A Minimal Closure Algorithm for Quantifying Impact of Requirement Changes on Network Performance

Olusegun Folorunso, Adio T. Akinwale, Oluwole Adekanmbi
Department of Computer Science,
University of Agriculture, Abeokuta, Nigeria

ABSTRACT: In network systems design, it is often difficult to quantify the impact of network performance brought by the requirement changes because of the complexities and dependencies among the changed and unchanged requirements. In this paper, the relations of these dependencies such as bandwidth size, number of user etc are analyzed and established. The scope of the impact caused by requirement changes is identified using Kraft inequality and the impact was quantified. A minimal closure algorithm developed to quantitatively evaluate the effect of requirement changes is presented. Lastly, the feasibility of this evaluation algorithm is shown through a case study.
KEYWORDS: Requirement change, Dependency, minimal closure Algorithm, Network performance

1 Generalities

With the rapid growth of computer technology in all the areas of applications, which include streaming video or voice, bulk file transfer or e-mail, instant messaging etc, both the scale and complexities of the network performance management increases at an unprecedented rate. It is well known that the requirement changes have enormous influence on all aspect of network performances such as blends of capacity, latency, reliability and etc. For example, the recent introduction of new 458 Numbering Plan Area in the state of Oregon, presented additional network requirement for reading

E-911. Failure to address these changes may result in the inability of calls to complete to the OS/DA network [Aud06]. More so, in software requirement such as secure virtual Private Network allow the creation of authentication but not encryption [VPN08].

Similarly, in Boston College (BC), minimum network requirement access changes every year. For Boston College wireless network access, a wireless 802.11a/g/n- compliment network card with windows software is needed while /0/100/1000 Ethernet card and cable are also required for Boston College (BC) wired networks. Also, in BC, Macafee Boston College Anti-virus software must be installed before accessing internet which is an additional requirement change [req93].

Therefore, the evaluation of the impact attributable to requirements change has played a vital role in requirements change, and it can result in a positive impact on software project network performance management. The requirements changes, discussed in this paper, are modifications to existing requirements or new requirements that may or may not affect existing requirements. The evaluation of requirements change impact includes determination of the scope of requirements change and analysis of the potential influence brought by it [AB93] [YLM10]. It has been a problem of how to identify the scope of dependencies because the impact caused by dependency factor is rarely recognised in the existing evaluation methods which brings about uncertainty to the result.

In this paper, the problem is solved by analyzing and establishing the relation between these dependencies. The minimal closure algorithm is used to normalize redundant dependencies and establish the impact caused as a result of requirement change and the impact was quantified using Kraft inequality. Finally, the algorithm is shown through a case study of a network environment.

The rest of the paper includes requirements dependency in section 2, theoretical forms of requirement dependencies in section 3, identifying dependency in section 4, using Kraft inequality to quantify the impact of requirement change in section 5, case study in section 6 and finally conclusion in section 7.

2 Requirement Dependency

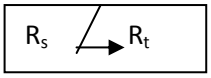
As a result of the dependencies between requirements, the change of some requirements would have an impact on correlated requirements, which result easily to the diffusion of impact. This phenomenon makes the evaluation of impact both uncertain and difficult, therefore, the key to resolving this issue is to correctly determine the scope of impact of changing requirements.

Definition 1: A requirement dependency is a relationship that signifies that the change of a single or a set of requirement elements requires the change(s) of other requirement elements for their specification or implementation. This means that the complete semantics of the depending elements is either semantically or structurally dependent on the definition of the source elements [spe10].

That a requirement R_t determines on the other requirement R_s is denoted as

Requirement Dependency (RD): $R_s \rightarrow R_t$

and the requirement R_t does not determine on the requirement R_s is denoted as



Requirement Dependency (RD):

where R_s is the source of requirements dependency, and R_t is the target of requirements dependency.

The requirements dependency is practically regarded as a description of relations among requirements at the coupling aspect, which can be divided into implicit dependency and explicit dependencies.

3 Theoretical Forms of Requirement Dependencies

The theoretical forms of requirement dependencies used in the work are transitivity and partially [Ber76] [GUW09].

3.1 Reason for Transitivity Requirement Dependencies

$$if \quad R_{a_1} R_{a_2}, ..., R_{a_n} \rightarrow R_{b_1} R_{b_2}, ..., R_{b_m} \quad and$$

$$R_{b_1} R_{b_2}, ..., R_{b_m} \rightarrow R_{c_1} R_{c_2}, ..., R_{c_k} \quad hold$$

in requirement R *then*

$R_{a_1} R_{a_2},...,R_{a_n} \rightarrow R_{c_1} R_{c_2},...,R_{c_k}$ *also holds*

in requirement R

To test whether $R_{a_1} R_{a_2},...,R_{a_n} \rightarrow R_{c_1} R_{c_2},...,R_{c_k}$ *holds we need to compute the closure* $\{R_{a_1} R_{a_2},...,R_{a_n}\}^+$

if $R_{a_1} R_{a_2},...,R_{a_n} \rightarrow R_{b_1} R_{b_2},...,R_{b_m}$ *then*

$R_{b_1} R_{b_2},...,R_{b_m}$ *are very useful in the closure of* $\{R_{a_1} R_{a_2},...,R_{a_n}\}^+$ *then we can use*

$R_{b_1} R_{b_2},...,R_{b_m} \rightarrow R_{c_1} R_{c_2},...,R_{c_k}$ *to add*

$R_{c_1} R_{c_2},...,R_{c_k}$ *to the closure* $\{R_{a_1} R_{a_2},...,R_{a_n}\}^+$

we conclude that $R_{a_1} R_{a_2},...,R_{a_n} \rightarrow R_{c_1} R_{c_2},...,R_{c_k}$ *hold for any requirement that satisfies both* $R_{a_1} R_{a_2},...,R_{a_n} \rightarrow R_{b_1} R_{b_2},...,R_{b_n}$ *and* $R_{b_1} R_{b_2},...,R_{b_n} \rightarrow R_{c_1} R_{c_2},...,R_{c_k}$

Reason of Partial Requirement Dependencies

if $\{R_{b_1} R_{b_2},...,R_{b_m}\} \subseteq \{R_{a_1} R_{a_2},...,R_{a_n}\}$ *then*

$R_{a_1} R_{a_2},...,R_{a_n} \rightarrow R_{b_1} R_{b_2},...,R_{b_m}$

Transitivity and partially dependencies may lead to double or more requirements

if $R_{a_1} R_{a_2},...,R_{a_n} \rightarrow R_{b_1} R_{b_2},...,R_{b_m}$ *then*

$R_{a_1} R_{a_2},...,R_{a_n} R_{c_1} R_{c_2},...,R_{c_k} \rightarrow R_{b_1} R_{b_2},...,R_{b_m} R_{c_1} R_{c_2},...,R_{c_k}$

for any set of requirements $R_{c_1} R_{c_2},...,R_{c_k}$. *Some of the requirements of* C *may also be in requirement* A*'s or* B*'s or both.*

4 Identifying Dependency Set

This paper uses the minimal cover algorithm and closure to remove redundancy among requirements dependency.

4.1 Description of Minimal Closure Algorithm

The minimal cover algorithm tends reduce the requirement dependency to its minimal requirement dependency which is as result of requirements changes. This algorithm helps to remove redundant requirement dependencies

4.2 Algorithm for Minimal Cover

Input: a set of requirement dependency, RD
Method:
RD_{min} := RD;
replace each RD: $X \rightarrow A_1, A_2, A_3, ..., A_k$ by
$\{X \rightarrow A_i / i = 1, ..., k\}$
for each RD: $X \rightarrow Y$ is in RD_{min} do
if $(RD_{min} - \{X \rightarrow Y\})^+ = RD_{min}^+$ then
delete $X \rightarrow Y$ from RD_{min}
for each RD: $X \rightarrow Y$ is in RD_{min} do
if $(RD_{min} - \{X \rightarrow Y\}\{X - A \rightarrow Y\})^+ = RD_{min}^+$ then
replace $X \rightarrow Y$ with $X - A \rightarrow Y$
merge RD's of the same L.H.S;
return RD_{min}

4.3 Algorithm for Closure

Input: *a set of components*: $\{A_1, A_2, ..., A_n\}$ *and a set of requirement dependencies* (*RD's*)

Output: *the closure* $\{A_1, A_2, ..., A_n\}^+$

1. *if necessary splits the RD's of S such that RD in S has a* sin*gle component on the right hand side*

2. *Let X be a set of components that eventually will become the closure*
   Initialize X to be $\{A_1, A_2, ..., A_n\}$

3. Re*peatedly search for some RD*

$$B_1, B_2, ..., B_m \rightarrow C$$

such that all of $B_1, B_2, ..., B_m$ *are in the set of components X, but C is not.*

Add C to the set X and repeat the search. Since X can only grow, and the number of components is finite, eventually nothing more can be added to X, and this step ends.

4. *The set X, after no more components can be added to it, is the correct value of* $\{A_1, A_2, ..., A_n\}^+$

4.4 Why Using Closure Algorithm for Requirement Changes of Components?

In step 3 *of closure a*lg*orithm of* sec*tion* 4.3

$B_1, B_2, ..., B_m \rightarrow C$ *holds for every component C in X*

That is every component R satisfying all of the RD in S also satisfies

$B_1, B_2, ..., B_m \rightarrow C$.

Basis: *The basis case is when there are zero steps.*

Then C must be one of $B_1, B_2, ..., B_m$ *and surely* $B_1, B_2, ..., B_m \rightarrow C$ *holds in component because it is a trivial requirement dependency.*

Induction: *For the induction,* sup*pose component C was added when we used the requirement dependency* $A_1, A_2, ..., A_n \rightarrow C$ *of S*

we know by the inductive hypothesis that requirement component satisfies

$B_1, B_2, ..., B_m \rightarrow A_1, A_2, ..., A_n$

Suppose two components of R *are required on all of* $B_1, B_2, ..., B_m$. *Then since* R *satisfies* $B_1, B_2, ..., B_m \rightarrow A_1, A_2, ..., A_n$, *the two components must satisfy* $A_1, A_2, ..., A_n$ *since* R *satisfies* $A_1, A_2, ..., A_n \rightarrow C$, *hence it satisfies* $B_1, B_2, ..., B_m \rightarrow C$.

5 Using Kraft Inequality to Quantify the Impact of Requirement Change

5.1 Impact Degree (ID)

The whole scope of impact affected by requirement change is obtained by minimal cover and closure algorithm. To quantify the degree of impact between requirements after the change, this paper introduced the Kraft inequality degree between 2 requirements elements. Kraft's inequality was published by Kraft [Leo49]. However, Kraft's paper discusses only prefix codes, and attributes the analysis leading to the inequality to Raymond Redheffer. The inequality is sometimes also called the **Kraft–McMillan theorem** after the independent discovery of the result by McMillan [Bro56]. McMillan proves the result for the general case of uniquely decodable codes, and attributes the version for prefix codes to a spoken observation in 1955 by Joseph Leo Doob.

More specifically, Kraft's inequality limits the lengths of codewords in a prefix code: if one takes an exponential function of each length, the resulting values must look like a probability mass function. Kraft's inequality can be thought of in terms of a constrained budget to be spent on codewords, with shorter codewords being more expensive.

i. If Kraft's inequality holds with strict inequality, the code has some redundancy.
ii. If Kraft's inequality holds with strict equality, the code in question is a complete code.
iii. If Kraft's inequality does not hold, the code is not uniquely decodable.

5.2 The Structure of Kraft Inequality Coding Tree

Weights are associated with the branch point of the trees as follows; starting with weight 1 at the root, the 2 nodes at level 1 each carries weight ½, the 4 nodes at level 2, each carry weight ¼ and so on. Figure 1 shows dependencies between requirements

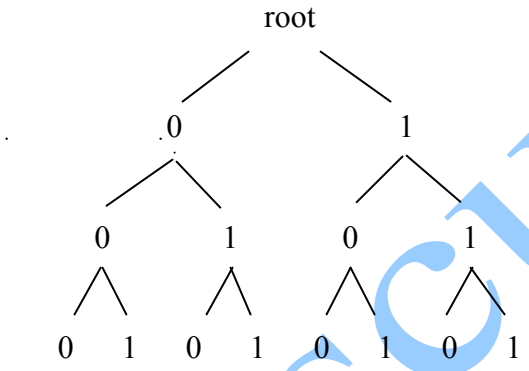


Figure 1. Dependencies among Requirements

The Impact Degree (ID) between two requirements in figure 1 is given as in figure 2.

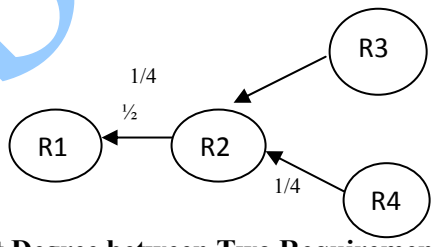


Figure 2. Impact Degree between Two Requirements

ID (R_1, R_2) =1/2
ID (R_1, R_3) = 1/4+1/2=3/4
ID (R_1, R_4) =1/2+1/4=3/4
ID (R_2, R_3) =1/4
ID (R_2, R_4) =1/4

5.3 Implementation of Evaluating Algorithm

i. We can obtain a whole set of impact at the initial state of dependencies. while some requirements changes take place, we need to evaluate the propagation of impact and impact degree on the requirement change as a result of minimal cover
ii. To extract the minimal cover of the requirement dependencies
iii. To eliminate redundant dependencies of change impact, this is to get minimal subset of dependencies.
iv. To quantify and evaluate impact degree (ID) of initial and minimal requirement dependencies.

6 Case Study

In this section we propose an example of the requirement dependencies needed for the performance of a network. We illustrated the minimal cover and closure algorithm to deduce the minimal requirement and the Kraft inequality is used to quantify the impact. All these have been discussed in the previous section. In a network environment, we extract a set of requirements and dependencies as follows:

i. Volume of Packet Sent (V_P)
ii. Network Distribution Tools (N_D)
iii. Bandwidth Size(B_S)
iv. Network Performance (N_P)
v. Network Policy (N_O)
vi. Number of User (N_U)

Considering the requirement, we notice that some requirements will be dependent on other such that there exist a schema for the given requirements which is as follows;

R (V_P N_D B_S N_P N_O N_U)

The initial requirement dependencies are as follows;

$$\mathbf{N_U \rightarrow V_P}$$
$$\mathbf{N_D N_U \rightarrow N_P}$$
$$\mathbf{B_S \rightarrow V_P N_D}$$
$$\mathbf{N_U \rightarrow B_S}$$

$$\mathbf{V_P \rightarrow N_D}$$
$$\mathbf{B_S N_O \rightarrow N_P}$$

Following the minimal cover algorithm

Step 1: to make the Left Hand Side (LHS) atomic

Replacing the requirement dependency with multiple requirement on the Right Hand Side (RHS) with equivalent set of requirement dependency with R.H.S containing only one requirement Minimal Cover Algorithm

Replace such RD $X \rightarrow \{A_1, A_2...A_n\}$ in R by RD's $X \rightarrow A_1$, $X \rightarrow A_2$,....,$X \rightarrow A_n$

$$\mathbf{N_U \rightarrow V_P}$$
$$\mathbf{N_D N_U \rightarrow N_P}$$
$$\mathbf{B_S \rightarrow V_P}$$
$$\mathbf{B_S \rightarrow N_D}$$
$$\mathbf{N_U \rightarrow B_S}$$
$$\mathbf{V_P \rightarrow N_D}$$
$$\mathbf{B_S N_O \rightarrow N_P}$$

Step 2

Remove any redundant RD's, using the same minimal cover algorithm

For each RD $X \rightarrow A$ in R

{

Compute X^+ with respect to the set of dependencies$(R-(X \rightarrow A))$

If X^+ containing A, then $R=(R-(X \rightarrow A))$

}

The closure algorithm discussed earlier is used compute X^+

1. For $\mathbf{N_U \rightarrow V_P}$, compute $\mathbf{N_U^+}$ under $\mathbf{(R-(N_U \rightarrow V_P))}$
   $\mathbf{N_U^+ = N_U B_S V_P N_D N_P}$
   Since V_P is in N_U^+, so $\mathbf{R = R - (N_U \rightarrow V_P)}$
2. For $\mathbf{N_D N_U \rightarrow N_P}$, compute $\mathbf{N_D N_U^+}$ under $\mathbf{(R-(N_D N_U \rightarrow N_P))}$
   $\mathbf{N_D N_U^+ \rightarrow N_D N_U B_S V_P}$
   Since $\mathbf{N_P}$ is not in $\mathbf{N_D N_U^+}$, then $\mathbf{R = R}$
3. For $\mathbf{B_S \rightarrow V_P}$, compute $\mathbf{B_S^+}$ under $\mathbf{(R-(B_S \rightarrow V_P))}$
   $\mathbf{B_S^+ = B_S N_D}$
   Since V_P is not in B_S^+, then $\mathbf{R=R}$
4. For $\mathbf{B_S \rightarrow N_D}$, compute $\mathbf{B_S^+}$ under $\mathbf{(R-(B_S \rightarrow N_D))}$

$B_S^+ = B_S N_D V_P$
Since N_D is in B_S^+, then $R = R - (B_S \rightarrow N_D)$

5. For $N_U \rightarrow B_S$, compute N_U^+ under $(R-(N_U \rightarrow B_S))$
$N_U^+ = N_U$
Since B_S is not in N_U^+, then $R=R$

6. For $V_P \rightarrow N_D$, compute V_P^+ under $(R-(V_P \rightarrow N_D))$
$V_P^+ \rightarrow V_P$
Since N_D is not in V_P^+ then $R=R$

7. For $B_S N_O \rightarrow N_P$, compute $B_S N_O^+$ under $(R-(B_S N_O \rightarrow N_P))$
$B_S N_O^+ \rightarrow B_S N_O N_D V_P$
Since N_P is not in $B_S N_O^+$ then $R=R$

From this minimal cover algorithm, the minimal requirement dependency is:

$$N_D N_U \rightarrow N_P$$
$$B_S \rightarrow V_P$$
$$N_U \rightarrow B_S$$
$$V_P \rightarrow N_D$$
$$B_S N_O \rightarrow N_P$$

Step 3:
Check if any requirement on the L.H.S of the RD can be removed

1. For $N_D N_U \rightarrow N_P$
For N_D: Compute N_U^+ under $(R - \{N_D N_U \rightarrow N_P\} \cup \{N_U \rightarrow N_P\})$
$N_U^+ = N_U B_S N_P V_P N_D$
N_D is in N_U^+, so N_D is a redundant requirement, then
$R = R - \{N_D N_U \rightarrow N_P\} \cup \{N_U \rightarrow N_P\}$

2. For $B_S N_O \rightarrow N_P$
For B_S: Compute N_O^+ under $(R - \{B_S N_O \rightarrow N_P\} \cup \{N_O \rightarrow N_P\})$
$N_O^+ = N_O$
B_S is not in N_O^+, so B_S is not a redundant requirement
For N_O: Compute B_S^+ under $(R - \{B_S N_O \rightarrow N_P\} \cup \{B_S \rightarrow N_P\})$
$B_S^+ = B_S V_P N_P N_D$
N_O is not in B_S^+, so N_O is not a redundant requirement

From this minimal cover algorithm, the minimal requirement dependency is:

$$N_U \rightarrow N_P$$
$$B_S \rightarrow V_P$$

$$N_U \rightarrow B_S$$
$$V_P \rightarrow N_D$$
$$B_S N_O \rightarrow N_P$$

To quantify the impact degree of the initial requirement dependency:
Using Kraft inequality, we quantify the impact of the initial requirement dependency which is called the **IMPACT DEGREE (ID)**

$$N_U \rightarrow V_P$$
$$N_D N_U \rightarrow N_P$$
$$B_S \rightarrow V_P$$
$$B_S \rightarrow N_D$$
$$N_U \rightarrow B_S$$
$$V_P \rightarrow N_D$$
$$B_S N_O \rightarrow N_P$$

The transition diagram for the initial requirement dependency is given in figure 3

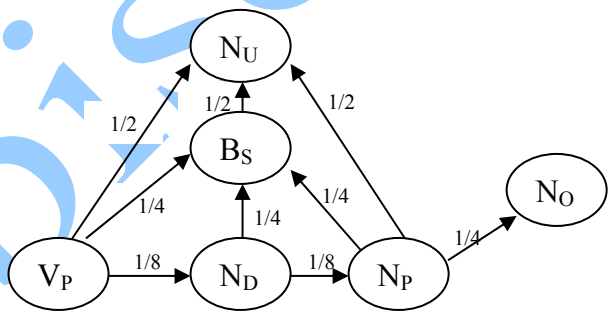


Figure 3: Transition Diagram for Initial Requirement Dependency

Using the transition diagram in figure 3, the impact degree for the initial requirement dependency is shown in table 1.

Table 1. Impact Degree of initial requirement dependency

R_S	N_U	N_P	N_O	V_P	N_D	B_S
R_T	B_S	N_U	N_P	N_O	V_P	N_D
$ID(R_S,R_T)$	½	0	1/4	1/4	1/8	1/4

To quantify the Impact degree of the minimal requirement dependency as a result of requirement change:

Using the kraft inequality, we quantify the impact of the minimal requirement dependencies which is referred to as **IMPACT DEGREE (ID)**

Due to minimal cover algorithm, the initial requirement dependency is normalized to

$$\mathbf{N_U \rightarrow N_P}$$
$$\mathbf{B_S \rightarrow V_P}$$
$$\mathbf{N_U \rightarrow B_S}$$
$$\mathbf{V_P \rightarrow N_D}$$
$$\mathbf{B_S N_O \rightarrow N_P}$$

The transition diagram of minimal requirement dependency is given in figure 4.

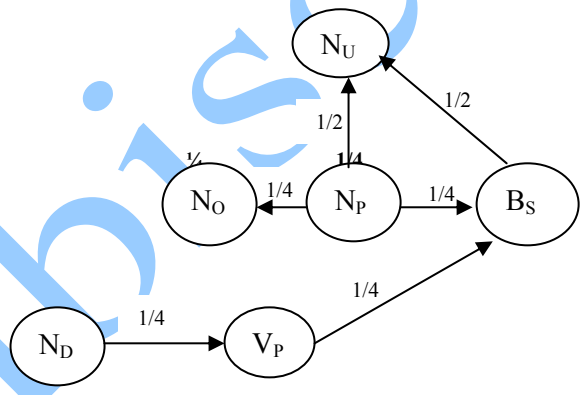


Figure 4. Transition Diagram of Minimal Requirement Dependency

Using the transition diagram in figure 4, the impact degree for the minimal dependency is shown in the table 2.

Table 2: Impact Degree of Minimal Requirement Dependency

R_S	N_U	N_P	N_O	V_P	N_D	B_S
R_T	B_S	N_U	N_P	N_O	V_P	N_D
$ID(R_S,R_T)$	½	½	1/4	0	1/4	1/8

6.1 Comparing the two tables above, some conclusions are drawn:

In spite of the inaccuracy of measure approach, the results derived from the tables show that the minimal requirement dependency as a result minimal cover and closure algorithm requirements change has a greater positive impact degree and also ascertain much more than the initial requirement dependency that the number of user in a network determines the amount of bandwidth size required to increase the performance of a network i.e ($\mathbf{N_U \rightarrow B_S}$ and $\mathbf{N_U \rightarrow N_P}$) or amount of bandwidth size required and performance of a network is dependent on number of user and i.e. $\mathbf{dep(N_U,B_S) \cap dep(N_U,N_P) \rightarrow dep(N_U,B_S N_P)}$

Conclusion

This work discussed the impact on network performance caused by requirement change at the aspect of requirement dependency. Minimal cover and closure algorithms were used to evaluate requirement changes based on the dependency. A Kraft inequality was employed to quantify the impact of requirement change on network. The result showed that the algorithms had a greater positive impact degree than assumption requirement dependency.

References

[Aud06] **Audience** - *CLCC Research, ISP IX6,* Wireless Oregon Verizon, USA, 2006

[AB93] **R. S. Amold, S. A. Bohner** - *Impact Analysis-towards a Framework for Comparison,* Proceeding of the International Conference on Software Maintenance, pg 292-301 1993.

[Arm71] **W. W. Armstrong** – *Dependency Structures of Database Relationships, Proceeding of the 1971 IFIP Congress,* pg 580-582, 1971

[Ber76] **P.A. Bernstein** – *Synthesizing Third Normal Form Relations from Functional Dependencies,* ACM Transaction on Database Systems, Vol 1 (4), pg 277-298, 1976

[Bro56] **M. Brockway** – *Two Inequalities implied by Unique Decipherability*, IEEE Trans. Information Theory, Vol 2 (4), pg 115–116, IEEE Computer Society 1956

[CS01] **P. Carlsharmre, K. Sandahl** - *An Industrial Survey of Requirements Interdependencies in Software Product Release Planning,* Proceedings of Fifth IEEE International Symposium on Requirements Engineering, pg 84-91, IEEE Computer Society, 2001.

[EN94] **R. Elmasri, S.B. Navathe** – *Fundamental Database Systems,* Benjamin Publishing Company, 1994

[GUW09] **H. Garcia-Molina, J. D. Ullman, J. Widom** – *Database Systems*, Addison-Wesley Prentice Hell, 2009

[GV02] **J. Giesen, A. Volker** – *Requirement Interdependencies and Stakeholders Perferences,* Proceedings of IEEE Joint International Conference on Requirements Engineering, pg 206-209, IEEE Computer Society, 2002

[Leo49] **K. Leon** – *A Device for Quantizing Grouping and Coding Amplitude Modulated Pulse*, M.Sc Thesis, Electrical Engineering Department, Massachusetts Institute of Technology, Cambridge, 1949

[LK98] **S. Lock, G. Kotonya** - *Requirement Level Change Management and Impact Analysis*, http://info.comp.lancs.ac.uk/publications/Publication_Documents/1998-Lock-Internal.pdf

[RJ01] **B. Ramesh, M. Jarke** - *Toward Reference Models for Requirements Traceability*, IEEE Transactions on Software Engineering, vol 27(1), pg 58-93, IEEE Computer Society, 2001

[req93] http://www.bc.edu/office/help/gestated/network/require_min.html

[SS96] **M.R. Strens, R.C. Sugden** - *Change Analysis*, *A Step Towards Meeting the Challenge of Changing Requirements*, Proceedings of the IEEE Symposium and Workshop on Engineering of Computer Based Systems (ECBS), 1996.

[spe10] Specification of UML 2.0 – http://www.uml.org

[VPN08] VPN Technologies – Definitions and Requirements, VPN Consortium, http://www.vpuc.org/vpu-technologies.html, 2008

[WD04] **L. Wen, R.G. Dromey** - *Requirements Change to Design Change*: *A FormalPath. Software Engineering and Formal Methods,* Proceeding of the Second international Conference, 104-113, 2004

[YLM10] **H. Yang, Z. Liu, Z. Ma** - *An Algorithm for Evaluating Impact of Requirement Change,* Journal of Information and computing Science. pg 48-54, 2010