Uluru : A platform for adaptive mobile multimedia applications

Andrew Tokmakoff¹, Henk Eertink¹, Johan de Heer¹, John Anijs², and Jaap Reitsma¹

¹Telematica Instituut, P.O. Box 589, 7500 AN Enschede, The Netherlands
²MobiView, P.O. Box 10, Postvak TA13, 1200 JB Hilversum, The Netherlands

[Andrew.Tokmakoff;Henk.Eertink;Johan.deHeer;Jaap.Reitsma]@telin.nl
John.Anijs@mobiview.nl

0-7803-8603-5/04/$20.00 ©2004 IEEE.

Abstract

Uluru is a software platform that provides a number of enabling services to mobile multimedia applications including content-personalisation and charging. This paper describes the design of such an enabling platform and explores its applicability from a business perspective. We describe the platform’s functionalities and overall architecture, and discuss Callsong, an application that makes extensive use of Uluru’s personalization and charging services.

1. Introduction

Uluru is an experimental platform for the delivery of audio and video interactive media via mobile packet-switched networks (such as GPRS, UMTS and CDMA2000). It forms the “missing link” between the audio-visual domains of content facilitating companies (i.e. “Media Gateways” with multi-channel distribution) and mobile networks operators. Uluru has tackled a number of research issues, but within this paper we will limit our scope to Uluru’s handling of personalization and charging.

This article’s structure is as follows: Firstly we provide an overview of the platform’s functionalities and architecture. After this we discuss a number of field trials that demonstrate Uluru’s use in practice, and propose extensions to the platform.

2. Content-rich mobile applications

From a mobile network operator’s perspective, the success of broadband mobile services is highly dependent on the ready availability and acceptance of attractive applications and content. Applications that end-users are willing to pay for must be made available to enable telecom operators to recoup their significant investment in 2.5 and 3G telecommunications system infrastructure.

Uluru acts as a link between content providers and access network providers. This effort has brought together players from two market segments: Telecom and Media, and produced a platform that is positioned for use in a business-to-business (B2B) setting.

For certain services, a mobile operator may wish to emphasize its unique character by offering exclusive access to services, whereas content providers have a strong interest in making their content available for as broad an audience as possible, irrespective of the mobile operator that individuals are associated with. This introduces a need for operator-independent charging.

In this situation, a platform “sits in the middle” between content providers and mobile operators. Such a platform is likely to be operated by a “facilitator”, who plays a central role in the content value chain.

3. Platform Functionalities and Architecture

Mobile applications often share a number of common functionalities such as payment, personalization, location-awareness and content management. Uluru’s common functions (Services) are provided by the platform to its hosted applications, since it is much more convenient for application developers to leverage common services, rather than being required to “re-invent” them for each specific mobile application. Uluru is a platform that is capable of hosting 3rd party application logic and making common services available to these applications. Uluru provides a set of coarse-grained Services to the applications that it hosts. These functions are made available to Applications via the Uluru Service API.

Figure 1 illustrates how Uluru Services are made available to applications via the Uluru Service API and also how adapters are used to interface Services with their associated back-end Commercial-Off-The-Shelf (COTS) components.
Uluru makes use of the Façade Pattern \[1\] to encapsulate the complexities of the Uluru Service API and make the Services coarse-grained. One of the platform’s design goals was to facilitate a “mix and match” approach to back-end components, whilst minimizing the engineering impact of such back-end component changes.

By using Adapters, it is possible to easily replace back-end COTS applications with different solutions that provide the same desired functionality. However, this process requires that a new COTS-application-specific adapter be developed, whose function is to map the COTS component’s interface to the well-defined Uluru service API. This approach decouples the COTS systems from the platform, avoids vendor lock-in and insulates the platform from the underlying technologies. A similar approach has been successfully used in Chamois \[2\], where a number of research and COTS products were integrated into a unified Knowledge Management system.

### 4. Interaction between applications and Uluru

In order to access Uluru Platform functions, applications are required to instantiate an UluruFactory, which they subsequently use to authenticate themselves with the platform. The result of successful authentication is an UluruSession, which the application may then use to obtain references to Service instances within the platform. These Service instances implement (at least) one of the Uluru Service Interfaces. For example, in Figure 2, the ContentManagementService is accessed by an application so that it can perform a content search and a subsequent content fetch. Uluru Services act as “Business Delegates” \[1\], as they manage the complexity of interacting with the service’s specific implementation technology (e.g. EJBs) and handle service-level exceptions, whilst maintaining a simple interface for use by the client applications.

The system is monitored through the use of a generalized Event Visualizer\[3\], which is a generic solution that monitors events in an Application and subsequently provides a graphical representation of the system dynamics. This is particularly useful when demonstrating the system, as it provides insight into the platform’s underlying processes.

### 5. Personalization

Uluru makes use of the Duine Toolkit, which is an SDK that allows developers to integrate prediction engines into their own applications. Predictions are made based on information about the user that is stored in “user profiles”. The result of a prediction engine is the retrieved set of information items, along with added data, which indicates how “interesting” each piece of information is for the user. Examples of prediction techniques include social filtering, case-based reasoning (CBR), information filtering, item-item filtering, genre Least Mean Square (genreLMS), stereotype comparisons, most popular etc.

In contrast to most currently available personalized information systems that focus on using a single prediction technique or a fixed combination of two or three techniques, Duine combines multiple techniques in a dynamic and intelligent way, thereby providing more accurate and stable predictions \[3\].

Duine allows the dynamic use of different combinations of prediction techniques at runtime where the choice for such combinations is based on up-to-date knowledge about the user, other users, the information and the system itself.

\[1\] http://mis.sourceforge.net/
A combination of prediction techniques is denoted as a “prediction strategy”. The prediction techniques generate predictions based upon the user profile and information, whereas prediction strategies choose one or more predictors (prediction techniques and/or other prediction strategies) that generate predictions on behalf of the strategy (see Figure 3). The 1..n indications show that for example the prediction engine provides feedback to 1 or more predictors and that every prediction technique is used by 1 or more strategies.

Using Duine, the application developer is free to choose a set of prediction techniques and strategies to be used by the prediction engine. These prediction strategies choose (at runtime) the best (combination of) prediction techniques at the moment an actual prediction is required, taking into account the most up-to-date knowledge about the current user, other users, the information and the system itself.

**Figure 3: Prediction Strategies and Techniques.**

Uluru has a metering, accounting and charging infrastructure [4] that allows hosted Applications to charge for their services. The Parlay Group\(^2\) is in the process of developing a set of technology-independent APIs that enable 3rd party telecommunications services to be developed in a vendor-neutral manner. The Parlay Specifications have been developed jointly by the European Telecommunications Standards Institute (ETSI), and the Third Generation Partnership Program (3GPP).

The Parlay APIs provide access to a number of general functionalities including: Charging, Mobility, Terminal Capabilities, Presence and Availability Management etc. Within Uluru, we utilized Parlay’s Charging API specification to enable the platform to be easily integrated with mobile operator payment systems. Using a Parlay bridge, the platform can be used to make payments, perform reservations and act on reservations. Uluru contains a charging engine that takes usage metrics and a set of business rules as parameters and calculates an associated charge as an output. This charge can be applied directly to a user’s prepaid mobile phone account when the service has been successfully delivered. Alternatively, the calculated charge may be used to make a “reservation” on the prepaid account, where the usage metrics passed to the engine are “best-guess expected usage” figures.

Uluru provides a set of “Payment Instruments”, each of which can be accessed using the Parlay API. Using this approach, the FinEx system can interact with different payment instruments using exactly the same interface and semantics. For example, the platform can use Payment Instruments that realize their Payment functionality via credit card clearance, mobile operator prepaid account or even through PayPal (email-based). As far as the platform is concerned, each Payment Instrument is treated in the same manner and the details of how the payment is effected remain hidden.

**7. Uluru Applications**

Uluru provides an infrastructure for the delivery of a wide range of multimedia content and advanced mobile “infotainment” applications to users of the current generation of multi-media capable mobile phones. A number of different Uluru applications have been developed and trialled including:

- **VIPGolf**: an application that can be used while golfing that provides video footage of the course and a set of useful golfing tools
- **Newsflash!**: an application that provides users with a personalized selection of breaking news items with text, images, audio and video content items.

In the next section, we present “Callsong”, an Uluru application that makes use of the platform’s Personalization and Accounting/Charging functions.

**7.1. Callsong**

The broadcast trial application Callsong is a personalized audio broadcasting service whose focus is on user interaction with broadcast audio content. NCRV (one of the major Dutch public broadcast organizations) showed an early interest to participate in Uluru user trials. Their team examined the possibilities of Uluru and compared it with their needs, resulting in a personalized broadcast application. CaliSong listeners can:

- interact with the submitted program at any moment (by performing a track “skip”),
- influence the presented content (by providing +ve or –ve feedback to the personalization engine) and,
- communicate with other listeners (by recommending “favorite” songs to CallSong users via messaging).
Callsong presents a personalized playlist which must be a balance between two conflicting user requirements; a desire for repetition versus a desire for surprise [5]. Callsong applies the following steps:

1. A query selects a subset of potential interesting content; due to the small scale, we take all content.
2. For each item in the subset the actual prediction and confidence is calculated, using the Duine system.
3. Using a stochastic model, a short list is created, which is an order of magnitude larger than the final playlist delivered to the user. The model assures that the majority of items have a favorable prediction, but also allows a pass-through for unfavorable items, with a low chance. In addition, certain filters are applied that can, for example, prevent favorable items from returning too quickly (rotation time).
4. Finally, the playlist is created by using a constraint satisfaction problem solver [6].

To prevent excessive calculation time for a given playlist, the maximum number of iterations is limited and when this limit is reached, the “best” solution found so far is returned. The constraints to satisfy are:
   a. All items must be different.
   b. Transitions between genres of successive (music) items must be gentle.
   c. The must start and end with a music item.
   d. Editorial items must be followed by a Music item.
   e. The playlist must contain at least 1 editorial item.
   f. The playlist must contain at least 40% music.
   g. The playlist must contain favorable items.

Users can configure their predictions by manipulating a number of content formulas. The formulas are named after three popular (public) radio stations and reflect the editorial style of the station, bridging the gap between users and the particular content characteristics. During song playback, users can indicate whether they want more (strongly favorable rating), less (slightly negative rating), skip the song completely (strongly negative rating) or just let it play (slightly positive rating). These ratings are immediately used when generating the subsequent playlist. User feedback learned that users wished to be able to skip items that they had previously judged positively and also that users have different preferences in regard to the speed with which their profile adapts.

8. Further Work

MPEG-21 standardization [7] in the areas of Digital Item Adaptation (DIA) and DRM are certainly applicable to applications operating within Uluru’s domain. We are currently investigating how we can integrate these emerging technologies into Uluru. One promising initiative in the DRM domain is OpenIPMP3, which is developing an open, standards-compliant DRM platform based upon J2EE technologies. We also plan to integrate Uluru with WASP4, a platform for context-aware services, thereby extending Uluru to provide location-based services to its hosted applications.

9. Acknowledgements

The following people have contributed to the design, development and/or research aspects of Uluru: Frank Urbanus, Ton van der Poel, Allard van der Bas, Wim Nouwens and Peter Fennema. We would also like to acknowledge the valuable contributions made by the Ericsson, NOB and Mobiview research team members who are not individually mentioned here. Duine was developed and designed by Mark van Setten and implemented by Jaap Reitsma and Peter Ebben.

10. References


3 http://www.openipmp.org/
4 http://www.freeband.nl/projecten/wasp/ENindex.html