Requirements Defect Density Reduction Using Mentoring to Supplement Training

John Terzakis
Intel Corporation
e-mail: john.terzakis@intel.com

Abstract— **Requirements authors typically receive little formal university training in writing requirements. Yet, they are expected to write requirements that will become the foundation for all future product development. Defects introduced during the requirements phase of a project impact multiple downstream work products and, ultimately, product defect and quality levels. Many companies, including Intel Corporation, have recognized this skills gap and have created requirements training classes to address this issue. While effective in providing the fundamentals of good requirements writing, much of this knowledge can be misapplied or lost without proper mentoring from a requirements Subject Matter Expert (SME). Our experience over the last decade at Intel has found that adding SME peer mentoring improves both the rate and depth of proper application of the training, and improves requirements defect density more than training alone. This paper will present data from a case study demonstrating the issues with training alone and the benefits of combining training with SME mentoring in order to achieve a greater than 75% reduction in requirements defect density.**

Keywords-requirements specification; requirements defects; requirements defect density; training; mentoring.

I. INTRODUCTION

While bachelor degrees exist for a variety of Engineering disciplines, degrees and even undergraduate courses in Requirements Engineering are scarce. Primary requirements authors (those whose primary role is to elicit and write requirements) may have some training. However, secondary authors (those whose primary role is architecture, development, testing, etc.) may have little or no training. As Berenbach, et al, state "Requirements analysts typically need significant training, both classroom and on the job, before they can create high-quality specifications." [1]. To close this skills gap, many companies have created in-house requirements courses or contracted third-party trainers to teach the basics of well-written requirements. Many are based on the IEEE 830 standard, [2] or the good, practical books published in the field over the last decade [3], [4]. At Intel, in-house requirements courses have been taught to over 13,000 students since 1999. While useful for providing an initial understanding of the issues and challenges of requirements authoring, the knowledge gained through these courses can be misapplied or lost due to the inexperience of authors in writing effective requirements. By pairing with a SME, the authors can be provided with early feedback on the deficiencies of their requirements.

This paper examines the requirements defect density rates for two secondary authors on software projects who attended a requirements writing course and then were mentored on subsequent revisions of their requirements specifications.

II. INITIAL CLASSROOM TRAINING

Both requirements authors attended a training session on requirements writing prior to beginning work on their Software Requirements Specification (SRS). These training sessions focused on the issues with natural language, attributes of well-written requirements, a consistent syntax for requirements and an introduction to Planguage (Planning Language). Issues with natural language in the training included ambiguity, weak words, unbounded lists and grammatical errors. Ten attributes of well-written requirements were shown and explained in detail. A requirements syntax of the form:

[Trigger][Precondition] Actor Action [Object]

was presented in the internal training. Finally, an overview of Tom Gilb's Planguage [5] was taught, along with exercises to reinforce the concepts.

Following the class on requirements writing, both authors began writing their requirements and submitted early samples for review. These early samples showed requirements defect densities of about 10 and 5 major defects per page respectively. These figures represent the baseline for this paper. While some of the key concepts were applied (a consistent syntax, use of Planguage), other key concepts were not (authors' continued use of weak words, failure to check requirements for the ten attributes, logic issues, etc.). With this baseline in place, we began mentoring each of the authors. Note that the examples that follow have been slightly modified from their original form to maintain author confidentiality

III. MENTORING

Our mentoring consisted of reviewing the requirements, identifying requirements quality issues and then working with the authors to rewrite the requirements. Here is an initial sample requirement from the first author:

The software should have radio style buttons to enable/disable graphics cards.

Issues with this requirement include use of a weak word (should), design statement (radio style buttons), the use of a slash and vagueness ("graphics cards"). Our mentoring

 ISBN: 978-1-61208-202-8

sessions focused on discussing how to correct the issues and rewrite the requirements. The updated requirement became:

The software shall display an option to enable or disable graphics cards installed in the PCIe bus.

By the latter revisions of the SRS, this author was self-reviewing requirements using a checklist provided in the requirements training class. Our reviews of subsequent requirements revealed that they required only minor rewrites and contained far fewer defects.

Initial samples from the second requirements author demonstrated similar issues. Here is a sample:

The software needs to provide the ability to wake on a wireless LAN event.

This requirement was missing a trigger (what causes the software to wake?), lacked an imperative (needs) and is ambiguous (what event?). After mentoring, the rewritten requirement became:

When the operating system (OS) is in a sleep state and the software detects a Magic Packet on the wireless network, the software shall wake the OS.

Defined: Magic Packet: A broadcast frame containing anywhere within its payload 6 bytes of 1's (0xFFFF FFFF FFFF) followed by 16 repetitions of the system MAC address.

This particular author embraced the training to the extent that he would help others to correct their requirements during review meetings.

IV. RESULTS

The requirements defect densities for each author were tracked from an initial version (0.3) of the SRS to a released version (1.0). This process took approximately one year in each case. The results appear in Tables I and II that follow.

Table I: Requirements Defect Density, Author #1

Rev	# of Defects	# of Pages	Defects/ Page (DPP)	% Change in DPP
0.3	312	31	10.06	
0.5	209	44	4.75	-53%
0.6	247	60	4.12	-13%
0.7	114	33	3.45	-16%
0.8	45	38	1.18	-66%
1.0	10	45	0.22	-81%
Overall % change in DPP revision 0.3 to 1.0: ***-98%***				

Table II: Requirement Defect Density, Author #2

Rev	# of Defects	# of Pages	Defects/ Page (DPP)	% Change in DPP
0.3	275	60	4.58	
0.4	350	78	4.49	-2%
0.5	675	125	5.40	+20%
0.7	421	116	3.63	-33%
0.75	357	119	3.00	-17%
1.0	115	122	0.94	-69%
Overall % change in DPP revision 0.3 to 1.0: ***-79%***				

V. CONCLUSIONS

The requirements defect density data indicates that large reductions can be achieved by combining training with mentoring and that mentoring benefits continue for many months after training. Initial defect density rates following training were high in each case (about 10 and 5 defects per page respectively). By combining requirements SME mentoring with this initial training, defect rates dropped by over 75% in each case. Similar reductions have been observed with other requirements authors when mentoring is combined with training. .

REFERENCES

[1] Berenbach, B., Kazmeier, J., Paulish, D. and Rudorfer, A., *Software & System Requirements Engineering in Practice*, McGraw Hill, March 26, 2009

[2] IEEE Std 830-1998, "IEEE Recommended Practice for Software Requirements Specifications", the Institute of Electrical and Electronics Engineers, Inc., June 25, 1998

[3] Wiegers, K., *Software Requirements, 2nd Edition*, Microsoft Press, March 26, 2003.

[4] Kotonya, G. and Sommerville, I., *Requirements Engineering: Processes and Techniques*, John Wiley & Sons Ltd., August 25, 1998.

[5] Gilb, T., *Competitive Engineering: A Handbook For Systems Engineering, Requirements Engineering, and Software Engineering Using Planguage*, Butterworth-Heinemann, June 25, 2005.

 ISBN: 978-1-61208-202-8