RESEARCH ARTICLE OPEN ACCESS

Measuring Non-Functional Requirement via Cloud Hosted Application in Favour of Booking System

Anamika Sharma,Brijesh Pandey
(Department of Computer Science,Goel Institute of Technology & Management,Lucknow)

Abstract:

Functional requirements are elaborated and mapped in many ways. Software development totally based on functional requirements, but NFRs are included in those ones that supply the norms while implementing the code. At the designing process, often we have forgotten the inclusion of the NFRs sometimes it becomes expensive to handle it. Now in the era of cloud technology, it becomes very important since response latency and concurrent load becomes more unsafe by public networks because now it is not enough to develop a cloud system with only functional requirement they need to include NFRs as integral parts. Here in our work we have concentrated on modelling of NFRs and a covert from UML model to source code. In our work we have chosen three parameter request response time, concurrency and pick seat time.

***Keywords*—UML, NFRs, Latency, HTML5**

I. INTRODUCTION

When we build software, we have to gather all the requirements regarding that software. The initial phase of the software development life cycle is requirement gathering [1]. Both end user and developer are affected by these requirements. Whole requirement body is categorized into functional and non-functional requirements[2]. In which functional requirements are primarily revealed because it affects the life cycle of development directly.Software development totally based on functional requirements, but NFRs are included in those that supply the norms while implementing the code[3]. At the designing process, many authors have concerns about the NFRs and the problems of their inclusion[4]. Pavlovski and Zou [R1] describe NFRs as particular performance and operational constraints, such as work expectations and policy constraints[5]. While the reality is that, NFR's are described in many ways. We learned, talked, and even faced these NFR's but just because NFR's does not affect the software directly so these are not considered acutely as they should be. Glinz [6] gives the advice that we have to make two separate parts of functional and NFRs and grouped them into two different sets so both requirements can inherently considered while developing the applications. Alexander [7] said that when we focuses on the language which used to describe the requirements, we find a particular word as postfix that is '-ility'[8]. Examples of these words are portability and maintainability. His work is totally oriented on the recognition of NFR's[9]. Our work establish on theirs by applying domain specific models using flexibility mechanisms construct into standard modelingnotations[10].

II. NEED OF THE STUDY

Ranabahu and Sheth discus that, when we represent cloud application requirements there are four different modelling semantics necessary which include data, functional, non-functional and system[11]. Their efforts oriented on functional and system requirements. There are some similar aspects in Ranabahu and Sheth and my work[12], but that is only in NFRs from the system point of view[13]. They build on the work done by Stuart in his workshops. There are three phases of the cloud application life cycle, in that Stuart defined well-formed modeling languages to model the cloud computing requirements. Those three phases of cycle are development, deployment, and management. Our work used to add those semantic categories of NFRs which are missing[14].

III. RESEARCH PROPOSAL

All paragraphs must be indented. All paragraphs must be justified, i.e. both left-justified and right-justified.For the study of all this, we consider mainly three NFRs that is response time, concurrency and pick seat time to implement our system "ticket booking system". We model in UML and OCL deploying stereotypes to apply the additional required semantics for each NFR. To implement the NFRs, we focus on generation of codes of the model. We generate these codes for the NFRs which are used in cloud application. Then all applications use the thread on the server side for every user.

A. Request response time

In an online system, every one (client) interact with any application (on the server) by just making the request about that application in Cloud system. That request response time differentiates any application with other application and minimum request time is the main aspect by which every user selects its application. This time is one of the top most performance measures in our ticket booking system. We take this NFR and depict it as a 'Response time' stereotype in our UML activity diagram (Figure). This stereotype directly connects every relation between client and server. Generally, response time can be classified as the elapsed time between the requests has placed and the first response made. In an activity diagram, a control flow can be set the stereotype. We use control flow in the algorithm for applying this stereotype. In an initial step, we measure the time before the request is sent to the server. When a response is received, then the difference between times is checked. The response time is described by the difference of send and received time. The different latency requirements are depicted by the specific stereotypes. For example "low latency" and "high latency". The permitting time for every stereotype can be defined through runtime configuration. For calculating the average response time for the whole system, we measure response time for every request generated by users is measured. By this approach, we measure an appropriate all over system performance. After that we can make the comparison of it over time. If there is a pattern of increased average response time, we can again notify the average response time of every module/ type of request and then find the bottlenecks. For ensuring this NFR, we use the implementation of Algorithm 1. In the algorithm, there is a rollback situation of any half work done. The client gives the notification to the server when the timeout occurs. The algorithm enables the server to rollback. In the algorithm, the client informs the server to enabling to rollback for any incomplete work, if there is a timeout condition occurs.

B. Concurrency

Concurrency is a robustness computation of any application, mainly for any online ticket booking system. This threshold concurrency is represented as 'Simultaneous Users' stereotype in the UML activity diagram (Figure). We make a pool of threads of size on stereotype. We execute this phenomenon by swapping the threads. All these requests are handled by the server by just making them in a queue. For processing every request, each instance of the request is pulled from the queue and is assigned to a thread from the pool. For execution of this stereotype that is measuring the concurrency, we judge and notify the latency of the request. We note particular timing at which the request is sent to the server and also measure that when the response is received from the server. We take latency for the request as the time difference between when the request is sent and server replied to that. This latency of each request is noted and the queue time is adjoined to the record. For measuring the system performance totally, we use that measured record of latency. These records are used for comparison over time. If the average latency time increases, we can again take the average latency of each module/type of requests and find the bottlenecks. When measuring the concurrency stereotype, the bottleneck is frequently caused by a pool of threads. That bottleneck is smaller than the demand on the server. To guarantee this NFR, we use the implementation of Algorithm 2.

C. Pick seat time

There is also a situation in which a user does not give any response to a running form in a definite time period. This is the major and essential NFR for many systems. In the online ticket booking system, the user goes through the system, then, select or pick the seat. When the user selects the seat, the resources are locked from other current user. The time duration of the locks are held needs to be minimized. The form response time requirements are represented in the standardized form in our UML activity diagram as 'Limited user time'. In the ticketing application system, the stereotype which is particular to the booking seats activity. The definite time is allotted to the user for picking its seat for response. For implementing this idea, we hold together, an event and request submission of the client. For measuring the time factor used by the user in the ticket booking system, the client application polls are used. This application poll continuously to checks that if the request is sent within the specified time. If a user makes late and take more time than the time specified by the stereotype, then the user receives a message in which by default all locks have been released. There is another situation in which the user's request is sent before the specified time, and then the user will continue their task and proceed to the next activity. To guarantee this NFR, we use the implementation of Algorithm 3

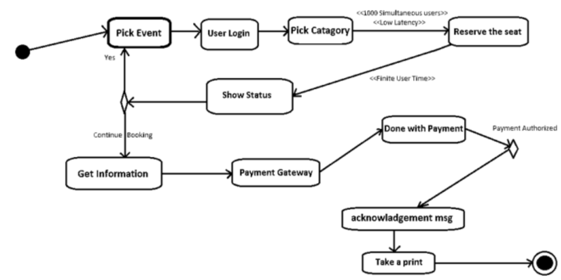

Figure.1 Activity Diagram

D. ALGORITHMS:

Algorithm 1. Request Response Timeout

INPUT:HTML5 of request to Server, completion_ time

OUTPUT: HTML5 of response with server

Send request to server Set timer to hit every second Set holdTimeLimit = 0 Do

Check if response is received

While holdTimeLimit<compeletion_time or response received

If response not received

Set response as "Expired Time" error

Set response to timeout error

Examine server of compeletion_time

End if

Return response

Algorithm 2. Concurrency

INPUT: HTML5 of request, compeletion_time**OUTPUT:** HTML5 of data entered or HTML5 with error

Check if any threads in pool

If no threads in pool Set timer to hit every second Set holdTimeLimit = 0 Do

Check if thread available in the pool

While holdTimeLimit<compeletion_time or thread received

If not thread received

Set response to timeout error

ELSE

Execute request in thread

End if

Return response

Algorithm 3. **User Response Timeout**
INPUT:HTML5 of form to display,compeletion_time
OUTPUT: HTML5 of data entered or HTMl5 with error
Display form to user Set timer to hit every second
Set compeletion_time = 0 Do
Check if response is received
While holdTimeLimit<compeletion_time or response received
If response not received
Notify user of time expiration
Set response to timeout error
End if
Return response

NON-FUNCTIONAL REQUIREMENTS THAT AFFECT THE USER'S SELECTION TOWARDS THE CLOUD SERVICE PROVIDER:-

In today's scenario, there are many service cloud providers in the market. But we also make our selection on the basis of NFRs which directly affects the user. This is discussed before, in the start of this paper. Here we make a chart of leading applications which shows the situation of market of cloud providers.

Security Feature:-Amazon Web Service uses several Operational Security features like Vulnerability management, Malware prevention, Monitoring, Incident management, Server and Software Stack Security, Trusted Server Boot, Secured Service APIs and Authenticated Access, Data Encryption, Network Firewall Rule Maintenance.

Here are the cloud leaders we will profile:

Amazon Web Services
Microsoft Azure
IBM Cloud
Google Cloud Platform

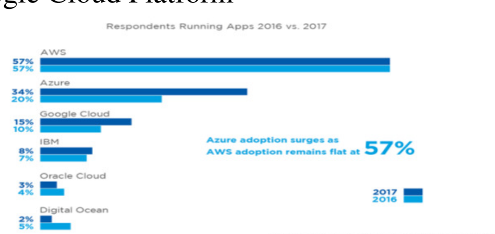

Figure 2:

5.5. PUBLIC CLOUD PROVIDERS COMPARISON CHART

When we look at the comparison of the dominant and leading public cloud providers we have to be very careful: few of the services truly line up in an "apples-to-apples" similar style. There are several cloud providers which provide their services to us but we choose our provider on the basis of some important features. The chart below should help us to get started.

Requirements	**Google cloud**	**Amazon Web Service**	**IBM**	**Microsoft Azure**
Compute	Bare Metal Servers Virtual Servers Power8	EC2	Compute Engine App Engine	Virtual Machines
Storage	Object Storage Block Storage File Storage Mass Storage Servers	S3 EBS EFS Glacier	Cloud Storage Persistent Disk	Blob Storage Queue Storage File Storage Disk Storage
Database and Data warehouse	Data Services Big Data Hosting MongoDB Hosting	Aurora RDS DynamoDB Redshift	Cloud SQL Cloud Bigtable Cloud Spanner Cloud Datastore	Data Lake storage SQL Database Document DB Table Storage SQL Data Warehouse
Container	Containers	Container Registry Container Service	Container Engine Container Registry Containe	Container Registry Container Service

			r Builder	
Serverless/Faas	OpenWhisk	Lambda	Cloud Function	Functions
Analysis	Analytics Service Cloudera Hosting	Athena EMR Kinesis	BigQuery Cloud Dataflow Cloud Dataproc Cloud Datalaba	HDInsight Stream Analysis
Artificial Intelligence	Watson	Lex Polly Recognition Machine Learning	Cloud machine Learning Engine Cloud Natural Language API Cloud Speech API Cloud TransactionAPI Cloud Vision API	Machine Learning Cognitive Services Bot Services Data Lake Analytics
Internet Of Things	Internet of Things	IOT Platform Greengrass		IOT Hub Event Hub
Backup and Disaster Recovery	Backup			Backup Site Recovery
In-Memory Technology		Elastic Cache		Redis Cache

5.5. BASIC NON-FUNCTIONAL REQUIREMENTS COMPARISON OF LEADING CLOUD PROVIDERS

When we look at public cloud providers which provide basically the internet, there are a great number of options in the market to select our appropriate service. As an example, more than 95public clouds are registered with the monitoring service [R12].Now days, every public cloud provider has several proposals of heterogeneous services for their clients. We cannot directly compare these services with other cloud provider's services because of diversity in services. Just because our work is focuses on Non-Functional Requirements we choose some leading cloud providers here, which are in the market and provides us their utilities:

Non Functional Requirement	**Google Cloud Platform**	**Amazon Web Service**	**IBM**	**Microsoft Azure**
Security features	App Engine only	Amazon Inspector, Secured Socket Layer(SSL) Certificates	Secured Socket Layer(SSL) Certificates	Azure Security Center, Secured Socket Layer(SSL) Certificates
Web Firewall		AWS Web Application Firewall	Hardware Firewall	Azure Application Gateway
i) Preventive measures	Moderate	Moderate	Basic	Basic
ii) Reactive measures	Moderate	Moderate	Basic	Basic

Reliability	Good	Good	Good	Average
Scalability	Good	Good	Good	Good
Support		Good and chargeable		Good
Compliance		AWS Artifact		Azure Security & Compliance
Availability (%)	99.95	99.95	99.95	99.95
Server Performance (Over a period)	Excellent	Good	Good	Excellent and consistent
Tools/ framework	Python 2.7, Java 7, PHP, Node.js, and Ruby	Amazon machine image (AMI), Java, PHP, Python, Ruby	Ruby, PHP, JAVA, Python, Node.js, ASP.Net,	PHP, ASP.NET, Node.js, Python

Database RDS	MySQL(Cloud Sql), Big Query	MySQL, MsSQL, Oracle		Microsoft SQL Database
Data Warehousing	Big Query	Amazon Redshift	dashDB for Analytics	Azure SQL Data warehouse
Pricing	$0.15 per cluster per hour, Nearline storage $0.01/ GB/Month, $0.05/ GB/Month	Instances range from $0.113/hour to $6.82/hour, with volume discounts available for reserved instances. Storage prices range from $0.095/ GB/month to $0.125/ GB/month.	Storage $0.0295 - $0.0354/ GB/Month	Instances range from $0.02 to $1.60 per hour. Storage prices range from $0.07/GB/ month to $0.12/GB/ month, depending on level of redundancy.
Sample Average Pricing (4GB RAM, CPU 2 Core,100GB	69$ per month	62$ per month	115$ per month	99$ per month

HD, Bandwidth 100GB, Linux)				
Trial Offering	30-day trial	New users can get 750 hours, 30GB storage and 15GB bandwidth for free with AWS's Free Usage Tier.	30-day trial, 2 GB of runtime and container memory for free	Free 30-day trial with a limit of up to $200 is available for new users.
Limitations	A global fiber network, Analytics that crunch petabytes in minutes.	AWS is a complex mixture of services. As your workflows become more complex and you use more services it can be difficult to project expenses. However,	IBM improves a collaboration of essential data.	Minimal, easy-to-use portal interface may not be so appealing to command line gurus.
		Amazon offers a monthly calculator to help estimate your costs.		
Data Centers Location	America, Asia, Europe	US East, US West, South America, Europe/ Middle East/Africa & Asia Pacific	USA, Netherland, INDIA, China, Germany, Australia, Canada	US East, US Central, US West, Europe, East Asia, Southeast Asia
CDN Locations (Edge)	North America, South America, Europe, Asia	North America, South America ,Europe/ Middle East/Africa & Asia Pacific	**Europe, Middle East and** Africa, Japan, North America, **Asia-Pacific**	US East, US North, US Central, US South central , US West, Europe, Asia Pacific / Rest of World
Acceptance By User(Rating)	4.5 , 177 reviews	4.4, 187 reviews	4.4, 49 reviews	4.4, 183 reviews

IV. CONCLUSIONS

In this work, we show that we can map NFR's to many cloud based applications using UML stereotypes. As we know that UML is the modeling diagram in which we show the process in an incremental and interactive way. We expand the NFR's to design the model for Cloud based application rather than functional requirements. We model in UML and OCL deploying stereotypes to apply the additional required semantics for each NFR. We focus on basic three NFR's by which we gather the information about our end user's transaction and response. Future work will enhance my work to include these NFR's for modeling and converting the codes of NFR's into cloud application tools and also enhances the type of NFR's for other upcoming methodologies. By this, we can develop the quality of these methodologies. Our work enhances the performance and characteristics of applications. This will also reduce the error rate.

REFERENCES

[1] C. J. Pavlovski and J. Zou, "Non-functional requirements in business process modeling," Proceedings of the Fifth on Asia-Pacific Conference on Conceptual Modelling, vol. 79, 2008.

[2] M. Glinz, "Rethinking the Notion of Non-Functional Requirements," Third World Congress for Software Quality, Munich, Germany, 2005

[3] Alexander, I, "Misuse Cases Help to Elicit Non-Functional Requirements," Computing & Control Engineering Journal, 14, 40-45, 2003.

[4] A Saxena, S Sharma,PAgarwal, C Patel "SSTL Based Energy Efficient FIFO Designfor High Performance Processor of Portable Devices " in International Journal of Engineering and Technology (IJET)Vol 9 No 2 Apr-May 2017DOI: 10.21817/ijet/2017/v9i2/170902113.

[5] R. Ajith and A. Sheth, "Semantic Modeling for Cloud Computing, Part I," Computing, vol. May/June, pp. 81-83, 2010.

[6] https://www.cloudorado.com/cloud_providers_comparison.jsp

[7] A Saxena,CPatel,M.Khan "Energy Efficient ALU Design Based On Voltage Scaling" in Gyancity Journal of Electronics and Computer Science,Vol.1, No.1, pp.29-33, September 2016ISSN: 2446–2918 DOI: 10.21058/gjecs.2016.11006.

[8] John, S. & Laurie, W. (2013). Automated Extraction of Non-functional Requirements in Available Documentation, IEEE.

[9] Asghar, S., & Umar, M. (2010). Requirement engineering challenges in development of software applications and selection of customer-off-the-shelf (COTS) components. International Journal of Software Engineering, 1(1), 32-50.

[10] Pandey, D., Suman, U., &Ramani, A. K. (2010, October). An effective requirement engineering process model for software development and requirements management. In Advances in Recent Technologies in Communication and Computing (ARTCom), 2010 International Conference on (pp. 287-291). IEEE.

[11] Selvakumar, J., &Rajaram, M. (2011). Performance Evaluation of Requirements Engineering Methodology for Automated Detection of Non Functional Requirements. International Journal

[12] Dwork, C., Rothblum, G. N., &Vadhan, S. (2010, October). Boosting and differential privacy. In Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on (pp. 51-60). IEEE.

[13] ASaxena, S Gaidhani, A Pant, C Patel "Capacitance Scaling Based Low Power Comparator Design on 28nm FPGA" in International Journal of Computer Trends and Technology (IJCTT) – Volume X Issue Y- Month 2015.

[14] ASaxena,CPatel, P Verma, PGautam "Cloud forecaster: Gizmo for Evaluation of Bulky Cloud Computing Surroundings to Propagate ICT based Education" September 2017International Journal of Engineering and Technology 9(4):3396-3400DOI: 10.21817/ijet/2017/v9i4/170904177