A Metrics Based Approach for Identifying Requirements Risks

Valerie Wyatt, Justin DiStefano, Mike Chapman, Edward Aycoth
Metrics Data Program, NASA IV&V Facility
100 University Drive, Fairmont, WV 26554, USA
Valerie.S.Wyatt@ivv.nasa.gov, Justin.S.Distefano@ivv.nasa.gov, Edward.K.Aycoth@ivv.nasa.gov, M.Chapman@ivv.nasa.gov

Abstract

The NASA Independent Verification &Validation (IV&V) Facility's Metrics Data Program (MDP) has been tasked with collecting data in the form of metrics on software products from various NASA Projects. The goals of the program include:

- *Improve the effectiveness of software assurance,*
- *Evaluate the effectiveness of current metrics,*
- *Identify and include new metrics,*
- *Improve the effectiveness of software research, and*
- *Improve the ability of projects to predict software errors early in the lifecycle.*

This article presents a model for accomplishing these goals from a requirements position. Identification of metrics from a requirements perspective approach presents a particularly difficult challenge. First, there are few automated tools to assist in collection of requirement based metrics. Secondly, few metrics have been identified for requirements. In this article, an approach is presented for capturing requirements measurements generated utilizing Goddard Space Flight Center (GSFC) Software Assurance Technology Center's (SATC) Automated Requirements Measurement (ARM) tool. These measurements are used in combination with newly identified measurements to identify and assign a risk level metric to each requirement. The assigned requirement risk level represents an indicator for early lifecycle analysis for use in prediction of problem requirements and areas within the software that could be more prone to errors. The early identification of high risk areas allows for risk mitigation and application during planning of future development, test, and maintenance activities.

1 Introduction

The NASA Independent Verification &Validation (IV&V) Facility's Metrics Data Program (MDP) has been tasked with collecting data in the form of metrics on software products from various NASA Projects. The goals of the program include:

- Improve the effectiveness of software assurance,
- Evaluate the effectiveness of current metrics,
- Identify and include new metrics,
- Improve the effectiveness of software research, and
- Improve the ability of projects to predict software errors early in the lifecycle.

These MDP goals when approached from a requirements analysis position present a particularly challenging task for the following reasons:

- There are few automated tools to support requirements analysis or metric collection.
- The lack of automated support lends itself to requiring long hours of manual, labor intensive reviews.
- Currently, there are no metric definitions for indicators of requirement quality or risk indication.

This article presents an approach utilizing a combination of measurements captured from detailed reports generated from execution of SATC's ARM tool and manual identification of new requirement measurements. The metrics are assigned weighted values in a formula to determine their resulting associated risk level.

The data set utilized in this article is a small data set obtained from the MDP for an instrument data processing unit. The requirement data set requirements traceability matrix containing 279 detailed requirements. Reuse requirements are identified and were extracted. They were not used as a part of this study. A total of 160 requirements were identified. Of the 160 requirements 114 could be traced down to the functional implementation level by the MDP Team. The associated error rate data for modules was also identified. The tracing of requirements to the lowest functional level allows the requirement model presented to determine whether requirements assigned higher risk levels correspond to areas in the software that were prone to higher error rates. The average risk level across each

Computer Software Configuration (CSC) is calculated and provides insight into high risk CSCs. These CSCs when considered together with criticality factors can assist requirements analysts in prioritizing and focusing analysis activities on high risk CSCs to requirements driving this increase in complexity.

The remainder of this paper is structured as follows: Section 2 is a general overview and background of the NASA IV&V facility, its mission, and the Metrics Data Program. Section 3 is an overview of the ARM tool and metrics it collects. Section 4 provides a description of the MDP team's use of ARM measurements. Section 5 provides a description of new requirement measurements defined by the MDP Team. Section 6 describes Risk Level definitions and calculations. Section 7 is a description of the project which was used during this study. Section 8 describes the findings and results. Section 10 is a description of ongoing and future work.

2 Background of NASA IV&V and MDP

The NASA IV&V Facility located in Fairmont, West Virginia was established in 1993 as part of an agency-wide strategy to provide the highest achievable levels of safety and cost-effectiveness for mission critical software. The IV&V Facility was founded under the NASA Office of Safety and Mission Assurance (OSMA) as a direct result of recommendations made by the National Research Council (NRC) and the Report of the Presidential Commission on the Space Shuttle Challenger Accident. Since then, the IV&V Facility has experienced continual growth in personnel, projects, capabilities, and accomplishments. The IV&V Facility's efforts continually contribute to the improvement of NASA's safety record since its inception. Today, the IV&V Facility is a part of GSFC.

The NASA IV&V MDP project is being developed by Galaxy Global Corporation, Inc. for NASA. The primary objective of the MDP is to collect, validate, organize, store and deliver software metrics data. The MDP website provides access to the data repository containing software metrics and associated error data at the function/method level. The data repository stores and organizes the data which has been collected and validated by the MDP. This data is made available through the website with approval of the project(s) which work in cooperation with the MDP. The requirements model presented in this article represents an evolving approach to collecting metrics to support early error prediction from requirements analysis.

3 ARM Tool Metrics

SATC developed an early lifecycle tool for assessing requirements that are specified in natural language. The ARM tool searches a requirement specification document for key words and phrases identified by SATC as quality indicators. It is believed these metrics can support NASA project managers in assessing the quality of requirements specification documents and assist in identifying risks associated with poorly specified requirements that could impact the project. The tool does not attempt to assess the correctness of the requirements specified. It assesses the structure of the requirements document and individual specification and the vocabulary used to state requirements.[1]

Metrics generated by the ARM tool include:

- Imperatives – Words and phrases that command that something must be provided ("shall")
- Incompletes – Indications of incomplete requirements (TBD, TBR, etc.)
- Option – Words that loosen the specification by giving the developer latitude ("should")
- Weak Phrases – Multiple interpretations or ambiguous terms
- Continuances – Phrases that follow an imperative and introduce lower level specification requirements.
- Directives – measure of references to figures, tables, etc.
- Lines of Text – measure of physical lines of text

The first five measurements are of particular interest to the MDP Team for looking at each requirement individually. Directives and Lines of Text may provide indications of document quality but not individual requirement indications.

4 MDP Use of ARM Metrics

The MDP Team executes the ARM Tool against a select requirements document producing the generated report. As a part of the complete ARM generated report, a series of detailed reports are also generated. Detailed reports are generated for Imperatives, Continuances, Options, Weak Phrases, and Incompletes. These reports identify each individual requirement having an occurrence of the key words or phrases identified by ARM. Requirements containing more than one key word or

[1] Wilson, William M., Rosenberg, Linda H., and Hyatt, Lawrence E., Automated Quality Analysis of Natural Language Requirement Specification, October 1996.

phrase are captured in multiple reports. A Microsoft Access database has been created for managing requirements and data as it is collected. The requirements from a select specification document are extracted and imported into this database. Each of the detailed reports generated by the ARM Tool are extracted and imported into tables in the database. These tables are linked to the requirements table by the requirement number. Figure 1 illustrates the table created and their relationships.

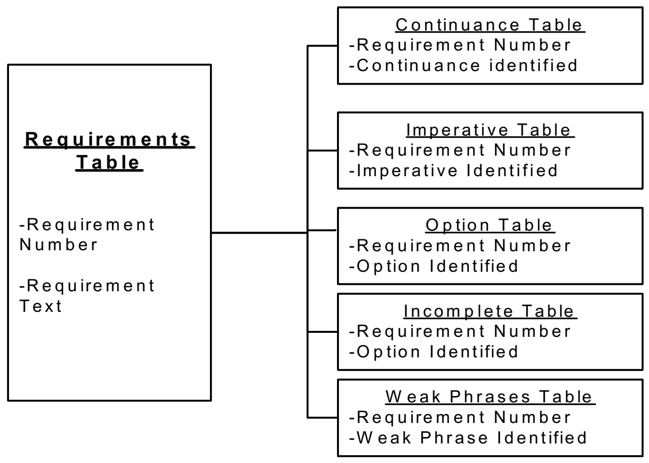

Figure 1

An access query is then executed that produces a count of the occurrences of key words and phrases in each individual requirement. This allows a requirements analyst to easily identify individual requirements containing various combinations of potentially problematic statements based on ARM results. See Figure 2 below.

Id	Imperative	Incomplete	Option	Weak Phrase	Continuance
5.14.1.9	3	0	0	1	0
5.14.1.8	1	0	1	0	0
5.14.1.7	1	0	0	0	0
5.14.1.6	1	0	0	0	1

Figure 2

5 New Metrics

In addition to measurements generated using ARM reports, three new measurements have been identified. These include:

- Actions –the number of actions the requirement must perform
- Sources –the number of data sources or interfaces the action may need to be performed against.
- Conditionals – whether the requirement needs to address multiple conditions (i.e. if, then, when, in the event of)

These additional measurements are generated in the access database by reviewing each individual requirement using an access form that provides the requirement number, text, and data captured from ARM.

Currently the capturing of these measurements requires a manual, labor intensive review of each requirement. Each requirement is displayed together with its ARM associated metrics. The identification of actions, sources, and any conditional statements are then assigned by reviewing the requirement and capturing the measurements through the form interface.

6 The Risk Level

Upon completion of capturing measurements associated with each requirement, an associated risk level is assigned to each requirement. The three risk level definitions are identified below:

- Level 1 – The lowest possible risk level. This risk level captures requirements involving single, relatively non-complex requirements (ex. The boot CSC shall set the processor speed to 20 MHz).
- Level 2 – This level captures requirements containing multiple actions and/or sources. These generally contain more than one imperative and/or continuances. (ex. The boot CSC shall test and clear RAM on power-on and reset).
- Level 3 – This risk level indicates requirements that contain conditionals, incompletes and possibly multiple sources and/or actions. (ex. The boot monitor of the boot CSC shall be activated if the Clear-To-Send signal is active on the XXX interface. If no activity is detected on the XXX interface within 30 seconds the boot shall load a configuration of RTOS according to the XX_Index in the identified System_Block).

Assigning of the appropriate risk level is an automated process. It is calculated based on a series of weighted averages for the number of occurrences of measurements captured for each requirement. The weighting assigns requirements containing conditional statements and incompletes the heaviest weight. Those with imperatives, continuances, actions and sources receive slightly less weight while options and weak phrases receive the least. Requirements identified for risk level 3 represent requirements that contain conditional statements and/or incompletes together with multiple imperatives, actions or sources. It is believed these requirements are representative of requirements that are more complex to implement and require additional testing to address multiple conditions. They will require more effort to maintain and increase the likelihood of introducing errors into the system. Risk level 2 requirements are moderately complex and do not contain conditional statements or incompletes. They may contain multiple imperatives, weak phrases, options and/or continuances. Risk level 1 requirements do not contain more than one imperative and contain no continuances. They may contain weak phrases, options, and/or imperatives. These requirements are usually defined in a clear, concise manner. See Figure 3.

Id	Action	Source	Cond-itional	Imper-ative	Incom-plete	Option	Weak Phrase	Contin-uance	Risk Level
5.18.3.2	4	3	1	3	0	0	0	1	3
5.18.3.1	1	2	0	1	0	0	0	0	2
5.18.2.1	1	1	0	1	0	0	0	0	1
5.18.1.3	1	1	0	1	0	0	0	0	1

Figure 3

7 Validation of the Model

The remainder of this paper summarizes the data and findings of the case study. The data set chosen was drawn from the MDP repository. It was selected to assist in providing validation of the requirements analysis model. The data set will be referred to as CM1. The CM1 data set was chosen because it provided a small set of detailed functional level requirements, error data was available, and traceability to the lowest functional level was possible. The CM1 data set was captured in a requirements traceability matrix consisting of 279 software requirements. Although software reuse was present in this development project, the heritage requirements were differentiated from those that were new development. Those requirements identified as reuse are not taken into consideration for this model. The total number of non-reuse requirements was 160 used in this case study.

The requirements traceability matrix identified the Unit Test providing validation for each requirement. A review of the Unit Test procedures found names of the functions being executed to test the requirement were captured in the test procedure. The requirements being validated by the test were also identified in the test procedure.

Configuration management data from the CM1 data set was used to identify the incorporation of changes to functions for implementation of problem reports. This change data provides the data needed to associate error density data to the lowest functional level. As a result the number of errors associated with each function could be identified.

8 Results

The MDP Team was able to establish traceability to the lowest functional level of implementation for 114 out of 160 requirements identified in the CM1 projects RTM. The requirements were traced to the lowest functional level using the Unit Test procedures containing names of the functions the test would be executing to test the requirements. This trace represents a 71% trace rate.
Figure 4 provides an example of the data captured.

Id	Associated Function	Risk Level	Error Count
5.13.3.3	tmaliHkGet	1	1
5.13.3.1	tmaliNextEvent	2	0
5.13.2.1	tmaliRingBufInit	2	1
5.13.1.2	tmaliTask	2	4
5.13.1.1	tmaliTransferToQueueISR	2	1
5.13.3.1	tmaliwait	2	2

Figure 4

The CM1 data set consists of 9 CSC's. The average risk level and total error count was calculated for all functions in each of the 9 CSC's.

CSC	Risk Level Average	Total Function Error Count
TMALI	2.052	13
BIT	1.888	3
SCUI	1.756	3
CCM	1.742	9
ADC	1.625	5
ICUI	1.611	1
TIM	1.391	5
DCI	1.375	2
SSI	1.333	1

Figure 6

Figure 6 summarizes the findings when focused at the CSC grouping level. This data indicates an association between the CSC receiving the highest average risk level and the total errors occurring in the CSC. The TMALI CSC received the highest average risk level at an average of 2.052. The functions making up the TMALI CSC also received the highest number of errors. Those CSC's with the lowest average risk level received the fewest errors.

Figure 5 summarizes findings between the associated requirement risk levels assigned and error data rates for modules traced to the requirements.

Risk Levels	**1**	**2**	**3**
Total Requirements	50 44%	41 36%	23 20%
Total Modules Traced	125	106	35
Modules with error	38 30%	33 31%	12 34%

Figure 5

Figure 5 provides a summary of data results captured. The total number of requirements that could be traced to modules was 114. Of these, 50 were assigned Risk Level 1. These 50 Risk Level 1 requirements traced to a total of 125 modules of which 38 had at least one error for a 30% error rate. The number of Risk Level 2 requirements mapped to modules was 41. These 41 requirements traced to 106 modules 33 of which had at least one error for a 31% error rate. We saw fewer requirements in the Risk Level 3 category (23). Seeing more requirements at the Level 1 and a continuing decrease to fewer Level 3 requirements indicates the majority of requirements are defined in a clear, concise manner. The 23 Level 3 requirements to 35 modules. Of the 35 modules traced to Level 3 requirements 12 had errors. Although the number of requirements, as well as the modules traced to those requirements for implementation decreased, the percentage of errors show only a slight increase.

In attempting to validate the model using CM1 data some problems were identified that are believed to skew the data. These problems were encountered because in many cases there is no clear one-to-one relationship between errors and requirements. There is a one-to-many relationship from requirements to modules for implementation. These modules trace to requirements that are assigned different risk levels. When changes are incorporated to modules for correcting errors the error gets traced to multiple requirements. This resulted in one error being counted multiple times for requirements with different risk levels. An example can be seen in Figure 6.

ECR_ID	Modules Affected	Id	Risk Level
15-03691-adc-56	ccmCmdDispatch	5.12.3.2	2
15-03691-adc-56	ccmCmdDispatch	5.12.3.6	2
15-03691-adc-56	ccmCmdDispatch	5.12.3.7	3
15-03691-adc-56	ccmCmdDispatch	5.12.1.16	2
15-03691-adc-56	ccmCmdDispatch	5.12.1.18	1
15-03691-adc-56	ccmCmdDispatch	5.12.1.15	2
15-03691-adc-56	ccmCmdDispatch	5.12.1.7	1

Figure 6

10 Future Work

Additional research is on-going to further validate the metrics captured using this requirements analysis model. The CM1 data set indicates the model can assist requirements analysts performing early analysis in identifying areas (CSC's) in the software that could be prone to errors. Future work to validate this model includes further refinement and definition of measurements and application of the model to a larger scale project. Root cause analysis on errors that impact more than one requirement is necessary. Elimination of this inflation involves error report analysis to determine and associate the error data to only the associated requirement and remove applying the error count to multiple requirements where possible.

Several challenges lay ahead in the future work for refinement and validation of this model, the first of which requires more data for validation. The difficulty is not necessarily in acquiring the data but being able to identify traceability to the lowest functional level. This traceability is required to be able to map associated error data to the requirements. The MDP Team continues to establish this traceability as more data is collected. In addition, the challenge in applying the model to a larger scale project is the manual labor involved in collection of measurements for actions, sources, and conditionals in requirements. Currently, a proposal has been identified to develop an automated tool for extraction of the number of actions, sources and conditional measurements for each requirement in a specification document.

The MDP Team is continually collecting data and generating metrics. Currently metrics identified in this case study are from the MDP repository.

11 References

[1] Wilson, William M., Rosenberg, Linda H., and Hyatt, Lawrence E., Automated Quality Analysis of Natural Language Requirement Specification, October 1996.