Overcoming challenges in delivering services to social networks in location centric scenarios

Tobias Lange*
SAP Research
52 Merivale St.
South Brisbane, Australia
tl@einbecker.net

Marek Kowalkiewicz
SAP Research
52 Merivale St.
South Brisbane, Australia
marek.kowalkiewicz@sap.com

Thomas Springer
Dresden University of Technology
Nöthnitzer Str. 46
Dresden, Germany
thomas.springer@tu-dresden.de

Tobias Raub
SAP Research
52 Merivale St.
South Brisbane, Australia
t.raub@sap.com

ABSTRACT

With the advent of ever more powerful mobile devices over recent years, an increasing wealth of technical functionalities has become ready for use at once. These were previously only available in separate specialized devices with limited functionality or non-mobile equipment. In addition, mobile platform providers have taken a more open approach, enabling community-members to develop applications for their platforms and to deliver them as readily consumable services to the public. Both trends combined have led to a significant increase in the number of innovative mobile applications. More recently, leveraging mobile users' geolocation for provision of services has become the focus of a number of organizations active in the field. In this paper we propose a solution that addresses some challenges when creating location based social networks and offering relevant services to participants in these networks. We have applied this solution in a use case with an Australian based transportation service provider.

Categories and Subject Descriptors

H.5 [**Information Interfaces and Presentation**]: User Interfaces

General Terms

Prototyping, Screen design, User-centered design

*Tobias Lange worked on this topic during his internship at SAP Research

ACM LBSN '09, November 3, 2009. Seattle, WA, USA

Keywords

location based social networks, dynamic service provision

1. INTRODUCTION

Mobile applications that provide services using geolocation information are becoming more and more popular, with significant increase in numbers over recent years. Applications such as Google Latitude make extensive use of existing centralized frameworks to facilitate creation of social networks focusing on exchange of self-location information. Existing large social networks—originally not considering their users' geolocation as an important factor—start to make proper use of such assets. However, we are still waiting for a "classic" social network to extensively exploit geolocation.

We have looked into enabling geolocation features in a community that has historically been forming a real-world social network as opposed to a virtual one. The community is formed around a large transportation services provider in Australia. As one way of improving their brand image, the service provider, further referred to as TSP (Transportation Services Provider), has decided to focus on: (1) enabling their customers to submit roadside assistance request information electronically with a focus on GPS-enabled mobile devices, (2) let their customers form ad-hoc social networks, (3) deliver geographically relevant information (such as petrol prices or road hazards). TSP clearly sees the need for a solution to create a Location Based Social Network and not only facilitate communication in this network but also offer relevant services to such a customer base.

With the TSP, we identified a set of services to be offered based on geolocation information. We have also identified a set of requirements and challenges we had to address before achieving the expected outcomes. In result, we have created (1) a framework to distribute, store, and reuse geolocation information, (2) a novel approach to creating cross-platform native web-service based applications, and (3) an infrastructure to deliver customized user interfaces to web services.

In this paper we address the issue of creating multi platform mobile applications for enabling location based social networks and providing relevant services for them. We focus

on a wider set of technologies required to build an end-to-end solution. The contribution of this work is in analyzing existing solutions and positioning them against requirements expressed by service providers and social network facilitators. We also introduce a solution for realizing scenarios where location based social networks contain geographically relevant services, and where development efforts in creating and modifying client applications are reduced to the minimum.

In Section 2 we analyze already existing frameworks for UI generation and device independent applications. We intentionally do not focus on existing social networking solutions, as they are well known to the readers. In Section 3 we list most important issues raised by our commercial partners when developing location based social network applications. This forms a basis for a depiction of a solution we have worked on, described in Section 4. In Section 5 we analyse performance of our approach. We conclude our work in Section 6.

2. STATE OF THE ART

Several methods exist to create device independent applications [4]. During design time, all of these use a device-independent application description, be it in the form of a markup language or a higher programming language. Also, the user always faces a device specific interface—the pixels will always be somehow drawn by the operating system's graphic system. However, the steps in between are hugely different. There are solutions that compile platform independent code to platform specific code that then has to be compiled by the platform's SDK. An overview of the possible translation efforts is shown in Figure 1.

Some frameworks allow the direct compilation of the general application description into platform specific byte code. If this is the case, the general application description could actually already be platform independent source code. This model can be found with a range of multi-platform development frameworks, the most prominent being QT and GTK. Others take XML-based UI description languages and either create native interfaces, e.g. using Java AWT or Swing, or transform the XML with the help of XSL-T to, for example, HTML, WML or XUL [5, 10], XIML [13]. An alternative approach would be to facilitate mobile agents to generate and transport the user interface. All of these approaches are summarized by Mitrovic and Mena[9].

Other frameworks introduce a meta-language as the general application description that can be translated to individual, platform specific source code. An example of this method is the Simplified Wrapper and Interface Generator[14].

Another model is to compile the platform independent source code (the general application description) into platform independent byte code that can be interpreted (or just-in-time compiled) on the platform. Sun's Java platforms follow this model. Other examples include XAML [8].

The last option is to create a platform independent user interface description that is interpreted on the device during runtime. HTML, WAP, XForms [3], and the approach introduced in this paper all follow this approach.

Other examples of similar frameworks include: UIML [12, 1, 2], AUIML [7], UsiXML [6], Yahoo! Blueprint [11]

During the assessment of the existing solutions, we concluded that it is possible to create a device-independent

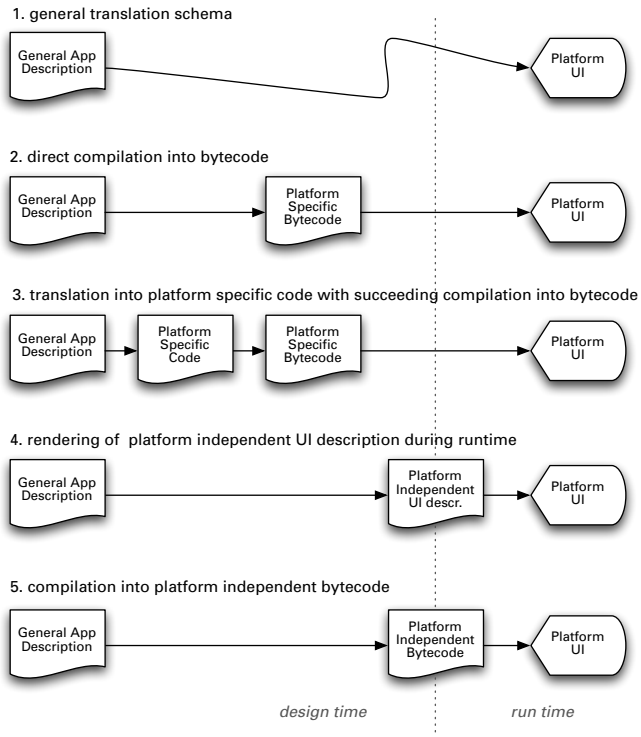

Figure 1: Translation approaches

application through the use of a specialized user-interface markup language that could describe both the user-interface patterns described in the pattern repositories mentioned above as well as those that we have discovered ourselves. While the languages that have been established work well in their domains, we believe they are not completely suitable for the use case that we target: none of them could describe all of the UI patterns [15] relevant for our application domain in easy, non-bloated markup that was truly platform or rendering independent and that could be rendered into native controls that behave in a way that conform to the platform's look and feel.

3. ISSUES

While discussing with TSP and other parties potentially interested in our solution, we have identified five important challenges that businesses meet when deciding to deliver some of their services via mobile channels. We only focused on services that relied on geolocation information in their functioning or whose relevance depended on the geographical location of a user. For instance a vehicle insurance claim sent from a mobile device would use geolocation information in the claim content, a roadside assistance request would be only available in the physical area that TSP is responsible for, and a buddy list function would require all participating mobile applications to transmit their location.

3.1 Multiple mobile platforms

When developing a mobile application for the general market, one has to consider multiple software platforms. Every major manufacturer of mobile phones relies on an operating system which is typically not compatible with those by other

manufacturers.

While it is possible to create web applications (applications that run in a web browser) that largely solve the problem of multiple mobile platforms, service providers are not always happy with such a solution. The major issue is achieving an acceptable user experience. It is virtually impossible to create a web application that preserves the native look and feel of a platform it is being accessed from. The response times of web applications are—in many cases—also below expected standards.

3.2 Relevance of services

Even a single service provider (such as a bank or an insurance institution) offers a large number of services, often too many for a person to sift through. That is why providers frequently use personalized catalogues of services. It is especially important to be able to offer contextually relevant services when the interaction involves small form factor devices, such as mobile phones.

We chose to use geolocation information as a primary context filter when offering services in location based social networking applications. As motivated by our work with the TSP, we have identified a set of services that could be offered to end users based on their geographical location and have focused on these.

3.3 Hardware access limitations

Geographically enabled applications require access to hardware devices on the phones, such as GPS chips. Web browsers on mobile phones—with notable exceptions—in general do not yet provide access to such hardware of the phones. In our implementation for the TSP, we also require access to other devices, such as a photo camera.

3.4 Task orientation vs. application orientation

From the user experience point of view, it is required that application users are given a consistent application interface. Specifically, currently—even if a group of services is related to one task—every service made available on a mobile phone is "wrapped" in one application. For instance motorists may need to use a number of mobile applications while driving: an application to alert about road hazards, an application to communicate with fellow motorists (for instance family members in another car), and a roadside assistance service.

3.5 Dynamic provision of services

Experiences of the TSP show that it is important to be able to dynamically modify capabilities of applications offered to end users (for instance restricting ability to request a specific type of service, instead of a generic one, in case of emergency situations that put high load on the TSP). It is also important to be able to deliver new services dynamically, even if not explicitly requested by users. Recent examples of floods in the TSP region triggered a need to deploy a "flood-update" service to all mobile phones of the motorists in the affected area.

4. OUR APPROACH

In our work, we have focused on addressing the issues listed in the previous section, while preparing a set of applications requested by the TSP. We have built a solution

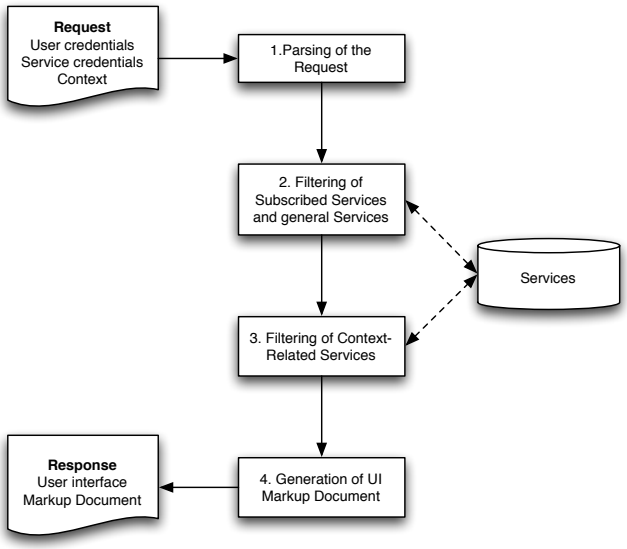

Figure 2: The process of generating markup for a mobile phone application based on context information (geolocation) of the social network participant.

which incorporates a server running in a public cloud, providing services enabling social networking capabilities and intermediating communication between client applications and back end systems of the TSP. To address the issue described in Section 3.1, we have developed a markup language for describing user interfaces of the mobile applications. The markup language is interpreted by framework applications developed separately for each mobile operating system to create native-look-and-feel applications. In our approach we address UI meta levels from a very generic UI concept description down to concrete UI implementation.

In order to address the issue from Section 3.2, the server stores information about geographical relevance of services (including information feeds), to offer only the geographically relevant services to the end users. Through using the framework applications written specifically for each mobile operating system, we can get easy access to hardware components of the mobile phones, therefore addressing the issue listed in Section 3.3.

We have also developed our framework applications in a way that one application on the mobile phone could incorporate an arbitrary number of services in it (limited only by software platform limitations and common sense in user experience design). The server component can dynamically push UI description components (using the developed markup language, cf. Figure 2). Through this approach we have addressed the issues listed in Sections 3.4 and 3.5. The approach resembles Web browser application architecture, however there are crucial differences, mostly related to native look and feel and hardware integration, as discussed above.

5. VALIDATION

To test the performance of the implemented solution and compare it to web applications as well as traditional native applications, we set up a test application that was imple-

Approach	Web	Traditional	Proposed
App. Startup	**≈ 3.3s**	**1.2s**	**1.6s**
Load into memory	≈ 1.3s	0.5s	0.5s
Request Generated	—	—	0.1s
Response Received	≈ 0.5s	—	0.2s
Fully loaded	1.5s	0.7s	0.8s
Responsiveness	**good**	**optimal**	**optimal**
Pan and Zoom	small lag	no lag	no lag
Text pin onto map	≈ 1s	no delay	no delay
Text input	slight lag	no lag	no lag
Submit to server	≈ 0.5s	0.2s	0.2s

Table 1: Performance comparison of a Web, Traditional Native Application and our approach. Shown are the mean values of 20 test runs. The standard deviation of each of the measured values was small (maximum of 0.2s), and because of the small sample size, is not shown here.

mented in all three ways. The application is a simple map display that shows pins based on a GeoRSS feed. It also includes a single text field where the user can enter text and submit this text to a web server. We tested these applications on an original iPhone and an iPhone 3G, using both Wi-Fi and the cellular technologies 3G, EDGE, and GPRS. The performance figures provide only a rough estimate of the performance of our solution, but we believe that they provide a general trend that holds true throughout the application scenarios we envisioned.

The usage of 3G/EDGE/GPRS instead of Wi-Fi increased the response times by the web server by around 0.2s (GPRS: 0.4s), but we did not include this parameter in the table in order to maintain simplicity. There was no noticeable difference between the speed of 3G and EDGE, and the larger GPRS number is mainly due to the areas we could test GPRS-only reception—which meant a general lower quality of cellular network reception, as 3G and EDGE were not available anymore. As our markup (both for the request and the response) is fairly lightweight, not much data needs to be transmitted which, in our opinion, explains the similar results using different cellular technologies.

As this comparison shows, the use of generic user interface markup is only slightly slower when launching the application than the traditional model, but both are starting at least twice as fast as a web application. After launch, the performance tests show our approach to be as fast and responsive as a traditional application—which is reasonable, as our application is creating native controls in memory during start-up, so it is using the same model as a traditional application during main application runtime. Although web applications do fairly well on an iPhone, during user tests all participants felt the web application was slightly lagging when compared to native applications or our approach for most usages.

Noticeably, the computing time needed for the interface generation did not increase much when the interface became more complex—even an application interface with twenty different types of views and elements within those views, took only around 1s to be generated once the markup was received.

6. SUMMARY AND OUTLOOK

With this work, we have proven that it is possible to use a markup language to dynamically provide user interfaces to web services to mobile applications running on various platforms. We have introduced a new language (facilitating embedded XForms markup) to transmit information about web services, data models, instance data, and a user interface layer using all of the previously mentioned to mobile devices during runtime. With this technology, one can create applications that take the current context of the user into account and request the server to provide this markup for the services that are relevant to the users in their context. The proposed solution delivers superior user experience and performance of native applications while being as flexible in terms of development and deployment as web applications. The architecture can be used in various application scenarios and allows to rapidly prototype ideas in the vast area of web-service-based mobile applications. We are currently deploying our solution together with aforementioned TSP.

7. REFERENCES

[1] M. Abrams, C. Phanouriou, A. Batongbacal, S. Williams, and J. Shuster. UIML: an appliance-independent XML user interface language. *Computer Networks-the International Journal of Computer and Telecommunications Networkin*, 31(11):1695–1708, 1999.

[2] M. Ali, M. Abrams, and I. Harmonia. Simplifying Construction of Multi-Platform User Interfaces Using UIML. In *UIML Europe 2001 Conference*. Harmonia, Inc., 2001.

[3] J. M. Boyer. XForms 1.1 W3C Candidate Recommendation. *World Wide Web Consortium*, 2007.

[4] M. Cusumano and D. Yoffie. What Netscape learned from cross-platform software development. 1999.

[5] D. Hyatt, B. Goodger, I. Hickson, and C. Waterson. XML user interface language (XUL) 1.0. *Mozilla.org*, 2001.

[6] Q. Limbourg, J. Vanderdonckt, B. Michotte, L. Bouillon, and V. Lopez-Jaquero. Usixml: A language supporting multi-path development of user interfaces. *Lecture Notes in Computer Science*, 3425:200, 2005.

[7] R. Merrick. AUIML: An XML Vocabulary for Describing User Interfaces. Device Independent User Interfaces in XML, 2001.

[8] Microsoft. Extensible application markup language, 2008.

[9] N. Mitrovic and E. Mena. Adaptive user interface for mobile devices. *Lecture Notes in Computer Science*, 2545:29–43, 2002.

[10] Mozilla Developer Center. XUL, 2009.

[11] Y. D. Network. Yahoo! mobile, 2009.

[12] OASIS. User interface markup language.

[13] A. Puerta and J. Eisenstein. XIML: A Universal Language for User Interfaces. White paper, 2001.

[14] SWIG. Simplified wrapper and interface generator, 2009.

[15] Welie. Patterns in Interaction Design, 2009.