7-2016

Lack of Attention to Singular (or Atomic) Requirements Despite Benefits for Quality, Metrics and Management

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

Shingo Takada
Keio University

Natsuko Noda
Shibarua Institute of Technology, nnoda@shibarua-it.ac.jp

Recommended Citation
William L. Honig, Natsuko Noda, and Shingo Takada. 2016. Lack of Attention to Singular (or Atomic) Requirements Despite Benefits for Quality, Metrics and Management. SIGSOFT Softw. Eng. Notes 41, 4 (August 2016), 1-5. DOI: http://dx.doi.org/10.1145/2967307.2967315

Lack of Attention to Singular (or Atomic) Requirements Despite Benefits for Quality, Metrics and Management

William L. Honig
Department of Computer Science
Loyola University Chicago
Chicago, Illinois 60611 USA
+1.312.915.7988
whonig@luc.edu

Natsuko Noda
Department of Engineering and Design
Shibaura Institute of Technology
Minato, Tokyo 108-8548 JAPAN
+81.3.6722.2764
nnoda@shibaura-it.ac.jp

Shingo Takada
Dept. of Information and Computer Sci.
Keio University
Hiyoshi, Yokohama 223-8522 JAPAN
+81.45.566.1757
michigan@ics.keio.ac.jp

ABSTRACT
There are seemingly many advantages to being able to identify, document, test, and trace single or "atomic" requirements. Why then has there been little attention to the topic and no widely used definition or process on how to define atomic requirements? Definitions of requirements and standards focus on user needs, system capabilities or functions; some definitions include making individual requirements singular or without the use of conjunctions. In a few cases there has been a description of atomic system events or requirements. This work is surveyed here although there is no well accepted and used best practice for generating atomic requirements. Due to their importance in software engineering, quality and metrics for requirements have received considerable attention. In the seminal paper on software requirements quality, Davis et al. proposed specific metrics including the "unambiguous quality factor" and the "verifiable quality factor"; these and other metrics work best with a clearly enumerable list of single requirements. Atomic requirements are defined here as a natural language statement that completely describes a single system function, feature, need, or capability, including all information, details, limits, and characteristics. A typical user login screen is used as an example of an atomic requirement which can include both functional and nonfunctional requirements. Individual atomic requirements are supported by a system glossary, references to applicable industry standards, mock ups of the user interface, etc. One way to identify such atomic requirements is from use case or system event analysis. This definition of atomic requirements is still a work in progress and offered to prompt discussion. Atomic requirements allow clear naming or numbering of requirements for traceability, change management, and importance ranking. Further, atomic requirements defined in this manner are suitable for rapid implementation approaches (implementing one requirement at a time), enable good test planning (testing can clearly indicate pass or fail of the whole requirement), and offer other management advantages in project control.

Categories and Subject Descriptors
D.2.9 [**Software Engineering**]: Management – *software quality assurance, software process models*. K.6.3 [**Computing Milieu**]: Management – *software process*

General Terms
Management, Measurement, Documentation, Verification.

Keywords
Atomic requirements, atomic use cases, singular requirements, requirements creation, requirements metrics, requirements verification and validation, development process, software engineering.

1. INTRODUCTION
We wish to call attention to the apparent lack of work on what makes "a requirement" – a single and indivisible statement of system capability that can be used to support software engineering processes. The benefits of such singular, indivisible, or "atomic" requirements seem obvious, including enhanced measures of requirements churn, ease of traceability to other development deliverables, and improved metrics on requirements quality.

Despite such benefits, attention and debate on how to define a singular or atomic requirement does not seem widespread in either academia or industry. We suggest a draft definition of atomic requirement, relate the definition to past work, note the potential advantages of working with atomic requirements, and present a brief example, in hopes of motivating further attention to and discussion of the topic.

2. THE NEED
There is an abundance of software engineering work in the general area of requirements. One useful (and much used and adapted) definition of a requirement from IEEE is a "statement which translates or expresses a need and its associated constraints and conditions" [6, paragraph 4.1.17]. Although the definition refers to "a need" in the singular it is not very precise on exactly what a single need is or how to create such single (or what we term "atomic") requirements during the requirements creation processes.

In the seminal paper on software requirements quality [3] Davis et al. proposed quantitative measures for requirements specifications. Many of the proposed metrics assumed that individual requirements could be identified and calculations performed using each requirement in a complete system specification.

In [3], for example, the "unambiguous quality factor" is defined as a percentage using the "number of requirements for which all reviewers presented identical interpretations". Similarly, the "verifiable quality factor" is based on the cost and time required to verify each individual requirement. These calculations require a clear identification, count, and the ability to iterate through a well-defined set of individual requirements (otherwise different readers or reviewers may calculate costs or percentages in ways that cannot be reconciled or compared).

There are many other lists of characteristics (also termed quality attributes or "ilities") of good requirements [2, 18 Chp 4, 14 Chps 9-11]. These lists include terms such as correct, unambiguous, complete, consistent, verifiable, etc. In many cases there is again an assumption that a single separate requirement can be identified and evaluated.

The same IEEE standard cited above includes separate "characteristics of individual requirements" and "characteristics of a set of requirements" [6, paragraphs 5.2.5 and 5.2.6]. However, somewhat unusually for such lists, the IEEE standard includes "singular" as one of the characteristics of an individual requirement. That characteristic is explained as "The requirement statement includes only one requirement with no use of conjunctions" [6, paragraph 5.2.5]. Again, while requirements writers may wish to adhere to such a characteristic for requirements, it is not sufficient to create an atomic requirement simply by prohibiting conjunctions.

3. DEFINITION OF "ATOMIC" REQUIREMENT

We wish to consider a single complete requirement documented as a whole to be an "atomic" requirement. Our working definition is: one atomic requirement completely describes a single system function, feature, need, or capability, including all information, details, limits, and characteristics. An atomic requirement statement may include both functional and nonfunctional aspects of the single function.

An atomic requirement could also be called an individual requirement, a single requirement, or a cohesive requirement. The goal is to make atomic requirements that are understandable, self-contained, and complete. Only information related to a single system capability is included in an atomic requirement; it covers the simplest and smallest amount of information that makes sense to describe separately.

To aid in clarity and conciseness of atomic requirements, we define key terms in a separate system glossary. The glossary can prevent confusion from the use of natural language and help to ensure that all users of the requirement understand the term the same way. The glossary may gather and fully define terms used in multiple requirements. For example, a system's UserId may be referred to in several requirements with its format, length, character set, etc. defined once and precisely in the glossary.

Some computer systems use or assume industry standards for various functions, calculations, data representations or interfaces. These standards can be referenced from within atomic requirements and the system glossary (they should not be extracted into the atomic requirement statements).

Required standards and other general information about the system such as product goals, stakeholders, and additional background can be documented in other sections of the requirements documentation. The goal with the glossary and all such other information is to support but keep relevant information separate from the itemized atomic requirements.

Finally, some aspects of requirements may apply generally across several functions and parts of a system. One example of such a universal requirement is user interface specification. For example, a system may need to work with various screen sizes, have a certain type of graphics, etc. These details can be separately defined in one or more atomic requirement; if helpful these general requirements can note which other requirements they are bound to.

We recognize that atomic requirements are unlikely to be as clearly distinguishable or indivisible as elements in the periodic table; however, the goal is to have as a single requirement statement something which is self-contained and as complete as reasonably possible.

Since atomic requirements cannot have precisely correct boundaries, the requirements worker will need to use judgement and common sense. The goal is to attempt to create smaller and individual requirements instead of larger and broader statements. A single atomic requirement should cover the simplest and smallest amount of information about the capabilities of the system that make sense to describe separately.

4. RELATED WORK

There does not seem to be a large body of work on what constitutes a *single* requirement or how they should be created and verified during the requirements specification process. In this section we note the most relevant items and relate them to our definition of atomic requirements.

4.1 Use Case Models and Events for Identifying and Analyzing Atomic Requirements

We feel traditional use case modelling [e.g., 19 Chps, 3,4] is a good way to begin to identify atomic requirements. A use case that describes a full system event or interaction from start to finish will often represent one or more atomic requirements.

The term "atomic use case" has been used with similar motivation by Nguyen and Dillon [9, 10, 11] as a way to ensure precise and complete understanding of functional requirements. This work centers on determining the various ways the system will respond to or implement actions as the result of input from an actor.

"As a definition, an atomic use case is conceived as an instantaneous (indivisible) response by the system that is positive in the sense that either it (1) effects a change of the system's state…., or (2) performs a query that is of interest to the user…" [9]. Exceptions and error cases are handled separately. In this work, the atomic use cases are used to develop implementation templates for the system. Atomic is used in the sense of a single, self-contained, and complete system interaction; hence, it is similar to our concept.

An earlier system design and analysis scheme [7] is similar with a focus on finding all the singular "events" to which a system must respond. Here, the focus is to find all possible events (including error cases and events which were expected but fail to take place). In many ways this use of events leads to a similar complete and indivisible list of system functions as we see for atomic requirements.

Models or formalisms for the analysis of requirements also need to identify individual requirements. For example, the Abstraction Level Hierarchy states "the term atomic requirement is used to denote simple specifications in contrast to more complex ones, and requirements expressed in a single sentence with one 'shall', but without excluding multiple logical predicates within" [17]. The model and formalism defined supports a hierarchy of requirements, includes design information, and facilitates reasoning about levels of abstraction; it has been applied to teaching embedded systems development to ensure full understanding of system requirements [16].

4.2 Other Work on Atomic Requirements

A similar concept of atomic requirements has been used in at least two industrial development methodologies. Both note the desirability of individual requirements and suggest techniques for identifying them as part of the process for generating good requirements.

The IBM Rational methodology lists several characteristics for "good" requirements, including atomic. Instructions for ensuring atomic requirements include: "The requirement should contain a single traceable element… Sentences including the words 'and' or 'but' should be reviewed to see if they can be broken into atomic requirements" [5, 23 Chp 1]. Note the similar focus on eliminating conjunctions as in the IEEE definition mentioned in section 2.

The Volere requirements methodology targets the identification of atomic requirements which are defined "When you have a requirement that is measurable, testable, traceable and detailed enough to define all aspects of a need without further breakdown then you have an atomic requirement". Individual requirements are combined into higher level groups when there are too many to manage individually. Groupings are termed "Business Use Cases", "Product Use Cases", "Features", "Components", etc. [15, 20]. Individual atomic requirements are numbered and tracked.

Given that the academic work on concepts similar to atomic requirements is apparently fairly limited, we also explored other uses of similar concepts from the broader field of system development.

One consulting and project management blog has used the term "atomic requirement" in a manner similar to our concept. The Tyner-Blain site advises: "(e)very requirement should be a single requirement. If we can say 'Half of this requirement is implemented' then this needs to be two or more requirements." Similar to our findings below (section 6.2) on the advantages of atomic requirements: "(e)ach requirement you write

represents a single market need, that you either satisfy or fail to satisfy. A well written requirement is independently deliverable" [21, 22].

There are other web materials with thoughts and recommendations similar to these; however, we have not been able to identify any large, multi-source, or highly cited body of work on any similar theme. Among other industrial and consulting materials we note here some of the most relevant.

The Planet Project blog has as its goal "explain how to write atomic functional system requirements so that the spec is easy to read, and ambiguity is kept to a minimum" [12, 13]. It defines atomic as "cannot decomposed (sic) further" and provides useful natural language templates for writing various kinds of atomic requirements statements.

The Mitre online System Engineering Guide lists among criteria for a requirements statement that it should be "(s)pecific and singular: Needed system attributes (e.g., peak load) are described clearly as atomic, singular thoughts" [8]. The intent here may be for short and concise statements that are less than what we envision as a complete atomic requirement.

The BA Times online newsletter for business analysts, refers to the work of Nguyen and Dillon (see section 4.1 above) and defines an atomic use case as "it is the basic, core and single action / step carried out by an actor. It has three main and important characteristics: 1. Is very unique building block and cannot be further broken down, 2. Effects a change in the system / application, and 3. Has a binary outcome" [1]. The same article raises the question "In this generation of Agile, SCRUM and other faster than ever technologies, do you still have the burden of creating a functional specification document…?" to which it implies a positive answer.

5. EXAMPLE ATOMIC REQUIREMENT

Consider the familiar login screen where a user begins access to a system or application. Students or novice requirements workers may feel that a suitable requirements statement is similar to:

System Access. System shall control access so that user is able to log in with password, log out, and reset password anytime.

However, there are numerous defects apparent in this simple statement; it is imprecise (is only a password necessary to gain access?), ambiguous (can user log out before log in?), and incomplete (what is necessary to reset a password?). Some of these faults are caused by combining what could be separate atomic requirements in a single statement.

Higher quality information may result if a single system interaction is defined separately and in detail. One atomic requirement might define the login process or event with separate requirements for logout, forgotten password, new user, etc. The actions taken by the user and the system responses including updates to the system state may be considered a single system function.

A possible atomic requirement might be:

Requirement 1 (Log In By user). The system shall allow users to log in by providing a UserId and Password at the LogInScreen. The system shall check the UserId and Password provided to determine if the user is known to the system, in which case the user is allowed access to the MainMenu; otherwise, an error message is displayed and the system stays on the LogInScreen.

Alternative responses are included in the requirement, not just the standard success scenario. Terms shown in PascalCase are defined in a system glossary, a subset of which is shown here in Table 1. The requirement is named, given a brief title, and numbered (number shown here is illustrative only). The two screen names are briefly defined in the glossary but would be fully defined elsewhere with a mockup of the actual user interface (not shown here).

However, there is potentially more information needed that may belong with this atomic requirement. For example, further details on how the UserId and Password are processed and more explicit error processing for failed login attempts. A more complete but still atomic requirement may be:

Requirement 1 (Extended) Log in By User. The system shall allow users to log in by providing a UserId and Password at the LogInScreen.

1.1 The system shall check the UserId and Password provided to determine if the user is known to the system, in which case the user is allowed access to the MainMenuScreen; otherwise, the login is unsuccessful, an error message is displayed and the system stays on the LogInScreen.

1.2 The set of currently known UserId's and associated Password's is stored in encrypted form inside the system; clear text of UserId and Password are never stored or saved inside the system. See Requirement 14 - User Administration for more details.

1.3 If the user attempts to log in unsuccessfully using any UserId twice in any 24 hour period, the user is warned that there is only one more opportunity to successfully log in before the account will be locked. After the third failure to log in the UserId is locked and the user is informed; future attempts to log in will be unsuccessful until the UserId is unlocked. See Requirement 15 - Unlocking and Resetting User Identification.

1.4 The system shall close the LogInScreen LogInTimeOut seconds after it is displayed if there is no user response or after a failed login attempt that resulted in the UserId being locked.

This requirement is still atomic since it details a single interaction between the user and the system; the interaction continues until the user has been successful or the login attempt is complete. Some of the additional information would be characterized as nonfunctional requirements. The other referenced numbered atomic requirements provide associated information but are separate functions with their own definitions (and are not shown here).

Some requirements writers may prefer the shorter, initial requirement statement. Others may prefer, and some types of systems may require, the more complete extended requirement statement. If the shorter form were used, additional information would likely be provided in other requirements (possibly separate atomic requirements statements focusing on system administration and security).

As noted in Section 3, requirements may reference required external standards. The astute reader may observe that the definitions of UserId and Password could be improved by referencing a suitable character set standard such as Unicode (and cleaning up what it means to be upper or lower case, dropping the 26 as a number of characters, etc.).

Table 1. Partial System Glossary

Term	Definition
LogInScreen	User enters UserId and Password and requests log in. Optional choices for new user and forgot password
LogInTimeOut	Unit: seconds. A system configurable value between 10 and 120 in increments of 5
MainMenuScreen	User selects from possible choices based on capabilities of UserId
Password	6 to 10 characters at least one of which must be a number; upper and lower case characters are distinct
UserId	6 to 10 characters at least one of which must be a number; characters one of 26 upper and lower case letters; numbers one of 0 to 9; upper and lower case characters are not distinguished

6. ADVANTAGES OF ATOMIC REQUIREMENTS

We believe that an increased focus on creating individual atomic requirements will improve requirement creation and also support other steps in system development and their associated measures and metrics. This section presents thoughts on these advantages; while much of this section is conjecture, opinion, or observation, it is intended to suggest areas for further investigation and experimentation.

6.1 Improved Requirements

The overall requirements generation, verification, and validation process is improved by a focus on atomic requirements. Simply by attempting to define "a" single requirement the resulting text is more likely to be complete. The mental discipline to keep asking "Is this all one requirement?" and "Is there anything missing from this requirement?" encourages higher quality. Otherwise there is too likely a tendency to rush onward to the next thought. By focusing on one requirement at a time there is an increased likeliness of getting that requirement correct and high quality.

As mentioned in section 4.1 atomic requirements are likely to be discovered and related to a single use case or a single event between the system being developed and the external environment. This sort of focus on the granularity or atomicity of the system being defined seems to be an important heuristic aid to getting requirements right.

While we believe that good judgement is necessary to decide what content goes into a single atomic requirement, once a suitable list of these singular requirements exists, it also provides a rough measure of the size of the system so defined. It may not be possible to precisely determine if a single requirement is indivisible; however, the count of atomic requirements at any given point does provide some measure of the size of the system. While a difference between 8 or 10 singular requirements may depend more on the requirements writers style and method used to identify atomic requirements, it is likely that there will be a significant difference between a system documented with 10 requirements and one with 50 (no matter what the difference in authors or methods).

Requirements processes often call for each requirement to be uniquely identified (e.g., [6, paragraph 5.2.8.1]). Similarly, it is usually recommended to rank requirements for importance or priority [6, paragraph 5.2.4] for example into categories such as essential, desirable, optional. As atomic requirements are being defined, it is easy to identify and count each requirement, assign a name and / or number the requirement, and rank its importance. This identifier can be used to track or trace that requirement through the entire development process. (Requirements identifiers should be unique and unchanged throughout the development process).

Requirement change or churn is a common concern during development. Atomic requirements allow better measurement of churn – any change to any one requirement can be considered one change. By being as small and self-contained as possible, atomic requirements make a simple count of number of requirements changes more meaningful and useful. The unique identification of requirements aids in such measurements. If the so called "simple" change affects a third of the systems requirements, is it really so simple after all?

6.2 Improved Development and Scoping

Atomic requirements provide a good base for other phases of development including testing. The set of atomic requirements being implemented in an upcoming release gives a clear definition to the expected functionality.

With well-defined atomic requirements it is possible to implement a single requirement at a time (and add a concise increment to the capabilities of the system). Likewise, the test(s) necessary to verify the system function added should be clear from the requirement definition. Good atomic requirements go hand-in-hand with testability. If it's not clear how to define the test(s) for a single requirement (and have a clear pass/fail test conclusion) then the requirement is likely incomplete, inconsistent with previously implemented requirements, or not truly atomic.

Testing of an atomic requirement should fully exercise the capability and either 100% pass or fail. While it may be possible that only some parts of the test fail, it will usually not be useful or meaningful to use the product in this state.

Properly defined atomic requirements may facilitate automated evaluation of requirement quality and testability. Recent work has investigated whether "existing requirements quality measures such as understandability / readability can be useful in predicting requirement testability" [4].

When development projects get into trouble, there is often a need to remove some capabilities from an upcoming release ("de-scoping"). Within a planned set of atomic requirements, it will be clear where and what to cut – one or more complete requirements can be eliminated or deferred to future releases. Obviously, this decision making is aided by clear requirement identification and traceability of requirements to other development deliverables including test cases and results.

6.3 Improved Management

Atomic requirements are also broadly useful in other areas of system development and can improve management accountability and performance.

Customer agreements, including contracts, can be based on specific atomic requirements. It is possible to assign value to individual requirements in a manner similar to planned value / earned value schedule calculations. Value may be calculated from the number of atomic requirements that are fully completed, tested, and delivered.

Atomic requirements provide a base for numerous metrics for management visibility in the development process and status. Examples include numbers of requirements created, validated, developed, tested, delivered, etc. At the start of development, atomic requirements with high quality structure and precision provide a sound foundation for measuring the rest of the development.

7. SUMMARY

When originally looking for past work on requirements metrics, we asked ourselves the question "What is a requirement?" since we wanted to be able to do metrics on each separate requirements statement. To our surprise we found little work on the topic of atomic requirements. While development processes that number or identify each requirement are common, they generally do not specify what, exactly, should be given a single identifier.

While there are mentions of working with singular or atomic requirements in the sources surveyed in section 4, there is no common view of what they may be beyond limiting the use of conjunctions. Hence, we offer a working definition of atomic requirement (section 3), which, while still imprecise and requiring the use of judgement, we feel is a start in the right direction. We plan to use this definition for experiments on how different types of requirements affect requirements metrics and quality.

As noted, atomic requirements seem to have broad and important advantages – helping to provide quality requirements specifications, improving process management including change control and traceability, supporting proper testing, and allowing thoughtful software engineering in general. Thus, we write this note to spawn more discussion and future research on the topic

8. REFERENCES

[1] BA Times 2014. New Age Requirements Capturing Methodologies- Are Requirement Documents Dead?

http://www.batimes.com/articles/new-age-requirements-capturing-methodologies-are-requirements-document-dead.html. Accessed 2016- 04- 22.

[2] Boehm, B. 2015. Architecture-Based Quality Attribute Synergies and Conflicts. In *Proceedings of the 2015 IEEE/ACM 2nd International Workshop on Software Architecture and Metrics* (SAM '15). IEEE Computer Society, Washington, DC, USA, 29-34. DOI=http://dx.doi.org/10.1109/SAM.2015.18.

[3] Davis, A., Overmyer, S., Jordon, K., Caruso, J., Dandashi, F., Dinh, A., Kincaid, G., Ledeboer, G., Reynolds, P., Sitaram, P., Ta, A. and Theofanos, M. 1993. Identifying and measuring quality in a software requirements specification. In *Proceedings of the First International Software Metrics Symposium* (May 21 – 22, 1993). DOI= 10.1109/METRIC.1993.263792.

[4] Hayes, J., Li, W., Yu, T., Han, X., Hays, M. and Woodson, C. 2015. Measuring Requirement Quality to Predict Testability. In *Proceedings of the 2015 IEEE Second International Workshop on Artificial Intelligence for Requirements Engineering* (AIRE) (AIRE '15). IEEE Computer Society, Washington, DC, USA, 1-8. DOI= http://dx.doi.org/10.1109/AIRE.2015.7337622.

[5] IBM Press 2007. Requirements Management Using IBM Rational RequisitePro. http://www.ibmpressbooks.com/articles/article.asp?p=1152528. Accessed 2016- 06- 01.

[6] IEEE Standard 29148. 2011. Systems and software engineering -- Life cycle processes -- Requirements engineering. ISO/IEC/IEEE. DOI= 10.1109/IEEESTD.2011.6146379.

[7] Miranda, E. L. 1989. Looking for the event list. *SIGSOFT Softw. Eng. Notes* 14, 5 (July 1989), 80-82. DOI= http://dx.doi.org/10.1145/71633.71641.

[8] Mitre 2016. Analyzing and Defining Requirements. https://www.mitre.org/publications/systems-engineering-guide/se-lifecycle-building-blocks/requirements-engineering/analyzing-and-defining-requirements. Accessed 2016- 04- 22.

[9] Nguyen, K. and Dillon, T. 2003. Atomic use case: a concept for precise modelling of object-oriented information systems. In *Object Oriented Information Systems* (Geneva, 2003). OOIS '03. Springer-Verlag Berlin Heidelberg, 400-411. DOI= 10.1007/978-3-540-45242-3_41.

[10] Nguyen, K. and Dillon, T. 2005. Atomic use case as a concept to support the MDE approach to web application development. In *Proceedings of International Workshop on Model-Driven Web Engineering*, 2005, Sydney, Australia.

[11] Nguyen, K., Dillon, T. and Danielsen, E. 2006. The concept of web event and a practical model-driven approach to web information system development. *International Journal of Web Information Systems* 2,1, 19-36. DOI= http://dx.doi.org/10.1108/17440080680000098.

[12] Planet Project 2009. Writing atomic functional requirements. http://planetproject.wikidot.com/writing-atomic-functional-requirements. Accessed 2016- 03- 16.

[13] Planet Project 2009. How to determine if a requirement is atomic. http://planetproject.wikidot.com/how-to-determine-if-a-requirement-is-atomic. Accessed 2016- 03- 16.

[14] Pressman, R, and Maxim, B. 2014. *Software Engineering – A Practitioner's Approach*. McGraw-Hill Education, New York, NY.

[15] Robertson, S. and Robertson, J. 2013. *Mastering the Requirements Process – Getting Requirements Right*, 3rd ed. Addison-Wesley, Upper Saddle River, NJ.

[16] Salzer, H., and Levin, I. 2004. Atomic requirements in teaching logic control implementation. In *Int. J Engineering Ed* 20,1, 46-51.

[17] Salzer, H. 2010. Abstraction level hierarchy: The model and its significance for software engineering. In *Proceedings of the 2010 IEEE International Conference on Software Science, Technology & Engineering* (SWSTE '10). IEEE Computer Society, Washington, DC, USA, 61-69. DOI= http://dx.doi.org/10.1109/SwSTE.2010.11

[18] Sommerville, I. 2015. *Software Engineering*. Pearson Education Limited, Harlow, England.

[19] Stumpf, R., and Teague, L. 2005. *Object-Oriented Systems Analysis and Design With UML*. Prentice-Hall, Upper Saddle River, NJ.

[20] The Atlantic Systems Guild 2009. Atomic Requirements: where the rubber hits the road. http://www.volere.co.uk/pdf%20files/06%20Atomic%20Requirements.pdf. Accessed 2016- 03- 16.

[21] Tynerblain.com 2006. Atomic requirements. http://tynerblain.com/blog/2010/09/14/atomic-requirements/. Accessed 2016- 03- 16.

[22] Tynerblain.com 2010. Writing atomic requirements. http://tynerblain.com/blog/2006/06/14/writing-atomic-requirements/. Accessed 2016- 03- 16.

[23] Zielczynski, P. 2007. Requirements Management Using IBM Rational RequisitePro. Pearson / IBM Press.