International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:11, No:1, 2017

Impact Analysis Based on Change Requirement Traceability in Object Oriented Software Systems

Sunil Tumkur Dakshinamurthy, Mamootil Zachariah Kurian

***Abstract*—**Change requirement traceability in object oriented software systems is one of the challenging areas in research. We know that the traces between links of different artifacts are to be automated or semi-automated in the software development life cycle (SDLC). The aim of this paper is discussing and implementing aspects of dynamically linking the artifacts such as requirements, high level design, code and test cases through the Extensible Markup Language (XML) or by dynamically generating Object Oriented (OO) metrics. Also, non-functional requirements (NFR) aspects such as stability, completeness, clarity, validity, feasibility and precision are discussed. We discuss this as a Fifth Taxonomy, which is a system vulnerability concern.

***Keywords*—**Artifacts, NFRs, OO metrics, SDLC, XML.

I. Introduction

CHANGE requirement engineering is a challenging task to maintain SDLC. The standard definition of traceability, as it is a relationship between two artifacts. In spite of two types of traceability, such as upstream and downstream traceability, starting from vision documents such as features to the software requirement and to test cases and test cases to software requirements were realized as shown in Fig.1.

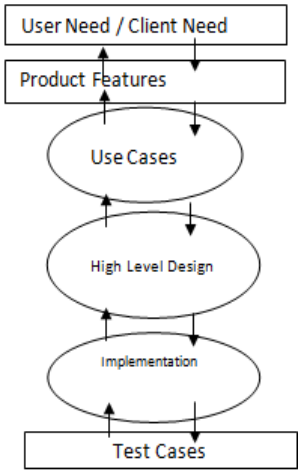

Fig.1 A model of upstream and downstream traceability

According to Leffingwell and Widrig [1], there are traceability matrices to trace the needs of the client from the requirements documents to the test cases, known as static traceability matrices. For high level of abstraction, the traceability matrix does not give dynamic links from the client needs to test cases. This paper discusses the implementation of the dynamic traceability matrix using XML to dynamically vary the high-level design changes with respect to change in user needs or client needs. Agile methodology is implemented for the dynamic traceability matrix which traces between the different forms of requirements that are the requirement specifications, design, implementation, and test cases. Also, this paper discusses the impact analysis of non-functional requirements such as stability, completeness, clarity, validity, feasibility and precision.

Sunil Tumkur Dakshinamurthy is a Ph.D candidate at Sri Siddhartha Institue of Technology, Karnatka, 572105, Tumkur, India (phone: +91 816 2200999, fax:+91 816 2200270, e-mail: suniltd@ ssit.edu.in,).
Mamootil Zachariah Kurian is with the Department of Electronics, Sri Siddhartha Institute of Technology, Tumkur, Karnataka, India (e-mail: kurianmz@ssit.edu.in).

II. Definitions

In object oriented systems we can consider features like object, classes, encapsulation, polymorphism and inheritance. These are explained in brief along with the concepts of requirement, a *non-functional* requirement, XML, Altova XMLSpy, and XML Schemas.

A. Object

Objects can be concrete such as a file system or conceptual such as scheduling policy in a multiprocessor operating system. Objects serve the purpose of understanding the real world and decomposing a problem into objects that depend on the nature of problem. An object can be an instance of a class.

B. Classes

The object with the same data structure (attributes) and behavior (operations) are grouped into a class. The following points on classes can be noted:

1) A Class is a blueprint that unites data and properties,
2) A Class is an abstraction of the real-world entities with similar operations,
3) A Class is an implementation of ADTs and
4) A Class identifies a set of similar objects.

C. Encapsulation

Encapsulation is nothing but data hiding. Here, it keeps data and the code safe from external interference and misuse.

D. Polymorphism

In the real world, the meaning of an operation varies with context and the same operation may behave differently in different situations. Polymorphism is achieved in many ways through function overloading, operator overloading and dynamic binding.

E. Inheritance

Inheritance allows new classes to be built from older and less specialized classes instead of being rewritten from scratch. In the real world, an object is described using inheritance. It derives general properties of an object by tracing an inheritance tree from one specific instance, upwards towards the primitive concepts at the root. The technique of building new classes from the existing classes is called inheritance.

F. Requirement

Requirement is derived directly from user depending on user needs. It may be a formally imposed document such as contract, standard or specification.

G. Non-Functional Requirement

A *non-functional* document can be viewed, it emphasizes on "How" an actual system "will do" rather than "what" the system "will do". From these definitions, we conclude that NFRs have features that define which, what kind of, or how many.

H. Extensible Markup Language (XML)

XML simplifies sharing, transport, changes of platform and availability of data. XML is not dependent on software and hardware for sharing, storing and transporting of data. XML gives importance to what data does. Most XML works as expected even if data is added or removed.

İ. Altova XMLSpy Interface

The XMLSpy interface is categorized into three vertical areas. These three areas contain, from left to right; 1. Project and information window; 2. Main and output window; and 3. Entry helper window.

J. XML Schema

An XML Schema describes the structure of an XML document. XML schema is to check whether it conforms to the requirements specified in the schema.

III. Literature Survey

First, different traceability techniques study were made by Bashir et al. [2] on origin-requirement traceability, requirement-requirement traceability, requirements-other artifacts traceability, other artifacts-other artifacts traceability. Chaumun et al. [3] discusses about the (OO) system to compute the impact of changes made to classes of the system, as this approach discusses different (OO) metrics for the high level design. Glinz [4] proposes the conceptual discrepancies especially characteristics or properties, attribute, quality, constraint and performance of requirement management. Non-functional traceability analysis is not discussed in detail with reference to high level design. Kang et al. [5] develop a representation formally that represents traceability between features and requirements at platform level. Rempel et al. [6] analyzed the quality and sustainability of a projects traceability strategy, however non-functional requirement impact on change requirements are loosely coupled in the project which has been analyzed. Ali et al. [7] approach is to have informational retrieval between text documentation and source code. Here, vector space model export adding knowledge to the traceability links extracted from CVS change logs. There is no statistical significance in the precision of the traceability links for high level design. Chung et al. [8] reminds about the non-functional framework in action, acquiring domain knowledge, identifying NFRs, dealing with priorities, recovering design rational, and evaluating the impact of decision and correlations. Also, discusses the types of NFRs, accuracy requirement, security requirement, performance requirements. Lastly, operationalization methods were pointed out. Khan et al. [9] realized the model of object oriented design for analyzability for NFRs. Somerville [10] classifies NFRs schema. Here, non-functional requirements have been addressed in international standards as part of the software and systems quality initiative. Both the earlier standard and its replacement include non-functional requirements, definitions and how to measure them as part of a system endeavor. Sunil et al. [11] provides a non-functional traceability analysis for change requirements in the software specification.

A. Identification and Organization of Non-Functional Requirements

De Weck et al. [12] discuss the relationship between non-functional requirements and what they term *ilities*. According to him, the complexity of modern systems and the scale of their deployments and the important side effects of their ubiquitous presence in the modern era provide an excellent discussion of the history associated with the expansion of the four classic systems *ilities* – Safety, quality, usability, and reliability.

B. Models for Non-Functional Requirements

i. Bohem's Software Quality Initiative
ii. Rome Air Development Center Quality Model
iii. Cavano and McCall's Model
iv. McCall's and Masumoto's Factor Model Tree

C. Introduction to Stability, Completeness, Clarity, Validity, Feasibility, Precision

Notational Taxonomy of NFRs for systems considers 27 NFRs as standards. The framework for the NFR Taxonomy has four concerns 1) System Design Concern, 2) System Adaptation Concern, 3) System Viability Concerns, and 4) System Sustainment Concern. However, here we are concerned about 5) System Vulnerability Concern, the fifth in the taxonomy of NFRs since stability, completeness, clarity, validity, and precision also has an impact on the requirement changes.

D. Measurement Scale for Traceability

We can construct traceability based on construct according to Adams [13], measurement attributes and an appropriate scale type. Here to evaluate traceability, it is necessary to answer the questions (yes or no) and the quality of the effort (how well) to provide traceability in the nine areas such as

IEEE std 1220 section 5.1.13, IEEE std 1220 section 5.2.1.1, 5.2.1.2, IEEE std 1220 section 5.3.1.1 (Detailed design), IEEE std 1220 section 6.3.1 (Functional analysis), IEEE std 1220 section 6.5.1, 6.5.18 (Design Synthesis) and IEEE std 1220 section 6.6.2.1, 6.6.8 (Design verification). The generalized equation for system traceability is given as:

$$T_{sys} = \sum_{i=1}^{n} T_i \quad (1)$$

The expanded equation for system traceability:

$$T_{sys}=T_1 + T_2 + T_3 + T_4 + T_5 + T_6 + T_7+T_8+T_9$$

Measurable characteristics – 1) Tcd – Conceptual design, 2) (Tpd1, Tpd2) – preliminary construct, 3) Tdd – detailed design, 4) Tfa – functional analysis, 5) (Ts1, Ts2) – Design synthesis, 6) (Tv1, Tv2) – Verification. The summation of the constructs (nine) will be the measure of the degree of traceability in a system design endeavor in Table I.

TABLE I
TRACEABILITY MEASUREMENT QUESTION LIKERT SCALE

Sl. No.	Measure	Descriptor	Measurement Criterion
1	0.0	None	No objective quality evidence is present
2	0.5	Limited	Limited objective quality evidence is present
3	1.0	Nominal	Nominal objective quality evidence is present
4	1.5	Wide	Wide objective quality evidence is present
5	2.0	Extensive	Extensive objective quality evidence is present

We should be able to measure non-functional attribute and also essential in system design endeavors. The four level construct for traceability is presented in Table II.

TABLE II
FOUR-LEVEL STRUCTURAL MAP FOR MEASURING TRACEABILITY

Sl.No	Level	Role
1	Concern	System design
2	Attribute	Traceability
3	Metric	System traceability
4	Measurable characteristic	Traceability of (1) conceptual design(T_{cd}), (2)preliminary design(T_{pd1},T_{pd2}), (3) detailed design (T_{dd}), (4) functional analysis(T_{fa}), (5) design synthesis (T_{s1},T_{s2}), and (6) verification (T_{v1},T_{v2})

IV. XML PARSER FOR HIGH LEVEL TRANSLATION

A Behavior tree from Wen et al. [14] is a formal, tree-like graphical form that represents behavior of individual. Here a traceable relation between the natural representation and its formal specification is concerned, for example, translation is carried out on a sentence-by-sentence or word-by-word basis.

A. Translation of Requirements

The first step is to translate the requirements in the software engineering design process so that step cannot be fully automated as the requirements are loosely coupled to the design, especially high-level design. Hence, markup language provides the links between the requirements which are in sequence and high-level design which are in synchronization.

B. Requirement Integration

- Functional requirement is translated into one or more corresponding requirements behavior tree(s) RBT.
- We can systematically and incrementally construct a design behavior tree (DBT) that will satisfy all its requirements by integrating the requirements.

C. Traceability in Software Engineering

- The first step is to translate functional requirements in RBT the other step is linking the RBT to design flow which may be fully automated or partially automated using XML Parsers.
- Before we create links through XML parsers, first we have to generate the edit behavior tree (EBT) which has the following steps:
- Start comparison with root node. Because root nodes exist in both the trees (old tree and new tree).
- Mapping is done to the child node set in both the trees i.e., old tree and in the new tree.
- In the old tree, the sub tree under the old node will be generated in the EBT as old.
- If a node exists in the new tree's child node set but not in the old tree's child node set, this node will be created in the EBT as a new node.
- In the new tree, the sub tree under the new node will be generated in the EBT as new.
- If a node exists in the child node sets of both trees, it will be generated in the EBT as an unchanged node.
- An unchanged node will be a new comparison node and the steps will go back recursively.

D. Transformation Rules for Change Requirements

There are different transformation rules "NEW to "NEW", "OLD" to "OLD", "UNCHANGED" to "UNCHANGED", "NEW MERGED" to "NEW EQUALS".

V. RESULTS

The attributes of supplementary requirements or NFRs fifth taxonomy (System Vulnerability Concern) are stability, completeness, clarity, validity and precision, which also has an impact on the requirement changes.

A. Stability

Stability is the probability that the feature will not change during the project. Another attribute that is applicable to supplementary requirements is stability shape. This attribute describes how customers' stability changes with the fulfillment of required metrics. Stability shape may have the following values:

Sharp. The metrics used in the requirement shall be fulfilled exactly as described. Imagine a real-time system for sorting packages. Packages are moving on the conveyor belt. The system scans an address label on the package and, based on the destination, instructs the diverter to divert to an

appropriate belt. The requirement is that the system shall calculate the diverter's proper action within one second. If it fails, the package moves past the diverter. In this case the response time shall be less than one second; otherwise, the whole system will not work. However, it does not matter if this is done in 0.99 seconds or 0.5 seconds. There is no additional value in having a shorter response time. Fig. 2 shows the stability shape for this requirement. For all values less than one second the stability is one, and for all values greater than one second the stability is zero.

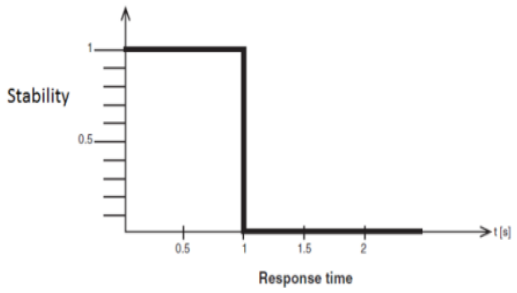

Fig. 2 Stability shape

B. Completeness

Completeness is the quality or state of being without restriction, exception, or qualification or it is nothing but perfectness. Fullness of the requirement changes satisfies the completeness of the quality of the project. Fig. 3 shows the characteristics of completeness of requirements.

- CMP=(NARS or NORS) – NIR
- CMP=Completeness of Requirements
- NARS or NORS = Number Of Actual Requirements or Number of Original Requirements
- NIR= Number of Incomplete Requirements

There are two cases or possibilities of Completeness of requirements:

1) Nearing to completeness of the requirements
2) Away from the completeness of the requirements

First Case. For example, if there are 10 Numbers of actual requirements or 10 Numbers of original requirements and 0 Number of incomplete requirements, we say that the requirement document is complete. If we take another example, 10 Numbers of original requirements and 10 Numbers of incomplete requirements then we say that the requirements are further away from completeness. If we take another example, 10 Numbers of the original requirements and 5 Numbers of incomplete requirement, then we can say that the requirements are nearing to the original requirements.

Second Case. For example, there are 0 Number of original requirements and 10 Numbers of incomplete requirements; we say that the requirement document is incomplete. If we take another example, 0 Number of the original requirements and 5 Numbers of incomplete requirements, then we say that the requirements are away from the completeness or total non-incomplete requirements. If we take another example, 0 Number of original requirements and 10 Numbers of incomplete requirements, then we say that the requirements is further away from the completeness or to the total non-incomplete requirements.

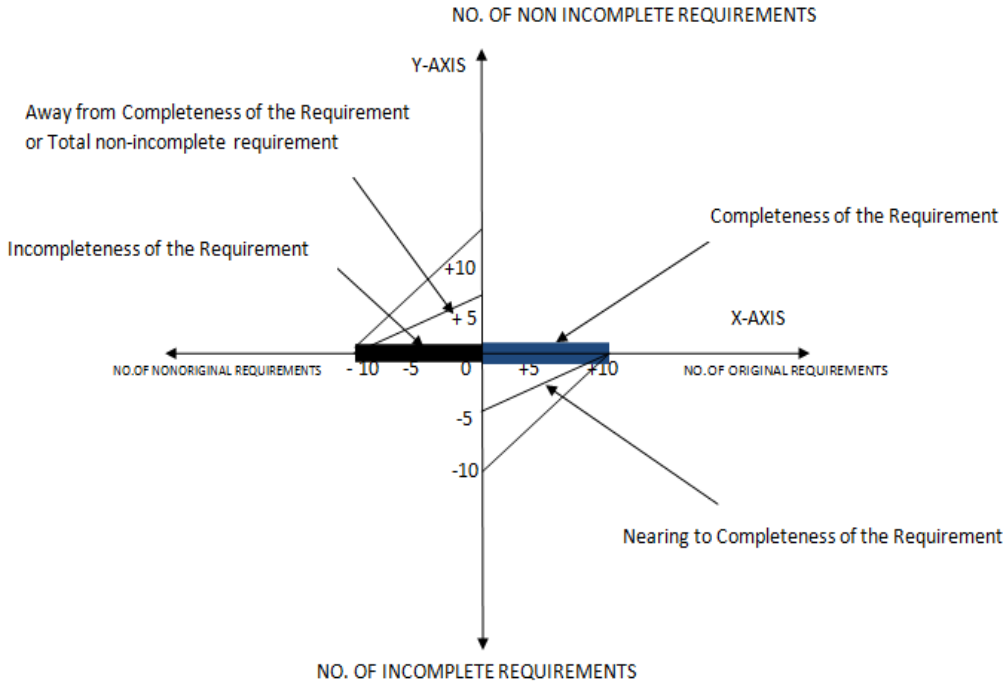

Fig. 3 Characteristics of completeness of requirements

C. Clarity

Clarity is the quality of being easily understood meaning there is no unclear requirements and the requirements are clearly understood.

- CL=NARS-NIR-UCLR
- CL=Clarity of the system
- NARS=Number or Actual Requirements or Number of Original Requirements
- NIR= Number of Incomplete Requirements in the system
- UCLR= Number of Unclear Requirements

D. Validity

Validity is the executed with the proper legal authority and formalities of the requirements.

E. Feasibility

Feasibility means capable of being used or dealt with successfully the requirements. If the development team does not implement then there is no value for requirements. There are many constraints for feasibility, Budget constraints, Hardware or software constraints, Staff head count, Professional experience, Schedule constraints.

- FR=IFR-UCLR
- FR=Feasibility requirements of the system
- UCLR=Number of Unclear Requirements

F. Precision

Precision means the quality of being precise: Exactness or accuracy of the requirements. Fig. 4 shows the precision of change requirements.

- PR=CMP+CL+FR
- PR=Precision requirements of the system
- CMP=Completeness of the system
- CL=Clarity of the system
- FR=Feasibility of the system

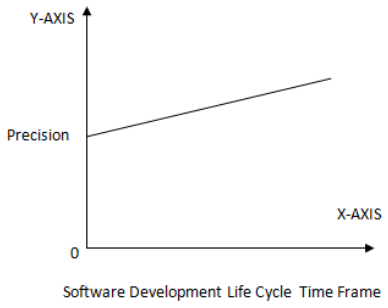

Fig. 4 Characteristics of Precision of Software Systems

VI. Conclusion

Here, in this work we conclude that high level design changes in software systems can be automated or semi-automated by dynamically linking the artifacts by using XML parsers that can create links between the requirement and high-level design; also, analyzed with the example of the impact of the non-function requirements such as stability, completeness, clarity, validity, and precision. It can be concluded that this fifth taxonomy for NFRs can be included as it is System Vulnerability Concern.

Acknowledgment

We would like to thank Peter Zielczynski of IBM Requisite Pro given us the permission to reference "Flight Booking System" Requirements and also Kevin MacG. Adams "Topics in Safety, Risk, Reliability and Quality" in "Non-Functional Requirements in Systems Analysis and Design" Springer and Altova XML Spy software (2016) for dynamically linking the artifacts of requirements to High Level design for taking as reference in this paper.

References

[1] Dean Leffingwell and D. Widrig "The Role Requirements Traceability in System Development", Rational Software 2002.
[2] Mohamamd Farahan Bashir, Muhammad Abdul Qadir, "Traceability Techniques: A Case Study", IEEE 2006, pp.265-268.
[3] M. Ajmal Chaumun, Hind Kabaili, Rudolf K. Keller and Francois Lustman, "A Change Impact Model for Changeability Assessment in object-Oriented Software System", NSERC and NSC, Canada 2000.
[4] Martin Glinz, "On Non-Functional Requirements", Proceedings of the 15th IEEE International Requirements Engineering Conference, Delhi, India,2007.
[5] S. Kang, "A Formal Requirementation of Platform Feature-to-Requirement Traceability for Software Product Line Development," Computer Software and Applications Conference (COMPSAC),2014, IEEE 38th Annual Conference.
[6] P. Rempel, "An empirical study on project specific traceability strategies" IEEE International Conference, Requirements Engineering Conference (RE), 2013.
[7] N. Ali, "Trust-Based Requirements Traceability" IEEE 19 th International Conference on Program Comphrension (ICPC).
[8] Lawrence Chung, Brian A. Nixon, Eric Yu John Mylopoulos – Non-Functional Requirements in Software Engineering, Springer Scinece + Business Medica, LLC.
[9] Suhel Ahmad Khan, Raees Ahmad Khan, "Analyzability Quantification Model of Object Oriented Design," ScienceDirect, Procedia Technology 4(2012)536-542, doi:10.1016/j.protcy.2012.05.085.
[10] I. Somerville. (2007) Software engineering (8th ed.). Boston: Pearson education.
[11] T D Sunil, M Z Kurian, "A Methodology to Evaluate Object-Oriented Software Systems Using Change Requirement Traceability Based on Impact Analysis", International Journal of Software Engineering & Application (IJSEA), Vol.5, No.3, May 2014.
[12] De. Weck, O.L Roos, D., & Magee, C.L. (2011) Engineering systems: Meeting human need in a complex technological work, Cambridge: MIT Press.
[13] Kevin MacG. Adams "Topics in Safety, Risk, Reliability and Quality", Non-functional Requirements in Systems Analysis and Design, Springer International Publishing Switzerland 2015.
[14] Lian. Wen, R Geoff. Dromey "From Requirements Change to Design Change: A Formal Path", World Appl.Sci.J.28(10):1366-1374,2013.