The Impact of Requirements on Software Quality Across Three Product Generations

John Terzakis
Intel
Hudson, MA USA
john.terzakis@intel.com

Abstract— **In a previous case study, we presented data demonstrating the impact that a well-written and well-reviewed set of requirements had on software defects and other quality indicators between two generations of an Intel product. Quality indicators for the second software product all improved dramatically even with the increased complexity of the newer product. This paper will recap that study and then present data from a subsequent Intel case study revealing that quality enhancements continued on the third generation of the product. Key product differentiators included changes to operate with a new Intel processor, the introduction of new hardware platforms and the addition of approximately fifty new features. Software development methodologies were nearly identical, with only the change to a continuous build process for source code check-in added. Despite the enhanced functionality and complexity in the third generation software, requirements defects, software defects, software sightings, feature commit vs. delivery (feature variance), defect closure efficiency rates, and number of days from project commit to customer release all improved from the second to the third generation of the software.**

Keywords-requirements specification; requirements defects; reviews; software defects; software quality; multi-generational software products.

I. INTRODUCTION

This paper is a continuation of an earlier short paper [1] that presented quality indicator data from a case study of two generations of an Intel software product. The prior case study compared the quality metrics for a first generation software product ("Gen 1") developed without a requirements specification (e.g., Product Requirements Document or Software Requirements Specification) versus a second generation product ("Gen 2") developed with one as its foundation. This paper includes the background and validation results from a third generation product ("Gen 3") that was designed and coded utilizing the set of requirements for the Gen 2 product as its basis. All three products were developed using traditional (i.e., waterfall) software development methodologies.

This paper is organized into eight sections. Section I provides an introduction. Section II gives the backgrounds on the three product generations. Section III presents the requirements defect rates in the three product requirements documents by revision. Section IV analyzes the predicted defect potential for the three products. Section V presents the test results for the first versus the second generation products. Section VI presents the test results for the second versus the third generation products. Section VII describes conclusions based on the data. Section VIII discusses possible future work.

II. PRODUCT BACKGROUNDS

The requirements for Gen 1 that existed were scattered across a variety of documents, spreadsheets, emails and web sites and lacked a consistent syntax. They were under lax revision and change control, which made determining the most current set of requirements challenging. There was no overall requirements specification; hence reviews were sporadic and unstructured. Many of the legacy features were not documented. As a result, testing had many gaps due to missing and incorrect information.

The Gen 1 product was targeted to run on both desktop and laptop platforms running on an Intel processor (CPU). Code was developed across multiple sites in the United States and other countries. Integration of the code bases and testing occurred in the U.S. The Software Development Lifecycle (SDLC) was approximately two years.

After analyzing the software defect data from the Gen 1 release, the Gen 2 team identified requirements as a key improvement area. A requirements Subject Matter Expert (SME) was assigned to assist the team in the elicitation, analysis, writing, review and management of the requirements for the second generation product. The SME developed a plan to address three critical requirements areas: a central repository, training, and reviews. A commercial Requirements Management Tool (RMT) was used to store all product requirements in a database. The data model for the requirements was based on the Planguage keywords created by Tom Gilb [2]. The RMT was configured to generate a formatted Product Requirements Document (PRD) under revision control. Architecture specifications, design documents and test cases were developed from this PRD. The SME provided training on best practices for writing requirements, including a standardized syntax, attributes of well written requirements and Planguage to the primary authors (who were all located in United States). Once the training was complete, the primary author submitted early samples of his requirements to the SME for review and constructive feedback. The requirements were then rigorously reviewed by both technical content experts and the SME at each major revision of the PRD.

The Gen 2 software product shared many of the same characteristics of the first product: it ran on similar platforms, was developed across multiple sites and had a two

 ISBN: 978-1-61208-283-7

year SDLC. However, it was far more complex than the first in that the software had to run with and implement functionality for a next generation Intel processor with a new microarchitecture. In addition, the multiple code bases were combined into a single release.

The Gen 3 software utilized the final set of requirements for Gen 2 as a basis for the initial PRD. The requirements SME remained with the team and worked with a new primary requirements author and later with four additional authors, who were located outside of the United States. No other requirements methods or practices changes were made. With the exception of adding support for a new CPU and the functionality it enabled (approximately 50 new software features), the basic attributes of the Gen 3 software were similar to those of Gen 2. The software development process did change slightly; the team switched to a continuous build for source code check-ins.

III. REQUIREMENTS DEFECT RATES

As mentioned previously, requirements for the first generation product were spread across documents, emails and web sites. Since the reviews were infrequent and informal, no data was captured to quantify the requirements defect levels. Furthermore, there was no one on the team able to objectively assess and measure defect levels.

For the second generation product, the SME reviewed each revision of the PRD, logged the defects and calculated the defect rate (measured in defects/page or DPP) for each revision. Requirements were evaluated using various checklists including the ten attributes of a well written requirement (complete, correct, concise, feasible, necessary, prioritized, unambiguous, verifiable, consistent and traceable), an ambiguity checklist (including vagueness, subjectivity, passive voice, and weak words) and a checklist to determine if a non-functional requirement is verifiable. Non-conformance to any of the checklist items constituted a requirement defect.

Initial requirements defect levels were high as this was the first formal set of requirements written by the author. However, with mentoring, peer reviews and stakeholder feedback, the requirements defect density for the PRD was reduced from about 10 DPP in an initial revision (0.3) to less than 1 DPP in the final revision (1.0), a reduction of approximately 98%. The results, published initially in another short paper [3], appear in Table I below.

TABLE I: GEN 2 REQUIREMENTS DEFECT DENSITY

PRD Revision	# 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%***				

The defect rate for each revision of the third generation software PRD was also assessed and recorded by the SME. In addition to a new primary author, four other authors contributed after the initial revision. Their impact can be observed at the release of revision 0.5, which shows an increase of 20% in the number of defects per page. Due to budgetary restrictions, these new authors had not been previously trained in writing requirements like the U.S. based author. They were mentored via phone by the SME for subsequent revisions. From initial to final revision, there was an approximate 80% decrease in the document defect levels. The details are presented in Table II that follows.

TABLE II: GEN 3 REQUIREMENTS DEFECT DENSITY

PRD Revision	# 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%***				

Defect prevention practices involved early inspections of requirements and a cross-functional review process. For both Gen 2 and Gen 3, the primary authors submitted initial samples of the requirements to the SME for inspection. The SME provided detailed feedback on the requirements defects and then mentored the authors on how to rewrite them so that they have a clear, common and coherent understanding amongst all stakeholders. Next, requirements were reviewed by peers for technical correctness, completeness and technical feasibility. Once the peer review was complete and the changes incorporated, the PRD was circulated for cross-functional feedback. To ensure better response rates, a "differences" document was circulated (generated from the RMT) that listed the changes between revisions. Also, several review meetings were held to discuss key sections of the PRD and obtain direct feedback on any issues.

This emphasis on minimizing requirements defects was driven by industry data linking requirements defects to software defects. Depending on the study [4], [5], [6], requirements defects are responsible for between 50% and 75% of the total number of software defects. The Chaos Reports by the Standish Group, including the one from 2009 [7], have identified requirements as one of the leading causes of project failures. Also, there is industry data from multiple sources [8], [9] indicating that the cost to fix a defect is at least 100 times more expensive in production than in the requirements definition phase. Consequently, there was considerable effort expended on the Gen 2 and Gen 3 products to focus on defect prevention rather than the traditional defect detection done by testing teams.

From a requirements perspective, the primary author for the Gen 2 product would have continued to create

 ISBN: 978-1-61208-283-7

requirements at a rate of about 10 DPP (or higher based on the defects the initial samples) without mentoring from the SME. Hence, the final revision of the PRD would have had approximately 450 defects as opposed to the 10 defects actually identified. Many of these defects would eventually have been coded into the product. Similarly, the Gen 3 author would have potentially introduced about 560 defects into the software without mentoring (vs. the 115 defects in the final revision of the PRD). The key question became whether this focused effort on requirements defect prevention would have any impact on software defects and other quality indicators.

IV. PREDICTING DEFECT POTENTIAL

To predict the defect potential [10] across the three generations of the software, we must analyze the various factors that impact the number and severity of software defects including: maturity of the team (development and validation), number of new features, the complexity of the new features, test coverage and stability of the code base at the start of the project. Comparing the three software development efforts, the teams were of about equal size and maturity and their development methodology was identical, specifically traditional waterfall. The validation teams were also of similar size and maturity. There was some overlap of personnel between projects. Overall, team maturity was consistent and thus should not influence software defect levels.

From a feature perspective, each new generation added features upon the base set of requirements from the previous generation. These new features were more complex, as they had to enable functionality in the newer Intel CPU that the software ran on. The number of requirements continued to grow since no requirements were removed for subsequent versions of the product. Test coverage of the software increased due to the introduction of formalized requirements starting with Gen 2. These factors would normally contribute to a rise in software defect rates.

The final factor to analyze is the stability of the code bases. The first version of the software consisted of multiple code bases with differing source code control systems (SCCS) and build processes. Code stability across each of the components was good. These code bases were merged into a single, unified software release at the start of the second project. After an initial period of instability due to integration issues associated with the move to a common SCCS and build process, the software should have reached the same stability as the original components since the code itself did not change. The impact of the continuous build process introduced in the third generation project also needs to be assessed. The smaller, incremental builds are likely to improve factors such as defect closure efficiency, feature variance and time from commit to product delivery since the team is able to get new builds in a much more timely manner (hours vs. days). As a result, issues can be resolved faster and this could enhance code stability. However, the more frequent build cycles can also lead to a higher number of sightings and defects because the test team can perform more testing. Thus, from a code stability perspective, software defect and sighting rates should be relatively unaffected by the code merge in Gen 2 and the continuous build process in Gen 3.

When all factors are taken into consideration, the defect potential should be *higher* for the second generation than the first and for the third generation versus the second. The key driver is number of features. Each of the subsequent products has many more features than its predecessor, those features are much more complex and the software runs on a more advanced version Intel CPU. The other factors are neutral relative to the defect potential.

V. TEST RESULTS: GEN 1 VS. GEN 2

The following data presents a comparison of the number of software defects at release, requirements volatility at major milestones, feature variance at major milestones, and defect closure efficiency at release between the first generation and second generation software products. One additional set of data is now available: time from committed product release date to actual product release date. Note that all of this data was collected for the products on two similarly configured mobile platforms only. The primary difference between them was the generation of the processor.

The most impressive set of test results is presented in Table III: the total number of software defects by type per product at the end of validation testing. SW defects rates impact not only development times and efficiency, but also customer satisfaction levels and brand reputation. The results are dramatic: each severity of software defect demonstrated a very precipitous decrease between generations. Of particular note are the 86% reduction in critical defects and over 50% drop in total defects. These results are extraordinary given the increased complexity of newer software.

TABLE III: NUMBER OF SW DEFECTS BY TYPE

Defect Type	Gen 1	Gen 2	Delta
Critical	21	3	-86%
High	137	69	-50%
Medium	111	62	-44%
Low	24	6	-75%
Totals:	293	140	-52%

Table IV compares the requirements volatility (1) per product at key milestones during development. Requirements volatility is a measure of how much the requirements are changing due to additions, modifications or deletions. A stable product will have a lower volatility index. At release, the second generation product had almost half the volatility of the first. An analysis of the Gen 1 requirements volatility by the development team revealed that most of it was due to changes needed to resolve

 ISBN: 978-1-61208-283-7

customer reported defects. Customer-driven change is evident from the tripling in the requirements volatility index from Alpha to Beta and the more than fourfold increase from Alpha to Release. For the Gen 2 product, customer defect levels were much lower and hence the volatility index increase from Alpha to Release was slightly over double.

TABLE IV: REQUIREMENTS VOLATILITY

Milestone	Gen 1	Gen 2	Delta
Alpha	0.4	0.4	0%
Beta	1.2	0.7	-42%
Release	1.7	0.9	-47%

$$\text{Volatility} = \frac{\#\text{ of added+changed+deleted requirements}}{\text{Total \# of requirements}} \quad (1)$$

Feature variance (2) per product at key milestones during development is displayed in Table V. This metric shows how well the features delivered in final product matched what was committed by the team to be delivered. At each milestone, the second generation product team was able to deliver between 1.67 times and 3 times as many supplementary features as the first generation product team. This is likely due to the efficiency gain of the Gen 2 team by not having to debug as many defects as the Gen 1 team (Table III).

TABLE V: FEATURE VARIANCE

Milestone	Gen 1	Gen 2	Delta
Alpha	0.05	0.15	+3.00x
Beta	0.15	0.25	+1.67x
Release	0.15	0.35	+2.33x

$$\text{Feature Variance} = \frac{\text{(Current - Planned Features)}}{\text{Planned Features}} \quad (2)$$

Table VI shows the software Defect Closure Efficiency (DCE) (3) at the end of validation testing for the first software release versus the second. DCE is measured by dividing the number of SW defects closed by the number of SW defects submitted. The goal is to have this percentage approach 100% (all defects closed) by release. A lower DCE is an indication that the development and validation teams are spending more time identifying, researching and correcting software defects (likely due to high defect levels). In this table, DCE at product release increased from about 69% in Gen 1 to about 87% in Gen 2, an improvement of over 25%.

TABLE VI: SW DEFECT CLOSURE EFFICIENCY

Milestone	Gen 1	Gen 2	Delta
Release	69%	87%	+26%

$$\text{DCE} = \frac{\text{Cumulative SW defects closed}}{\text{Cumulative SW defects submitted}} \quad (3)$$

Finally, Table VII provides a comparison of the number of days between the official project commit date and the actual customer release date for the two products. As part of project planning, the development team submits a full plan consisting of the features, resources, schedule (including release date), costs and risks. Once this project plan is approved, the project is committed. Factors such as inaccurate estimates, changing customer requirements, and technical problems can cause the actual delivery date to slip. The data shows that the second generation product was released 123 days earlier (or about 22% faster) than the first generation. The cost savings are substantial given the size of the project team (in excess of 100 people).

TABLE VII: PROJECT COMMIT TO RELEASE TIMES

Milestone	Gen 1	Gen 2	Delta
Project Commit to Release	564 days	441 days	-22%

In summary, all five quality indicators displayed substantial improvements at release to customer from Gen 1 to Gen 2 of the software:

- Total software defects: -52%
- Requirements volatility: -47%
- Feature variance: +2.33x
- Software DCE: +26%
- Time from project commit to release: -22%

VI. TEST RESULTS: GEN 2 VS. GEN 3

The next set of data compares the total number of software defects, total number of sightings, feature variance at release, defect closure efficiency and the number of days from project commit to product release between the second and third generation software products. This data was gathered from testing for the two products on all mobile and desktop platforms. This is different from the data presented in Section V, which was for the Gen 1 and Gen 2 products on two specific mobile platforms. Unfortunately, there is no way to extract the data for mobile platforms only from the Gen 3 data as only the combined results for all mobile and desktop platforms were available for that product. Access to this would have allowed a direct comparison among the Gen 1, Gen 2, and Gen 3 products. However, the findings that follow do demonstrate continued improvements on similarly configured hardware.

The total number of software defects and total number of sightings (open issues reported by the test team, not all are

 ISBN: 978-1-61208-283-7

defects) at release for the second and third generation software are shown in Table VII. The reduction in overall defects that started from Gen 1 to Gen 2 continued from Gen 2 to Gen 3, with a decrease of 35%. Total sightings dropped by 31% between releases. These figures are noteworthy due to the increased functionality in the Gen 3 product. Despite an almost tripling in the revision 1.0 PRD length (122 pages vs. 45 pages), total software defects and sighting still declined by sizable percentages.

TABLE VII: TOTAL # SW DEFECTS AND SIGHTINGS AT RELEASE

Milestone	Gen 2	Gen 3	Delta
Total Defects	1,060	690	-35%
Total Sightings	3,800	2,640	-31%

Table VIII displays the feature variance per product at release. The Gen 3 product team was able to deliver about 1.23 times as many supplementary features as the Gen 2 product team at release. Some of this gain is attributable to the continuous build process (and being able to respond to and fix defects faster), but a good percentage is due to the stability of the initial set of requirements. The Gen 3 team could spend more time on newly arriving customer requirements requests.

TABLE VIII: FEATURE VARIANCE AT RELEASE

Milestone	Gen 2	Gen 3	Delta
Release	0.35	0.43	+1.23x

Table IX presents the software Defect Closure Efficiency at first customer shipment for the second generation in comparison to the third generation software. In this table, DCE increased from slightly (about 7%) from 87% in Gen 2 to about 93% in Gen 3. This small gain is not totally unexpected given already high DCE for the second generation product. The quality of the requirements and continuous build process both appear to be positively affecting these figures.

TABLE IX: SW DEFECT CLOSURE EFFICIENCY

Milestone	Gen 2	Gen 3	Delta
Release	87%	93%	+7%

The next set of data in Table X provides a comparison of the number of days between the project commit date to delivery of the released product to the customer. Again, improvements were made from the second to the third generation as 84 days were removed from the schedule (a reduction of about 19%). If the data from the first generation product is included, there is a 207 day or an overall 37% decrease in time from commit date to customer delivery from Gen 1 to Gen 3. Again, this contributed to considerable cost savings based on the team size.

TABLE X: PROJECT COMMIT TO RELEASE TIMES

Milestone	Gen 2	Gen 3	Delta
Project Commit to Release	441 days	357 days	-19%

One final set of data involved customer satisfaction levels. Within the first six months of release, customers were asked to rate the quality of the software delivered according to a four point scale: poor (0), fair (1), good (2) and very good (3). Scores for Gen 1 software averaged between the "fair" to "good" range. For the Gen 2 product, they had moved into the "good" to "very good" range. The Gen 3 product was also in this range, slightly higher than Gen 2.

To summarize, five key quality indicators displayed continued improvements at customer release from Gen 2 to Gen 3 of the software:

- Total software defects: -35%
- Total sightings: -31%
- Feature variance: +1.23x
- Software DCE: +7%
- Time from commit to delivery: -19%

VII. CONCLUSIONS

The key quality result from this extended case study is that the dramatic reductions in total software defects observed from the first to the second generation software on mobile platforms (-52%) continued for the second to the third generation software on the combined mobile and desktop platforms (-35%). A number of factors could have had some impact in these results. They include applying lessons learned from one project to the next, augmented developer experience and maturity, enhanced code review practices and more rigorous unit testing prior to the start of validation. No doubt these factors had some influence on improving software defect levels. However, given the increased complexity of the second and third generation products, these factors should have had a minimal effect on total software defect density levels. Some other factor was playing a dominant role in these extraordinary quality results.

Based on these observations, the improved requirements were the major contributing factor to these reductions in software defects. The project participants noted that the focus on requirements defect prevention appreciably minimized requirements ambiguity, subjectivity and misinterpretation. In addition, non-functional requirements (quality and performance) were written to be more verifiable. The net result was fewer requirement defects propagating to downstream work products like architecture specifications, design documents, code, and test cases. As a result, the development team was able to release code to the test team with fewer defects despite considerable increases in functionality and complexity for the newer versions of the software.

 ISBN: 978-1-61208-283-7

The focus on requirements defect prevention also dramatically impacted other quality indicators including feature variance, defect closure efficiency and project duration. Since less time was spent fixing defects, the developers could spend more time adding feature requests arriving late in the software lifecycle. This is reflected in the increases of feature variance from the first to the second generation (2.33 times) and from the second to the third generation (1.23 times). These late feature additions allowed the teams to be more responsive to changing market conditions, competitive pressures and customer requests.

Defect closure efficiency increased by 26% and 7% respectively from Gen 1 to Gen 2 and from Gen 2 to Gen 3. There are likely three dynamics here: having requirements in a searchable database, better requirements quality and the continuous build process. Determining the source of the defect (e.g., requirement, code, or test case) was facilitated by these good requirements engineering practices.

Another important quality measure improvement was time from project commit to product delivery. From Gen 1 to Gen 3, a total of 207 days were removed from the schedule. Several factors probably influenced these numbers including the quality of the requirements, the requirements database, the merge of the code bases and the continuous build process. For a development team size of over a 100 people, the cost savings are measured in the millions of dollars.

While more subjective, customer satisfaction levels improved from Gen 1 to Gen 3. Scores for Gen 1 software averaged between the "fair" to "good" range. For both the Gen 2 and Gen 3 products, they had moved into to the "good" to "very good" range. This increase in customer satisfaction levels indicates that the focus on requirements produced a software product that more closely met the customer needs and expectations.

An analysis of the data presented in this paper confirms that a well-written and well-reviewed set of requirements is a major factor in overall software quality. They decrease the total number of software defects, minimize rework, reduce wasted effort, improve schedule predictability, and increase team velocity and efficiency. Since test and development teams are spending less time identifying and correcting defects in the code, they can focus more time on productive tasks such as adding functionality and being more responsive to changing customer needs.

VIII. FUTURE WORK

A fourth generation product is currently in development. The number and complexity of the new features continues to grow and almost none of the existing Gen 3 functionality will be removed. It will run on both desktop and mobile platforms (including the new Ultrabook™ products) incorporating a more advanced Intel processor. Software development methodologies remain essentially unchanged, although the team is currently in the process of investigating Scrum.

Similar to its predecessors, this project will leverage the final set of requirements from the previous software (Gen 3). The requirements engineering process is basically unchanged: the requirements SME is still assigned to the team, a requirements management tool is being utilized and the PRD will undergo frequent and comprehensive reviews. The largest variation in methods for this project is in the number of contributors to the PRD. The Gen 3 product had five primary authors, while this product already has over twenty authors. These additional authors are scattered across different sites and countries, making training and mentoring more complicated. To date, the length of the PRD has already tripled. How these requirements changes influence the overall software quality will provide good data for a subsequent paper.

REFERENCES

[1] J. Terzakis, "The impact of a requirements specification on software quality and other quality indicators", 19th IEEE International Requirements Engineering Conference (RE '11), 2011, Trento, Italy.

[2] J. Terzakis, "Requirements defect density reduction using mentoring to supplement training", Seventh International Multi-Conference on Computing in the Global Information Technology (ICCGI 2012), 2012, Venice, Italy.

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

[4] S. V and G. Nair T.R., "Defect management strategies in software development" from Recent Advances in Technologies, edited by M. Strangio, InTech, 2009

[5] T. King and J. Marasco, "What is the cost of a requirement error?", StickyMinds.com, http://www.stickyminds.com/sitewide.asp?Function=edetail&ObjectType=ART&ObjectId=12529&tth=DYN&tt=siteemail&iDyn=2

[6] G. Tassey, "The economic impacts of inadequate infrastructure for software testing", prepared by RTI (project Number: 7007.011), 2002

[7] "CHAOS summary 2009", The Standish Group, 2009.

[8] B. Boehm and V. Basil, "Software defect reduction top 10 list", IEEE Computer, January 2001

[9] D. Reifer, "Profiles of level 5 CMMI organizations." Crosstalk: The Journal of Defense Software Engineering, January 2007

[10] C. Jones, Software Quality: Analysis and Guidelines for Success, International Thomson Computer Press, June 14, 2000.

 ISBN: 978-1-61208-283-7