Examining the Relationships between Performance Requirements and "Not a Problem" Defect Reports

Chih-Wei Ho[1], Laurie Williams[2], Brian Robinson[3]
[1,2]*Department of Computer Science, North Carolina State University*
[1]*dright@acm.org,* [2]*williams@csc.ncsu.edu*
[3]*ABB Inc.*
brian.p.robinson@us.abb.com

Abstract

Missing or imprecise requirements can lead stakeholders to make incorrect assumptions. A "Not a Problem" defect report (NaP) describes a software behavior that a stakeholder regards as a problem while the developer believes this behavior is acceptable and chooses not to take any action. As a result, a NaP wastes the time of the development team because resources are spent analyzing the problem but the quality of the software is not improved. Performance requirements specification and analysis are instance-based. System performance can change based upon the execution environment or usage patterns. To understand how the availability and precision of performance requirements can affect NaP occurrence rate, we conducted a case study on an embedded control module. We applied the Performance Refinement and Evolution Model to examine the relationship between each factor in the performance requirements and the corresponding NaP occurrence rate. Our findings show that precise specification of subjects or workloads lowers the occurrence rate of NaPs. Precise specification of measures or environments does not lower the occurrence rate of NaPs. Finally, the availability of performance requirements does not affect NaP occurrence rate in this case study.

1. Introduction

Missing or imprecise requirements can lead stakeholders to make a variety of assumptions about the correct behaviors of the system. A defect[1] report describes a software behavior that a stakeholder regards as an anomaly. However, the development team may consider this software behavior acceptable and choose not to take any action. The development team designates such a defect report as "Not a Problem." Such decisions are usually made by management, who determine the features to be included in the software product, after discussions among the different stakeholders involved in the development project. A "Not a Problem" defect report, henceforth NaP, wastes the time of the development team and other key stakeholders since resources are spent on analyzing the problem but, in the end, the quality of the software is not improved.

The specification and analysis for performance requirements, such as response time, throughput, and resource consumption, is instance-based [17]. The performance described in requirements documents typically applies to the specified workloads and computation environment. Exhausting all possible conditions for performance requirements is nearly impossible and infeasible. A defect report may describe the system performance under conditions that are different from those specified in the requirements. In this situation, the development team can only decide whether such a defect report is a NaP based on experience. With complete and precise requirements, the defect submitter could avoid reporting a NaP entirely.

Our objective is to examine how missing or imprecise performance requirements affect the occurrence rate of NaPs related to software performance. To assess the precision level of a requirement, we developed a ranking scheme for the precision of performance requirements based on the Performance Refinement and Evolution Model (PREM) [9, 10]. We categorized performance requirements based on their precision level using this ranking scheme. Then we compared the NaP occurrence rates

[1] A defect is a product anomaly [12].

for the defect reports related to the performance requirements in different precision levels.

We conducted a case study to answer the following research questions: 1. What is the cost of investigating NaPs compared to confirmed defects? 2. Are performance-related defects more likely to be designated as NaPs than non-performance-related defects? 3. Do performance requirements specified with higher precision levels have lower NaP occurrence rates? The subject of this case study is a real-time embedded control module from ABB Inc. We collected the documented requirements and defect reports from a firmware development project for this module. Reducing the NaP occurrence rate can improve the efficiency of the development team by saving the time and resources spent on analyzing these defect reports. In this paper, we present the results from this case study, and share our findings of NaP occurrence rate comparisons.

The rest of this paper is organized as follows. Section 2 provides the related work for this study; Section 3 presents PREM and the performance requirements precision ranking scheme; Section 4 gives the data collection criteria and the analysis method we used in the case study; Section 5 shows the context of the case study; Section 6 provides the results and discussions of the case study; we conclude this paper and share the lesson learned in Section 7.

2. Related work

This section provides the related work for this study, including defect analysis, effects of requirements quality, and software performance requirements specification.

2.1. Defect analysis

A defect report describes anomalies found in a software system. Defect analysis can provide us insights into the problems of the software development process. In the Orthogonal Defect Classification (ODC) [5] approach, the analysis of the defect attributes shows the progress of a software project and the effectiveness and completeness of the verification process.

Defect analysis can also be used to improve requirements specifications. Wasson et al. [21] use the failure reports created by testers to identify problematic phrases that occur in requirements documents. In our previous work [11], we demonstrate a process to improve performance requirements specifications from failure reports. Lutz and Mikulski [15] use an adapted ODC approach to analyze how requirements discovery is resolved in testing and operations. The findings of their work suggest that requirements misunderstandings resolved during testing are often related to communication difficulties and subtle interface issues. Additionally, a false-positive defect report, which is a defect report that is resolved without any change in the software, provides potential information for requirements misunderstanding during operation. In our work, a NaP may result from requirements misunderstandings during software development. We examine how precise performance requirements can reduce the occurrence of NaPs.

2.2. Effects of requirements quality

Kamata and Tamai [14] examined the effect of requirements quality on project success and failure for 32 projects. In their work, the requirements quality was evaluated by the software quality assurance team based on the structure of IEEE Recommended Practice for Software Requirements Specifications [13]. The results of their study show that project success is related to requirements specifications. In particular, if the purpose, overview, and general context are well written, the project tends to finish on time and within budget. El Emam and Madhavji [7] propose to use the quality of requirements architecture and the quality of cost/benefits analysis, including 14 measures, in the requirements documents as the measure for successful requirements engineering process. However, they do not explore how the quality of requirements affects the overall software quality. In our work, we focus on the precision of each requirement rather than the quality of the overall requirements specification. Instead of project success, we examine relationships between requirements precision and NaP.

2.3. Performance requirements specification

Performance requirements can be specified qualitatively or quantitatively. Basili and Musa advocate that quantitative specification for the attributes of a final software product will lead to better software quality [4]. For performance requirements, Nixon suggests that both qualitative and quantitative specifications are needed, but different aspects are emphasized at different stages of development [16].

A performance meta-model describes the elements of a performance concept. Cortellessa [6] provides an overview of three performance meta-models, including UML Profile for Schedulability, Performance, and Time [17], Core Scenario Model [18], and Software Performance Engineering meta-model [19]. From

these three meta-models, Cortellessa generalizes a software performance concept into three areas: software behavior, workload data, and resources data. Adapted from Cortellessa's generalization, we rank the precision of performance requirements from four factors: subject, measure, environment, and workload. Section 3.2 provides the detail information for the ranking scheme.

3. Performance requirements precision ranking scheme

To compare the precision levels of performance requirements, we create a ranking scheme for performance requirements precision based on Performance Refinement and Evolution Model (PREM). In this section, we introduce PREM and the performance requirements precision ranking scheme used in this study.

3.1. Performance Refinement and Evolution Model

PREM is a guideline for evolutionary performance requirements refinement. A development team may use PREM to specify simple performance requirements early in the software lifecycle, and add more details in the performance requirements when the team gains more knowledge of the system performance. PREM is designed for the specification of three categories of requirements: response time, throughput, and resource consumption. PREM defines four levels of detail for a performance requirement. The levels are summarized in Table 1.

Table 1. The goals of PREM at each level

Level	Goal
PREM-0	• Identify the performance focus.
PREM-1	• Specify and verify the quantitative performance measures.
PREM-2	• Specify the execution environment. • Estimate the workloads. • Verify the quantitative measures.
PREM-3	• Collect execution environment and workload information from the field. • Adjust the performance requirements.

In PREM, the specification of a performance requirement starts at PREM-0, with a qualitative description. The qualitative descriptions point out the performance focus in the software. They serve as the starting points for requirements. The stakeholders will refine or evolve the requirements to more precise specifications throughout the software development lifecycle.

The development team refines a PREM-0 requirement to the PREM-1 level by including a quantitative measure. The performance measure needs to be meaningful and obvious to the customer. After quantitative requirements are specified, the development team can discuss with the customer whether the specified performance is feasible and acceptable.

In the PREM-2 level, more factors that can affect the performance are added to the requirements specification. Such factors include workload and execution environment. These factors can vary greatly in different deployment sites. Furthermore, the workload can be unpredictable before the system is put in the real world [20], making the decision of computation resources difficult. After the precise requirements are specified, they can be verified with performance prediction models such as a queueing network model. If the performance expectation of a requirement is not feasible, the development team needs to negotiate with the customer for a more reasonable and practical requirement.

Once the software system is in early release, such as a beta testing phase, continuously monitoring the performance can help us understand the workload of the system and how the workload affects the performance. The performance requirements in the PREM-3 level describe the actual workload and execution environment in the production environment. Experiences show that usually two to twelve months are required to collect representative workload data [3]. Even though PREM-3 requirements may be available late in the software development cycle, these requirements provide useful information if the software is going into a new release or is deployed in a new environment.

3.2. Performance requirements precision ranking scheme

Cortellesa generalizes a performance concept into three areas: software behavior, resources, and workload [6]. We adapted Cortellessa's generalization, and rank the performance requirements precision from four factors.

- *Subject* for which a performance requirement is specified, e.g. the elapsed time for online upgrade.

- *Measure* that describes the expectation for the performance of the software, e.g. 30 seconds.
- *Environment* that describes the computational resources available in the runtime environment, e.g. the recommended hardware configuration.
- *Workload* that specifies the demand intensity for the software, e.g. 20 devices that send 10 messages per second.

For each factor, we define several non-overlapped precision ranks based on PREM. We also assign a score for each rank. Table 2 shows the precision ranks, scores, and associated PREM levels for each factor.

Table 2. PR precision ranking scheme

Factor	Precision Ranks (Score)	PREM Levels
Subject	Scenario (2) Function (1) Scope (0)	
Measure	Quantitative (1)	1 and above
	Qualitative (0)	0
Environment	Specified (1)	2 and above
	Unspecified (0)	0 – 1
Workload	Quantitative (1)	2 and above
	Qualitative or unspecified (0)	0 – 1

In the ranking scheme, a *subject* can be a scenario, a function, or a scope. A scenario is a linear series of events [2] that leads to specified performance. A function is functionality the software system provides. A scope can be any subset of the software system and may including the whole system. A scenario describes a sequence of user actions or software behaviors. For example, "load the testing project in the controller → start the testing project → start the upgrade program from the in-field computer." A function describes one or more software behaviors or user/software interactions, e.g. "upgrading the firmware in the controller." A scope is a software component used in the system, e.g. "the firmware upgrade program." The whole system, if used as the subject of a requirement, is considered as a scope.

We rank a *measure* in two levels: quantitative and qualitative. A quantitative measure is specified with a number or a range of numbers with a measurement unit. A qualitative measure is a textual description of the performance. We consider the measure of a comparative requirement, such as "as fast as the last version," as qualitative.

We rank the *environment* factor based on its availability. The environment of a comparative requirement is categorized as not specified.

For the *workload* factor, the rank is based on whether a quantitative specification is available. If a qualitative workload is specified, the defect reporter still needs to make assumptions on the workload information as with unspecified workloads. Therefore, we rank qualitative and unspecified workloads with the same precision level. We categorize the workload for a comparative requirement as qualitative.

One may rank a comparative requirement based on its related requirement from an earlier version. However, the requirement from an earlier version may not be available. Additionally the development team may not be able to use the same performance measurement approach on an earlier release. Therefore, we rank comparative requirements as low precision in our scheme.

4. Case study

In this section, we provide details for the data collection and analysis steps used in this case study.

4.1. Data collection details and limitations

We collected and examined the defect reports from the case study subject. The source of the defect reports included test failures, customer-reported problems, manager requests, and team tasks. Due to the purpose of the study, we did not include the defect reports that were unrelated to the behaviors of the software under development. For example, in this case study, the development team used the defect tracking system for task planning. A defect report might describe the tasks the team planned to do, including features to add and updates to make. We do not include such defect reports in our analysis, as they did not describe any anomalous behavior of the system.

We cannot be sure whether a defect report is a NaP before it is closed, so we only included closed defect reports in the analysis. A development team might defer the resolution for a defect report due to lack of information, resource constraints, or release time pressure. Therefore, we cannot determine whether a deferred defect report is a NaP, and do not include these reports in our analysis.

We classified a defect report as performance-related and non-performance-related. We determine whether a defect is performance-related based on the symptom described in the defect report. If a defect report describes a symptom of unsatisfactory response time, throughput, or resource consumption, we consider it as performance-related. A non-performance-related defect may be caused by performance-related issues. For

example, an operation may fail because a dependent device does not respond in time. Such a defect is manifested as a functional problem. Therefore, we do not consider the defect as performance–related. The only exception is an unexpected program stop caused by insufficient memory. If a defect report states that insufficient memory caused an unexpected crash, we consider the defect as performance-related.

4.2. Analysis steps

Our analysis requires information on the traceability of defects to requirements. The NaP occurrence rate for a set of defect reports is calculated by dividing the number of NaP by the number of defect reports in the set. The confirmed rate of the same set of defect reports is (1 – NaP occurrence rate). We then apply a chi-square test to examine whether the effect of requirements availability is significant. To protect proprietary information, we only report proportional results in this paper. However, actual numbers were used in the statistical analysis.

We examine the effect of performance requirements precision on the occurrence rate of NaP from four factors: subject, measure, environment, and workload. To analyze the effect of each factor, we categorize the requirements based on their precision levels. We use the chi-square test to examine the significance of the relationship between NaP occurrence and the precision of each performance factor. To analyze the effect of multiple factors, we also use the summation of the precision scores from these factors as an overall precision score. We use a linear-by-linear association test to examine the relationship between the NaP occurrence rate and the overall performance requirements precision levels.

One should use appropriate statistical analysis approach based on the case study data to draw conclusions. The data in this case study is categorical in nature. Agresti provides extensive review of categorical data analysis methods in his book [1].

5. Case Study Context

We conducted a case study on a firmware development project for a real-time embedded control module from ABB Inc. This firmware supports a line of controllers with different computation and memory capacities. A controller is a processing unit that can control other devices, e.g. motors or turbines, and overall processes, e.g. power generation. A controller is highly configurable and is capable of running a large variety of processes, from small factories to large power plants.

The firmware was implemented as a hybrid of procedurally designed C code and object-oriented designed C++ code. The development team consisted of approximately eighty developers and twenty testers. The development and testing departments are distributed around the world.

This controller module has been used in the field for almost ten years. When we started this case study, the firmware had been through five major releases. We collected the defects that were found on the fifth major release.

In this case study, the source of the defect reports includes management, globally distributed testing teams, and customer failure reports. Many internal stakeholders are involved within the development team in this case study. Before a defect report is designated as a NaP, developers, management, and testers may participate in the investigation. However, the final decision makers are product line management and the main influence of their decision is the development department.

6. Results and Discussion

In August 2007, we collected the defect reports and documented requirements for a real-time embedded control module from ABB Inc. We analyzed more than 1,500 defect reports in this study. We also collected 33 documented performance requirements. In this section, we present and discuss the results of the analysis.

6.1. NaP and confirmed defect reports

Within the collected defects, 21.20% were designated as NaP. In this section, we explore the differences between NaP and confirmed defect reports, from the perspectives of communication, defect resolution time, and defect severity. We cannot measure the exact amount of communication solely from defect reports. Therefore, we use the number of informational notes added to a defect report as a surrogate measure of the amount of communication that occurred concerning the defect. The approximation is reasonable, as many of the notes represented weekly or monthly status update meetings by key internal stakeholders throughout the development project. We also cannot measure the defect resolution time solely from defect reports. We use the elapsed time that the defect remained unresolved to demonstrate the defect report resolution time. Defect severity is ranked from five to one with five being the highest level. Table 3

summarizes the differences between NaP and confirmed defect reports in this study.

Table 3. NaP and confirmed defects comparison

		NaP	Confirmed
Communication (as # of Notes)	Avg.	2.52	2.92
	Var.	4.41	6.11
Resolution Time (as Elapsed Time, Days)	Avg.	11.01	14.54
	Var.	230.88	206.02
Defect Severity	Avg.	2.53	2.73
	Var.	0.78	0.74

6.1.1. Communication. When the development team analyzed a defect report, team members added notes to the defect report to exchange opinions and ideas with each other while product management added notes to accept, reject, or comment on key issues. While we cannot measure the exact amount of communication the development team used to resolve each defect, we use the number of notes the team members added to the defect report as an approximation for this measure. The results show that, on average, developers use more notes in confirmed defect reports than in NaPs. We choose Wilcoxon-Mann-Whitney test to examine the distribution difference between NaP and Confirmed defect reports. The result shows that the difference between NaPs and confirmed defect reports, in terms of the number of notes, is statically significant ($P = 0.01$). To estimate the magnitude of difference, we use Hodges-Lehmann Estimation. However, the confidence interval on the magnitude of difference is 0.00, indicating that the sample variation is too high to make a good estimate. Therefore, we cannot estimate the difference between the numbers of notes used in NaPs and confirmed reports.

6.1.2. Resolution time. To estimate the amount of time needed to resolve a defect report, we use the elapsed time for the defect report from the submission of the report to the time it was disposed. We only include reports with severity levels four and five, and with an elapsed time of less than 60 days. The reason for only including these reports is that low severity defects were often ignored to provide timely response to more critical defects. Additionally, if a defect remained open for more than 60 days, either the priority for fixing the defect was low or the defect was not easily repeatable. In either case, the defect elapsed time does not accurately reflect the defect resolution time. In this case study, the average resolution time for a confirmed defect report is longer than that for a NaP. However, the elapsed time for a confirmed defect also includes the time to validate whether the defect has been fixed. For a NaP, because the developers do not change the software, detailed validation is unnecessary and only product management agreement is required. We use Hodges-Lehmann Estimation to examine the difference of the time difference for confirmed and NaP defect resolution. The result shows that, at 95% confidence level, confirmed defect report elapsed time is one to seven days longer than NaP elapsed time.

6.1.3. Defect severity. In this case study, the average severity of a NaP defect is lower than that of a confirmed defect (z-test, $\alpha = 0.05$). When a defect report shows higher severity, the development team tends to be more conservative before designating the defect report as NaP. Therefore, we expected that the average severity level for NaP to be lower than that for confirmed defect reports. The results from this case study show the same trend.

6.1.4. Summary. The results from this case study show that the amount of resolution time is reasonably at the same level as confirmed defect reports, even though the average severity of NaP is lower,. Furthermore, designating a defect report as a NaP, which account for 21.20% of defect reports, did not help the team improve the overall software quality. The efficiency of the development team can be improved if we lower the NaP occurrence rate.

6.2. Performance and non-performance defects

In the case study, only 2.6% of defect reports are related to performance. Table 4 summarizes the comparison between performance and non-performance defect reports. In this case study, the average number of notes used in performance defect reports is higher than in non-performance defect reports. The difference is statistically significant (Wilcoxon-Mann-Whitney test, $P < 0.01$). The observed average elapsed time for performance defect reports is longer than for non-performance defect

Table 4. Performance and non-performance defect comparison

		Performance	Non-Perf
Communication (as # of Notes)	Avg.	4.62	2.85
	Var.	15.79	5.72
Resolution Time (as Elapsed Time, Days)	Avg.	21.35	13.61
	Var.	394.86	207.46
Defect Severity	Avg.	2.96	2.68
	Var.	0.41	0.76
NaP Rate		36.69%	20.75%

reports. However, the difference is not statistically significant (Wilcoxon-Mann-Whitney test, $P = 0.17$). The average severity level of performance defect reports is higher than of non-performance defect reports (z-test, $\alpha = 0.05$). The NaP occurrence rate is also statistically higher for performance defect reports (chi-square test, $\alpha = 0.05$).

The control module in this case study is used in real-time systems. Responsive performance is required for the module to function properly. The higher average severity level reflects the team's emphasis on performance. The result also shows that performance defect report resolution required more time and team communication. Therefore, a performance defect report is likely more expensive to resolve than a non-performance defect report. The development team could have saved a lot of resources if the NaP occurrence rate were lower for performance defect reports.

6.3. Performance requirements availability and NaP occurrence rates

We could only identify the source of requirements documents for 50% of the performance defects. Table 5 shows the NaP occurrence rates for performance defects reports with and without documented requirements. For this analysis, the explanatory variable is the availability of requirements, and the response variables are the numbers of NaP and confirmed defect reports. The defects with specified requirements have a higher NaP occurrence rate in this case study. The difference is not statistically significant (chi-square test, $\alpha = 0.05$).

Table 5. Availability of PR and NaP occurrence rates

	NaP	Confirmed
With Req	36.36%	63.64%
Without Req	34.78%	65.22%
Total	35.56%	64.44%

We expected the NaP occurrence rate to be lower for defects that are traceable to documented requirements. However, the results show the opposite trend, although not statistically significant. In this case study, the higher NaP occurrence rate for performance defects with specified requirements can be explained with two reasons. First, the subject of the case study is a mature product. The development team knew the reasonable performance for the control module. When a stakeholder reports a performance defect, documented and implicit performance requirements play equally important roles. Second, the documented requirements did not provide enough precision. Therefore, a stakeholder may file a defect report based on a wrong assumption. We discuss the effect of performance requirements precision in the next subsection.

6.4. Performance requirements precision and NaP occurrence rates

We analyzed the relationship between the performance requirements precision and NaP occurrence rate with two approaches using the ranking scheme described in Section 3.2. First, we analyzed the effect of each factor in the ranking scheme to see how each performance factor affects NaP occurrence rate. Second, we analyzed the effect of combined factors to show how combined factors affect NaP occurrence rate.

6.4.1. Single factor analysis. For the performance defects with related, documented requirements, we calculated their NaP occurrence rates based on the precision of the related requirements. We rank the precision of performance requirements from four factors: subject, measure, environment, and workload. In this analysis, the explanatory variables are the precision ranks for each factor, and the response variables are the numbers of NaP and confirmed defect reports. Table 6 shows the NaP occurrence rates for defects reported against the different ranks of requirements precisions, grouped by performance factors. Workload shows a statistically significant effect of requirements precision on the NaP occurrence rate (chi-square test, $\alpha = 0.05$). The effects of the other three factors are not statistically significant.

Table 6. PR precision and NaP occurrence rates

Factor	Precision	NaP	Confirmed
Subject	Function	25.00%	75.00%
	Scope	50.00%	50.00%
Measure	Quantitative	44.44%	55.56%
	Qualitative	30.77%	69.23%
Environment	Specified	33.33%	66.67%
	NA	38.46%	61.54%
Workload	Quantitative	0%	100.00%
	NA	47.06%	52.94%

If, in a performance factor, the NaP occurrence rate is lower for the higher precision rank, the precision of documented requirements plays an important role when the development team decides whether a defect report is NaP. In this case study, only the subject and workload factors demonstrated this trend. The development team could lower the NaP rate by

specifying more precise specifications for these two factors. On the other hand, the precise specification of measure and environment factors did not lower the NaP occurrence rate. The decision of whether a defect report is a NaP might have come from resources other than the requirements. In this case study, some testing documents provided very detailed description of the testing environment. Without documented requirements for the runtime environment, the team could still use the information from the testing documents to determine whether a defect report is NaP.

6.4.2. Cross-factor analysis. We use the summation of the precision scores from the four factors to quantify the overall precision of a performance requirement, and find out the NaP occurrence rate for each overall precision level. Additionally, we calculated the precision score for a performance requirement from the two important factors identified in the single factor analysis. In this analysis, the explanatory variables are the precision scores, and the response variables are the number of NaP and confirmed defect reports. The results are shown in Table 7. A higher overall precision score lowers the NaP occurrence rate. However, the effect is not statistically significant (linear-by-linear association test, $\alpha = 0.05$). A higher precision score from the subject and workload factors also lowers the NaP occurrence rate for the related defect reports. This effect is statistically significant (linear-by-linear association test, $\alpha = 0.05$). The results again show that, for this case study, specification of precise subject and workload information with performance requirements is statistically related to a lower NaP occurrence rate.

Table 7. Precision scores and NaP occurrence rates

Score	NaP Occurrence Rate	
	Overall	Subject + Workload
0	44.44%	50.00%
1	33.33%	42.86%
2	NA	0%
3	37.50%	
4	0%	

6.4.3. Threats to validity. Construct validity involves establishing the measures for the concepts being measured [22]. We use a coarse-grained ranking scheme to evaluate the precision levels of performance requirements. In each performance factor, we only define two or three precision levels. Therefore, even if two requirements specify a particular performance factor at the same level, they may provide different amounts of details. A finer-grained ranking scheme can yield more detailed information on the requirements precision than the ranking scheme proposed in this paper. However, we believe a finer-grained ranking scheme needs to be domain-specific. The benefit of the ranking scheme proposed in this paper is the wide applicability. Any performance requirement can be evaluated with the precision ranking scheme.

Internal validity in this case study concerns the degree of cause-effect relationship between the precision levels of a requirement and the related NaPs. The cause of NaPs can be very complicated. Factors such as implicit knowledge and lexical correctness of requirements specification may also cause NaPs. We only analyzed the effect of performance requirement preciseness in this case study. The results also suggest that factors other than requirements preciseness played an important role for the cause of NaPs. We will need further investigations to find out other causes of NaPs and their effects on NaP occurrence rate.

External validity is the degree to which this case study can be generalized. The case study discussed in this paper is the sixth release from the same organization. We believe we can find similar results from earlier releases. However, we do not know if ABB software systems are representative of all industrial systems. Other organizations may use the results reported in this paper as a starting point for their own investigations. Although the effect of performance requirements precision on NaP occurrence rates may vary among development teams, the degree of the effect provides information on performance requirements improvement. We are developing an approach to improve performance requirements based on the degrees of the effects of performance requirement precision on NaP occurrence rates.

6.5. Additional observations

The customer is the ultimate judge of the software quality [8]. In this case study, only 1.45% of the defect reports were submitted by the customer. The customer reported only one performance-related defect. The low number of performance defect reported by the customer showed the high maturity of this software product in the regard of performance. Furthermore, this performance defect was not related to any defect that the development team designated as NaP during development.

Although not observed in this case study, the customer may file a problem report that the development team originally consider as NaP before release. The NaPs that turn out to be real problems after release reveal misunderstandings between the

customer and the development team, and can be prevented with rigorous requirements elicitation activities. On the other hand, our focus in this paper is on requirements misinterpretation within the development team. The development can use the results from this case study to prevent such misinterpretations with more specific requirements.

The development team fixed most of the customer-reported defects, including the one related to performance, in service packs. The team designated 16.67% of the customer defect reports as NaPs. We observed two themes in the customer NaPs. First, the customer had incorrect expectations due to misunderstandings of terminologies. For example, a customer reported that some messages in the "event list" did not appear in the "event and alarm list." However, the event and alarm list was designed to filter out some particular messages, while the event list shows all messages. These types of misunderstandings were resolved with documentation changes. Second, the customer used an inappropriate configuration in the control module. For example, a customer reported that some feature of the control module did not work with their customized program. After investigation, the development team found out that the customized program was the cause of the problem. These types of customer defects are resolved by customer support.

Among the performance defect reports with traceable requirements we collected in this case study, we found that 45% of defects were opened against the requirements that used scopes to specify their subjects, which accounted for 14.81% of the total performance requirements. A requirement specified with a scope is more general than a requirement specified with a more precise subject. However, if a defect is opened against a requirement with a lower precision subject, the defect is more prone to be designated as a NaP, as shown in the case study results. In this case study, 62.50% of the NaPs were designated as NaPs because the subject described in the defect report is more precise than specified in the related requirements. The development team closed such NaPs because they were "exceptions," as stated in some notes in the defect reports. However, those NaPs could have been avoided if the "exceptions" were specified in the requirements. In our previous work [11], we present a systematic approach to integrate such information into the requirements for future software releases.

7. Conclusion and lessons learned

In this paper, we explore the relationships between the quality of performance requirements and NaP occurrence rate. We examined defect reports in this case study, and found out that a NaP was as expensive to resolve as a confirmed defect report. Additionally, the NaP occurrence rate was significantly higher for performance-related defect reports than for non-performance-related defect reports. Designating a defect report as NaP does not improve the software quality. Therefore, reducing the NaP occurrence rate can save the team the time required to resolving such defect reports. With complete and precise requirements, the defect submitter can avoid reporting the NaP at all.

We first examined the effect of the availability of performance requirements. The results show that the NaP occurrence rate was similar for the defects with and without traceable performance requirements. Although the availability of performance requirements did not lower the NaP occurrence rate, this result suggests that the specified requirements were not precise enough for the team to determine whether a defect report is a NaP.

We then used a performance requirements precision ranking scheme based on PREM to quantify the precision of the performance requirements. This case study shows that the performance requirements specified with quantitative workloads had significantly lower NaP occurrence rate than those without. Performance requirements with a function description also had lower NaP occurrence rate than those specified with just a scope, although the difference was not statistically significant. A cross-factor analysis shows that the NaP occurrence rate had a lowering trend when the performance requirements were more precise. If we only consider the precision of the subject and workload, the effect of performance requirements precision on NaP occurrence rate was statistically significant.

In this case study, the precision for some performance factors did not show statistically significant effects on NaP occurrence rate. However, comparing the NaP occurrence rates for the requirements with a variety of precisions shows a direction for requirements process improvement. Documented requirements specification is not the only source for the development team to determine whether a defect report is a NaP. If a development team depends on the experience more than documented requirements to determine whether a defect report is NaP, we might observe a low NaP occurrence rate of the defect reports related to requirements specified with a low precision level. Incorrectly designed testing environment or a wrong testing tool may lead to erroneous testing results, and the tester might create

defect reports that describe false-positive problems. Therefore, an overall high NaP occurrence rate may also indicate a deficiency in the information available to testing teams.

Our previous work [11] provides a systematic approach to integrate failure reports from the customer into requirements specifications. In this paper, we focus on the analysis of the defects during development. We believe the investigation of NaPs can improve the communications with the customer. However, we do not have the data related to customer communication in this case study. We will continue our work in two directions. First, we have collected data from several industrial software projects. We will apply similar analysis on these projects. Second, the NaP occurrence rate comparison presented in this paper also shows the direction for requirements improvement. We will build and validate a requirements improvement framework based on NaP occurrence rate comparison.

8. References

[1] Agresti, A., *Categorical Data Analysis 2nd Edition*, New York, NY, Wiley Inter-Science, 2002.

[2] Alspaugh, T. A., A. I. Antón, T. Barnes, and B. W. Mott, "An Integrated Scenario Management Strategy," in *Proceedings of the 1999 International Symposium on Requirements Engineering*, pp. 142-149, Ireland, Jun 1999.

[3] Avritzer, A., J. Kondek, D. Liu, and E. J. Weyuker, "Software Performance Testing Based on Workload Characterization," in *Proceedings of the 3rd International Workshop on Software and Performance*, pp. 17-24, Rome, Italy, Jul 2002.

[4] Basili, V. R. and J. D. Musa, "The Future Engineering of Software: A Management Perspective," *IEEE Computer*, vol. 24, no. 9, pp. 90-96, Sep 1991.

[5] Chillarege, R., I. S. Bhandari, J. K. Chaar, M. J. Halliday, D. S. Moebus, B. K. Ray, and M.-Y. Wong, "Orthogonal Defect Classification -- A Concept for In-Process Measurements," *IEEE Transactions on Software Engineering*, vol. 18, no. 11, pp. 943-956, Nov 1992.

[6] Cortellessa, V., "How Far Are We from the Definition of a Common Software Performance Ontology," in *Proceedings of the 5th International Workshop on Software and Performance*, pp. 195-204, Illes Balears, Spain, Jul 2005.

[7] El Eman, K. and N. H. Madhavji, "Measuring the Success of Requirements Engineering Processes," in *Proceedings of the 2nd International Symposium on Requirements Engineering*, pp. 204-211, York, UK, Mar 1995.

[8] Fox, C. and W. Frakes, "The Quality Approach: Is It Delivering?," *Communications of the ACM*, vol. 40, no. 6, pp. 24-29, Jun 1997.

[9] Ho, C.-W. and L. Williams, "Deriving Performance Requirements and Test Cases with the Performance Refinement and Evolution Model (PREM)," Department of Computer Science, North Carolina State University *Technical Report No. TR-2006-30*, Nov 2006.

[10] Ho, C.-W. and L. Williams, "Developing Software Performance with the Performance Refinement and Evolution Model," in *Proceedings of the 6th International Workshop on Software and Performance*, pp. 133-136, Buenos Aires, Argentina, Feb 2007.

[11] Ho, C.-W., L. Williams, and A. I. Antón, "Improving Performance Requirements Specifications from Field Failure Reports," in *Proceedings of the 15th International Requirements Engineering Conference*, pp. 79-88, New Delhi, India, Oct 2007.

[12] IEEE, *IEEE Std. 982.2-1988: IEEE Guide for the Use of IEEE Standard Dictionary of Measures to Produce Reliable Software*, 1988.

[13] IEEE, *IEEE Std 830-1998: IEEE Recommended Practice for Software Requirements Specifications*, 1998.

[14] Kamata, M. I. and T. Tamai, "How Does Requirements Quality Relate to Project Success or Failure?," in *Proceedings of the 15th International Requirements Engineering Conference*, pp. 69-78, New Delhi, India, Oct 2007.

[15] Lutz, R. R. and I. C. Mikulski, "Resolving Requirements Discovery in Testing and Operations," in *Proceedings of the 11th IEEE International Requirements Engineering Conference*, pp. 33-41, Monterey Bay, CA, Sep 2003.

[16] Nixon, B. A., "Managing Performance Requirements for Information Systems," in *Proceedings of the 1st International Workshop on Software and Performance*, pp. 131-144, Santa Fe, NM, Oct 1998.

[17] OMG, *UML Profile for Schedulability, Performance, and Time Version 1.1*, 2005.

[18] Petriu, D. B. and M. Woodside, "A Metamodel for Generating Performance Models from UML," in *Proceedings of the 7th International Conference of the UML*, pp. 41-53, Lisbon, Portugal, Oct 2004.

[19] Smith, C. U. and C. M. Lladó, "Performance Model Interchange Format (PMIF 2.0): XML Definition and Implementation," in *Proceedings of the 1st International Conference on the Quantitative Evaluation of Systems*, pp. 38-47, Enschede, The Netherlands, Sep 2004.

[20] Trott, B., "Victoria's Secret for Webcasts Is IP Multicasting," *InfoWorld*, Aug 1999.

[21] Wasson, K. S., K. N. Schmid, R. R. Lutz, and J. C. Knight, "Using Occurrence Properties of Defect Report Data to Improve Requirements," in *Proceedings of the 13th International Requirements Engineering Conference*, pp. 253-262, Paris, France, Aug 2005.

[22] Yin, R. K., *Case Study Research: Design and Method, 3rd Edition*, Sage Publications, 2003.