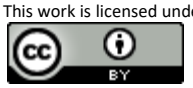
Available online at: https://ijact.in

Date of Submission	22/01/2020
Date of Acceptance	15/02/2020
Date of Publication	02/03/2020
Page numbers	3578-3585 (8 Pages)

Cite This Paper: Y Al-Kasabera, W. Alzyadat, Aysh A, S Al Showarath, Ahmad T. An automated approach to validate requirements specification, 9(2), COMPUSOFT, An International Journal of Advanced Computer Technology. PP. 3578-3585.



ISSN:2320-0790

An International Journal of Advanced Computer Technology

AN AUTOMATED APPROACH TO VALIDATE REQUIREMENTS SPECIFICATION

Yazan Al-Kasabera, Wael Alzyadat, Aysh Alhroob, Suleyman Al Showarah, Ahmad Thunibat
Department of Software Engineering
Isra University, Jordan

Abstract: Requirements engineering processes aim to acquire functions, services and constraints. These processes are important to satisfy the customer by applying correctness, completeness through consistency according to the control instructions to achieve product quality. Both functions and services face changeability issue that is hard to regulate, depending on the precise request of the customer. This research addresses the achievement of correctness, completeness, and consistency by applying an automated approach. The evaluation is established using a standard use case diagram from the UML official website. The proposed approach detects the incorrect requirement specifications to enhance Software quality. The proposed approach includes three levels; the first level is the Structured Document, the second level is the Dynamic Language, which describes the transforming of use case diagram as dynamic, and the third level is the completeness checking procedures, which is based on the implemented standard rules. The approach is supported by a programmed tool on MS excel and XML due to IBM Rational Rose and Visual Paradigm and experimented "Online Shopping" use case diagram as a case study.

Keywords: Correctness, Completeness, Consistency, Requirements Specification, Use Case Diagram, Quality

I. INTRODUCTION

The terms; Correctness and Completeness (C&C) are two sides of one coin, which are involved in several software phases and will be defined relying on their processes that exist. The initial phase in the software development process is requirements that deals between customer and software developer, intended to system domain and how to establish the product, involving abstraction and invisibility issues that appear clearly in quantification and measurement. Likewise, there are also complicated efforts to describe and organize software, in ways that will facilitate services change during the processes of their design, implementation, testing, and maintenance.

Addressing incompleteness negatively influences the quality of produced artefact via requirements model, design model or the other software components through the methodologies that applied the investigated completeness issue in software engineering (Coughlan & Macredie, 2002; Eckhardt, Vogelsang, Femmer, & Mager, 2016; Zowghi & Coulin, 2005), especially, in requirements phase is that quality requirements engineers; incorrectly describe how to build the system more than its functionality (Swarnalatha, Srinivasan, Dravid, Kasera, & Sharma, 2014).

This research focuses on three steps in requirements phase; First, user requirements, which is collecting the services that the customer needs in the system, presented by high-level requirements, as well called raw requirements (Düchting, Zimmermann, & Nebe, 2007; Swarnalatha et al., 2014).

Second, Requirements elicitation is capturing the requirements which rely on the emergent and collaborative view of requirements, elicitation and communication are both required to encompass the user to reduce error-prone requirements that came from early-stage from the user, as well as the purpose of requirements elicitation are to ensure successful requirements gathering (Coughlan & Macredie, 2002).

Third, Software Requirements Specification (SRS) stage named system requirements is a detailed description of what the system should do, which are derived from the user requirements, and modelled using formal or semi-formal methods and languages (dos Santos Soares & Vrancken, 2008). It further discussed the requirements evaluation and requirements prioritizing stages of the requirements phase.

One of the SRS issues is correctness, facing the formal and informal view, which occurs effectively from customer requested services to control hierarchy)Alzyadat, AlHroob, Almukahel, & Atan, 2019; Larsson & Borg, 2014(. On another hand, the completeness of issue is considered an upper level of correctness, taking consistency among correctness of the services (Kamalrudin & Sidek, 2015).
The challenge is how to achieve a complete SRS, entailing correct and consistent requirements, through the UML use case diagram.

II. BACKGROUND AND RELATED WORK

Correctness is a perspective that can be defined as, the adherence to the specifications that regulate how users can interact with the approach, and how it should behave when it is used appropriately, that the approach planned tasks as defined by its specification [10]. Meanwhile, completeness is how much does a set of functions covers all the specified tasks and user objectives [11], further to which subject data associated with an entity has values for all expected attributes, and related entity instances in a specific context of use [12].

Software Requirements Specification (SRS) is complete [13] if it includes the following elements: First, all the requirements related to functionality attributes in SRS should be treated. Second, the definition of the responses of the software to all realizable classes of input data, in all realizable classes of situations, note that it is important to specify the responses to both valid and invalid input values. Third, all the diagrams, labels, figures, term definitions and measures should be referenced and labelled.

According to Alzyadat, et al. [7] who defined many characteristics that make requirements better, such as correctness, completeness, consistency (3Cs), feasibility, usability and many others, in a way that each characteristic benefit in RS and quality.

Jahanshani, et al. [14] introduced the importance of the quality through the current research on Tata motors industrial company for automobiles in India, via a questionnaire of more than 60 questions, some are derived from previous researches, and the else are designed to evaluate customer service, product and loyalty, and analyzed them with ANOVA test and SPSS16. They stated the relation between good quality and customer loyalty for the product and company, that customer service quality and product quality mostly affect the customer loyalty, yet when the customer get what he wanted as he wanted, he would be satisfied which leads to better quality, the reason for achieving customer loyalty.
In addition to according to Goofin and Price (1996), the importance of customer service comes from achieving better quality, more sales and income, and competitive level in the market.

Naeem, et al. [10] defined C&C under the quality umbrella, stated three problems of requirements with examples in web applications, and defined three strategies to solve them, furthermore stated a benchmark table as a guide for requirements engineers of the predefined problems and their solutions, which affect the quality through requirements negotiation.

The research of Zowghi and Gervasi [15] introduced C&C in two points of view, (1) formal, that correctness is a combination between completeness and consistency, (2) practical, that correctness is a satisfaction of specific business goals needed by the customer, and he presented review papers about correctness, completeness, and consistency relating them with real-life practice. Furthermore, stated an automated tool from related work [16] and resulted in proving C&C formally.

Firesmith [17] detailed five common problems of requirements, explained their negative consequences and solutions introducing the C&C issue, such as poor requirements quality, requirements not traced, inadequate requirements process and unprepared requirements engineers, as well stated that these problems are interleaved with each other, that means if a company had one of the problems, it is probably had another interrelated problem.

An experiment of Larsson and Borg [8] (Alhroob, Imam, & Al-Heisa, 2018) proposed ten challenges faced requirements engineering aligning them to Verification and Validation (V&V), explained each one and its effect on quality, one of them is: defining complete requirements; stating that requirements changes continuously as new requirements arrive, so an audit is done which is documented in an audit log, the audit contains the challenges in the requirements like requirements conflicts and missing requirements, audits include representatives from developers, business analysts and testers at least. While change happens in an audit like requirements conflict, for example, the business analysts perform a new audit for the same representatives to revise and refine the audit, documenting them in a new log, and this iteration is repeated until no change is found. Other examples of challenges proposed are: defining good verification process and verifying quality requirements.

A systematic review of Kamalrudin and Sidek [9] stated a review discussion on the (3Cs) in the requirements validation process defining each criterion, then introduced traceability and its importance of keeping in touch with requirements, forward to analysis approach and divided it into two parts (1) heuristic analysis which is subjective, (2) formal mathematical analysis, then eventually produced a heat map to reveal the most commonly used approaches and methodologies.

Kalinowski, et al. [18] presented a study on incomplete and hidden requirements based on a survey called Naming the Pain in Requirements Engineering

(NaPiRE) in Austria and Brazil from 14 and 74 companies respectively, both used plan-driven (waterfall) and agile methodology (scrum), they stated many problems in requirements which some of them are: missing qualifications of requirements engineering team members, lack of experience, missing domain knowledge, unclear business needs and poorly defined requirements, and introduced solutions to them, but the most remarkable problem was incomplete and hidden requirements that occupied the top of the list in Austria and the second in Brazil.

Kuchta [19] claimed that no one can guarantee absolute completeness, and stated metrics to measure weakness in SRS completeness, the measurements were divided into direct measures counted in quasi-graph storage, and indirect measures calculated by some formulas. Then took a sample of SRS document, prepared three incremented versions and evaluated them; the first version was prepared very thoroughly of 69 functional requirements, and the evaluation results in valuing to completeness, consistency, and correctness from low to high respectively. Based on the first version, the second version with 36 functional requirements and the evaluation results in increasing completeness, consistency but decreasing of correctness value and critical, exceptional and breakdown situations existed, which needed a functional requirement for each situation to prevent or fix, that resulted in 99 new functional requirements. The third version, formally unresolved goals such as needs, tasks, and problems existed, the reason that laid consistency increasing, which results in 30 new requirements that evaluated in increasing completeness, consistency and correctness. So consistency has an impact on completeness yet without quality or consistency evaluation, about 2/3 of the functional requirements will be ignored, that affect the SRS to be incomplete.

Swathi [20] focused on software quality since it is a key-value to the final product, Miss. G. Swathi and Dr Ch GVN Prasad stated an interpretation of many requirements, each set has a kind of a problem to solve clarifying that the quality of the product, depends on the requirements initialized to specification documents, and how the requirements volatility affects the software development life cycle by impacting the time, cost, effort and final product quality. Then introduced some examples of poor requirements and clarified them, coming to a result that requirements can be improved by paying attention to requirements activities, so "Effective Requirements Practices" is suggested to improve writing requirements in SRS.

Femmer, et al. [21] stated the requirements smells by doing an analysis a so-called light-weight which was based on the Natural Language of the "International Standard - Systems and software engineering -- Life cycle processes --Requirements engineering" [22], and applied the approach on two case studies taken from two different companies, which contained 339 requirements and 53 use cases extracted from 9 specifications of previous companies. "Ambiguities or incomplete requirements specifications can lead to time and cost overrun in the project. Requirements (bad) smells, which are concrete symptoms for a requirement artefact's quality defect".

Gigante, et al. [23] stated quality and its importance by defining (3Cs), and their effect on quality, yet proved completeness against Natural Language (NL) (external context, or unstructured textual requirements), by adopting ontologies beside the Resource Description Framework (RDF) triplets, and other heuristics and Natural Language Processing (NLP) tasks, to verify the High-Level Requirements (HLR) against System Requirements (SR) representing external context, so if the difference between them is 0 then, a redundancy exists which doesn't add any value, so the smaller the distance between them the more completeness is achieved [24].

III. METHODOLOGY

The approach describes each level, starting with use case diagram and ending with the achievement of completeness, using a programmed tool to verify the inner process of each level, to ensure clear achievement of C&C through the use case diagram based on UML.

The approach illustrated in Figure 1 shows the levels that use case diagram will go through, to reach a result that C&C in the SRS document is proved.

Then a feedback must return informing the developer to check the use case diagram or the SRS, the cycle continues to match all the elements, then the achievement of the completeness criterion will be reached.

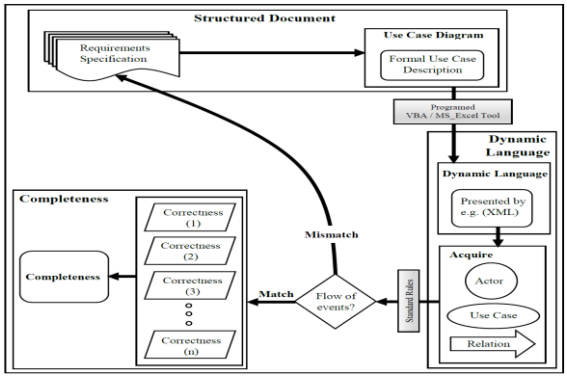


Fig. 1: Proposed Approach

The approach consists of three levels:

A. *Level 1: Structured Document*

The initial execution in this level is a use case diagram presented by UML which is a dynamic form, each use case in the diagram has its scenarios entailing the flow of events, connected with actors by specific types of relations.

B. *Level 2: Dynamic Language*

The elements (actor, use case and relation) will be extracted from the use case diagram, and whilst the process is to transform it as dynamic to the text as a static representation to acquire these elements, the language XML is suggested. Meanwhile, the use case diagram as an

initial for this level is decomposed into its elements, as mentioned earlier, which are actor, use case and relation, identified by their IDs according to XML.

C. Level 3: Completeness

At the beginning of this level, each use case is checked according to its formal description and by whom it is communicated, further to the type of the relation (UML, 2017), so according to these rules, if the acquired elements match the use case diagram then the correctness criterion will be achieved, if not, then a feedback must return informing the developer to check the use case diagram or the SRS, the cycle continues to match all the elements, then the achievement of the completeness criterion will be reached.

The map-rules implement rules in UML models are used from "IBM Knowledge Center" [26] and the UML [25] sites as standards, with instructions represented by pseudo-code and a programmed tool with Microsoft Excel, to check the correctness of the use case diagram elements:

A. Actor

"Classifier" that plays a role interacting with the system boundary and connected to the use case to gain services from, It can be hardware, software or human.

B. Use case

"Behavior classifier" that specifies a full useful functionality of an action that collaborates with one or more actors, each use case has a service to each actor connected to it.

C. Relationship

is a relation that connects two classifiers interacting together like use cases or classifiers, that describes the nature of the relationship rules, and appears in many forms; a solid line between the connected classifiers, arrow-headed with a dashed line or triangular arrowhead.

On other side the relationship contains three types of relations as follows.

D. Association

The association is a relation connecting two classifiers describing the relation reasons and rules, and it is represented as a straight line between the three classifiers are, only binary associations are allowed, the association is between actor and use case, or use case and use case. Actor may connect one or more use cases. And use case may be connected to one or more actors.

The table 1 shows the association between two actors clarifying that no rule between two actors is accepted except the Generalization.

TABLE I: THE RULES BETWEEN TWO ACTORS

Pseudocode (Actor to Actor)

A is the first classifier
B is the second classifier
IF A = "Actor" and B = "Actor" Then check if Relation between A and B = "Generalization" Then
print "Right Generalization"
Else
Print "Wrong Relation or Classifier"
End if

The relation between the actor and the use case is limited to an association named "Communicate" in the programmed tool as shown in table below.

TABLE 2: RULES BETWEEN ACTOR AND USE CASE

Pseudocode (Actor to Use case)

A is the first classifier
B is the second classifier
IF A = "Actor" and B = "Use case" Then check if Relation between A and B = "Communicate" then print "right Communicate"
Else
Print "Wrong Relation or Classifier"
End if

In the case of the relation between two use cases, there are no forbidden relations according to the standard rules, and Table 3 shows the "Communicate" association, Table 4 shows the "Include" association and Table 5 shows the "Extend" association.

TABLE 3: RULES BETWEEN TWO USE CASES IN THE ASSOCIATION RELATION

Pseudocode (Use case to Use case)
A is the first classifier B is the second classifier IF A = "Use case" and B = "Use case" then check IF Relation between A and B<> "EXTEND" Then check if Relation between A and B ="Communicate" then print "right Communicate"

E. Include

It is a directed relation where one base use case includes the functionality of the included use case, and it appears as a dashed line with an open arrowhead pointing to the included use case.

i. Does not have names, only the keyword "Include", and if a name is added it appears beside the include connector.
ii. The relation is only between use cases, no actors involved.
iii. Split large use cases into other use cases to simplify them.
iv. Extract the same behavior of more than one use

case.

TABLE 4: RULES BETWEEN TWO USE CASES IN THE INCLUDE RELATION

Pseudocode (Use case to Use case)
A is the first classifier B is the second classifier IF A=”Use case” and B=” Use case” Then check if the Relation between A and B <> ” Extend” then check if the Relation between A and B=”Include” then print “Right Include” else print "Wrong R Relation

F. Extend

It is a directed relation between two use cases that the extension use case extends the behavior of the base use case. If the base use case is meaningful by itself then, there is no need for the extension use case.

i. Part of a use case is an optional system behavior.
ii. Executing a sub-flow under specific conditions.
iii. The possibility of inserting many behavior segments in a base use case.
iv. The relation is between use cases only.

TABLE 5: RULES BETWEEN USE CASE AND USE CASE IN THE EXTEND RELATION

Pseudocode (Use case to Use case)
A is the first classifier B is the second classifier check if the Relation between A and B =”Extend” then check if A= ”Use case” and B=” Use case” then print “Right Extend

G. Generalization

It is a relation between at least two classifiers like generalization between classes, the child use case is a type of the parent (general) use case, having the same relations and operations of the general use case, and it is represented as a big triangular arrowhead pointing to the general use case.

i. Can be between actors only.
ii. Can be between use cases only.
iii. Cannot be between actor and use case.

So table 6 shows the “Generalization” between two actors, and table 7 shows the “Generalization” between two use cases.

TABLE 6: RULES BETWEEN TWO ACTORS IN THE GENERALIZATION RELATION

Pseudocode (Actor to Actor)
A is the first classifier B is the second classifier If A= “Actor” and B= ”Actor” then check if the Relation between A and B = ”Generalization”) then print “Right Generalization” else print " Wrong Relation or Classifier "

TABLE 7: RULES BETWEEN TWO USE CASES IN THE GENERALIZATION RELATION

Pseudocode (Use case to Use case)
A is the first classifier B is the second classifier If A= “Use case” and B=” Use case” then check if the Relation between A and B = ”Generalization”) then print “Right Generalization

IV. EXPERIMENT AND DISCUSSION

The experiments show the effectiveness of the correctness and completeness in requirements specification approach, in a way that it can extract the use case diagram elements, to match them with the formal description of the use cases scenarios that make it possible for anyone to use the approach, as well as the requirements, as formal RS. The approach experimented an “Online Shopping” use case diagram that was chosen from UML [25] as a case study as shown in Figure 2.

Online Shopping

Correct case:

The Online Shopping is presented in the use case diagram representing the RS, from the UML site which is Level 1 (Structured Document) in the approach.

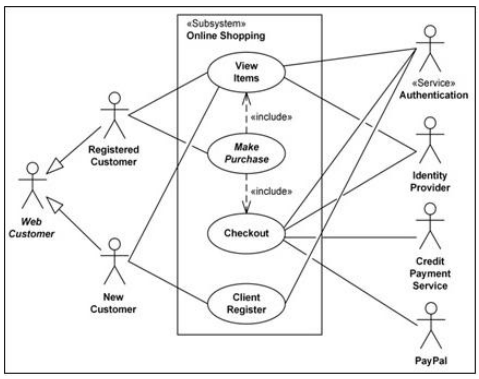


Fig. 2: Online Shopping Use Case Diagram

The applied approach starts by transforming the use case diagram into XML format, through opening the saved Rational Rose use case diagram from Visual Paradigm and exporting it to MS excel sheets format, so each element is identified by an ID, mentioning the relationship between the classifiers connecting their IDs, the XML is stored in excel sheet presenting the major items (columns) such as Element Stereotype, Element ID, Element Name, Relation

Stereotype, Element Stereotype, (From) ID and (To) ID, Each element in the diagram is transformed to details in the excel sheet, presented as IDs, names, stereotypes for use cases and actors. And for the Relations; they are presented by the IDs of the classifiers where they are connected from and to.

There are many details in the stored excel, but the items are selected and filtered as shown in table 8. The abbreviation in the titles of the next tables stands for the following terms:
X: (From) ID, Y: (To) ID, C1: From Classifier Stereotype, C2: To Classifier Stereotype, From: (From) Name, To: (To) Name, Z: Result, A: Actor, U: Use case, L: Relation, C: Communicate, G: Generalization, I: Include, E: Extend, R: Right and Cl: classifier.

TABLE 8: EXTRACTING IN-BOUNDARY ITEMS TO ACQUIRE USE CASE DIAGRAM ELEMENTS

ES	ID	EN	ID1		ID2	
A	2	Web Customer	<<Actor>>	public	No	
U	4	Make Purchase	<<UseCase>>		No	
U	6	Client Register	<<UseCase>>		No	
A	8	Registered Customer	<<Actor>>	public	No	
C	10		Unspecified	4		8
G	12			2	8	
A	14	New Customer	<<Actor>>	public	No	
G	16			2	14	
C	18		Unspecified	6		14
U	20	View Items	<<UseCase>>		No	

In Table 9 is the result of filtering the form in Table 8 to be more understandable by removing the out of boundary data and collecting the elements of use case diagram as follows, which implements Level 2 (Dynamic Language):

- ES: Element Stereotype: is the UML type of each element.
- ID: Element ID: is the identifier for each element
- EN: Element Name: is the actor or the use case name in the use case diagram.
- ID1: is the id of the source element that the relation is connected with.
- ID2: is the destination element that the relation is connected with.

TABLE 9: FILTERED FORM FOR THE IN-BOUNDARY ITEMS

X	C1	L	Y	C2	Z	From	L	To
2	A	G	14	A	R G	Web Customer	G	New Customer
4	U	I	38	U	R I	Make Purchase	I	Checkout
6	U	C	26	A	R C	Client Register	C	Authentication
8	A	C	20	U	R C	Registered Customer	C	View Items
8	A	G	2	A	R G	Registered Customer	G	Web Customer
8	A	C	4	U	R C	Registered Customer	C	Make Purchase
14	A	C	20	U	R C	New Customer	C	View Items
14	A	C	6	U	R C	New Customer	C	Client Register
14	A	G	2	A	R G	New Customer	G	Web Customer
20	U	I	4	U	R I	View Items	I	Make Purchase

Showing the final result of acquiring C&C in Table 10, through applying the approach, according to the rules which leads to achieving Level 3 (Completeness).

TABLE 10: RESULT OF APPLYING C&C

X	C1	L	Y	C2	Z	From	L	To
2	A	G	14	A	R G	Web Customer	G	New Customer
4	U	I	38	U	R I	Make Purchase	I	Checkout
6	U	C	26	A	R C	Client Register	C	Authentication
8	A	C	20	U	R C	Registered Customer	C	View Items
8	A	G	2	A	R G	Registered Customer	G	Web Customer
8	A	C	4	U	R C	Registered Customer	C	Make Purchase
14	A	C	20	U	R C	New Customer	C	View Items
14	A	C	6	U	R C	New Customer	C	Client Register
14	A	G	2	A	R G	New Customer	G	Web Customer
20	U	I	4	U	R I	View Items	I	Make Purchase
20	U	C	32	A	R C	View Items	C	Identity Provider
20	U	C	26	A	R C	View Items	C	Authentication

Some changes were made on the Online Shopping use case diagram (in bold) to falsify them, yet to ensure the success of applying the UML rules and achieving the C&C as in Figure 3.
The modifications were made by IBM Rational Rose software and some were changed manually from the excel sheet because some relations were unaccepted by Rational Rose such as association relation between actors.

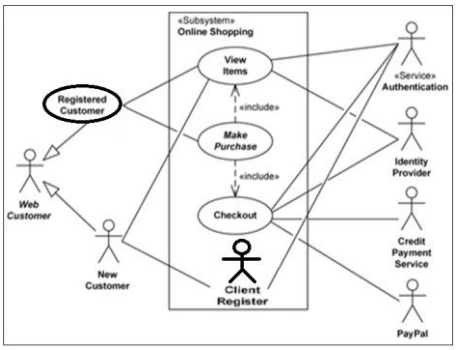


Fig. 3: Changing Some Elements Of Online Shopping Case Study (In Bold)

TABLE 11: ERROR RESULTS AFTER CHANGING ELEMENTS

X	C1	L	Y	C2	Z	From	L	To
12	A	I	10	U	Wrong L or Cl	Receptionist	I	Schedule Patient Hospital Admission
12	A	C	8	U	RC	Receptionist	C	Schedule Patient Appointment
12	A	C	6	U	RC	Receptionist	C	File Medical Reports
22	U	I	36	A	Wrong L or Cl	Patient Registration	I	Patient Hospital Admission
22	U	C	12	A	RC	Patient Registeration	C	Receptionist
22	U	E	10	U	RE	Patient Registeration	E	Schedule Patient Hospital Admission
22	U	E	8	U	RE	Patient Registeration	E	Schedule Patient Appointment
30	U	G	36	A	Wrong L or Cl	Outpatient Hospital Admission	G	Patient Hospital Admission
32	U	G	36	A	Wrong L or Cl	Inpatient Hospital Admission	G	Patient Hospital Admission
32	U	I	2	U	RI	Inpatient Hospital Admission	I	Bed Allotment

The correct use case diagram is not a basis for the false one, which means that if any incorrect use case diagram was evaluated through applying the C&C approach, it will detect the errors.

The wrong case will pass through the same approach procedure of the right one before (Figure 2), and by applying the approach according to the UML rules in section IV:

Rule 1- Actor to Actor: the relation between two actors as association (communicate as named in the approach) is wrong, it should be only (Generalization), and in the incorrect case study the wrong relations result is shown in table 11.

V. CONCLUSION AND FUTURE WORK

This work addressed two main challenges; the first challenge was C&C that was covered in RS scope, the second challenge was the consistency that appeared among the requirements correctness presented in UML, especially through the relation boundary between actors and use cases.

The approach included three levels; the first level was the Structured Document, that showed the RS represented by use case diagram entailing the scenarios formal description, the second level was the Dynamic Language, which described transforming the use case diagram as dynamic, to textual XML as static using specific software, then extracted use case diagram elements; actor, use case and relation, and the third level was the Completeness, which was based on the implemented standard rules. Comparing the rules to the formal description of the use cases scenarios of the use case diagram in the first level, if not matched, the RS or the use case diagram must then be modified, and if matched for each requirement addressing consistency among them, completeness will be achieved.

The approach was supported by a programmed tool on MS excel and XML due to IBM Rational Rose and Visual Paradigm and experimented “Online Shopping” use case diagram [25] as a case study.

The contribution of the study was to establish C&C with consistency among. Concerning the RS, to minimize the customer modifications, to achieve quality.

This research addressed the 3C’s in RS through UML use case diagram, and the tool based on the standard rules, it was proved that C&C was improved in RS scope.

- To verify Extend relationship through sub-rules.
- To adjust this approach on different methodology and technique such as together J.
- To fully enhance the automation of C&C to the rest of the software engineering life cycle like Design, Implementation, and Test, to achieve Quality.

VI. REFERENCES

[1] J. Coughlan and R. D. Macredie, "Effective communication in requirements elicitation: a comparison of methodologies," *Requirements Engineering*, vol. 7, pp. 47-60, 2002.

[2] J. Eckhardt, A. Vogelsang, H. Femmer, and P. Mager, "Challenging incompleteness of performance requirements by sentence patterns," in *2016 IEEE 24th International Requirements Engineering Conference (RE)*, 2016, pp. 46-55.

[3] D. Zowghi and C. Coulin, "Requirements elicitation: A survey of techniques, approaches, and tools," in *Engineering and managing software requirements*, ed: Springer, 2005, pp. 19-46.

[4] K. Swarnalatha, G. Srinivasan, N. Dravid, R. Kasera, and K. Sharma, "A survey on software requirements engineering for real time projects based on customer requirements," *Int'l J of Advanced Research in Computer and Communication Engineering*, vol. 3, 2014.

[5] M. Düchting, D. Zimmermann, and K. Nebe, "Incorporating user centered requirement engineering into agile software development," *Human-computer interaction. Interaction design and usability*, pp. 58-67, 2007.

[6] M. dos Santos Soares and J. L. Vrancken, "Model-Driven User Requirements Specification using SysML," *JSW*, vol. 3, pp. 57-68, 2008.

[7] W. J. Alzyadat, A. AlHroob, I. H. Almukahel, and R. Atan, "fuzzy map approach for accruing velocity of big data," *Compusoft*, vol. 8, pp. 3112-3116, 2019.

[8] J. Larsson and M. Borg, "Revisiting the challenges in aligning RE and V&V: Experiences from the public sector," in *Requirements Engineering and Testing*

(RET), 2014 IEEE 1st International Workshop on, 2014, pp. 4-11.

[9] M. Kamalrudin and S. Sidek, "A review on software requirements validation and consistency management," *International Journal of Software Engineering and Its Applications*, vol. 9, pp. 39-58, 2015.

[10] M. A. Naeem, U. Waheed, and S. F. A. Raza, "Requirement Correctness Problems and Strategies for Web Applications," *Pakistan Journal of Engineering, Technology & Science*, vol. 6, 2017.

[11] ISO/IEC, "Software Product Quality," in *25010*, ed, 2017, p. 3.

[12] ISO/IEC, "Quality of Data Product," in *25012*, ed, 2008, p. 4.

[13] A. Al-Hroob, A. T. Imam, and R. Al-Heisa, "The use of artificial neural networks for extracting actions and actors from requirements document," *Information and Software Technology*, vol. 101, pp. 1-15, 2018.

[14] A. A. Jahanshani, G. M. A. Hajizadeh, S. A. Mirdhamadi, K. Nawaser, and S. M. S. Khaksar, "Study the effects of customer service and product quality on customer satisfaction and loyalty," International Journal of Humanities and Social Science, 2014.

[15] D. Zowghi and V. Gervasi, "The Three Cs of requirements: consistency, completeness, and correctness," in *International Workshop on Requirements Engineering: Foundations for Software Quality, Essen, Germany: Essener Informatik Beitiage*, 2002, pp. 155-164.

[16] D. Zowghi, V. Gervasi, and A. McRae, "Using default reasoning to discover inconsistencies in natural language requirements," in *Software Engineering Conference, 2001. APSEC 2001. Eighth Asia-Pacific*, 2001, pp. 133-140.

[17] D. Firesmith, "Common Requirements Problems, Their Negative Consequences, and the Industry Best Practices to Help Solve Them," *Journal of Object Technology*, vol. 6, pp. 17-33, 2007.

[18] M. Kalinowski, M. Felderer, T. Conte, R. Spínola, R. Prikladnicki, D. Winkler, *et al.*, "Preventing incomplete/hidden requirements: reflections on survey data from Austria and Brazil," in *International Conference on Software Quality*, 2016, pp. 63-78.

[19] J. Kuchta, "Completeness and Consistency of the System Requirement Specification," in *FedCSIS Position Papers*, 2016, pp. 265-269.

[20] G.Swathi,Dr.Ch GVN Prasad,Arruri Jagan, Int. J. Comp. Tech. Appl., Vol 2 (3), 631-638, "Writing Software Requirements Specification Quality Requirements: An Approach to Manage Requirements Volatility," 2011.

[21] H. Femmer, D. M. Fernández, E. Juergens, M. Klose, I. Zimmer, and J. Zimmer, "Rapid requirements checks with requirements smells: two case studies," in *Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering*, 2014, pp. 10-19.

[22] ISO/IEC/IEEE Draft International Standard - Systems and Software Engineering -- Life Cycle Processes -- Requirements Engineering," in ISO/IEC/IEEE P29148_FDIS, September 2018 , vol., no., pp.1-104, 7 Sept. 2018.

[23] Gigante G., Gargiulo F., Ficco M. (2015) A Semantic Driven Approach for Requirements Verification. In: Camacho D., Braubach L., Venticinque S., Badica C. (eds) Intelligent Distributed Computing VIII. Studies in Computational Intelligence, vol 570. Springer, Cham

[24] A. A. Wael ALzyadat, "Development Planning in the Big Data Era: Design References Architecture," *International Journal of Recent Technology and Engineering (IJRTE)*, vol. 8, p. 4, 2019.

[25] UML. (2017, 5/12/2017). *The Unified Modeling Language*. Available: https://www.uml-diagrams.org/ (Last Accesed Nov. 2019)

[26] IBM. (2017, 5/12/2017). *Relationships in use-case diagrams*. Available: https://www.ibm.com/support/knowledgecenter/SS8PJ7_9.5.0/com.ibm.xtools.modeler.doc/topics/crelsme_ucd.html (Last Accessed Nov. 2019)