover, gaining from the Kinect capabilities, the hand can be rotated in any angle as long as it faces towards the Kinect sensor.

There is an on-going research yielding to implementation of a gesture recognition system based on a pre-defined database. We plan to build a more sophisticated system which will recognize not only static gestures but also dynamic gestures and their combination.

Eventually, the system will decide the meaning of a gesture also from the movement characteristics and starting and ending position of the hand.

2.3.4. Eye Navigation

Currently, there are few methods how eye controlling can be implemented. We choose the simplest and the most natural way. This system has only one part. It is the static RGB camera, which is available commonly in every laptop. The cameras in laptops usually have different image resolutions. It cannot be said, that higher resolution means better camera. Picture quality depends on the size of camera lens and sensor. From our experience, laptop web cameras don't have very good picture quality, but with the help of special image filters it can be useful for our application. In our case, better camera definitely means better and more accurate results.

2.3.4.1. Algorithm

As a middleware for kinect device we have available both Open NI and the official MS Kinect SDK. For image processing we use Egmu CV library. This library is in basic wrapper for popular Open CV library which provides compatibility with .NET framework.

Pupil detection with CDF analysis

The region of interest (ROI) consists of those areas that contain the eyes. If we find this region, we can reduce the computational cost by reducing the search space. We can find the region of interest by dividing the face region vertically into top and bottom parts. Then the top part is partitioned horizontally into left and right segments. At this stage we have two regions where each one has an eye.

The center of pupil is detected by an adaptive approach. After finding the ROIs, that each one contains an eye, the following approach is performed on both regions. At the first step we filter each ROI by the cumulative distributed function (CDF) of that region.

We find the CDF by integrating the histogram of each of the ROI by using:

$$CDF(r) = \sum_{w=0}^{r} P(w) \quad (1)$$

where P(w) is the histogram representing the probability of occurrence of gray level w and 0 ≤ r ≤ 255. It was found that the pixels of eyelid and pupil have CDF ≤ 0.05, so ROI is filtered by:

$$I'(x,y) = \begin{cases} 255; & CDF(I(x,y)) \le 0.05 \\ 0; & \text{otherwise} \end{cases} \quad (2)$$

where I(x,y), is the original eye region and, I'(x,y) is the image that only contains the pixels of pupil and upper eyelid and, those parts that their CDF is less than 0.05. We can see the result of this section in Figure 8a).

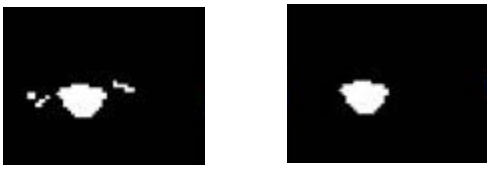


Figure 8: ***a)*** *Image after CDF* ***b)*** *Image a) after erosion morf. oper.*

The result of the first part contains regions rather than pupil. We perform morphology to remove these parts. The erosion morphological operation by 2×2 structure element is a suitable candidate for this purpose. Figure 8b) illustrates the outcome of this step which is approximately the pupil.

In the other approach we use a kinect RGB camera. The person sitting in the front of the kinect (monitor) will be asked to look with its eyes to the highlighted points on the screen with the still head. Simultaneously the application by kinect depth camera measure the head distance from the kinect (monitor). For the purpose of the triangle calculation we will determine the dimensions of the screen. Application will calculate the variance of the pupil movement exactly the same like in section calibration and also it will calculate the angles by which the pupil must be diverted from straight position to see the edge of the monitor. With knowing the head distance from monitor we can recalculate the variance and angles when the distance is changed to ensure the accuracy of the controlling.

2.3.4.2. Conclusion

Our experiments have shown that the methods are highly dependent on the quality of illumination of the environment. The algorithm works properly only under certain lighting conditions which are difficult to maintain for longer periods of time. Additionally, sensors which are used are primarily constructed with insufficient image resolution in comparison to size and distance of human eyes from the sensor.

2.3.5. Speech Synthesis

Speech synthesis is an important area of multimodal interface development. Duplex speech communication creates a user friendly environment. Speech commands are usually used by users to control features of their television. Display opportunities should be extended with speech reaction. Speech is required mainly in case when the user is not in eye contact with display. Various actions can cause this situation. Other scenarios are those, where video stream cannot be interrupted but engine needs to bring some message to the user, e. g.: "You have one new mail" or "Go out for a walk, you need fresh air and movement".

Speech synthesizer is a device that provides voice communication. From wide range of speech synthesis kinds e.g.: formant, sinusoidal, HMM, etc. we use concatenative speech synthesis. We use database with units cut from recorded human speech. Speech units differ from each other in length e. g.: phonemes, diphones and triphones. We use diphones. The larger units provide better quality but the disadvantage is that the database is larger.

2.3.5.1. Algorithm

Speech synthesis for Slovak language is built on a modular architecture. This solution has many advantages, e. g.: depending on the demand and, of course also, availability of resources it can be chosen which module and how many times should contribute to the process of speech synthesis. Intermodular data exchange of partial results is based on XML RPC protocol. Data are transmitted encapsulated in packets mainly in our internal XML standard. The whole process is controlled by a supervisioning node in a hub topology.

This solution is currently limited to the Slovak language. However, the solution can be applied to other language speech synthesis. Modular speech synthesizer is developed as the stand alone demo. This application can be integrated to common system. Unique interface is required for this purpose.

2.3.5.2. Conclusion

Basic functionality of speech synthesis can be implemented as an entity of words and phrases or voice commands. In this way is possible to ensure support for foreign languages in relatively short time. The main advantage of this solution is naturalness of speech. One disadvantage is, that system is very limited.

In general, the question is, in which scenarios artificial speech is needed. As it was afore mentioned, it is in time, when user should be noticed. User should be noticed either according own personalized settings e. g.: five minutes before news, or after too long period watching TV, or before some special scenes that are very funny or scary, violent, obscene or in appropriate for children. TV can attract attention to prevent missing of seeing the chance of scoring in soccer. In the case, when children watch TV immediately after they come home from school, TV can ask them, if their homework is finished.

3. Context-Aware and Multi-User Recommendation Engine

The HBB Next recommendation engine provides functionality to Applications and Services to receive context-aware personalized multi-user & multi-device-based content recommendations. To optimize recommendations, context information from users in a group and the collection of user profile information will be used for content filtering, as well as collaborative content filtering algorithms, all of them serving as input for the service personalization processes. The recommendation engine will enable content recommendations coming from both the broadcast and broadband domain, by building on algorithms for merging the HBB metadata. To provide this functionality, a metadata merging function will be implemented. This chapter describes the current status of the context-aware multi-user recommendation engine.

3.1. Overview

The current demo consists of a number of modules that work together in order to identify the people in the group and recommend them a set of movies. The modules that are implemented are listed in the UML sequence diagram in Figure 9.

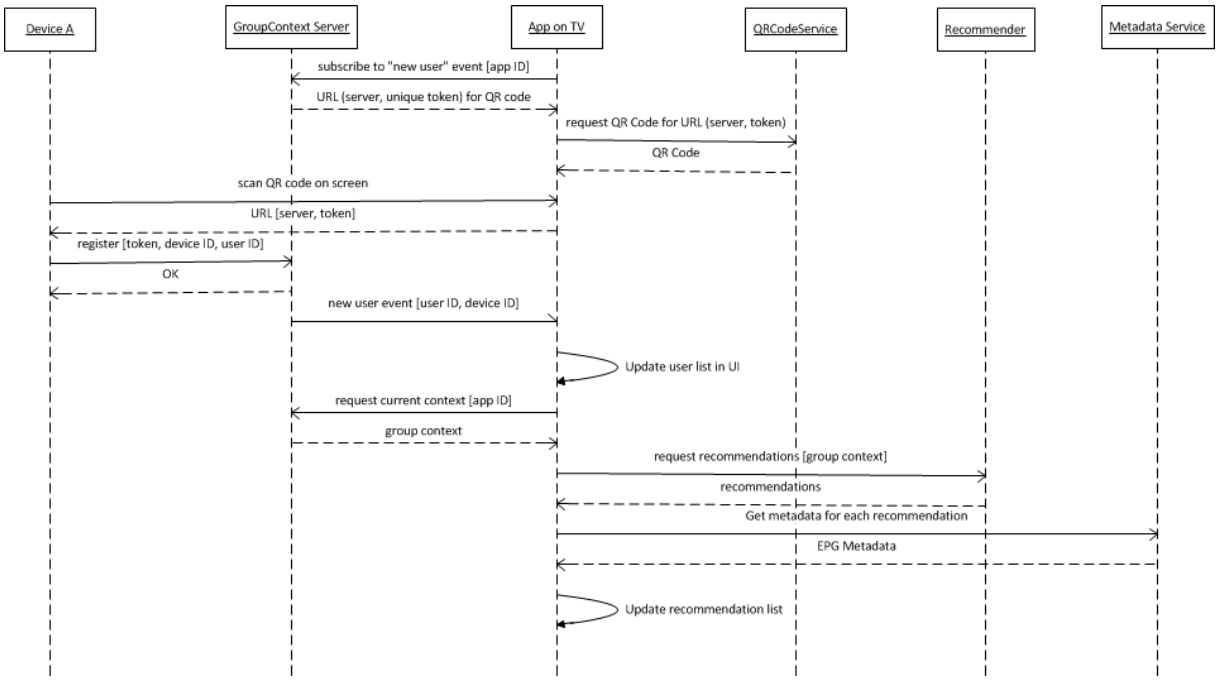


Figure 9: Sequence diagram

Users identify themselves by scanning a QR code on the screen of the TV / computer, using an Android app. The App lets the Group Context Server know that a user has joint. The Group Context Server in turn notifies the HBB Next application on the TV. This application requests a new list of recommendations that is tailored to the new group context.

All modules that are listed in Figure 9 are described in detail in the following sections.

3.2. Group Context Server

The Group Context Server keeps track of the users that joined a certain application. It notifies the applications that registered themselves at the server when a user joins or leaves. This notification is done in an asynchronous way using a technique called “long polling”. Long polling is an efficient variation on traditional polling and emulates an information push from the server to the terminal (i.e. client). With long polling, the client requests information from the server in a similar way to a normal poll. However, if the server does not have any information available for the client, instead of sending an empty response, the server holds the request and waits for some information to be available. Once the information becomes available (or after the connection timed out), a complete response is sent to the client. The client will normally issue another long polling request immediately, so that the server will almost always have an available waiting request that it can use to deliver data in response to an event. Figure 10 illustrates the message sequence of HTTP long polling.

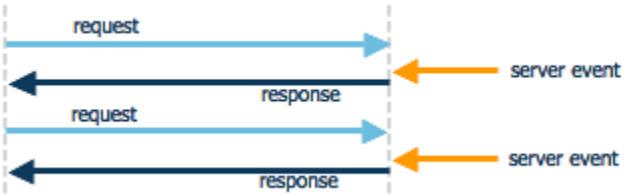


Figure 10: HTTP Long Polling

The Group Context Server is a temporary component that will eventually be replaced by the Notification Service. Up until then it is in place to be able to simulate and demonstrate group context changes.

The following two subsections describe the design and protocol of the Group Context Server respectively.

3.2.1. Design

The design of the Group Context Server is fairly rudimental. It consists of a data structure that stores the active applications and their active users. Since the server should be able to handle many concurrent sessions, Java's concurrency library is used to make the data structures that contain the information thread safe. The REST API of the Group Context Server is implemented using the Restlet framework [22]. This framework enables developers to easily create REST services by providing them with an API through which REST resources can be implemented and wired to the service.

Figure 11 shows a UML diagram of the Group Context Server's design. Requests to the service are received by an instance of the GroupContextService class. This instance routes the request to one of the four resources that perform the requested action. The AppsResource, AppResource and UserResource update the group context that is kept in the GroupContext singleton. The GroupContextResource notifies the clients when the group context changes and therefor it attached itself as a GroupContextListener to the GroupContext. After each update of the context the all GroupContextListeners are notified of the change. When this event is received by the GroupContextResource it releases the pending request and returns information about the event that just happened.

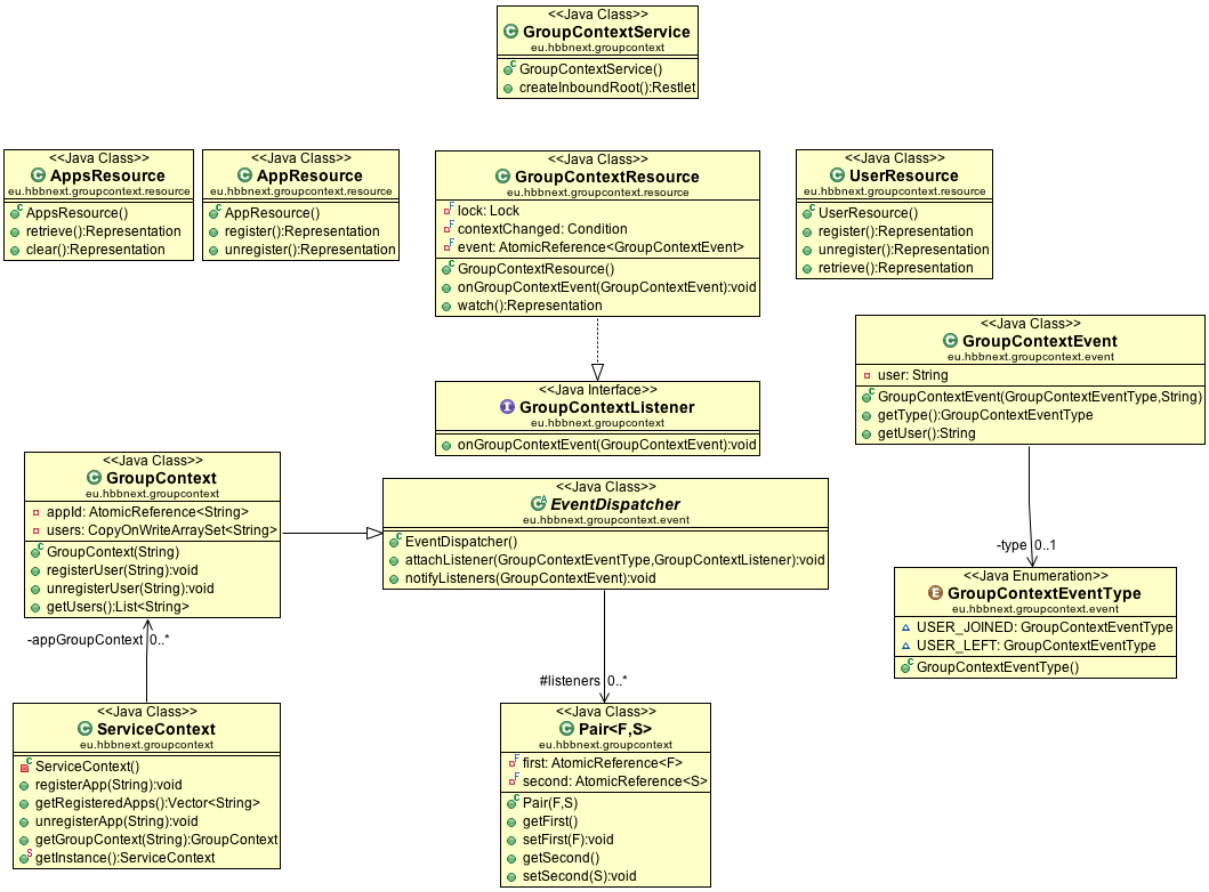


Figure 11: UML diagram of the Group Context Server

3.2.2. Protocol

The API of the Group Context Server is simple. It contains eight methods that can be called using REST. The methods allow the application developer to (un)register the app and to get a notification when the group context related to the app changes. Other methods allow users to (un)register themselves with respect to a certain app. The methods provided by the Group Context Server API are:

- **Get registered applications**. Get all applications that registered themselves at the Group Context Server.
- **Unregister all applications**. Unregister all applications that are registered at the Group Context Server.
- **Register application**. Register an application with a specific application ID.
- **Unregister a specific application**. Unregister an application with a specific application ID.

- **Register user**. Register a user with a specific user ID at the specified application.
- **Unregister user**. Unregister a user with a specific user ID from the specified application.
- **Get registered users**. Get a list of all users that are currently registered at the specified application.
- **Wait for group context change**. Issue a long polling request, waiting for a group context change event.

The Group Context Server API is described in detail in annex A.

3.3. QRCode Service

The QR Code Service is a small service that can create QR Codes that encode a specified URL. The service is used in the demo application and allows users to identify themselves at the Group Context Server through the Android Demo App.

3.3.1. Design

The implementation of the service utilizes two external Java libraries: ZXing and Restlet. The first is used to create the QR code while the latter makes it available as a REST service.

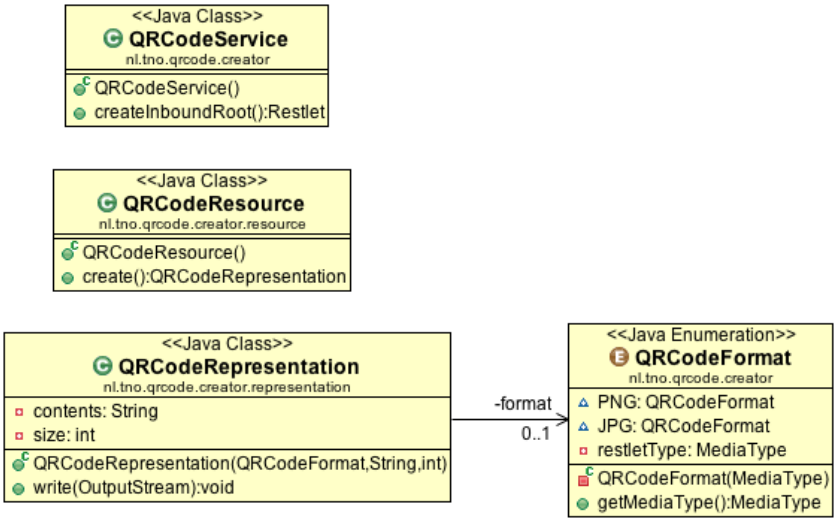


Figure 12: UML class diagram of the QRCode Service

The UML class diagram in Figure 12 shows that the QRCode Service required only four classes to be implemented.

3.3.2. Protocol

The protocol of the QR Code Service is very straightforward and consists of only one method that can be called using REST.

Get QR Code

URL:	/QRCodeService/<version>/qrcode
Method:	GET

Parameters

url	The URL to encode.
size	The height and width of the QR code in pixels. Default is 300px.
format	The format of the resulting QR code. Default is PNG another option is JPG.

3.4. Metadata Integration Service

The Metadata Integration Service (MIS) is a modular Multi-Metadata-Format translation tool. It consists of several input modules converting diverse input formats such as DVB-IP, YouTube, UPnP, Podcasts, and TV-Anytime XML (local / remote). Supported output formats are limited to TV-Anytime and DublinCore [1] formats.

The MIS allows users to search and find multimedia content across different domains, and supports automated and user generated metadata (indexing). Technically, this is implemented as depicted in Figure 13 – various software modules, the so-called “domain-specific metadata handlers” are plugged into the metadata integration system, managing conversion of domain-specific mechanisms, formats and classifications (e.g. genres) into and out of the API. For example, the DVB-IP module maps queries coming from an EPG application through the metadata API onto DVB-IP specific methods, and converts query results, as well as formats (TV-Anytime Phase II), back to the API, and therefore towards the EPG. The complete framework is implemented in Java, in order to remain platform-independent.

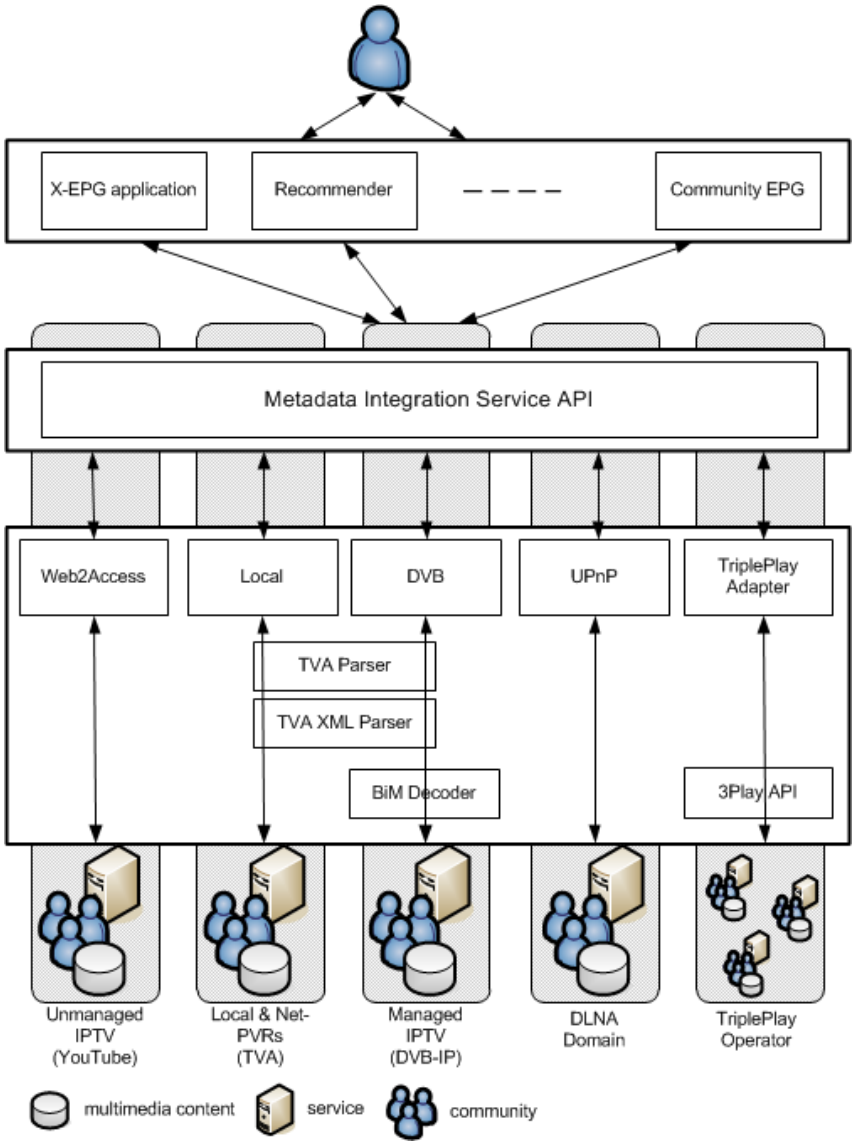


Figure 13: Hybrid Metadata Scenario - technical approach

Service Discovery and Metadata retrieval are broad areas and span several domains. In HBB-NEXT, Service Discovery remains semi-static – depending on the dynamic solutions within the different underlying domains (broadcast / broadband). For example, in the DVB-IP domain ([2],[3],[4],[5]), the client automatically gets informed about new services and content being available within the current managed IPTV network, via the SDnS and BCG mechanisms respectively. Therefore, the Service Discovery is fully dynamic in this area.

Within the Web2.0 world, things are different. New services offering multimedia content, which usually are presented as web sites, have to be discovered by the user himself – either by explicitly searching for them, or by receiving recommendations via emails or instant messaging.

However, there are almost no mechanisms available that allow a fully dynamic service discovery within the web – and if they are, they are either proprietary (and in most cases closed), or still in an early stage of development. But still, at least a few web service providers offer APIs that can be used to access their content, without the need of using the web browser, such as YouTube [6] with its GDATA API [7].

Within HBB-NEXT, no new metadata format as such will be designed, as in our opinion, enough of them already exist. Therefore, we define an API / interface that describes parameters of multimedia items, such as a title or a set of content genres, at runtime - by means of methods in order to access these parameters. Although heavily inspired by TV-Anytime [8], we do not specify how instances of metadata representations shall be made persistent, i.e. in XML. We leave that open, in order to allow compatibility to existing metadata formats (i.e. TV-Anytime, MPEG7 [9], DIDL [10] etc.), as well as to future ones. To allow applications to access persistent representations of content metadata, we define an interface that offers access to format-specific implementations which provide XML based format exports (the “ContentInformationExporter”).

3.4.1. Design

This section describes the details of the domain-specific implementations of the cross domain metadata API, as depicted in Figure 13. They provide detailed information about implementations within the DVB-IP and the Web2.0 domain.

Managed IPTV / DVB-IP

The DVB-IP module implements the DVB-IP specification, and therefore acts as a standards-compliant DVB-IP client. It retrieves and analyses Service Discovery and Selection (SD&S) Records, as well as EPG-Data for Broadband Content Guides, via http as well as via Multicast.

Content Classifications (content genres) are, for the case they are encoded according to the DVB SI Content Descriptor, being mapped towards the classifications of the Metadata API – which are the ones defined by the TV-Anytime Forum [8]. In case the genres are already encoded according to the classification scheme defined by the TV-Anytime forum, there is no need to translate them, so they will just be kept as provided.

The same mappings will also take place for classifications of audio- and video codes, Audio/video attributes such as resolution, aspect ratio, number of audio channels, and subtitle encodings.

The implementation supports dynamic updates of service - and content – metadata, as defined by DVB. Once services have changes, the client gets informed via the SD&S update mechanisms. The same applies to changes within the content metadata. Specifically, the client will be able to detect changes of certain metadata descriptions by comparing existing fragmentID and fragmentVersion pairs, with newly announced ones. Once this happens, the DVB-IP module will automatically inform registered ContentChangeListeners via the HBB-NEXT specific counterparts of the API – i.e. the “notifyChanged” method.

Unmanaged IPTV / Web2.0

YouTube is a social media web site, enabling users to share videos, comment on them and rate them. Videos can also be tagged, and scene-specific annotations can be created. Except for annotations, all information is available in machine-readable form as Atom feeds, an IETF standard for feed syndication which originated in weblogging and podcasting. This is part of the Google Data API, which in its most recent version also implements publishing based on AtomPub. Google’s Java implementation of this API was used as a basis of the YouTube module.

YouTube Google Data API

The YouTube specific implementation API uses the GData Java API to make search requests and resolve metadata for specific videos and to publish, comment and rate videos. After the Atom representation of the metadata is retrieved and de-serialized into Java objects, the implementation class translates it onto the MIS - API. A lot of it is quite easy to map, such as the title, synopsis etc. Some other pieces of information require a bit more effort to map which will be shown in the following sub-sections.

Metadata Mapping

Because YouTube only allows a very limited set of categorizations or genres and only one per video, it is not possible to create a 1:1 mapping with any given TV-Anytime input when publishing. A mapping in each direction has been created in the form of TV-Anytime classification scheme mappings. If a video is published on YouTube and has more than one content classification, the first one is mapped to a "YouTube category" and additionally, to preserve as much information as possible, all content classifications' plain text names are used as keywords.

When metadata about a video has been retrieved as the result of a query or CRID resolution, it is put into a list of known items together with the listener and the current time. The YouTube service then checks at regular intervals whether the item changed.

In order to do this in a resource efficient way, YouTube's GData API provides a query method parameter "last update" which only retrieves the item if it has changed since the given time. The update interval can be configured using the properties file.

3.4.2. Protocol

The design of the RESTful API has not been realised yet, and is planned for year 2. Therefore, currently only the "real Java API" is being used in the implementation of the prototype.

3.5. Metadata Enrichment Service

The Metadata Enrichment Service (MES) is a modular & multi-source implementation that allows enrichment of a given set of metadata (describing i.e. one programme event) by metadata retrieved from linked open data clouds (LODs), such as DBPedia [11] or IMDB [12].

The MES supports the formats and classifications designed within the MIS, thereby making it easy for an application to use – the data types remain the same, for implementers it's just 2 new methods that can be used.

Figure 14 depicts the overall process of the HBB-NEXT metadata aggregation & content recommendation:

- metadata aggregation (grey area / MIS)
- metadata resolvement (blue are / the process of finding LODs holding matching metadata for a programme event)
- metadata enrichment (red area / receiving rich metadata from LODs)
- metadata filtering & recommendation calculation (green area)

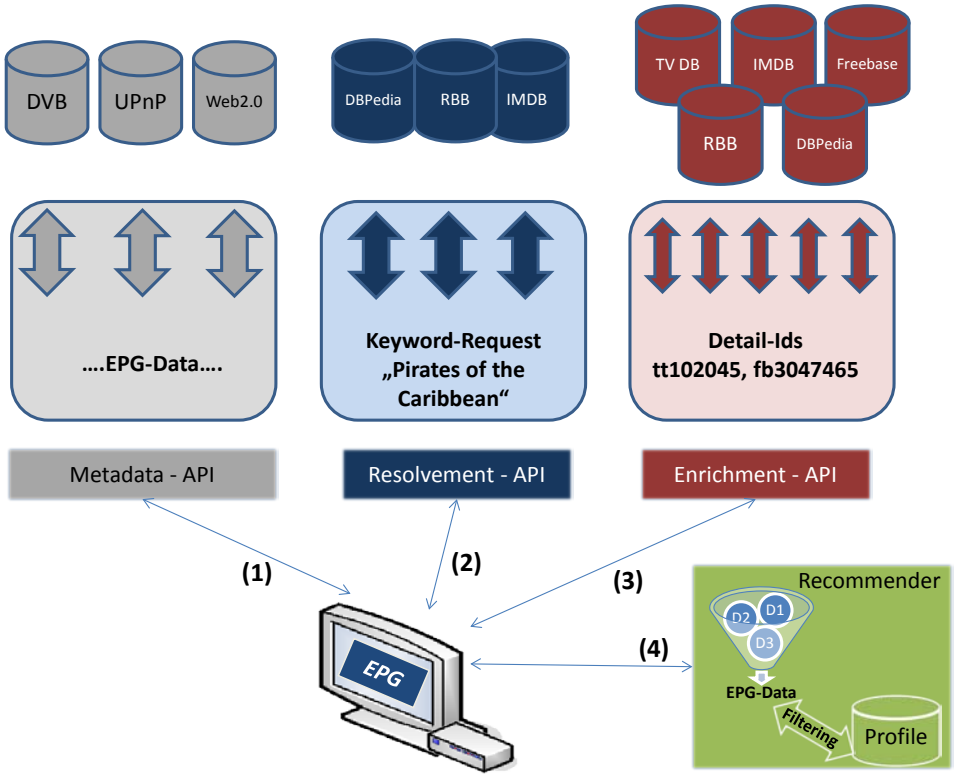


Figure 14: Overall process of creating a "personalized rich EPG"

Currently connected LOD sources are DBPedia, IMDB, and Freebase [13].

3.5.1. Protocol

The design of the RESTful API has not been realised yet, and is planned for year 2. Therefore, currently only the "real Java API" is being used in the implementation of the prototype.

3.6. EPG Metadata Service

The demo application that will be described in detail in section 3.11 requires metadata such as title, actors, icon, etc. to display the recommendations that are calculated by the recommendation engine. This metadata is made available through the EPG Metadata Service. The service is a temporal solution and will be replaced by the Metadata Enrichment Service described in section 3.5 once its REST interface is implemented.

3.6.1. Design

The design of the EPG Metadata Service is as illustrated in Figure 15. The module uses the Restlet library for its REST interface. Furthermore is uses the DVB data parser developed by THM as part of the Test Application (see section 3.9).

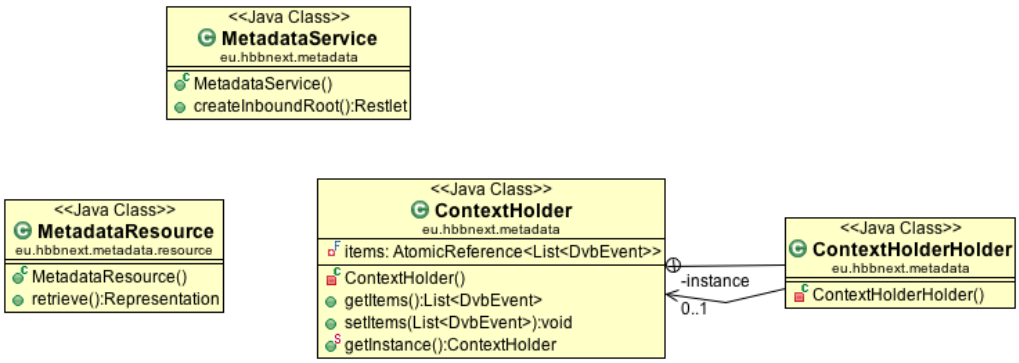


Figure 15: UML class diagram of the EPG Metadata Service

3.6.2. Protocol

The protocol of the EPG Metadata Service is very straightforward and consists of only one method that can be called using REST.

Retrieve metadata

URL: /MetadataService/<version>/metadata

Method: GET

Parameters

i Items, the itemId's for which the metadata should be retrieved (comma separated)

3.7. First Multi-User Recommendation Engine

The recommendation engine module provides functionality to Applications and Services to receive context-aware personalized multi-user and multi-device-based content recommendations. For optimizing recommendations, context information from users in a group and the collection of user profile information will be used for content filtering, as well as collaborative content filtering algorithms, all of them serving as input for the service personalization processes. The recommendation engine will enable content recommendations coming from both the broadcast and broadband domain by building on algorithms for merging the HBB metadata. For providing this functionality a metadata merging function will be implemented.

3.7.1. Design

The Recommendation Engine module is built on top of the Personal Recommendation Engine Framework (PREF), developed by TNO. PREF is a proven, flexible and scalable framework that enables researchers to easily develop recommender algorithms. It provides the means to store and access users, items, ratings and metadata. Furthermore it provides the algorithm designer with hooks and tools to create new algorithms and to reuse existing ones.

PREF has a layered architecture, as shown in Figure 16. Requests from the outside world are received by the REST API. Depending on the request, it might propagate through the layers, all the way to the database. Direct requests to the database or intermediate layers are not supported.

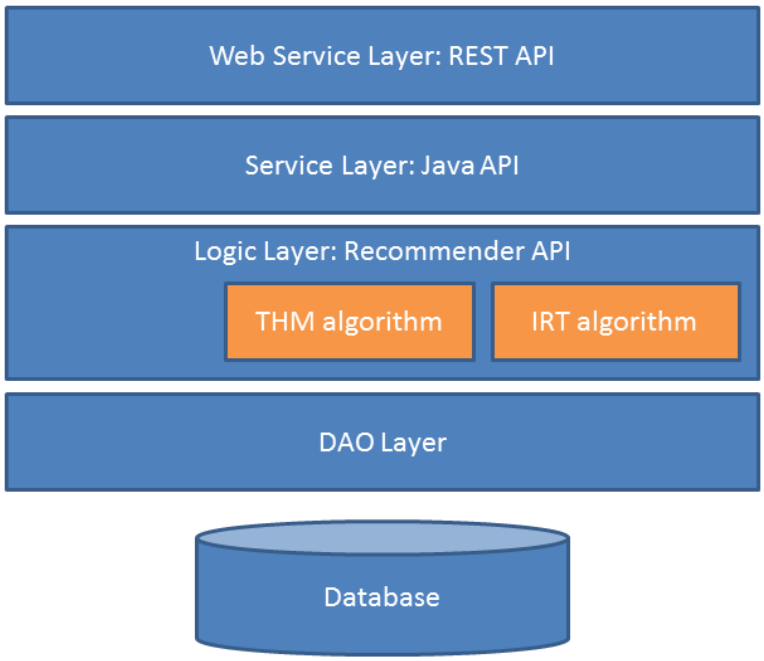


Figure 16: PREF layered architecture

Web Service Layer

The web service layer handles all the requests from the outside world. It provides a RESTful API that can be called in order to get recommendations, among others. The methods that are provided are described in detail in subsection 3.7.3.

Service Layer

The service layer is quite similar to the web service layer when it comes to the methods it offers. The purpose of the service layer is to make it easy for PREF to support different web service API's. Besides REST it supports SOAP for example. Other protocols can also be supported, simply by mapping their methods onto the methods provided by the service layer.

Logic Layer

The logic layer is where the heavy lifting is done. This layer contains all the logic that is responsible for the calculation of the recommendations.

DAO Layer

The Data Access Object layer is the layer that wraps the database and allows the logic layer to access users, items, ratings, and so on. Furthermore it provides a number of interfaces to store, edit and delete objects, without exposing the database that is used. This enables the use of different DBMS vendors, without the need to rewrite logic layer code.

Model

All the above layers, except the web service layer, share the same data model. This data model contains classes for users, items, ratings, recommendations, and so on. The web service layer might have its own data model that is translated to the inner data model by the service layer.

The algorithms designed and implemented by THM and IRT (and described in the next section) are part of the system's logic layer.

3.7.2. Algorithms

The current version of the recommendation engine utilizes a technique called collaborative filtering to calculate item recommendations. Later on in the project a hybrid approach that combines collaborative filtering with content based filtering might be investigated.

To contrive collaborative filtering, the following methodology is applied. The first step is to create a matrix that shows the correlation between the users and the items. With correlation here the user rating of the items is meant. The second step is to identify the k-nearest neighbors of the user or group of users for which the recommendation should be created. Based on the information about the k-nearest neighbors, subsequently a list of recommended genres can be created. The last task is to filter the set of items, which match the recommended genres. The result of this filtering is a set of user(s) recommended items.

User-Item matrix

A user-item matrix is a structure, which contains the users, the items and the correlation between them. For correlation here the rating by a user for an item is meant. The data in this matrix is originates from e.g. some kind of user profiles. Table 1 should illustrate such a user-item matrix.

	Item 1	Item 2	Item 3	Item 4
User 1	5	1	-	3
User 2	4	0	1	2
User 3	3	5	3	1
User 4	4	-	2	1

Table 2: Correlation between users and items.(ratings(0 to 5))

k-nearest neighbors

To calculate the k-nearest neighbor(s) an algorithm called: "Absolute Cosine Similarity" [14] is applied. Here, k represents the amount of users. The absolute cosine similarity is split into two steps. For one thing, the cosine similarity between the users is calculated. The cosine similarity is defined in equation (1). In the second step, the ratio of the vector magnitude is calculated. The ratio is defined in equation (2) and equation (3). In the equations the users are interpreted as vectors, which include the ratings.

$$CS_{sim}(i,j) = \frac{\vec{i} * \vec{j}}{\|\vec{i}\| * \|\vec{j}\|} \quad (1)$$

$$abs_{ib} = \frac{\|\vec{i}\|}{\|\vec{j}\|}, \text{ if } \|\vec{i}\| < \|\vec{j}\| \quad (2)$$

$$abs_{ib} = \frac{\|\vec{j}\|}{\|\vec{i}\|}, \text{ if } \|\vec{i}\| > \|\vec{j}\| \quad (3)$$

The result of the multiplication of the cosine similarity with the ratio of the vector is the absolute cosine similarity. Equation (4) shows the formula for the absolute cosine similarity.

$$ACS_{sim} = CS_{sim} * abs_{ib} \quad (4)$$

The value of the absolute cosine similarity made possible to specify the k-nearest users.

Recommended items

Based on the items, which are rated by the active and k-nearest users, a list of recommended items can be created. To accomplish that, the list of all items will be filtered and a list of items, which coincide with items rated by the active and k-nearest users, will be created.

Item filtering

In the list of items, rated by the active and the k-nearest users, it is not considered, that the items has different ratings. To put out the items first, which are best rated by the user, the list has to be sorted.

3.7.3. Protocol

The Multi-User Recommendation Engine provides three main methods through its REST interface: one to get content recommendations based on user/group preferences, one to get (group) recommendations similar to a given item, and one to get items similar to a specific item (based on item metadata). Each of these methods is described in more detail in annex B. The message format of the results must yet be agreed upon.

3.8. Preference Service

The Preference Service is the place where information required to calculate context-aware multi-user recommendations is stored by the Personalisation Engine and requested by the Recommendation Engine. The service contains information regarding users, items, ratings and preferences.

3.8.1. Design

Since the Preference Service is built on top of the Personal Recommendation Engine Framework, its design similarly to the Recommendation Engine. The service layer, as shown in Figure 16, is shared by both modules. The web service exposes different methods to the outside world. The Preference Service REST API contains methods to feed the recommendation engine with data on users, items and ratings. The available methods are explained in the following subsection.

3.8.2. Protocol

The REST API provided by the Preference Service contains a number of methods to create, read, update and delete (CRUD) users, items, aliases, characteristics, ratings and preferences. These methods are listed below.

- Get / create / Delete (selected) item(s)
- Get / add / update / delete item characteristics
- Get / add / update / delete item aliases
- Create / delete user(s)
- Get / add / update / delete user aliases
- Get item ratings
- Rate items / delete item and list ratings

The methods are described in detail in annex C.

3.9. Test Application

The test application creates a list of recommended TV events for a specific user or a group of users. To achieve that, the application operates with collaborative filtering like described in subsection 3.7.2.

After the start of the application, it loads the user profiles. While loading, the user valuations for the specific genres are transferred into a data structure named “User item matrix”. Also the application loads all DVB events into a list.

On the graphical user interface (Figure 18) the operator of the application has the ability to choose one or more users for whom he wants to create a recommendation. With a click on the button “Create User Recommendations” (Figure 18) the operator begin with the recommendation creation process.

Based on the user item matrix, the application calculates k (in test application k=4) users, whose valuations quite similar to the active user. Merged with the active user these users offer the genres that are interesting for the active user. As the last step the application filters the loaded dvb events with the genres which are interesting for the active user.

As result the operator gets the list of recommended TV events for the user or group of users he chose.

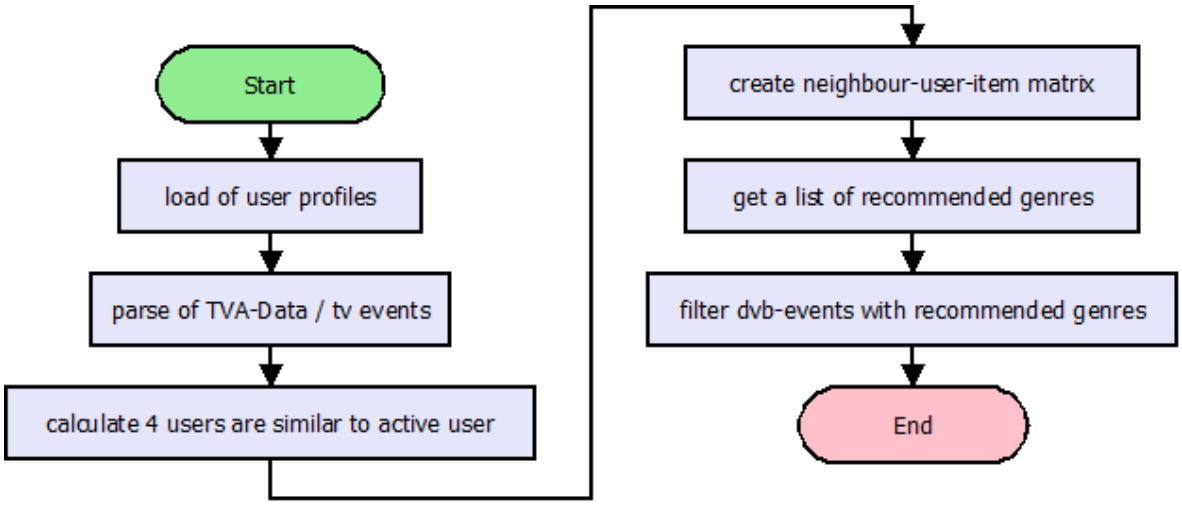


Figure 17: Simplified application flow

3.9.1. Graphical user interface

The picture below (Figure 18) shows the graphical user interface of the demo application. The list box on the left side of the application includes a list of all users loaded from user profiles. Under this list box the operator have the ability to choose between a single and a multi user recommendation. The checkboxes enable or disable the multi selection feature of the list box. The text box on the right side of the application is the output window for recommendations.

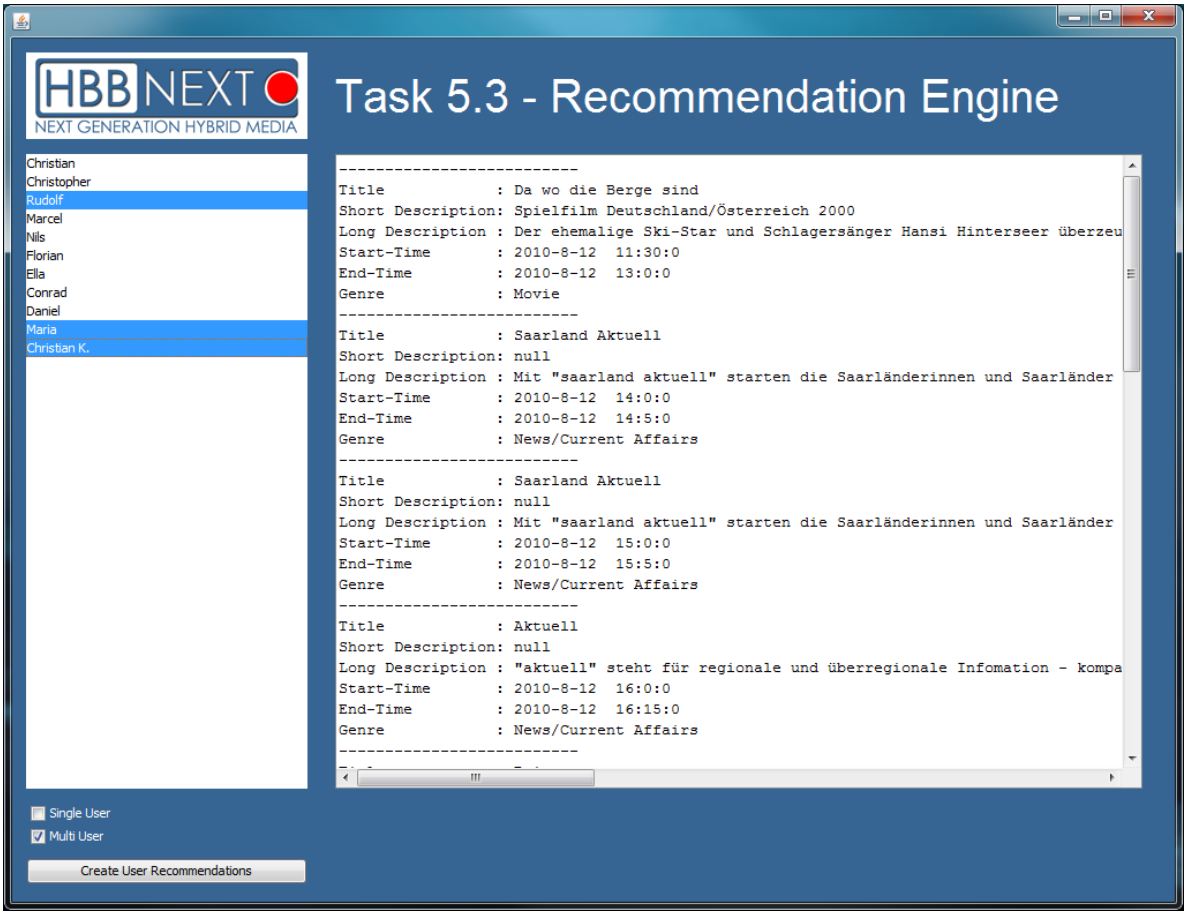


Figure 18: GUI

3.9.2. Architecture

The following illustration shows the class diagram of the recommendation engine.

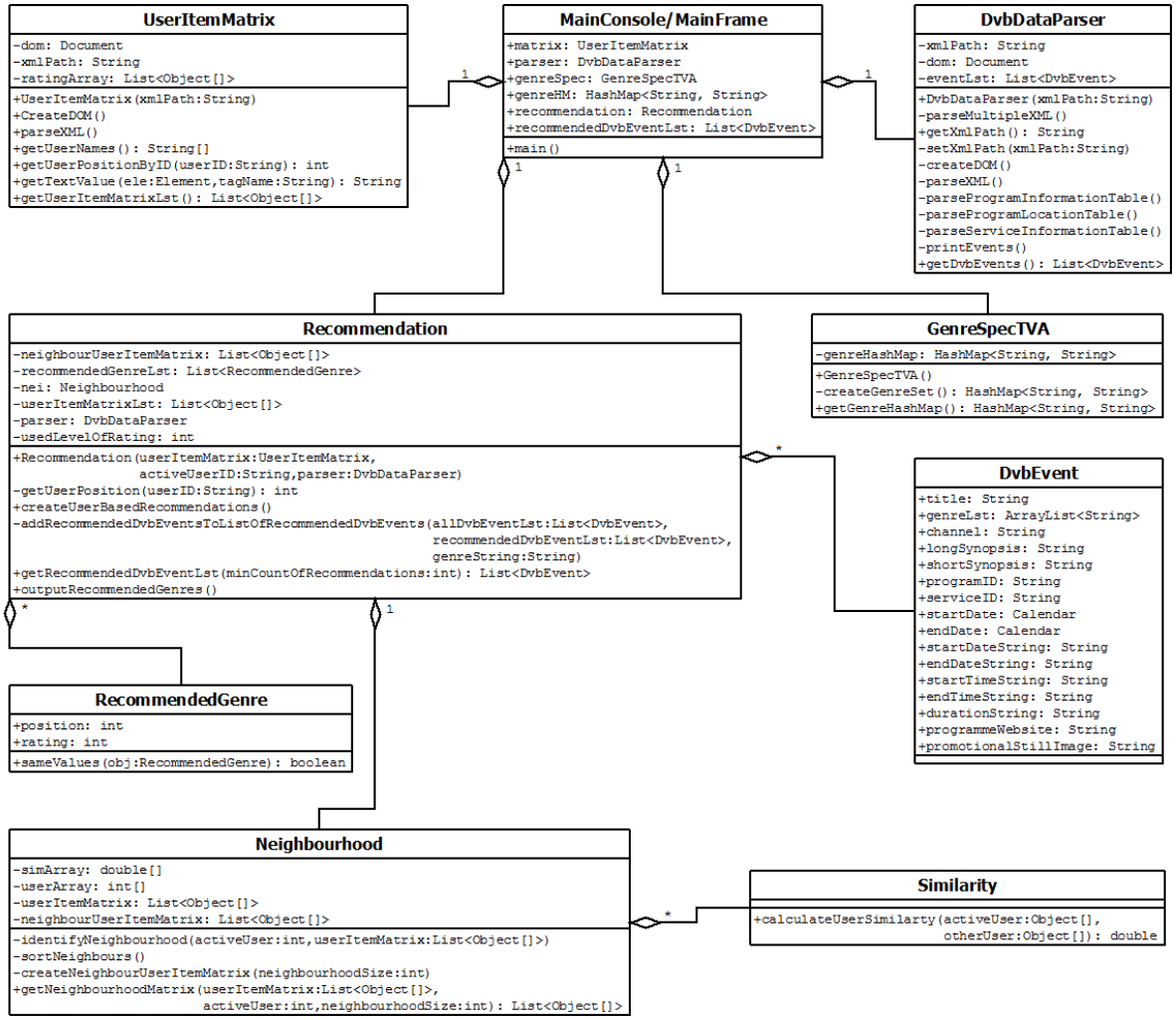


Figure 19: Class diagram

Class: MainConsole/MainFrame

The MainConsole or MainFrame class serves as the operating part of the test application.

Class: UserItemMatrix

The UserItemMatrix class is used to parse user profiles and return a user-item-matrix. Because the user profiles are in a xml file, a function is used that parse the xml files to load the user profiles. Additional this class includes functions to output the name or the position of a specific user.

Class: DvbDataParser

As well as the UserItemMatrix class the DvbDataParser class is used to load data from xml files. It loads all dvb event xml files from a specific folder. This is done by the parseXML function. To return a list of all dvb events the class offers a get function.

Class: GenreSpecTVA

This class creates a hash map which contains all possible genres with genre id and genre name.

Class: Recommendation

The Recommendation class handles the creation of a list of recommendations for a specific user. The constructor initiates the object with use of loaded data. After initialization, functions can be used to create and return a user based recommendation.

Class: DvbEvent

An object of the DvbEvent class includes all data items, are loaded for a dvb event from the dvb event xml files.

Class: RecommendedGenre

The RecommendedGenre class provides parameters to hold the position and the rating of a genre. Because genres are also saved in a list, each genre has e specific position. Then the user ratings are saved in a list of objects of the RecommendedGenre class.

Class: neighborhood

The class neighborhood includes functions to identify, sort, and return users are similar to the active user. As output an object of this class will offer a user item matrix with users are quite similar to the active user.

Class: Similarity

The algorithm to calculate the similarity between two users is implemented in this class.

3.10. Android Demo App

Users identify themselves at the Group Context Server using a smartphone by scanning a QR Code that is displayed by the HBB Next App. The App that is created for the demo runs on an Android phone. It allows you to register and unregister different users using a single smartphone. This makes possible the use of a single phone rather than several phones to demonstrate the group recommendations enabler.

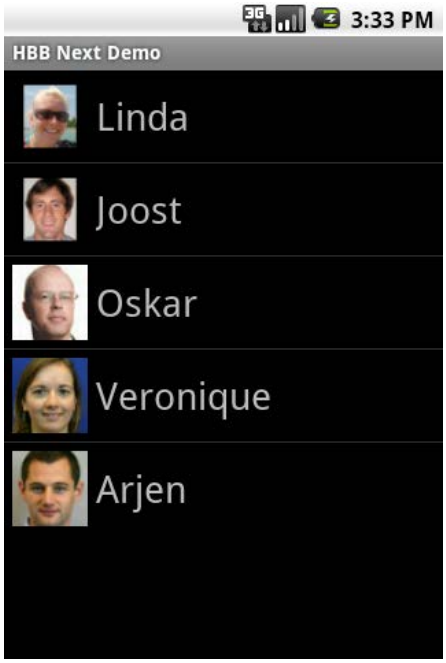


Figure 20: Screenshot of the Android App

The app lists a predefined set of users as shown in Figure 20. When a user is selected, the app switches to the QR code scanner and tries to decode the barcode. Once it detects a barcode, it decodes it and extracts the URL of the Group Context Server from it. Using this URL and the user's unique id, it registers the user at the Group Context Server. When the registration completed successfully, the app returns to the start screen, listing the users. Users can register themselves at multiple (TV) applications, as long as they display a different QR code. To see all applications at which a user is registered, perform a "long click" (click and hold the user for a while) on the user. After the long click, a user specific screen appears, listing the ids of all applications at which the user is registered. When a long click is performed on an application's id, the user is unregistered at that application.

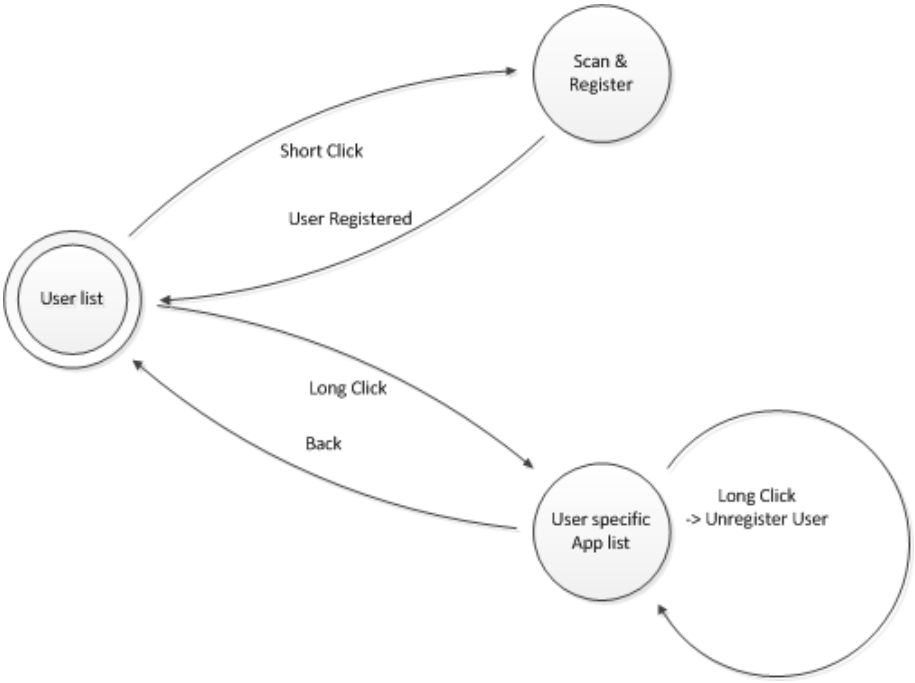


Figure 21: Android App Flow

Figure 21 shows a state diagram corresponding to the app's flow.

3.10.1. Design

The design of the Android App mainly consists of two Activities and an application context (as shown in Figure 22). The two activities correspond with the "User list" and "User specific App list" states in Figure 21. The functionality of the third state ("Scan & Register") is provided by the Android App called "Barcode Scanner". This App is called through an implicit intent provided by the ZXing library [15]. The SpecificUserActivity utilises an AppDataLoader to retrieve the active applications of a specific user. The loading of the application list happens on a separate thread to keep the user interface responsive.

The list of users that is displayed in the start screen of the app is contained by the application's context.

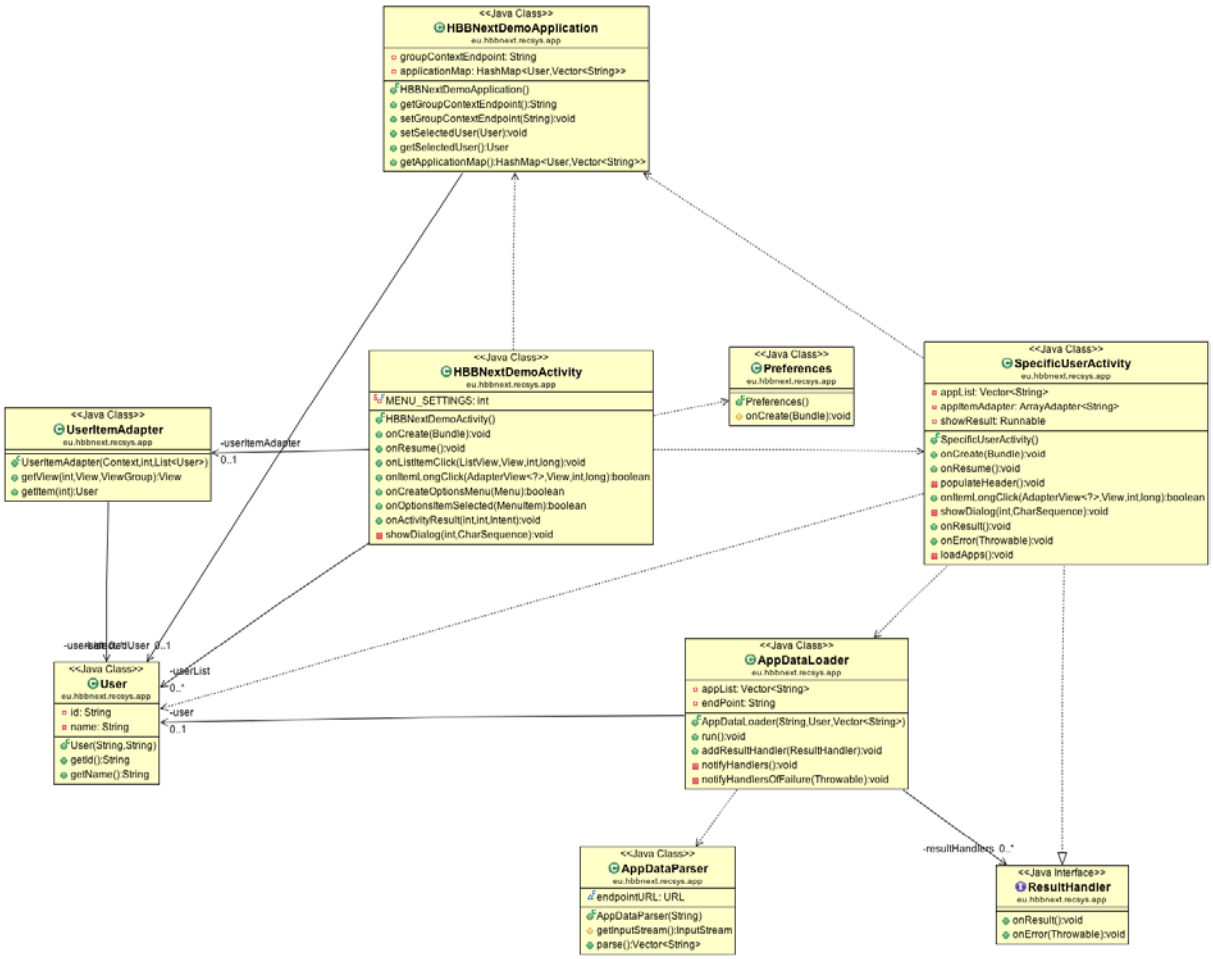


Figure 22: UML class diagram of the Android Demo App

3.10.2. Protocol

The Android Demo App does not provide an API to other modules, but is calls the Group Context Server to register and unregister users.

3.11. First demo application

All the modules described above come together in the HTML demo application. As shown in the sequence diagram (see Figure 9), the HTML application starts the sequence by registering itself to the Group Context Server. From that moment on, users can register themselves at the application with their Android smartphone. When they do, the group context changes and an event is sent to the HTML application. The application requests the new group context and retrieves the corresponding recommendations from the recommendation engine. In order to display the recommendations, metadata is requested from the metadata service.

Then the application waits for the next change in the group context. When the browser window is closed, the application unregisters itself from the Group Context Server.

3.11.1. Design

The HTML demo application consists of a single HTML file that includes/contains some JavaScript and CSS. The JavaScript that is included is the JQuery 1.7.1 library, which is used to perform AJAX requests and to modify the page contents. The included CSS is used to style the webpage.

3.11.2. Protocol

The HTML demo application does not provide an API to other modules.

4. References

[1] DublinCore: http://dublincore.org

[2] ETSI TR 102033: Architectural framework for the delivery of DVB-services over IP-based networks: http://www.etsi.org/deliver/etsi_tr/102000_102099/102033/01.01.01_60/tr_102033v010101p.pdf

[3] ETSI TS 102034: Transport of MPEG-2 TS Based DVB Services over IP Based Networks: http://www.etsi.org/deliver/etsi_ts/102000_102099/102034/01.04.01_60/ts_102034v010401p.pdf

[4] ETSI TS 102814: Ethernet Home Network Segment: http://www.etsi.org/deliver/etsi_ts/102800_102899/102814/01.01.01_60/ts_102814v010101p.pdf

[5] ETSI TS 102822: Broadcast and On-line Services: Search, select, and rightful use of content on personal storage systems ("TV-Anytime"); Part 3: Metadata; Sub-part 1: Phase 1 - Metadata schemas: http://www.etsi.org/deliver/etsi_ts/102800_102899/1028220301/01.04.01_60/ts_1028220301v010401p.pdf

[6] YouTube: www.youtube.com

[7] YouTubeGData: YouTube GData API, http://code.google.com/apis/youtube/2.0/reference.html and http://code.google.com/apis/youtube/developers_guide_protocol.html

[8] TVAnytime Forum: http://tech.ebu.ch/tvanytime/

[9] MPEG, MPEG-7 Overview, Oct 2004, http://www.chiariglione.org/mpeg/standards/mpeg-7/mpeg-7.htm

[10] MPEG-21 DIDL Format: http://xml.coverpages.org/MPEG21-WG-11-N3971-200103.pdf

[11] DBPedia: http://www.dbpedia.org

[12] iMDB: http://www.imdb.com

[13] FreeBase: http://www.freebase.com

[14] Uberall2012: Phd Thesis - Christian Überall – A Dynamic Multi-Algorithm Collaborative-Filtering System – 2012

[15] ZXing: http://code.google.com/p/zxing/

[16] XMPP: http://xmpp.org

[17] RabbitMQ: http://www.rabbitmq.com

[18] RabbitMQ javascript: http://www.rabbitmq.com/blog/tag/javascript

[19] OpenStack: http://www.openstack.org

[20] Deliverable 6.1.1: http://www.hbb-next.eu/index.php/documents

[21] Deliverable 3.2: http://www.hbb-next.eu/index.php/documents

[22] Restlet: http://www.restlet.org

5. Annexes

A. GroupContext Server API specification

The methods that are provided by the Group Context Server are specified below.

I. Get registered applications

Get all applications that registered themselves at the Group Context Server.

URL: /GroupContextService/<version>/apps

Method: GET

II. Unregister all applications

Unregister all applications that are registered at the Group Context Server.

URL: /GroupContextService/<version>/apps

Method: DELETE

III. Register application

Register an application with a specific application ID.

URL: /GroupContextService/<version>/apps/<appId>

Method: PUT

IV. Unregister a specific application

Unregister an application with a specific application ID.

URL: /GroupContextService/<version>/apps/<appId>

Method: DELETE

V. Register user

Register a user with a specific user ID at the specified application.

URL: /GroupContextService/<version>/apps/<appId>/users/<userId>

Method: PUT

VI. Unregister user

Unregister a user with a specific user ID from the specified application.

URL: /GroupContextService/<version>/apps/<appId>/users/<userId>

Method: DELETE

VII. Get registerd users

Get a list of all users that are currentlu registered at the specified application.

URL: /GroupContextService/<version>/apps/<appId>/users

Method: GET

VIII. Wait for group context change

Issue a long polling request, waiting for a group context change event.

URL: /GroupContextService/<version>/apps/<appId>/context

Method: GET

B. Recommendation Engine API specification

The methods that are provided by the Recommendation Engine are specified below.

I. Get recommendations

URL: /RecEngine/<version>/<party>/recommendations

Method: GET

Parameters

g	Group, specified by their foreignUserIds (comma separated)
s	Size, the maximum number of items to return
rc	RecommendationContext paremeters, specified as “<tag>:<value>” (comma separated)

II. Get content-content recommendations

URL: /RecEngine/<version>/<party>/recommendations/similarto/<foreignItemId>

Method: GET

Parameters

g	Group, specified by their foreignUserIds (comma separated)
s	Size, the maximum number of items to return
rc	RecommendationContext paremeters, specified as “<tag>:<value>” (comma separated)

III. Get similar content

URL: /RecEngine/<version>/<party>/similarto/<foreignItemId>

Method: GET

Parameters

g	Group, specified by their foreignUserIds (comma separated)
s	Size, the maximum number of items to return
rc	RecommendationContext paremeters, specified as “<tag>:<value>” (comma separated)

C. Preference Service API specification

The methods that are provided by the Preference Service are specified below.

I. Get (selected) items

URL: /PrefService/<version>/<party>/items

Method: GET

Parameters

q	Query used to filter the items, represented as “<tag>:<value>”, where <tag> and/or <value> can be *
ps	PageSize, the maximum number of results to return
p	Page, the page to return
e	Expand, the properties to expand (comma separated)

II. Get specific item

URL: /PrefService/<version>/<party>/items/<foreignItemId>

Method: GET

Parameters

e Expand, the properties to expand (comma separated)

III. Create items

URL: /PrefService/<version>/<party>/items

Method: PUT

Body

```
<items>
     <item>…</item>
     <item>…</item>
     …
</items>
```

IV. Create single item

URL: /PrefService/<version>/<party>/items/<foreignItemId>

Method: PUT

V. Delete items

URL: /PrefService/<version>/<party>/items

Method: DELETE

Body

```
<items>
     <item>…</item>
     <item>…</item>
     …
</items>
```

VI. Delete single item

URL: /PrefService/<version>/<party>/items/<foreignItemId>

Method: DELETE

VII. Get item characteristics

URL: /PrefService/<version>/<party>/items/<foreignItemId>/characteristics

Method: GET

Parameters

q Query used to filter the item characteristics, represented as "<tag>:<value>",

where <tag> and/or <value> can be *

ps PageSize, the maximum number of results to return

p Page, the page to return

VIII. Add/Update item characteristics

URL: /PrefService/<version>/<party>/items/<foreignItemId>/characteristics

Method: PUT/POST

Body

```
<characteristics>
      <characteristic>
            <tag>…</tag>
            <value>…</value>
      </characteristic>
      …
</characteristics>
```

IX. Delete item characteristics

URL: /PrefService/<version>/<party>/items/<foreignItemId>/characteristics

Method: DELETE

Parameters

t Tags, the characteristics with these tags should be deleted (comma separated)

X. Get item aliases

URL: /PrefService/<version>/<party>/items/<foreignItemId>/aliases

Method: GET

XI. Add/Update item aliases

URL: /PrefService/<version>/<party>/items/<foreignItemId>/aliases

Method: PUT/POST

Body

```
<aliases>
     <alias>
          <foreignid>…</foreignid>
          <otherparty>…</otherparty>
          <otherforeignid>…</otherforeignid>
     </alias>
     …
</aliases>
```

XII. Delete item aliases

URL: /PrefService/<version>/<party>/items/<foreignItemId>/aliases

Method: DELETE

Parameters

p Parties, the partyId's for which the aliases must be deleted (comma separated)

XIII. Create users

URL: /PrefService/<version>/<party>/users

Method: PUT

Body

```
<users>
     <user>…</user>
     <user>…</user>
     …
</users>
```

XIV. Create single user

URL: /PrefService/<version>/<party>/users/<foreignUserId>

Method: PUT

XV. Delete users

URL: /PrefService/<version>/<party>/users

Method: DELETE

Body

```
<users>
     <user>…</user>
     <user>…</user>
     …
</users>
```

XVI. Delete single user

URL: /PrefService/<version>/<party>/users/<foreigUserId>

Method: DELETE

XVII. Get user aliases

URL: /PrefService/<version>/<party>/users/<foreignUserId>/aliases

Method: GET

XVIII. Add/Update user aliases

URL: /PrefService/<version>/<party>/users/<foreignUserId>/aliases

Method: PUT/POST

Body

```
<aliases>
     <alias>
          <otherparty>…</otherparty>
          <otherforeignid>…</otherforeignid>
     </alias>
     …
</aliases>
```

XIX. Delete user aliases

URL: /PrefService/<version>/<party>/users/<foreignUserId>/aliases

Method: DELETE

Parameters

p Parties, the partyId's for which the aliases must be deleted (comma separated)

XX. Get item ratings

URL: /PrefService/<version>/<party>/users/<foreignUserId>/ratings

Method: GET

Parameters

i Items, the itemId's for which the ratings must be retrieved (optional, comma separated)

q Query used to filter the items for which the user's rating must be returned, represented as "<tag>:<value>", where <tag> and/or <value> can be *

ps PageSize, the maximum number of results to return

p Page, the page to return

Rate items

URL: /PrefService/<version>/<party>/users/<foreignUserId>/ratings

Method: PUT

Body

```
<ratings>
    <rating>
        <foreignitemid>…</foreignitemid>
        <groupcontext>
            <foreignuserid>…</foreignuserid>
            <foreignuserid>…</foreignuserid>
            …
```

```
            </groupcontext>
            <utility>…</utility>
            <confidence>…</confidence>
      </rating>
      …
</ratings>
```

XXI. Delete item ratings

URL: /PrefService/<version>/<party>/users/<foreignUserId>/ratings

Method: DELETE

Parameters

i Items, the itemId's for which the ratings must be deleted (comma separated)

XXII. Get recommendation list ratings

URL: /PrefService/<version>/<party>/listratings

Method: GET

Parameters

l Lists, the listId's for which the ratings must be retrieved (comma separated)

XXIII. Rate recommendation list

URL: /PrefService/<version>/<party>/listratings

Method: PUT

Body

```
<ratings>
      <rating>
            <listid>…</listid>
            <utility>…</utility>
            <confidence>…</confidence>
      </rating>
      …
</ratings>
```

XXIV. Delete recommendation list ratings

URL: /PrefService/<version>/<party>/listratings

Method: DELETE

Parameters

l Lists, the listId's for which the ratings must be deleted (comma separated)