Improving the Structure and Content of the Requirement Statement

William Scott
Joseph Kasser
Xuan-Linh Tran
Systems Engineering and Evaluation Centre
University of South Australia
Mawson Lakes
SA, Australia, 5096

Abstract. This paper discusses the perennial problem of poor requirements and summarises an attempt to mitigate the problem using an object-oriented approach by developing and using a software tool named Tiger Pro. Next the early results of using Tiger Pro in the classroom are discussed. The major outcome was the transition of the classroom discussion from a focus on the structure of the requirement sentence to a focus on the difficulty of writing a good requirement, or a focus on the content of the requirement sentence.

The paper then considers the distinction of structure and content of a requirement and the levels of examination. This can be shown to classify the attributes of a good requirement. The classification of the attributes of a good requirement reveals that the content of an individual requirement is the focus of examination but requirement management tools focus on the structure of the requirement set. The paper concludes by proposing a structure to capture the information in a requirement that facilitates the writing of better requirements.

Background

(Goldsmith, 2004) wrote that the process of "defining business requirements is the most important and poorest performed part of system development". This statement should not come as a surprise as it has been made several times over the last decade including (Kasser and Schermerhorn, 1994; Jacobs, 1999; Carson, 2001; Hooks, 1993). Yet in all this time, nothing seems to have changed. Many commercially available tools that have been developed over the last decade for requirements engineering yet none can assist their users to differentiate between a good and a bad requirement.

This situation does not mean that research into the problem was not taking place. (Kasser, 2002) described a prototype simple stand-alone tool for improving the wording of the requirement statement. The tool evolved into FRED (Kasser, 2004a), then Tiger and finally into Tiger Pro (Kasser, Tran and Matisons, 2003). Tiger (and Tiger Pro), when used in the classroom, produced the following three significant results discussed below.

- A change in perspective.
- The acceptance criteria.
- The reading level of the requirement statement.

A change in perspective. Tiger was used in a class tutorial or workshop on requirements engineering in three postgraduate courses. Before Tiger was introduced, the discussions in the tutorials focussed on the structure and format of requirements. After Tiger had been introduced and used to elucidate sample requirements, the focus of the in-class discussions changed to cover the difficulties of writing good requirements. This was a significant shift in perspective (Kasser, Tran and Matisons, 2003).

The acceptance criteria. Tiger Pro contains a database field for documenting the acceptance criteria for a requirement. This has resulted in better requirements because the dialogue as a result of the question "how will we know if or when the requirement is met?" clarifies the intent of the need statement; sometimes resulting in rewording of the requirements statement. Thus, the acceptance criterion property of the requirement helps meet the third characteristic of a good requirement, namely it is something the user really wants (Kasser, 2004b). Moreover, at the end of the first exercise in which acceptance criteria were introduced, one student demanded both an elucidation function for acceptance criteria in the way that Tiger elucidated requirements, and guidelines for writing good acceptance criteria. Until then the concept of acceptance criteria had been not even been considered.

The reading level of the requirement statement. The research is ongoing. Thus, in mid May 2005, Tiger Pro was used in a Requirements Workshop in which there were 85 participants who were professionals in the Defence Industry involved in writing or using requirements. In this workshop pre- and post- workshop questionnaires were handed out to participants. The questionnaires comprised a set of 37 good and defective requirements. For the pre-workshop assessment, participants were required to read each requirement and decide if it was good or bad. In the post-workshop assessment, the same set of requirements was provided but the order was randomly mixed. In addition, the participants were required to read each requirement and decide if it was good or bad, they were also required to state the type of defect they found in the requirements.

Comparing the number of correct decisions from the post-workshop questionnaire with the pre-workshop questionnaire showed an improvement of 71%. It was also noticed that some requirements proved to be difficult for most participants and some participants were confused when identifying different types of defect. To assess whether the level of reading difficulty in the requirements given in the workshop had any affect on the participants' responses, both the pre- and post- workshop correct answers were plotted against the Flesch-Kincaid Grade Level readability score provided in Microsoft Word as shown in Figure 1 and Figure 2 (Tran and Kasser, 2005).

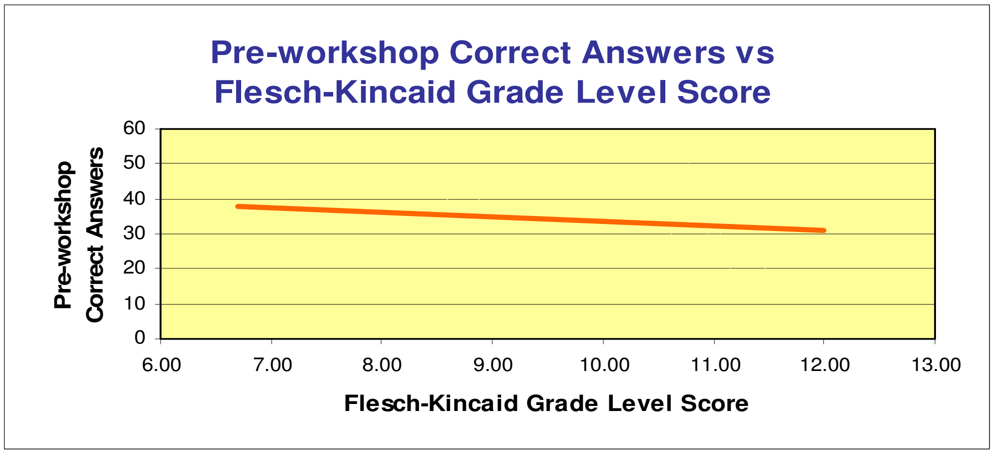

Figure 1 Pre-workshop correct answers vs Flesch-Kincaid Grade Level Score

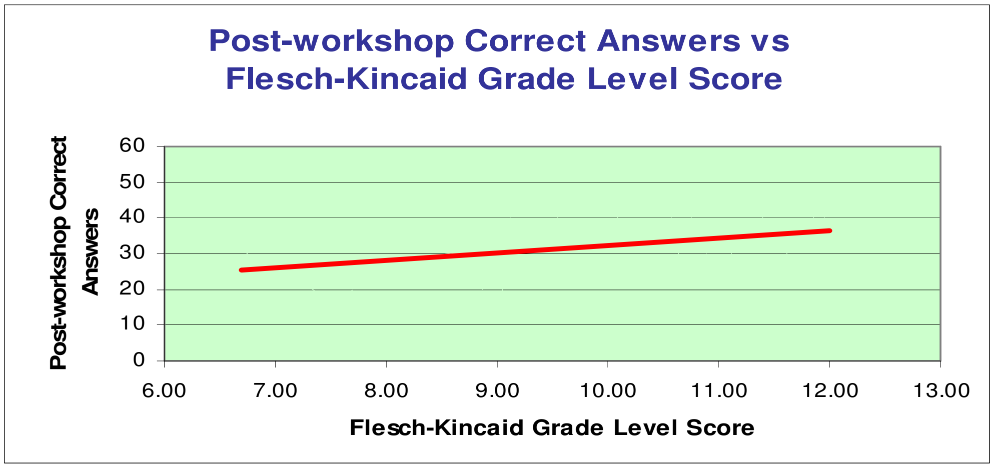

Figure 2 Post-workshop correct answers vs Flesch-Kincaid Grade Level Score

The following deductions can be made from Figure 1 and Figure 2.

- In the pre-workshop survey (Figure 1), as the reading grade level of the requirements increases, the number of correct answers decreases.
- In the post-workshop survey (Figure 2), as the reading grade level of the requirements increase, the number of correct answers also increases.

The first finding seems to be intuitive once articulated. However, the only reference to the reading level of a requirement that has been found in the literature is (Wilson, Rosenberg and Hyatt, 1997). (Wilson, Rosenberg and Hyatt, 1997) discuss the development of an Automated Requirements Measurement (ARM) tool by the Software Assurance Technology Center at the Goddard Space Flight Center. The ARM tool focuses on the grammar of the sentence and the use of "weak phrases" or "poor words" such as "large", "rapid" and "many". The ARM tool does not attempt to assess the correctness of the document, it assesses the structure of the requirements document and the vocabulary used to state the requirements based on the desirable characteristics for requirements specifications (IEEE 830-1993). While the ARM tool does provide the four readability statistics provided by Microsoft Word for the requirements specification document, as of January 29, 2003 it did not seem to make use of the statistics.

The second finding needs further investigation. The activities in the workshop seem to have improved the situation. This may be because the workshop focused on the more complicated requirements statements such as the one shown in Figure 3 taken from (STDADS, 1992).

Assessment through Focus on Content

The introduction of Tiger and Tiger Pro saw the change in focus from the structure of the requirement statement to the content. It did this by drawing the attention of the workshop participants to the presence of poor words (that complicated the subsequent verification of the requirement) in the requirement statement and sorting them in to the following five categories of defects (Kasser, 2004a).

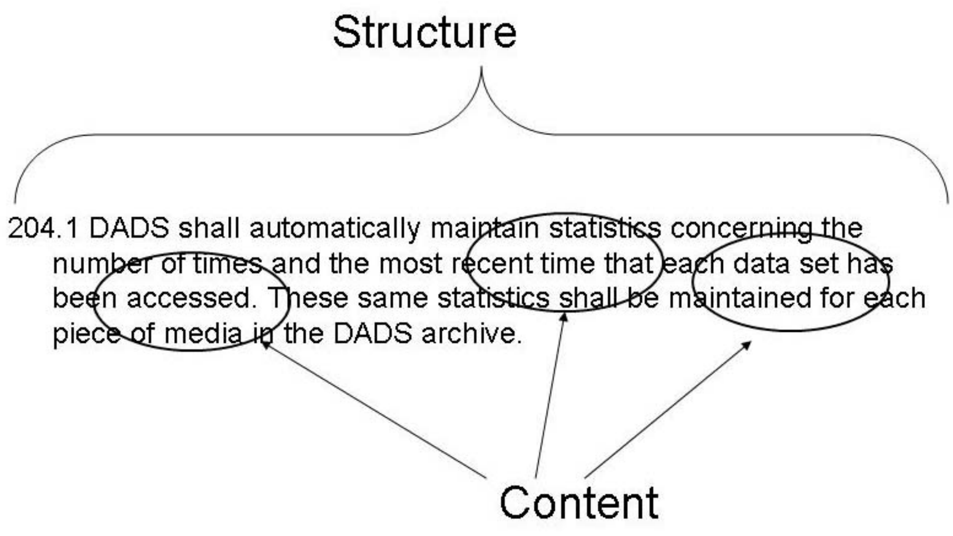
Structure
204.1 DADS shall automatically maintain statistics concerning the number of times and the most recent time that each data set has been accessed. These same statistics shall be maintained for each piece of media in the DADS archive.
Content

Figure 3 Separation of Structure and Content of the Requiremet Statement

- Multiple requirements in a statement: when the word "shall" appears more than once. This implies complication in the Requirements Traceability Matrix (RTM).
- Possible multiple requirements in a line: when words such as "and", "or" appear in a requirement statement. This situation needs clarifying as it may be or may not be a defect.
- Not verifiable word: when an un-testable or un-verifiable word appears in the statement. For example, words like "best practice", "etc.", etc. In these cases, the meaning needs to be clarified to facilitate Test and evaluation.
- Use of wrong word: when the words "should", "must", or "will" are used instead of "shall". This ensures consistency and MIL-STD-961D compliancy.
- User defined poor word: this type allows users to define words that 'shall' not appear in the requirements text for any number of user defined reasons.

The defects detected by Tiger relate to the structure (Types 1 and 2) and content (Types 3, 4 and 5) of the requirement statement.

Structure Vs Content of a Requirement

In discussions, it was observed that confusion existed arising from the level of consideration of the requirements set. Requirements can be examined at two significant levels.

The requirement statement. At the statement level, the structure is based upon a stylized version of English grammar. This stylization is based on standards for writing requirements and rules for writing technical documents. This has the effect of removing some of the less predictable styles used in writing prose. The content is the particular words used in the sentence. Standards also play an important role in the selection of words and the subsequent meaning they convey. An example of this is the use of the verb "shall" over alternatives, such as "will" or "must" (MIL-STD 490). Tiger Pro works at the content level to perform the assessments. Tiger Pro examines the content in search of particular words or phases based on (Kasser and Schermerhorn, 1994). These are used as markers that indicate the potential for a common error has been detected. Suppositions are made about the structure based on the content.

The requirements set. This level has the organisation of the database as the structure. The content is the combined meaning of the requirements. Current requirements management tools focus on providing the best possible structures to assist but lack functionality in helping the content. Requirements are often stored as text and the examination of the lower level is left to the user.

The criteria to be assessed when examining requirements can be used to illustrate this distinction. (Schneider and Buede, 2000) examined various methods and texts to create a list of criteria of a good requirement. This list can be rearranged, as seen in Table 1, into the levels and whether they apply to the structure or content of the requirement. This table is interesting to examine as the problems encountered are often at the requirement level which indicates there is the potential for significant benefits in the quality of requirements by analyzing the content of the individual requirements. Unfortunately, today's requirements management tools focus on the content-free manipulation of the requirement set.

	Structure	**Content**
Individual Requirement	Organised/Format Understandable	Complete Consistent among parts Correct Design Independent Feasible Testable Unambiguous Understandable
Set	Organised/Format Storage Traceable to need	Annotated/Necessary Complete Consistency among requirements Consistency with need Design Independent Feasible

Table 1 Structure and Content of Requirements

A Structure to Capture Requirements

Requirements management tools provide the structures for a requirement set and leave the structure of the individual statements to the users. This allows the users to be free to define the requirements as best suits their needs. However, a higher level of detail is needed for the requirements management tools to perform advanced assessment functions, such as the quality assessment of the content seen in Table 1. The advanced functions would need to have a mechanism of extracting the semantic content of the requirement sentence. This mechanism would populate a structure that reflects the semantic content of the individual requirements.

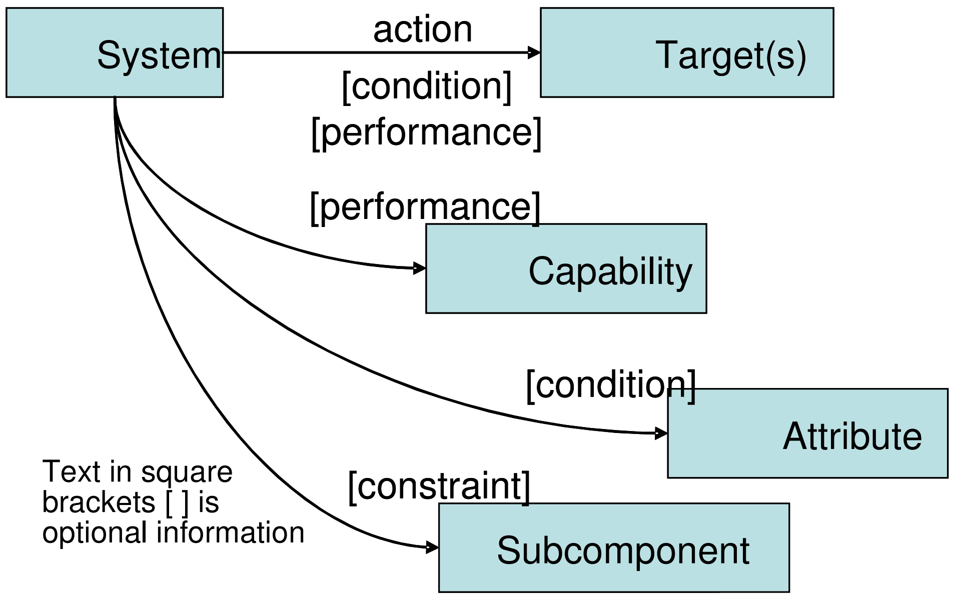

Figure 4 System Information Provided by Requirements

Figure 4 shows a proposed structure that captures the aspects of a system. Requirements are written to describe the relations represented by one of the arrows in the figure. A requirement specifies the following aspects of a system:

- Actions: the actions to be performed to what targeted objects. This information often includes conditional information on when the action is to be performed and/or performance constraints on the action.
- Attributes: the physical characteristics of the system, such as colour. Characteristics may change depending on predefined conditions.
- Subcomponents: the predefined physical architectures and the constraints placed upon the arrangements. For example, specifying the minimum number of seats in the system.
- Capabilities: the utility characteristics of the system, such as carrying capacity.
- Modes/states: the expected modes of operation. This can be viewed as a special type of capability. These are often associated with the conditions placed on actions and/or attributes.

It is possible to map common types of requirements onto this generic structure. Requirements are labelled according to the type and level of information portrayed.

Functional Requirements define the actions performed by the system with optional information on condition or performance. For example, a system may need to alert the user of a waiting message (if this is a low priority message system, the timeliness of the alert is not of importance) but still needs to be specified.

"When a new message is waiting, the system shall alert the user within TBD seconds of the message being created."

Utility Requirements encapsulate the desired capabilities of the system. These capabilities can have desired measures which make the requirement also a performance requirement. For example, a combat management system may need to have a means of identifying allied forces:

"The system shall discern allied forces from enemy forces"

Performance Requirements specify the bounds on the acceptable performance of the system. These include the minimum acceptance criteria and maximum bounds on the performance. This can be a subclass of functional requirements as they define a function to be performed and constrain the criteria of the system. Alternatively, performance requirements can be used to describe limitations or minimum acceptance criteria on the desired capabilities. For example, carrying capacity may need to be specified with the desired level:

"The system shall have a carrying capacity of more than 20 kilograms."

Design Constraints restrict the attributes and design of the system. Attributes may have conditions on the attributes to be seen. For example, software attributes can change (such as colour) to reflect the mode:

"The user interface shall have a red[1] border when the system is in Armed Mode."

A potential benefit can be seen as any attempt to capture multiple arrows in an individual requirement would violate the practice of a single point per requirement. The structure is simple yet effective.

The structure is also useful in identifying the desired information for a requirement. For example, when looking at expressing the need to be able to carry a certain weight, the requirement could be worded as:

The system shall be able to carry more than 20 kilograms.

This requirement can be better expressed when consideration of the structure is taken into account. The example attempts to define the constraint on a capability, namely carrying capacity. This information can be included into the requirements:

The system shall have a carrying capacity of more than 20 kilograms.

This is much more specific on the desires of the system and also assists the evaluation of the requirement.

The automated elicitation tool is also useful in standardising the text from multiple authors. This removes the personal style of an individual while improving the overall readability and consistency of the document. One rule, in compliance with MIL-STS-961, is to state exceptions and conditional criteria before the remainder of the statement. This rule exists as sometimes the whole of text of a long requirement is not read early on in the design phases. For example

The system shall respond to a query within 2 seconds unless an alternative response time is provided

[1] The specific shade of red needs to be specified elsewhere such as I the glossary.

Should be rewritten as

Unless an alternative response time is provided, the system shall respond to a query within 2 seconds

With information in this structure, additional automated assessments can be made about the content of the system definition. Comparison of the links in the representation (and the attached information) can identify matches, overlaps and contradicting statements within the requirements set. Higher level assessment procedures can subsequently be built to provide additional assessment.

Conclusion

The quality of a requirement can be assessed by an examination of the content. Tiger Pro implements an assessment algorithm that examines the content of each requirement and provides a measure of the quality of the requirement set from the perspective of test planning. The use of this tool has shown a shift in the discussions towards the content rather than the structure.

References

Carson, R. S., "*Keeping the Focus during Requirements Analysis"*, the 11th International Symposium of the International Council on Systems Engineering, 2001.

Goldsmith, R. F., *Discovering Real Business Requirements for Software Project Success*, Artech House Inc., Boston, MA, 2004.

Hooks, I., "*Writing Good Requirements"*, Proceedings of the 3rd NCOSE International Symposium, 1993.

Jacobs, S., "*Introducing Measurable Quality Requirements: A Case Study"*, the IEEE International Symposium on Requirements Engineering, 1999.

Kasser, J. E., "*A Prototype Tool for Improving the Wording of Requirements"*, Proceedings of the 12th International Symposium of the INCOSE, 2002.

Kasser, J. E., "*The First Requirements Elucidator Demonstration (FRED) Tool,"* Systems Engineering: The Journal of the International Council on Systems Engineering, Volume 7, no. 3, 2004a.

Kasser, J. E., "*Researching the Properties of Object-Oriented Requirements"*, The Conference on Systems Engineering Research, 2004b.

Kasser, J. E. and Schermerhorn, R., "*Determining Metrics for Systems Engineering"*, The 4th Annual International Symposium of the NCOSE, 1994.

Kasser, J. E., Tran, X.-L. and Matisons, S., "*Prototype Educational Tools for Systems and Software (PETS) Engineering"*, Proceedings of the AAEE Conference, 2003.

Schneider, R. E. and Buede, D. M., "*Properties of a High Quality Informal Requirements Document"*, the 10th International Symposium of the INCOSE, 2000.

STDADS, *ST DADS Requirements Analysis Document (FAC STR-22), Rev. C, August 1992, as modified by the following CCR's:- 139, 146, 147C, 150 and 151B*, NASA/Ford AeroSpace, Greenbelt, MD, 1992.

Tran, X.-L. and Kasser, J. E., "*Improving the recognition and correction of poorly written requirements"*, the Systems Engineering Test and Evaluation (SETE) Conference, 2005.

Wilson, W. M., Rosenberg, L. H. and Hyatt, L., "*Automated Analysis of Requirements Specifications"*, the IEEE International Conference on Software Engineering, 1997.