# Consistency Verification and Quality Assurance (CVQA) Traceability Framework for SaaS

Sukhpal Singh
Computer Science & Engineering Department
Thapar University
Patiala, India
ssgill@hotmail.co.in

Inderveer Chana
Computer Science & Engineering Department
Thapar University
Patiala, India
inderveer@thapar.edu

***Abstract*—Software as a service (SaaS) is a kind of cloud service which offers software services through internet. SaaS is commonly utilized and it provides several benefits to service consumers. To realize these benefits, it is essential to evaluate the quality of SaaS and manage the relatively higher level of its quality based on the evaluation result and verify consistency of artifacts. Hence, there is a high demand for devising a quality model to evaluate SaaS cloud services. Conventional frameworks do not effectively define excellence assurance strategies for SaaS development and a traceability framework for consistency verification. In this paper, we propose the excellence assurance strategies for SaaS development and a traceability framework where all the artifacts (Data to Trace) can be correlated and the consistency of artifacts can be verified. Till traceability framework is considered an important ability, required high quality SaaS Cloud Services cannot be provided to cloud consumers. Finally, we define our continuing efforts to characterize the inherent trade-off between effective excellence assurance strategies for development and consistency verification of artifacts, a critical problem in SaaS cloud computing. The consistency of artifacts has been validated by Z Formal specification language in general. Using the proposed framework, Saas services with high quality can be efficiently developed and consistency can be verified.**

***Keywords—SaaS development; traceability framework; high quality services; cloud computing.***

## I. INTRODUCTION

Cloud computing has emerged as one of the most influential paradigms in the IT industry in recent years [1]. Cloud computing is a new computing paradigm that is built on Virtualization, parallel and distributed computing [19], utility computing, and service-oriented architecture [12, 20]. In the last several years, cloud computing has emerged as one of the most influential paradigms in the IT industry, and has attracted extensive attention from both academia and industry [21]. Cloud computing holds the promise of providing computing as the fifth utility [2] after the other four services (basic needs). The benefits of cloud computing [22] include reduced costs and money outflows, increased operating efficiencies, scalability, flexibility, time to put in market. Different service-oriented cloud computing models have been suggested, including Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) [23].

Software as a service [3], occasionally indicated to as "on-demand software" [24], is a software delivery model in which software and associated data are centrally put on on the cloud. SaaS is typically gain access by users consuming a thin client through a web browser. SaaS has turn into a public distribution [23] prototypical for several business applications, containing accounting, association, customer relationship management (CRM) [25], management information systems (MIS), enterprise resource planning (ERP) [26], invoicing, human resource management (HRM) [27], content management (CM) and service desk management [4]. SaaS has been incorporated into the strategy of all leading enterprise software companies [4] [5]. One of the biggest selling points for these companies is the potential to reduce IT support costs by outsourcing hardware and software maintenance and support to the SaaS provider [7]. Cloud Computing (CC) is emerging as an effective reuse prototypical [28], where software functionality, hardware computing power, and other computing resources are distributed in the method of service so that they turn into accessible broadly to users [8]. Software as a Service (SaaS) is one type of cloud services, where software functionality is delivered as a service [9] [10]. SaaS provides benefits to service consumers; no initial cost to purchase software, free of maintenance/updates, accessible through the Internet, high availability, and pay-per-use pricing [30]. That is, SaaS should maintain a relatively higher level of its quality than conventional system [23]. Consequently, these intrinsic features of SaaS require more rigorous quality measurements. According to Webster's Dictionary [31], ''Traceability is the capability to monitor or study out in detail, or step by step, the history of a certain action or a procedure". Therefore traceability can be well-defined as the history of manufactured goods in terms of the direct properties of that product and/or belongings that are related with that product when these products have been focus to specific value-adding procedures with related invention means and in related ecological circumstances [12]. The data regarding interactions at beginning may be used upstream in the supply chain or downstream. Moreover, the material can be used for reporting purposes, either in the supply chain or to third parties. Traceability [31] among software product line development artifacts is more complex than traceability among single software development artifacts, because it deals with commonalities and variability's [33]. Therefore, consistency verification among artifacts becomes more important [13]. But,

978-1-4673-4529-3/12/$31.00 ©2012 IEEE

complex traceability leads to complex verification. When the complexity of traceability is high, a formal approach will be useful to help verifying the consistency.

The main goal of our work is to establish a framework for automated formal verification of consistency among artifacts along with effective excellence assurance strategies. The motivation behind this paper is to propose excellence assurance strategies for SaaS development along with consistency verification of artifacts.

The paper is structured as follows: Related work has been presented in Section 2. In Section 3, a description of Consistency Verification and Quality Assurance (CVQA) Traceability framework has been presented. The validation of CVQA Framework has been presented in Section 4. In Section 5, the contribution of proposed CVQA Framework has been described. The conclusion and the future work have been presented in Section 6.

## II. RELATED WORK

Most of current works are not for SaaS but for certain targets such as a conventional software or SOA based system. Due to the situation, it is hard to evaluate the quality of SaaS and judge which SaaS is better. Therefore, our work provides excellence assurance strategies to evaluate SaaS effectively.

Kim's work defines a model for web services quality management and quality factors in the process of developing and using web services [6]. This work suggests six quality factors and their several sub factors. Also, it provides metrics to measure quality factors. Hence, it is required that this model is customized and extended to evaluate the quality of SaaS.

C. Zhu et al. [14] analyzed domain requirements dependencies' influence on product line architecture, and suggested formal mapping rules from desires to feature [18] [32] and from feature to architecture. But it's not clear whether their mapping rules can be applied for automated verification or not.

Nicolas Anquetil et al. [15] explains the management of traceability links" criterion was adopted to analyze the capacity of each traceability tool to create and maintain trace links (manual or automatic) and what kind of trace information is generated. The traceability queries" criterion analyzes what searching mechanism is available from the tools to navigate among the artifacts and respective trace links, varying from simple queries to navigate among the related artifacts, to more sophisticated queries that support advanced functionalities, such as coverage analysis or change impact analysis [11, 18].

Alexander Shraer et al. presented [16] the traceability view" criterion characterizes the supported views (tables, matrix, reports, and graphics) that each tool provides to present the traceability information between artifacts [12]. The extensibility" criterion evaluates if any tool offers a mechanism to extend its functionalities or to integrate it with any other software development tools.

The main role of this research is to create a foundation for formal verification of consistency among artifacts, by CVQA Framework and then verifying their consistency using Z specification language.

## III. CVQA TRACEBILITY FRAMEWORK

### A. Problem Statement

Describe real excellence assurance strategies [18] for the software development life cycle phases and to define a traceability framework where all the artifacts can be cross related and consistency can be verified.

### B. Excellence Assurance Strategies

To develop an effective SaaS based application, there is a need for some guidelines improve the quality of software [13] [18, 33]. The Table I show the excellence assurance strategies for SaaS based application development.

TABLE I. SAAS EXCELLENCE ASSURANCE STRATEGIES

| Phase | Instructions |
|---|---|
| Requirement gathering | SaaS requirement Specification |
| | Requirement safety analysis |
| | V & V requirement analysis |
| | CM Requirement analysis |
| Domain analysis | Commonalties and verification model |
| Functional sculpting | Use case, DFD |
| Structural sculpting | ER diagram, Structure chart, class diagram |
| Dynamic sculpting | Activity diagram, sequence diagram |
| Architectural design | h/w and s/w Architecture |
| | Design safety analysis |
| User interface design | Design safety analysis |
| | V&V design analysis |
| Database design | CM design report, database schema |
| Implementation | Code testing |
| | CM implementation report |
| | Code safety analysis |
| | V&V implementation |
| Assimilation and analysis activities | The system builds documentation |
| | Integration safety analysis |
| | V&V integration & Test |
| | CM integration analysis |
| Authentication activity | Authentication protection investigation |
| | V&V validation analysis |
| | CM validation analysis |
| Deployment | Installation configuration |
| | Installation safety analysis |
| | V&V installation analysis |
| | CM installation report |
| | Operation manuals |
| | Training manual |
| Operation and Maintenance | Maintenance manual |
| | Change safety analysis report |
| | V&V change report |
| | CM change report |

### C. Consistency verification

To verify the consistency of artifacts and data that has been collected through various data collection techniques through consistency verification matrix as shown in TABLE II and the abbreviation used in the table is DtT= Data to Trace. The Z specification language will be used for verification of data by using the schema described in section 4.

TABLE II. CONSISTENCY VERIFICATION TRACEABILITY MATRIX

| Phase | DtT-1 | DtT-2 | …… | DtT-n |
|---|---|---|---|---|
| Requirement gathering | √ | | | |
| Domain analysis | | √ | | |
| Functional modeling | | | √ | |
| Structural modeling | | | | |
| Dynamic modeling | | | | √ |
| Architectural design | | | √ | |
| User interface design | √ | | | |
| Database design | | | | |
| Implementation | | | | |
| Integration and testing activities | | √ | | |
| Validation activity | | | | |
| Deployment | | | √ | |
| Operation and maintenance | | | | |

### D. *CVQA Framework*

We have proposed a Consistency Verification and Quality Assurance (CVQA) Traceability framework where the cloud providers give the facility of estimation and comparison of cost according to specified billing rules. The implementation of CVQA enables the SaaS based application development with QA strategies and verification with tracing the data. This framework improves satisfaction and quality in SaaS Cloud Development with Software Quality Attributes [12, 13] and Software Reuse [33]. The CVQA Traceability Framework has been described in Fig. 2. The various interactions [18] in SaaS based software services have been described in Fig. 1.

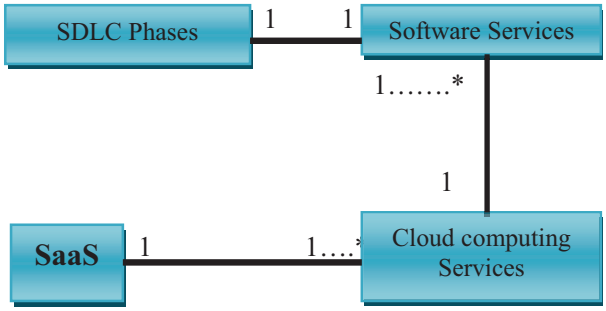



Fig.1. Criterion in communication purpose

## IV. VALIDATION OF CVQA FRAMEWORK

Formal specification can serve as a single, reliable reference point for who investigate customer requirements, those who implement programs to satisfy those requirements and those who verify the results [17]. CVQA Framework stated above can be verified using formal language like the Z specification language. In Z specification [17, 18], schemas are used to describe both the static and dynamic aspects of a system. Z decomposes specifications into manageably sized module's called schemas: Schemas are divided into three parts: 1. A state, 2. A collection of state variables and their values and 3. Operations that can change its state [29].

This section explains how the CVQA Framework deals with the requirements. The set of all requirement names and identification number as the basic types of the specifications [29].

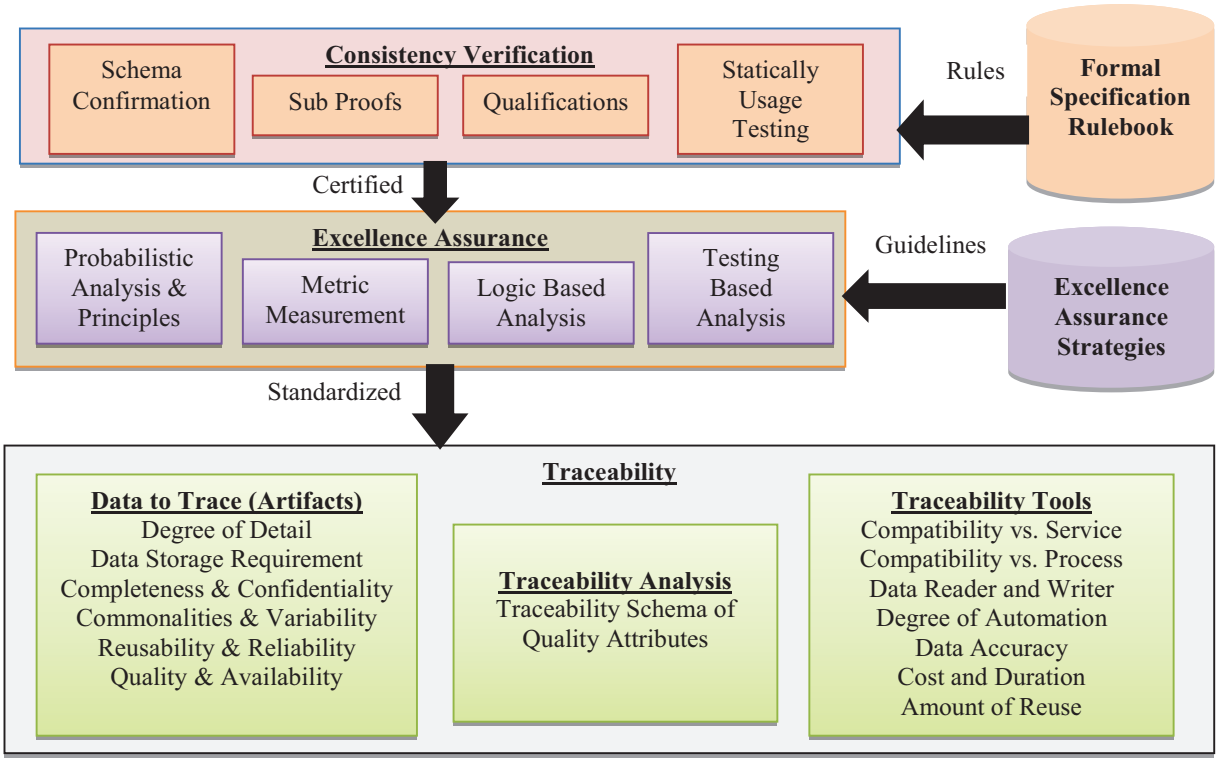
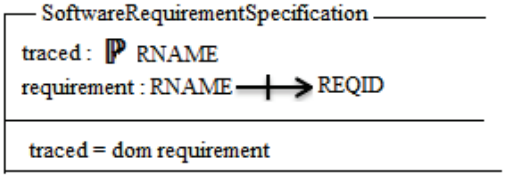



Fig. 2. SaaS CVQA Traceability Framework

[RNAME, REQID]

The first aspect of the requirement traceability is its state space.




In our work, the space of requirement traceability has been described and the two variables represent important observations which can make of the state [29].

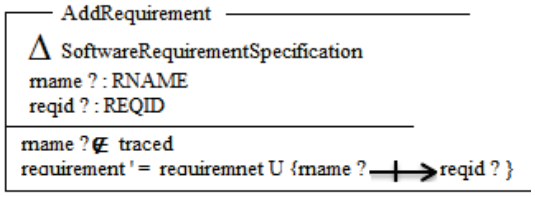
- *traced* is the set of rnames with existed requirements.
- *requirement* is a function that when applied to certain names, gives the requirement identity (REQID) associated with them.
- set *list* is the same as the domain of the function requirement the set of names to which it can be validly applied.

traced = {Reusability, Reliability, Availability, Quality}

*requirement* = { Availability →R-1,
Reliability →R-2,
Reusability →R-4,
Quality →R-3}

The invariant is satisfied because requirement details a REQID for four names in traced. There are some operations that can apply on the requirement traceability:

The first of all there is to add a new requirement, and we describe it with scheme:




The Δ *SoftwareRequiremnetSpecification* alerts us to the fact that the schema is describing a state change: it introduces four variable traced, requirement, traced' and requirement'. The first two are observations of the state earlier the modification, and the most recent two are interpretations of the state after the change [29].

We expect that the set of rnames known to SoftwareRequiremnetSpecification will augmented with new rname.

traced' = traced U {rname ?}

We can prove this from the specification of AddRequirement using the invariants on the state before and after.

traced' = dom requirement'

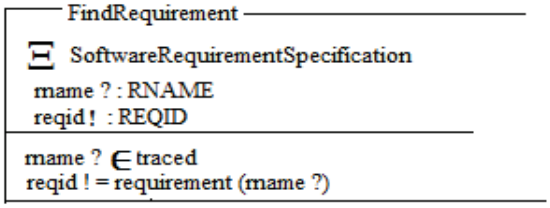
= dom (requirement U {rname ? → reqid?})

= dom requirement U dom {rname ? → reqid?}

= dom requirement U {rname ?}

= traced U {rname ?}

In FindRequirement for, traceability schema finds the requirement for user applications.



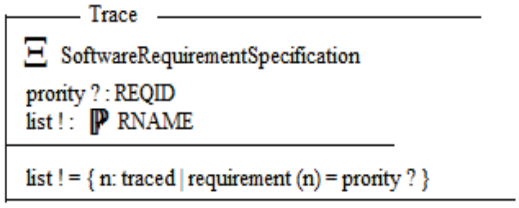

The declaration *Ξ SoftwareRequiremnetSpecification* indicates that this is an operation in which the state does not change, the value of list' and requirement' of the observations after the operation are equal to these values traced and requirement.

Including *Ξ SoftwareRequiremnetSpecification* above the line has the same effect as including ΔSoftwareRequiremnetSpecification above the line and two equations below it.

traced' = traced
requirement' = requirement

The other notation (!) for an output the FindRequirement operations takes a names as input and yield corresponding requirement as output.

The most useful operation on requirement traceability is one to find which project has requirements on a given reqid. The operation has an input priority? And one output, list! Which is set of rnames of requirement? There may be zero, one or more project with requirements with particular requirement identity, to whom requirement list should be sent.




This time there is no pre-condition. The list! is specified to be equal to the set of all values n drawn from the set list such that the value of the requirement function at n is priority? [29]. In general, b is a member of the set {a: C |……..a……} exactly if b is a member of C and the condition ……b……, obtained by replacing a with b, is satisfied:

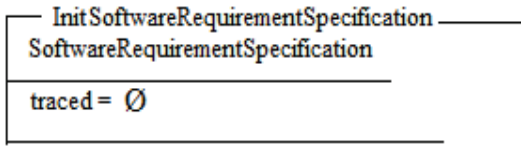
b ∈ {a: C|…a…..} ↔ b ∈ C □ (….b….)

m ∈ {n: list □ requirement (m)= priority? }

↔ m ∈ list □ requirement(m) = priority?

A name m is in output set list! exactly if it is known to the requirement traceability and the requirement recorded for it is priority? [29].

The given below schema identify the initial state of the system:

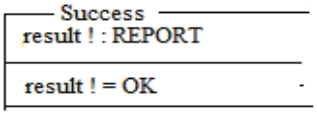



This schema describes a Software Requirement Specification in which the set known is empty: in consequence, the function requirements empty too.

We shall add an extra output! to each operation requirement traceability. The result will be OK after successful execution of given process [29], but it may take the other value already_traced and not_traced when error is detected. REPORT defines the set contains three values.

REPORT::=OK/already_traced/not_traced

The result should be OK after proper execution of success schema without saying how the state changes.

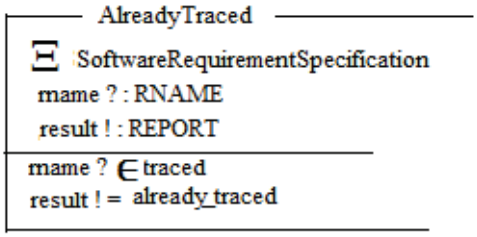



SoftwareRequirementSpecification □ Success

The conjunction operator □ of the schema calculates allows us to combine this description with our previous description of SoftwareRequirementSpecification:

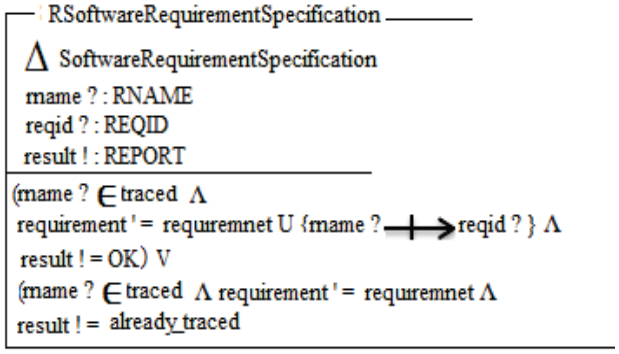
The process for accurate input has defines both acts as described by SoftwareRequirementSpecification and produces the result OK.

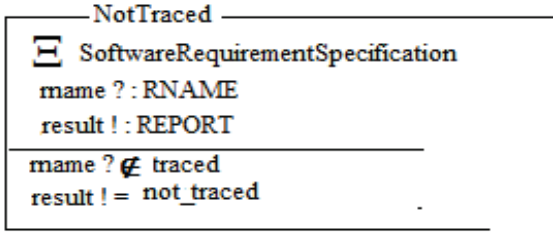
Schema specified that the report already_traced should be produced when input rname? Is already a member of traced.




*Ξ SoftwareRequirementSpecification* specifies that if the error occurs, the state of the system should not change.

RSoftwareRequirementSpecification$\hat{=}$ (SoftwareRequirement Specification) V already_traced




RFindRequirement $\triangleq$ (FindRequirement ∧ Success) ∨ not_traced

Traced operation can be called at any time, it never results an error robust version need only add reporting of success.

RTraced $\triangleq$ Traced ∧ Success

## V. CONTRIBUTION OF PROPOSED CVQA FRAMEWORK

- Make sure that each task gives its predictable result with Functional Testing, all functions combine to deliver the desired business result with System Testing and new changes did not adversely affect other parts of the system with Regression Testing.
- The artifacts should be easily traced as well as verified correctly through this framework.

## VI. CONCLUSION AND FUTURE SCOPE

Our proposed CVQA Framework is validated by Z specification language defined in section 4, as presented in Figure 2. To recognize the paybacks of SaaS, it is essential to make a well-defined Consistency Verification and Quality Assurance (CVQA) Traceability framework available. In this paper, we presented a systematic process for verifying high quality with this framework. Through CVQA Framework verification and quality of SaaS based applications development can be improved.

The future scope of this work is to analyze and to incorporate risk factors in Cloud Development systematically and find the critical success factors of the CVQA framework and also identify the various risk factors using risk analysis of introducing excellence assurance strategies and consistency verification in SaaS and offer a framework that will help us to achieve high quality in Cloud Development. Consistency Verification along with Quality Assurance can also be

automated in SaaS using an automated tool. The automated model solves the problems of quality and verification of SaaS based application development to a large extent.

## REFERENCES


[1] Zhiguo Wan et al., HASBE: A Hierarchical Attribute-Based Solution for Flexible and Scalable Access Control in Cloud Computing, IEEE Transactions On Information Forensics And Security, VOL. 7, NO. 2, APRIL 2012, 743-754.

[2] R. Buyya, C. ShinYeo, J. Broberg, and I. Brandic, "Cloud computingand emerging it platforms: Vision, hype, and reality for delivering computingas the 5th utility," Future Generation Comput. Syst., vol. 25, pp. 599–616, 2009.

[3] Panker, Jon; Lewis, Mark; Fahey, Evan; Vasquez, Melvin Jafet (August 2007). "How do you pronounce IT?". TechTarget. Available:http://searchservervirtualization.techtarget.com/definition/How-do-you-pronounce-IT, Retrieved 10 August 2012.

[4] "Software as a Service (SaaS)". Cloud Taxonomy. Retrieved 24 August 2012. Available: http://cloudtaxonomy.opencrowd.com/taxonomy/software-as-a-service/.

[5] Laird, Peter (5 June 2008). "How Oracle, IBM, SAP, Microsoft, and Intuit are Responding to the SaaS Revolution". Laird OnDemand. Blogspot. Retrieved 12 July 2012. Available: http://peterlaird.blogspot.in/2008/06/how-oracle-ibm-sap-microsoft-and-intuit.html.

[6] Jutras, Cindy. "QAD On Demand gives manufacturers the tools they need to become global". Aberdeen Research Group. TheStreet.com Inc. Retrieved 22 June 2012. Available: http://bx.businessweek.com/enterprise-software/view?url=http%3A%2F%2Fc.moreover.com%2Fclick%2Fhere.pl%3Fr3936360089%26f%3D9791

[7] "How SaaS Is Changing the Face of Enterprise IT Support". Dell.com. Available: http://www.dell.com/content/public/notfound.aspx?aspxerrorpath=/d/large-business/saas-changing-enterprise.aspx, Retrieved 16 July 2012.

[8] Weiss, A. "Computing in the Cloud," netWorker, Vol. 11, No. 4, pp. 16-26, 2007.

[9] Gillett, F.E., "Future View: New Tech Ecosystems of Cloud, Cloud Services, and Cloud Computing," Forrester Research Paper, 2008.

[10] Turner, M., Budgen, D., and Brereton, P., "Turning Software into a Service," IEEE Computer, Vol. 36, No. 10, pp. 38-44, 2003 S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[11] A. Regattieri et al., Traceability of food products: General framework and experimental evidence, Journal of Food Engineering 81, Available: www.elsevier.com/locate/jfoodeng (2007) 347–356.

[12] Clements, P. and Northrop, L., Software Product Lines: Practices and Patterns, Addison-Wesley, 2001.

[13] Tonny Kurniadi Satyananda et al., Formal Verification of Consistency between Feature Model and Software Architecture in Software Product Line, International Conference on Software Engineering Advances (ICSEA 2007), 2007, pp.10.

[14] Chongxiang Zhu, Yuqin Lee, Wenyun Zhao, and Jingzhou Zhang, A Feature Oriented Approach to Mapping from Domain Requirements to Product Line Architecture, Software Engineering Research and Practice, 2006.

[15] Nicolas Anquetil et al., A model-driven traceability framework for software product lines, Journal Software and Systems Modeling (SoSyM), Volume 9 Issue 4, September 2010 , Pages 427-451.

[16] Alexander Shraer et al, Venus: Verification for Untrusted Cloud Storage, CCSW'10, October 8, 2010, Chicago, Illinois, USA, 19-29.

[17] Spivey, J.M.: The Z notation: a reference manual, 2nd edn. Programming Research Group, University of Oxford.

[18] Pressman, R.S.: Software engineering: a practitioner's approach, 7th edn. ISBN: 0073375977.

[19] Rao Mikkilineni et al., Service Virtualization Using a Non-von Neumann Parallel, Distributed, and Scalable Computing Model, Journal of Computer Networks and Communications Volume 2012 (2012), Article ID 604018, pp:1-10.

[20] Daniel Nurmi et. al., Eucalyptus : A Technical Report on an Elastic Utility Computing Architecture Linking Your Programs to Useful Systems, UCSB TECHNICAL REPORT, 2008, pp:1-16.

[21] Buyya, R., Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities, High Performance Computing and Communications, 2008. HPCC '08. 10th IEEE International Conference, 5 - 13.

[22] Nurmi, D., The Eucalyptus Open-Source Cloud-Computing System, Cluster Computing and the Grid, 2009. CCGRID '09. 9th IEEE/ACM International Symposium, 124 - 131.

[23] Alexander Lenk et. al., What's inside the Cloud? An architectural map of the Cloud landscape, Proceeding CLOUD '09 Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing, Pages 23-31.

[24] Michael Armbrust et al., A view of cloud computing, Magazine Communications of the ACM, Volume 53 Issue 4, April 2010 Pages 50-58.

[25] Khalid Rababah et. al., Customer Relationship Management (CRM) Processes from Theory to Practice: The Pre-implementation Plan of CRM System, International Journal of e-Education, e-Business, e-Management and e-Learning, Vol. 1, No. 1, April 2011, 22-27.

[26] Fitzgerald, A., Enterprise resource planning (ERP)-breakthrough or buzzword, Factory 2000, 1992. 'Competitive Performance Through Advanced Technology'. Third International Conference on (Conf. Publ. No. 359), 291 - 297.

[27] Singh, R., Human resource management: the key to competitive advantage, Engineering Management Journal, Jun 1992, Issue: 3, 116-120

[28] L. Ghanmi, E-Design Based on the Reuse Paradigm, Proceedings of the conference on Design, automation and test in Europe, IEEE Computer Society Washington, DC, USA, 2002.

[29] Spivey, J.M.: The Z notation: a reference manual, 2nd edn. Programming Research Group, University of Oxford.

[30] Chaisiri, S., Cost Minimization for Provisioning Virtual Servers in Amazon Elastic Compute Cloud, Modelling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2011 IEEE 19th International Symposium, 85 - 95.

[31] Yumi Park et al., The Impact of RFID-based Traceability System on Perceived Competitive Advantage in the Food Industry, Oklahoma State University, [White Paper] 4301-4306.

[32] Chongxiang Zhu et al., A feature oriented approach to mapping from domain requirements to product line architecture, Journal of Computer Research and Development, 2007, 1-10.

[33] A. Mili, S. Fowler, R. Gottumukkala, and L. Zhang. Software reuse cost estimation. Technical report, CSEE Dept, West Virginia University, http://www.csee.wvu.edu/reuseroi/, Nov. 1999.