ITERATIVE REWARDS GENERATOR FOR MOBILE TELECOMMUNICATIONS OPERATORS

Marius Cristian URECHE Mircea Florin VAIDA Alin Gheorghe VOINA
Faculty of Electronics, Telecommunications and IT, Technical University of Cluj-Napoca, Romania
George Baritiu 26-28, 400027 Cluj-Napoca, Romania, Email: Cristian.Ureche@com.utcluj.ro

Abstract: Dealing with large amounts of data always was a delicate operation. Mobile telecommunications operators need high quality management systems to provide top-class services for their customers. The challenge is to assure data processing in a real-time or nearly real-time manner while facing the dynamic changes that are the heart of this field of engineering. We propose one targeted software architecture for iterative rewards generator enabling mobile telecommunications operators to grow their revenue by implementing iterative rewarding campaigns.

Keywords: *mobile telecommunications, rewards generator, intelligent network platforms, reward-based loyalty campaigns.*

I. INTRODUCTION

The field of mobile telecommunications has changed dramatically in the past 20 years. Without respect to the technologies involved, mobile telecommunications operators try to increase their revenue by means of coverage area, quality of services and marketing campaigns [1]. Reward-based loyalty campaigns are also a viable alternative as they provide a flexible opportunity for operators to keep in touch with customers' needs and expectations.

Loyalty campaigns are triggered using several strategies: top-up based, CDR based, SMS subscription. Top-up based strategy implies the application of a rewarding policy when the recharged amount is equal to or greater than a specific value. For instance, the mobile telecommunications operator would like to reward a subscriber if he makes a recharge of minimum 10$.

In the field of telecommunications, data retention generally refers to the storage of call detail records (CDR) [2] of telephony and IP detail record (IPDR) [3] of internet traffic by governments and commercial organizations. By means of CDR and IPDR analysis, rewarding mechanisms could be triggered. For instance, any subscriber that talks 50 minutes on network during a month would be granted 10 SMS.

SMS subscription is intensively used by operators having very poor rewarding tools usually on the emerging markets. However it is generally accepted to be a simple and successful technique to subscribe to a rewarding campaign. This is a unidirectional approach using simple techniques for subscribers' rewarding. This paper describes the issues and challenges that appears when staggered (iterative) rewards and bundles are intended to be applied.

II. REWARD-BASED LOYALTY CAMPAIGNS

Rewarding campaigns are generally recognized as a very powerful tool to maintain subscribers' loyalty and satisfaction [1]. Therefore, new strategies for assuring a long-term relationship with subscribers aroused. As a fact, iterative rewarding strategies, as well as bundles of atomic rewards were proposed by MTOs. *Bundle rewards* are defined as packages of elementary rewards that entitled customers are offered during loyalty campaigns.

One has to define iterative rewards and bundle rewards, which will be named from this time forth complex rewards, and then apply them to subscribers via Intelligent Network Platforms (INP). Usually, mobile telecommunications operators externalize these complex operations to specialized enterprises having high-level expertise in the field. Even so, the challenges are great. For instance, bundles need to be split up in elementary rewards (e.g. calls on net, calls of net, SMSs, GPRS / 3G traffic, airtime rebate etc.). These elementary rewards are usually generated in CSV files. But INPs are of great diversity. Some of them are capable of dealing with multiple types of rewards while others can manage only one single type (e.g. ring-back tone –RBT-platforms firstly released in 2002 that are very popular these days can deal with that special type of ring-back tone only).

Things tend to have the same complexity for iterative rewards. The need for confirmation after a rewarding operation is crucial for MTO's Operational and Management Systems (OMS). For taxation and CDR purposes, a reliable feedback is necessary. When rewards are iterative, confirmation should arrive before the next step of the staggered reward is applied. Therefore, rewarding mechanisms must be fast and sustainable.

It was a time when database engines did the entire work. But as business logic becomes more complex, the need for enhanced architectures emerged. Rewarding campaigns applications are no more just an SQL-scripting affair.
Today, rewards generating applications usually implies the following steps:

a) *Defining the purposes of the loyalty campaigns* – the MTO should clearly identify the reasons for which a particular campaign is created. Generally, MTOs wish to grow their Average Revenue per User (ARPU) or to persuade subscribers to stay with them and hence to mitigate churn rate.

Manuscript received July 28, 2010; revised October 30, 2010

Proper triggers should be identified for every one of the types of campaigns above. For instance, when an ARPU growing campaign is conceived, triggers like top-up level (the amount one prepaid subscriber recharges the mobile phone account with) should be considered. The telecom industry faces the largest churn rates. To properly manage the lifetime value of their customers, loyalty campaigns are of great interest for MTOs.

b) *Segmentation of subscribers* – it divides subscribers into groups based on the underlying needs or characteristics. Segmentation helps to create valuable propositions that uniquely serve the target client's cluster. As large amounts of data are implied, data mining techniques are extensively used. A. Pryke and R. Beale presented in [4] a data mining system used to mine CDR data in order to obtain usage patterns for companies with large numbers of calls.

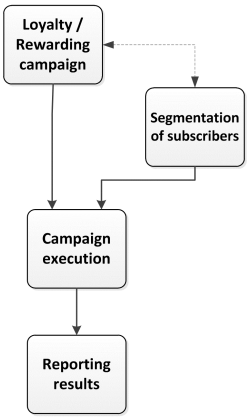

Figure 1. Steps in rewarding / loyalty campaign

c) *Establishing the rewarding strategy* – MTOs have to decide what kind of rewards the campaign will be dealing with. MTOs in emerging countries extensively use simple type of atomic rewards (e.g. 10 SMSs, 1$ discount), while MTOs playing in more sophisticated markets try to catch the eye of subscribers with iterative rewards (e.g. 20 minutes on net every week, for 5 weeks) and bundles as well (e.g. 10 MMSs and 5 MB of GPRS traffic).

d) *Reporting campaign results* – bottom-line results need to be analyzed by MTOs to adapt strategies accordingly. Hence, reporting tools are used to sum up the obtained results. Common results to include in campaigns' reports are:

- The number of subscribers that reacted to the campaign with regard to the total number of targeted subscribers,
- What kind of rewarding campaigns are of great interest for subscribers: top-up based, CDR based etc,
- ARPU evolution,
- The number of subscribers that extended their contract (if a loyalty campaign is implied).

Rewarding campaigns solutions are extensively developed towards an automatically manner of execution. MTOs are constantly demanding new features and only scalable architectures could meet that demands.

III. PROPOSED ARCHITECTURE

Working in the field of telecommunications, we understand the issues that usually arise in the field of rewarding campaigns. We propose a software architecture that tries to minimize bottlenecks and assures a modular conception for the application itself.

Figure 2 presents the global perspective on the entire architecture. We will explain the complete architecture and then we'll focus on Iterative Rewards Application (IRA).

There are four major parts that our architecture implies:

1. *Loyalty campaigns application (LCA)* is in charge of defining the campaigns, performing the segmentation of subscribers, defining the triggers that rewarding mechanisms will be started by, managing the states that the campaign will pass through (e.g. defined, started, executed).

LCA Graphical User Interface (LCA GUI) is provided for setting up rewarding campaign characteristics. MTO employees use this GUI to accurately define campaigns and to supervise the campaign states.

LCA Database (LCA DB) stores information related to subscribers, campaigns and rewards offered per campaign. It makes use of multiple threads paradigm to "watch" subscribers' behavior and to initiate the rewarding procedure.

2. *Iterative rewards application (IRA)* is the core of the architecture. It implements the features that MTOs are demanding these days: a variety of rewards to be applied based on a predefined schedule.

IRA Configuration GUI enables MTOs' engineers to set-up the IRA Windows Service. It also offers the possibility to defined iterative rewards. We propose an iterative reward entity comprising four elements:

Element 1: Atomic reward: the MTO defines a list of predefined elementary rewards that the rewarding applications is entitled to apply: calls on network / off network, SMS, MMS, RBT, GPRS traffic, airtime rebate / deduction, lifetime extension etc.

Element 2: Cyclicity: the rewards can be applied daily, weekly, monthly or in predefined days.

Element 3: Number of iterations: the number of iterations could be finite or not, but usually the number of iterations is finite to better report bottom-line results.

Element 4: Partitioning aspects: for certain atomic rewards (SMS, MMS, calls), it is valuable to have the possibility to split the total amount rewarded in bunches. By default, these bunches are equal but one can find useful to specify a particular configuration for partitioning.

Example: Let's suppose one subscriber is rewarded 100 MB of global extra internet traffic that should be applied once a month for 4 months. By default, the user will be granted 25 MB of internet traffic each month, but there could be a need to apply the reward in an arithmetic progression manner, namely 10 MB the first month, 20 MB the second, 30 MB the third and finally, 40 MB the forth month.

IRA Windows Service is the generator kernel. It is triggered daily by the LCA by means of an SQL Server Scalar-valued Function. The IRA Windows Service will be described in depth later in this chapter.

The iterative rewards having been generated by the IRA Windows Service, it's time to "move" them towards the MTO platforms. We identified 2 approaches to successfully achieve this operation.

3. One option is to generate File Transfer Access and Management (FTAM) protocol compatible files in a comma separated value (CVS) format and upload them to the operator using an SFTP – FTAM gateway. This is the

preferred approach when dealing with legacy systems that usually are incapable of communicating using a standardized communication protocol.

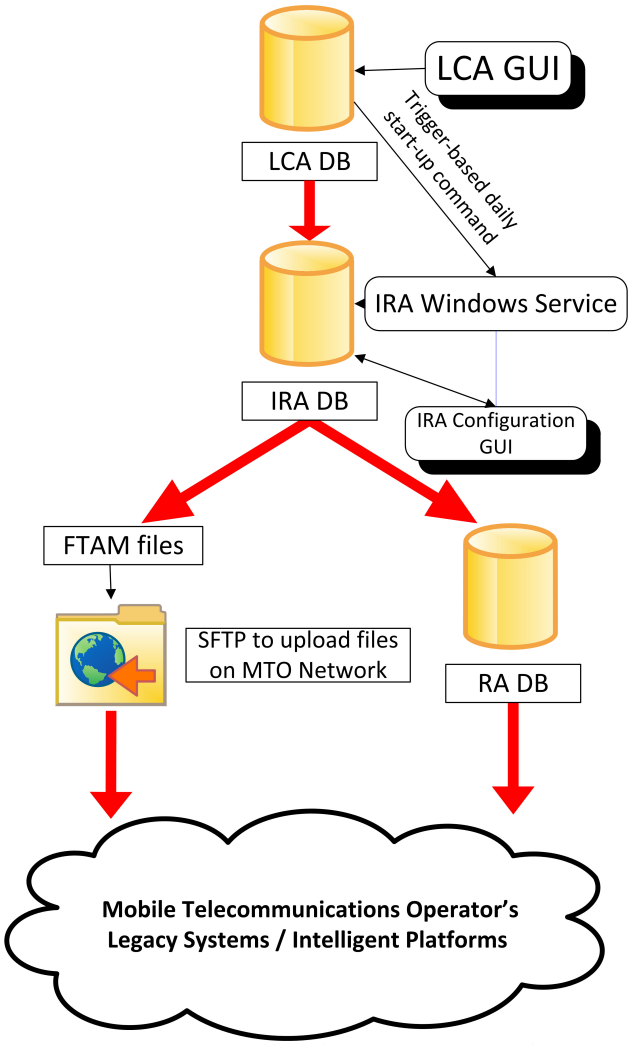

Figure 2. Proposed architecture for rewarding application

4. For Intelligent Network platforms, the Rewarding Application (RA) could be used. It stores processed rewards and uses a Rewarding Application Service (RAS) to negotiate a communication with each IN Platform and transfer the rewards directly without using any CSV file.

These options should be taken into account as our architecture is supposed to assure retro-compatibility with legacy systems that MTOs are still using.

The second part of this chapter will thoroughly describe IRA level of the architecture presented above.

As the business logic of loyalty campaigns tends to be more and more complex, working with SQL scripts exclusively became a non-productive manner of creating software in the field of telecommunications. Dynamic scripts in SQL are often a good solution, but too much dynamic SQL can alter significantly the performance. Therefore, we tried to move the business logic towards services and to implement a kind of "Separation of Concerns".

IRA Windows Service is the heart of our architecture. As shown in Figure 2, it is daily triggered by the LCA to process rewards accumulated during 24 hours.

Figure 3 presents the IRA WS – centric perspective. Raw rewards are stored in the IRA DB before processing. IRA WS "learns" how to process row rewards by looking–up for iterative rewards entities that are referenced by each raw reward. As explained before in this chapter, iterative rewards entities consist of 4 elements (atomic reward, cyclicity, number of iterations and partitioning aspects).

MTOs usually want to implement a hold-out policy to avoid situations when customers are rewarded many times in multiple campaigns during 24 hours. For instance, if the hold-out policy states that the maximum number of rewards a subscriber can receive each day is 3, than IRA Windows Service should apply only top three most important rewards and all the other rewards intended to be applied to that subscriber are ignored.

The criterion to decide between raw rewards assigned to the same subscriber is a priority field that each raw reward has. It could be a global value per campaign (as one subscriber can be rewarded only one time per cycle per campaign) or a value calculated taking into account multiple aspects: the moment in time when the raw reward was triggered, the atomic reward effectively to be applied in an iterative manner etc. Some MTOs consider that it's worth applying SMS bonuses instead of free calls when defining hold-out policies. Following the same reasoning mechanism, applying a lifetime extension would be more important than offering for free a ring-back tone.

As the volume of data to be processed each day is expected to be huge (tens of millions of rewards per day), we propose the batch processing technique to be used. The algorithm has the following steps:

Step1: once a day after IRA WS is triggered, raw rewards are inspected and a decision is made concerning hold-out policy. Non-eligible rewards with regard to the hold-out policy are removed from IRA DB and a log mechanism [8] is used to track them.

Step2: A batch of eligible raw rewards is extracted from IRA DB. Every raw reward contains the following essential information:

MSISDN	Iterative reward ID	Priority	Total amount	Expiry period (days)
+40723555666	7	2	100	30

Table 1. Raw reward essential information

Step 3: A look-up is performed by IRA WS to retrieve iterative reward definition (atomic reward, cyclicity, number of iterations and partitioning aspects).

Iterative reward ID	Atomic reward	Cyclicity	Iterations number	Partitioning aspects
7	SMS	weekly	3	Apply reward in equal bunches

Table 2. Example of iterative reward entity definition

Step 4: IRA WS will compute the list of dates when bunches will be applied starting from the current day. If the example above is considered and if start date is July 4, then the list will contain the following 3 dates: July 4, July 11 and July 18.

Step 5: Partitioning aspects are considered when the total amount is split in bunches. We'll take into consideration the

example above. The total amount of 100 SMSs should be applied in 3 equal bunches. But as non-integer values for the number of SMSs make no sense, IRA WS will decide that the values to be applied iteratively should be 33, 33 and 34.

Step 6: Having obtained the dates and the values for bunches, we have effectively obtained the processed reward (see Figure 3). If FTAM protocol is used, then one file per campaign per atomic reward is generated for each date.

For instance, IRA WS will append one record for each of the iterations considered in the example above. The FTAM file structure is particular for each platform, but usually, each record includes the MSISDN of the subscriber, the amount and the expiry date.

Example:
File name: SMS_FTAM_20100704.IN
+40723555666,33,2010-08-03
File name: SMS_FTAM_20100711.IN
+40723555666,33,2010-08-10
File name: SMS_FTAM_20100718.IN
+40723555666,34,2010-08-17

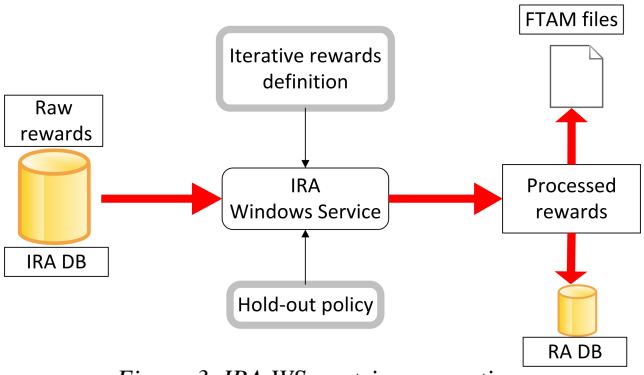

Figure 3. IRA WS-centric perspective

If RA Service is used, then FTAM files are not generated and processed rewards are stored in RA DB.

Every day, IRA WS will upload FTAM files generated for that day using a SFTP – FTAM Gateway solution. If RA Service is in use, then it will extract the processed rewards to be applied each day from RA DB, initiate a communication link with INP and transfer the rewards.

IV. IMPLEMENTATION, INTEGRATION AND RESULTS

The concepts presented above were concretized in a software application that is in the site integration test (SIT) phase at this moment (July 2010). MTO Maxis Malaysia accepted to test the proposed solution. We have configured 10 types of elementary rewards (including calls, SMSs, MMSs, internet traffic, lifetime extension, lifetime reduction, airtime rebate).

Our solution integrates with Matrix as LCA and InTeleStage Profiler working as segmentation tool, both provided by Business Logic Systems. Data are stored using Storage Area Network (SAN) architecture for security reasons and for lowering Total Cost of Ownership (TCO). Raw rewards are processed daily at noon. There are more than 1 million rewards to be processed each day. FTAM files are used for storing and transferring rewards as Maxis has legacy platforms for applying rewards and a RA Service-like solution was not purchased by the MTO.

The installed solution works well with Maxis' infrastructure. There are minor issues to be resolved regarding feedback for certain types of rewards (e.g. lifetime reduction), but we envisage the migration (respecting the general trend of MTOs) towards an all-IP network.

A hold-out policy was requested by Maxis. The maximum number of rewards a user might be applied daily is two.

Table 2 presents a real world example. Let's suppose that the TableRawRewards in IRA DB contains the following rows:

Raw Reward Id	MSISDN	Priority	Hold Out Policy
1	40722123456	23	4
2	40722123456	4	3
3	**40722123456**	**1**	**1**
4	**40722123456**	**4**	**2**
5	**40722334455**	**5**	**2**
6	40722334455	6	3
7	40722334455	24	4
8	**40722334455**	**3**	**1**

Table 2. Raw rewards that need a hold-out policy

By applying a hold-out policy, our application assures that for each MSISDN, only top two raw rewards with regard to the minimal priority will be processed (the bolded ones in Table 2). All the others will be tracked into a log table and then removed from TableRawRewards.

An elegant implementation for hold-out policy is presented below:

```
1  USE [ira_db];
2  UPDATE dbo.TableRawRewards
3  SET HoldOutPolicy = ho.HoldOutValue
4  FROM dbo.TableRawRewards trr
5  INNER JOIN
6  (
7  SELECT
8      RawRewardId,
9      ROW_NUMBER() OVER
10     (
11     PARTITION BY MSISDN ORDER BY
12     Priority
13     )
14     AS HoldOutValue
15     FROM dbo.TableRawRewards
16     )ho
17 ON trr.RawRewardId = ho.RawRewardId
18 GO;
```

We are using IRA DB (line 1) to update column HoldOutPolicy in TableRawRewards from default database schema dbo (lines 2, 3).

Calculating the hold-out value is the key point of this Sql query. The reason for performing this calculation is that Priority column might have different values ranging from 1 to 100. IRA WS knows the maximum number of rewards / MSISDN / day, but it is unaware of the priorities raw rewards could have. For instance, taking into account MSISDN 40722334455 from Table 2, even though the maximum number of rewards / MSISDN / day is 2, the

rewards that match the hold-out policy have values of 3 and 5 for the priority field.

For calculating the hold-out value we used a partitioning strategy (lines 7-16). Rewards are grouped by MSISDN, and then ordered by Priority. Then, an inner join operation with the nested selection is performed for updating HoldOutPolicy field.

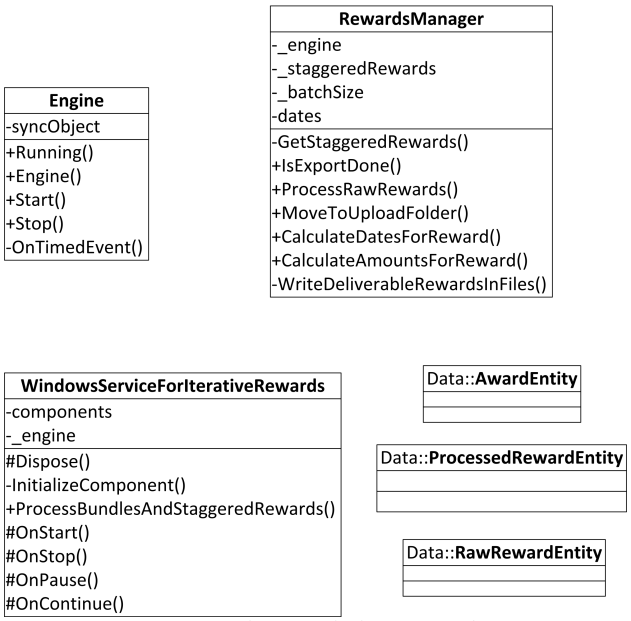

Figure 4. Fundamental classes used in our implementation

The hold-out values are calculated once a day, at noon, for all the rewards triggered in the last 24 hours. There are a couple of minutes LCA is not sending rewards anymore towards IRA DB and that is the moment hold-out values are calculated. After that, IRA DB accumulates new rewards to be processed the next day at noon. IRA WS knows these raw rewards should not be processed as they have no hold-out values calculated.

Figure 4 presents the most important classes we used in our implementation. We'll focus on methods rather than properties as the functional behavior is of great concern. The Windows Service itself is represented by the class WindowsServiceForIterativeRewards. It could be installed and then run as any other Windows Service. In fact, it is a kind of buffer between the Windows Server Operating System and the Engine class. IRA WS instantiate the Engine class which runs on a repetitive basis. The Engine has access to the methods exposed by the RewardsManager class who is in charge of manipulating raw rewards to produce processed rewards. We have modeled the rewards in classes RawRewardEntity and ProcessedRewardEntity, both inheriting AwardEntity.

The Engine class uses a timer to run repetitively:

```
public Engine()
{
timer = new System.Timers.Timer();
timer.AutoReset = true;

// Hook up the Elapsed event for the timer.
timer.Elapsed += OnTimedEvent;
}
```

This timer is set up by the time the Windows Service presented above instantiates and starts the engine.

```
public void Start()
{
   timer.Interval = interval * 1000;
   timer.Start();
}
```

IRA WS was implemented in C# as a Windows Service. The application targets .NET Framework 3.5. It looks for a start message coming from LCA and once started runs on a repetitive basis until the end of the day:

```
private void OnTimedEvent(object source,
ElapsedEventArgs e)
{
   if (rewardsManager.IsExportDone())
   {
   //Phase 1
   rewardsManager.ProcessRawRewards();
   //Phase 2
   rewardsManager.MoveToUploadFolder();
   }
}
```

Each iteration has two phases:

Phase 1: extract a batch of raw rewards that respect the hold-out policy implemented above in this chapter, perform the business logic and store them in FTAM compatible files or in RA DB.

```
public void ProcessRawRewards()
{
   IList<RawRewardEntity> rawRewards;

   rawRewards = rewardsManager.
   GetRawRewards( _batchSize,
   _maxCampaignsPerUser);

   foreach (var r in rawRewards)
   {
   dates = CalculateDatesForReward(
   rawReward, staggeredReward);

   List<double> amounts =
   CalculateAmountsForReward(rawReward,
   staggeredReward);

   CreateProcessedReward(dates, amounts);
   }
   WriteDeliverableRewardsInFiles();
}
```

Phase 2: look for FTAM files that must be transferred to INP via SFTP. Files are moved in the source directory of the SFTP module. The SFTP module queries repetitively the source folder and transfers the files to the operator's platform.

Rewards are correctly applied with lags varying in between 30 and 300 seconds (because of the FTAM protocol utilization), but these are acceptable values for MTOs.

V. CONCLUSIONS

High-level software in telecommunications has the particularity to deal with large amounts of data. MTOs want to execute multiple campaigns each day to study customers' behavior and of course, to increase ARPU.

We have seen that the dynamic of cellular

communications' industry made dynamic SQL a good approach, but nowadays, as the number of subscribers increase and campaign's business logic tends to be more and more complex *out-of DB processing* should be considered for reliability and maintenance reasons, as far as for automatic execution.

We have presented a generic architecture to be used in telecommunications software applications and we've focused on iterative rewards generation. The architecture is modular and scalable as IRA WS and IRA DB can be extended or exchanged with other functionalities: bundles processing, lucky draws etc.

Site-integration tests proved that the software implementation for the architecture presented in this article is capable of improving MTOs ability to create rewarding / loyalty campaigns in an automatic manner.

ACKOWLEDGEMENT

This research was partially supported by PN2 IDEI project, no. 1083 from Romanian National Authority for Scientific Research (2010).

REFERENCES

[1] T.J.Gerpott, W. Rams, A.Schindler, "Customer retention, loyalty and satisfaction in the German mobile cellular telecommunications market", *Telecommunications Policy*, vol. 25, pp. 249-269, 2001.

[2] "The Big Max: Applying Business Intelligence to maximize network revenue*", Billing and OSS World*, 2008. [Online]. Available: http://www.billingworld.com/ebooks [Accessed: July 1, 2010].

[3] M.A. Bihina Bella, J.H.P. Ellof, M.S. Olivier, "Using the Internet Protocol Detail Record standard for Next-Generation Network billing and fraud detection", Pretoria, South Africa, 2005. [Online]. Available http://mo.co.za/ [Accessed: July 1, 2010]

[4] A. Pryke, R. Beale, "Interactive Comprehensible Data Mining", *Ambient Intelligence for Scientific Discovery*, Springer Berlin, vol. 3345, pp.48-65, 2005.

[5] IETF RFC 1415, "FTP – FTAM gateway specification", [Online]. Available http://tools.ietf.org/html/rfc1415 [Accessed: July 1, 2010]

[6] M.C. Ureche, M.F. Vaida, M. Cremene, "A study of workflow programming technologies and features for dynamic adaptation", D*istributed environments. Adaptability, semantics and security issues,* pp.27-34, 2009.

[7] P. Nielsen, M. White, U. Parui, Microsoft Sql Server 2008 Bible, Wiley Publishing, 2009, ISBN 978-0-470-25704-3

[8] S.S. Dragos, M.F. Vaida, "Centralized Logging of Multi Tier Applications over Web Services with NLog Custom Targets", 2010 IEEE International Conference on Automation, Quality and Testing, Robotics (*AQTR) - Theta 17 - Cluj-Napoca*, 28-30 May, 2010 IEEE Catalog Number CFP10AQT-PRT(CDR), pp. 240-244, ISBN 978-1-4244-6722-8(6723-5).

[8] C. Peiris, D. Mulder, S. Cicoria, A. Bahree, N. Pathak, "Pro WCF, Practival Microsoft SOA Implementation", Apress, 2007, ISBN 978-1-59059-702-6.

[9] P. Delias, K. Ntalianis, A. Doulamis, N. Matsatsinis, "Automating Marketing Campaign Management Through an Agent-based Workflow Management System", *13th WSEAS International Conference on Computers*, Rhodes, Greece, 2010.

[10] C. Hesselman, C. Giannelli, "Adding value to the network: Exploring the Software as a Service and Platform as a Service Models for Mobile Operators", *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering,* vol. 12, pp. 13-22, ISBN 978-3-642-03568-5, 2009.

[11] S.L. Chan, V.H. Ip, V. Cho, "A model for predicting customer value from perspectives of product attractiveness and marketing strategy", *Expert Systems with Applications*, vol. 37, pp. 1207-1215, 2010.