A Mobile-Based System for Content Delivery over SMS

Mauro Ricardo da S. Teófilo
Nokia Institute of Technology
Manaus, Brazil
Email: mauro.teofilo@indt.org.br

Vicente Ferreira Lucena Junior
Federal University of Amazonas
Manaus, Brazil
Email: vicente@ufam.edu.br

Luiz Carlos A. M. Cavalcanti
Nokia Institute of Technology
Manaus, Brazil
Email: luiz.cavalcanti@indt.org.br

Daniel Risi
Nokia Institute of Technology
Manaus, Brazil
Email: daniel.risi@indt.org.br

Thomaz Philippe C. Silva
Nokia Institute of Technology
Manaus, Brazil
Email: thomaz.silva@indt.org.br

***Abstract*—The proliferation of mobile applications for smartphones has been based on the presence of a data connection between those devices and the Internet. Therefore, users of entry-level mobile phones as well as users who cannot afford a data plan are excluded from the current dynamics of mobile applications consumption. A possible approach to include those users is the usage of SMS as the communication channel. Although SMS is widely supported by carriers and extremely popular among users, it is not trivial to replace the data connection capabilities with SMS. This work addresses this issue with an integrated solution composed by a client platform and a server-side system.This solution enables GUI-based mobile applications to be transferred over a short number of concatenated SMSs. It also presents components for the creation, deployment and distribution of mobile applications. In order to attest the relevance of such solution, an experiment was conducted and its results indicate a substantial level of acceptance by target group.**

I. Introduction

In recent years, the mobile applications business has grown considerably in importance, which can be at least partially explained by the worldwide increment of mobile internet penetration. The estimated mobile penetration rate was 67% in poor countries in 2010, and a worldwide mobile subscription rate was 73%[1]. In fact, either the acquisition or the usage of mobile applications for both smartphones and feature phones is based on the existence of a data connection from the mobile device to the Internet. Data connections allow the transportation (download) of mobile applications from a mobile application store or market place to the destined mobile device, as well as the transportation of application or user data between the mobile applications and their server-side counterpart logic (e.g. Facebook server, Twitter server, and various other 3rd party servers)[2].

This scenario, however, excludes the vast majority of mobile phone users from the mobile applications consumer segment, especially in emerging regions[3]. Owners of entry-level phones as well as those who cannot afford a costly data plan have so far been underestimated as potential mobile applications customers by manufacturers and carries alike. Moreover, cell towers are rarely built in areas with low population density, GRPS is relatively bandwidth constrained, and faster connectivity (e.g. 3G) is typically only available in urban areas[4]. Figure 1 illustrates the biggest carrier network's 3G coverage in Brazil[5].

Further constraining mobile information access is the prevalence of simple low-cost "feature phones" with limited processing and communication capabilities. Despite the excitement surrounding "smartphones" with increased functionality, 2011 estimates indicated that the ratio of feature phones to smartphones globally was 4 to 1[4][6]; in the same year, smartphones comprised less than 25% of the global handset market[1]. The resulting mobile landscape in emerging regions is a dominance of voice and SMS over data. Reflecting this trend is a range of popular SMS-based applications and services such as mobile money transfer[7], mHealth [8], and social networking[9]. With global SMS volume tripling in three years from 1.8 trillion in 2007 to 6.1 trillion in 2010, SMS will likely continue to remain a primary communications channel in emerging regions.

Among the reasons behind this disparity, one can distinguish the technical obstacles presented to application and content transportation to a mobile device in the absence of a data connection. Both the size of the mobile apps (whether compiled to machine code or byte code, or compressed scripts that are expected to be executed by an interpreter on the mobile device) as well as the size of the HTML, RSS, and other XML and JSON data formats used in state-of-the-art services on the Web, are prohibitively large for transportation over alternative means, such as popular messaging technologies (SMS, USSD, etc.).

The work presented in this paper proposes an end-to-end solution called GEMS (Growth Economies Mobile Service) that allows SMS to act as a viable way to provide a satisfactory application store experience. By utilizing a mobile platform containing a set of GUI components designed to interpret and mount applications based on a custom protocol, the

MOBILWARE 2013, November 11-12, Bologna, Italy

DOI 10.4108/icst.mobilware.2013.254273

communication with the server-side logic can happen over a small amount of concatenated messages. The server platform is also designed to allow the generation of new applications as well as the integration with existing web services through a set of building blocks and specific tools. Thus, application developers can easily deploy and publish new applications, making it possible to build a full ecosystem around SMS applications.

This solution was tested in Brazil with a selected target group from the low Total Cost of Ownership segment, and the findings of such experiment suggest relevance and adequacy of an SMS application store among that user group.

Fig. 1. Biggest carrier network 3G coverage in Brazil.

II. Related Work

There have been certain attempts to solve the presented issue, which can be classified in two categories:

1) "vanilla client", where a plain textual command set is offered to the users, who can type in an SMS the command they would like to execute on the application logic that resides on the server and send the SMS to a specific access number that is connected to the server-side application logic. Examples of such approaches include AppShup[10], UjU[11], ForntLineSMS[12], RapidSMS[13] and MobileDeck[14][15]. These approaches do not focus on how to make mobile apps small enough in size so that can be downloaded to the mobile device over SMS (or other 2G technologies like USSD); rather they focus on how to easily construct the server-side application logic that will process the plain textual messages sent over SMS from the mobile device to the application server and construct the plain textual SMSs that contain the response and which are sent back to the mobile device.
2) "browser client", where a general purpose browser like a WAP browser is used as the mobile device platform for running the mobile application. An example of such approaches is the work done in the WAP Forum (WAP over GSM USSD Specification, WAP-204-WAPOverGSMUSSD-20010730-a). These approaches support the creation of a GUI-based application on the mobile device and thus a much richer UX compared to the approaches in the previous category. However, they have two limitations: first, due to the general purpose GUI construction they support, the size of the mobile application that needs to be downloaded from the application server is still too large to fit in few SMSs; second, the reference implementation of the WAP Forum effort as well as the evolution of those ideas in proxy-based browsing, require continuously interactions with the application server (or network proxy) in order to retrieve the mobile app every time it is launched, or to execute some complicated application logic that the mobile browser cannot execute.

In[16] is proposed a server-side architecture approach that allowed developers to implement and deploy web services that could be requested by mobile client applications that employ SMS as communication channel. In this approach, the authors presented an architecture model named SMBots that provided SMS handling between the MobileDeck like clients and services, services hosting and life-cycle control using OSGi[17] as the services manager.

III. GEMS Concept

The GEMS idea is based on a client platform for the mobile device, which supports a finite set of GUI templates, i.e. visual forms that dictate the layout of the mobile phone screen. The set of GUI templates contains different kinds of menu lists, text editors and input forms, and text layouts. A mobile application for this client platform is a graph of GUI template instances, where each GUI template instance is a reference to a predefined visual form plus text and references to preloaded texts and graphics that are used to fill in the visual form. Most of the mobile application description is codified references to preloaded constructs that are available to the client platform, thus small mobile applications can fit to as little as three SMSs. More complex mobile applications are decomposed to a core application and a set of plugins, each fitting in three to four SMSs. The core application gets downloaded first, and when the user attempts to access functionality of the application that belongs to a missing plugin, the client platform dynamically downloads the missing plugin from the server. These mobile application access small packages of content, again fitting in three or four SMSs. Hence, any interaction between the mobile application and the internet-based server is done over a small ammount of SMSs.

The main difference with the vanilla-client approaches is obvious: our idea supports GUI-based mobile applications, which provide a much richer UX.

GEMS is more similar to the browser-client approaches. The main difference with the browser-client approaches is that GEMS system supports small-sized mobile applications that can be downloaded and installed on the client platform. Thereafter, accessed by the user in an offline mode until there is need to send a request to the server, whereas the browser-client approach supports medium-sized applications that are fetched from the server or need to contact the server in order to execute complex logic. On the other hand, GEMS concept has the limitation that mobile apps cannot have arbitrary visual layouts on the screen; rather, each mobile application screen must conform to the specs of one of the supported GUI templates out of which it is composed.

The GEMS was designed to improve MobileDeck solution[14]. The MobileDeck application was launched in the Brazilian market in 2009, where currently (March 2013) has more than 800k active users. The main limitation of MobileDeck solution is a lack of downloadable applications, since the SMS protocol for it only supplies content for embedded applications. MobileDeck channels are integrated seamlessly into GEMS ecosystem.

All GEMS supported applications can be searched, downloaded and installed through a Application Store feature, embedded in GEMS client component.

The GEMS system architecture is composed of four major components:

- GEMS Client: The mobile client is embedded software that hosts lightweight applications. It acts as a container for multiple lightweight applications. The Client stores all the data and code templates that define each lightweight Application.
- Cloud Platform: The GEMS Cloud is a server-side platform that:
  - Connects via SMS to a GEMS Client
  - Provides service building blocks to cloud-enable lightweight Applicationss
  - Computes the instructions sent to it by a GEMS Application and returns the result. Hosts a Service that runs on behalf of each type of Application.
  - Multiple instances can be deployed in different data centers
  - Highly Available and Highly Scalable
- Application Creation Toolkit: Create a new GEMS application, which also creates a corresponding Service within the cloud platform. The diagram in Figure 1 shows a high level view of both the Client on the Device and the entire Cloud Platform. The diagram uses the Application Store as an example to highlight the interactions that occur within the Cloud Platform components. The diagram also shows the relationships between Tools and Cloud Platform components.
- Service Building Blocks: The service building blocks represent independent transaction or services capabilities that are accessed by a service.

All those items will be examined in next sections of this paper.

IV. GEMS Client

The GEMS client platform consists of virtual machine (VM), a table of localized texts (TXT), a library of graphics (ICO), and a list of graphical layout templates (GUIT).

The VM interprets a sequence of instructions, which explain how to combine TXT items and ICO items (possibly with text sent from the server) in order to instantiate GUIT objects and define transitions between them based either on user-generated events (user input) or network-generated events (incoming SMSs). In other words, the VM implements a sort of a state machine where each state represents an instantiation of a GUIT with parameters from the TXT and the ICO tables and each transition between states is triggered by user- or network-generated events. Every GEMS app is a set of such instructions for the VM.

When launched from its home screen icon, the Client opens into an Application Manager View that includes an Application Store icon. The GEMS Application Store is part of the client. Lightweight Applications purchased by the mobile user will remain accessible via Icons within the Application Manager View 2.

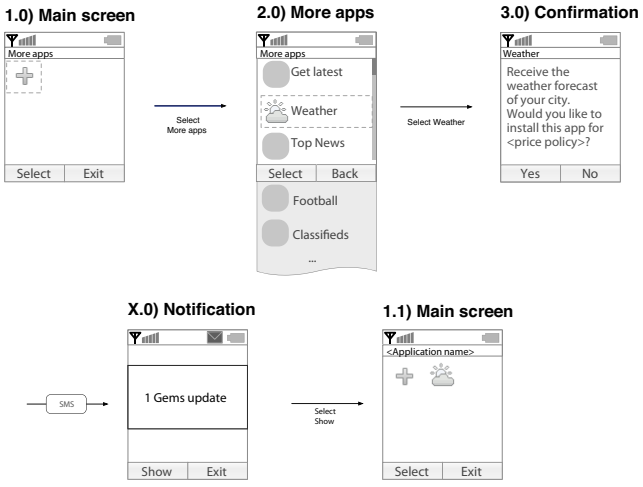

Fig. 2. GEMS Application store.

Based on that, GEMS application is a sequence of instructions written to perform a specific task with the GEMS VM. The sequence of instructions is defined by the designed GEMS communication protocol, which contains all definition about actions, GUI elements, and so on.

Every application on GEMS platform should be representable as an N-ary tree. The tree concept was used to better represent its behaviors, actions, and functionalities. Every node represents an element (not necessarily a GUI element), and every link is an action that leads to the next node. Polish notation is used in order to represent the tree in a text format. Every section of a tree is described with its children count, and content. More precisely, a GEMS tree node is represented textually by its number of children, node type (which element

is being described) and its parameters. That way, a slightly more realistic tree would look like Figure 4, which represents the screen flow illustrated in Figure 3.

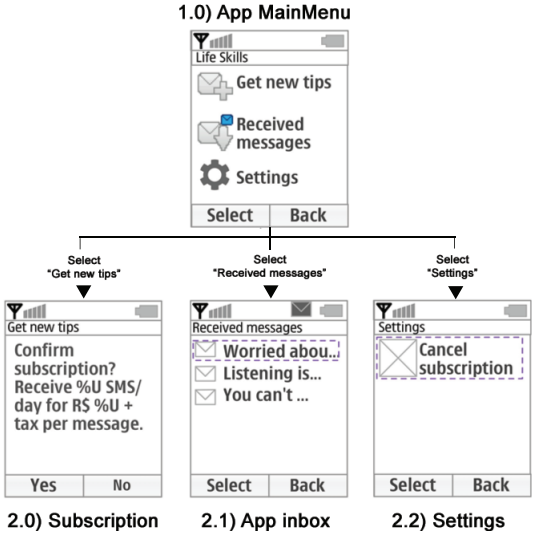

Fig. 3. A practical example of a GEMS app flow.

```
3|Life Skills|4|5|41||
    4|
        1|40|64|67|
            0|12|1||
        1|3|Get new tips|82||
            1|5|
                1|30|33|0,00||
                    0|10|SB#SU&3&1&4|41|
        2|27|Received messages|9||
            0|39|1|
            0|26|1|4||63|
        1|3|Settings|7||
            1|2|
                1|3|Cancel subscription|5||
                    1|5|
                        1|30|69|||
                            0|10|SB#US&3&233|41|
```

Fig. 4. GEMS protocol sample, which builds the flow illustrated in Figure 3. Note that, this string has line breaks for clarity purposes, the protocol don't support them.

Although GEMS has its own protocol, the GEMS VM was designed to interpret other protocols. To achieve it a new protocol interpreter must be previously integrated with GEMS VM. Then, multiples protocols interpreters can coexist in GEMS Client, enabling the client to be integrated with current actives SMS services, like Twitter SMS API [1].

V. Cloud Platform

The Cloud Platform is the server side system that enables a large number of services to be accessed by the user. This platform provides a logical infrastructure that allows GEMS developers and partners to build services and make them available to the users as GEMS ecosystems.

[1]http://twitter.com/

When a user performs a service request through the GEMS Client, an SMS message is sent to the Cloud Platform. The Cloud Platform processes this message, computes the service instruction contained in the message, stores data, saves business logs and commonly returns a SMS back as response to the GEMS Client. This is the main flow of data into the Cloud Platform.

Different capabilities are required by the services that run in GEMS. A Service Building Block (SBB) is developed for each required capability to provide specific functionalities that can be reused by the services. The SBBs exist to provide the infrastructure to run the ecosystems developed over GEMS. For instance, a news service requires that an advertisement campaign must be sent in background to GEMS Clients while the service is idle. This service implementation in the Cloud Platform would utilize an Advertisement SBB to support the news service with advertisement capabilities. *Billing*, *Advertisement*, *Application Inventory*, *Application Recommendation* and *Content Manager* are examples of SBBs.

Figure 3 presents the Cloud Platform context by the Appication Store service point of view. This service is responsible for maintain a list of GEMS Applications available to be downloaded, allow the users to search for applications, to buy applications and to remove applications from their installed list. In order to execute each functionality, the Application Store service rely on specifics SBBs to create richer user experience. For instance, the Application Store requests to the Application Inventory SBB to update users data with most recent users requests. It also access Application Recommendation SBB to deliver applications that might be from user interests. During users interaction with the Application Store services, advertisement targeted to specific users have been generated and are presented to users via Application Store. Apart from that, billing data for applications downloaded by users are constantly generated by the Application Store.

Given that the Application Store is the main business of this work, the Cloud Platform provides the resources to allow external developers and also GEMS developers to create and upload GEMS Applications to the Application Store. There are no logical limits in the number of applications made available by GEMS. As an example of applications that can be created we have: *Home Banking*, *News*, *Sports*, *Soap Operas*, etc. Each of those applications must be created through an Application Creation Toolkit.

After the deployment of an Application, an entirely new ecosystem is ready to be accessed by users. The users will use search and recommendation functions of the Application Store to find Applications according to their interests. After a GEMS Application download, the user is ready to enjoy its functionalities through the GEMS Client.

VI. Application Creation Toolkit

The Application Creation Toolkit (ACT) provides a friendly interface that allows users without technical programming background to create and deploy GEMS Applications on the Cloud Platform. In order to achieve this, the ACT will offer

a set of visual building blocks that the GEMS Application Developers will put together in order to mount an application that satisfy their business needs. The GEMS Application Developers are the users of the ACT that want to have their applications available on GEMS.

A set of *Application Creation Building Blocks* are provided as visual components that will be assembled to create a GEMS Application. They can be of the following types:

- *Container*: represents the main visual component that composes the application screen.
- *Item*: represents the information items presented inside the containers. Items can contain elements of the Action type.
- *Action*: represents components that trigger events on GEMS like messages, navigation flows and requests to the Cloud Platform.
- *Content Provider*: it represents how data of a given set of Application Creation Building Blocks are fulfilled. It can be of static or dynamic type.

All the available visual components must be present in the main screen of the ACT where the GEMS Application Developers can visualize them. They will be placed in a separate area of the user screen called *Component View*. The number of components and their configuration must reflect the actual GEMS Client version. Hints about the functionality of each component must be provided to the users before they actually decide to use it (when the user passes the mouse over the component, for instance). There must be a way to differentiate or grouping similar categories of components like Containers, Items, and Actions.

In addition to this view, it will also be present the *Application Creation Canvas* which is an area of the ACT where the application composition is made. It contains the whole set of building blocks, properties and component connections that will form a given application. This view, is the area that reflects the current status of the application composition. The GEMS Application Developers will see the available list of components and will be able to add those components to the canvas. Once in the canvas, the components can be moved and edited. Figure 5 presents a sketch of the ACT with the Component View and Application Creation Canvas.

Each component has specific set of properties that can be edited by the GEMS Application Developers in order to customize the applications. Validation rules for properties values must be applied as defined on each component specification. Properties view must be presented in a standard way for all components.

Connections between two components determine navigation flow rules between them. Connections between components can be of one of the following types:

- Adding a component of the type Item inside a compatible component of the type Container;
- Linking a component of the type Item to a component of the type Container;
- Linking a component of the type Action to a component of the type Container.

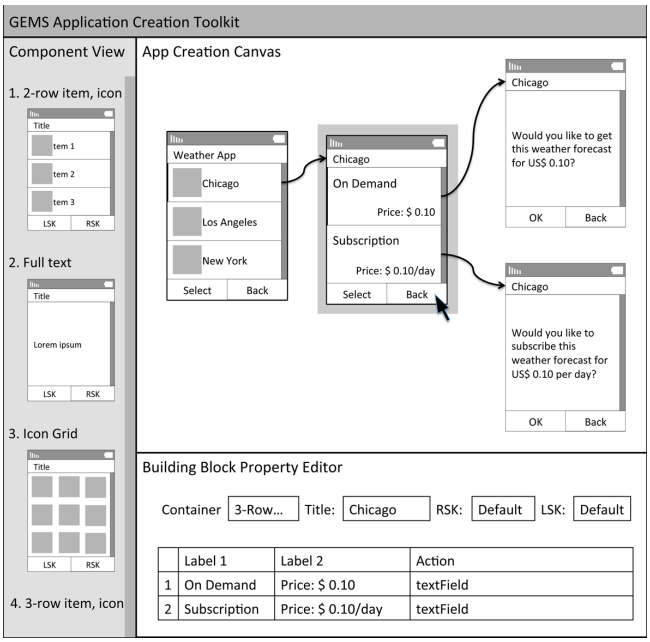

Fig. 5. Application Creation Tool presenting the Component View and the Application Creation Canvas.

No component without any in/out connection (orphan components) can exist on the canvas at application saving or updating time. This will invalidate the GEMS Application configuration.

The GEMS Application Developers will be able to save the work in progress for further development. It will be also possible to reload a saved GEMS Application to edit on the application. After the work is completed, the developers will be able to publish their applications to the Cloud Platform. In doing so, GEMS Application metadata is created into Cloud Platform Application database. For brand new GEMS Applications, they will be immediately available for recommendations and search in the Application Store service. For already published applications, the platform must retrieve the entire database of users that have already subscribed for this application and push the new configuration to their phones.

As a part of the ACT, there must be a module called GEMS Application Emulator. This is a widget that enable the developers to see an application behaviour during development. This module will provide the means to emulate GEMS Applications behaviour as similar as possible of what would be viewed in the mobile phone.

VII. Service Building Blocks (SBB)

The Cloud Platform offers a set of SBBs that can be reused by the GEMS Application developers while they are building an application. During the development, some functionalities might be required by the developer to finish the application. Some examples of functionalities are: security, advertisemnt, billing, recommendation, etc.

For each new type of service capability demanded to build GEMS Applications, a new SBB must be implemented. As

the goal of GEMS is to support ecosystems to be created in a dynamic mobile environment, new SBBs are going to be implemented to improve the support to GEMS Applications developers.

In this work we are going to present five fundamental SBBs. Notice that it is not in the scope of this paper to perform a deep investigation about the functionalities present in each SBB.

- *Content*: Provides communication with external content providers. Each GEMS Application that processes the service business in a external content provider servers must set properties of this component during the application development.
- *Billing*: Required by applications to manage the billing information about themselves in the GEMS usage.
- *Advertisement*: Provides a set of capabilities to insert advertisement in applications.
- *Application Inventory*: Used by services that requires data of the GEMS Applications. For instance, applications downloaded by a user, application properties and application metadata are example of the data returned by this SBB.
- *Security*: Provides several levels of security for the messages being transferred during the user interaction with applications.

The GEMS Application interactions with SBBs are defined by the developer. During the application development the developer has the possibility to integrate the SBB functionalities in a way that makes sense to the application context. In the ACT, the SBB's functionalities are available as actions. The developer has the possibility to select the required SBB, select the required action and integrate it in the application accordingly.

VIII. GEMS Ecosystem

GEMS can be understood as an entire ecosystem in a sense that it contains more dimensions than the technical infrastructure per se (mobile and cloud). The architecture based on SBB together with the tools and the numerous combinations of GUI components that can be accessed on the client side enables services to be quickly rolled out, also facilitating their operation and the relations with business partners that will provide apps, service transactions, and content.

Figure 6 shows a high level view of the possible interactions between the GEMS client on the device and the entire cloud platform, including the system tools.

The diagram shows a hypothetical flow (A, in a dotted line) taking the App Store as an starting point and exemplifying how the App Store Service can interact with other SBBs, resulting in a series of possible data exchanges (advertisement, billing, etc), which will result on other possible interactions with the tools. Eventually the App Store service would send the corresponding protocol to install (mount) the chosen app on the client.

If, on a second example, Football news is taken as a starting point (B; full, gray line) it is possible to predict that the mobile app will interact with its server-side counterpart. This service will then access the Content SBB, which will be exchanging data with the content manager tool.

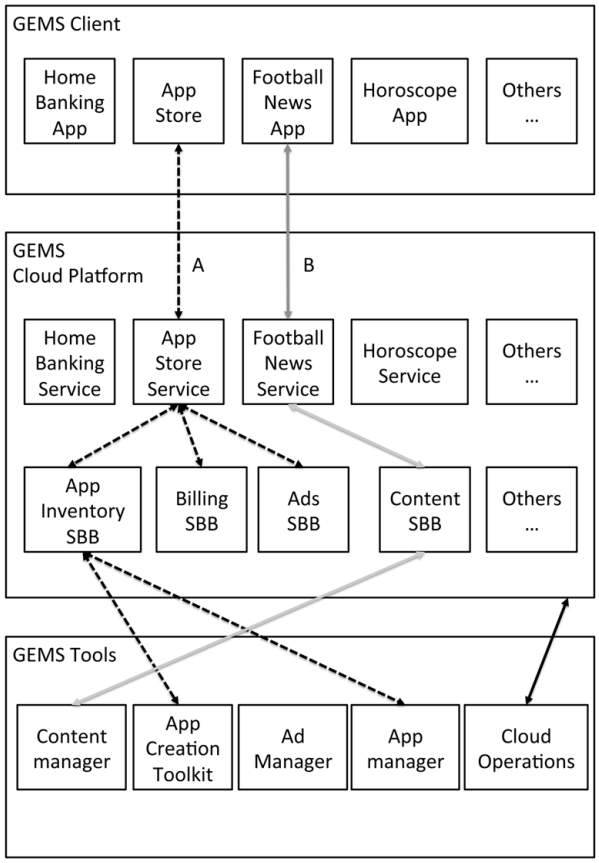

Fig. 6. High level view of component interaction.

IX. Experiment

This section describes a preliminary experiment carried out in order to investigate the GEMS initial perception and its services, and observe the comprehension of the icons, text messages and forms of interaction.

In order to evaluate the GEMS concept, the embedded GEMS client will be used by different mobile phone users. All these users are considered as a potential customer on our evaluation. 929 people were considered in this experiment and their profile is described as follows:

- They were 399(43%) men and 530(57%) women;
- User age was between 18 and 40 years old, with an average age of 28. This represents the major concentration of mobile phone users in Brazil;
- Most of the users had high-school level of education (71%);
- Social classes where distributed among upper-medium (19%), lower-medium (79%) and low (2%);
- Income was up to 2 Brazilian minimum wages (about US$ 622);

- They were not mobile internet users and had low internet usage on computers;
- The most used services of the participants in mobile device are: voice calls, text messages and camera;
- We did not consider real users of the mobile phone under observation. It avoids different experience levels of handling the mobile phone.

A large user base was chosen for server load and traffic behavior testing, and those results are beyond the scope of this paper.

A brand new low-end feature phone (a US$ 50 Nokia X1-00 device) with the GEMS client embedded and a prepaid SIM card were delivered for each experiment's participant, free of charge. Instructions were provided with the package, covering GEMS basic usage.

Each participant was able to use the device for two months. The participant was responsible for provide funds to the SIM card. This experiment intended to simulate a real GEMS system usage. After that, the devices were given back and the users' impressions were collected by an interviewing team.

During the experiment, the participant could access the MobileDeck's channels[14], the Nokia Life[2] service, and Twitter service over SMS.

MobileDeck channels are simple content providers grouped by subject such as: world news, Brazilian news, soap opera summaries, horoscope, weather, religion, jokes, etc. Those channels could be accessed on-demand (paid per message) or through an experimental subscription model of 3 days with 3 messages per day. An option to cancel a subscription was also available at any moment. No channel was preloaded, leaving the user free to search for desired channels through a keyword search or through a recommendation service on the Application Store embedded in GEMS client.

Nokia Life is an on-going service that provides information on several communication skills like: developing interpersonal skills, building self-confidence and improving financial literacy. Just as MobileDeck channels, users could ask for content on-demand or subscribe for 3 days.

Twitter application is a front-end for the existing Twitter SMS service[18]. It does not use any GEMS server-side component, but the client is built entirely with GEMS template solution, just as MobileDeck channels and Life Skills application.

X. Results

In order to analyze acceptance and relevance of GEMS system, an external research group was responsible for interviewing and compiling user's opinions.

The participant responded to several affirmation with a numerical value ranging form 0 to 10. A score from 0 to 5 meant "Disagree", 6 to 8 meant "Fairly agree", and 9 to 10 meant "Strongly agree". A final question left space for the user to point out positive and negative aspects of GEMS usage during the experiment.

[2]http://www.nokia.com/in-en/life/

The collected results are summarized in Table I, Table II, Table III and Table IV.

Score	Strongly agree	Fairly agree	Disagree
Rating	57%	26%	17%

TABLE I
OVERALL EXPERIENCE ("I FELT COMFORTABLE WITH THE OVERALL USAGE OF GEMS.")

Score	Strongly agree	Fairly agree	Disagree
Rating	54%	28%	18%

TABLE II
EASE OF USE ("I FELT COMFORTABLE TO DO THINGS FOR THE FIRST TIME, SPECIALLY WITHOUT HAVING TO LOOK FOR THE INSTRUCTIONS BOOKLET.")

Score	Strongly agree	Fairly agree	Disagree
Rating	57%	24%	19%

TABLE III
PURCHASE INTENTION (IF GEMS WERE RELEASE ON MARKET. I WOULD PAY FOR ITS SERVICES)

Score	Strongly agree	Fairly agree	Disagree
Rating	46%	26%	28%

TABLE IV
SERVICE COSTS FAIRNESS (IN TERMS OF COST, GEMS IS APPROPRIATE WITH MY OVERALL LIFE COST, AS WELL AS SIMILAR SERVICES)

Positive aspects mentioned:

- Relevant content;
- Service agility, even on remote areas;
- Gathering of information on a single application;
- Receiving subscribed information is better than look for it everyday;
- Good for people with no time or availability of internet.

Negative aspects mentioned:

- A lot of content was not delivered, even though was billed;
- Twitter application doesn't have all features;
- High daily cost compared to cheap internet access out of the mobile phone;
- Applications' English names don't help;

MobileDeck channels had good perception regarding relevance and earned value, but users complained about convoluted usability. Further studies should be done to improve that aspect. The most used MobileDeck channels are shown in Table V.

Twitter application got a very positive feedback, regarding ease of use and satisfaction about the features presented. Users praised the possibility to receive a lot of Twitter content, following news websites' twitter accounts, with no cost.

Channel	Rating
Soap opera	26%
Football	20%
Psalms	13%
Horoscope	9%
Weather	8%

TABLE V
FAVORITE MOBILEDECK'S CHANNELS

Life Skills got a very bad perception, due to lack of content and technical problems with the service that made it unavailable to a great amount of users. A quick study on the matter revealed that one of the carriers was blocking Life Skills messages on their SMS broker.

The Table VI shows the favorite application among the ones available on the experiment.

Application	MobileDeck	Twitter	Life Skills
Rating	46%	45%	9%

TABLE VI
FAVORITE APPLICATION

The results lead us to believe that GEMS is relevant and suitable to provide information access for people with low internet availability. We also believe that the user experience is straightforward, easy, but with some room for improvement.

The billing system also needs improvement. Nowadays, very cheap, focused SMS services are available for users, and GEMS current model is perceived as an expensive one. Lower monthly and weekly fees should be available in the future, so the system could not only be relevant, but also attractive for lower-income users. In future experiments, a more robust and reliable SMS broker should be use, since message losses and improper billing had impact on users' perception on the overall experience.

XI. CONCLUSION

There is a huge number of people that do not have internet access on their mobile phones . Moreover, great part of these users have access to feature phones with less processing and memory capacity than smartphones. This work presented GEMS: a mobile-based system that allows the creation and distribution of content as services that reach users through SMS.

In this work, we presented a mobile client platform that supports the installation of mobile applications that fits in a small number of SMS. In our approach, the mobile applications are services created by content providers using the presented App Creation Toolkit. The applications are built using a set of Service Building Blocks that provides capabilities like security, billing, advertisement, content delivery and applications metadata recovery. All the applications are made available in a app store that the users can interact through SMS in order to reach the applications that most fit their demands. Finally, we presented the Cloud Platform system that provides the logical infrastructure that allows GEMS developers and partners to build services and make them available to the users as GEMS ecosystems.

After carrying out a experiment to test all GEMS ecosystem in a real world scenario, we figured out that GEMS system indicates potential to solve the presented issue, bringing Internet services to low end devices. However, more detailed experiments must be performed with the purpose of confirming GEMS system adoption by target users.

REFERENCES

[1] *International telecommunication union*. [Online]. Available: http://www.itu.int/

[2] J. Harno, "Impact of 3g and beyond technology development and pricing on mobile data service provisioning, usage and diffusion," *Telemat. Inf.*, vol. 27, no. 3, pp. 269–282, Aug. 2010. [Online]. Available: http://dx.doi.org/10.1016/j.tele.2009.10.001

[3] E. Brewer, M. Demmer, M. Ho, R. J. Honicky, J. Pal, M. Plauche, and S. Surana, "The challenges of technology research for developing regions," *IEEE Pervasive Computing*, vol. 5, no. 2, pp. 15–23, Apr. 2006. [Online]. Available: http://dx.doi.org/10.1109/MPRV.2006.40

[4] J. Chen, "Re-architecting web and mobile information access for emerging regions," Ph.D. dissertation, 2011.

[5] *GSM Association Association*. [Online]. Available: www.gsma.com

[6] *Mobithinking global mobile statistics 2011*. [Online]. Available: http://mobithinking.com/mobile-marketing-tools/latest-mobile-stats/

[7] *M-pesa*. [Online]. Available: http://www.safaricom.co.ke

[8] *FrontlineSMS*. [Online]. Available: http://www.frontlinesms.com/

[9] *Gupshup*. [Online]. Available: http://gupshup.me/

[10] *AppShup - http://api.smsgupshup.com/*. [Online]. Available: http://api.smsgupshup.com/

[11] L. Wei-Chih, M. Tierney, J. Chen, F. Kazi, A. Hubard, J. G. Pasquel, L. Subramanian, and B. Rao, "Uju: Sms-based applications made easy," in *Proceedings of the First ACM Symposium on Computing for Development*, ser. ACM DEV '10. New York, NY, USA: ACM, 2010, pp. 16:1–16:11. [Online]. Available: http://doi.acm.org/10.1145/1926180.1926200

[12] K. Banks and E. Hersman, "Frontlinesms and ushahidi - a demo," in *Proc. Int Information and Communication Technologies and Development (ICTD) Conf*, 2009.

[13] *RapidSMS - http://www.rapidsms.org/*. [Online]. Available: http://www.rapidsms.org/

[14] *Reference deleted for double-blind review.*

[15] *Proceedings of the Eleventh Workshop on Mobile Computing Systems & Applications*. New York, NY, USA: ACM, 2010.

[16] *Reference deleted for double-blind review.*

[17] D. Marples and P. Kriens, "The open services gateway initiative: an introductory overview," *Comm. Mag.*, vol. 39, no. 12, pp. 110–114, Dec. 2001. [Online]. Available: http://dx.doi.org/10.1109/35.968820

[18] *Twitter SMS Commands*. [Online]. Available: https://support.twitter.com/groups/34-apps-sms-and-mobile/topics/153-twitter-via-sms/articles/14020-twitter-sms-commands