A context-aware architecture for IPTV services personalization

Marek Dabrowski*, Justyna Gromada
Orange Labs Poland
Warsaw, Poland
{marek.dabrowski, justyna.gromada}@orange.com

Hassnaa Moustafa†
Intel Labs, Intel Corporation
Hillsboro, Oregon, USA
hassnaa.moustafa@intel.com

Jacky Forestier
Orange Labs, France
Paris Area, France
jacky.forestier@orange.com

Abstract

TV and video services landscape is currently undergoing significant changes. Traditional TV broadcasting model is supplemented and often even replaced by digital content distribution services over the Internet. As a result of this trend, a range of services and contents available for users is rapidly expanding. As a side-effect, designing efficient user interfaces for discovering the content, as well as for manipulating associated interactive services, becomes more and more cumbersome. Pervasive computing and context-awareness principles seem to be promising for making user interaction with the system more seamless and fluid. This paper discusses context-awareness as a new trend for services personalization, defining the meaning of ”context” and different contextual data relevant for IPTV (Internet Protocol Television) services. A novel architecture for unified storage and processing situational data in IPTV service domain is presented, together with discussion of its implementation issues and validation by testbed experiments. The UP-TO-US context-aware architecture is designed for seamless monitoring of user's environment (including networks and terminals), interpreting user's requirements and making the user's interaction with the TV system dynamic and transparent. Consequently, content personalization is achieved, matching the user's needs and current state of his surrounding environment.

Keywords: IPTV, architecture, context-awareness, personalization.

1 Introduction

Ubiquitous access to TV and video content becomes a part of everyday life for most of the people. With the advancement in electronic equipments and the networks capacities, users are accustomed to accessing multimedia content anytime and anywhere. They copy video and movie files to their devices that allow consuming the content while being away from home, they actively search for video content on Internet, they access and watch movies thanks to their Video on Demand (VoD) subscription. However, growth of content catalogues and proliferation of additional interactive services results in undesired complication of user interaction with the system. TV service customization for a user is considered as promising solution to save time and effort required for browsing vast catalogues of available contents and zap between large number of programs that appear on the TV menu. One of the possible ways to achieve personalization of TV services is through providing customized recommendations for users, matching their interests

*Corresponding author: Orange Labs Poland, St. Obrzezna 7, p. 229, Zaklad Platform Uslugowych i Middleware, 02-691 Warsaw, Poland, Tel: +48-226995706

†This work has been done by the author during her stay in Orange Labs

and preferences. Vast literature is available on automatic recommendation systems (see e.g. quite recent results of real-life experiments with TV recommender systems presented in [10]). Recommender systems may analyze user's previous content choices to learn and match his habits, or may exploit the similarities between certain user categories to infer their likes and dislikes based on observation of other similar users. Going further, context-aware systems aim to adapt automatically to user's situation, e.g. a context-aware IPTV could find the program that suits the user and his current situation: staying at home or on the train, in the noon or in the evening, being in front of TV set or mobile device. In each situation, the expected behavior and service choice of the user may be different and one can expect that some similarities and patterns could be found and exploited by the recommendation algorithm, e.g. knowing that a user particularly likes watching sports in the evening time, while staying in his bedroom.

The work presented in this paper explores the context-awarenss paradigm in the area of IPTV and identifies essential building blocks that have to be deployed for gathering, storing and providing contextual data to IPTV service applications. A novel context-aware architecture presented here removes the barriers of previous works by providing a unified and standarizable representation of contextual data from different domains (user, network and context). Combining this information with static user profile provides the IPTV system with a deep knowledge of the user, his preferences and current situation (location, type of device, bandwidth constraints). UP-TO-US architecture is open an extensible thanks to relying on easily accessible web services interfaces. It protects privacy of sensitive data by giving users full control over their personal information, whether explicitly provided or implicitly gathered by the system. Consequently, the personalization of service may be greatly improved by dynamic learning of user's lifestyle and more precisely anticipating his needs in given situational context. In particular, promising use-cases supported by UP-TO-US context-aware architecture are:

- *Content adaptation according to each individual user and group users' preferences*: allowing each user or a group of users to have personalized content matching their preferences.
- *Content customization according to the user context and QoE (Quality of Experience)*: allowing each user to have personalized content matching the user context (age, gender, region, preferences, location in the home environment or outside, activity) and thus optimizing the level of satisfaction.
- *Content following the user during his mobility in his domestic sphere*: allowing the user to move around within his domestic sphere while continuing accessing his IPTV service personalized according to the characteristics of the device in his proximity.
- *Content personalization during nomadism*: allowing the user to access his personalized IPTV content in a nomadic situation like in a hotel, in a friend's house or anywhere outside his domestic sphere.

The rest of the paper is organized as follows. Section 2 gives an overview of the related works on context-awareness for IPTV services and discusses some of their limitations . Section 3 introduces a context-aware IPTV architecture developed within the UP-TO-US project, aiming to provide enhanced IPTV services personalization. Then, section 4 discusses the implementation issues and details of UP-TO-US prototype and section 5 its validation by testbed experiments. Finally, the paper ends by a conclusion highlighting some points for the future work.

2 Related works

2.1 Context definition

The idea of utilizing environment information in computing systems was advocated by Mark Weiser [16]. He first introduced the term 'pervasive' which refers to the seamless integration of devices into the users' everyday life. However, the term "context-aware computing" was not defined until 1994 by Schilit and Theimer [13], who described context as "location of use, the collection of nearby people and objects, as well as the changes to those objects over time". Much of the early work on context-aware systems used similar extensional definitions which defined context by enumerating the constituting parameters. Brown et al. [5] enumerate "location, time of day, season of the year, and temperature". These definitions are very special and only reflect the types of information that have been used by the researchers in their context-aware applications. A general definition of context was proposed by Chen and Kotz [6]: "Context is the set of environmental states and settings that either determines an application's behavior or in which an application event occurs and is interesting to the user." Considering the IPTV service, context can be considered as any information that can be used to characterize the situation of an entity related to the IPTV service. An entity could be the user, device, network and service itself. Thus we define four types of contexts for IPTV chain including *user*, *device/terminal*, *network* and *service* domains. In order to enable context-aware IPTV for enriched services personalization, variety of information expressing current situation of user, device, network, content and service needs to be collected and processed. Such information is called contextual information and needs to be efficiently gathered and processed in real-time during service access.

User context includes information about the user, which could be static information, dynamic information and inferred information. The static information describes the user's personal information which is stored in the database (ex, name, age, gender, geographical region, and input preferences). The dynamic information is captured by sensors or by other services (e.g. user's location "where in the domestic sphere or outside" , agenda, usage history, activities/mood, perceived quality). The inferred information is a high-level information, which is derived from other information (e.g. user's action "user is going to the bed" is inferred from the change of location).

Device/terminal context includes information about the devices (terminals), which could be the device identity, location, status (turned on or off, volume), device capabilities including the screen capacity, device proximity with respect to the user, available network connectivity at the device, active session at the device.

Network context includes information about the network. It includes the access technology type, e.g. ADSL (Asynchronouos Digital Subscriber Line), fiber, 3G or LTE (Long Term Evolution) and its maximum transmission capacity, which for a given user may be limited by his subscription parameters. It concerns also dynamic QoS (Quality of Service) information like current load, available bandwidth, experienced packet loss ratio, delay and jitter.

Service context includes information about the service, which could be the service description and requirements. For example, the information about the broadcast TV and VoD (Video on Demand) service includes the content language and format, the content description (content metadata: title, gender, start and end time, channel, actors, directors, synopsis, episode name, episode ID, audience, duration, year, country of origin, studio, available languages), the content location and access rights and the user's content consumption history (view, purchases, ratings).

Privacy protection is a critical issue for personalized IPTV services acceptability by the users. With the added capability of viewing online content and accessing Internet services, it is more urgent for IPTV service providers to incorporate mechanisms into their delivery platforms to preserve their customers' privacy. Privacy protection is also a valuable asset that IPTV services must guarantee to differentiate this premium offer from Web approach where privacy may be weakened. For assuring privacy the collection of personal data should be controlled and identification/authentication/authorization should be carried out in an effective manner. In addition, handling personalized IPTV services require enhancement of user identity management. Some direct requirements are: i) STB (Set-Top-Box) devices must be able to manage personal accounts and numerous innovating technologies that are available for a user-friendly logging on. One can mention NFC (Near Field Communication) or RFID (Radio Frequency Identification) technologies. ii) Binding of different user identifiers and handling of heterogeneous methods for user identification and authentication must also be supported. iii) Access to third party services should also benefit from Single Sign On procedure once the user is logging on the IPTV home page.

2.2 Context-aware IPTV solutions

Although context-awareness can bring additional smart services and useful functionalities to IPTV systems and allow for enriched personalization, it is still a challenge for operators to make IPTV services context-aware and to use the acquired context information to realize personalized TV services. Several solutions for context-aware TV systems have been proposed employing either a distributed or a centralized approach. In the former, several entities have the capacity to acquire, process, and store the context information, while cooperating with each other to provide context-aware services. While, the later approach treats and stores the collected context information in a centralized server that could be located in the domestic sphere (for instance, server attached to the STB - ”partially centralized” or in the operator network ”fully-centralized”).

A distributed context-aware interactive TV solution is proposed in [11], implementing software agents on top of physical devices (STBs, mobile phones) for context acquisition, treatment and storage, where each device agent discovers the other devices agents for exchanging/analyzing the context information acquired. In this solution the user locations are acquired by sensors while other context information of terminals (e.g. screen size, supported content format) is provided by the terminals themselves. The network related context is acquired through the ”active network” principle, used to route the content according to the context (e.g. traffic condition, user's location, etc.) through active nodes that process the contextual information and make a decision on the best network path to deliver the content. Finally, the service context information is provided by the content provider and is stored in the content server at the service provider or operator side. In this solution, the computing capacity of the distributed devices limits the service performance, and the context information is simple and falls short to reflect the user preference, which in turn limits the service personalization.

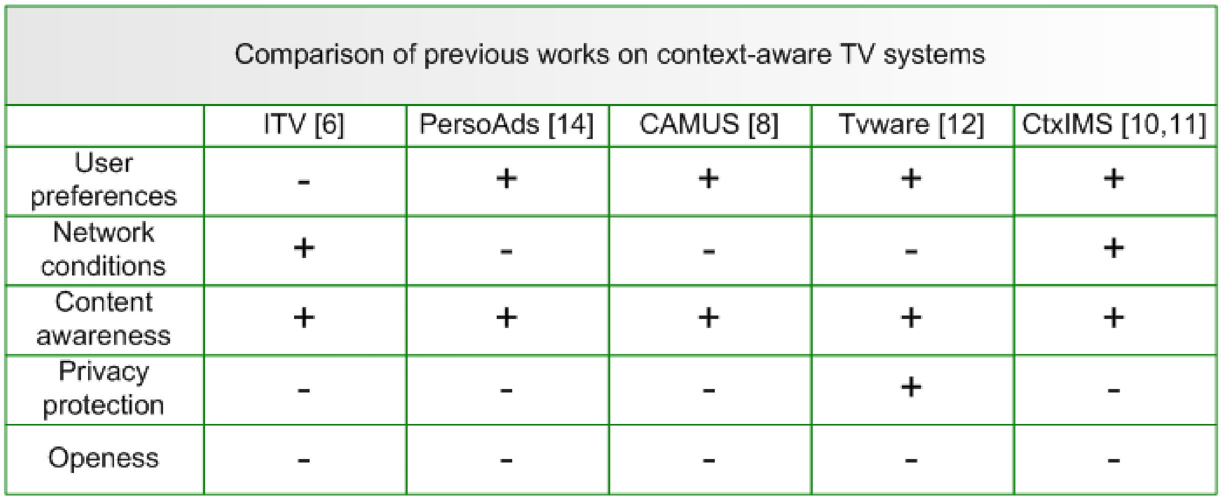
In [4], a partially-centralized context-aware TV architecture is proposed for the selection and insertion of personal advertisement in the broadcast content, based on the aggregation of past sequence of individual contexts (i.e. past viewing) and the association of the current user context to those past contexts in order to determine the most appropriate advertisement. The context is described as: i) user location at home, ii) identity of user and device, content identity and other related information such as category, channel type), and event identity (information that might have an impact on the user's watching behavior), iii) user's activity and iv) time. This solution is mainly targeting home networks and does not consider the network contexts and the service personalization is limited to personal advertisement insertion.

A client-server approach based TV system is proposed in [12] aiming to realize the TV Set automatic control and personalized content recommendation through presenting a personalized EPG (Electronic Program Guide). A Service Agent Manager (SAM) communicates with physical devices (sensors or equipment like TV Set), and notify the CAMUS (Context-Aware Middleware for Ubiquitous System) when each device detects some noticeable changes. On the other hand it receives commands from the CAMUS to control the devices. The context information used in this solution does not consider the network context. In addition, no privacy protection exists for the user context information.

Another client-server approach based personalized recommendation architecture for Digital TV is presented in [7] in which the used context information is divided into five dimensions: who (identity), when (time), where (location), what (activity, the content information), and how (how can the user receive the service, through a mobile, portable or fixed device?). The privacy protection in this work is carried out through storing the user's personal context in his devices, which is not an always practical and efficient approach, and it limits the nomadic service access through passing by other's devices.

A proposal of context-aware IPTV architecture based on NGN (Next Generation Network) and IMS (IP-Multimedia Subsystem) has been presented in [14] and [15], from which the architecture presented in this paper inherits many principles and design of the Context-Awareness System. However, [14] and [15] have been focused on specific IPTV platform type that was based on IMS network standards, while the proposal discussed in this paper is agnostic to IPTV solution and could be plugged to any external system thanks to well defined and simple HTTP-based interfaces.

Figure 1 presents a comparison of previous works, taking into account such criteria like awareness of user preferences (profile), awareness of network conditions (like bandwidth constraints), awareness of content data, strict privacy protection of user data, and openess of architecture (flexibity to support various services and gather contextual information from different underlying content distribution architectures). As one can notice, prior works are limited and do not fully support requirements for openess, security of users data and completness of gathered context data.



Comparison of previous works on context-aware TV systems

	ITV [6]	PersoAds [14]	CAMUS [8]	Tvware [12]	CtxIMS [10,11]
User preferences	-	+	+	+	+
Network conditions	+	-	-	-	+
Content awareness	+	+	+	+	+
Privacy protection	-	-	-	+	-
Openess	-	-	-	-	-

Figure 1: Comparison of previous works

The context-aware IPTV system discussed in this paper has been designed as common effort of researchers from four European countries in the scope of EUREKA/CELTIC project UP-TO-US [3]. General assumptions and high-level architecture of UP-TO-US Context-Aware System (CAS) have been previously presented in paper [8]. CAS collects, processes and delivers context information to context-aware applications. Unlike previous works, the UP-TO-US approach takes into account different types of context data in IPTV value chain (user, network and service), combining it with dynamically learned user's lifestyle model in order to provide a personalized IPTV experience. Thanks to a modular layered architecture, standards-based data model and easily accessible web interfaces, the UP-TO-US CAS is open and flexible to accomodate new services, and it can be plugged to any underyling content distribution architecture. In particular, it supports in unified way the managed operator-controlled IPTV platforms, and open WebTV architectures. Strong emphasis is put on protecting user's sensitive data, giving customers full control over how their private data is used in the system. In this paper the generic architecture described in [8] is elaborated with more detailed definitions of communication flows and messages exchanged on system interfaces. Moreover, the prototype implementation is described, as well as results of experiments which prove validity of proposed approach.

3 UP-TO-US context-aware architecture

From high-level system architecture point of view, CAS is situated on top of the network and it is agnostic with respect to the underlying network and IPTV architecture. It is designed to serve as a universal context aggregator and provider of context information to other entities of the global architecture. As depicted in Figure 2, context-awareness architecture in UP-TO-US is a three-tier architecture, with the following layers distinguished: Layer 1 (Context Acquisition), Layer 2 (Context Data) and Layer 3 (Context Applications)

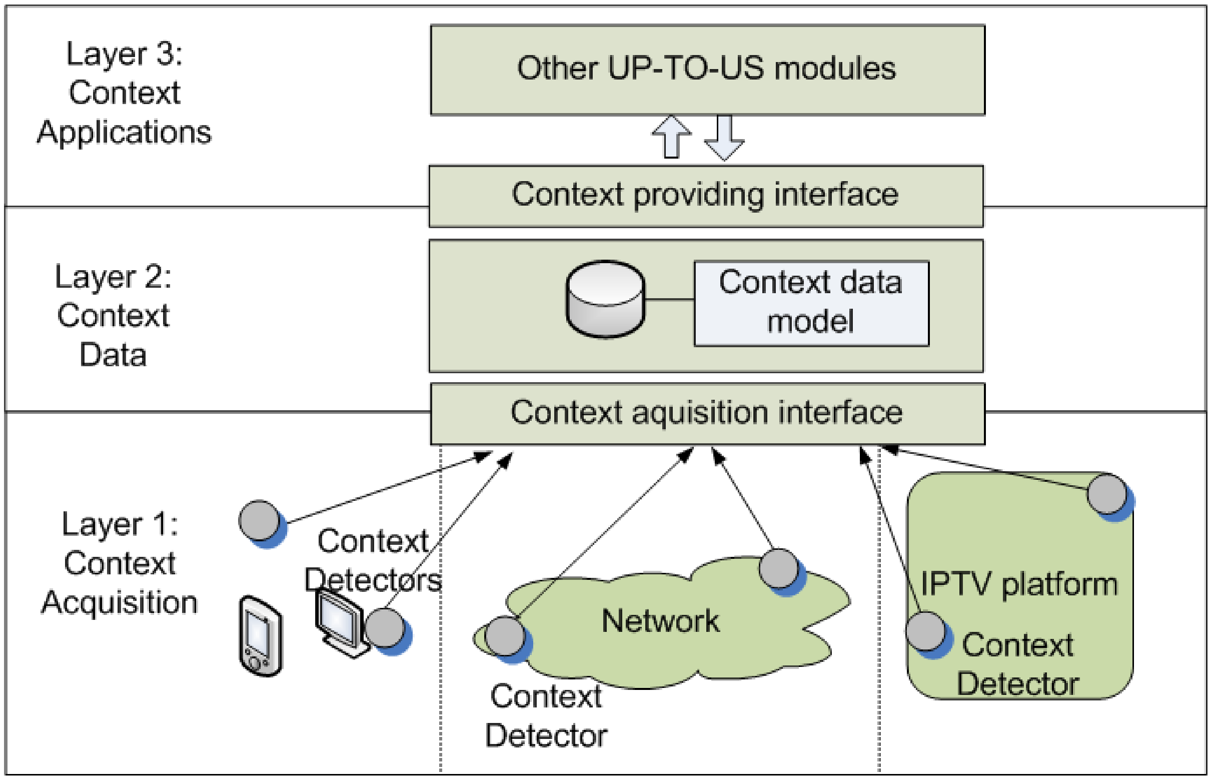


Figure 2: High-level context-awareness architecture in UP-TO-US

Layer 1: Context Acquisition. The Context Detectors located in this layer gather and pre-process raw context data. The Context Detector can be a physical sensor, e.g. RFID reader, or a logical sensor: piece of software embedded in other architecture element (e.g. STB or IPTV platform server). Context Detectors can use different context gathering methods and protocols. UP-TO-US specifies a common interface through which Context Detector can put gathered data in CAS database. From the point of view of type of collected context information (see section III for more details on context types), three context domains are distinguished:

- User context domain: environment sensors in the home sphere (e.g. for user location information), as well as information from user equipment (e.g. about available devices and their capabilities),
- Network context domain: network context information like bandwidth, QoS, etc.,
- Service context domain: IPTV service metadata.

Layer 2: Context Data. UP-TO-US Context-Awareness System (CAS) is situated at this layer. CAS is a central repository of context information in UP-TO-US system. Its main role is to:

- Aggregate, store and process context data from user, network and service domain,
- Provide context to other modules in a unified way,
- Trigger actions of other UP-TO-US modules upon detection of changes in dynamic context information.

The heart of the system is a contextual database, storing context data accordingly to technology agnostic generic context model. The context database stores the dynamic context information and relies on User Profile Module (UPM) for managing static user profiles. CAS context acquisition interface is a kind of adapter to technology-dependent context sensing systems. It maps information from context sensors to the generic internal context model of CAS. The CAS data model (see Figure 3) describes the user, his environment (terminals, locations and networks), services (subscriptions, available content and consumed content) as well as currently active content sessions. Design of the data model has been a challanging task, as it has to describe entities of different universes, and at the same time be compressed in a structured, comprehensive information format allowing for efficient data exchange between UP-TO-US modules. As a design choice, CAS data model extends standardized metadata description format TVAnytime [2]. CAS data is exposed through context providing interface based on web-services technology and thus enabling easy access to contextual information by upper-layer applications (respecting user privacy rules about which data can or cannot be exposed).

Layer 3: Context Application. This layer comprises other modules of UP-TO-US system, which act at the application layer and extend the standard IPTV platform capabilities to provide context-aware and personalized IPTV services to the users. The application layer modules, which interact with context information stored in CAS, are:

- Recommendation System (RecSys) uses dynamic context information for producing content recommendations that are best adapted to the user profile and current situation, e.g. user location, used device, current QoS and expected QoE (Quality of Experience) level, etc.
- User Profile Module (UPM) analyzes user's content consumption history traces collected by CAS for the purpose of dynamic learning user preferences and habits. This dynamic profile may be additionally enhanced with information from user's profiles on social networking sites (e.g. what

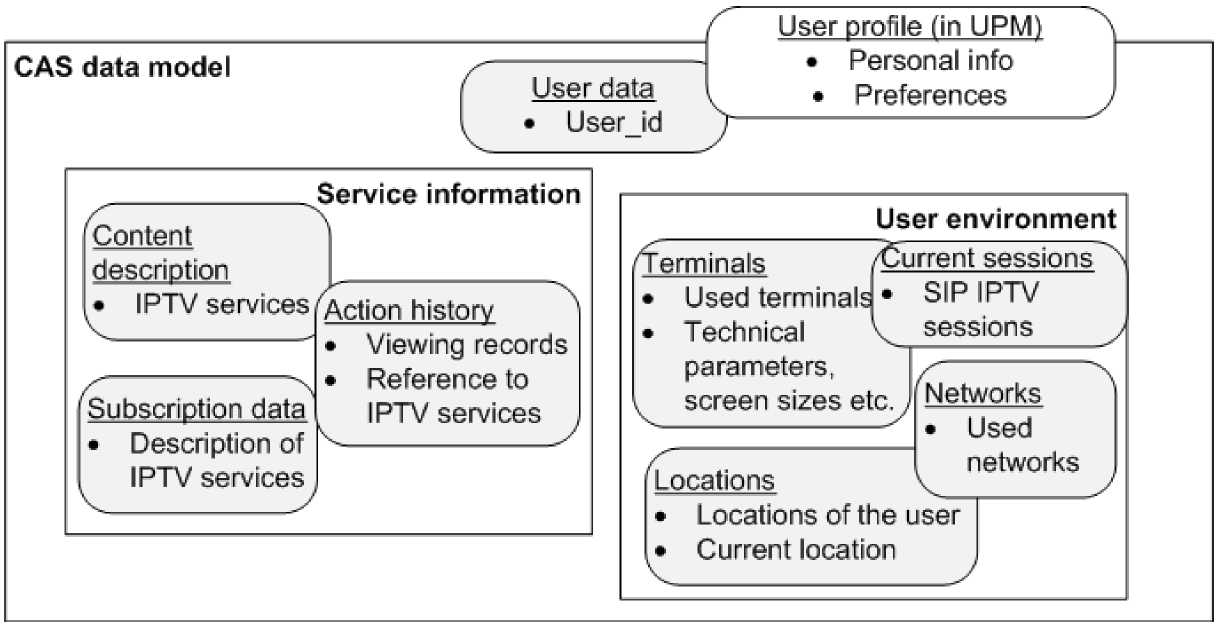


Figure 3: CAS Data Model overview

the user likes, his preferred activities, friends data), which are collected by component called Social Networking and Linked Data Module [9].

- UE Service Continuity Module uses context information to manage the mobility procedures. Terminal mobility and session mobility procedures may be activated based on context, e.g. currently available devices in user's vicinity, their capabilities etc.

- Nomadic Service Module (NSM) manages the nomadic access procedures, e.g. access to my IPTV services from friend's house (served by the same or other IPTV operator), or from a hotel room. The nomadic status of a user (whether he is in his own house or in nomadic situation) will become a part of dynamic status information.

Any operation of providing context data by CAS to other module will be preceded by consulting the Privacy and Security Module (PS) for verifying the user policies concerning access to context data. Users are able to configure flexible policies regarding who and in which situation may access their context data. Full user control over access to context information (possibility to easily block and restore access) is crucial for maintaining the trust and acceptance of context services which make use of highly sensitive and private data.

4 Context-Aware System implementation in UP-TO-US

A prototype of UP-TO-US system has been implemented by the project and deployed in partners testbeds for evaluation testing. Although a complete system has been implemented including all UP-TO-US architecture modules, this paper focuses specifically on context-awareness subsystem as described in section 3. The implementation technologies have been chosen aiming to meet the basic requirement of hiding the complexity of underlying data model (with data gathered from user domain and from different entities of end-to-end IPTV architecture) within a set of simple and easily accessible API functions exposed to higer layers.

4.1 Used technologies

The CAS has been built with use of Ruby on Rails (RoR) [1] technology. It has been chosen among many others because it best meets the requirements of fast and easy data driven system implementation in REST style as it cooperates with many databases and supports REST interface natively. It is an open-source web framework based on MVC (Model-View-Controller) architecture so implementation is very flexible and well supported by RoR community. RoR application needs a web server to be run. Among many available web servers, the most popular in case of RoR are: Lighttpd, Apache, Cherokee and WEBrick - a simple web server distributed with Ruby. The last one has been chosen to be used in CAS module because it seemed sufficient for this purpose. The designed data model has been implemented as a relational database. The DBMS (Database Management System) has been chosen to be MySQL - strongly supported by Rails and typically used in this case.

4.2 Internal architecture

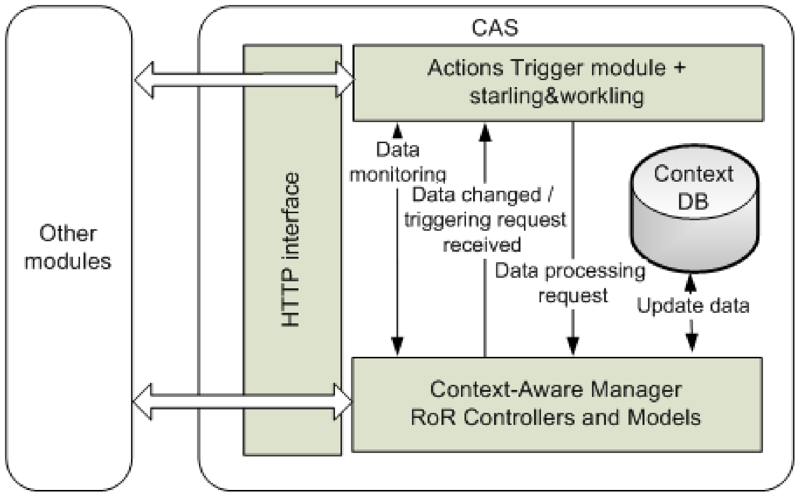

CAS
Other modules
HTTP interface
Actions Trigger module + starling&workling
Data monitoring
Data changed / triggering request received
Data processing request
Context DB
Update data
Context-Aware Manager RoR Controllers and Models

Figure 4: CAS internal architecture

The internal software architecture of CAS is presented in Figure 4 and it consists of:

- HTTP Interface is a functionality provided by WEBrick server and HttpClient Ruby Gem, which enables HTTP requests creation, sending them and receiving.
- Context-Aware Manager (CAM) is implemented by Controller and Model blocks of RoR project. It collects context data via HTTP interface, processes it and stores it in Context DB. It is also responsible for collecting HTTP requests from other modules (Recommendation Module, Mobility Module, etc.), preparing and returning the response.
- Context DB stores gathered data in MySQL database. The names of particular tables correspond with the names of Models, Views and Controllers (according to RoR project guidelines).
- Action Trigger is responsible for monitoring data in Context DB and, sending periodically notifications or occasionally requests to other modules, depending on such aspects as the time, data changed or the received request.

4.3 Interface modes

CAS interfaces have to support different access modes, to meet the requirements of contextual applications that may need to retrieve contextual data at a given moment, or continously over service lifetime. Thus, HTTP interfaces of CAS may work in push or pull mode. In pull mode (request/response), which is presented in Figure 5a) the external module queries CAS for context information and CAS immediately prepares and sends back the answer (after verifying the user's policy settings in PS module). In push mode (subscribe/notify) presented in Figure 5b) the external module subscribes to receive notifications from CAS containing context information. In this case, Action Trigger module decides when CAS sends the notifications to the subscribed module. Before sending the notification CAS needs to ask PS module for user's policy.

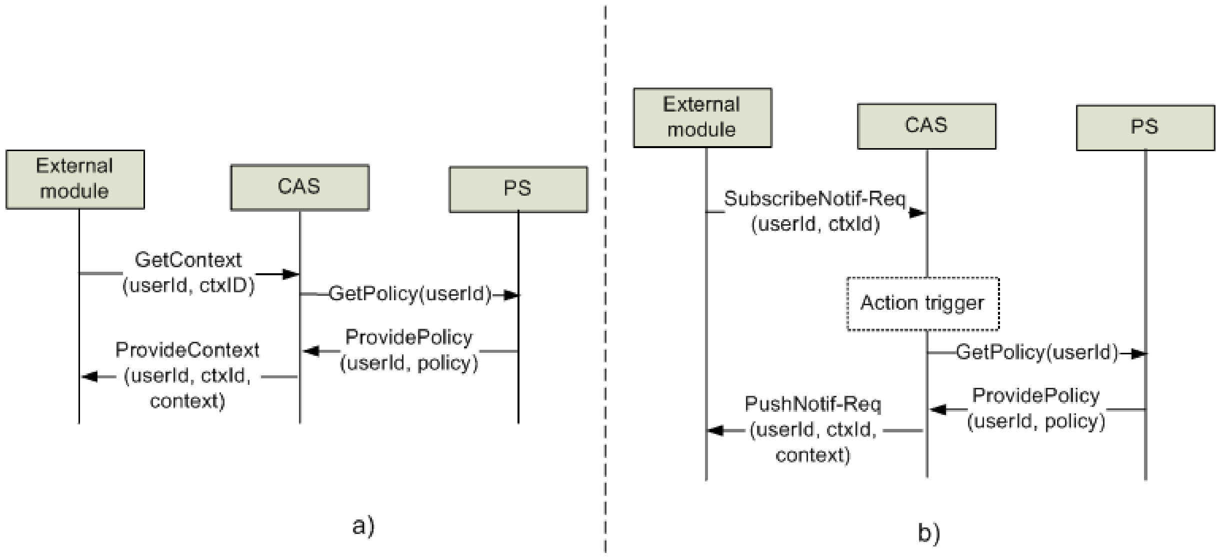

External module
CAS
PS
GetContext (userId, ctxID)
GetPolicy(userId)
ProvidePolicy (userId, policy)
ProvideContext (userId, ctxId, context)
a)
External module
CAS
PS
SubscribeNotif-Req (userId, ctxId)
Action trigger
GetPolicy(userId)
ProvidePolicy (userId, policy)
PushNotif-Req (userId, ctxId, context)
b)

Figure 5: Interaction modes over CAS context providing interface: a) pull mode and b) push mode

In some cases of both presented modes, CAS needs to work in an asynchronous mode - to behave as an HTTP client. The HTTPClient gem of Ruby is used to prepare and send messages. There are also two additional modules: Starling and Workling which are engaged in case of sending notifications by CAS to other modules. Starling is installed as a Ruby gem and Workling as a plugin. Starling is a message queue while Workling provides an API which enables running code in the background. Thanks to these two modules, notifications can be sent in a proper order and in the background, not blocking CAS behavior.

4.4 Data model implementation

CAS is a system responsible for acquiring, storing and providing data. Data modeling and it's implementation is therefore a crucial aspect for the system. It should be well adjusted to the system requirements to enable efficient data processing and make the system work effectively. It should enable work on large amounts of data coming from different sensors, having different nature: static, dynamic, inferred. As other modules are responsible for advanced logic of data processing, CAS is not obliged to provide this functionality - it still should enable data processing but not very complex, nor complicated. There were two main candidates that meet these requirements: Ontology and Relational Database. Ontology is a semantic approach which describes the knowledge about the world: abstract beings, particular objects, their properties and relations between them. According to Tom Gruber: "An ontology is a specifications of conceptualization". Some works have been made to examine Ontology as an underlying data model.

The model created with this approach (based on OWL format, made e.g. with use of Protege tool) can be easily extended to new functionalities if needed. Large sets of context information can be stored with use of Triple Stores (e.g. MySQL). Data processing is effective and easy (even very complicated operations) thanks to built-in reasoning. Despite all presented advantages, some problems related to implementation disabled continuation of work with use of ontology approach. Chosen technology (Ruby on Rails) and semantic library (ActiveRDF) have proven not to be ready to support this approach due to many errors in the library and lack of active maintenance. The other important candidate for CAS data model and finally chosen for further work was Relational Database approach. Relational Databases are well adapted for large data set storage. There is a wide choice of RDBMS implementations available (e.g. MySQL DBMS has a support in such technologies as Ruby on Rails, Java, etc.). They do not provide any special tool for data processing like Reasoners in case of Ontologies but SQL rules are sufficient to build basic rules as needed in this case. Data model created with this approach is not so easily extendible as in case of Ontology, but it is not mandatory as in case of UP-TO-US system the scope of modeled world is well defined. All these features make Relational Database a perfect choice for CAS data model.

4.5 Context-awareness API

Access to CAS data by other modules is realized through context-awareness API exposed by context providing interface. The design of API functions had to reflect all needs of higher layer modules, which realize the logic of contextualized service. Figure 6 depicts the interfaces between CAS and particular modules. All of them use HTTP/REST protocol with XML content. The XML content (except from Ca interface) is a proprietary extension of TV Anytime Standard.

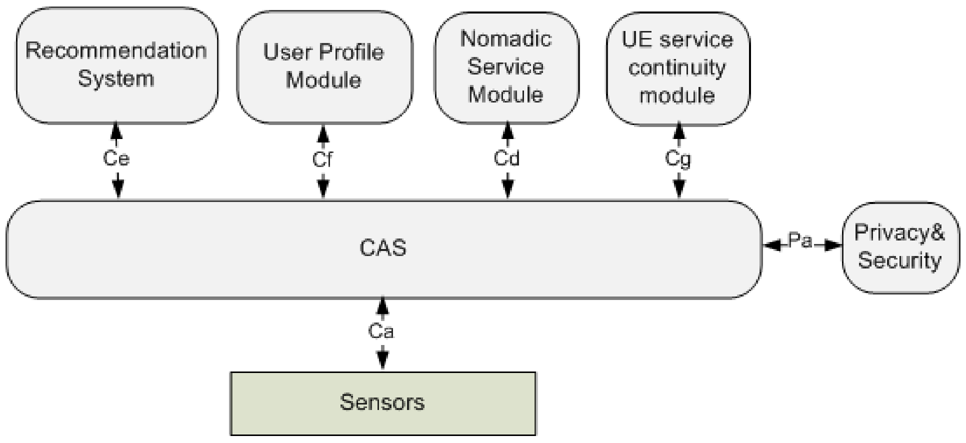


Figure 6: CAS interfaces with other modules

Ca interface is used to send context information from sensors to CAS. No standard messages, neither their content have been defined for this interface. The types of exchanged messages depend on used IPTV platform and it's elements that provide context information to CAS.

Ce interface enables communication between CAS and Recommendation System. RecSys role is to prepare a content recommendation that best fits user's static preferences and his current situation. It needs to know this current context so it asks CAS with HTTP request:

```
HTTP GET //CAS_IP:port/users/userId/context_status HTTP/1.1
```

Parameter of this request is *userId* which is the identifier of the user. In response, CAS sends HTTP 200 OK message with user's context as a Body (in XML format) with such information as: current user's location, currently used terminals, conditions of the networks that these terminals are attached to, etc.

Pa interface enables communication between CAS and Privacy and Security Module. When CAS receives a request from an external module (RecSys, Mobility Module, etc.), it needs to verify first if the user agreed to share context information with these modules. For this purpose, CAS sends a request to PS module:

```
HTTP GET //PS_IP:port/uptous/UPM/Cf/UserProfile-Req/userId HTTP/1.1}
```

Parameter of this request is *userId* which is the identifier of the user. In response, CAS expects an HTTP 200 OK message with user's policy description as a Body (in XML format). The most important elements of this policy for CAS are: *UsageEnvironment* (only when it's attribute *protected* is set to *false*, CAS is allowed to share any context information with external modules) and *UsageHistory* (only when it's attribute *allowCollection* is set to *true*, CAS is allowed to store, process and share user's actions history with other modules).

Cf interface enables communication between CAS and User Profile Manager. The following messages are exchanged on this interface:

- When CAS receives a message from one of the sensors concerning a user that is not known yet in the system, it needs to check in the UPM (where users' profiles are stored) if the user with given userId exists and to download his static profile. For this purpose, CAS sends a request to UPM module:

  ```
  HTTP GET //UPM_IP:port/uptous/UPM/Cf/UserProfile-Req/userId HTTP/1.1
  ```

  Parameter of this request is *userId* which is the identifier of the user. In response, CAS expects an HTTP 200 OK message with user's profile as a Body (in XML format) with such information as: possibly used terminals, possibly used networks, possibly visited places, groups belongings, disabilities, mobility preferences etc.

- As UPM is responsible for building dynamic user's profile based on user's actions, it should be notified by CAS about user's activities. For this purpose, CAS sends periodically (frequency can be set) notifications to UPM about user's actions:

  ```
  HTTP PUT //UPM_IP:port/UpToUs/UPM/Cf/PUH/userId HTTP/1.1
  ```

  Parameter of this request is *userId* which is the identifier of the user. The Body of each message contains XML structure with information about user's actions (where an action can be for example a fact of watching certain TV program) such as: action type, action start time and duration, type of the terminal on which the action has been performed, conditions of the network that the terminal has been attached to in the moment of the action, program and services to which this action concerns, etc.

- When UPM gets a notification from CAS about user's action concerning program or service that it does not know yet, it sends to CAS a request about additional information concerning this program (as program identifier is not sufficient to build extended dynamic user profile, description of the program is needed to build user's preferences):

```
HTTP GET //CAS_IP:port/programs_details/crid(serviceId) HTTP/1.1
```

Parameter of this request is *crid*, which is the identifier of the program in standardized format - Content Reference Identifier or *serviceId*, which is the identifier of the service (e.g. TV channel name). In response, CAS sends HTTP 200 OK message with program's information as a Body (in XML format) such as: program's titles, synopses, credits, keywords, genre etc.

Cg interface enables communication between CAS and UE Service Continuity Module as the UP-TO-US project supports the mobility service (for terminals and sessions) according to user's preferences and his context status. The following messages are exchanged on this interface:

- The mobility module needs to know the mobility preferences of the user before performing any mobility action. It sends a request to CAS:

  ```
  HTTP GET //CAS_IP:port/users/userId/mobility_preferences HTTP/1.1
  ```

  Parameter of this request is *userId* which is the identifier of the user. In response, CAS sends HTTP 200 OK message with information about user's mobility preferences as a Body (in XML format) such as: preferred access networks list (in order of preference) and preferred mobility mode: manual, semi-automatic or automatic. Mobility preferences should be previously acquired from UPM with user's profile on Cf interface.

- To perform a terminal mobility service, the mobility module needs to know (beside the user's mobility preferences) possible active devices where the user's multimedia session can be transferred. For this purpose, it sends a request to CAS:

  ```
  HTTP GET //CAS_IP:port/users/userId/devices_info HTTP/1.1
  ```

  Parameter of this request is *userId* which is the identifier of the user. In response, CAS sends HTTP 200 OK message with information about user's devices (these devices are attached to the network and user is registered on them) as a Body (in XML format) such as: device's type, device's location, access networks that the device is attached to, etc.

- To perform a session mobility service, the mobility module needs to know (beside the user's mobility preferences) all multimedia sessions started on particular terminal when it was attached to particular access network. All these multimedia sessions should be continued on the same terminals but in another access network. For this purpose, it sends a request to CAS:

  ```
  HTTP GET //CAS_IP:port/users/userId/sessions_info
  ```

  Parameter of this request is *userId* which is the identifier of the user. In response, CAS sends HTTP 200 OK message with information about user's multimedia sessions as a Body (in XML format) such as: identifier of the terminal where the multimedia session has been activated, multimedia session identifier, multimedia session description, program's identifier that this sessions concerns, etc.

Cd interface enables communication between CAS and Nomadic Service Module as the UP-TO-US project supports the service of nomadic access to user's IPTV platform (e.g. from a friend's place or a hotel - a visited domain) together with recommendation service according to user's preferences and settings saved in his home domain. CAS takes active part in the nomadic service provisioning. Remark that all modules (CAS, NSM and RecSys) have two instances: one in Home Domain and the second one in Visited Domain. As the context status of the user staying in Visited Domain is saved in the Visited CAS but content recommendation is generated by Home RecSys, Visited NSM needs to get the context status of the user from Visited CAS and send it to the Home NSM and further to the Home CAS. For this purpose the following messages are sent:

- Visited NSM asks Visited CAS for current user context:

  ```
  HTTP GET //CAS_IP:port/users/userId/context_status HTTP/1.1
  ```

  Parameter of this request is *userId* which is the identifier of the user. In response, CAS sends HTTP 200 OK message with user's context as a Body (in XML format) with such information as: current user's location, currently used terminals, conditions of the networks that these terminals are attached to, etc.

- When Home NSM gets the user's context status from Visited NSM, it updates Home CAS with acquired information concerning user's context:

  ```
  HTTP PUT //CAS_IP:port/users/userId/nomadic_context HTTP/1.1
  ```

  Parameter of this request is *userId* which is the identifier of the user. The context status of nomadic user (all information acquired previously from Visited NSM) is sent in the Body of the request (in XML format). When Home CAS saves the context status of the nomadic user, the Home RecSys is able to get it and generate a content recommendation for this user.

4.6 Personalized IPTV service scenario

This section presents an example scenario for IPTV service enhanced with contextual information. Figure 7 explains how the context data is gathered, processed and exchanged within the architecture described in section 3. The scenario corresponds to the use-case where IPTV system produces a personalized content recommendation for the user, based on his dynamically updated profile and current context. This use-case is a reference scenario for tests of UP-TO-US architecture in testbed environement.

When user turns on the STB, it connects to the IPTV platform through a platform-depended process (1). As a pre-requisite, a user has to identify himself in the system using platform specific mechanism (e.g. by presenting personal RFID badge to the tag reader embedded in the STB, or other user identification method). After performing the authentication procedure, STB sends a context update message to CAS (2), using push method of CAS context acquisition interface. The context update includes user identifier, device identifier, as well as location data (in the case of STB the location is fixed and may correspond to the room where it is installed - living room, bedroom, kitchen,...) and some network-level information (e.g. estimated QoS level on the path between STB and media server). Then, CAS requests UPM (3) for additional information about the profile of the user. Before answering, UPM verifies (4) in the PS module if the user's privacy policy settings allow for sharing profile data with CAS. If yes, it sends a response to CAS (5) which creates a context status entry for given user and updates it with current dynamic context data obtained from STB in step (2).

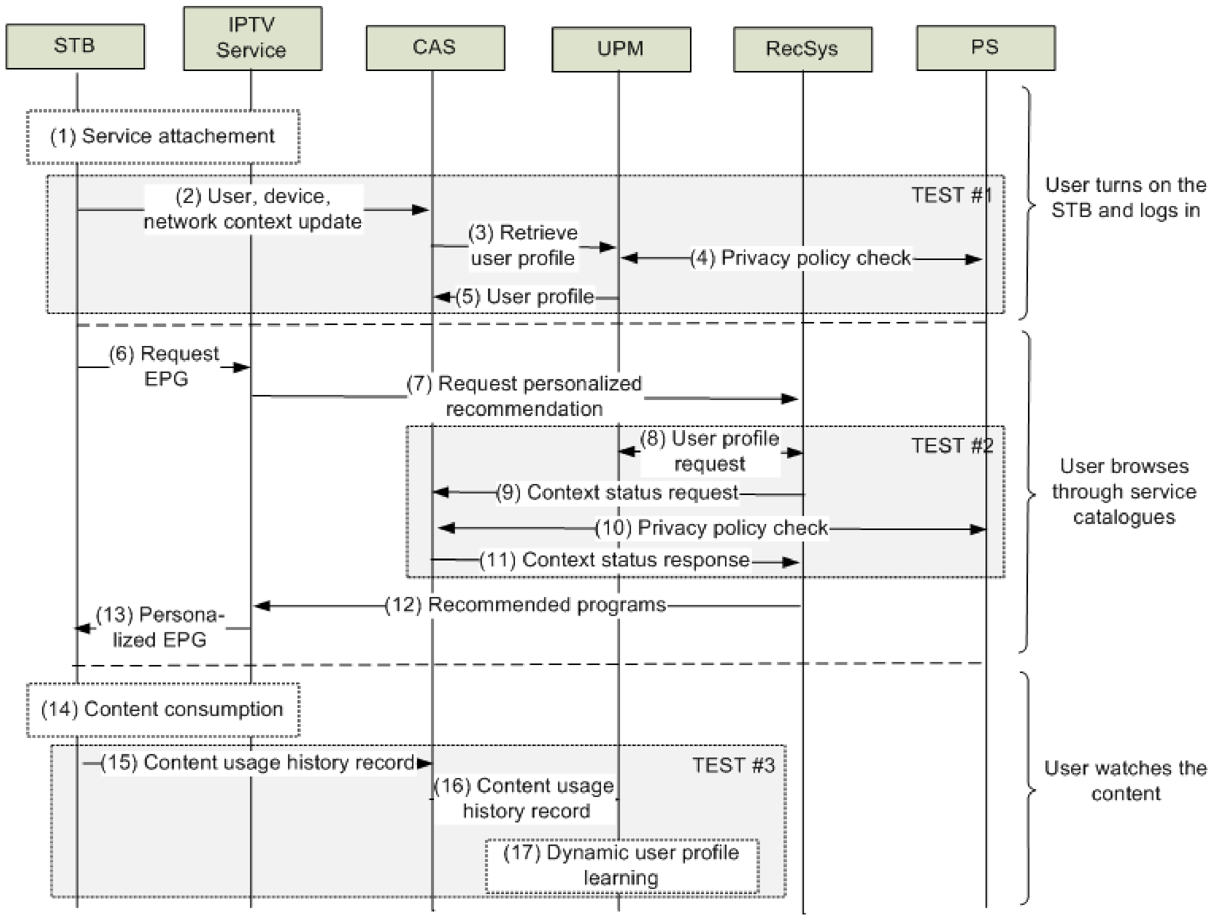

STB
IPTV Service
CAS
UPM
RecSys
PS
(1) Service attachement
TEST #1
(2) User, device, network context update
(3) Retrieve user profile
(4) Privacy policy check
(5) User profile
User turns on the STB and logs in
(6) Request EPG
(7) Request personalized recommendation
TEST #2
(8) User profile request
(9) Context status request
(10) Privacy policy check
(11) Context status response
(12) Recommended programs
(13) Personalized EPG
User browses through service catalogues
(14) Content consumption
TEST #3
(15) Content usage history record
(16) Content usage history record
(17) Dynamic user profile learning
User watches the content

Figure 7: Personalized IPTV call-flow scenario. Shaded blocks correspond to execution of CAS tests, as explained in detail inside the article

User browses the service GUI and requests displaying EPG (Electronic Program Guide) personalized for him. The EPG is requested by STB through communication with IPTV platform (6). However, before providing the answer to STB, the IPTV platform communicates with UP-TO-US RecSys, asking it for preparing a selection of programs recommended for given user (7). The RecSys prepares a recommendation taking into account list of programs available in EPG, user profile information (preferred program types etc.) obtained from UPM in step (8), and current context status for which it asks CAS in step (9). After positive verification of user's privacy policy (10) CAS provides current status to RecSys (11). RecSys sends the recommendation to IPTV service platform (12), which on the other hand communicates it to STB (13).

Now, user may select a program to watch among the recommended list (14). After the content session has finished (the program has ended or the user has abandoned it), STB sends a content usage history record to CAS (15). The record contains a reference to unique program identifier and a summary description of situation (context) in which the content has been consumed (e.g. location and type of a device). After forwarding it to UPM (16), this kind of information is used by advanced preference learning algorithm (17). As a result of this process, the user's profile is dynamically updated with his actual preferences concerning accessed content and environmental situation (e.g. the system may learn that a user usually likes to watch sport programs in the evening in his bedroom).

5 Validation by testbed experiments

5.1 Testbed description

The UP-TO-US prototype described in section 4 has been deployed in testbed located in France Telecom laboratories in Paris. A series of tests have been conducted to verify compliance of implementation with architecture specification and to proof efficiency of proposed solution for supporting IPTV with context-aware information. The testbed schema is outlined in figure 8. As this paper focuses on context-aware subsystem and its role in providing personalized service, only a subset of UP-TO-US prototype is depicted here: CAS, UPM, RecSys and PS (see section 3 for reminder of module's roles in the system). A software-based STB has been enhanced with additional functions for accessing UP-TO-US features. For compliancy with previously deployed components, an IMS-based IPTV platform has been used (see [15]), consisting of IMS Core, SSF (Service Selection Function) and MF (Media Function) nodes. However, remark that UP-TO-US architecture supports any IPTV platform technology through open HTTP-based interfaces. IMS-based solution used for the tests is just one of possible deployment scenarios and it has no impact on presented test results and messages examples.

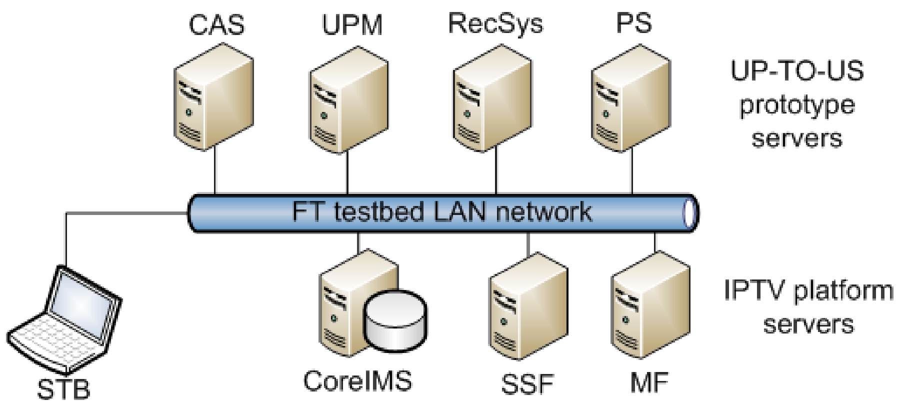


Figure 8: UP-TO-US prototype installation

5.2 Test results

The test scenario corresponds to the use-case described in section 4. A home scenario is emulated, with user Bob (`bob@open-ims.test`) accessing his personalized IPTV service through STB located in his bedroom. After he logs-in, a personalized content recommendation is displayed and Bob starts watching one of the recommended TV channels (”TF1”).

TEST 1: update of user, device and network context by STB. The goal of this test was to verify the behavior of STB and CAS in the initial phase, when user logs in the system and his context record is created in CAS database for the first time. The test execution has been initiated by placing the RFID tag associated with user profile `bob@open-ims.test` (previously created in UPM database) close to the STB. This action has triggered the following functions of the STB. First, a packet-level measurement process, using popular IPerf tool, has been started to assess current network conditions on the path from the user device to media function node, from which the content will be accessed. Then, STB has send an update message to CAS over its Ca interface (message (2) on figure 7), informing about status update of user `bob@open-ims.test`. The contents of this message, as captured during test execution with Wireshark network analyzer, are presented in figure 9. The user identifier, STB device and its type

```
PUT /users/bob@open-ims.test/stb_login HTTP/1.1
Content-Type: text/xml
<?xml version='1.0'?>
<context>
  <local_time>2012-10-24 09:33:59</local_time>
  <device>
    <device_serial>103415</device_serial>
    <type_of_device>Set-Top Box</type_of_device>
    <type_of_network_interface>FixMedium</type_of_network_interface>
    <ip_address>172.20.197.134</ip_address>
  </device>
  <network>
    <bandwidth>10</bandwidth>
    <jitter>0</jitter>
    <latency>0</latency>
    <datagram_loss>0</datagram_loss>
    <measurement_start_time>2012-10-24 09:33:59</measurement_start_time>
    <measurement_duration>0:0:21</measurement_duration>
  </network>
</context>
```

Figure 9: Test results: contents of context status update, captured by Wireshark on the interface between STB and CAS

identifier, access network type, IP address, and measured network conditions are included in XML body of the message. After the sequence of interactions not reported here (messages 3 to 5 in figure 7) the CAS database was filled with a current snapshot of user situation (device and network status) as well as details of user profile retrieved from UPM. This concludes TEST 1 with a positive result.

TEST 2: user status query. After skipping a series of message exchanges, which do not involve CAS and thus are not reported here, TEST 2 aimed to verify the key functionality of CAS that is handling the query for current context situation of a user. Execution of this test has been initiated by the RecSys module, which has been asked to prepare a personalized and contextualized program recommendation for user `bob@open-ims.test`. After exchange with UPM for retrieving relevant user profile parameters (see message (8) in figure 7) RecSys has sent a request (9) to CAS over Ce interface, to query for context status information of user `bob`. CAS has contacted the PS module (10) to check user's privacy settings and the result has been positive i.e. the user allows for sharing his status with TV recommendation system. Now CAS may return the answer (11) to RecSys. The contents of this message, captured with Wireshark tool during the test execution, are presented in figure 11. Correct values of user, device and network status parameters, as injected in CAS during the previous test, are observed. The STB displays recommendation of TV channels for Bob (see 10). Concluding, the test result is positive.

TEST 3: usage history update. After content consumption by the user, a usage history record is processed by CAS and sent to UPM module, where it will be used as input parameter for user preference learning algorithm. Test 3, which was aimed to verify this process, has been initiated by STB, where the test user has accessed and watched some content item (see message block (14) in figure 7). On finish of the video session, a usage trace record (15) has been created by STB and sent to CAS over Ca interface. CAS aggregates usage history records received over a period of time (e.g. one day) and sends a collective record to UPM. For the purpose of test, just a single trace has been sent to UPM in

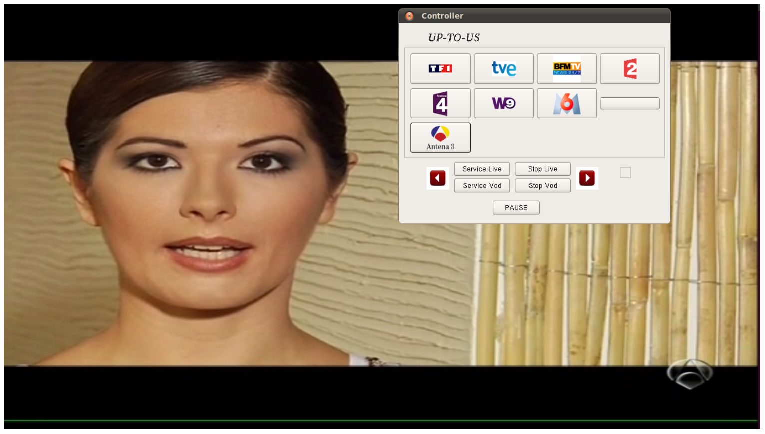


Figure 10: Personalized content recommendation for user Bob

```
HTTP/1.1 200 OK
<?xml version="1.0"?>
<up2us:ContextStatusInformation xmlns:mpeg7="urn:tva:mpeg7:2008"
xmlns:up2us="urn:tva:up2us:2012" xmlns:tva="urn:tva:metadata:2011"
xmlns:tva2="urn:tva:metadata:extended:2011"
xmlns:mpeg21="urn:tva:mpeg21:2011"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:tva:up2us:2012 Up2us.xsd">
  <up2us:UserIdentifier protected="false">
    <mpeg7:Name>bob@open-ims.test</mpeg7:Name>
  </up2us:UserIdentifier>
  <up2us:UserLocation>
    <up2us:Name>Bedroom</up2us:Name>
    <up2us:NameTerm href="urn:tva:metadata:extended:cs:PlaceTypeCS:2011:1.2">
      <mpeg7:Name>Bedroom</mpeg7:Name>
    </up2us:NameTerm>
  </up2us:UserLocation>
  <up2us:TerminalInformation terminalInformationId="103415">
    <tva2:TerminalType href="urn:tva:metadata:extended:cs:TerminalTypeCS:2011:SET-TOP BOX">
      <tva:Name>Set-Top Box</tva:Name>
    </tva2:TerminalType>
    <up2us:NetworkInformation>
      <up2us:NetworkCapability maxCapacity="8000" minGuaranteed="800"/>
      <up2us:NetworkCondition duration="00:00:21" startTime="2012-10-24 09:33:59 UTC">
        <mpeg21:AvailableBandwidth average="10.0" minimum="" maximum=""/>
        <mpeg21:Delay packetTwoWay="" packetOneWay="0.0" delayVariation="0.0"/>
        <mpeg21:Error packetLossRate="0.0" bitErrorRate=""/>
      </up2us:NetworkCondition>
    </up2us:NetworkInformation>
  </up2us:TerminalInformation>
</up2us:ContextStatusInformation>
```

Figure 11: Test results: contents of context status answer, captured by Wireshark on the interface between CAS and UPM

message (16), which contents, as captured by Wireshark tool, are presented in figure 12. The usage record contains user identifier, content identifier (composed by unique program identifier and a reference to service, e.g. TV channel, over which it has been retrieved). In addition, information about the total duration of the content and total time it has been watched by the user is also provided (these times may differ if user has voluntarily skipped watching the content before it has ended). The contents of captured messages match the actual parameters of the test, so as a conclusion the result is positive.

```
PUT /UpToUs/UPM/Cf/PUH/bob@open-ims.test HTTP/1.1
<?xml version="1.0"?>
<up2us:UsageHistory xmlns:mpeg7="urn:tva:mpeg7:2008"
xmlns:up2us="urn:tva:up2us:2012" xmlns:tva="urn:tva:metadata:2011"
xmlns:tva2="urn:tva:metadata:extended:2011"
xmlns:mpeg21="urn:tva:mpeg21:2011" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:tva:up2us:2012 Up2us.xsd">
  <up2us:UserIdentifier protected="false">
    <mpeg7:Name>bob@open-ims.test</mpeg7:Name>
  </up2us:UserIdentifier>
  <up2us:UserActionHistory>
    <up2us:ObservationPeriod>
      <mpeg7:TimePoint>2012-10-24 00:00:00 UTC</mpeg7:TimePoint>
      <mpeg7:Duration>23:59:59</mpeg7:Duration>
    </up2us:ObservationPeriod>
    <up2us:UserActionList numOfInstances="1" totalDuration="01:30:00">
      <up2us:ActionType href="urn:tva:metadata:cs:ActionTypeCS:2010:view"/>
      <up2us:UserAction>
        <mpeg7:ActionTime>
          <mpeg7:MediaTime>
            <mpeg7:MediaTimePoint>2012-10-24 10:30:12 UTC</mpeg7:MediaTimePoint>
            <mpeg7:MediaDuration>01:20:00</mpeg7:MediaDuration>
          </mpeg7:MediaTime>
          <mpeg7:GeneralTime>
            <mpeg7:TimePoint>2012-10-24 10:30:12 UTC</mpeg7:TimePoint>
            <mpeg7:Duration>01:30:00</mpeg7:Duration>
          </mpeg7:GeneralTime>
        </mpeg7:ActionTime>
        <mpeg7:ProgramIdentifier>1</mpeg7:ProgramIdentifier>
        <up2us:ServiceName length="short">TF1</up2us:ServiceName>
      </up2us:UserAction>
    </up2us:UserActionList>
  </up2us:UserActionHistory>
</up2us:UsageHistory>
```

Figure 12: Test results: contents of usage trace update, captured by Wireshark on the interface between CAS and UPM

More extensive tests have been conducted to verify operation of all modules of proposed IPTV services personalization architecture. However, as this paper focuses specifically on context-awareness architecture, just a relevant subset of tests has been reported and for the remaining results (e.g. related with preparation of dynamic user profiles, recommendation algorithms, mobility and nomadic access procedures) the reader is invited to consult the project webpage in reference [3]. Summarizing, the outcome of the tests positively verifies UP-TO-US CAS as an architecture for supporting IPTV systems with context-awareness capabilities.

6 Conclusion

This paper has discussed IPTV services personalization through applying context awareness. Context awareness principle enhances traditional IPTV systems through providing smart and advanced services with appropriate content recommendation and adaptation and hence allowing services interactivity and personalization. This paper has introduced a novel architecture for context-aware IPTV, aiming to provide enhanced services personalization. The proposed model has been implemented as a proof of concept and tested in a series of experiments in controlled testbed environment. The results of tests related with context-awareness architecture are reported in this paper for illustration of context-aware capabilities supporting personalization of IPTV service catalogue. The next steps of this work are to continue testbed experiments aiming to prove the effectiveness of complete UP-TO-US architecture, with interactions between all modules for end-to-end service scenarios. Acceptance tests with real users are also planned to validate expected usage impact and adoption of personalized IPTV services on the European markets.

Acknowledgements. This work has been supported by the Eureka Celtic UP-TO-US (User-Centric Personalized IPTV UbiquitOus and SecUre Services) European project.

References

[1] Ruby on Rails. Ruby on Rails website `http://rubyonrails.org`.

[2] TV Anytime. TV Anytime website `http://tech.ebu.ch/tvanytime`.

[3] UP-TO-US: User-Centric Personalized IPTV UbiquitOus and SecUre Services. Project website `http://up-to-us.rd.francetelecom.com`.

[4] T. A. and S. V. Gopalan S. Context aware personalized ad insertion in an interactive tv environment. In *Proc. of the 4th Workshop on Personalization in Future TV (TV'04), Eindhoven, The Netherlands*, August 2004.

[5] P. Brown, J. Bovey, and X. Chen. Context-aware applications: from the laboratory to the marketplace. *IEEE Personal Communications*, 4:58–64, 1997.

[6] G. Chen and D. Kotz. A survey of context-aware mobile computing research. Technical Report TR2000-381, Dept. of Computer Science, Dartmouth College, 2000.

[7] S. da Silva, F., L. Alves, and G. Bressan. PersonalTVware: A Proposal of Architecture to Support the Context-aware Personalized Recommendation of TV Programs. In *Proc. of the 7th European Conference on Interactive TV and Video (EuroITV'09), Leuven, Belgium*. ACM Press, June 2009.

[8] M. Dabrowski, J. Gromada, and H.Moustafa. Context-awareness for IPTV Services Personalization. In *Proc. of the 6th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS'12), Palermo, Italy*, pages 37–44. IEEE, July 2012.

[9] V. Damjanovic, G. Geuntner, S. Schaffert, T. Kurz, and D. Glachs. Semantic Social TV. In *Proc. of 2011 NEM Summit, Implementing Future Media Internet, Torino, Italy*, September 2011.

[10] R. Gregory-Clarke and C. Newell. TV Recommender System Field Trial Using Dynamic Collaborative Filtering. In *Adjunct Proc. of the 10th European Interactive TV Conference (EuroITV'12), Berlin, Germany*, July 2012.

[11] S. J.B.D., G. R., F. G.B., and M. E.D.S. Modeling of user interaction in context-aware interactive television application on distributed environments. In *Proc. of the 1st Workshop on Personalization in Future TV (TV'01), Sonthofen, Germany*, July 2001.

[12] A. Moon, H. Kim, K. Lee, and H. Kim. Designing CAMUS based context-awareness for pervasive home environments. In *Proc. of International Conference on Hybrid Information Technology (ICHIT'06), Jeju Island, Korea, LNCS*, volume 4413, pages 666–672. Springer-Verlag, November 2006.

[13] B. N. Schilit and M. Theimer. Disseminating active map information to mobile hosts. *IEEE Network*, 8:22–32, 1994.

[14] S. Song, H. Moustafa, and H. Afifi. Context-Aware IPTV. In *Proc. of the 12th IFIP/IEEE International Conference on Management of Multimedia and Mobile Networks and Services: Wired-Wireless Multimedia Networks and Services Management (MMNS'09), Venice, Italy, LNCS*, volume 5842, pages 189–194. Springer-Verlag, October 2009.

[15] S. Song, H. Moustafa, and H. Afifi. Personalized TV Service through Employing Context-Awareness in IPTV/IMS Architecture. In *Proc. of the 3rd International Conference on Future Multimedia Networking (FMN'10), Krakow, Poland, LNCS*, volume 6157, pages 75–86. Springer-Verlag, June 2010.

[16] M. Weiser. The computer for the 21st century. *Scientific American, 265(3):66-75*, 266(3):66–75, 1991.

Marek Dabrowski joined Orange Labs Poland in 2000. He received Ph.D. degree in Telecommunications from Warsaw University of Technology, Institute of Telecommunications in 2005. His research interests cover social and interactive TV services, context-awareness and personalization, IPTV and WebTV content distribution, network QoS aspects. He participated in EU funded research projects like "AQUILA", "EuQoS", "MoMe", as well as recently in EUREKA/CELTIC project "CELTIC UP-TO-US" were he served as WP leader. In Orange Labs he managed internal projects related with anticipation and development of new services in the area of digital content distribution. He has publications at national and international conferences, and co-chaired a workshop during major international event.

Justyna Gromada is an R&D engineer at Orange Labs Poland since 2008. She works in Content Services and Technologies Section. Justyna graduated from Warsaw University of Technology in 2009 with M.Sc. degree in Telecommunications - Master Thesis concerned the IMS platform. Data modeling (semantic approach, relational databases, graphs) and services development (Java, Ruby on Rails) are her main fields of interest and expertise.

Hassnaa Mustafa is a research scientist at Intel Labs, Oregon, USA since October 2012. She was an R&D engineer at France Telecom (Orange Labs), France during the period January 2005 – September 2012. She obtained her Tenure in Computer Science in June 2010 from the University of Paris XI, her PhD in Computer and Networks from Telecom ParisTech in December 2004 and her Master degree in distributed systems in September 2001 from the University of Paris XI. Her research interests include media networks, video delivery optimization, QoE, services personalization and content adaptation. She worked for many years on ad hoc and vehicular networks routing and AAA and is regular IETF participant and IEEE member. She has over 70 publications in international conferences and journals including IEEE and ACM and she is a co-author of two books published by the CRC press. She was a guest editor for several journals and is active member in TPCs for IEEE, ACM and IFIP conferences since many years. She also co-chaired several workshops and conferences symposiums.

Jacky Forestieris an inventor and a research engineer in France Telecom R&D (Orange Labs) since 1996. He is graduated on Internet Network and Services (2001) by CESI Cachan (France). From 2002 till 2006, he was also owner and CEO of a SME specialized on research in workflow process for administration and insurance. He has several patents and one of them is now referenced by Comverse and IBM (USA). Jacky has published 3 papers in international conferences.