Monitoring system and alarms visualization
Abstract

Along with the popularization of the SMS (Short Message Service) as a communication tool among big companies such as flight and train companies, post offices and banks, a new branch started to take place in the mobile communication industry: bulk-SMS business. The companies send the SMS to their costumers through an aggregator, which in turn send the SMS further to the customers’ mobile phone. The aggregator consequently needs to monitor the SMS traffic constantly so they can assure good quality of their services. In case of an interruption in the system, their support group needs to be notified directly, so the problem can be fixed straightaway. The aim of this project was to develop an application integrated to the intranet of a company that works in the business of bulk-SMS (CLX Networks), in order to replace the current monitoring system, called OpenNMS. The system was developed using Zend Framework (PHP) and MySQL database.

The project resulted in an application that allows the administrator to configure alarms and notifications and to add clients and suppliers accounts to the monitoring system. Whereas the interaction between the user administrator and the system is already implemented, the connection between the system and the CDR- and RR-database is still under development.
Preface

This thesis is part of the Web Development program 120 hec at Linnaeus University in Kalmar. Monitoring system was a degree project suggestion from CLX Networks, an aggregator in Bulk-SMS business in Sweden.

I would like to thank the whole staff of CLX networks office in Kalmar, for the support they gave me and for the opportunity to work with them. I would also like to thank my tutor Daniel Toll for the valuables tips and ideas.

Kalmar, 2011-05-25

Michelle Leite Santana
## Table of Contents

Abstract .................................................................................................................. I  
Preface .................................................................................................................. II  
1. **Introduction** ................................................................................................. 1  
2. **Background** ............................................................................................... 2  
   2.1 CLX Networks: monitoring system .............................................................. 3  
   2.2 Requirements .............................................................................................. 7  
   2.3 Problem description ................................................................................... 7  
   2.4 Delimitation ............................................................................................... 7  
3. **Method** ....................................................................................................... 9  
   3.1 Work Method ............................................................................................. 9  
      3.1.1 Unified Process .................................................................................. 9  
      3.1.2 Documentation ................................................................................. 9  
   3.2 Development Method ............................................................................... 10  
      3.2.1 Development Environment ............................................................... 10  
      3.2.2 Implementation Architecture ......................................................... 10  
      3.2.3 Testing ............................................................................................ 11  
      3.2.4 Database .......................................................................................... 12  
      3.2.5 RRD-Tool ....................................................................................... 12  
   3.3 Method discussion .................................................................................... 13  
      3.3.1 Zend framework ............................................................................... 13  
      3.3.2 Zend framework problems ............................................................... 14  
      3.3.3 Testing ............................................................................................ 15  
   3.4 Implementation ......................................................................................... 16  
      3.4.1 Work Method .................................................................................... 16  
      3.4.2 Development Method .................................................................... 16  
      3.4.3 Database .......................................................................................... 17  
      3.4.4 Subversion ....................................................................................... 17  
4. **Results** ....................................................................................................... 18  
   4.1.1 Surveillance .......................................................................................... 18
1. Introduction

A monitoring system is an application or a hardware that constantly monitors the performance of a computer network in order to detect failures or slow down occurrences. When complications are detected, the system dispatches alarms such as email and/or Short Message Services (SMS) to notify the system administrators (http://en.wikipedia.org/wiki/Network_monitoring).

Web-based monitoring systems work in a similar way to other systems or hardware, but with the advantage of being managed from any computer with access to Internet connection.

This current thesis describes the development of a web-based monitoring system proposed by CLX Networks, a Swedish company that performs Bulk-messaging services offering business oriented SMSs.

Currently, CLX Networks uses as a monitoring system an open source application called OpenNMS (Network Management System). This application is used to manage the alarms, the notifications and to present graphs. However it has the disadvantage of not being integrated with the company’s intranet resulting then in longer processes when any information from the intranet database needs to be used by OpenNMS.

The objective of this project was to develop an application totally integrated with CLX Networks intranet in order to replace their current monitoring system in a near future.
2. Background

In the latest years, with the trivialization of the cell phones, the SMS (Short Message Text) has been extensively used as a communication tool. A broad range of companies has also been taking advantages of SMS to reach their customers and also to improve the marketing of their businesses.

Banks, for example, have been frequently using SMSs to alert customers on account transactions; flight and train companies deliver ticket numbers and flight information to their customers via SMS (http://en.wikipedia.org/wiki/Bulk_messaging). In Sweden, SMS has been broadly used by the post office, logistic companies, and also used by doctors and dentists as appointment reminder tools.

Companies and organizations send large amounts of SMS via an aggregator, which has coupling to several mobile phone operators. Therefore an aggregator needs to monitor the SMS traffic to assure security for their clients and also find out problems in the system in an early stage (Frid, J. & Karlsson, F., 2009).

CLX Networks is a Swedish company founded in 2008 that, as mentioned in the previous section, provides Bulk-SMS services. Therefore the company works as an aggregator and sends a large number of SMS daily to different mobile phone operators.

According to Frid, J. & Karlsson, F. (2009), an aggregator is the step between a client and the mobile operator. A client is, for example, a flight or train company that sends SMS to their customers with the ticket number. The SMS is sent from the company to an aggregator and then, the aggregator recognizes which is the customer’s mobile operator and sends the SMS. The aggregator often sends SMS through another operators (also called gateways or suppliers) in order to either reduce the costs or because the aggregator does not have coupling directly to the receiver operator (figure 1).
2.1 CLX Networks: monitoring system

Currently CLX Networks uses OpenNMS, an open source network management application platform (http://www.opennms.org/about/) as their monitoring system. The application monitors any slow down and interruptions in the server and the SMS traffic such as the delay of the SMS deliveries and the number of delivery reports they receive. Besides the application shows different types of graphs to make it easier to visualize all the information obtained. The administrators of the OpenNMS can also configure threshold values and notification groups and associate them with a client or supplier account.

A notification group is a list of names and contact information of CLX Networks’ employees who will receive the notification (alarms) by SMS and/or email.

A client or supplier account is the account that is being monitored. It can refer to a client product, for example the post office that buys a specific product (gateway) from the aggregator, and to a supplier, that refers to a gateway, which is being monitored.

The thresholds are certain values that the application keeps observing in order to dispatch alarms. Figure 2 illustrates an example of a threshold in OpenNMS and how it can be associated with a client account. Notice that:

- S_DR is the number of delivery reports (every time a SMS reaches its destination, the system receives a delivery report).
- S.SM is the number of sent SMSs.
• Value is the threshold value (in percentage) that OpenNMS will be in search of.
• Re-arm represents the value (in percentage) that resets the alarm for that account.
• Trigger means how many times the thresholds should be read in a row, before it dispatches the alarms.

Figure 2 Threshold example

The example above exemplifies the monitoring of a hypothetically client account Customer1 gw140. When the number of delivery reports is lower than 60% of the number of sent messages, the system starts to count the triggers. If the value continues to be lower than 60% during 4 updates in a row, then the system notifies Nisse by email and Bosse by email and SMS.

In this example, the system will not dispatch any alarms if the number of delivery reports reaches the threshold value sporadically and if it does not reach the number of triggers specified in the threshold. According to Stefan Skedebäck (personal communication, 09, may, 2011) it may be a minor problem with the server, like a drop or a slow down for example and will not be representative for a notification.

Usually the number of delivery reports never reaches 100%, since mobiles can be turned off, out of area and even phone numbers that do not exist.

Figure 3 shows the layout of the thresholds configuration in OpenNMS.
Figure 3: Thresholds layout in OpenNMS
The major problem with the current application is that it has no access to the company intranet, which contains the accounts information. Consequently the connection between a threshold and an account needs to be done manually. According to Stephan Skedebäck (personal communication, 09, may, 2011) a supplier or client account is inserted manually in a PHP script. A virtual IP address and an html file are created for each account added in the script, in order to be monitored by OpenMNS. See the following figure.

Figure 4: Current notification system architecture.
2.2 Requirements

As a consequence of the OpenNMS limitations, CLX Networks has defined the need of a more flexible application to monitor the quality of SMS traffic and graphs visualization.

The requirements for the new system are:

• Intranet integration: the new monitoring system should have access to the intranet database
• Create, edit and delete thresholds and thresholds groups
• Create, edit and delete notification groups.
• Developed with Zend Framework and MySQL.
• Graphs visualization through RRD tools

2.3 Problem description

Based on the CLX Networks needs, this project deals with the development of a monitoring system that would replace OpenNMS in order to monitor the SMS traffic of clients and suppliers. During the planning and development of the project, it was necessary to explore:

• CLX Networks intranet structure
• Intranet database model
• Zend framework
• RRD-tool

2.4 Delimitation

The application developed during this current work is going to replace specially the PHP script and virtual IP addresses for each account that is monitored. See figure 6 for more details.
Figure 5: Delimitation of this study
3. Method

This study was developed during a period of ten weeks and the implementation was completed during the first seven weeks at CLX Networks office in Kalmar.

3.1 Work Method

3.1.1 Unified Process

The work presented here was planned according to the UP (Unified Process) method. The UP is an iterative and incremental software development process organized into iterations (Larman, 19). Every iteration had the length of one week and had its own requirements analysis, design, implementation and testing. At the end of the iteration, a new functionality was implemented, tested and integrated to the rest of the system. Unified Process organizes the project in four phases:

1. Inception: This phase was performed during a meeting with CLX Networks staff in Kalmar when the vision of the project was presented for the first time.
2. Elaboration: Identification of requirements and risks and iterative implementation of the architecture.
3. Construction: Iterative implementation of the remaining elements and preparation for deployment.
4. Transition: Test and deployment.

3.1.2 Documentation

Each iteration was documented on Evernote application, so it was also available online from any computer with an Internet connection. At the end of each week, an evaluation of the work was done and according to the results, new goals to the next iteration were established and the risks list was updated.
3.2 Development Method

The application was implemented with the server-side script language PHP, JavaScript and Zend Framework. The use of these techniques was a requirement of CLX Networks, so the application could be easily integrated with the intranet.

3.2.1 Development Environment

3.2.1.1 Zend Framework and Aptana Studio

Zend Framework (ZF) is an open source, object-oriented framework implemented in PHP 5 (http://en.wikipedia.org/wiki/Zend_Framework). The framework (version 1.11.3) was installed in a Mac Book Pro along with Zend server.

Aptana Studio was chosen as the development environment for coding, because it was already installed on the computer. In order to make periodic backups and to code PHP properly the following plugins were installed on Aptana Studio:

- **PHP development tools**: have the components necessary to develop PHP. It makes easier to navigate among classes and methods and helps coding with intelliSense (auto complete functionality).

- **Subclipse SVN**: provides the functionality to interact with a Subversion server.

3.2.1.2 Subversion

Subversion is a version control system that allows different persons to be working in the same project simultaneously. The code is checked out from a repository, the developer makes the changes and then commits it back to the Subversion server. Subversion tracks of the code changes and merges the modifications into the code. (http://agile.csc.ncsu.edu/SEMaterials/tutorials/subclipse/).

3.2.2 Implementation Architecture

Zend Framework provides Model-View-Controller (MVC) implementation that can be used to structure an application.

MVC is a design pattern that separates the layers of an application such as presentation, business logic, and data access and organize them in Models, Views and Controllers.
This organization allows that multiple views and controllers can be associated with the same model (http://www.enode.com/x/markup/tutorial/mvc.html). MVC pattern has become indispensable for large projects, because it keeps the code well organized. Its organization also helps when several people work in the same project (http://framework.zend.com/manual/en/learning.quickstart.intro.html). Figure 6 shows MVC pattern in more details.

![MVC Diagram](image)

**Figure 6: Model-view-controller pattern**

**Model**: implies the functional core of an application such as data access layers and business logic.

**View**: gets data from the model and presents it to the user

**Controller**: binds the views and models together, decides which view to display based on the users requests and send the data to the view.

### 3.2.3 Testing

Zend framework is integrated with PHPUnit. Unit testing is a method of software testing that performs a battery of tests in small part of the code, checking if it behaves as expected. It is used for identifying defects in the code during its performance (http://phpunit.sourceforge.net/).

In Zend framework, all controller test cases should extend Zend_Test_PHPUnit_ControllerTestCase. Which in turn extends PHPUnit_Framework_TestCase. To be able to test the models, the test class should
extend PHPUnit_Extensions_Database_TestCase. Then the structure is shaped and it becomes possible to call the assertions functions.

Assertions are the center of unit testing. They are used to verify if the obtained result is the same as the expected (figure 7).

```
class EqualsTest extends Zend_Test_PHPUnit_ControllerTestCase
{
    public function testFailure() //fail
    {
        $this->assertEquals(1, 0);
    }

    public function testFailure2() //fail
    {
        $this->assertEquals('bar', 'baz');
    }

    public function testSuccess() //assertion
    {
        $this->assertEquals("bar", "bar");
    }
}
```

Figure 7: Example of assertions

### 3.2.4 Database

The database used for the development of this project was MySQL. Zend server contains, among other components, phpMyAdmin, which is free software used for the administration of MySQL databases ([http://www.phpmyadmin.net/home_page/index.php](http://www.phpmyadmin.net/home_page/index.php)).

### 3.2.5 RRD-Tool

RRDtool (Round Robin Database) is a database designed to store time series of data and it is structured to receive data at predefined time intervals. It can be configured to calculate the rate of changes from the previous to the current value and store this information. It can also be set up to create graphs ([http://www.mrtg.org/rrdtool/tut/rrd-beginners.en.html](http://www.mrtg.org/rrdtool/tut/rrd-beginners.en.html)).

PHP needs to have RRDtool extension in order to run RRDTool. It is possible to download this RRDtool source from the Internet and transfer it to the PHP-5.1.3 extensions directory and then recreate the PHP configuration file. Due to lack of time and other minor problems, RRD-tool was not installed or even tested in the current application.
3.3 Method discussion

3.3.1 Zend framework

Zend is an object-oriented framework and it means that its classes can be extended infinitely. This characteristic makes Zend a very flexible framework, but on the other hand a difficult tool to be understood in the beginning. When a developer jumps in a project that had been already started and has more than a hundred of different classes, it is not uncommon that an inexperienced developer writes unnecessary code because of lack of both knowledge of the whole framework and the wrong usage of class inheritance.

During this first experience with Zend, one of the advantages of being working with the framework and not only with an usual PHP application was that form validation was a very simple process through the function addValidator(). It is possible to pass a constructor, a validator name or even a regular expression to this function (figure 8).

```php
$form = new Zend_Form; // render as a HTML form tag
$form->setAction('/resource/process')
    ->setMethod('post');
$form->setAttrib('id', 'login');  // render as an HTML input tag of type text
$username = new Zend_Form_Element_Text('username');
// Passing the text element type to the form:
$form->addElement($username);
// validate with a constructor
$username->addValidator( new Zend_Validate_Digits() );
// validate with validator name
$username->addValidator('alnum');
// validate with regex
$username->addValidator('regex', false, array('/^[a-z]/i'));
// element is required
$username->setRequired(true);
// string to lower case
$username->addFilter('StringToLower');
```

Figure 8: Example of how to validate a form in Zend framework

Another advantage in working with Zend is the process of insert, update and delete data in a database and to work with transactions in Zend is a very flexible process when the class extends AbstractDataAccessLayer (figure 9).
Figure 9: Example of the usage of transaction in Zend

3.3.2 Zend framework problems

Zend front-end formatting is not so easy to learn and configure as those back-end functionalities mentioned above.

During the development of the application some problems regarding form structuring were encountered, especially when multiple checkbox were implemented and needed to be nested and checked by default. To check the checkboxes by default is a very simple practice in an ordinary PHP application, but Zend renders all the checkboxes as a single array and once one attribute is passed to one element of the array, then all the other elements also get that attribute. Figure 10 shows this multiple checkbox example comparing a common PHP form and a form in Zend Framework. All the input tags in Zend get the checked attribute. In Zend, the best way to manipulate how the elements of a form will be displayed is through helpers, but its usage is not so intuitive.
One easier way to fix that was by creating two arrays, one for the values checked by default and another for the non-checked values. Then add each array to a different checkbox element.

Another drawback with Zend is the lack of support or a public community. Most of the Zend courses are expensive and there are not many public tutorials. The process of installation was also complicated and long.

### 3.3.3 Testing

The major mistake with the current development method was the lack of priority of the test cases during the first iterations. Unit test can affect the whole system and it
should be done often. The reason the tests were not prioritized in the beginning was that the goals of each iteration were focused on the requirements of the system functionalities, in other words, the use-cases. Besides there were many techniques that should be understood in order to get the application running until the deadline.

3.4 Implementation

3.4.1 Work Method

The development of this project along with Unified Process was a good experience. It helped keeping the project moving in a good pace and at the end of each iteration a new small functionality was implemented, tested manually and delivered. The staff of CLX Networks checked and gave feedbacks according to the pre-defined requirements.

Evernote was another good experience; especially because it was available online and it could be reached even from a cell phone application.

3.4.2 Development Method

During this two-year program we, as students, have frequently worked both individually and in groups and in most cases the applications were developed from scratch. The major difference and most valuable experience concerning the development of this project was the possibility to work with a real-world application.

The challenge was to understand how the CLX intranet was structured as a Zend framework application, in order to be able to continue with the implementation of the monitoring system.

The goal was to integrate the monitoring system with the Zend application and for that it was suggested to create a controller that would manage all the user actions related to the monitoring system. This controller is called Surveillance and it should manage the configuration of thresholds, notification groups and clients and suppliers accounts, previously explained in this thesis.

The accounts are the center of the surveillance, because they would be bonded to a threshold and to a notification group, in order to have a specific monitoring.
The development of the controller and the views that would display the accounts information were the goals of the first iteration. Besides, the familiarization with the Zend framework, code standard and structure was carried out.

The development process continued on the other iterations:

- Iteration 2: database modeling (see appendix 1) and implementation of the tables related with the thresholds. Development of thresholds view and insert threshold functionality.
- Iteration 3: implementation of insert, edit and delete thresholds.
- Iteration 4: code refactoring and insert, edit and delete threshold groups. Binding threshold groups with the accounts.
- Iteration 5: database modeling (see appendix 2) and insert, edit and delete notification groups
- Iteration 6: binding notification groups with the accounts, database modeling 3, testing.
- Iteration 7: database modeling (see appendix 3) and code refactoring
- Iteration 8, 9 and 10: testing, report and documentation.

3.4.3 Database

At CLX Networks, the new system functionalities are first implemented in a test database environment, tested properly and then transferred to the system main schema. This procedure promotes security in the development process and protects the company database. If something was deleted by mistake or any other problem occurred it did not affect the official application or the main schema database.

3.4.4 Subversion

A branch called surveillance was also created so the surveillance could be implemented without affecting the whole intranet. Some minor problems emerged sometimes after merging the branch with the trunk, like conflicts or test failures.
4. Results

4.1.1 Surveillance

The outcome of this work is an application that has Surveillance controller that manages the views and the information of the alarms configuration (see appendixes 4 to 7). It appears in the main navigation of the intranet website as shown in figure 11 below.

![Figure 11: Surveillance controller and its views](image)

4.1.2 Tables

The tables presented in each view have a predefined layout that is similar for all tables displayed in the intranet. The layout is produced through a jQuery table template (figure 12).

```javascript
$options = CLX_View_Helper_DataTable::$PARAM_DEFAULT_ARRAY;$options[CLX_View_Helper_DataTable::$PARAM_DISPLAYLENGTH] = 50;$addThresholdReport = $this->report_threshold;

$addThresholdGroupReport = $this->addThresholdGroupReport;
$e = '<div style="margin-top:10px;margin-bottom:10px">"'.
   $this->dataTable( "thresholdGroupsReport", $addThresholdGroupReport, $options ) ." </div>";
```

![Figure 12: Table formatting with jQuery template](image)

All tables (except the overview table) have edit and delete links that allow the management of information in every row of the table (figure 13 to 15).

![Figure 13: Notification group table](image)
Through the surveillance overview table the administrator can make the binding between an account and a threshold group and a notification group. The client and supplier tab allows the user to navigate between clients and supplier accounts. The edit surveillance button redirects the page to the edit surveillance form, where the administrator can either add or remove accounts to this table (figure 16).

4.1.3 Forms

The forms are used to add and edit information to the tables (figure 17 to 20).
4.1.4 Integration

4.1.4.1 Alarm handler

Alarm handler is the script that allows a connection between the surveillance, the intranet subscription system and the CDR (Call Detailed Records) engine.

Surveillance is referred here as the accounts that are being monitored and therefore such accounts are connected to a threshold and a notification group.

The CDR engine is still under development, and it will connect the intranet with the CDR database (that receives information about the SMS traffic) and the RRD tools.
(that process the information from the CDR database and make graphs and calculations).

Even though a detailed explanation about the CDR engine and the subscription system was out of scope of this thesis, the subscription system can be summarized as an engine that checks if there is any subscription to be delivered to any subscriber. The subscription is for example an information about the company products or invoices. The subscriber can be an employee or a client. The system sends the information of a subscription to a subscriber by email and by SMS.

Mattias Johansson (personal communication, 2011, 04, 25) recommended the implementation of an Alarm handler that would send the surveillance information to the subscription system, which in turn sends it to the people listed in the notification group for the account. Appendix 8 illustrates a sequence diagram showing the flow of information between the surveillance and subscription system.
5. Conclusion

The application developed during this study is a good foundation for the further development of the monitoring system. Whereas the interaction between the user and the system is ready to be used, the connection between the system and the CDR- and RR-database is still under development, as mentioned previously.

Now that the intranet is prepared to receive data for monitoring the SMS traffic, the proposal of a new architecture for the monitoring system is possible (figure 21):

1. CDR-database stores SMS transaction record, like for example, which client, supplier and SMS status.
2. CDR engine will receive information from the CDR database and then will process the information and send it continually to the RRD.
3. With every data update, RRD calculates the rate of changes and create graphs pictures that will be displayed in the intranet database.
4. If the values reach the thresholds, the CDR engine sends the information to AlarmHandler (in the intranet).
5. Finally AlarmHandler sends the alarms by email and/or SMS.

![Diagram](image)

Figure 21: new monitoring system architecture proposal
6. References

6.1 Electronic sources

*OpenNMS*

http://www.opennms.org/about/  
2011-05-10

*phpMyAdmin*

http://www.phpmyadmin.net/home_page/index.php  
2011-05-21

*PHPUnit*

http://phpunit.sourceforge.net/  
2011-05-21

*Realsresearch - Software engineering group*

http://agile.csc.ncsu.edu/SEMaterials/tutorials/subclipse/  
2011-05-18

*RRDTool*

2011-05-18

*Wikipedia*

http://en.wikipedia.org/wiki/Bulk_messaging  
2011-05-07

http://en.wikipedia.org/wiki/Network_monitoring  
2011-05-07

2011-05-10

*XML Middleware*

http://www.enode.com/x/markup/tutorial/mvc.html  
2011-05-21

*Zend Framework*

2011-05-21
6.2 Books


6.3 Reports

7. Appendix

Database Model

Appendix 1: Database Modeling (version 1)
Appendix 2: Database Modeling (version 2)

Appendix 3: Database Modeling (version 3)
Layout

Appendix 4: Overview view

Appendix 5: Edit surveillance view
Appendix 6: Thresholds view
Appendix 7: Notification view
Appendix 8: AlarmHandler sequence diagram