6-1-2021

Establishing Suitable Metrics to Encourage Broader Use of Atomic Requirements

William L. Honig
Loyola University Chicago, whonig@luc.edu

Establishing Suitable Metrics to Encourage Broader Use of Atomic Requirements

William HONIG
Department of Computer Science
Loyola University Chicago
Chicago, USA
whonig@luc.edu

***Abstract*—Despite the apparent benefits of singular, individual, or atomic requirements, their use remains limited, and teaching their creation is difficult. The acceptance of a set of requirements metrics specifically designed to evaluate atomic requirements may lead to their better utilization and improved requirements engineering. Ten metrics designed to measure atomic requirements are presented here: five used on individual requirements statements and five applied to a requirements document or set of requirement statements. Example metrics for individual requirements include Requirement Atomic Completeness and Requirement Atomic Verifiability. The set of metrics are designed to work with natural language requirements statements and have been used in education as part of a basic requirements inspection process. Further work is necessary to make the metrics more consistent and precise including exploring the use of tools that may automate some of the measures and make them less subjective.**

***Index Terms*—Requirements Metrics, Atomic Requirements, Requirements Engineering Education, Requirements Inspection**

I. Introduction

There is longstanding agreement that the success and quality of computing applications are closely correlated with high quality requirements [1], [2]. Requirements engineering, and much of software engineering in general, rely on metrics for evaluating or measuring both applications and the process of creating them [3], [4]. Metrics specifically targeting requirement quality have been an enduring theme [5], [6], and continue to evolve despite recent concern of a potential excessive focus on measurement [7].

One approach to requirements quality uses individual, singular, or atomic requirements to gain better focus and understanding of one aspect of the application at a time; this approach offers greater precision, clarity, and control of requirements [8], [9]. However, few have proposed or tested metrics for this type of requirement. Section II provides a definition of "atomic" requirements with a brief example. Section III introduces a set of metrics suitable for use with atomic requirements.

Metrics, specifically for requirements, have been used in academic courses to improve requirements engineering education [10], [11]. Section IV overviews a process using the new set of metrics with advanced undergraduate students. Results from using the metrics with student teams are mixed; Section V discusses possible ways to improve the metrics and education on creating atomic requirements.

II. Definitions and Background

A. Defining Atomic Requirements

Work on requirements and requirements engineering often begins with the IEEE requirement definition as a "statement with translates or expresses a need and its associated constraints and conditions" [12, par. 4.1.17]. This requirement engineering standard also includes, in a list of quality characteristics of individual requirements "singular" which is defined as "The requirement statement includes only one requirement with no use of conjunctions" [12, par. 5.2.5].

Although this standard does not use the term atomic, others have amplified and refined the idea of singular into the concept of atomic requirements, atomic requirement statements, or atomic use cases [9], [13], [14], [15], [16]. These generally lead to defining singular to mean a complete and indivisible requirement documented as a whole. Following that work, this paper uses as the definition:

one atomic requirement completely describes a single system function, feature, need, or capability including all information, details, limits, and characteristics.

As a brief example, consider a basic system login screen. A basic initial requirement statement might be: "Login - System shall allow login with userid and password, logout, and password reset".

As several functions or activities are included or assumed in the above, a more atomic set of statements would be those shown in Table I (only partially covering everything in the initial statement).

Intuitively, the list in Table I seems to be more singular or atomic. Requirement 4 also adds a non-functional requirement that was not explicit in the original statement.

To go beyond the intuitive feeling, metrics may be a way to assist the requirements writer. If these metrics can be used to measure the atomicity of requirements statements during requirements generation, they can become a tool for the requirements writer to use.

TABLE I. PARTIAL SET OF ATOMIC REQUIREMENTS

Identifier	***Title***	***Login Requirement Statements***
Req 1	Login Method	System allows users to login by providing **UserId** and P**assword** at the **LogInScreen**.
Req 2	Password Change	Users may change their current **Password** at any time after login is successful.
Req 3	Password Reset	Users who are unable to login successfully after three attempts are allowed to reset their **Password**.
Req 4	Password Security	User **Password** is not stored in clear text anywhere in System.
Glossary		Terms appearing in **PascalCase** are defined in separate glossary with format, encodings, etc.

The set of requirements metrics defined here have evolved in an academic setting and been used to teach atomic requirements to students.

III. METRICS FOR ATOMIC REQUIREMENTS

Given the potential represented by atomic requirements, and despite the difficulties of the imprecise definition, it is nevertheless attractive to measure and evaluate them with metrics. The metrics set described here is intended to work with any set of (supposedly) atomic requirements describing some application, system component, or subsystem. It is designed to measure the requirements quality in general (not solely if they are proper atomic requirements or not).

These metrics are divided in two categories: those applied to an individual requirement statement and those which measure the complete set of requirements. The values of each individual metric are assigned during an inspection of the full set of requirements (for an overview of the inspection process used to date, see the Section IV).

The value for each of the metrics is an integral number from 1 to 10. This range of values was picked arbitrarily to ensure adequately wide spread of metric values.

Although not covered in detail here, these metrics and the process used, assume a system or application glossary exists. Key terms used in various requirements are defined once with all details in the glossary to ensure consistent use across the separate requirements statements. The glossary includes details on length, size, capacity, units, and format when relevant to the defined terms.

A. Metrics For a Single Atomic Requirement

Table II presents a working set of metrics to measure each atomic requirement individually, giving a high-level description of each. This section describes the motivation for these metrics and further details on the current methods for measuring them.

The first two metrics, Requirement Correctness (Ra1) and Requirement Unambiguity (Ra2), are like oft-used terms for describing requirement quality [5], [7]. They have their usual definitions and are evaluated in the usual way. For purposes of this paper, which chiefly looks at the issues raised with atomic requirements they will be only briefly discussed.

TABLE II. METRICS FOR INDIVIDUAL REQUIREMENTS

	Metric	***Brief Definition***	***Range***
Ra1	Requirement Correctness	Is individual requirement properly defining a genuine system function and need?	1-10
Ra2	Requirement Unambiguity	Is the individual requirement clear and understandable to the expected users of the document?	1-10
Ra3	Requirement Atomic Completeness	Does the individual requirement include everything necessary to fully understand the desired functionality?	1-10
Ra4	Requirement Atomic Verifiability	How adequately can this individual requirement be tested with a result able to show 100% passed or failed?	1-10
Ra5	Retirement Atomic Undecompos-ability	Would further breakup of this requirement into separate parts be extremely difficult or detract from understandability?	1-10

The Requirement Correctness (Ra1) metric measures the legitimacy and genuine need for the capability or features described. For a high-quality measure this metric would use a formal requirements verification process (in a simpler, educational environment, the inspection team supplies a subjective rating). Requirements Correctness must consider whatever information the requirement provides on priority, importance, optionality, etc.

The Requirement Unambiguity (Ra2) metric evaluates the understandability of the requirement for intended audience(s) including stakeholders, developers, testers, etc. The goal is to prevent differing interpretations of the requirement by different readers; the system glossary supports this evaluation.

These two metrics are included in the individual metrics set to ensure that the overall value of the requirement is analyzed. The remaining three metrics address particularly the atomicity of the individual requirement.

1) Requirement Atomic Completeness (Ra3)

The Requirement Atomic Completeness (Ra3) metric determines if everything that belongs in the requirement is present. A simple chemistry analogy with an element shows the intent of this metric – to completely describe an element includes listing things such as its symbol, atomic number, common isotopes, atomic weight, etc. It's unlikely that an element would be completely described without each of these.

Of course, potential atomic requirements are nowhere near as indivisible as chemical elements. The goal for the requirement inspection process is to measure how well a single requirement statement meets the completeness goal; to do so the following checklist is used:

- Are all likely varieties and values of inputs covered including clear indications of which are legitimate and which are invalid?
- Does the requirement cover all possible variations and sub cases for the feature or function, and clearly specify which ones are to be handled?

- Is it clear what outputs, changes in system state, and other results must or may be expected?
- Is there nothing missing to make the requirement completely describe a single function or feature; are there any "What about this…" questions that can still be asked?
- Does the glossary include every term necessary to understand this requirement; are each of these terms defined fully in the glossary?

Currently, no individual values or weights are provided with this checklist. The reviewers determine a single metric value for the requirement statement after considering all the points in the checklist. The stronger agreement or belief that the statement is true yields a higher value for the metric.

2) *Requirement Atomic Verifiability (Ra4)*

The Requirement Atomic Verifiability (Ra4) metric evaluates the atomicity of a single requirement by measuring the likelihood that completing suitable test cases would indicate that the requirement was fully met or not met. In other words, a truly atomic requirement can be tested as a single unit and give a definitive pass or fail result. Returning to the chemistry analogy most elements can be tested to determine what they are – it is either carbon or it's not.

In other words, the metric measures the degree to which the tests are bound together and mutually interdependent. This checklist supports the inspection (using the same one to ten scale where ten means "most likely", "most difficult", etc.):

- How obvious is it what to do first in the test case (where to begin)? Are the inputs or stimulus to begin the test easy to determine?
- How well defined are the outputs or results of the feature or function? Are the values or state changes clearly determined (at least within each alternative)?
- Would it be difficult or impossible to skip one or more test cases (or steps in a test case) and still determine if the test passes (for at least some alternatives in the requirement)?
- How obvious is/are the test case(s) needed to test this requirement (if no test cases now exist)?

This checklist is, of course, easier to evaluate if the test(s) are already defined and documented and the tests are traceable back to the requirements they evaluate. Otherwise, the inspection team must use subjective evaluation to determine this metric.

3) *Requirement Atomic Undecomposability (Ra5)*

The Requirement Atomic Undecomposability (Ra5) metric determines if anything can be removed from the requirement (and likely put elsewhere in another atomic requirement). In other words, would removing some part of the requirement statement leave an inconsistent or ill defined description of the function. In basic chemistry, elements retain their atomic number, etc., even when take part in reactions with other materials.

A general term for this characteristic might be stickiness or cohesion; a checklist to prompt thinking includes these questions:

- Is everything in the requirement necessary and important for it to be understandable? In other words, if something were to be removed, would the requirement no longer make sense or become incomplete?
- Is it likely that separating the requirement into two or more parts would be difficult, less understandable, or be likely to cause redundancy?
- Would a user or customer be able to express agreement or disagreement with the requirement with a clear "yes" or "no"?
- Would it be difficult to partially implement this requirement and achieve a working application or system?

When requirements have been defined with use cases, an atomic requirement will be tied to a single use case and a complete system event or interaction between the environment and the system. Automated processing of requirements statements to test if they are decomposable may also by possible [17].

B. *Metrics for a Set of Requirements*

Table III lists metrics which measure a set of requirements – typically the complete requirements document for a system, subsystem, or application. These metrics are based on commonly used terms for requirements quality [7] and are not new or specific to atomic requirements; however, the methods for evaluating them are adapted and specialized to ensure atomicity is part of the metric. Each metric is valued with a single number.

Requirement Completeness (Rd1) evaluates the set of requirements statements for completeness. When used with a set of atomic requirements this evaluation offers a way to determine if any other singular requirement needs to be added. The granularity of atomic requirements may make this test easier – it becomes a simple question: shall another requirement be added to the set?

Requirement Consistency (Rd2) is also more straightforward when based on atomic requirements. Although it may be conceptually challenging, the metric can be calculated by looking at every possible pair of two requirements and asking if those two are consistent. The metric value is then reduced for any pair which raises concerns of compatibility or consistency.

Requirement Importance Ranking (Rd3) tests the existence of an importance ranking (e.g. essential, desirable, optional [18]) and subjectively evaluates the appropriateness of the rankings stated. Atomic requirements lend themselves to ordering and counting of requirements by their individual rankings; this comparison can support selecting a value for this metric (e.g. when the majority of requirements are all "essential" it may not genuinely represent what's needed).

TABLE III. METRICS FOR SET OF REQUIREMENTS

	Metric	*Brief Definition*	*Range*
Rd1	Requirement Completeness	Is this set of atomic requirements complete – does it provide a full definition of functionality for the system or subsystem?	1-10
Rd2	Requirement Consistency	Is this set of atomic requirements internally consistent, with no contradictions, no duplication between individual requirements?	1-10
Rd3	Requirement Importance Ranking	Are each of the atomic requirements individually assigned suitable importance categories and are the assignments appropriate?	1-10
Rd4	Requirement Traceability	Are each of the of atomic requirements uniquely identified with unchanging identifiers?	1-10
Rd5	Requirement Purity	Is this set of atomic requirements free from system design, project schedule, staffing, and other non-requirements material?	1-10

Requirement Traceability (Rd4) evaluates the ability of the requirements to support typical approaches to traceability (linking to design, tests, and system changes). Each atomic requirement should have a unique and unchanging identification (such as a sequential number). Document section or paragraph numbers do not meet this need as they change often. When clearly delineated atomic requirements are used with a suitable naming scheme, this metric requires little subjective thought. The metric can be given the maximum value if the scheme is used consistently through the entire set (and the value reduced for omissions, duplications, or other defects). An expanded definition, and more consistent evaluation of this metric would be possible using models of the traceability [19].

Requirement Purity (Rd5) measures the appropriateness of the requirement set as pure statements of system need without inappropriate details about design, implementation, schedule, etc. While atomic requirements do support clear linkage with design choices, inclusion of design within the requirement set makes it difficult to clearly delineate separate atomic requirements. For example, many requirements may imply that information will be stored or updated in a typical relational data base to support information queries; however, the structure and keys for that data table likely would be implied across many separate requirements.

While not a metric shown in Table III, the count of individual atomic requirements is a valuable piece of information as well. Well-structured and complete atomic requirements, due to their singularity and specificity, are likely to each require a significant and separable amount of design, implementation, and testing during development. While very approximate, the total requirement count provides an indication of system size; changes in the number offer a clear insight into requirement churn.

IV. PROCESS AND RESULTS

The set of metrics for measuring atomic requirements has been used in an academic setting. This section briefly describes the process used and discusses the effectiveness of the process.

Table IV is an overview of the process for inspecting or reviewing requirements as used in the classroom. A system glossary giving definitions of key terms used in the requirement statements is part of the review.

Each individual metric is given a value from one to ten. The metrics (Table II and III) are stated so that agreement with the statement indicates a higher number. A value of ten indicates full agreement with the metric statement with no concerns, ambiguity, or lack of information.

The process and metric set have been used twice with student teams doing semester-long development projects; typical teams have these characteristics:

- Teams of 5 to 6 advanced undergraduate students without assigned roles.
- Projects requiring information and logic of medium complexity implemented in 3 iterations over 15 weeks.
- Teams produce requirements from high level needs statements, select what goes into each system iteration, and write or update requirements for each iteration.
- Requirements begin with the definition of a use case model and selection of specific use cases to implement in the current iteration.
- Requirement inspection takes place upon team approval of requirements and during design and implementation.
- Teams were instructed that the metric values should attempt to be accurate; lower values did not impact course grades in any way.

A total of nine teams with more than fifty students have used the process in multiple iterations. The observations here are from the third iteration of each team's development. The earlier iterations were used to improve student familiarity with the metrics and the use of software engineering processes in general (the course is the first use of formal process for most students).

TABLE IV. REQUIREMENTS INSPECTION PROCESS

Entry Criteria	Atomic Requirements Exist	Uniquely numbered atomic requirements have been created, in natural language or other form, and are ready to review	□
	Glossary Exists	Key system or application terms fully defined in a glossary to support all requirements statements	□
	Time to Review Requirements	An inspection team has been assigned to evaluate the requirements and generate metrics	□
Process		Calculate metrics during formal review of Requirements Document	
Exit Criteria	Metrics Generated	Requirements metrics have been generated and are available in the system repository	□
	Time Spent Recorded	The total time in person minutes has been recorded	□
	Requirement Count Recorded	A count of the total number of individual requirements that were inspected is available in the system repository	□

During the inspection, the inspection team members (minimum of four) reach consensus on the value for each metric. Since each atomic requirement has five metrics (Ra1 to Ra5 in Table II), many values are generated. With twelve individual requirements (a typical number in the course projects) a total of 65 values are recorded – including the five over all measures for the set of requirements (Table III).

A. *Sample Metrics Produced*

Fig 1. shows data for metrics from all the teams that have used the process; focusing on the three metrics that are most closely tied to the atomicity. The data shown are for three requirements selected as important by the teams; this selection was done at the end of the development. Each of these requirements was then separately evaluated by two instructors without knowledge of the team's self-rating.

The first graph shows the comparison for Requirement Atomic Completeness (Ra3). Most data points fall below the diagonal line – the independent instructor ratings were generally lower than that of the student team. Apparently, student teams still struggle after two process iterations to realistically evaluate their atomic requirements for completeness.

Comparing metric values for Requirement Atomic Verifiability (Ra4), the second graph, shows the opposite tendency as student teams were less confident in their atomic requirement statement's ability to drive creation of a definitive set of test cases. Likely, the instructors experience allowed them to imagine better testing approaches than the students (specific test cases were not usually available with the requirements).

The third graph shows the results of the metric value comparison for Requirement Atomic Decomposability (Ra5) for the same three requirements from each of the student teams. For this metric, the divergence between instructor and student values is greatest. A separate study, using text classification [17], shown similar confusion between automated and human results. This variation suggests that evaluating whether a requirement statement can be further decomposed is difficult and probably lacks sufficient guidelines.

These data are less definitive than wished for to drive and improve a process for the generation of atomic requirements. The metric values generated vary widely between student teams and instructors and student teams struggle to measure their own requirements.

B. *Subjective Feedback*

Other feedback and evaluation, however, does offer some insight into why the metrics performed the way they did. The following points summarize comments from student surveys and discussions during team presentations.

First, getting students (and occasionally professional developers) to generate good requirements is difficult, given the common rush to code. The metric Requirement Purity (Rd5), while not discussed above, showed considerable tendency to combine design and other internal system decisions with requirements. By the end of three iterations, the student teams did improve performance notably in this area and recognized that as an accomplishment.

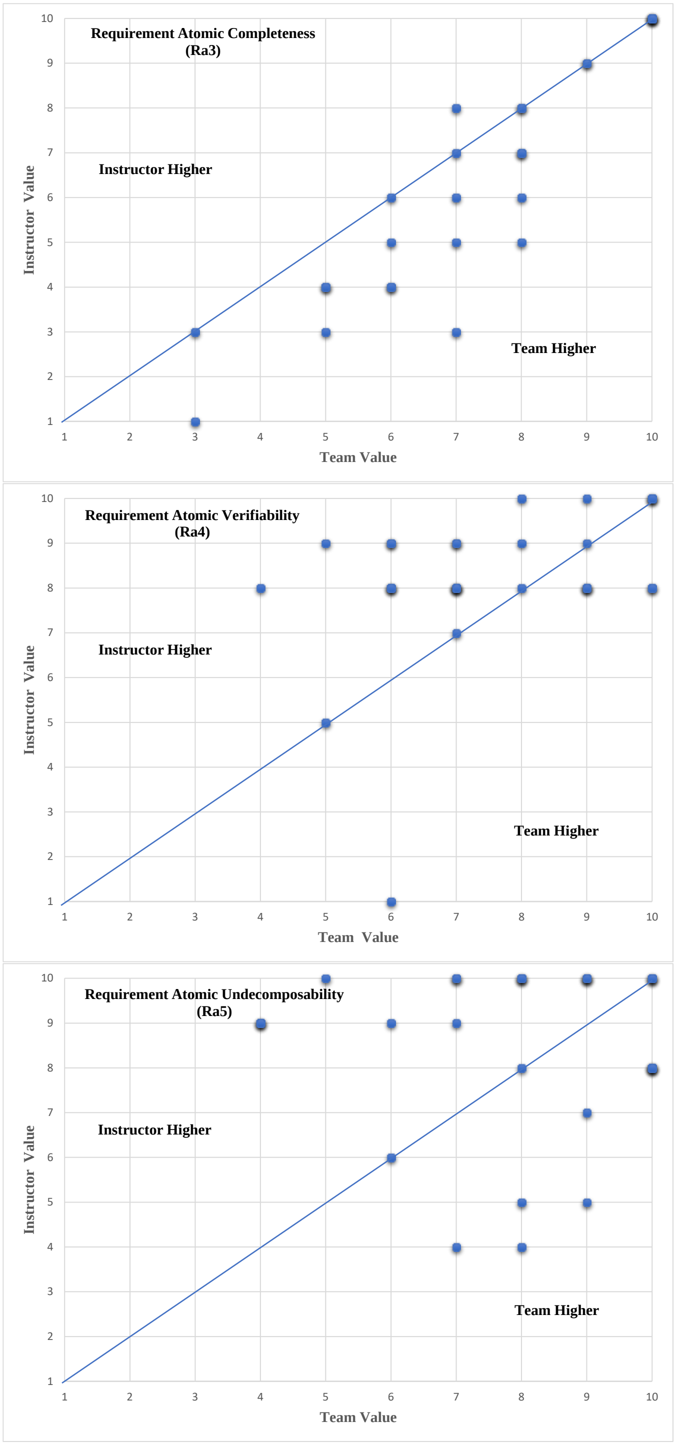

Fig. 1. Comparison of team and instructor metric values (Ra3, Ra4, Ra5)

Second, use case models and narratives appeared to be a natural lead into thinking about atomic requirements. The discipline of identifying individual use cases served as a first step to thinking about individual atomic requirements. Thinking about a use case as a single system event and response helped students begin the process of partitioning the system into separate functions.

Third, aiming to produce atomic requirements aided the student decision making about what to do in each iteration. Once individual requirements were delineated (even if not perfectly atomic), the teams had a natural way to remove work from the current development iteration or to make something optional.

V. Observations and Going Forward

To date, this attempt to use a set of metrics to measure atomic requirements (and educate computer science students on how to generate them) has not shown a clear success. Assigning values to metrics (and decision making about those values) is imprecise. At best, it seems to demonstrate that looking at the metrics proposed here helps students to think about how to write requirements.

Refinement to the metric set and further work on their definition and measurement techniques may still be promising. To continue, these steps seem to offer potential:

- Reducing the range of metric values from ten currently used to five or six may simplify calculations and comparisons.
- Some of these metrics, e.g., Requirement Consistency (Rd2), seem amenable to mechanization using machine learning or automated tools. Work on extracting dependencies from requirements [20] can provide a first step and is directly relevant to Requirement Consistency. A more prescriptive structure of natural language to facilitate an automated first review of proposed atomic requirements statements [21] may be helpful.
- A more controlled experiment, as in [22], [23], could contrast student efforts on requirements with and without metric use. This approach would be interesting after the set of metrics for atomic requirements has been further developed.

Atomic requirements statements will never be as clearly separable and indivisible as the atomic elements of chemistry. Still, the benefits of atomic requirements are worth striving for. Experiments with students offer a path to improving requirements engineering in general; hence, there's no reason not to continue searching for metrics that can help create atomic requirements.

References

[1] N. Wirth, "A plea for lean software," Computer, vol. 28, no. 2, pp. 64-68, Feb. 1995, doi: 10.1109/2.348001.

[2] S. Lane, P. O'Raghallaigh, and D. Sammon, "Requirements gathering: the journey," J. Decision Systems, vol. 25, pp. 302-312, June 2016, doi: 10.1080/12460125.2016.1187390.

[3] W. Humphrey, "Review of the state-of-the-art," Proc. 5th Intl. Software Process Workshop ISPW '90, pp. 7-11, October 1990, doi: 10.5555/317498.317687.

[4] B. Nuseibeh, and S. Easterbrook, "Requirements engineering: a roadmap," Proc. Conf. on The Future of Software Engineering, ICSE '00, pp. 35–46, May 2000, doi: 10.1145/336512.336523.

[5] A. Davis, S. Overmyer, K. Jordon, J. Caruso, F. Dandashi, A. Dinh, G. Kincaid, G. Ledeboer, P. Reynolds, P. Sitaram, A. Ta, and M. Theofanos, "Identifying and measuring quality in a software requirements specification," Proc. First International Software Metrics Symposium, May 1993, doi: 10.1109/METRIC.1993.263792.

[6] B. Boehm, "Architecture-based quality attribute synergies and conflicts," 2nd Inter. Workshop on Software Architecture and Metrics, pp. 29-34, 2015, doi: 10.1109/SAM.2015.18.

[7] S. Gregory, "Requirements engineering: the quest for meaningful metrics: time for a change?," IEEE Software, vol. 36, no. 6, pp. 7-11, Nov.-Dec. 2019, doi: 10.1109/MS.2019.2933685.

[8] H. Salzer, "Requirements atomization in software engineering education," Research proposal, Tel Aviv University, May 2003.

[9] W. Honig, N. Noda, and S. Takada, "Lack of attention to singular (or atomic) requirements despite benefits for quality, metrics and management," SIGSOFT Softw. Eng. Notes 41, 4, pp. 1–5, July 2016, doi: 10.1145/2967307.2967315.

[10] M. Bano, D. Zowghi, A. Ferrari and P. Spoletini, "Inspectors academy: pedagogical design for requirements inspection training," IEEE 28th Intl. Requirements Engineering Conf. (RE'20), pp. 215-226, 2020, doi: 10.1109/RE48521.2020.00032.

[11] A. Dekhtyar, B. da Silva and K. Slocum, "Teaching requirements engineering for all: a preliminary report," IEEE 28th Intl. Requirements Engineering Conf. (RE'20), pp. 370-375, 2020, doi: 10.1109/RE48521.2020.00050.

[12] IEEE Standard 29148, "Systems and software engineering - life cycle processes - requirements engineering," 2011, doi: 10.1109/IEEESTD.2011.6146379.

[13] H. Salzer, "ATRs (atomic requirements) used throughout development lifecycle," Proceedings of Quality Week, 1999.

[14] H. Salzer, and I. Levin, "Atomic requirements in teaching logic control implementation," Intl. J. of Engineering Education vol. 20, no. 1, pp, 46-51, 2004.

[15] K. Nguyen, and T. Dillon, "Atomic use case: a concept for precise modelling of object-oriented information systems," Object Oriented Information Systems, OOIS '03, Springer-Verlag Berlin Heidelberg, pp. 400-411, doi: 10.1007/978-3-540-45242-3_41.

[16] K. Nguyen, and T. Dillon, "Atomic use case as a concept to support the MDE approach to web application development," Proc. Intl Workshop on Model-Driven Web Engineering, 2005.

[17] F. Halim and D. Siahaan, "Detecting non-atomic requirements in software requirements specifications using classification methods," 1st International Conference on Cybernetics and Intelligent System (ICORIS), pp. 269-273, 2019, doi: 10.1109/ICORIS.2019.8874888.

[18] D. Gause, and G. Weinberg, Exploring Requirements – Quality Before Design, Dorset House, 1989.

[19] J. Holtmann, J. -P. Steghöfer, M. Rath, and D. Schmelter, "Cutting through the jungle: disambiguating model-based traceability terminology," IEEE 28th International Requirements Engineering Conference (RE'20), pp. 8-19, doi: 10.1109/RE48521.2020.00014.

[20] Y. Priyadi, A. Djunaidy and D. Siahaan, "Requirements dependency graph modeling on software requirements specification using text analysis," 1st International Conference on Cybernetics and Intelligent System (ICORIS), pp. 221-226, 2019, doi: 10.1109/ICORIS.2019.8874920.

[21] Huertas, Carlos, and Reyes Juárez-Ramírez, "NLARE, a natural language processing tool for automatic requirements evaluation," Proc. CUBE Intl. Information Technology Conference, pp. 371-378, 2012, doi: 10.1145/2381716.2381786.

[22] D. Sjoeberg, et al., "A survey of controlled experiments in software engineering," IEEE Trans. Software Engineering, vol. 31, no. 9, pp. 733-753, Sept. 2005, doi: 10.1109/TSE.2005.97.

[23] B. Kitchenham, et al., "Preliminary guidelines for empirical research in software engineering," IEEE Trans. Software Engineering, vol. 28, no. 8, pp. 721-734, Aug. 2002, doi: 10.1109/TSE.2002.1027796.