Yousef Daradkeh, Dmitry Namiot, Manfred Sneps-Sneppe/ International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com
Vol. 2, Issue 2,Mar-Apr 2012, pp.710-714

Telecom Mashups: a Practical Example

Yousef Daradkeh[1], Dmitry Namiot[2], and Manfred Sneps-Sneppe[3]
[1] Department of Computer Information Systems,The Jordan of University, Aqaba Branch, Jordan
[2] Lomonosov Moscow State University, Moscow, Russia
[3]Ventspils University College, Ventspils International Radio Astronomy Centre
Inzenieru st 101, Ventspils, LV-3601 Latvia,

Abstract
This paper describes a practical example of telecom mashup. Application (service) that mixes SMS, IVR and web services for presenting an anonymous classified service. This example demonstrates the power of telecom mashups, provides a useful use case for mixing open web API's and telecom applications.

Keywords **– service, SMS, IVR, API, REST, voice.**

1. INTRODUCTION
In Web development, a mashup is a Web page or application that uses and combines data, presentation or functionality from two or more sources to create new services.
The term implies easy, fast integration, frequently using open APIs (an interface implemented by a software program that enables it to interact with other software) and data sources to produce enriched results that were not necessarily the original reason for producing the raw source data.

The main characteristics of the mashup are combination, visualization and aggregation. Mashup is important to make more useful already existing data, moreover for personal and professional use. [1]

The emerging Web 2.0 marketplace presents an important opportunity for Telecom operators to sell their own capabilities and content as services. Operators have a wealth of content associated with their network as well as core network enablers, e.g. call control, presence and messaging, which could serve as potential new revenue streams in a Web 2.0 world. Moreover, with the looming threat from Internet companies, there is an increasing need for operators to make both core and value-added functions reusable and mashable. [2].

There are basic things. From the practical point of view mashup (telecom mashup particularly) means some combination of applications. For the telecom mashups it means that we will not or, in the most cases, cannot develop the whole application as a monolithic system. We can count several reasons for that. It could be a complexity of development, lack of the proper description for the whole business process, lack of the expertise etc.

In the applications development mashups plays the same role as components on the low level programming. With mashups development we are assembling our system from the individual blocks. We can treat that blocks as programming components but usually, especially in the telecom development they have a much bigger level of integration.

II. THE TASK
Now let us go to the development. Our idea was to develop classified system for the mobile users. Or classified system mobile users can easily access to. Simply, our users should be able to place individual messages right from the mobile. But it is only a part of the story. The big problem for the private classifieds is privacy issue. Suppose you are selling something. Placed an ad and sold your stuff at the end of the day. But your ad is still alive, your contact numbers are alive, you are still getting unwanted calls etc. So our idea was to place ad without revealing end-user contact details. Lets readers easily contact with publishers, lets publishers easily get responses but in the same time keep publisher's contact info private. In other words our classified system should support the ability for readers get in contact with authors without knowing their contact details. And ability for authors collects the responses without publishing their actual contact info. That was actually our task and its description and especially our solution listed below are innovative.

III. THE SOLUTION
Internal consistency should be maintained regarding Two things at the first hand – how to place ad and how to store them? We decided to let users place ad by SMS. It is the most mobile friendly way and the most simple to use. Just send a text to some service number and your ad is alive. At this moment everything is quite standard. As the service number we've used an ordinary SIM card with GSM modem that can programmatically collect incoming messages and proceed them by our wish.

The next question is obviously where and how to save the messages. And the way we store our messages will define how to show them also. We've decided to use an existing blog platform for saving our messages. What do we need only – is a blog platform with open API, that let us publish the messages programmatically. Our GSM modem will collect messages from the service number and publish them in the blog via the public API. What do we win with this solution?

We've got a backend ready for accepting messages and we've got a standard mechanism for showing messages too. Blog platform lets access to blogs from mobile phones (mobile internet), our messages store is just yet another blog and we can use build-in mobile access for reading our messages from mobile phones.

So we've solved the problem with creating/publishing/reading messages using free available tools/services from the Net. In our practical case we've used livejournal.com service, but actually any blog platform with open API is Ok. Google's Blogspot for example is a good candidate too. Many modern blog platforms have got well designed mobile versions. For example, Wordpress and Posterous (it is what we've tested with this mashup). And of course, with the blog platform we can get hosting too (e.g. free blogspot hosting from Google).

But what about the contact information? In other words what shall we publish in our classified? Our GSM modem is getting SMS for the publisher. There is a text we need to publish and a real phone number used for sending that SMS. We can publish text as is but we will not publish that actual phone number. Here goes another source we've used in our mashup – Voice SMS. And that is the main idea (main trick) behind this mashup.

Voice SMS is IVR based service that lets call to some service number, leave (record) voice message and set mobile phone number this message addressed too. As soon as voice message is recorded service sends an ordinary SMS to the target phone (phone, associated with the given message). This message contains a standard text, like 'You've got a new message' as well as the phone number for the Voice SMS service. So it is possible to call back Voice SMS service right from SMS (just one click, because phone number in the text is getting recognized by the build-in SMS client) and listen the recorded message. And IVR application lets respond to that message with a new Voice message etc. You can reply (leave a voice message), record new message etc. You can treat this at the first estimation as a voice mail with SMS notifications. But there are several significant differences.

At the first hand it is the lack of pre-registering. You do not need to create your mailbox in advance. You can start use it any time – just record the message and set mobile phone number this message addressed to.
For the telecom provider it is a simple swap: one SMS (notification) versus one voice call (for listening, replying etc.)
And of course Voice SMS is an all-sufficient service that could be used separately from classified system. Here it is just a good fit for communications in our classified system. And there is one important feature that makes Voice SMS really different from voice mail systems. Voice SMS was developed for using in web mashups too. In other words this service has got own API. And this API (technically it is HTTP based REST) lets you, in particularly, replace the real phone number with some unique digital code. In other words, instead of sending (addressing) our voice SMS to some real number we can send (address) it to some virtual number that corresponds of course to some real mobile number. And what is important here – we can create that mapping (virtual number, real number) programmatically.

So finally, our Voice SMS lets you use a virtual number (just a code) instead of the target mobile phone. It means that when you send Voice SMS you can any time use that number instead of the real phone. Where this number (code) goes from? Real phone number phone owner can register such code for own real phone number via web portal or via API. And tell it to the potential correspondents. Like old pager system – use that code for reaching me. And because there is an API for this operation than any third party application can register a virtual number for the user.

And with all this we can keep in mind that virtual number exists only on the Voice SMS interface level. Internally, service has got of course the full information about the mapping (virtual code, real phone number). So when user types from the phone a virtual code for addressing Voice SMS to, our server is aware about the real phone number notification SMS will send to. It is very important for the operators too. They may have law restrictions for anonymous calls (messages). And virtual numbers in our Voice SMS service comply with them. On the end user level virtual number hides the real number (so, it is like anonymous messages), but on the system level the service owner is completely aware of course about the real numbers.

As the last important things we can note, that with this approach we don need mandatory follow to one-to-one relations in our (real number, virtual number) mapping. It is actually one-to-many. For one real phone number we can create more than one virtual number (code). Actually we can threat our virtual codes (numbers) as the temporary keys used during the communications sessions. For the each séance we can create own virtual number for example.

Now we have all the pieces for assembling our service. GSM modem gets a new SMS with text ad. Service application gets an original phone number from received SMS and creates (register) virtual number code for it (via API). As a next step this service puts text ad and virtual number into blog (publishes it as a new message).

So as soon as some reader approaches the mobile version of our blog he/she will see text ads (as is, as they were prepared by the authors) as well as virtual codes for contacts via Voice SMS for the each ad. Each that contact info looks like actually as an old well known pager info: "For contact call service number NNNN and leave your message to the member MMMMM". So any interested reader can dial, record the voice message and type virtual number as an address. He/she obviously does not need to remember that virtual number – it is printed right in the ad. And of course, ad owner (publisher) does need to know about his/her virtual numbers too. If I am

publisher for example, it does not matter which number is used to get in contact with me. I know only, that my real phone number is not disclosed.

As soon as reply message is recorded, ad owner (owner of the virtual number) will get SMS notification about this new recording. He/she can simply call a service number, IVR application recognizes his phone and let him/her listen recorder messages, respond for them etc. So there is no need to register in advance again.

So finally our service is able to organize the first contact anonymously. And the rest of the deal depends on the both parties of course. They can continue use Voice SMS and keep contacts undisclosed, or inform each other about the real contacts data and go next without our Voice SMS.

And because we are using virtual number (codes) for contacts it is very easy to stop accept messages for any ad. Ad publisher (virtual number owner) can simply call from his/her phone to the Voice SMS service number. IVR recognizes his/her number, detects all the virtual number associated with this phone and offer options to delete any (all) of existing mappings. As we said above our server is pretty aware about the mappings (real phone, virtual number). This knowledge supports the game. So as soon as some virtual code is deleted, any future attempt to leave message for it leads to the standard system alert like 'This code does not exist anymore, you could not contact through it'.

VI. SPECIAL TELECOM MASHUPS

The Now let us see on the above-mentioned mashup from the system point of view. What we did there actually? It is a telecom service, at the first hand, that uses telecom part for the communications and web as a data store. Unified access (web based API's) let us quickly build this useful solution, redeploying many existing (and in the many cases – open sourced) components. But once again – it is a telecom service at the first hand. And actually the telco could get the benefits from such kind of mashups. Because this mashup stimulates the usage of the basic services (calls and messages). Basic services are billed for end-users. Even without the using any additional price for service calls (premium numbers) any call/SMS message is billed as per end-user plan for example. So actually the telco (not internet companies) could get benefits from some mashups. And "some mashups" here are the mashups that actually stimulate the usage of the basic telecom services or getting access to data specific to the telecom. And we would like to highlight at least two prospect areas where, by our opinion, the telecom could get advantages from mashups. And the solutions (API's) for the both below-described areas are yet to come.

At the first hand it is call control on the mobile terminal itself. Curiously, but even for the more than powerful modern smart phones there are no API's for call control. From the development point of view we have a lot of tools for access to the web right from the phone for example and we have almost nothing for the call processing. There are simply no tools for the dealing with the smart phone as a phone.

Lets us start from end-users (services) prospects. We are talking about potential services that use mobile phone number as a service number and lets proceed incoming calls right on the phone. For example, accept a call and play back a media file located on the phone (and recorded previously with the same phone) – own auto-answering service that could be switched on/off by demand. Or call forwarding depends on the given conditions. Another possible service: accept a call and record it as a media file (for storing on the phone or even publishing on the web). Or simply IVR service, that could be deployed right on the phone.

Very clean set of useful services. They are clean and useful from the end-user prospects and in the same time they are tight connected with the basic telecom services (read – tight connected with the billed services). Incoming calls are ordinary paid calls without any premium numbers and services will be deployed on end-users handsets without dealing with operator's data-centers.

But at this moment we have nothing for mobile call control services from the developer's point of view.

iPhone is closed platform and does not provide any hooks here, there are nothing similar in Android or Symbian world too.

On JME world we have (theoretically) Java Telephony API (JSR253). The Java Telephony API (JTAPI) was designed by a consortium of industry-leading computer and telecommunications companies interested in designing a portable, object-oriented API for computer-telephony integrated call control. The requirements for this API were that it be simple, easy to implement on top of existing CTI platforms, integrate both first-party and third-party call control models, and be scalable and extensible [3]. But this JSR is actually never implemented on practice. There are simply no handsets with JTAPI support. So this more than interesting area is waiting for open API's and mashups.

In the meantime we can mention here cloud phone API's. If the registration (sign in) process is easy, call control based mashup could be deployed on the cloud. It is not your own phone number, but the development for the modern systems like Twilio [7] is really easy. Twilio allows your web application to easily make and receive phone calls and SMS text messages using a simple web service API and basic web programming.

And we can also mention here our own development – web proxy for Asterisk [8]. Actually this tool was developed before any modern cloud API's. We propose to integrate a new component (proxy) into the Asterisk platform. The main functionality of the proxy is to translate telecommunication calls into HTTP requests to external web services. Telecommunication services are located separately from the

PBX, while the information they receive from Asterisk is presented as a HTTP-request. Technically, HTTP GET/POST request is a request, in which external telecommunications service passed information about the subscriber's name - CallerIdName, caller's number - CallerIdNumber and and called number - Extension. Upon receiving necessary parameters, such as (calling/called number) a web service produces and forwards its instructions to the proxy. The latter receives and translates them into Asterisk instructions. The development of such services under the architecture described above is similar to a conventional CGI-script, for which there is a plenty of programming tools. As a result a programmer doesn't need to be familiar with the Asterisk API. Asterisk is open source software PBX, but with the special hardware it can land (proceed) ordinary phone calls. And with GSM cards for example, you can terminate GSM (mobile) calls on your Asterisk. So with this open source proxy you can write web applications for dealing with calls on your own SIM-card for example.

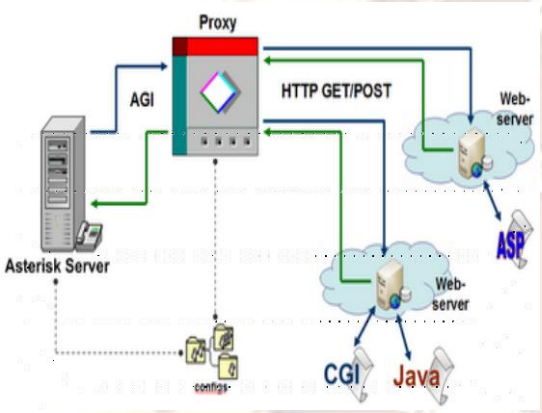

Figure1. The following picture illustrates the process:

Once again – with plain old CGI scripts web developers can write call processing for mobile calls. But it is still a separate server. As we note above the whole development should be moved to the mobile phone itself. My own mobile phone should be a server by demand for example. This promising opportunity is still open.

Another interesting area for telecom is geo fencing in location based systems (LBS). As per Wikipedia a Geo-fence is a virtual perimeter for a real-world geographic area.
Virtual perimeters could be static (pre-defined) or dynamic. Dynamically generated geo-fence could be for example a radius around a store or point location. Or a geo-fence can be a predefined set of boundaries, like school attendance zones or neighborhood boundaries. Custom digitized geo-fences are also in use.
When the location-aware device of a location-based service (LBS) user enters or exits a geo-fence, the device receives a generated notification. This notification might contain information about the location of the device. The geo-fence notice might be sent to a mobile phone for example as well raise any another type of actions [4].
We can describe geo fencing as proactive LBS. In the many cases proactive systems are much more convenient than reactive ones, where the user has to explicitly request for location-based data. There are different kinds of location events the GPS position fix can be tested for, for example, whether the user is in close proximity to a point of interest (POI) or to another user. Why it is interesting? It looks like geo-fences become one of the hottest areas in Location Based Services. Right now the original development for LBS is more or less completed, got some common standards and applications (think about the various implementations of Places services like Foursquare, Twitter places, Google places, Facebook places etc.) All of them let you either share location info or get some things nearby. And geo-fence opens the door to some personalization. You can get some custom tailored data right on the place, especially when both data stream and place (area) for receiving it are dynamically generated. The key word here is "proactive". Subscribers should get notifications automatically. At the second, geo-fence will be in the nearest future very deep integrated with sensors and M2M applications. City sensors in the various "smart city" projects will define geo-fences and notification messages.
The word "proactive" means that for the implementation we should have some form of the monitoring. Where can we set this monitoring? Obviously, the following three options cover all the cases:

a) client side checking
b) some external server
c) on the operator's side

Let us start with client side checking. It should be a background application that constantly works on the phone and compares its current location with the pre-defined (or preloaded) areas. As soon as our phone is in the area covered by the trigger we should raise an appropriate alarm. The problem for such kind of services is just one but a very significant – battery life. And as soon as we start to think about battery, we should switch from GPS usage to some solutions like CellID for the positioning and loose the precision.
Any solution for the server-side processing will simulate at the end of the day the mobile operator. Operator in the mobile network is an analogue of server side processing that monitors its clients (mobile terminals, phones). So our conclusion is simple. The Geo Fence should be a part of telecom API's. That is the point. Here is at least one huge advantage to do geo fencing on the operator's side. Operator is aware about the current location of its subscribers. It means operator can provide a mass delivery for the messages. With operator based monitoring we do not need load applications on the client side (load applications on the phones). Operators are monitoring the terminal's locations anyway (e.g. it could be a law requirement for the emergency calls). And via public API this monitoring could be connected to some actions (e.g. notifications). The typical use case: mobile subscriber switches on deals alert before of the weekend shopping. This action from the operator's point of view could be a trigger for

location detection/monitoring (right before the switching off/unsubscribe command will arrive). Another example – right now telecom providers can organize cell broadcast services. It is the typical example for the proactive service. So practically the question is to provide an interface (API) for the developers that let them describe broadcast areas in geo terms. And we think that geo-fence API for telecom is the most promising area for such kind of services. They (API's) are just starting. For example Sprint Geofence API allows the developer to define an area to determine if a Sprint device is inside or outside the defined area. [5] Another interesting example is OpenApiService from Alcatel-Lucent [6]. These are probably the first attempts to do something in this area. API's are not compatible, there are no so many words about the practical implementations, but we think it is a very promising area and it is again tight connected with the basic telecom services. Simply, it is a thing (entity), mobile operator can bill for. Geo fence is actually an area where operators can gain their natural advantages with the help of mashups.

V. CONCLUSION

This paper describes a telecom mashup combines several telecom services (GSM modem for reading SMS, Voice SMS and customized IVR platform) and open web interfaces (open API for blog platform) for creating useful mobile service. The idea of mashups lets quickly build (assemble) services using best attributes of different worlds. In this case phone is used as it is supposed to use – for communications and web used as a data store. So this mashup actually stimulates the usage of basic telecom services: calls and messages. Based on that idea (stimulate the basic services) we highlight two prospect areas where telecom could get natural advantages via mashups development.

REFERENCES

[1] http://en.wikipedia.org/wiki/Mashup_%28web_application_hybrid%29.
[2] Banerjee N., Dasgupta K., Mukherjea S., "Providing Middleware Support for the Control And Co-ordination of Telecom Mashups", In Proceedings of Middleware Workshop on Next-Generation Converged Networks and Applications (MNCNA), Nov. 2007
[3] http://java.sun.com/products/jtapi/index.jsp
[4] http://en.wikipedia.org/wiki/Geofence
[5] Sprint http://www.wipconnector.com/apis/comments/sprint_geofence_api
[6] Alcatel http://wiki.pre2.openapiservice.com/API:Geofencing
[7] Twilio http://www.twilio.com
[8] A new approach for the development of telecom services http://asterisk.linkstore.ru