A Solution Design and Test Case for Invoice Generation System in Operational Dynamic Telco Industry

MOHD RIDZAM AHMAD, AHAIRUL AFZA ABDUL WAHAD, KOH TIENG WEI*, KHAIRONI YATIM SHARIF
Software Engineering Research Group, Department of Software Engineering and Information System, Faculty of Computer Science and Information Technology
Universiti Putra Malaysia
43400 UPM Serdang, Selangor Darul Ehsan
MALAYSIA
ridzam@gmail.com, sa.airul@gmail.com, *twkoh@upm.edu.my (*corresponding author*), khaironi@upm.edu.my

Abstract: - An Invoice Generator system helps the Telco Company to accelerate invoice generation for customer billing purposes. However, companies face difficulty to perform test on the Invoice Generator system with the rapid changes of Billing and Rating plan over time. Since the ERP and CRM systems are central to the functioning of the Telco Company, it is important to not disrupt these systems in the process of performing better output for the invoice statement. Thus, a proper technique, design and test cases required for the company to execute the software testing. Finally, the company could use the design solution described in this study to minimize errors in invoice productions.

Key-Words: - Invoice generation system, solution design, test case, telco industry

1 Introduction

In a Telco domain industry, companies should provide great services to the customer. However, there are few challenges faces by these companies to accelerate the invoice production with the optimal cost in dynamic business environment today. Besides that, increasing number of system in generating invoice is very competitive for the companies to get the high quality of invoice. This study will provide the Telco domain companies a solution design to execute testing on the invoice generator system which best suit in the company practise.

Fundamentally, billing process involved provisioning services within the billing system that importing the Call Detail Records (CDRs) from various sources like ERP and CRM to cater rates calculation and billing information which finally works as input towards invoice generator system.

Software testing is a technique and method used for validating functionality of software application to meet state holder business requirement. It is based on the specification gathered in the earlier design stage of software development. These tasks will help developer finding possible numbers of faults and errors occurs in software development. The faults and errors can categorise in different severity level to prioritise the application fault fixing [1].

The first step in software testing is selecting efficient and effective techniques from various methods in testing. These techniques implemented with systematic procedure to ensure the tester get best quality result when testing of the system. As example process in decision of selecting the metric-based approaches in test estimation, choosing between black box or white box for test design and which testing techniques used statics or dynamic [2].

Inefficient of testing techniques may reflect software application reliability in the future usage. Most of the cases in software failure, have related with problem in past development phase. It have resulted the increasing of software maintaining and creates financial losses to the stakeholders. To overcome these issues, the industries must conduct total software testing but this process is time consuming and need lot of resources [3]. Because of this, we knew how important to ensure software testing applied in correct manner and being utilised properly [4].

Therefore selecting suitable testing techniques is very challenging task to optimise techniques capability in finding flaws of software application. The person involved in testing must able to select suitable techniques within a factor of limited resource faced by stakeholder, cost effective, time management, programming languages, platform use and type of faults predicted [3].

The development of testing techniques is including the creating of test case. In IEEE 610 definition, a test case in software engineering is a set of condition under which tester will determine whether an application or software is working correctly or not. It defines the pre-conditions of testing, set of input values, set of expected result, how to execute the test and expected post-conditions.

The test case is critical success factor in software testing because the main goal of test case is fully testing the software application. The suitable designs of test cases will be able to detect numbers of flaws occurs in software application. The test cases must generate more efficient testing capability and also in the same time, able reducing the cost of software testing. A good quality of test cases has ability to adequate and fulfils the test objective. It has overall criteria and features that contributing efficient software testing for benefit of stakeholders [4].

2 Industrial Problem

The software testing practices still lack of reliability tool to conduct quality test in software development. Software community still finding the way to improve the testing techniques and tool to support test processes. Industries needed more sophisticated tool and seamless integration between build and test [2].

J.Lee et al. [2] has stated current status of software testing in the industries. It shows the software testing still faced issues listed below:

- Usage of software testing methods and tools by the industry is still low
- Industries have experiences difficult in using software testing
- Usage of software testing methods and tools is limited. It only applied at ease activities like tracking the status of defects.
- There are high demand in support for interoperability of software development and testing.
- There is need for guidance for industry in knowing the capabilities of software testing methods and tools.

In Telco industry, currently they’re not using appropriate approach in conducting software testing activities. They still used same methods and tools in testing for different types of projects. We must know that not all testing techniques appropriate with this industry, it must define properly for efficient testing result. Also involvement of software testing should start at earlier design phase where tester’s team can do static review of requirements to ensure that all these being include in testable condition [4].

Inclusive with the software testing, there is generation of test cases. The idea of test cases is a set of question that tester would like to ask for the program when finding the faults and errors exist in software application [5]. They are two main objectives test cases must achieve:

- Finding the faults and errors could trigger problem in software application
- Provided information refer by development team in fixing the faults and error

David Hendrick [4] has mention problem about test cases generation. The generation activity could create problem of bottleneck in the project if not done earlier after requirement specification signed off. Therefore, the generation activity should be conducted parallel with coding development.

The lack of well define of testing techniques also effect the generation of test cases. Because of that, it’s important when selecting testing techniques and which one is suitable in a given context of stakeholder [1]. This problem should cater in this researcher to build a suitable framework in view of Telco industry.

3 Related Work

In conducting the software testing, we will select appropriate techniques to improve and verify the quality of software product. Figure 1 below shows the different purpose of software testing in the information flow [1]. It will contribute very useful information at different stage for project manager when testing the software application.

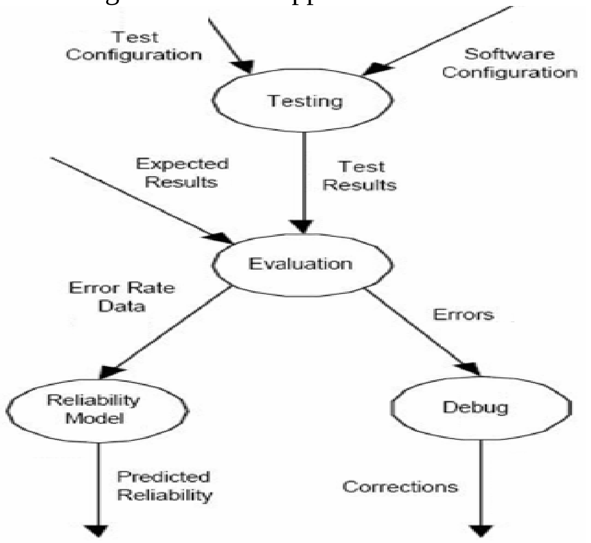

Figure 1: Test Information Flow (Adapted in Sheikh Umar Farooq [1].

There are different types of software testing techniques and can be classified based on approaches [2]. It listed as below:

- Structural Based – In this approach test cases are generated using system source code or control flow graph of the program.
- Functional Based - In this approach test cases are generated using system specifications to test the functionality of software.
- Gray Box Based - In this approach both structural and functional information are used for generating test cases.
- Non-Functional Based - Non-functional testing is testing of how the system works at all test level. It based on the test required to measure characteristics of system and software that can be quantified on a varying scale.

The software testing techniques also can be classified in term on it algorithm as refer below:

- Random Based Algorithm – In this approach test cases and events sequences are generated randomly. It mostly used for analyse the efficiency of other algorithms.
- Search Based Algorithm–In this approach the problem of generating test cases are considered as an optimisation problem and try to find optimise solution including best test set for the problem under test.
- Data Mining Based Algorithm – The goal in this approach is analysing the input or output of program under test to reduce the number of test cases by eliminating unimportant and infeasible cases.

Based on classification identified above, test cases must fulfil all the functionality features as below [5]:

- Ability to find defect could affect the system failures
- Maximising the bug count in testing activities to overcome flaws is software application.
- Ability to block premature product release or shipment that could affect system reliability.
- Provide information that guides the manager in making decision of product release.
- Ability that could minimising the cost of technical support when product shipped.
- Make sure software development team fulfilled the requirement specification stated in design stage.

However, to assess the quality depends on the nature of the product, and usually refers to the guidelines and definition by the stakeholder.

4 Flow Process

Figure 2 gives an overview of the bill formatting flow when Invoice Generator system is implemented. During bill run, BCH generates multiple XML files per invoice to be processed by BGH. BGH merges the multiple XML files into one XML file per invoice and stores in the BGH output directory. A manual script is used to move the XML files from multiple BGH output directories to one directory in the SAN storage that will be used as the input directory by Invoice Generator system. Invoice Generator system polls the input directory and start processing the XML files. Invoice Generator system produces invoices in PDF format and stores them in the SAN storage. The PDF file path is updated to BILLING database during PDF generation.

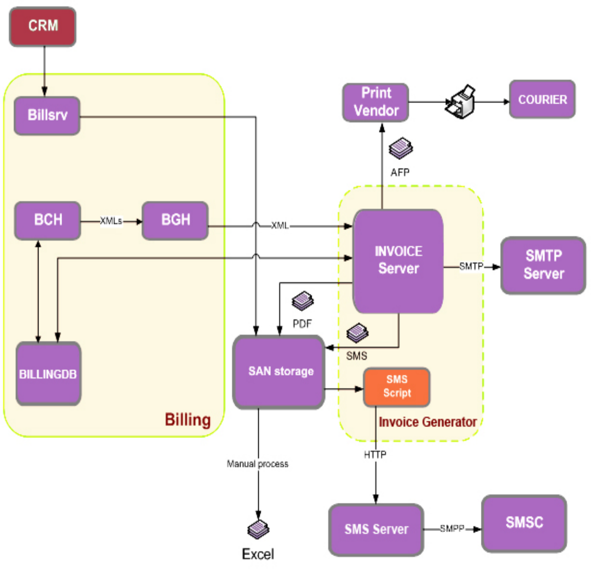

Figure 2: Bill Formatting Flow Process

CRM able to download the PDF bills online after PDF invoices are generated. Invoice Generator system sorts and bundles invoices in AFP format to be sent to the Print Vendor by the Company. Invoice Generator system generates emails and sends them to the SMTP server via SMTP protocol. Invoice Generator system generates SMS text files and stores them in the SAN storage. A manual script is executed to read the text files and send them to SMS server via HTTP protocol for sending to SMSC. Bill summary report for large account will be created manually in Excel format by referring to the PDF bills generated for the account.

The scope of this study is to provide an overview of the architecture and also solution design of the Invoice Generator system solution which will be handling input files for invoicing purposes.

5 Proposed Solution Design

The proposed solution design has been divided into four components: Input Collection, Document Design, Post Processing and Output Distribution.

5.1 Input Collection

Input directory for the XML files to be polled by Invoice Generator system and XML part to be used as value to the invoice will be defined in Input Collection.

5.1.1 Directory Input Connector

Directory input connector will be used to retrieve files from a specified directory. The XML files produced by BGH needs to be copied to the specified directory for Invoice Generator system to process. The XML files will be deleted from the specified directory after processed by Invoice Generator system.

5.1.2 Event

The XMLIN configuration is used to specify which fields to select from the input XML file and to create the structure of the Message. XPath expressions and XSLT patterns are used to map the values in the XML tags to fields created. Recurring data are defined as fields within a block.

5.2 Document Design

The bill layout, display and template will be defined in the document design

5.2.1 Bill Layout

The Design tool can be used to design the layout and content of the bill document. The flow of the design and the output of this process can be delivered to multiple output connectors, for example PDF and AFP. Overlays containing logo can be added and positioned in the pages of the document using Storyteller tool. Different processes can be used to define different bill layouts, where the selection criteria of process to be used will be defined via scripting.

5.2.2 Itemized Bill

The logic used to determine whether to display the Itemised Bill section for each contract in the bill is as the following:

Step 1:
If exists
PerCTInfo[@CT=’A’]/SumItem/AggSet/Att[@Ty=’SN’ and @Id=’ITEMI’]
Get the values for the following fields within the SumItem:
SumItem/SrvStatus/@Status
SumItem/SrvStatus/Date[@Type=’SRV_END’]/@Date
Step 2:
If @Status=’a’ for the largest @Date, then
Display Itemised Bill section
Else do not display Itemised Bill section

The logic above will be checked within Invoice Generator system scripting in Storyteller process.

5.2.3 Display of Credit Limit

The credit limit of the customer billed will be retrieved directly from BILLING database by executing the following query:
select cslimit from customer_all
where custcode = $custcode;
where $custcode is the customer code of the customer billed.

The SQL query will be executed using Invoice Generator system ODBC script functions which will connect to an ODBC data source to execute SQL statements.

5.2.4 Display of Rate Plan Name

To display the rate plan name for each contract in the Account Summary page, the following logic is implemented in Invoice Generator system scripting within Storyteller process:
Step 1:
Get the following values from recurring charges in the input XML:
PerCTInfo[@CT=’A’]/SumItem/AggSet/Att[@Ty=’TM’]/@Id
PerCTInfo[@CT=’A’]/SumItem/AggSet/Att[@Ty=’SN’]/@Id
PerCTInfo[@CT=’A’]/SumItem/SrvStatus/Date[@Type=’SRV_END’]/@Date
where
PerCTInfo[@CT=’A’]/SumItem/AggSet/Att[@Ty=’SN’]/@Id = COMFE or TEL or 3GDAT or FRDAT
Step 2:
The TM_Id with the largest@Date value will be used to retrieve the rate plan name from MPUTM.tbl in the following sequence of SN_Id:
a) COMFE
b) TEL
c) 3GDAT
d) FRDAT

5.2.5 Display of Deposit

As deposit amount information are stored in the BILLING database. A script needs to be created to calculate the total deposit amount paid by the customer and update it to a new DEPOSIT table for Invoice Generator system access to avoid performance impact. This script needs to be executed before Invoice Generator system begins processing the input files.

An SQL query will be executed using ODBC functions by the Invoice Generator system to retrieve the total deposit amount from the BILLING database and update it to the new DEPOSIT table in Invoice Generator system. The total deposit amount will then be displayed in the corresponding pages of the document in the design process.

5.2.6 Update of PDF Filepath to Billing Database

The full PDF filepath will be updated to the BILLING database based on the @Refnum value of the bill. Invoice Generator system ODBC functions can be used to execute the SQL update statement by connecting to BILLING database directly during processing of each input file.

5.2.7 Email Template

SMTP output connector can be used to send emails via a SMTP mail server. The email information like sender, subject and recipient email address can be configured in the Edit mail settings of the email connector.

The Storyteller process can be used to design the email template and the output will be used to generate the email body. The PDF bill generated from another process which is stored on disk can be used as the email attachment by setting the file path of the bill using a variable. The attachment name can be configured as well to be different from the original PDF filename.

5.2.8 SMS Template

The SMS template can be defined in a configuration text file for easy maintenance. Invoice Generator system will fetch the SMS template from the configuration file during runtime and replace the variables defined in the template with the input data via scripting.

5.3 Post Processing

Output documents can be stored into a document repository, and then retrieved using a post-processor and Post Processor Query (PPQ). All document sorting and bundling features are then applied to the documents retrieved from the repository via the post processor.

5.3.1 Optical Mark Reading (OMR)

OMR codes will be printed based on the result of bar functions that are implemented. A separate function will be defined for each of the position of the bar according to the specification provided in the Requirement Specification document.

5.3.2 Sorting

Sort keys need to be stored as metadata with the document in the post processor repository. The following are the sort keys defined for sorting documents in AFP format:

a) postcode sequence number

The sequence number for postcodes will be configured in the POSTCODE_REGION table in Invoice Generator system database. A function will be created to get the sequence number from the table based on the postcode of the customer billed.

b) total number of pages

The total number of pages generated for the bill.

5.3.3 Bundling

The following are the bundling keys to be stored with the documents in the post processor repository:

a) page groups – to group the bills based on the total number of pages generated
b) region – to group the bills based on the region of the postcode from the bill

The keys will be used to retrieve documents stored in the post processor repository.

5.4 Output Distribution

Bill format and the file output connector will be defined in the output distribution.

5.4.1Bill Format

The bill format short description can be used to identify the combinations of bill formats to be generated for each customer. The following table describes the bill mediums to be configured in BILLING:

Table 1: Bill Format Configuration

Short Description	Long Description	PDF	AFP	Email	SMS
PAES	PDF, Print, Email, SMS	X	X	X	X
PAE	PDF, Print, Email	X	X	X	
PAS	PDF, Print,	X	X		X

	SMS				
PES	PDF, Email, SMS	X		X	X
PA	PDF, Print	X	X		
PE	PDF, Email	X		X	
PS	PDF, SMS	X			X
P	PDF	X			

Invoice Generator system will identify the bill formats to generate by getting the bill medium value from the input XML file, and send the input to the correct processes to generate the output accordingly.

A small migration needs to be performed on the existing customers in BILLING to update the bill mediums of the customer according to Company's bill segregation. The default bill medium needs to be set as well for handling new customer creation.

5.4.2File Output Connector
File output connector is used to generate output documents in PDF and AFP formats on the file system.

Output from the same Storyteller process can be sent to multiple output connectors by using an array variable to store the names of the connectors to connect to. To suppress AFP generation for zero bills, the following logic needs to be implemented before storing the connector names in the array variable:

If the length of /Bill/Header/@RefNum field = 10
store PDF connector name into array variable
else store PDF and AFP connector name into array variable

5.4.3Email Output Connector
The SMTP output connector will be used to send the output via an SMTP mail server. The email server settings can be configured in the physical layer of the Platform in the Invoice Generator system.

5.4.4SMS Text File
A text file will be created for each billing date and the MSISDN and SMS text found in the input XML files will be appended as a row into the text file. This will be implemented by using scripting within the Invoice Generator system. The rows will be appended when the input XML files are processed by Invoice Generator system during runtime.

One SMS text file will be created for each input XML file, containing one MSISDN and SMS text in a row per contract found. This will be implemented by using scripting within the Invoice Generator system.

6 Proposed Test Cases
Test cases required for the system testing which carry out from the components explained in Section 5 are discussed below:

6.1Input Collection

6.1.1 Directory Input Connector
a) To verify that the XML files is copied to Invoice Generator system input directory.
b) To verify that Invoice Generator system will be deleting the XML files after processing it.

6.2 Document Design

6.2.1 Itemized Bill
a) To verify that Invoice Generator system will display the Itemised Bill section in the bill when the option is turn on
b) To verify that Invoice Generator system will not display the Itemised Bill section in the bill when the option is not turn on

6.2.2 Display of Credit Limit
a) To verify that the credit limit is displayed when the option is turn on.
b) To verify that the credit limit is not displayed when the option is turn off.

6.2.3 Display of Rate Plan Name
a) To verify that the rate plan name is displayed when the option is turn on.
b) To verify that the rate plan name is not displayed when the option is turn off.

6.2.4 Display of Deposit
a) To verify that the deposit is displayed when the option is turn on.
b) To verify that the deposit is not displayed when the option is turn off.

6.2.5 Email Template
a) To verify that the email template is used when generating the email as below:
b) To verify that PDF invoice is included in the email without the barcode and OMR
c) To verify that the attachment file naming convention: eBILL_DDMMYYYY_CustId.pdf

6.2.6 SMS Template
a) To verify that only 1 SMS Notification is generated for Customer having only 1 MSISDN.

b) To verify that 2 SMS Notifications is generated for Customer having 2 MSISDN.

6.2.7 Mapping configuration

a) To verify that all the mapping tables is exist in the specified directory.

6.3 Post Processing

6.3.1 Optical Mark Reading (OMR)

a) To verify that the OMR is generated base on the configuration.

6.3.2 Bundling

b) To verify that the bundling for page group is based on the configuration.

6.4 Output Distribution

6.4.1 Bill Medium

a) To verify that the bill format is generated base on the configuration.

6.4.2 Email output connector

a) To verify that the recipient of the email receive the email.

6.4.3 SMS Text File

a) To test SMS Notification for Customer having only 1 MSISDN.
b) To test SMS Notification for Customer having more than 1 MSISDN. All MSISDN should receive the SMS Notification.

7 Conclusion

The study proposed solution design and test cases to test the Invoice Generator system. As a conclusion, a solution design for Invoice Generator system testing covered the components on Input Collection, Document Design, Post Processing and Output Distribution. Test cases were derived from the four components to fulfill the best practices in a Telco company. However, future work should be conducted on the integration parts to support the current conclusion.

8 Acknowledgements

This research work has been funded by the RU-Putra Grant of Universiti Putra Malaysia (UPM) in collaboration with the Malaysian Ministry of Education (MOHE) under the project no. GP-IPM/2013/9404800. The authors would like to thank the Research Management Centre of UPM and MOHE for their support and cooperation including students and other individuals who are either directly or indirectly involved in this project.

References:

[1] Sheikh Umar Farooq,S.M.K. Quadri,“Evaluating Effectiveness of Software Testing Techniques With Emphasis on Enhancing Software Reliability”,Journal of Emerging Trends in Computing and Information Sciences, December 2011

[2] Mohammad Reza Keyvanpour, Hajar Homayouni, and Hossein Shirazee ,“Automatic Software Test Case Generation: An Analytical Classification Framework”, International Journal of Software Engineering and Its Applications, October 2012.

[3] J. Lee, S. Kang, D. Lee, “Survey on software testing practices”, IET Software, April 2011

[4] David Hendrick, “Using Grounded Theory to Develop a Frameworkfor Software Testing Best Practice in aTelecommunications Company”, Dublin Institute of Technology for the degree of M.Sc. in Computing (Information Technology, January 2013

[5] Cem Kaner,“What Is a Good Test Case?”,Florida Institute of Technology, Department of Computer Sciences, May 2003