2008

SOFTWARE DEFECT REDISCOVERIES: CAUSES, TAXONOMY AND SIGNIFICANCE

Shyamsundar Kulkarni
Western University

SOFTWARE DEFECT REDISCOVERIES: CAUSES, TAXONOMY AND SIGNIFICANCE

(Spine title: Software Rediscoveries: Causes, Taxonomy and Significance)

(Thesis format: Monograph)

by

Shyamsundar Kulkarni

Graduate Program in Computer Science

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science

Society of Graduate and Postdoctoral Studies
The University of Western Ontario
London, Ontario, Canada

THE UNIVERSITY OF WESTERN ONTARIO
SCHOOL OF GRADUATE AND POSTDOCTORAL STUDIES

CERTIFICATE OF EXAMINATION

Supervisor	Examiners
Dr. Nazim Madhavji	Dr. Mike Bauer
Supervisory Committee	Dr. Mark Perry
Dr. Mechelle Gittens	Dr. Matt Davison

The thesis by

Shyamsundar Kulkarni

entitled:

Software Defect Rediscoveries: Causes, Taxonomy and Significance

is accepted in partial fulfillment of the
requirements for the degree of
Master of Science

Date_____

_____
Chair of the Thesis Examination Board

Abstract

A software defect rediscovery is a software failure caused by a previously reported defect. It has been observed that a majority of field software failures are rediscoveries which account for typically 50% but sometimes as much as 90% of the total failures. A number of causes for defect rediscoveries have been identified in the literature and solutions have been designed to address some of them. For an organization aiming at reducing the cost due to the rediscoveries, it is important to understand the significance of each of these causes. The significance of each cause will guide the organization to utilize the known solutions or design new ones if necessary, to ultimately reduce the cost to the organization due to rediscoveries.

In this thesis, we identify and define the causes for rediscoveries, both on the side of the software provider as well as the software user and design a taxonomy for these causes. We establish the significance of each of the causes for rediscoveries by conducting two exploratory based empirical case studies.

The overall findings of this study suggest that the delay on the software providers' side to provide the patch contributes to approximately 50% of the rediscoveries; whereas, that on the software users' side to install the patch contributes to approximately 50% of the rediscoveries. This overall result is further broken down quantitatively into specific causes, which are all described in this thesis.

From a practitioner's point of view, the results of this investigation will provide the decision support regarding the designing and prioritizing of various policies and solutions aimed at reducing the cost due to rediscoveries. From a research point of view, the results of our investigation add to the existing body of knowledge on the causes for rediscoveries in software systems.

Keywords: software engineering, software rediscoveries, causes for rediscoveries, empirical study, and software quality

Acknowledgements

First and foremost, I would like to thank my supervisor, Dr. Nazim H. Madhavji, and my co-supervisor, Dr. Mechelle Gittens, for their guiding support, motivation and supervision in the carrying out of this research.

I am also thankful to all my colleagues: Andriy Miranskyy and Remo Ferrari for numerous discussions that helped in refining my research investigation; Zude Li, Syed Shariyar Murtaza and Colin Taylor for their help with data gathering; Quazi Rahman for his helpful comments and suggestions during my research.

I would like to thank Mark Wilding and Dave Godwin from IBM, Toronto for their help with research validation and overall support throughout the research process.

I also would like to thank Mitch Dawson and Marguerite Doyon from Research Infotech, London for their help with research design and validation as well as data gathering.

Lastly, I would like to thank the participants of the study; I greatly appreciate the effort that was expended in the research, and hope that the research output will be of good value to everyone.

TABLE OF CONTENTS

List of Tables

List of Figures

Chapter 1: Introduction

1.1 Motivation

It is difficult to create a very large, complex software system that is completely free of defects [Adams 1984]. Once the software, whether the very first release or subsequent enhancements with new features, is developed and released to the field many users run the software and report problems. These problems are addressed by the software provider in several steps. Problems are subjected to diagnosis to identify the defect, fixes for these defects are created, and interim versions of the software are released to the field [Lee Iyer 2000]. The interim versions containing defect fixes are known as software patches or more simply as patches. This process is represented in figure 1.

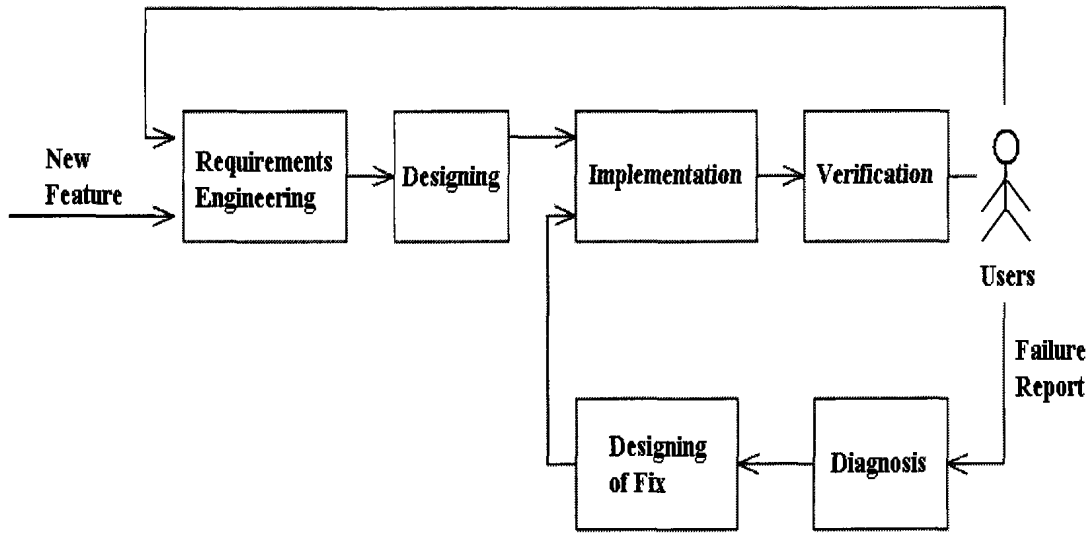


Figure 1 Software Development and Service Process

A software defect rediscovery, henceforth referred to as rediscovery, is a software failure caused by a previously reported defect. It has been observed that a majority of field software failures are rediscoveries. The study by Lee and Iyer shows that about 70% of the reported software failures are rediscoveries [Lee Iyer 2000]. The study by Alan Wood shows that more than 50% of the reported software failures are rediscoveries [Wood 2003]. The study by Brodie et al. shows that rediscoveries are

common in large software products and form a major cost component of product support [Brodie 2005]. The research further quantifies the number of rediscoveries as accounting for typically half and sometimes as many as 90% of the software failures.

Rediscoveries exist for several reasons. Firstly, resolving a failure to identify a defect, designing, testing and disseminating of the fix of the software defect can take a significant amount of time. In the meantime, the defect can cause failures in the same user site or different user sites [Lee Iyer 2000]. Secondly, patch installation requires system downtime. However, business requirements limit the amount of downtime available to the users. This may force the users to postpone the patch installation to the next available maintenance opportunity. This delay can cause rediscoveries [Lee Iyer 2000] [Baumann 2004] [Baumann 2005] [Altekar 2005]. Thirdly, a purported patch for a defect can fail [Lee Iyer 2000]. Fourthly, users who do not experience the failure due to a particular defect may not be inclined towards installing the available patch for that defect for fear that doing so might cause new problems, as is sometimes the case with patches [Lee Iyer 2000] [Altekar 2005] [Ballintijn 2005] [Gerace 2005] [Jansen 2005] [Gkantsidis 2006] [Pasala 2006] [Crameri 2007] [Pasala 2008]. Users like to test the patches before installation to overcome the skepticism that it might cause new problems [Beattie 2002]. This delay could also lead to rediscoveries. Also, some users limit the number of patches they install on their systems since they want to keep the changes made to their systems minimal due to this skepticism [Crameri 2007]. This could lead to more rediscoveries. Fifthly, patch management involves a number of sub-processes like analyzing the risk to which the systems are exposed, testing the patches before installation, the actual patch installation on production systems, etc which necessitates considerable IT staff resource. Some users do not have adequate IT staff resource for this purpose, which forces them to limit the frequency of patch installation on their systems [Beattie 2002] [Gerace 2005]. Finally, the complexity of the patch installation process might cause the users to limit the frequency of patch installation on their systems [Wood 2003].

A variety of approaches have been proposed to reduce the product support cost due to rediscoveries. The research by Adams [Adams 1984], Cobb et al. [Cobb 1992], Lee, Pitt and Iyer [Lee Pitt Iyer 1996], Lee and Iyer [Lee Iyer 2000], and Brodie et al. [Brodie 2005] are all efforts in this direction, each directed at one or more causes for the rediscoveries discussed before. However none of these solutions individually can be a "silver bullet" against the rediscoveries since none of them address all the causes for rediscoveries. An assortment of these solutions and some new ones may be necessary to eliminate most of the cost due to rediscoveries. Hence, for an organisation aiming at reducing the cost due to the rediscoveries, it is important to understand the significance of each of these causes for rediscoveries. The significance of each cause will guide the organization to utilize the known solutions or design new ones if necessary to ultimately reduce the cost to the organisation due to rediscoveries.

1.2 Purpose of the Study

A number of causes for rediscoveries have been mentioned in the literature as identified in section 1.1. Also, a variety of approaches have been mentioned in the literature which target one or more of these causes for rediscoveries as identified as well in section 1.1 and further elaborated on in chapter 2. However none of these solutions individually addresses all the causes[1] for rediscoveries. An assortment of these solutions and some new approaches may be necessary to eliminate most of the cost due to rediscoveries. Hence, for an organisation aiming at reducing the cost due to the rediscoveries, it is important to understand the significance of each of these causes for rediscoveries. The significance of each cause will provide decision support for designing and prioritizing of various policies and solutions targeted at the rediscovery causes so that associated costs can be reduced.

From a software engineering practitioner's point of view the knowledge regarding various causes for rediscoveries and their significance will guide an organisation, which is aiming to reduce the cost due to rediscoveries, to cost-effectively adopt solutions to

[1] Here we do not address rediscoveries caused due to the software providers' decision to release a software product with known defects.

reduce the cost due to rediscoveries. From a research point of view, the results of this investigation would add to the existing body of knowledge on the causes for rediscoveries in software systems.

1.3 Significance and Originality of the Study

Understanding the causes for rediscoveries and their significance is critical to reduce the cost due to rediscoveries. The insight gained from the study may prove invaluable in this case. To achieve this, the study is aimed at creating a list of causes for rediscoveries identified both from literature and industry practitioners. A software development organisation operates under critical resource constraints. The cost of any effort directed at a 'cause' must justify the benefits gained from the effort. Some causes may be more significant than others. Hence, it is not only important to identify the causes but also to determine the significance of each cause, which is one of the goals of the study.

In the limited work done on software rediscoveries, to our knowledge, there is no work which exclusively deals with causes for rediscoveries and their significance, which we have accomplished in our study.

1.4 Thesis Organization

The thesis is structured as follows: Chapter 2 presents a synthesis of related works from the research literature; Chapter 3 presents the literature analysis and research problem in detail; Chapter 4 presents the rediscovery cause taxonomy; Chapter 5 is where we address the significance of rediscovery causes, and present the methodology of our investigation and also the two case studies conducted as a part of this research; Chapter 6 summarizes the thesis by discussing the implications of the findings and conclusions from the study, and also suggests future work in this area.

Chapter 2: Background

There has been extensive research regarding software defect rediscoveries – the cost incurred due to the rediscoveries [Lee Iyer 2000] [Wood 2003] [Brodie 2005], the causes for rediscoveries [Lee McRee Bartlett 1996] [Baumann 2004] [Baumann 2005] [Altekar 2005] [Ballintijn 2005] [Jansen 2005] [Gkantsidis 2006] [Pasala 2006] [Crameri 2007] [Pasala 2008] and various approaches to reduce the cost due to the rediscoveries [Adams 1984] [Cobb 1992] [Lee Pitt Iyer 1996] [Lee Iyer 2000] [Brodie 2005] [Thornton Quema 2005] [Dungan 2004] [Beattie '2002]. All these approaches target one or more causes for rediscoveries identified in section 1.1. These approaches can be broadly classified into three categories based on the cause they target. Preventive maintenance policy deals with the delay in dispatching of the fix to users which can cause rediscoveries [Adams 1984]. Diagnosis technologies deal with the delay in the diagnosis of the failure to identify the defect [Cobb 1992] [Lee McRee Bartlett 1996] [Lee Iyer 2000] [Brodie 2005]. Patch risk evaluation approaches deal with the skepticism of users that the patch would cause more harm than good on installation [Thornton Quema 2005] [Dungan 2004] [Beattie 2002]. Finally, we deal with technologies which automate most of the tasks involved in patch installation, thus simplifying it. It is important to note here that a complex patch installation process is one of the reasons for users not frequently installing patches made available by the software provider, on their systems thus causing rediscoveries.

2.1 Preventive Maintenance Policy

The delay to dispatch the patch for a defect can add to the overall delay on part of the software provider in providing the patch for a defect thus causing rediscoveries at user sites. Some defects cause isolated rediscoveries among users whereas others are more widespread. Thus the cost of this delay is more in case of defects which cause widespread rediscoveries than in case of defects which cause isolated rediscoveries. Hence it is prudent to immediately dispatch the fixes for defects which are known to cause widespread rediscoveries among users. This policy was introduced and advocated by E.N. Adams [Adams 1984] and is known as preventive maintenance. E.N. Adams further concluded in his study that defects which surface during the initial days of

product release are the ones which cause widespread rediscoveries among users. Although these defects form a small percentage of total defects discovered throughout the product life cycle they account for relatively large percentage of rediscoveries [Mullen Gokhale 2005]. Hence, he advises that preventive maintenance policy makes most sense in case of defects identified in the initial stages of product release. However we can argue that this can be extended beyond the "version 1.0" release of the software product. It is a common knowledge that in the case of a commercial software product, features are continuously being added to the software product as it evolves. Hence each release of a new feature can itself be regarded as "version 1.0" release for that particular feature and expect that the defects related to that feature which surface during the initial days after this feature release will cause widespread rediscoveries among users who are using that particular feature. Hence, it is possible to extend the application of this policy throughout the life cycle of a software product to reduce the cost due to rediscoveries.

2.2 Diagnosis Technologies

Each failure has to be analyzed to identify the defect in software, which is termed as diagnosis [Lee Iyer 2000]. The delay in diagnosis can add to the overall delay on the part of the software provider to provide the fix for a defect. Cobb et al. addressed a systematic failure data collection method to support manual diagnosis performed by analysts which will hasten the failure resolution [Cobb 1992].

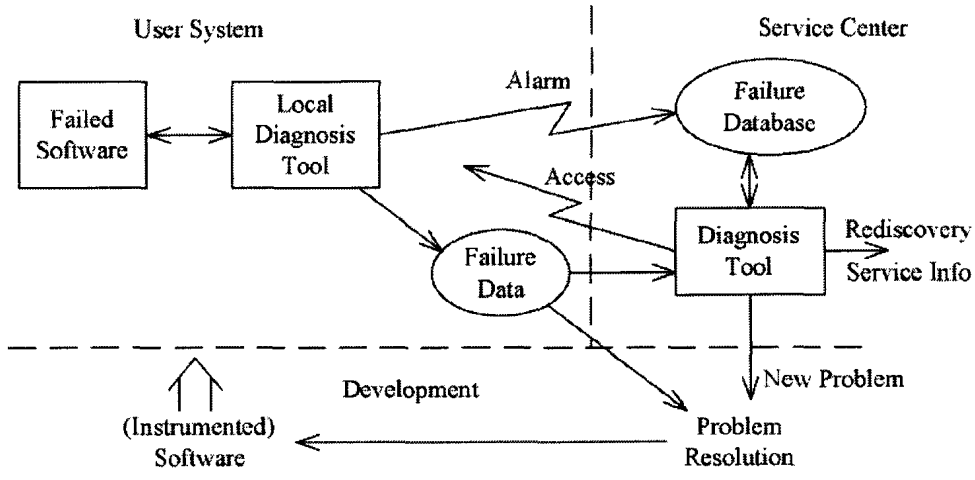


Figure 2 Diagnosing rediscovered software problems
[Lee Iyer 2000]

Diagnosis consumes many service resources when no automated help is available. In work by Lee and Iyer [Lee Iyer 2000], the researchers propose an approach to automatically determine whether a new failure reported is a rediscovery or not. The block diagram of the system designed by Lee and Iyer is shown in figure 2. At each user site, along with the actual software, a local diagnosis tool is installed. Whenever a failure occurs the local diagnostic tool collects various data regarding the failure and sends it online to the diagnostic tool in the service center. The diagnostic tool in the service center compares this data with the data of the previous failures to determine whether this new failure is due to an already known defect, in which case it is a rediscovery. Hence, the system after analysis can determine whether the current failure is a 'rediscovery' or a 'discovery'. In case of a rediscovery, the service information necessary to recover from the current failure and to prevent such failures in future is immediately dispatched to the user. And in the case of a discovery the development team is engaged to determine the defect in the software product which caused the failure and fix it. The patch containing the defect fix is later dispatched to the user. The work by Brodie et al. [Brodie 2005] proposes a solution similar to the one by Lee and Iyer [Lee Iyer 2000].

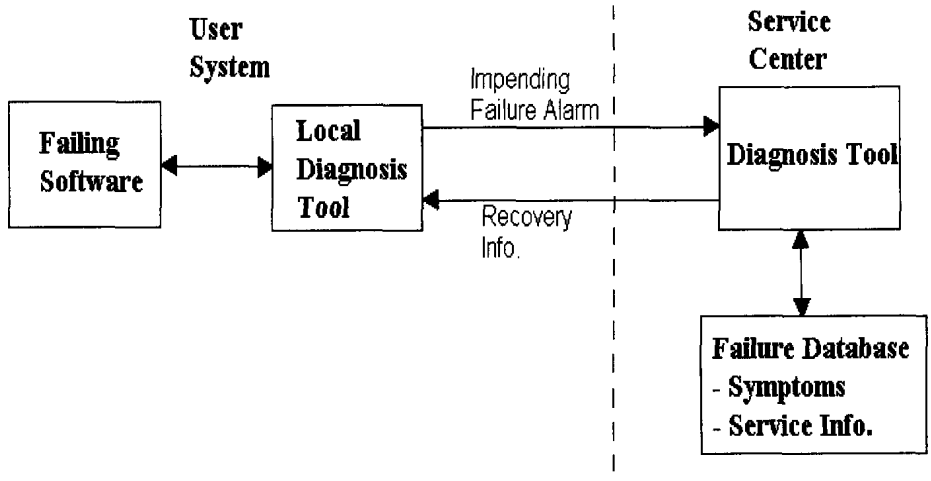


Figure 3 Diagnosis and recovery from rediscovered software problems [Lee McRee Bartlett 1996]

Lee, McRee and Bartlett [Lee McRee Bartlett 1996] propose an approach to recover from impending failures due to known defects. The idea is to use the knowledge of the characteristic symptoms of an already known defect and the appropriate recovery action for the defect to detect the impending failure and recover from it. The whole process is shown by the block diagram in figure 3. A local diagnosis tool, installed with the software at the user site, detects an impending failure through data collected during program execution. It sends the relevant data online to the diagnosis tool at the service center. The diagnosis tool at the service center compares this data with the data of historical failures in the failure database. If a suitable match is found, which means the impending failure is in fact an impending rediscovery, the relevant service information is extracted and sent online back to the local diagnosis tool at the user site. The local diagnosis tool interprets this service information and takes necessary steps to recover from the impending failure.

2.3 Patch Risk Evaluation

One of the causes for rediscoveries is the users' failure to install the patch in spite of it being made available by the software provider. One of the reasons for a user not to install the patch is the skepticism of the user that the patch will cause more 'harm' than 'good'. The 'harm' may come in the form of a breakage in functionality, degradation in performance, etc. The work by Thornton and Quema [Thornton Quema 2005] deals with the skepticism of users that patching their systems may cause more harm than good. They provide a decision support tool for the users to evaluate the risk due to patch installation on their systems. One common reason for this skepticism is the existence of third-party and in-house applications that are probably not tested with the patch by the software provider.

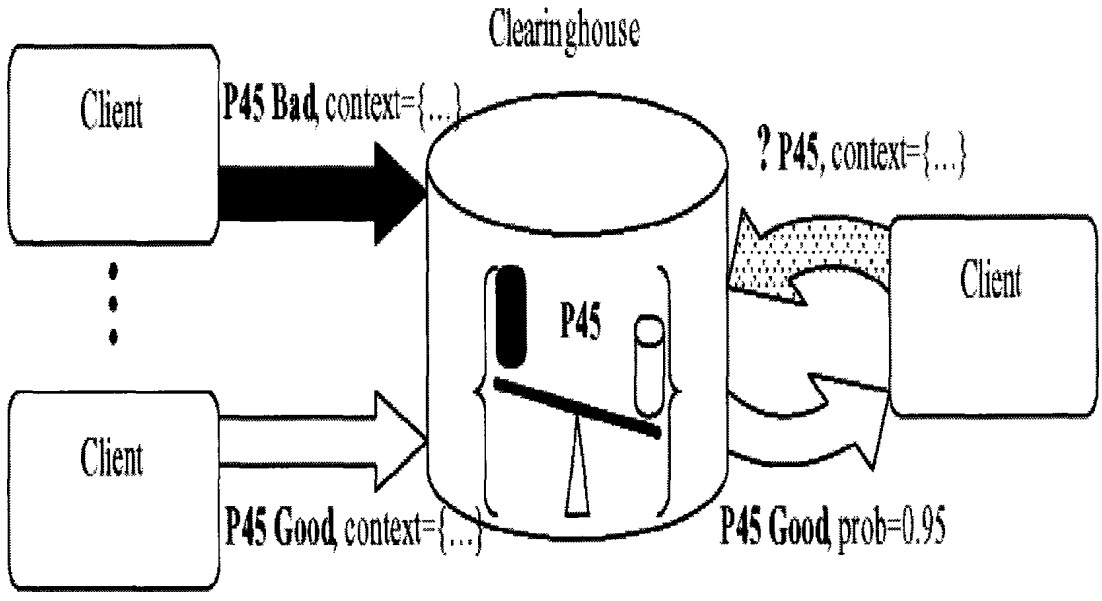
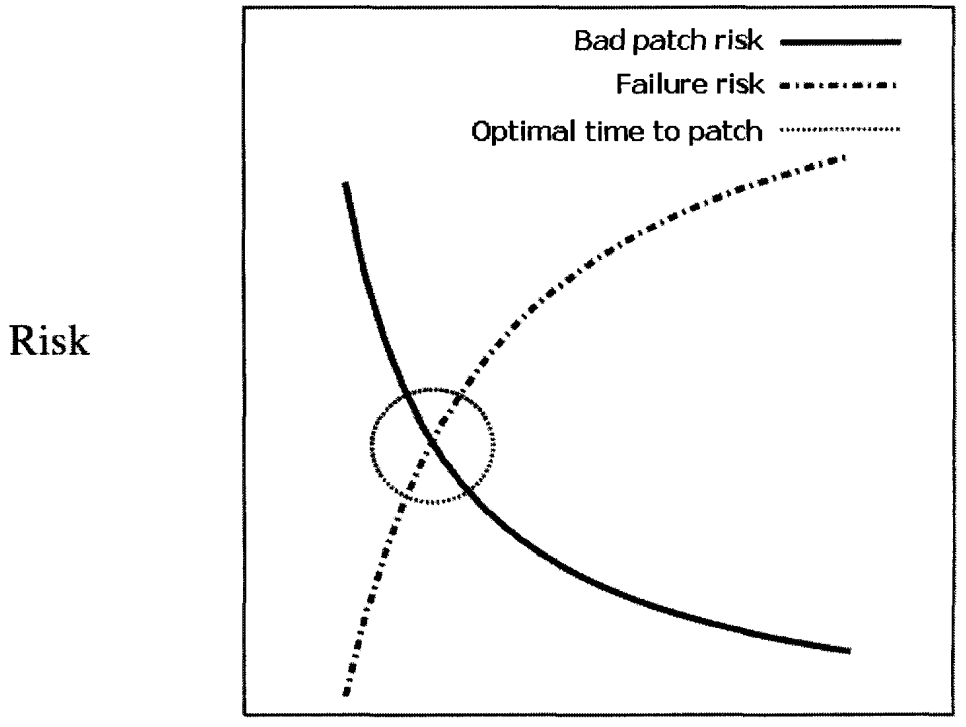


Figure 4 Patch Risk Evaluation
[Thornton Quema 2005]

Thornton and Quema [Thornton Quema 2005] advocate a system which has a central repository where users who have already installed the patch upload their experience, 'good' or 'bad', with respect to the patch and their context which is nothing but their configuration and interfacing applications. This activity is shown by the clients on the left side of the clearing house in figure 4. Any new user can now get a measure of the 'goodness' of the patch with respect to his context by querying the repository. This activity is shown by the client on the right side of clearing house in figure 4. This measure of 'goodness', which is shown as 'prob=0.95' for patch P45 as an example in figure 4, can be used for decision making related to patch installation.

The work by Beattie et al. also deals with the skepticism of users that patching their systems may cause more harm than good [Beattie 2002]. When a new patch is released not all users install the patch immediately as some are skeptical about the 'goodness' of the patch. Here the term 'goodness' means the ability of the patch to render the systems working normally without any issues related to functionality or performance. A patch is defined as 'good' if does not cause any issues with system functionality or performance after installation. Users who have IT staff resources will do a second round of testing

before installing the patch. Any information about encountered problems related to the patch will be shared with others, like the provider of the patch and in other user forums including those on the internet. Users who have not installed the patch and have no resources to test the patch themselves will make use of this information to decide whether to install the patch or not. Just ignoring the patch will increase the risk of hitting the relevant failure with time. However, the risk of a bad patch decreases with time as other people install the patch and share their experience. Hence, there is an optimal time after the release of the patch when systems can be patched. The study by Beattie et al. calls this the 'optimal time to patch' [Beattie 2002].



A hypothetical graph of risks of loss from failure and from application of bad patch. The optimal time to patch is the time where the risk lines cross

Figure 5 Optimal time to patch [Beattie 2002]

The work by Dungan et al. also deals with the skepticism of users that patching their systems may cause more bad effects than good effects [Dungan 2004]. The policy that Dungan et al. propose for estimating the likelihood that a patch will be destabilizing is

based on the following thesis: patches from one software provider that impact dynamic libraries or configuration files loaded by applications from another software provider are the least likely to have been previously tested together. The organization releasing the patch may be able to test some applications beforehand, but it is unlikely to be able to test all third-party applications affected by the patch. These patch-application combinations pose the greatest concern during patch deployment. The policy Dungan et al. propose for targeted rollout during the testing phase is to patch the minimum number of machines that still provides the required coverage of impacted applications.

The suite of components implemented by Dungan et al. provides the self-monitoring infrastructure to allow this targeted deployment. Even with this ability to target, an organization may still not have sufficient testing resources to thoroughly test patches before deployment. In this case, these same polices can also guide the deployment of patches automatically. If a phased rollout is desired, the patches can be incrementally rolled out to machines based on each machine's mix of running applications – the component suite provides sufficient visibility that an organization can avoid patching all of the machines performing one operation at once. If instead the desire is to classify some patches for immediate deployment, and to delay other patches for testing, those patches that do not directly affect libraries or configuration settings used by mission-critical third-party applications can be identified and chosen for immediate deployment.

2.4 Patch Management Tools

One of the causes for rediscoveries is the users' failure to install the patch in spite of it being made available by the software provider. One of the reasons for a user not to install the patch is the complexity of the patch installation process. State-of-the-art patch management tools automate many aspects of the patch installation process. These relieve the user from the complexity of the patch installation process. Commercial products such as Microsoft Baseline Security Analyzer [MBSA], Tivoli [Tiv], Microsoft Systems Management Server [SMS], or Corporate Windows Update [WU] offer the ability to automatically check for the availability of relevant patches, download and apply them. These tools are especially critical when patches need to be

installed over a network on a large number of systems. However, these solutions deal only with simplifying the patch installation process.

Chapter 3: Literature analysis and Research Problem definition

Each of the approaches described in chapter 2 is targeted towards one or more causes for rediscoveries. Preventive maintenance policy and diagnosis technologies are targeted at reducing the delay on the part of the software provider to provide the patch to the user. Patch risk evaluation technologies provide decision support to the users in their patch installation decisions by resolving their skepticism to some extent. Patch management tools simplify most of the aspects of the patching process and eliminate complexity from the process to encourage users to install patches whenever they are made available by the software provider.

However, none of these solutions individually can be a 'silver bullet' to eliminate most of the cost due to rediscoveries. Hence, we need an assortment of these solutions, and possibly new ones, to eliminate most of the cost due to rediscoveries. In such a situation it becomes increasingly important to understand the phenomenon of rediscoveries more deeply. That is, we need to understand not only the causes for rediscoveries but also any inter-relationships among them (i.e., *taxonomy*) and the *significance* of each cause. The taxonomy adds to the knowledge-base on rediscoveries. From a cost perspective, the significance of a cause is the proportion of the cost that particular cause is contributing to the total rediscovery cost. Knowing the significance of each cause could guide organisations to efficiently utilize the known solutions, or design new ones if necessary, to ultimately reduce software costs due to rediscoveries.

To our knowledge, there is no published literature on taxonomy for causes of defect rediscoveries or the relative significance of the various causes of rediscoveries. Thus, the focus of our research is to consolidate the knowledge regarding various causes for rediscoveries from different sources such as the literature and software practitioners, and empirically establish the significance of these causes. Given the fragmented body of knowledge on the topic of software rediscoveries, the findings from this study would add considerably to this knowledge base.

With the preceding as the backdrop, the overall research question is:

What are the causes of software rediscoveries, including the taxonomy describing the inter-relationships between causes, and what is the significance of each of these causes?

We split this overall question into two parts:

P1: *What are the causes of software rediscoveries, including the taxonomy describing the inter-relationships between causes?*

P2: What is the significance of each of these causes?

These two questions are dealt with, respectively, in Chapter 4 and Chapter 5.

Chapter 4: Rediscovery causes and their Inter-relationships

In this chapter, we present our research with respect to a part of our overall research question stated in chapter 3. The particular research question we are addressing in this chapter is as follows:

P1: What are the causes of software rediscoveries, including the taxonomy describing the inter-relationships between causes?

Based on background literature and practice assessment, we have designed a hierarchical classification of various causes for rediscoveries, as depicted in Figure 6. Table 1 describes each of the causes in this taxonomy. The hierarchy in Figure 6 is based on two fundamental branches - patch not available to the user (cause-1) and patch available to the user (cause-2) – because of the separation of concern between the software provider and the software user. These two causes are further decomposed into related sub-causes.

The motivation for this hierarchical classification was to establish relationships between causes to be able to clearly identify the stakeholders under whose scope each cause prevails. The stakeholders in this phenomenon of rediscoveries are the software provider and the software user. For example, cause-1 (patch not available to the user) and its sub-causes are clearly in the scope of software provider as the causes address the delay on the software provider's side to provide a patch to the software users. Similarly, cause-2.2 and its sub-causes are in the scope of the software user as the causes address various reasons for a software user not installing a patch. Another motivating factor was to identify the various causes at a granularity where they can be addressed by specific solutions - existing or new. For example, causes-1.1 deals with 'Delay in defect diagnosis'. The 'Diagnosis Technologies' mentioned in section 2.1 can be adopted by a software provider to address this cause, i.e. 'Delay in defect diagnosis'. Hence the two objectives for creating the rediscovery cause taxonomy were as follows:

T1: Establish relationships between causes to be able to clearly identify the stakeholders under whose scope each cause prevails.

T2: Identify the various causes at a granularity where they can be addressed by specific solutions - existing or new.

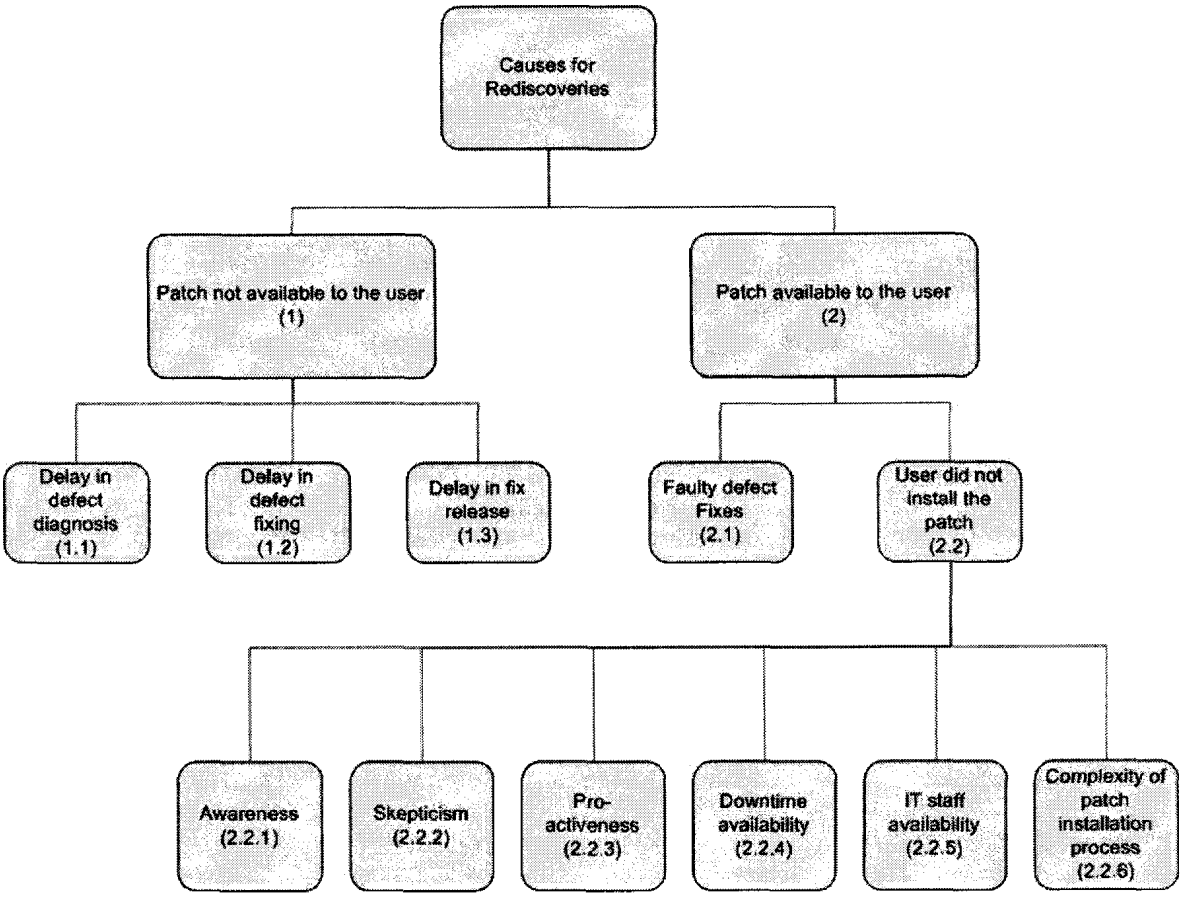


Figure 6 Rediscovery cause taxonomy

Cause	Explanation
1	Rediscoveries can occur where the patch for the defect is not available to the user. Here the cause is the delay on the part of the software provider to provide the relevant patch to the user [Lee Iyer 2000].
1.1	Rediscoveries can occur due to delay in defect diagnosis. Diagnosis is the resolution of the defect from the failure. If proper data is not collected at each failure, its resolution may become difficult and hence it may take more failures before the defect is finally identified [Lee Iyer 2000].
1.2	One of the delays on the software provider's side could be the delay in actually fixing the defect after its resolution. This may be due to a variety of reasons like inadequate number of developers, poor development process, poor developer quality - less experience, untrained, etc. [Lee Iyer 2000].
1.3	Software Providers normally release interim versions of the software periodically. These releases, called patches, contain fixes to the new defects which have been found in the software system. Hence there is a delay in time between the actual fixing of the defect by the developers and the time when the patch containing fix actually gets into the users' systems. This delay can be a cause of more rediscoveries [Lee Iyer 2000].
2	Rediscoveries can occur even when the software patch is provided by the software provider [Lee Iyer 2000].
2.1	Sometimes a purported fix can fail. This can occur due to improper diagnosis-development-testing-release process on the software provider's side. This can cause more rediscoveries in the users' sites. [Lee Iyer 2000]
2.2	Rediscoveries can occur when users fail to install the available software patch on their systems. The specific reasons are listed from 2.2.1 to 2.2.7 [Lee Iyer 2000].
2.2.1	**Awareness:** Some users may not know about the new defects discovered in the software product and the relevant patches. This lack of awareness may prevent them from installing the patches which will result in more rediscoveries [Lee Iyer 2000] [Gerace 2005].

Cause	Explanation
2.2.2	**Skepticism:** Some users are skeptical about patches that they can sometimes cause changes to system behavior – issues with functionality, performance or interfacing. This skepticism forces the users to test the patches before installation which involves expenditure of lot of resources in terms of people and time. The underlying delay results in more rediscoveries. Also, some users limit the number of patches installed on their systems as they want to keep the changes made to their systems minimal due to this skepticism. This failure to keep the systems up to date may cause more rediscoveries [Lee Iyer 2000] [Altekar 2005] [Ballintijn 2005] [Gerace 2005] [Jansen 2005] [Pasala 2006] [Crameri 2007] [Pasala 2008].
2.2.3	**Pro-activeness:** Some users don't prefer to install a software patch for a particular defect unless they themselves hit a failure due to that defect. This lack of pro-activeness results in more rediscoveries [Lee Iyer 2000] [Gkantsidis 2006].
2.2.4	**Downtime availability:** Software patch installation needs system downtime. However downtime is limited due to business requirements. This lack of adequate system downtime results in decrease in the frequency of patch installation on the systems. This failure to keep the systems up to date results in more rediscoveries [Lee Iyer 2000] [Baumann 2004] [Altekar 2005] [Baumann 2005] [Gerace 2005].
2.2.5	**IT staff availability:** Patch management involves a number of sub-processes like analyzing the risk to which the systems are exposed, testing the patches before installation, the actual patch installation on production systems, etc which necessitates considerable IT staff resource. Some users do not have adequate IT staff resource for this purpose, which forces them to limit the frequency of patch installation on their systems [Beattie 2002] [Gerace 2005].
2.2.6	**Complexity of patch installation process:** The patch installation process may include various sub-processes like taking backups and restoration which in the absence of proper tools can be quite tedious when the number of systems to be patched is large. This complexity of patch installation process may discourage the users from frequently installing patches on their systems. This failure to keep the systems up to date results in more rediscoveries [Wood 2003].

Table 1 Short Description about causes for rediscoveries

The taxonomy depicted in Figure 6 has been construct-validated by 6 experts - researchers and practitioners - involved in the contextual research project of which the present thesis forms a part.

Expert	Researcher / Practitioner	# years of Experience	Experience	Areas of Expertise
1	Researcher	30	Developer (small scale); Architect; Researcher; Research Lead; Consultant; Pedagogue;	Requirements Engineering, Empirical Studies, Software Quality, Software Maintenance, Software Process Engineering.
2	Practitioner	18	Developer (large scale); Project Manager; Architect; Consultant;	Operating Systems, Databases, Hardware Utilization (memory, cache, etc.) Networks, Software Quality, Serviceability.
3	Researcher and Practitioner	15	Developer; Quality Assurance Specialist; Consultant; Research Lead; Collaboration Manager; Team Lead;	Software Engineering, Software Quality Assurance, Software Maintenance, Project Estimation.
4	Practitioner	20	Quality Assurance Specialist; Project Manager;	Software Quality Assurance and Customer Service
5	Researcher and Practitioner	10	Developer (small to large scale); Architect; Project Manager; Researcher; Consultant ;	Software Quality, Software Maintenance
6	Practitioner	6	Quality Assurance Specialist;	Software Quality Assurance with focus of test methodologies development and implementation; Applied research

Table 2 Experts who participated in the validation of rediscovery cause taxonomy

Construct validity refers to the degree to which inferences can legitimately be made from the operationalizations in the study to the theoretical constructs on which those operationalizations were based [SRM]. The theoretical constructs here are nothing but the two objectives for creating the rediscovery cause taxonomy – T1 and T2, both of which were validated by the experts in Table 2 as being in accordance with our research goal P1. The two types of *construct validity* that were applied in this case were *content* and *face* validity [SRM].

Content validity is based on the extent to which the operationalization reflects the specific intended domain of content [Carmines 1991]. This was established in our study by examining key software conferences, journals and magazines for literature relevant to the topics of software rediscoveries and patch management. We found 43 research papers relevant to these topics out of which 13 were central to our work.

In *face validity,* "you look at the operationalization and see whether 'on its face' it seems like a good translation of the construct" [SRM]. *Face validity* is concerned "with how a measure or procedure appears. Does it seem like a reasonable way to gain the information the researchers are attempting to obtain? Does it seem well designed? Does it seem as though it will work reliably?" [CSU]. This is met in our study by involving six experts in reviewing the operationalization, both content and form. This was carried out in a series of 6 iterative discussions with the experts in Table 2. The coverage of literature was considered to be comprehensive and the analysis of literature to be acceptable by the experts. In the end, rediscovery cause taxonomy was considered as reasonable based on the criteria for the hierarchical classification, which are nothing but the theoretical constructs – T1 and T2.

Chapter 5: Significance of Rediscovery causes

In this chapter, we present our research with respect to the second part of the overall research question stated in chapter 3. The particular research question we are addressing in this chapter is as follows:

P2: What is the significance of each of the causes for software rediscoveries?

We first describe the research methodology used, followed by the case studies conducted using this methodology.

5.1 The Research Methodology

Starting with goal P2 above, it is important to derive specific objectives in a top-down fashion to facilitate quantitative interpretation. The Goal Question Metric Approach [Basili 1994] is a valuable tool which describes a top-down approach to divide goals into objectives. These objectives, defined in quantitative terms, will drive the investigative process. Based on the quantitative interpretation of the objectives we identify various sources of data necessary for the study. Once the various sources of data are identified, we define various methods for data acquisition followed by data analysis and interpretation.

5.1.1 The Goal Question Metric Approach

The Goal Question Metric (GQM) approach is based upon the assumption that for an organisation to measure in a purposeful way it must first specify the goals for itself and its projects, then it must trace those goals to the data that are intended to define those goals operationally, and finally provide a framework for interpreting the data with respect to the stated goals. It shows that it is important to make clear, at least in general terms, what information needs the organisation has, so that these needs for information can be quantified whenever possible, and the quantified information can be analysed as to whether or not the goals are achieved.

5.1.1.1 GQM Measurement Model

The GQM measurement model [Basili 1994] has three levels:

1. **Conceptual Level (GOAL):** A goal is defined for an object, for a variety of reasons, with respect to various models of quality, from various points of view, relative to a particular environment. Objects of measurement are
    - **Products:** Artefacts, deliverables and documents that are produced during the system life cycle; for example, requirements specification document, design document, source code and test suite.
    - **Processes:** Software related activities normally associated with time; for example, requirement specification, designing, coding and testing.
    - **Resources:** Items used by processes in order to produce their output; for example, personnel, hardware, software and office space.

2. **Operational Level (QUESTION):** A set of questions is used to characterize the way the assessment or achievement of a specific goal is going to be performed based on some characterizing model. Questions try to characterize the object of measurement (product, process, resource) with respect to a selected quality issue and to determine its quality from the selected viewpoint.

3. **Quantitative Level (METRIC):** A set of data is usually associated with every question in order to answer it in a quantitative way. That data can be
    - **Objective:** If they depend only on the object that is being measured and not on the viewpoint from which they are taken; for example, number of versions of a document, staff hours spent on a task, size of a program.
    - **Subjective:** If they depend on both the object that is being measured and the viewpoint from which they are taken; for example, readability of text, level of user satisfaction.

5.1.2 Formulation of Research Objectives at the GQM Operational Level

The very first step with GQM is to redefine our research goals in a more quantitative manner. To establish the significance of various causes for rediscoveries, we need to

first define the term 'significance' in quantitative terms. The definition for 'significance' of a cause for rediscoveries should be a measure of the proportion of the cost contributed by that particular cause to the total cost incurred due to all rediscoveries. Also, it is important to define the term in such a way that it can be reduced to an appropriate metric, quantitative or qualitative, which can be determined from an accessible source of data. The definition which we adopt is as follows:

Significance of a 'cause' for rediscoveries: The significance of a 'cause' for rediscoveries is defined as the percentage of rediscoveries caused by that particular 'cause'.

It is important to note here that the above definition does not take into consideration that the cost of each rediscovery may not be the same. However, data which is necessary to find the cost of each rediscovery was not available in our case. That is one of the subjects of a follow-up work to this one. Hence, we base the above definition of 'significance' on the assumption that cost of each rediscovery is the same.

Using this definition for significance of a cause for rediscoveries and the rediscovery taxonomy presented in Figure 6, we describe the research objectives in the format prescribed by the GQM measurement model as in Table 3 to 13:

Goal	**Purpose**	Determine
	Issue	the significance of cause-1.1 (Delay in defect diagnosis) with respect to
	Object (Process)	Occurrence of software defect rediscoveries
	Viewpoint	Based on our definition of 'Significance' (section 5.1.2)
Question	**Q1**	What is the percentage of rediscoveries which occur due to cause-1.1(Delay in defect diagnosis)?
Metrics	**M1**	(Number of rediscoveries which occur before defect is diagnosed) * 100 / (Total number of rediscoveries)

Table 3 GQM Research Objective 1

Goal	**Purpose**	Determine
	Issue	the significance of cause-1.2 (Delay in defect fixing) with respect to
	Object (Process)	occurrence of software defect rediscoveries
	Viewpoint	Based on our definition of 'Significance' (section 5.1.2)
Question	**Q2**	What is the percentage of rediscoveries which occur due to cause-1.2 (Delay in defect fixing)?
Metrics	**M2**	(Number of rediscoveries which occur after the defect is diagnosed but before the defect is fixed by making necessary source code changes) * 100 / (Total number of rediscoveries)

Table 4 GQM Research Objective 2

Goal	**Purpose**	Determine
	Issue	the significance of cause-1.3 (Delay in fix release) with respect to
	Object (Process)	occurrence of software defect rediscoveries
	Viewpoint	Based on our definition of 'Significance' (section 5.1.2)
Question	**Q3**	What is the percentage of rediscoveries which occur due to cause-1.3 (Delay in fix release)?
Metrics	**M3**	(Number of rediscoveries which occur after the defect is fixed by making the necessary source code changes but before the fix is bundled into patches and released to users) * 100 / (Total number of rediscoveries)

Table 5 GQM Research Objective 3

Goal	Purpose	Determine
	Issue	the significance of cause-2.1 (Defective fixes) with respect to
	Object (Process)	occurrence of software defect rediscoveries
	Viewpoint	Based on our definition of 'Significance' (section 5.1.2)
Question	**Q4**	What is the percentage of rediscoveries which occur due to cause-2.1 (Defective fixes)?
Metrics	**M4**	(Number of rediscoveries which occur after the defect fix is released in a patch and the user has installed the patch) * 100 / (Total number of rediscoveries)

Table 6 GQM Research Objective 4

Goal	Purpose	Determine
	Issue	the significance of cause-2.2 (User did not install the patch) with respect to
	Object (Process)	occurrence of software defect rediscoveries
	Viewpoint	Based on our definition of 'Significance' (section 5.1.2)
Question	**Q5**	What is the percentage of rediscoveries which occur due to cause-2.2 (User did not install the patch)?
Metrics	**M5**	(Number of rediscoveries which occur after the defect fix is released in a patch and the user has not installed the patch) * 100 / (Total number of rediscoveries)

Table 7 GQM Research Objective 5

Metrics M1 to M5 can be determined by using the defect and failure data collected in issue tracking systems by the software provider. Such defect and failure data would include various attributes like date of occurrence of the failure or defect, date on which the diagnosed from the failure, date on which the defect was fixed, date on which fix for the defect was bundled into a patch and released, etc which can be used to determine metrics M1 to M5. However to determine the significance of causes from cause-2.2.1 to cause-2.2.6 the defect and failure data does not prove to be sufficient. We figured out in our situation that the only way to go after these causes is to ask the users directly whether a particular cause was relevant to them when it came to delaying or cancelling

a patch installation, which leads to the occurrence of rediscoveries. For cause-2.2.1, we do this by asking the software users (i.e., system administrators who handle patch installations) to rate the cause on a 7-point Likert scale of agreement with respect to their software providers' efforts to keep the software users (i.e., system administrators who handle patch installations) updated with information regarding new defects and the relevant patches. The Likert scale is a psychometric scale commonly used in questionnaires [Wiki]. For causes from cause-2.2.2 to cause-2.2.6, we try to establish the significance of the causes by asking the software users (i.e., system administrators who handle patch installations) to rate the causes from cause-2.2.2 to cause-2.2.6 on a 7-point Likert scale of importance with respect to their decision to delay or cancel a patch installation due to that particular cause. A questionnaire has been designed for this purpose. Again, GQM approach, Table 8 to 13, is used to design the questionnaire.

Goal	Purpose	Determine
	Issue	the significance of cause-2.2.1 (Awareness) with respect to
	Object (Process)	the occurrence of rediscoveries
	Viewpoint	from the software users' view point
Question	**Q6**	Please indicate your agreement with the following statement. In general, your organization's software vendors provide you with timely notices about new defects and patch releases. 1 – Strongly Disagree 2 – Disagree 3 – Somewhat Disagree 4 – Neutral 5 – Somewhat Agree 6 – Agree 7 – Strongly Agree
Metrics	**M6**	Average rating on the 7-point Likert scale
Question	**Q7**	Please indicate your agreement with the following statement. In general, your organization's software vendors provide you with relevant information regarding new defects to help you analyze whether your systems are under risk. 1 – Strongly Disagree 2 – Disagree 3 – Somewhat Disagree 4 – Neutral 5 – Somewhat Agree 6 – Agree 7 – Strongly Agree
Metrics	**M7**	Average rating on the 7-point Likert scale

Table 8 GQM Research Objective 6

Goal	**Purpose**	Determine
	Issue	the significance of cause-2.2.2 (Skepticism) with respect to
	Object (Process)	the occurrence of rediscoveries
	Viewpoint	from the software users' view point
Question	**Q8**	If there was a delay to install a patch from your software vendor in the last 24 months, how important was the following factor in your organization's IT department's decision to delay patch installation? There was a delay in testing the patch before installation on production systems. 1 – Not Important 2 – Slightly Important 3 – Somewhat Important 4 – Moderately Important 5 – Important 6 – Very Important 7 – Extremely Important
Metrics	**M8**	Average rating on the 7-point Likert scale
Question	**Q9**	What was the average delay in patch installation on production systems at your organization in the last 24 months due to the following factor? Testing the patch before installation. 1 – Less than 1 hour 2 – More than 1 hour but less than 1 day 3 – More than 1 day but less than 1 week 4 – More than 1 week but less than 1 month 5 – More than 1 month but less than 3 months 6 – More than 3 months but less than 6 months 7 – More than 6 months
Metrics	**M9**	Statistical mode of all the responses

Question	**Q10**	If there was a decision to cancel the installation of a patch from your software vendor in the last 24 months, how important was the following factor in your organization's IT department's decision to cancel the patch installation? We keep the changes made to our systems minimal. 1 – Not Important 2 – Slightly Important 3 – Somewhat Important 4 – Moderately Important 5 – Important 6 – Very Important 7 – Extremely Important
Metrics	**M10**	Average rating on the 7-point Likert scale

Table 9 GQM Research Objective 7

Goal	**Purpose**	Determine
	Issue	the significance of cause-2.2.3 (Pro-activeness) with respect to
	Object (Process)	the occurrence of rediscoveries
	Viewpoint	from the software users' view point
Question	**Q11**	Please indicate your agreement with the following statement. In practice, your organization would install a software patch even if you have not experienced a defect, which the patch is purported to fix. 1 – Strongly Disagree 2 – Disagree 3 – Somewhat Disagree 4 – Neutral 5 – Somewhat Agree 6 – Agree 7 – Strongly Agree
Metrics	**M11**	Average rating on the 7-point Likert scale
Question	**Q12**	If there was a delay to install a patch from your software vendor in the last 24 months, how important was the following factor in your organization's IT department's decision to delay patch installation? The systems were functioning normally and we had not experienced any defect which the patch was known to fix. 1 – Not Important 2 – Slightly Important 3 – Somewhat Important 4 – Moderately Important 5 – Important 6 – Very Important 7 – Extremely Important
Metrics	**M12**	Average rating on the 7-point Likert scale

Question	**Q13**	If there was a decision to cancel the installation of a patch from your software vendor in the last 24 months, how important was the following factor in your organization's IT department's decision to cancel the patch installation? The systems were functioning normally and we had not experienced any defect that the cancelled patch was known to fix. 1 – Not Important 2 – Slightly Important 3 – Somewhat Important 4 – Moderately Important 5 – Important 6 – Very Important 7 – Extremely Important
Metrics	**M13**	Average rating on the 7-point Likert scale

Table 10 GQM Research Objective 8

Goal	**Purpose**	Determine
	Issue	the significance of cause-2.2.4 (Downtime Availability) with respect to
	Object (Process)	the occurrence of rediscoveries
	Viewpoint	from the software users' view point
Question	**Q14**	If there was a delay to install a patch from your software vendor in the last 24 months, how important was the following factor in your organization's IT department's decision to delay patch installation? There was a delay due to lack of available system downtime. 1 – Not Important 2 – Slightly Important 3 – Somewhat Important 4 – Moderately Important 5 – Important 6 – Very Important 7 – Extremely Important
Metrics	**M14**	Average rating on the 7-point Likert scale
Question	**Q15**	What was the average delay in patch installation on production systems at your organization in the last 24 months due to the following factor? Lack of available system downtime 1 – Less than 1 hour 2 – More than 1 hour but less than 1 day 3 – More than 1 day but less than 1 week 4 – More than 1 week but less than 1 month 5 – More than 1 month but less than 3 months 6 – More than 3 months but less than 6 months 7 – More than 6 months
Metrics	**M15**	Statistical mode of all the responses

Question	**Q16**	If there was a decision to cancel the installation of a patch from your software vendor in the last 24 months, how important was the following factor in your organization's IT department's decision to cancel the patch installation? The lack of available system downtime limited the number of patch installations. 1 – Not Important 2 – Slightly Important 3 – Somewhat Important 4 – Moderately Important 5 – Important 6 – Very Important 7 – Extremely Important
Metrics	**M16**	Average rating on the 7-point Likert scale

Table 11 GQM Research Objective 9

Goal	**Purpose**	Determine
	Issue	the significance of cause-2.2.5 (IT staff availability) with respect to
	Object (Process)	the occurrence of rediscoveries
	Viewpoint	from the software users' view point
Question	**Q17**	Please indicate your agreement with the following statement. Your organization has adequate IT staff resource to timely address all potential patch installations. 1 – Strongly Disagree 2 – Disagree 3 – Somewhat Disagree 4 – Neutral 5 – Somewhat Agree 6 – Agree 7 – Strongly Agree
Metrics	**M17**	Average rating on the 7-point Likert scale
Question	**Q18**	If there was a delay to install a patch from your software vendor in the last 24 months, how important was the following factor in your organization's IT department's decision to delay patch installation? There was a delay due to lack of available IT staff to handle the patch installation process. 1 – Not Important 2 – Slightly Important 3 – Somewhat Important 4 – Moderately Important 5 – Important 6 – Very Important 7 – Extremely Important
Metrics	**M18**	Average rating on the 7-point Likert scale

Question	**Q19**	What was the average delay in patch installation on production systems at your organization in the last 24 months due to the following factor? Lack of available IT staff to install the patch. 1 – Less than 1 hour 2 – More than 1 hour but less than 1 day 3 – More than 1 day but less than 1 week 4 – More than 1 week but less than 1 month 5 – More than 1 month but less than 3 months 6 – More than 3 months but less than 6 months 7 – More than 6 months
Metrics	**M19**	Statistical mode of all the responses
Question	**Q20**	If there was a decision to cancel the installation of a patch from your software vendor in the last 24 months, how important was the following factor in your organization's IT department's decision to cancel the patch installation? The lack of available IT staff limited the number of patch installations. 1 – Not Important 2 – Slightly Important 3 – Somewhat Important 4 – Moderately Important 5 – Important 6 – Very Important 7 – Extremely Important
Metrics	**M20**	Average rating on the 7-point Likert scale

Table 12 GQM Research Objective 10

Goal	**Purpose**	Determine
	Issue	the significance of cause-2.2.6 (Complexity of the patch installation process) with respect to
	Object (Process)	the occurrence of rediscoveries
	Viewpoint	From the software users' view point
Question	**Q21**	Please indicate your agreement with the following statement. Your organization has adequate automation to make patch installation a straightforward task. 1 – Strongly Disagree 2 – Disagree 3 – Somewhat Disagree 4 – Neutral 5 – Somewhat Agree 6 – Agree 7 – Strongly Agree
Metrics	**M21**	Average rating on the 7-point Likert scale
Question	**Q22**	If there was a delay to install a patch from your software vendor in the last 24 months, how important was the following factor in your organization's IT department's decision to delay patch installation? There was a delay due to lack of adequate automation or non-usage of patch management tools, to handle the patch installation process. 1 – Not Important 2 – Slightly Important 3 – Somewhat Important 4 – Moderately Important 5 – Important 6 – Very Important 7 – Extremely Important
Metrics	**M22**	Average rating on the 7-point Likert scale

Question	**Q23**	What was the average delay in patch installation on production systems at your organization in the last 24 months due to the following factor? Lack of adequate automation of the patch installation process. 1 – Less than 1 hour 2 – More than 1 hour but less than 1 day 3 – More than 1 day but less than 1 week 4 – More than 1 week but less than 1 month 5 – More than 1 month but less than 3 months 6 – More than 3 months but less than 6 months 7 – More than 6 months
Metrics	**M23**	Statistical mode of all the responses
Question	**Q24**	If there was a decision to cancel the installation of a patch from your software vendor in the last 24 months, how important was the following factor in your organization's IT department's decision to cancel the patch installation? The lack of adequate automation of the patch installation process or non-usage of patch management tools, limited the number of patch installations. 1 – Not Important 2 – Slightly Important 3 – Somewhat Important 4 – Moderately Important 5 – Important 6 – Very Important 7 – Extremely Important
Metrics	**M24**	Average rating on the 7-point Likert scale

Table 13 GQM Research Objective 11

5.1.3 Motivation for choosing the Case Study approach as the research method

Further to designing the questionnaire using the GQM paradigm, we conducted two case studies on defect rediscoveries – to identify the significance of causes. Our motivation for choosing the case study approach as our research method is derived from the factors shown in Table 14 [Yin 2003]. Our research objectives span across two different stakeholders involved in software quality: the software provider and the software user. The research objectives implied by questions Q1 to Q5 relate to the

software provider; whereas, the research objectives implied by Q6 to Q24 relate to the software user.

Strategy	Form of Research Question	Requires Control of Behavioural Events	Focuses on Contemporary Events
Experiment	How, why?	Yes	Yes
Survey	Who, what where, how many, how much?	No	Yes
Archival analysis	Who, what, where, how many, how much?	No	Yes / No
History	How, why?	No	No
Case study	How, why?	No	Yes

Table 14 Relevant Situations for Different Research Strategies [Yin 2003]

For research objectives implied by questions Q1 to Q5, the root question is 'how much' (percentage of rediscoveries). This, 'how much' factor, filters our available strategies shown in Table 14 to 'Survey' and 'Archival analysis'. Also, each of these metrics, M1 to M5, does not require any control of behavioural events. Hence, our current list of available strategies remains unchanged. Although the metrics, M1 to M5, focus on both contemporary and historical events, since the historical data is readily available, it is prudent to utilize the historical data rather than using the data from contemporary events. This makes 'Archival analysis' our only option. However, it is the source of this data needed for archival analysis which forces us to rethink the term we use for this research method. The archival data belongs to a single software product. Hence, going by the classical definition of a Case study, an empirical inquiry that investigates a phenomenon within its real-life context [Yin 2003], we would like to term the research method a 'Case study' at the highest level. An 'Archival analysis' within a 'Case study', would be a detailed and more accurate description of the research method used in our study of research objectives implied by questions Q1 to Q5.

For research objectives implied by questions Q6 to Q24 the basic question is: Why do software users (i.e., system administrators who handle patch installations) not install a software patch whenever it is made available by the software provider? Hence the root

question here is 'why'. Also, it is not necessary to control the behavioural events to answer the question. Finally, the question 'does' focus on contemporary events as software installation and upgradation is a on-going activity for software users (i.e., system administrators who handle patch installations). Therefore, we conclude that a case study would be the appropriate research method to achieve our research objectives Q6 to Q24.

5.2 The Case Studies

We have undertaken two different case studies, one on a software provider and the other on a software user. The first case study is on the software provider of a commercial software system with many millions of lines of code, multiple versions and thousands of users. The second case study is on a medium sized [SME] (50-200 employees) software service provider which provides system administration services to various other organisations who actually use the software systems.

5.2.1 The Case Study – Software Provider

We start with the description of the components of research design followed by an analysis of the quality of research design. We then describe the customer technical support process (part of software maintenance) in general followed by a description of the various bookkeeping activities which take place during this process. In the end, we describe the formulation of quantitative level GQM research objectives, using various metrics involving the bookkeeping data from customer technical support process, from the operational level GQM research objectives.

5.2.1.1 Components of Research Design

For case studies, the following five components (i.e., Questions, Propositions, Unit of analysis, Logic linking data to propositions, and Criteria for interpreting the findings) of a research design are especially important [Yin 2003].

5.2.1.1.1 Questions

1. What is the significance of cause-1.1 (see Figure 6; Delay in defect diagnosis) with respect to occurrence of software defect rediscoveries?
2. What is the significance of cause-1.2 (see Figure 6; Delay in defect fixing) with respect to occurrence of software defect rediscoveries?
3. What is the significance of cause-1.3 (see Figure 6; Delay in fix release) with respect to occurrence of software defect rediscoveries?
4. What is the significance of cause-1.4 (see Figure 6; Faulty defect fixes) with respect to occurrence of software defect rediscoveries?
5. What is the significance of cause-1.5 (see Figure 6; User did not install the patch) with respect to occurrence of software defect rediscoveries?

5.2.1.1.2 Propositions

A proposition (i.e., hypothesis) directs attention to something that should be examined within the scope of the study. For example, let's consider the research question: How and why do organisations collaborate with one another to provide joint services (for example, a manufacturer and a retail store collaborating to sell certain computer products)? A prospective proposition for a case study involving the above research question would be: Organisations collaborate because they derive mutual benefits. This proposition, besides reflecting an important theoretical issue, also begins to indicate where one should look for relevant evidence, i.e., evidence for collaboration and benefits from that collaboration [Yin 2003].

However an exploratory study, where we do not begin the study with a theory but instead conduct the study to develop a theory which may be tested by another study, has a legitimate reason for not having any propositions [Yin 2003]. That said, an exploratory study should have some purpose as structured by the GQM statements. Therefore, instead of propositions, the design of an exploratory study should state this purpose, as well as the criteria by which an exploration would be judged successful. It happens so that by our research questions mentioned in section 5.2.1.1.1, our study is an

exploratory study. This is because we are not doing this study to test any theory but in turn to develop a theory regarding the 'significance' of various causes for rediscoveries. The research objectives implied by questions Q1 to Q5 from section 5.1.2 will serve as the purpose of this case study and the finding of valid answers to these questions would be the criteria for successful completion of the study.

5.2.1.1.3 Unit of Analysis

In order to investigate our research questions we need data regarding failures and defects of software products. Based on the feasibility in our current situation the best we can do is a single case study. Similar multiple case studies can follow this investigation for further proof. Hence, for the current study it is imperative that we select a typical case which is fairly representative of the population of software products based on the context of our research questions. We believe that in the context of our research questions, the case which is accessible to us is a typical case in the sense that the software development and service process of the software product is in agreement with the process shown in Figure 1. However software products can be characterized based on variety of factors such as whether the software product is commercial or not, or the number of lines of source code of the software product, or the number of versions of the software product till date, or the number of users of the software product, etc. Although we believe that these factors have no influence in the context of our research questions, as we understand it, we would like to state the characteristics of our case in the research design to assist other researchers and practitioners to derive conclusions about the results based on their understanding regarding these factors. The case under consideration for the study is a commercial software product, with millions of lines of code, multiple versions and thousands of users. To be specific, the data used in the study to investigate the research questions comprises of all the failures which occurred over a span of 4 years. The number of versions of the software product involved in the study is 15.

5.2.1.1.4 Logic linking data to propositions

This component of the research design is one of the least developed in case studies. One promising approach for case studies is the idea of "pattern matching" described by Donald Campbell [Campbell 1975]. An example of a "pattern" can be a graphical depiction of the data. Based on the propositions and the possible outcomes of the study, several prospective patterns of data are prepared. Each possible outcome is associated with a pattern. When the data is collected and its pattern determined, this resultant pattern is compared with all the prospective patterns. The outcome of the study is then determined by the outcome associated with the prospective pattern which most closely matches the resultant pattern. However, the exploratory nature of our study leaves us with no propositions and hence makes it impossible for us to use this approach. Hence, the purpose of the case study as mentioned in section 5.2.1.1.2 will be the sole guiding criteria in this regard.

5.2.1.1.5 Criteria for interpreting the findings

This component of the research design is one of the least developed in case studies. This component is important mostly in case studies with pre-defined theoretical propositions. In the case of "pattern matching" technique by Donald Campbell [Campbell 1975], which is described in section 5.2.1.1.4, this particular component of case study design would deal with the question of 'How close should the resultant pattern be to a prospective pattern in order to be considered a match' [Yin 2003]. However, the exploratory nature of our study does not provide us with any propositions. Hence, this component has no relevance in our case. Hence, the purpose of the case study as mentioned in section 5.2.1.1.2 will be the sole guiding criteria in this regard.

5.2.1.2 Quality of Research Design

The quality of empirical research needs to be established by four tests described below.

5.2.1.2.1 Construct Validity

Construct validity deals with establishing correct operational measures for the concepts being studied. In our study, we have concretely defined how the concepts under study

are formulated into research objectives using the GQM method in section 5.1.1. These research objectives have been further elaborated at the quantitative level in the section 5.2.1.6. By applying the GQM method we are ensuring that the operational measures used to analyze data are in tune with the high level goals of our research.

The construct validation of the operationalization was critical to the overall validity of the study. The two types of *construct validity* that applied to this operationalization were *content* and *face* validity [SRM]. These types of validity cannot be measured in a quantitative way, but we believe that we have met these through the extensive design procedures described.

Content validity is based on the extent to which the operationalization reflects the specific intended domain of content [Carmines 1991]. This was established in our study by covering all the "causes for rediscoveries" (causes – 1.1, 1.2, 1.3, 2.1 and 2.2) from Figure 6, which was in turn construct validated, using both content and face validity, by six experts in listed in Table 2.

Face validity is concerned "with how a measure or procedure appears. Does it seem like a reasonable way to gain the information the researchers are attempting to obtain? Does it seem well designed? Does it seem as though it will work reliably?" [CSU]. This is met in our study by involving six experts, listed in Table 2, in reviewing the operationalization, both content and form. The experts found the metrics M1 to M5, to be the correct operationalization of the research objectives implied by questions Q1 to Q5 respectively (see section 5.1.2 for questions Q1 to Q5 and metrics M1 to M5).

5.2.1.2.2 Internal Validity

Internal Validity is defined by Cooke and Campbell as the "approximate validity with which we infer that a relationship between two variables is causal" [Cooke 1979]. Internal validity is only a concern for causal case studies, in which an investigator is trying to determine whether event 'x' led to event 'y'. If the investigator incorrectly concludes that there is a causal relationship between 'x' and 'y' without knowing that

some third factor 'z' may actually have caused 'y', the research design has failed to deal with some threat to internal validity. However, this validity is not a concern in case of exploratory studies, which happens to be the type of study we are pursuing.

5.2.1.2.3 External Validity

External validity deals with the problem of knowing whether a study's findings are generalizable beyond the immediate case study. In this study, as mentioned in section 5.2.1.1.3, the case is 'typical' and fairly representative of the population of software products in the context of our research questions. However, as mentioned in section 5.2.1.1.3, there is a vast diversity of available software products. Hence, we describe the characteristics of our case suitably in section 5.2.1.1.3 to be able to generalize the findings of the study to software products with similar characteristics in the worst case.

5.2.1.2.4 Reliability

The objective of this test is to be sure that if a later investigator followed the same procedures as described by an earlier investigator and conducted the same case study all over again, the later investigator should arrive at the same findings and conclusions. The goal of reliability is to minimize the errors and biases in a study. In this study, we have properly documented our research process using GQM method from top to bottom, from research goals to research objectives, questions and metrics at the quantitative level. There is little or no room for errors and bias with respect to the metrics at the quantitative level. Hence, we conclude that our research design is reliable.

5.2.1.2.5 Conclusion Validity

Conclusion validity in the degree to which conclusions we make based on the findings are reasonable. We discuss the conclusions in section 6, and there we demonstrate that all our conclusions are rooted in the results, thereby maintaining conclusion validity.

5.2.1.3 The Customer Technical Support Process

The Customer Technical Support is an integral part of the software product life cycle. Today, factors such as shorter product life cycles and faster time to market present major challenges when it comes to customer satisfaction. Hence, an effective customer technical support process forms an important component of the software product package.

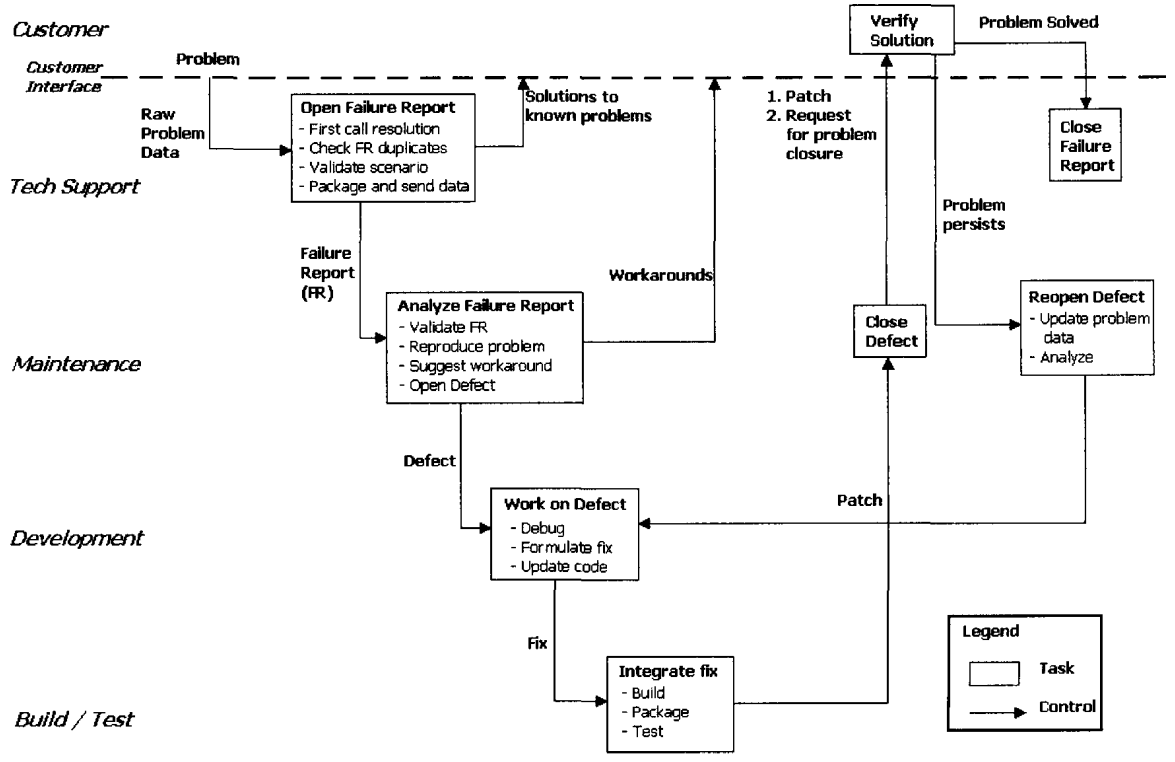


Figure 7 The Customer Technical Support Process

The customer, on experiencing a problem, calls the technical support of the software provider. The technical support team addresses the call. If the problem is due to a known one then they will provide the solution to the customer immediately. If the problem appears to be a new one then they validate the scenario under which the problem occurred by checking issues such as improper configuration of the software product, invalid use case, etc. Once the scenario is validated, all relevant data regarding the problem is collected and a Failure Report (FR) is opened.

The maintenance team validate the FR by subjecting it to defect diagnosis to verify that the failure is not due to a known defect. They will then try and reproduce the problem. If there is any possibility of a workaround then it is conveyed to the customer through the technical support team. A defect record is created and all relevant data and analysis is updated into the issue tracking system.

The development team analyzes the defect, designs the fix and updates the source code of the software product. The testing team then integrates the fix by building the executable from source code, and then packages it to produce a patch. The patch is then installed on the test machines and tested. The tested patch is provided to the maintenance team along with the test report. The maintenance team validates the test report to close the defect. It then provides the patch to the customer through the technical support team.

The technical support team provided the patch to the customer and requests for closure of the FR. If the customer finds the patch to be appropriate then he asks the technical support to close the FR. If the customer's problem continues to persist then the technical support team asks the maintenance team to reopen the defect and updates it with the latest data regarding the problem.

5.2.1.4 Bookkeeping across the Customer Technical Support Process

The Technical Support process involves various bookkeeping activities. Data regarding each failure and defect is recorded for managing the entire technical support process. The data model for the bookkeeping data regarding user failures and defects can be summarized by the Entity-Relationship Diagram below (see Figure 8).

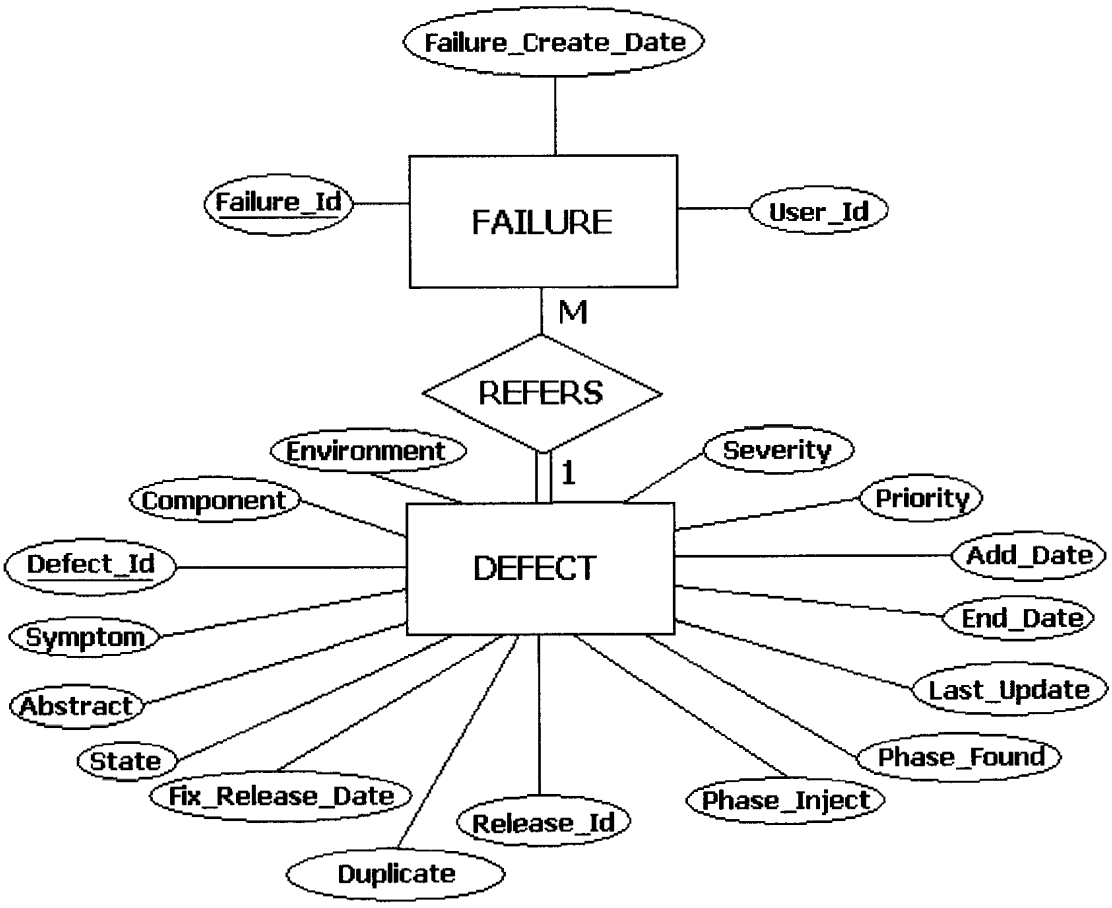


Figure 8 Entity Relationship Diagram of bookkeeping data in the technical support process

Attribute	Definition
Failure_Id	Unique identifier for the failure
Failure_Create_Date	Date on which failure was reported
User_Id	Identifier of the user who reported the Failure
Defect_Id	Unique identifier of the defect
Duplicate	Defect_Id of a duplicate defect
Abstract	A short description of the defect
Severity	The estimated impact of the reported problem. Attribute values can be: 1-Critical, 2-Severe, 3-Moderate, 4-Minimal
Priority	A parameter indicating 'the right to take precedence' among defects when it comes to fix development.
Release_Id	The release against which the problem is reported
Component	Name of the component of the software product where the defect exists.
Phase_Found	The phase of the development cycle where the defect was discovered
Phase_Inject	The phase of the development cycle where the defect was injected
Environment	The platform or environment under which the defect was observed
State	An indication of the progress made with respect to the defect. Attribute values can be: Open, Working, Verify, Cancel, Returned, Closed
Add_Date	Date on which the defect was created
End_Date	Date on which the defect was closed
Last_Update	Date on which the last update on the defect was made
Symptom	A description of the manifestation of the defect. (Eg., application crash, application hang)
Fix_Release_Date	Date on which the defect fix was released in a patch

Table 15 Attribute descriptions of the ER diagram (Figure 8) of the bookkeeping activities during customer technical support process

5.2.1.5 Dataset profile

The failure and defect data, whose schema is described in section 5.2.1.4, belongs to a commercial software product with millions of lines of code, multiple versions and thousands of users. The data used for the study comprises of failures that occurred over a period of about 4 years and belong to about 15 versions of the software product. The number of failures is significantly large but is not mentioned here because of confidentiality issues. However all the failures could not be accommodated in the

analysis due to missing attributes of the failure and defect data which were necessary for the analysis. Hence, only 24.15% of the total failures[2] were used for the analysis.

5.2.1.6 Analysis of Failure and Defect data

In this section we explain how we use the failure and defect data, whose details are given in section 5.2.1.4 and section 5.2.1.5, to determine the GQM metrics M1-M5 defined in section 5.1.2.

5.2.1.6.1 Metric M1

Metric M1, used to establish the significance of cause 1.1 (Delay in defect diagnosis), is defined in section 5.1.2 as follows:

$$\frac{\text{Number of rediscoveries which occur before defect is diagnosed}}{\text{Total number of rediscoveries}} \times 100$$

A single defect can cause multiple failures. Generally, each failure is analysed thoroughly by the maintenance team of the software provider to determine the defect which caused the failure. However, sometimes failures cannot be resolved to identify the associated defect due to insufficient data needed for analysis. Hence, there can be a number of failures, many of whom are rediscoveries, before the actual defect is diagnosed. The identification of such rediscoveries is a fairly simple task based on the information presented in section 5.2.1.4. Each failure refers to the relevant defect. The information with respect to a failure also includes the identification of the failure which actually helped in the diagnosis of the relevant defect. The date of occurrence of each failure is also available (Failure_Create_Date). Hence, the rediscoveries which occurred before the failure which 'originated' (caused a defect to be identified) the defect gives us the rediscoveries which occurred before the defect is diagnosed. The total number of rediscoveries can be easily determined as the difference of the total number of failures taken over all defects and the total number of defects.

[2] Note that these failures include both discoveries and rediscoveries of defects.

5.2.1.6.2 Metric M2

Metric M2, used to establish the significance of cause 1.2 (Delay in defect fixing), is defined in section 5.1.2 as follows:

$$\frac{\text{Number of rediscoveries which occur after the defect is diagnosed but before the defect is fixed by making necessary source code changes}}{\text{Total number of rediscoveries}} \times 100$$

Once the defect is diagnosed the maintenance team of the software provider designs a fix to eliminate the defect from the software product. The fix generally involves making changes to the source code of the software product and testing the fix. This process can sometimes take a significant amount of time during which rediscoveries of the defect continue to occur in the field. These rediscoveries can be easily identified based on the data presented in section 5.2.1.4. Whenever a new defect is diagnosed an entry is created into the issue tracking system. The corresponding data is available in the attribute 'Add_Date'. When the defect is eliminated by fixing it the maintenance team changes the 'State' attribute of the defect in the issue tracking system to 'closed'. The corresponding date is available in the attribute 'End_Date'. Hence, all rediscoveries which occurred after the 'Add_Date' and until the 'End_Date' give us the number of rediscoveries which occurred after the defect was diagnosed but before the defect was fixed by making the necessary source code changes.

5.2.1.6.3 Metric M3

Metric M3, used to establish the significance of cause 1.3 (Delay in fix release), is defined in section 5.1.2 as follows:

$$\frac{\text{Number of rediscoveries which occur after the defect is fixed by making the necessary source code changes but before the fix is bundled into patches and released to users}}{\text{Total number of rediscoveries}} \times 100$$

Generally, patches are released to the users periodically. There is pre-planned date on which the next patch is to be released. All defect fixes which are available by this date are bundled into the patch and released to the users of the software product. Hence, there is a certain delay after the defect is fixed before the fix is actually made available in a patch to the users during which rediscoveries continue to occur in the field. These rediscoveries can be easily identified based on the data presented in section 5.2.1.4. Each defect has a 'End_Date', which gives the date on which the defect was fixed, and 'Fix_Release_Date', which gives the date on which the fix was bundled into a patch and released to the users of the software product. Hence, all rediscoveries corresponding to the defect which occurred after the 'End_Date' until the 'Fix_Release_Date' give us the number of rediscoveries which occurred after the defect was fixed by making the necessary source code changes but before the fix was bundled into a patch and released to users.

5.2.1.6.4 Metric M4

Metric M4, used to establish the significance of cause 2.1 (Faulty defect fixes), is defined in section 5.1.2 as follows:

$$\frac{\text{Number of rediscoveries which occur after the defect fix is released in a patch and the user has installed the patch}}{\text{Total number of rediscoveries}} \times 100$$

In isolated cases, the fixes may turn out to be defective. In such case the rediscoveries continue to occur even after the patch containing the fix is available and is already installed by the users. These rediscoveries can be easily identified based on the data presented in section 5.2.1.4. In case of such a defect with a defective fix, a new defect entry is made into the issue tracking system. The symptom of the old defect entry in the issue tracking system is changed to a specific value 'X' and the 'Defect_Id' of the new defect is added to the 'Duplicate' attribute of the old defect entry. Hence, the number of rediscoveries after the 'Fix_Release_Date' of the old defect entry until the 'Fix_Release_Date' of the new defect entry gives us the number of rediscoveries which occur after the defect fix is released in a patch and the user has installed the patch.

5.2.1.6.5 Metric M5

Metric M5, used to establish the significance of cause 2.2 (User did not install the patch), is defined in section 5.1.2 as follows:

$$\frac{\text{Number of rediscoveries which occur after the defect fix is released in a patch and the user has not installed the patch}}{\text{Total number of rediscoveries}} \times 100$$

Knowing the rediscoveries which accounted for metrics M1 to M4 it is very simple to determine the number of rediscoveries which occurred because of the users' failure to install the patch. Any rediscovery which was not accounted in any of the metrics from M1 to M4 will be accounted here, which gives us the number of rediscoveries which occurred after the defect fix was released in a patch and the user did not install the patch.

5.2.1.7 Results and Interpretation

The results of the analysis of the data described in section 5.2.1.4 and section 5.2.1.5 are shown in the following table (Table 16).

Cause (see Figure 6)		Significance
Cause 1 (Patch not available to the user)	Cause 1.1 (Delay in diagnosis) *Metric M1*	1.1 %
	Cause 1.2 (Delay in fixing) *Metric M2*	39.8 %
	Cause 1.3 (Delay in fix release) *Metric M3*	7.1 %
Cause 2 (Patch available to the user)	Cause 2.1 (Faulty defect fixes) *Metric M4*	0 %
	Cause 2.2 (User did not install the patch) *Metric M5*	52 %

Table 16 Results: Case study – Software Provider

The same results are graphically represented by the following pie chart (Figure 9).

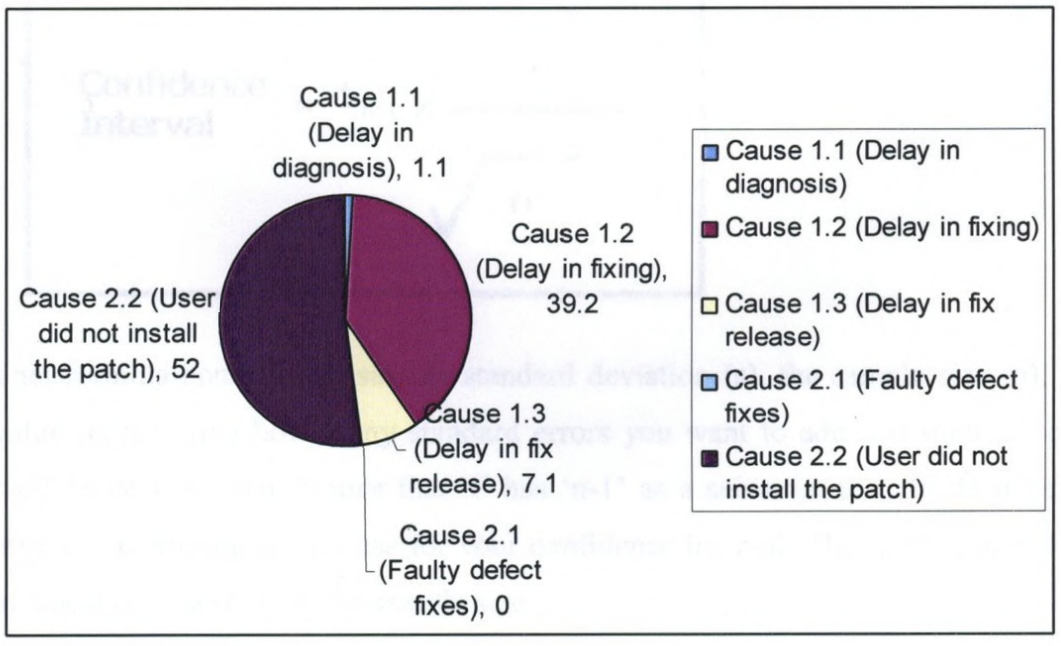


Figure 9 Results (Chart): Case study – Software Provider

Since only a sample (24.15%) of all the failures was used for analysis the results can be statistically interpreted [Rumsey] as shown in the following table (Table 17).

Confidence Level – 95 %		
Cause (see Figure 6)		**Significance**
Cause 1 (Patch not available to the user)	Cause 1.1 (Delay in diagnosis) *Metric M1*	0.63 to 1.57 %
	Cause 1.2 (Delay in fixing) *Metric M2*	37.6 to 42 %
	Cause 1.3 (Delay in fix release) *Metric M3*	5.95 to 8.25 %
Cause 2 (Patch available to the user)	Cause 2.1 (Faulty defect fixes) *Metric M4*	0 %
	Cause 2.2 (User did not install the patch) *Metric M5*	49.76 to 54.25 %

Table 17 Statistical interpretation of results: Case study – Software Provider [Rumsey]

The confidence level used for the statistical interpretation is 95%. The formula used for calculating the confidence intervals is as below [Rumsey].

$$\text{Confidence Interval} = t_{n-1} \times \frac{s}{\sqrt{n}}$$

This formula contains the sample standard deviation (s), the sample size (n), and a t-value representing how many standard errors you want to add and subtract to get the confidence you need. Notice that 't' has 'n-1' as a subscript to indicate which of the myriad t-distributions you use for your confidence interval. The 'n-1' is called degrees of freedom, where 'n' is the sample size.

To calculate the confidence interval for a cause we need to calculate the standard deviation (s) for that particular cause. Standard deviation (s) is calculated using the formula below [Rumsey].

$$s = \sqrt{p \times (1 - p)}$$

The variable 'p' here is probability in the case of a binomial distribution, in our case given by the significance of each cause calculated from the sample (see column with heading 'Significance' in Table 16).

5.2.2 The Case Study – Software User

The second case study is done on a medium size (50-200 employees) software service provider which provides system administration services to various other organisations who actually use the software [SME]. The employees of this organization, which provides system administration services, were requested to respond to a questionnaire and their responses were analyzed to achieve our research objectives.

5.2.2.1 Components of Research Design

For case studies, the following five components of a research design (i.e., Questions, Propositions, Unit of analysis, Logic linking data to propositions, and Criteria for interpreting the findings) are especially important [Yin 2003].

5.2.2.1.1 Questions

1. What is the significance of cause-2.2.1 (see Figure 6; Awareness) with respect to occurrence of software defect rediscoveries?
2. What is the significance of cause-2.2.2 (see Figure 6; Skepticism) with respect to occurrence of software defect rediscoveries?
3. What is the significance of cause-2.2.3 (see Figure 6; Pro-activeness) with respect to occurrence of software defect rediscoveries?

4. What is the significance of cause-2.2.4 (see Figure 6; Downtime availability) with respect to occurrence of software defect rediscoveries?
5. What is the significance of cause-2.2.5 (see Figure 6; IT staff availability) with respect to occurrence of software defect rediscoveries?
6. What is the significance of cause-2.2.6 (see Figure 6; Complexity of patch installation process) with respect to occurrence of software defect rediscoveries?

5.2.2.1.2 Propositions

As explained in section 5.2.1.1.2, an exploratory study by its nature has a legitimate reason for not having any propositions [Yin 2003]. This is because in an exploratory study we do not begin the study with a theory but instead conduct the study to develop a theory which may be tested by another study. However, an exploratory study should have some purpose. Therefore, instead of propositions, the design of an exploratory study should state this purpose, as well as the criteria by which an exploration would be judged successful. In our case, the research objectives implied by questions Q6 to Q24, defined in section 5.1.2, will serve as the purpose of the study as well as the criteria for successful completion of our study.

Appendix A contains the questionnaire which incorporates the questions Q6 to Q24.

5.2.2.1.3 Unit of Analysis

The unit of analysis is an SME [SME] providing system administration services to various other organizations. All the respondents of the questionnaire in Appendix A belong to this SME under study. It is important to note here that the study itself is exploratory in nature and hence based on the feasibility in our current situation the best we can do is a single case study. Similar multiple case studies can follow this investigation for further proof. We believe that in the context of our research questions, the case which is accessible to us is a typical case. The customer organisations of the organisation under study are diverse with respect to the various factors like software products under use, business models, industry, etc. However as the system administrators, who are the participants of the study and also the employees of the SME

under study, belong to the same organisation they may share some common characteristics with respect to the practice of system administration.

5.2.2.1.4 Logic linking the data to the propositions

As mentioned in section 5.2.1.1.4, this component of the research design is one of the least developed in case studies. The "pattern matching" approach by Donald Campbell [Campbell 1975] cannot be used here because of non-availability of propositions as our study is exploratory in nature. Hence, the purpose of the case study as mentioned in section 5.2.2.1.2 will be the sole guiding criteria in this regard.

5.2.2.1.5 Criteria for interpreting the findings

As mentioned in section 5.2.1.1.5, this component of the research design is one of the least developed in case studies. This component is important mostly in case of case studies with pre-defined theoretical propositions. However, the exploratory nature of our study does not provide us with any propositions. Hence, this component has no relevance in our case.

5.2.2.2 Quality of Research Design

The quality of empirical research needs to be established by four tests described below [Yin 2003].

5.2.2.2.1 Construct Validity

The construct validation of the questionnaire was critical to the overall validity of the study. The two types of *construct validity* that applied to the design of this questionnaire were *content* and *face* validity. These types of validity cannot be measured in a quantitative way, but we believe that we have met these through the extensive design procedures described.

Content validity is based on the extent to which a questionnaire reflects the specific intended domain of content [Carmines 1991]. This was established in our study by covering all the factors which influence software users in their decision making with

respect to the installation of a patch from Figure 6, which were in turn identified from literature dealing with software rediscoveries and patch management. All these factors, which are the causes for rediscoveries (causes 2.2.1 to 2.2.6 in Figure 6), have been identified in the taxonomy of rediscovery causes (see Figure 6), which in turn has been construct validated by the six experts listed in Table 2.

Face validity is concerned "with how a measure or procedure appears. Does it seem like a reasonable way to gain the information the researchers are attempting to obtain? Does it seem well designed? Does it seem as though it will work reliably?" [CSU]. This is met in our study by involving six experts (see Table 18) in reviewing the questionnaire, both content and form.

Expert	Researcher / Practitioner	# years of Experience	Experience	Areas of Expertise
1	Researcher	30	Developer (small scale); Architect; Researcher; Research Lead; Consultant; Pedagogue;	Requirements Engineering, Empirical Studies, Software Quality, Software Maintenance, Software Process Engineering.
2	Researcher and Practitioner	15	Developer; Quality Assurance Specialist; Consultant; Research Lead; Collaboration Manager; Team Lead;	Software Engineering, Software Quality Assurance, Software Maintenance, Project Estimation.
3	Practitioner	16	Statistical Consultant	Statistics, Optimization.
4	Researcher	6	Developer (small scale); Researcher; Pedagogue;	Software Quality, Software Metrics, Empirical Studies, Software Process(Agile).
5	Researcher	6	Lead programmer (mid-sized industrial projects); Researcher; Tester; Designer; Architect	Empirical studies, Requirements Engineering, Software Architecture, Software Quality, Video Game Design, Usability
6	Practitioner	4	Statistical Analyst; IT Technician	Statistics, Software System Administration

Table 18 Experts involved in the validation of questionnaire (see Appendix A for questionnaire)

5.2.2.2.2 Internal Validity

Internal Validity is defined by Cooke and Campbell as the "approximate validity with which we infer that a relationship between two variables is causal" [Cooke 1979]. As mentioned in section 5.2.1.2.2, this validity is not a concern in case of exploratory studies, which happens to be the type of study we are pursuing.

5.2.2.2.3 External Validity

External validity deals with the problem of knowing whether a study's findings are generalizable beyond the immediate case study [Yin 2003]. In this study, as mentioned in section 5.2.2.1.3, the case is 'typical' and fairly representative of the population of software administrators in the context of our research questions. As mentioned in section 5.2.2.1.3, all the respondents of the questionnaire are system administrators who belong to the same organization. However, they provide system administration services to various other organizations which are diverse with respect to factors like software products under use, business models, industry, etc.

5.2.2.2.4 Reliability

The objective of this test is to be sure that if a later investigator followed the same procedures as described by an earlier investigator and conducted the same case study all over again, the later investigator should arrive at the same findings and conclusions. The goal of reliability is to minimize the errors and biases in a study. In this study, we have properly documented our research process using GQM method from top to bottom, from research goals to research objectives, questions and metrics at the quantitative level. Also, the data is collected using a questionnaire. Hence, we conclude that our research design is reliable.

5.2.2.2.5 Conclusion Validity

Conclusion validity in the degree to which conclusions we make based on the findings are reasonable. We discuss the conclusions in section 6, and there we demonstrate that all our conclusions are rooted in the results, thereby maintaining conclusion validity.

5.2.2.3 Data Collection

Based on our case study questionnaire design (see Appendix A), the organization forming the unit of analysis gathered the data from the system administrators. This data was made available to us for analysis. The system administrators who participated in this case study are associated with the organization for various market research

activities. The participation was voluntary. Although these participants may not be representative of the entire population of system administrators they do have diverse background in terms of the types of software products they administer, the size of IT teams and organizations they work for and their experience level. This diversity has been quantitatively described in section 5.2.2.4.

5.2.2.4 Dataset profile

The number of system administrators involved in the case study was 100. The profile of the dataset with respect to various attributes is shown in the following figures (Figure 10 to Figure 14).

Attribute 1: Software Products administered by Respondents

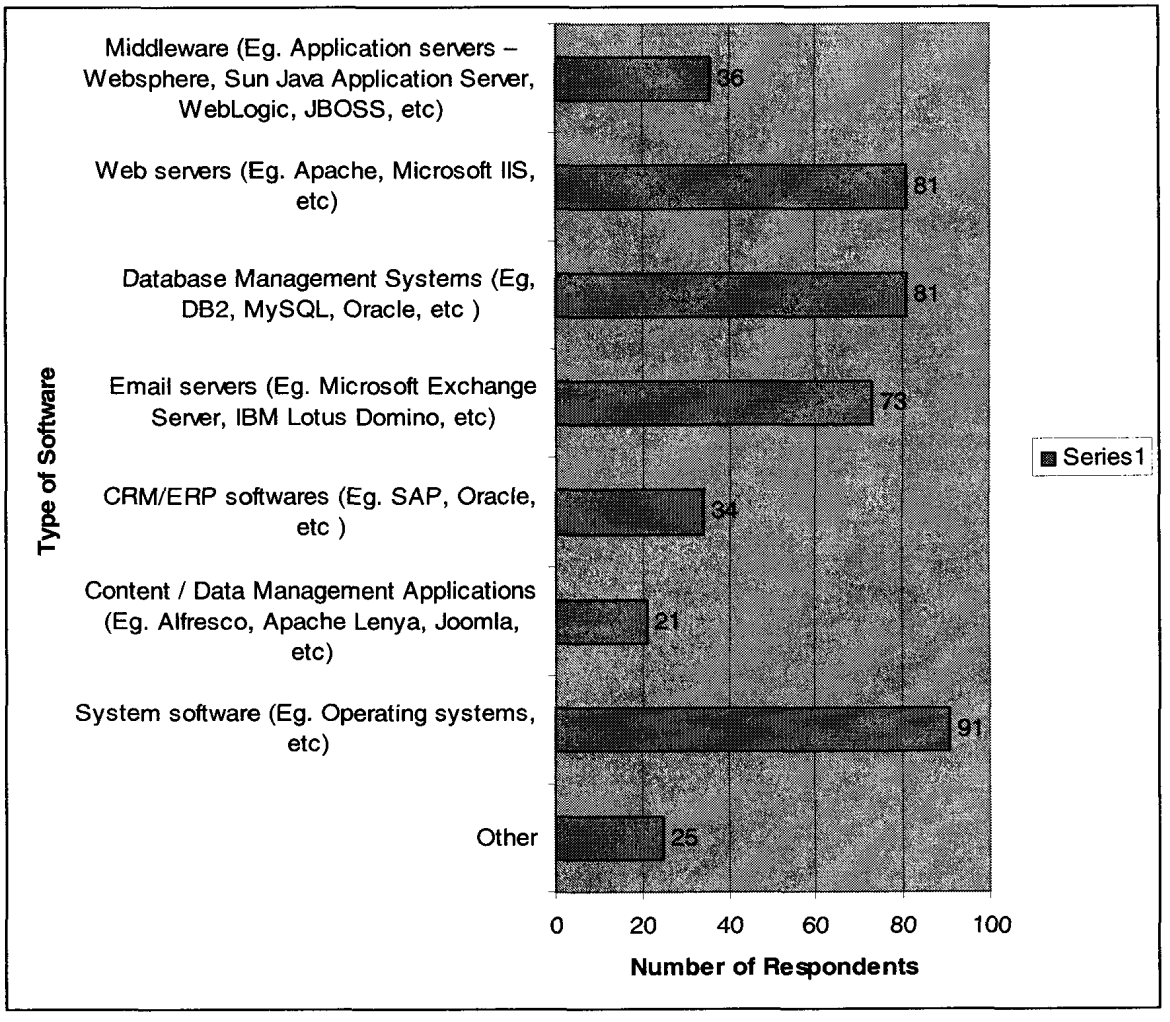


Figure 10 Demographic data of respondents – Software type

Attribute 2: Industries of Organizations for which the system administrators worked

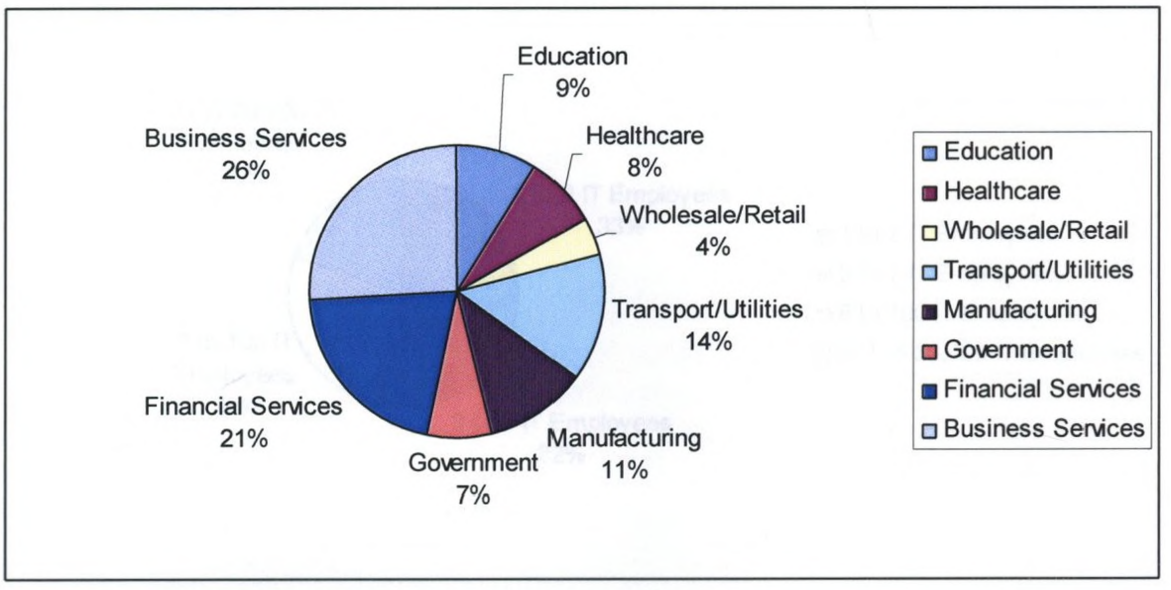


Figure 11 Demographic data of respondents – Industry

Attribute 3: Number of Employees of the Organizations for which the system administrators worked

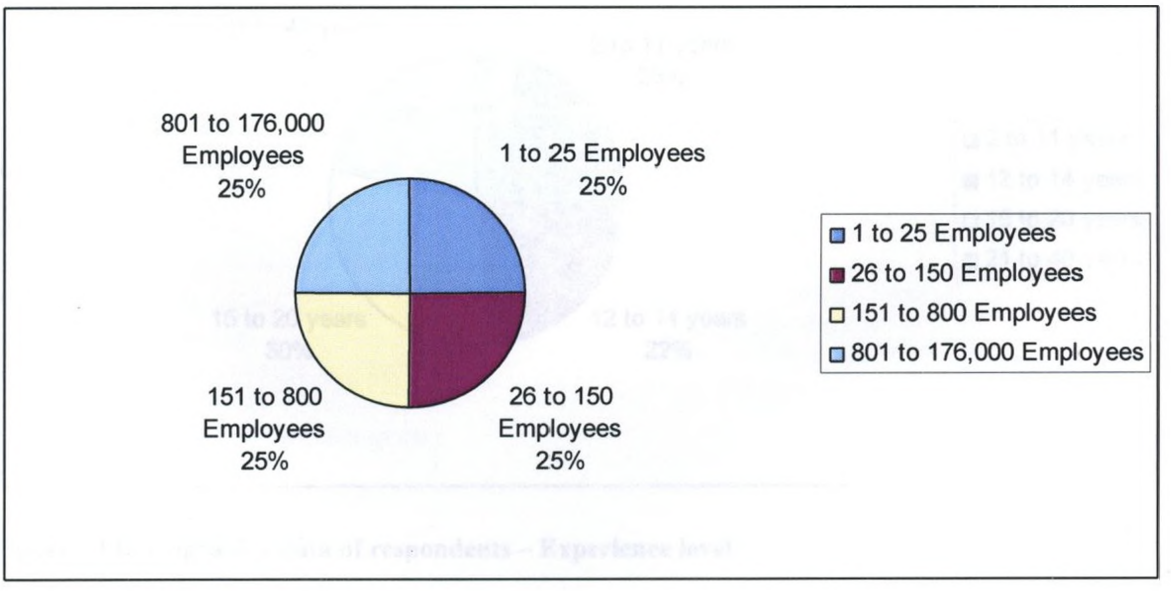


Figure 12 Demographic data of respondents – Organization size

Attribute 4: Number of IT Employees of the Organizations for which the system administrators worked

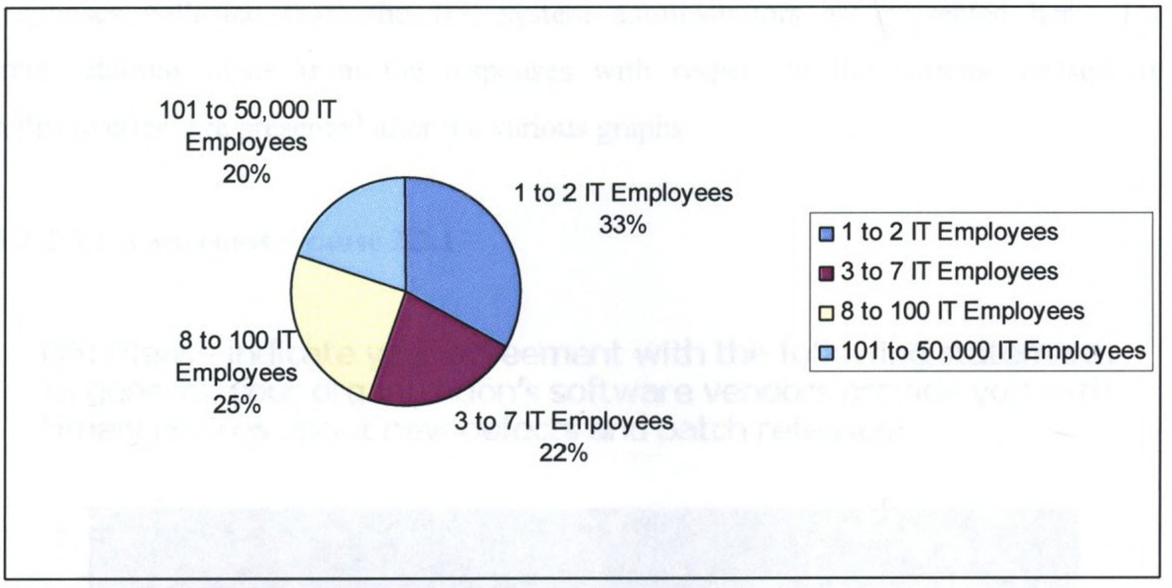


Figure 13 Demographic data of respondents – IT department size

Attribute 5: Experience level of Respondents

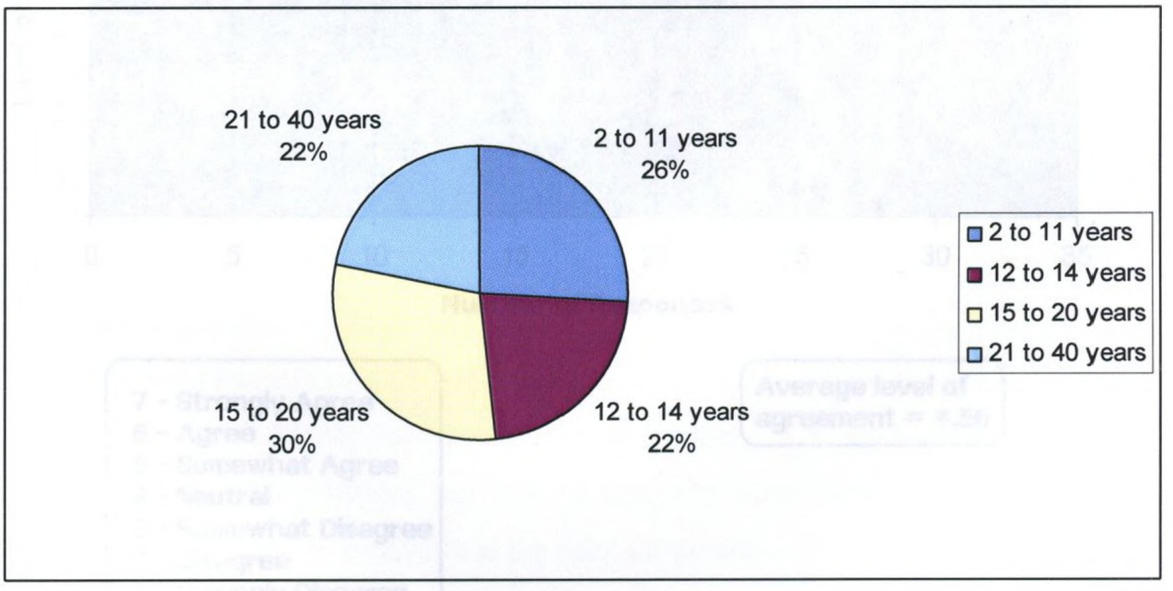


Figure 14 Demographic data of respondents – Experience level

5.2.2.5 Results and Interpretation

This case study was done to achieve research objectives Q5 to Q24 mentioned in section 5.1.2. The responses to questionnaire mentioned in section 5.2.2.1.2 (see

Appendix A) were collected from system administrators who work for an organization which provides system administration services to various other organizations. The responses collected from the 100 system administrators are presented here. The interpretations made from the responses with respect to the various 'causes for rediscoveries' are presented after the various graphs.

5.2.2.5.1 Awareness – cause 2.2.1

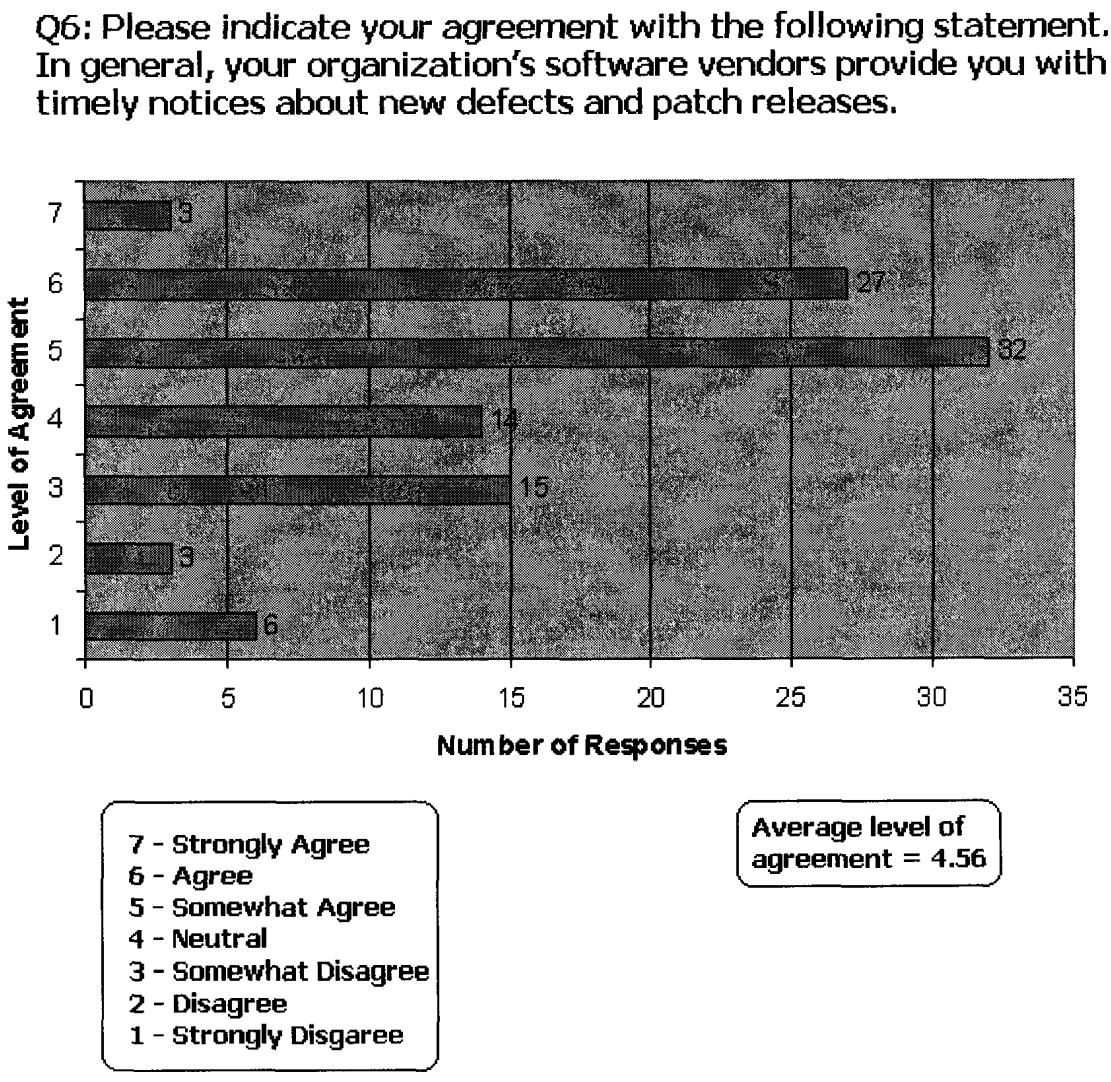


Figure 15 Results – Research Objective Q6

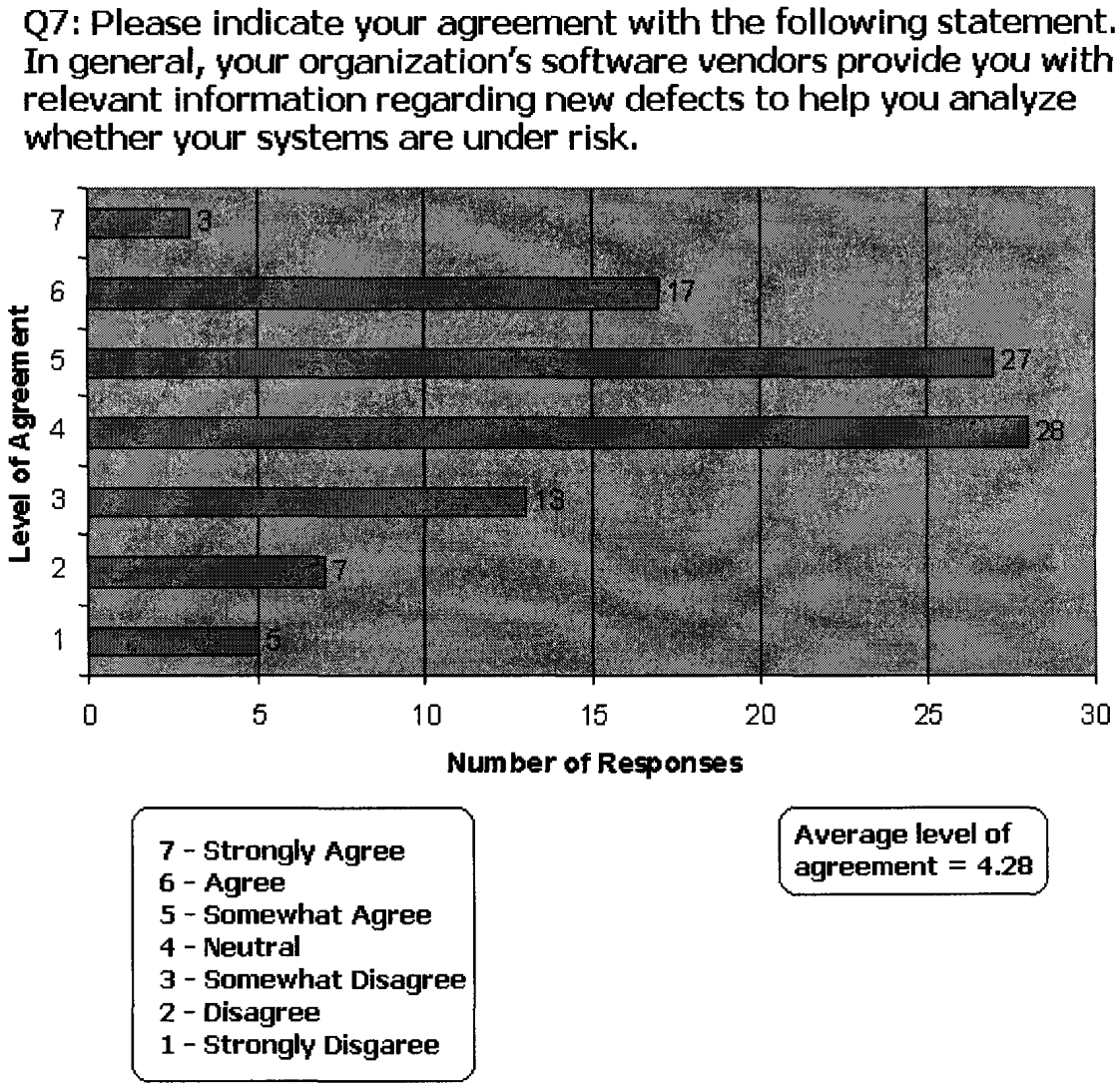


Figure 16 Results – Research Objective Q7

As can be seen from Figure 15 and Figure 16 the average levels of agreement for Q6 and Q7, 4.56 and 4.28, are just above the neutral point, i.e. 4. This shows that there is some effort from the software provider to keep the users aware of the new defects identified in the software product and the relevant patches. However, there is still scope for improvement. The feedback obtained from the system administrators who participated in the research sheds more light on these aspects. According to these participants, most software providers, generally, just post the information about new defects and patches on their support website. However, very few of the software providers personally notify the system administrators about this information. Also, the

information provided to the system administrators, to assess the risk their systems are under, is not user-friendly. It requires interpretation and extensive searching of this information on the part of the system administrators to extract the information they need to do the risk assessment.

5.2.2.5.2 Skepticism – cause 2.2.2

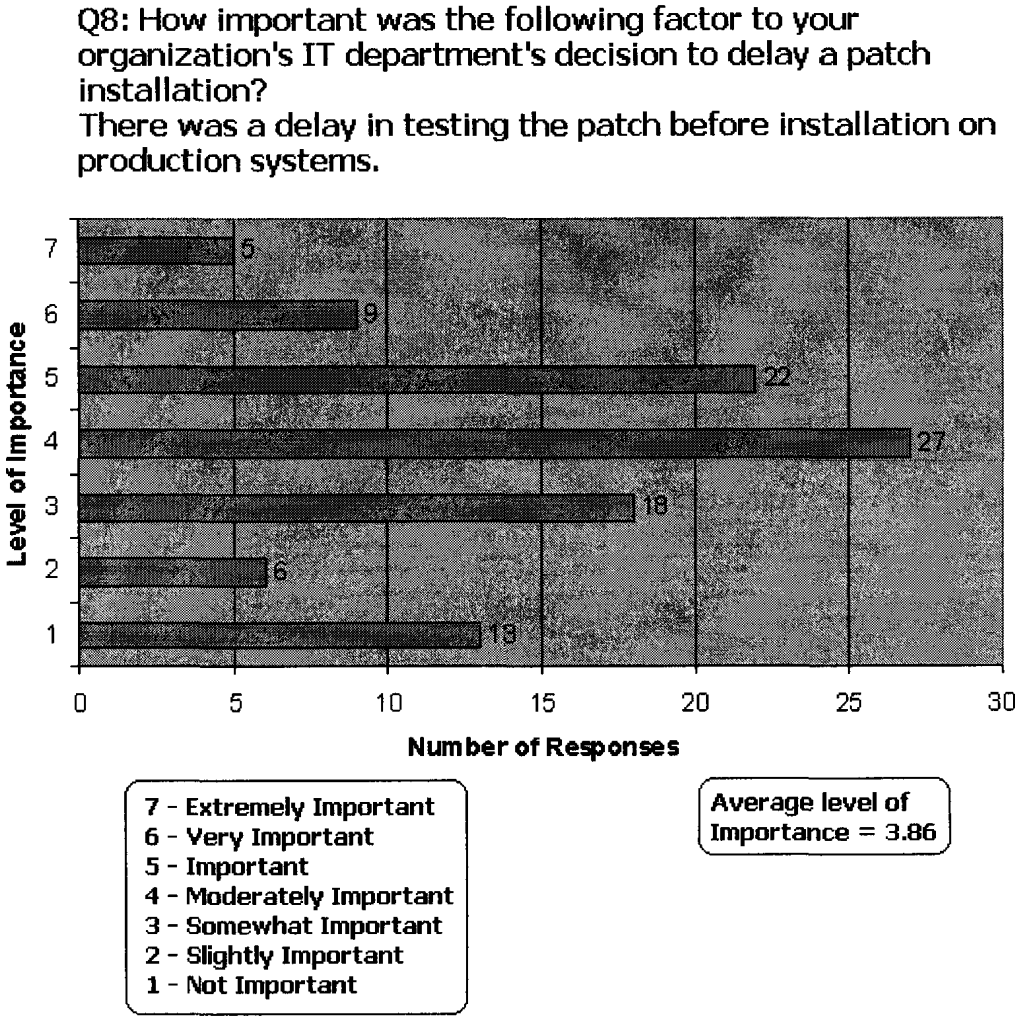


Figure 17 Results – Research Objective – Q8

Q9: What was the average delay in patch installation on production systems at your organization in the last 24 months due to the following factor?
Testing the patch before installation.

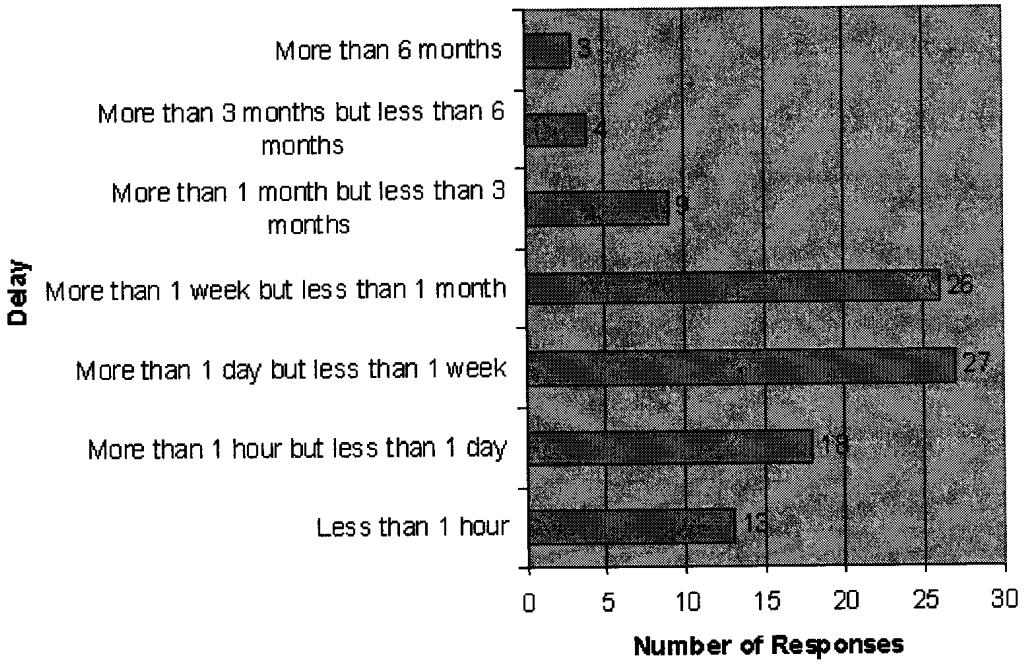


Figure 18 Results – Research Objective Q9

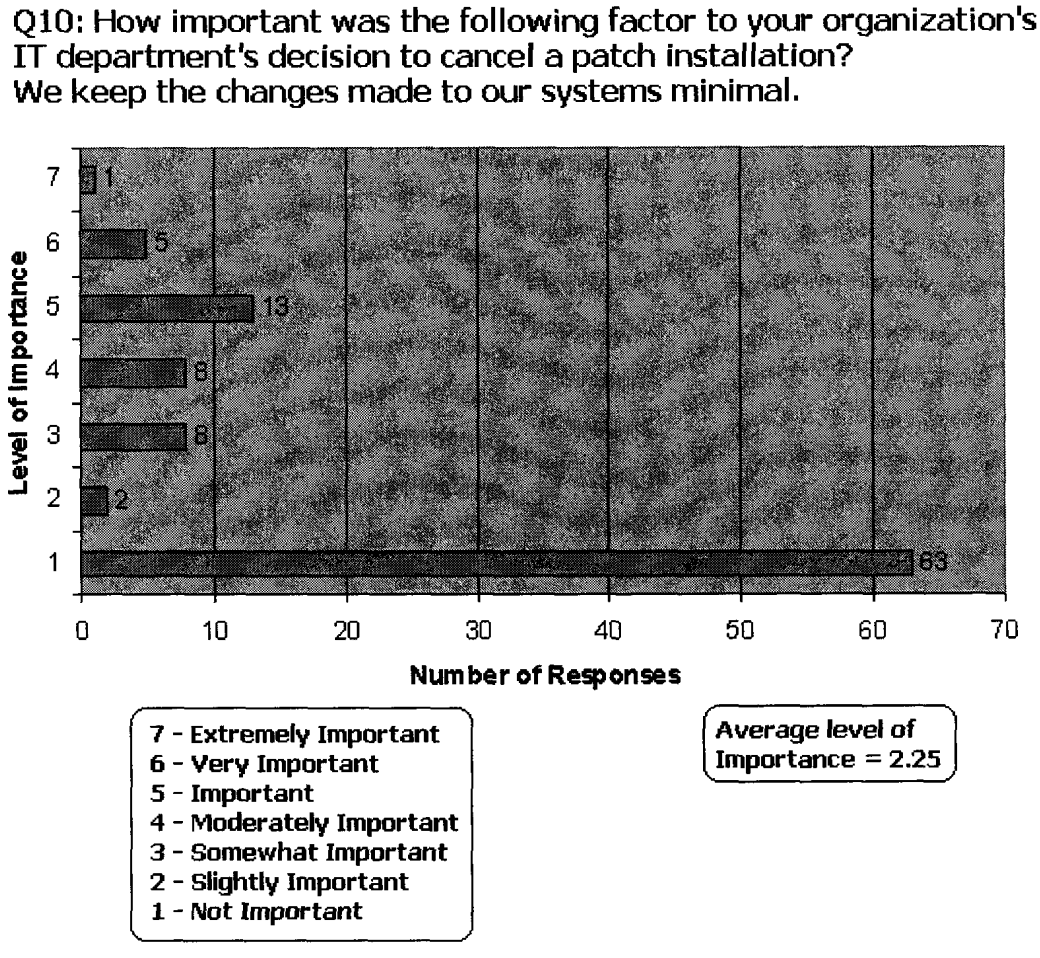


Figure 19 Results – Research Objective Q10

According to data in Figure 17 it can be observed that 87% of the system administrators attach some level of importance to testing the patches before installation on production systems, although knowing that it might cause a delay in the installation of patches on production systems. This shows that most system administrators were skeptical that sometimes patches can cause issues with their systems unless handled carefully. The delay due to testing depends on the actual process followed by the system administrator. It can be seen from the data in Figure 18 that 84% of the system administrators believed that the delay due to testing is less than a month and 58% of the system administrators believed that the delay due to testing is less than a week. The data in Figure 19 shows that 63% of the system administrators say that they are not reluctant to make changes to their systems just to avoid issues that might arise due to these changes. Patching the systems is one such change which is known to sometimes cause issues with the

functionality and performance of the systems. In general, from the above analysis we can infer that most system administrators are not reluctant to install new patches but are definitely skeptical about them. To overcome this skepticism they do invest considerable resource in terms of time to test the patches before installation on production systems.

5.2.2.5.3 Pro-activeness – cause 2.2.3

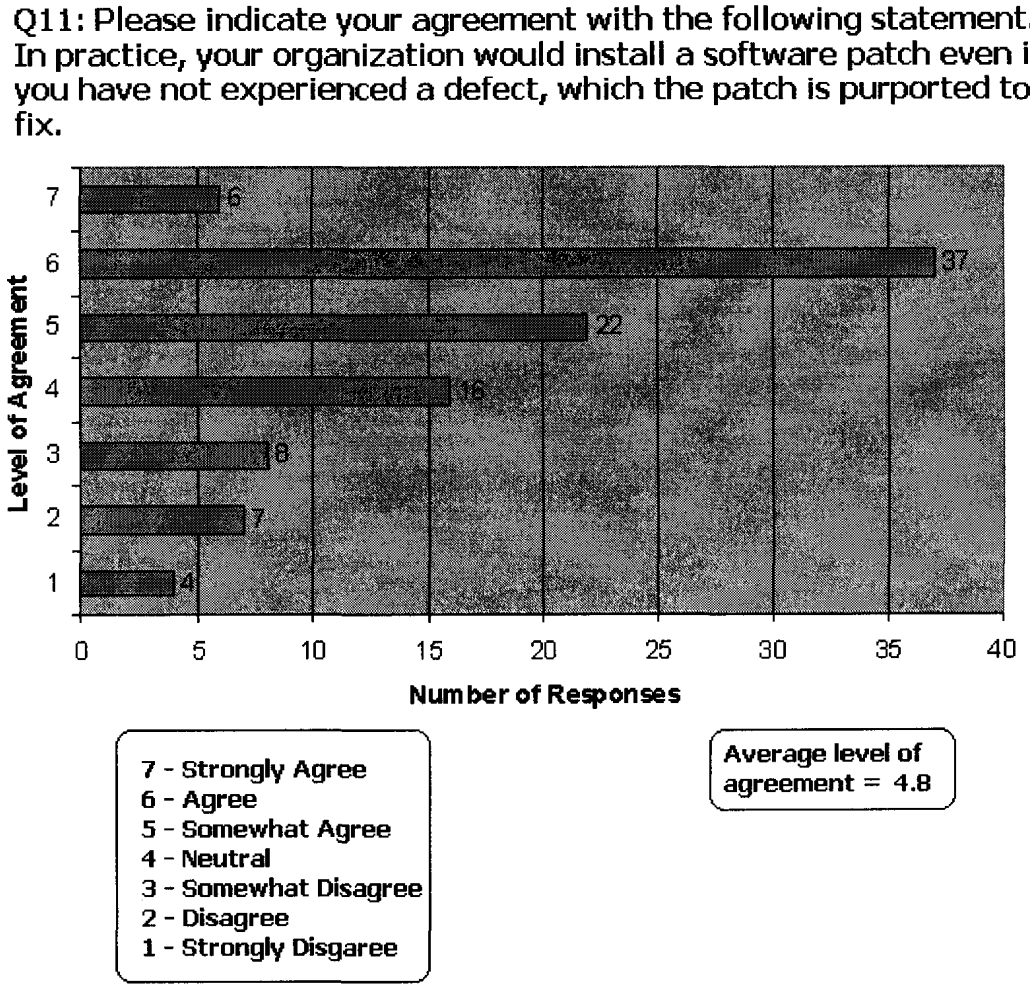


Figure 20 Results – Research Objective Q11

Q12: How important was the following factor to your organization's IT department's decision to delay a patch installation?
The systems were functioning normally and we had not experienced any defect which the patch was known to fix.

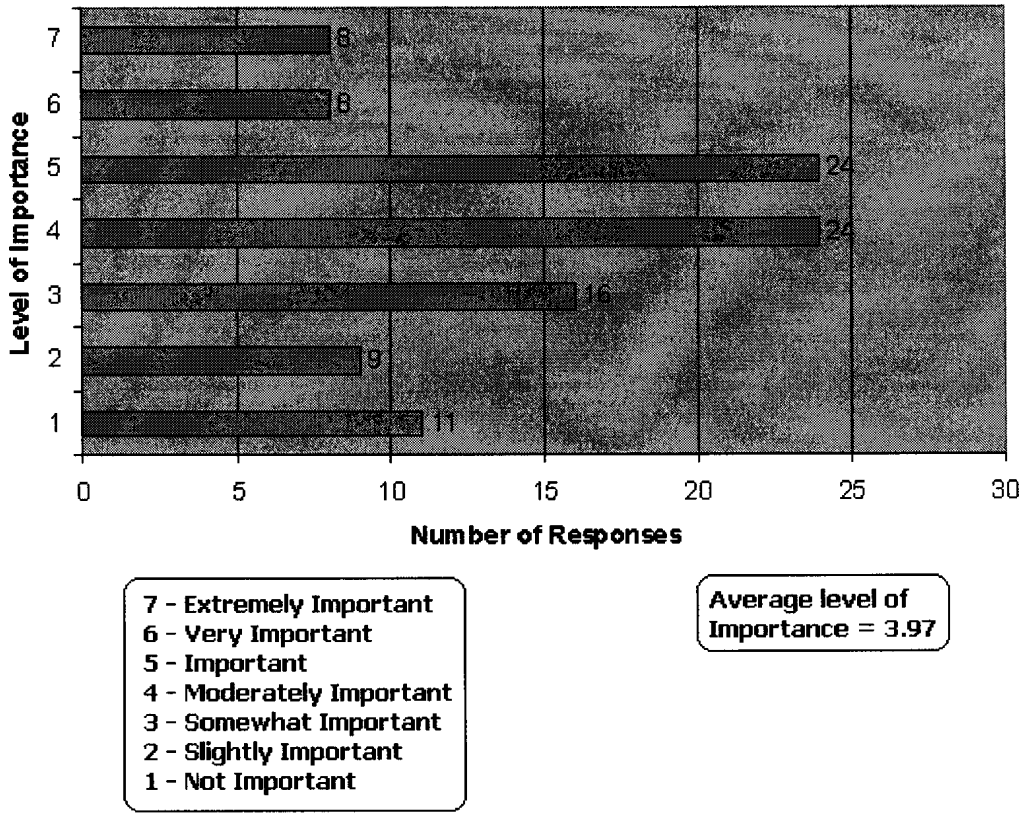


Figure 21 Results – Research Objective Q12

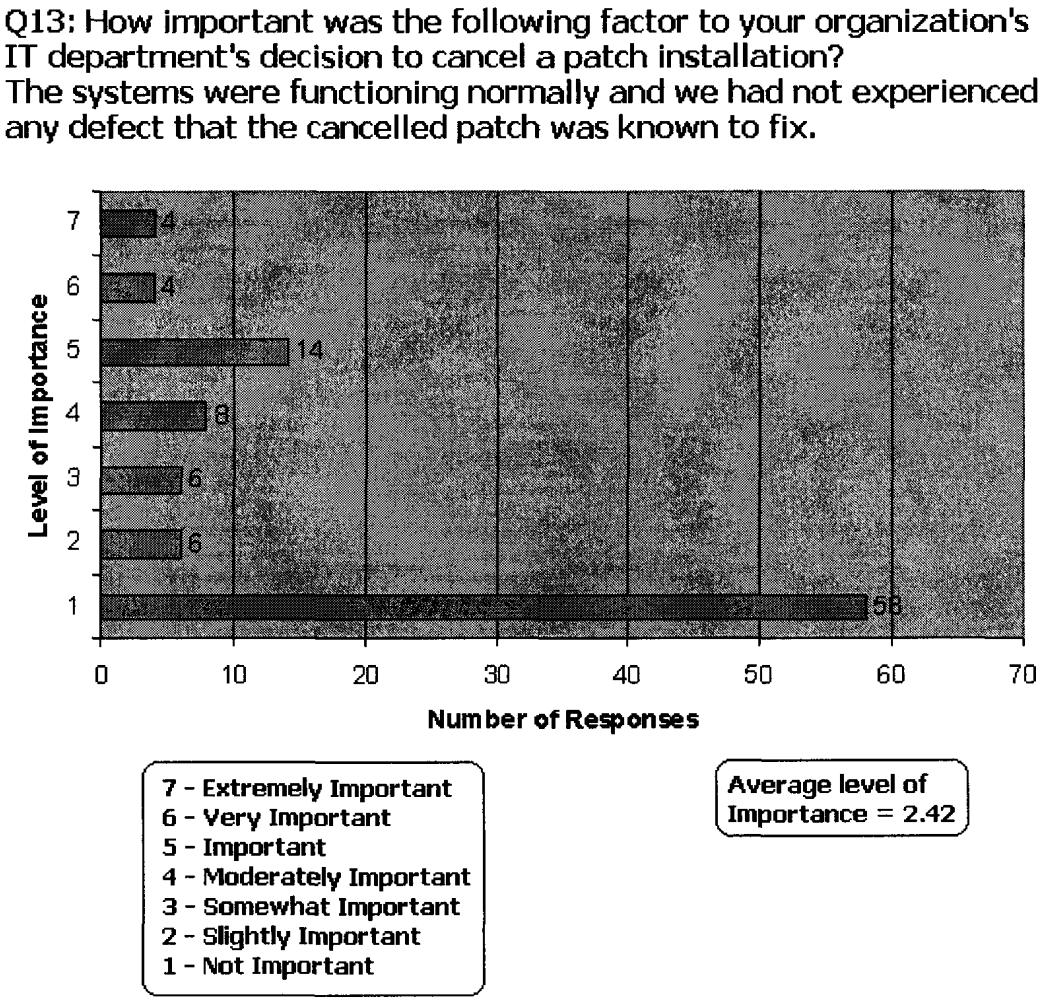


Figure 22 Results – Research Objective Q13

According to data in Figure 20 it can be observed that 65% of the system administrators attach some level of agreement to the fact that they install patches pro-actively. From the feedback from the system administrators, this is especially true in case of security patches. The data in Figure 21 shows that 89% of system administrators attach some level of importance to the fact that they would delay the installation of a patch if the systems are currently functioning normally. Although this may seem to be in conflict with the previous inference from Figure 20, a detailed analysis from the feedback obtained from system administrators shows that system administrators, generally, would not do an unscheduled maintenance for installing the patch if the systems are functioning normally, security patches being an exception. However, they would definitely install the patch during the next scheduled maintenance opportunity even if

the systems are functioning normally. Hence, the 'delay' which the system administrators refer to here is actually the delay due to non-availability of system downtime. The inference made from Figure 20 is further supported by the data in Figure 22 which shows that 58% of the system administrators believe that 'systems functioning normally' is no excuse to cancel a patch installation and hence establishing that most system administrators are pro-active when it comes to patch installation on their systems.

5.2.2.5.4 Downtime availability – cause 2.2.4

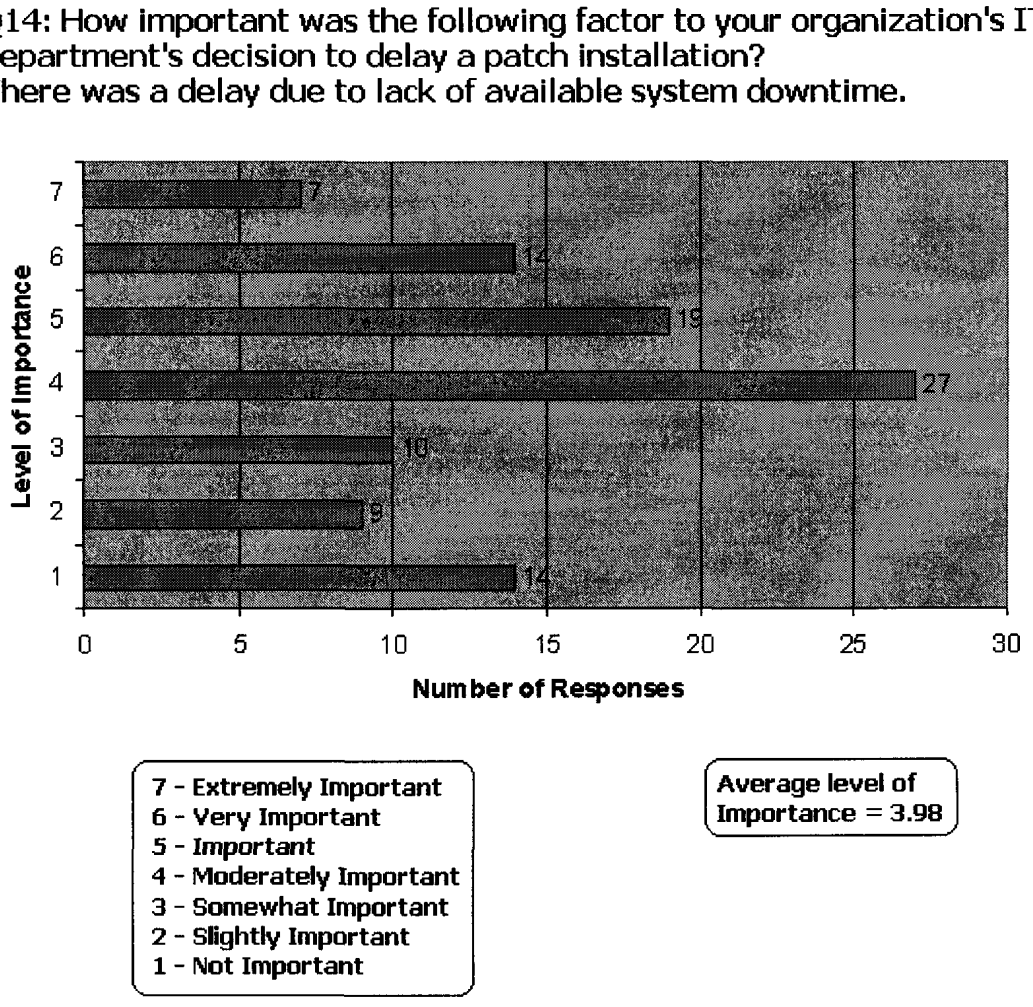


Figure 23 Results – Research Objective Q14

Q15: What was the average delay in patch installation on production systems at your organization in the last 24 months due to the following factor?
Lack of available system downtime.

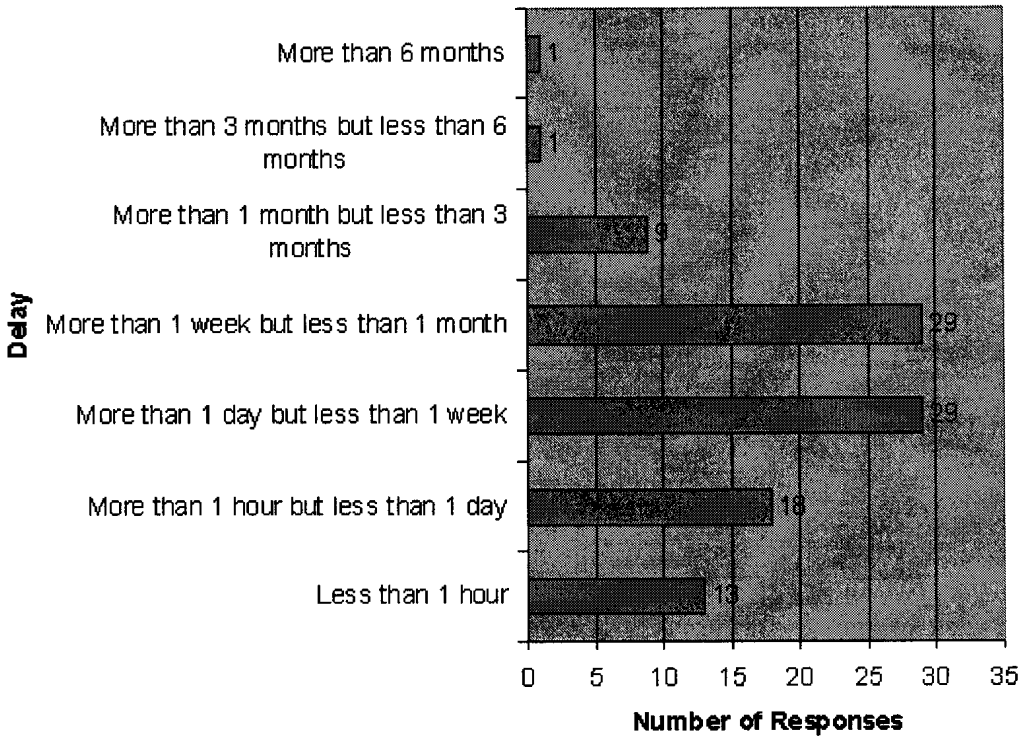


Figure 24 Results – Research Objective Q15

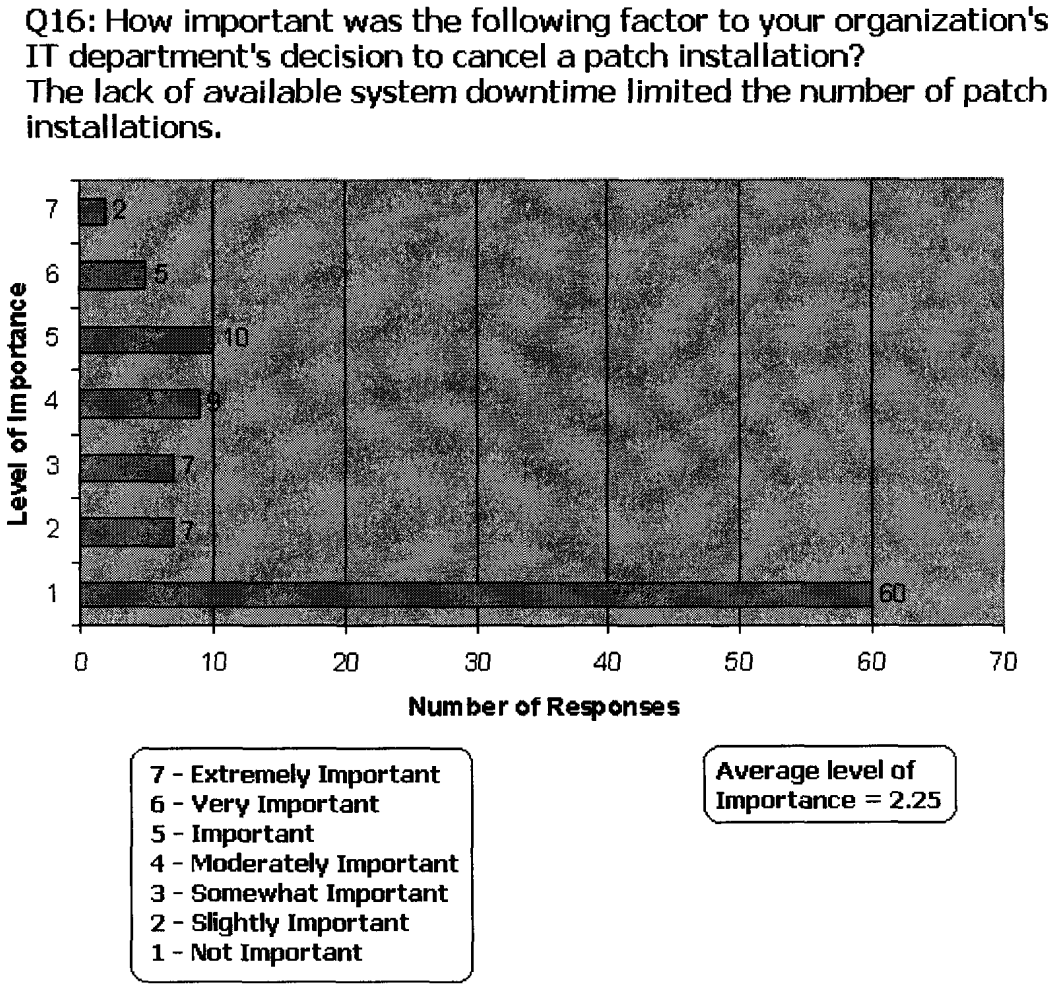


Figure 25 Results – Research Objective Q16

According to data in Figure 23 it can be observed that 86% of the system administrators attached some level of importance to availability of system downtime or the lack of it as a factor in their decision to delay the installation of a patch on production systems. 75% of the system administrators who responded said that they have redundant systems on standby. In spite of this it is still difficult to find downtime because of the fact that these redundant systems are mainly used for high availability and they can not be taken out of service for installing patches even if they are on standby. The data in Figure 24 shows that 89% of the system administrators believe that the delay in installing a patch on their production systems due to non-availability of downtime is less than one month and 60% of them believe that it is less than a week. The data in Figure 25 shows that 60% of the system administrators believe that non-availability of adequate system downtime did

not limit the number of patch installations made on their systems. To summarize the data in Figure 23, Figure 24 and Figure 25 we can say that availability of adequate system downtime continues to be an issue with respect to installation of patches in spite of availability of redundant systems. However, most system administrators do not cancel a patch installation due to non-availability of downtime although they delay the patch installation.

5.2.2.5.5 IT staff availability – cause 2.2.5

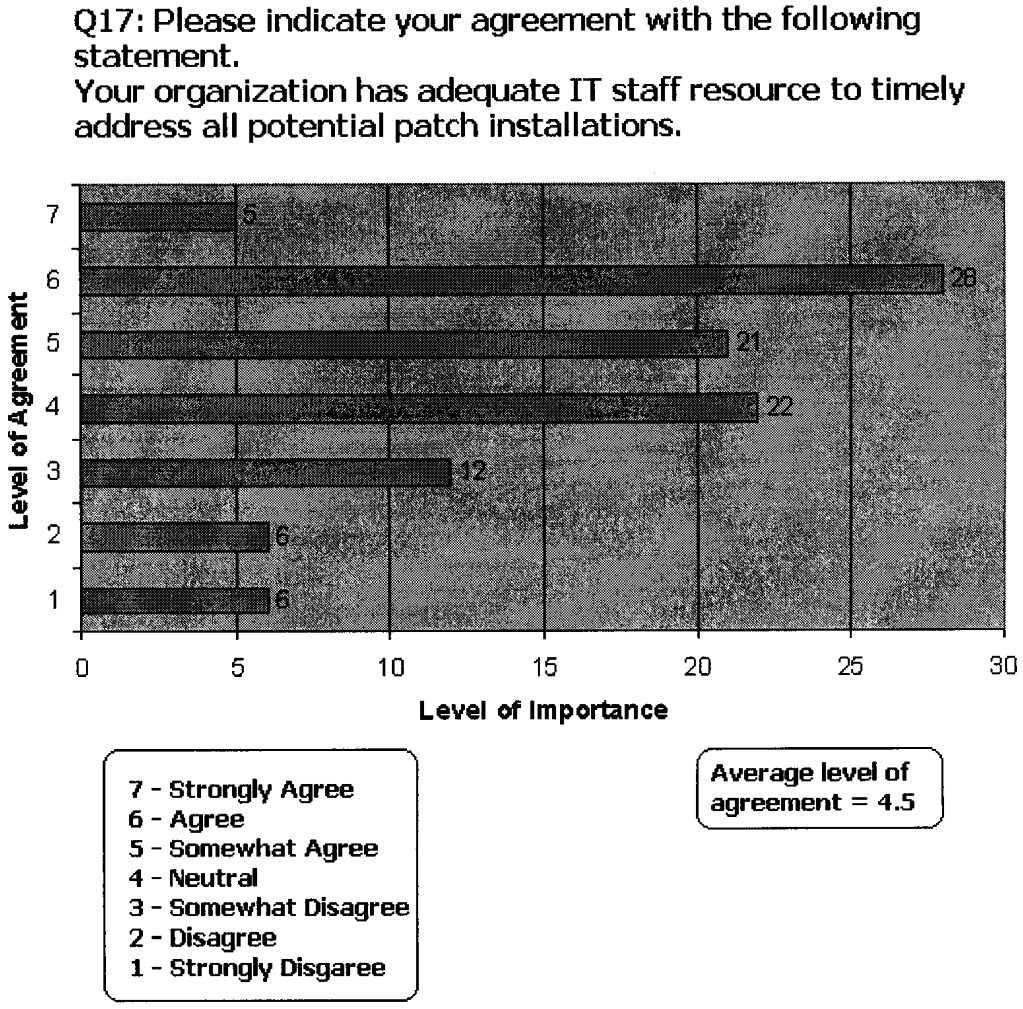


Figure 26 Results – Research Objective Q17

Q18: How important was the following factor to your organization's IT department's decision to delay a patch installation?
There was a delay due to lack of available IT staff to handle the patch installation process.

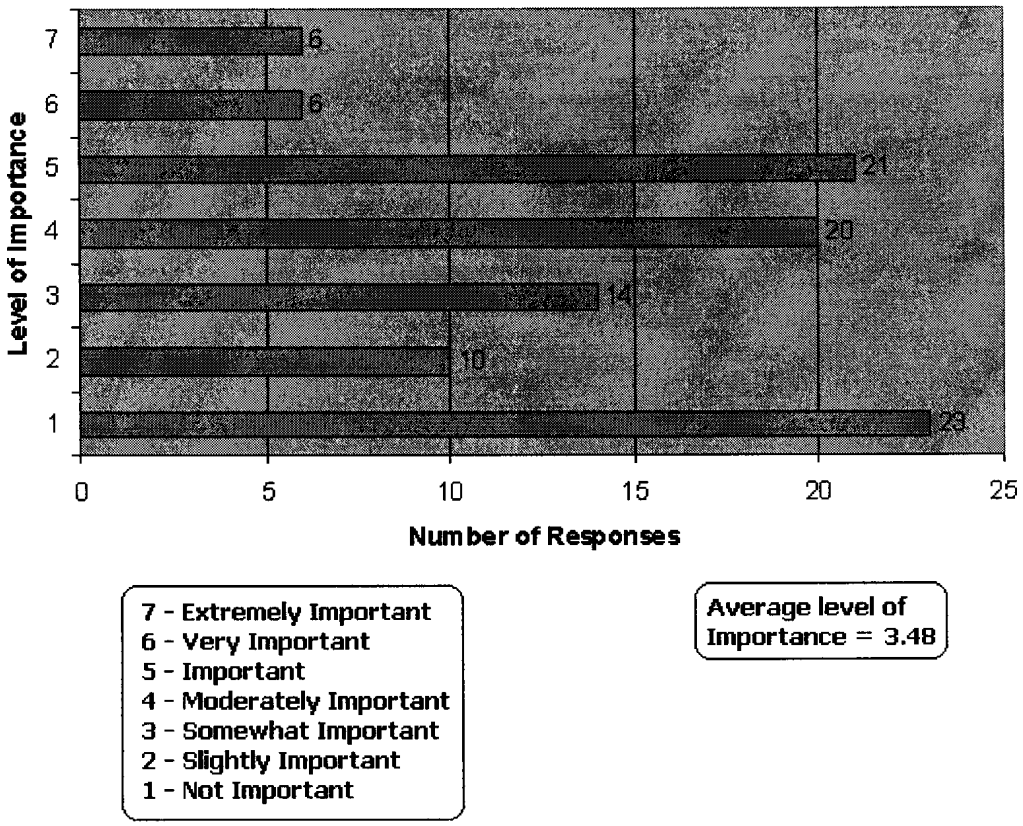


Figure 27 Results – Research Objective Q18

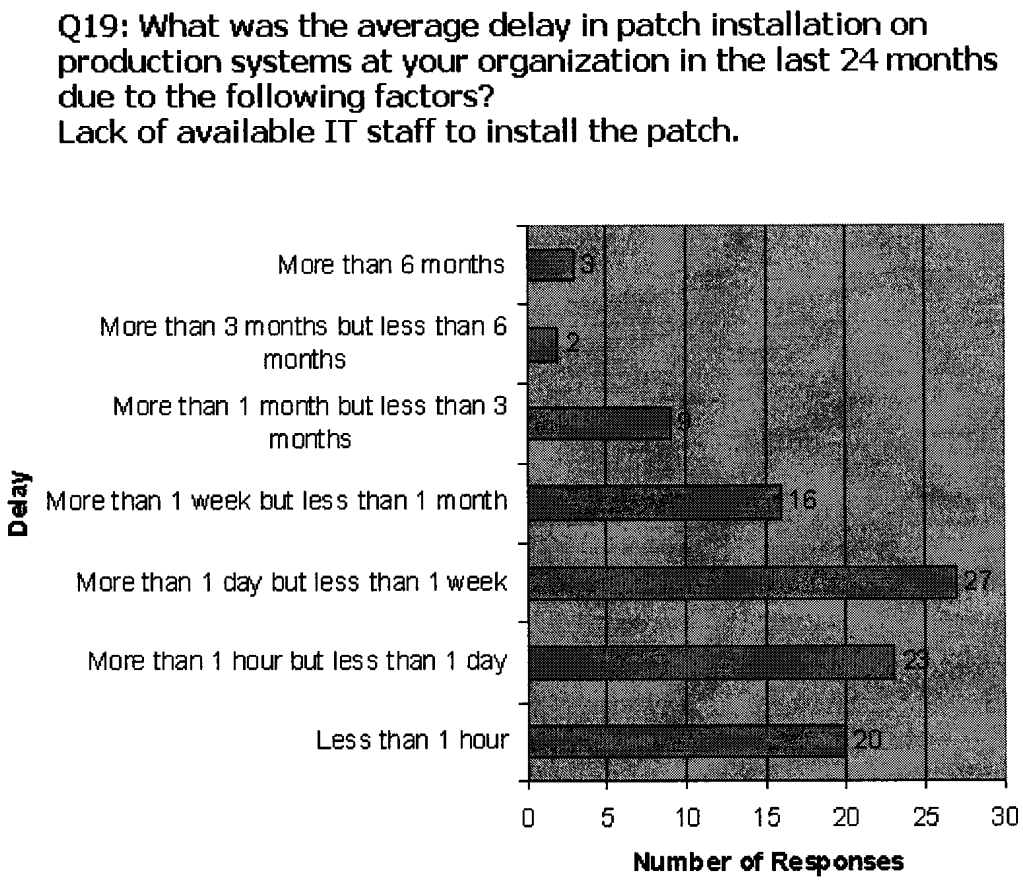


Figure 28 Results – Research Objective Q19

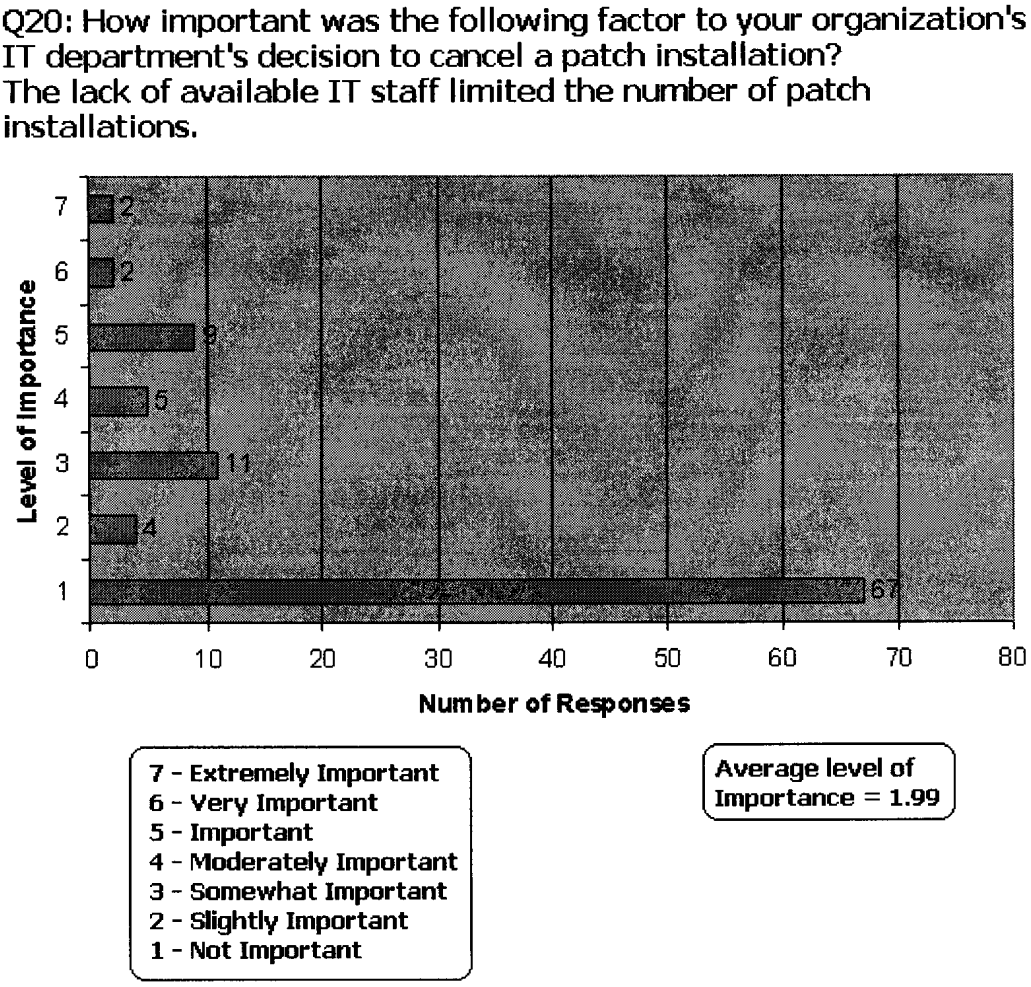


Figure 29 Results – Research Objective Q20

According to the data in Figure 26 it can be observed that 54% of the system administrators agreed to some level that they have adequate IT staff to timely address potential patch installations, whereas 24% of them disagreed to some level for the same and the remaining 22% were undecided. Also, from the data in Figure 27, the average level of importance for the 'lack of IT staff'' as a factor in the decision to delay a patch installation is 3.48, which is less than the average level of importance for any factor in this decision, which is 3.702. Also, from the data in Figure 28, 70% of the system administrators believed that the delay in patch installation due to 'lack of IT staff' is less than a week. Also, from the data in Figure 29, 67% of system administrators believed that 'lack of IT staff' had no importance at all when it came to the decision of cancelling a patch installation. Also, the average level of importance for this factor in

the system administrators' decision to cancel a patch installation is 1.99, which is less than the average level of importance for any factor, i.e., 2.174. From the data in Figure 26, Figure 27, Figure 28 and Figure 29 we can summarize that a majority of system administrators agreed more than they disagreed that their organization had the necessary IT staff resource to address potential patch installations in a timely manner. Also, 'lack of IT staff' has an average level of importance which is lesser than the average level of importance for any factor in both the decisions – decision to delay a patch installation and decision to cancel a patch installation.

5.2.2.5.6 Complexity of patch installation process – cause 2.2.6

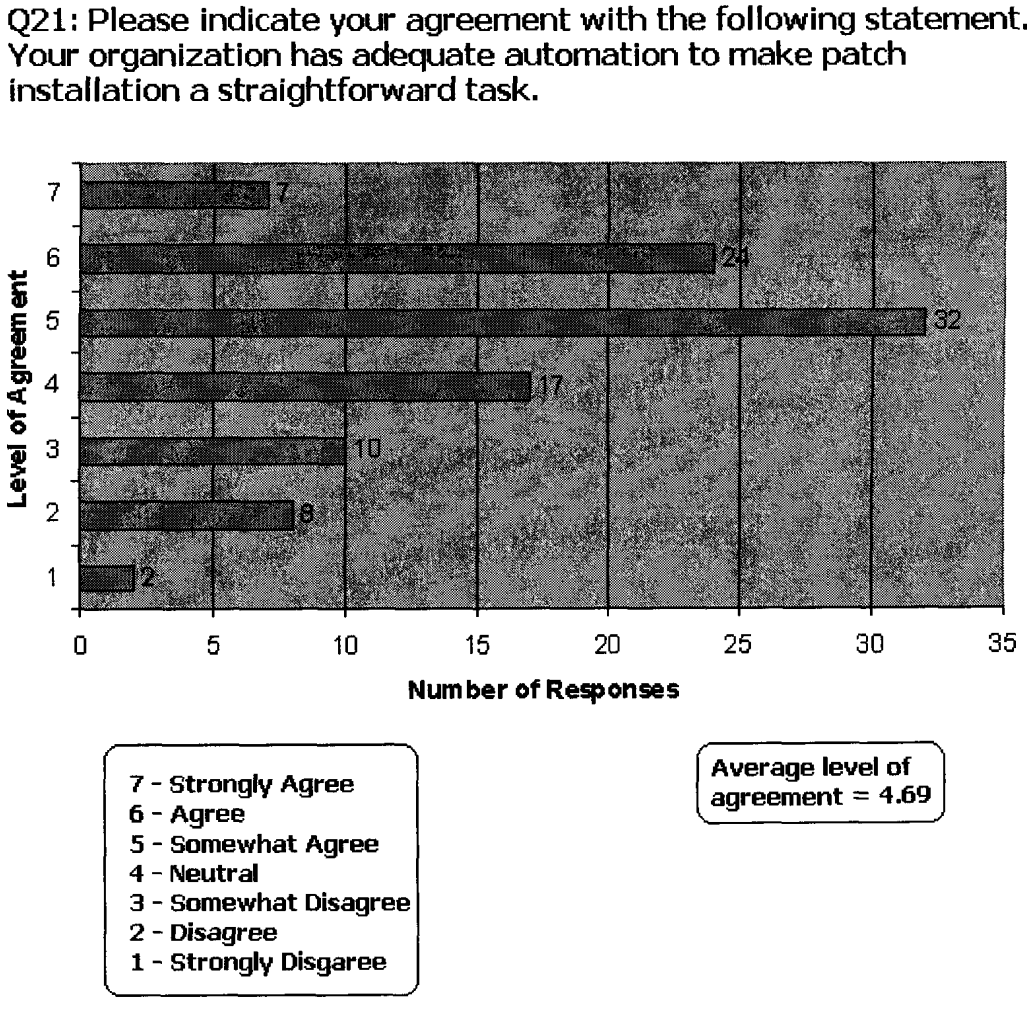


Figure 30 Results – Research Objective Q21

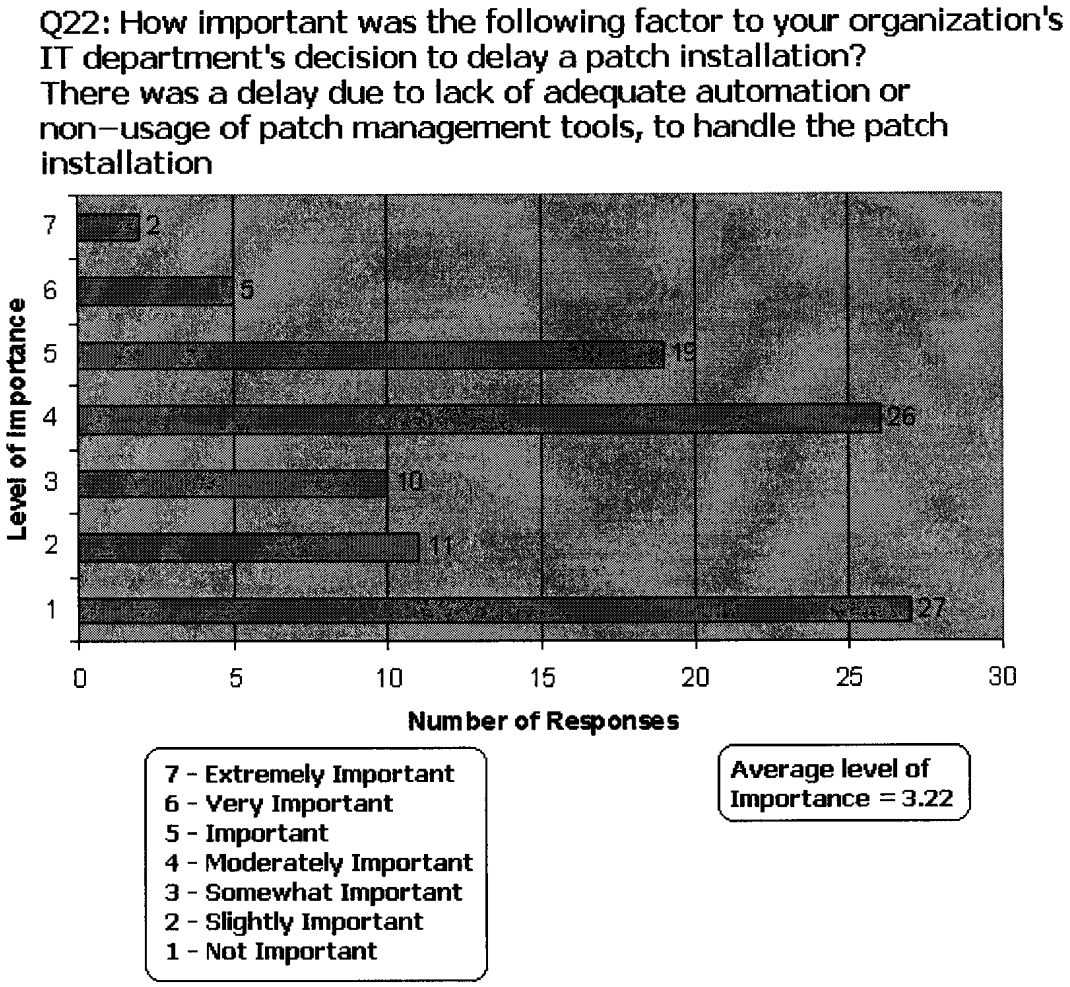


Figure 31 Results – Research Objective Q22

Q23: What was the average delay in patch installation on production systems at your organization in the last 24 months due to the following factor?
Lack of adequate automation of the patch installation process.

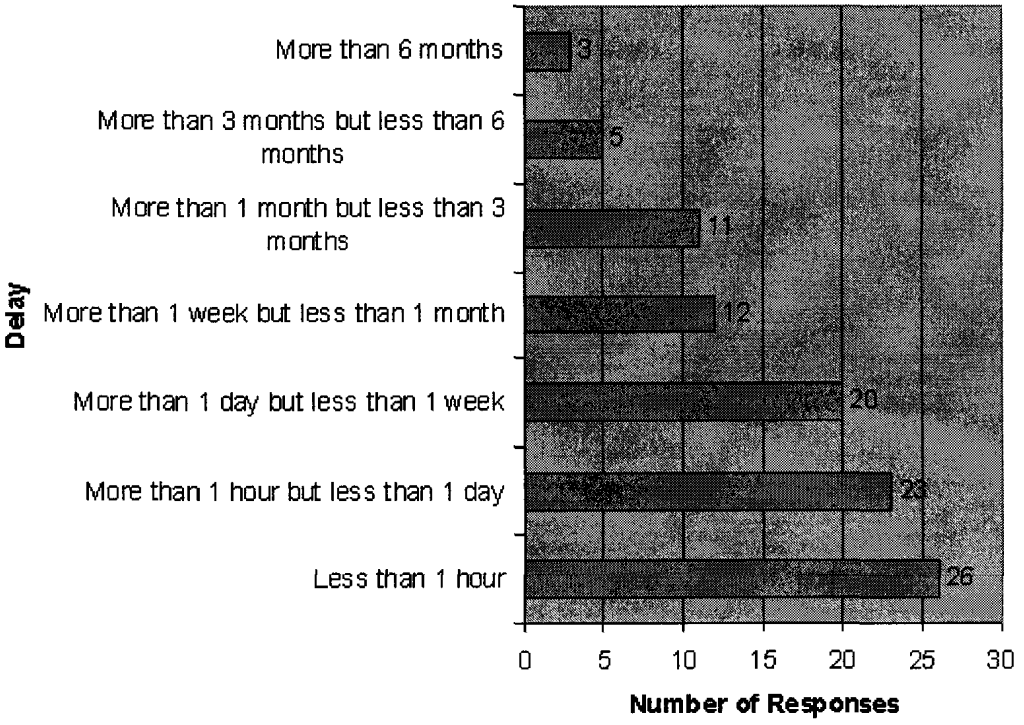


Figure 32 Results – Research Objective Q23

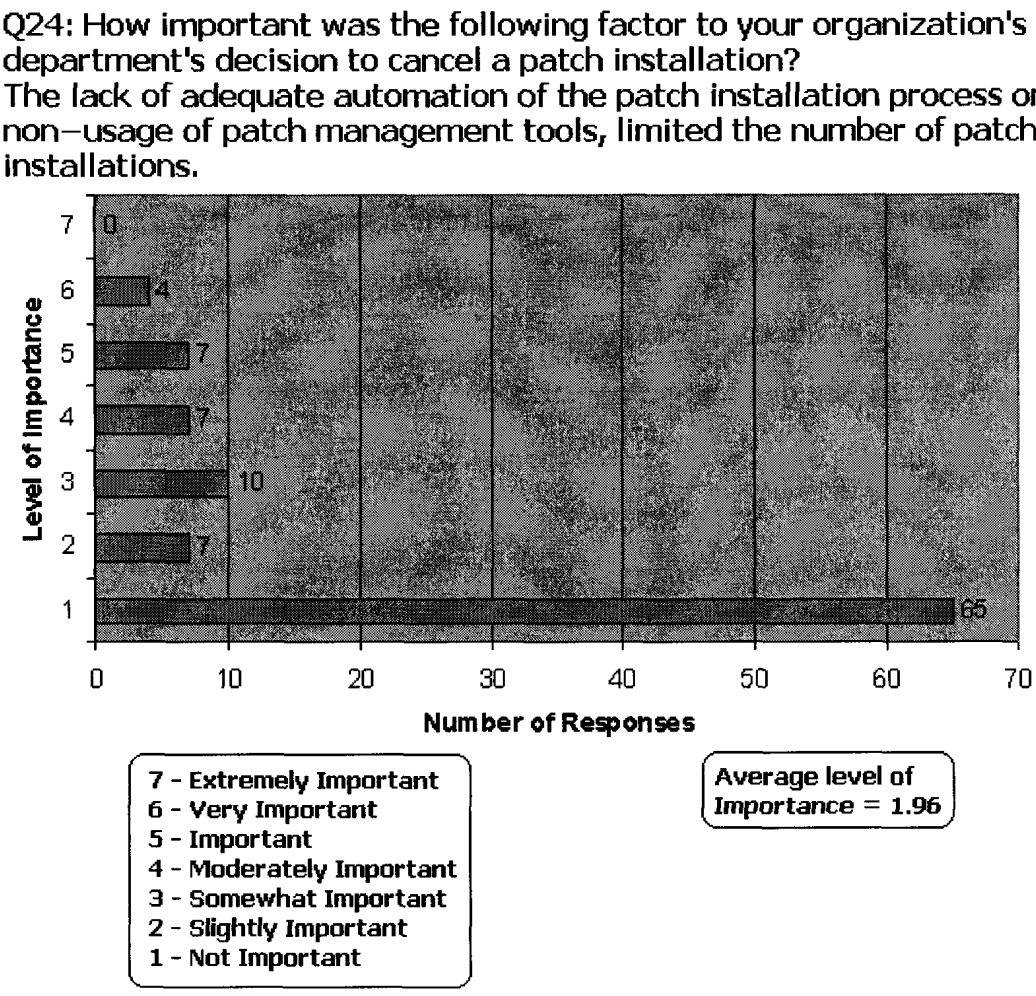


Figure 33 Results – Research Objective Q24

According to data in Figure 30 it can be observed that 63% of the system administrators agreed to some level that they have adequate automation to handle patch installations, whereas 20% of them disagreed to some level for the same and the remaining 17% were undecided. From the data in Figure 31, the average level of importance for the 'lack of adequate automation of the patch installation process' as a factor in the decision to delay a patch installation is 3.22, which is less than the average level of importance for any factor in this decision, i.e., 3.702. From the data in Figure 32, 69% of the system administrators believed that the delay in patch installation due to 'lack of adequate automation of the patch installation process' is less than a week. From the data in Figure 33, 65% of system administrators believed that 'lack of adequate automation of the patch installation process' had no importance at all when it came to the decision of

cancelling a patch installation. The average level of importance for this factor (1.94) is less than the average level of importance for any factor in the decision to cancel a patch installation (2.174). From the data in Figure 30, Figure 32, Figure 33 and Figure 34 we can summarize that system administrators agreed more than they disagreed that their organization had the adequate automation of the patch installation process to handle patch installations. The 'lack of adequate automation of the patch installation process' has an average level of importance which is lesser than the average level of importance for any factor in both the decisions – decision to delay a patch installation and decision to cancel a patch installation.

5.2.2.5.7 Relative importance of causes (2.2.2 – 2.2.6)

In this section, we discuss the relative significance of each of the causes for rediscoveries.

5.2.2.5.7.1 Relative importance with respect to delaying a patch installation

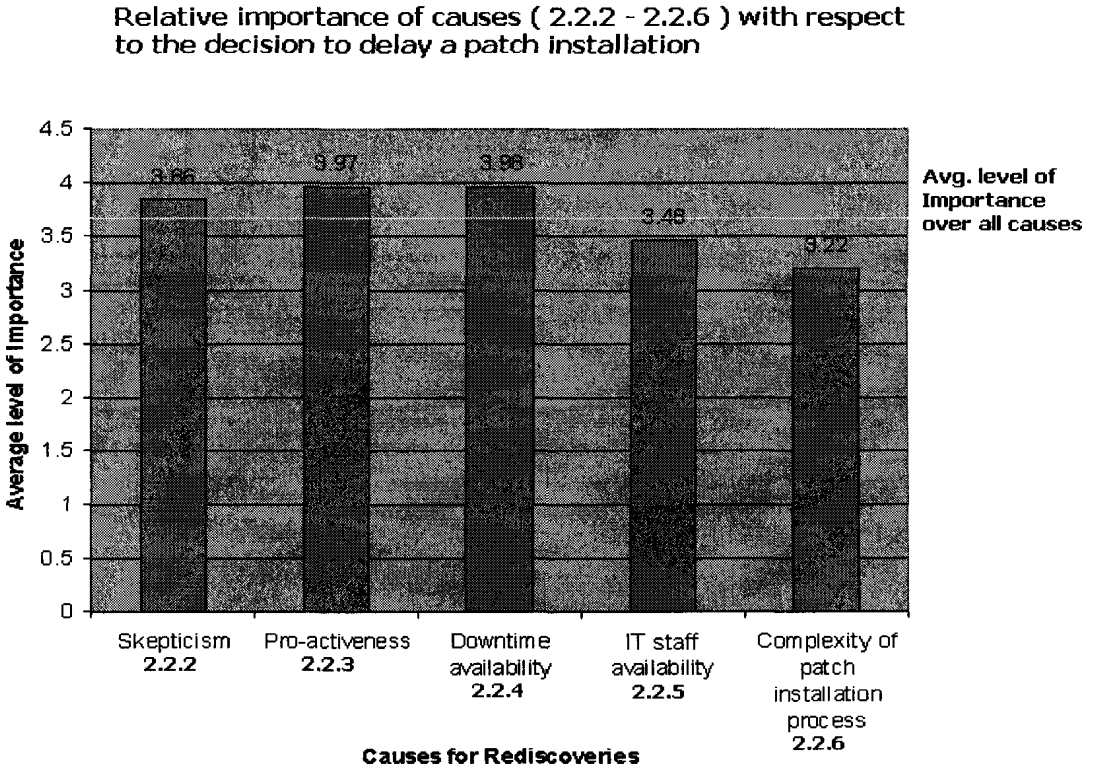


Figure 34 Relative importance of causes (2.2.2 – 2.2.6) w.r.t. the decision to delay a patch installation

Cause	Responses which said 'There was a delay'	Responses which said 'Delay more than 1 week'
Skepticism – cause 2.2.2	87 %	42 %
Pro-activeness – cause 2.2.3	42 %	-
Downtime availability – cause 2.2.4	86 %	40 %
IT staff availability – cause 2.2.5	77 %	30 %
Complexity of patch installation process – cause 2.2.6	73 %	31 %

Table 19 Relative delays in patch installation due to various causes for rediscoveries

According to data in Figure 34 it can be observed that 'Skepticism', 'Pro-activeness' and 'Downtime availability' were given above average importance whereas 'IT staff availability' and 'Complexity of patch installation process' were given below average importance by system administrators when it comes to delaying the installation of a patch. Also, the data in Table 19 shows that more system administrators reported that there was a delay due to 'Skepticism' and 'Downtime Availability' when compared to the number of system administrators who reported that there was a delay due to 'Pro-activeness', 'IT Staff availability' and 'Complexity of patch installation process'. Also, Table 19 shows that more system administrators reported that the delay due to 'Skepticism' and 'Downtime Availability' was more than a week when compared to the number of system administrators who reported that there was a delay due to 'IT Staff availability' and 'Complexity of patch installation process'.

5.2.2.5.7.2 Relative importance with respect to cancelling a patch installation

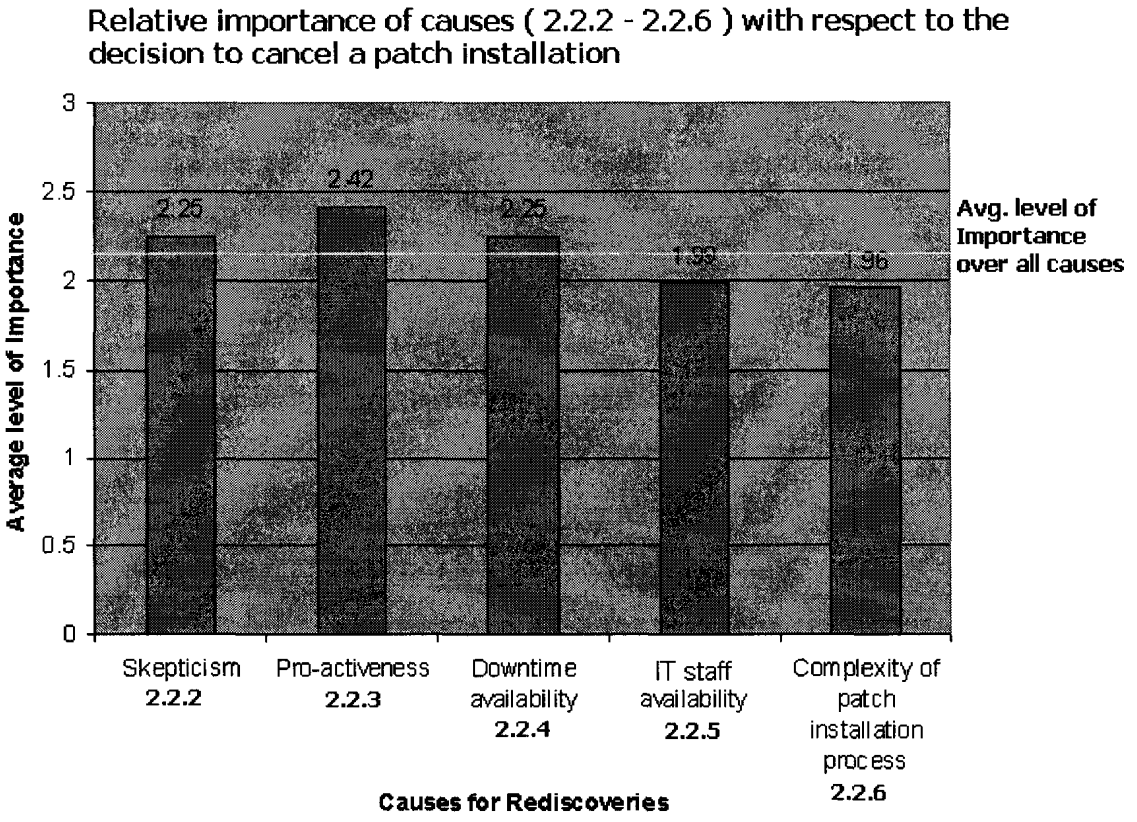


Figure 35 Relative importance of causes (2.2.2 – 2.2.6) w.r.t. the decision to cancel a patch installation

According to data in Figure 35 it can be observed that 'Skepticism', 'Pro-activeness' and 'Downtime availability' were given above average importance whereas 'IT staff availability' and 'Complexity of patch installation process' were given below average importance by system administrators when it comes to cancelling the installation of a patch. Also, from the feedback from the system administrators who participated in the study, it was apparent that the most important reason for cancelling the installation of a patch was when there was enough evidence to believe that their systems were not under any risk due to a defect in the software product which the patch was designed to fix.

5.2.2.5.8 Summary of results

In this section, we present the summary of the results of the case study done on software user.

- In overall, system administrators agreed more than they disagreed that their software providers provide them with timely notices about new defects and relevant patches.
- 58% of the system administrators said that they were pro-active in their patch management policy.
- 70% or more system administrators agreed that there was a delay in the patch installation on their production systems due to 'Skepticism', 'Downtime availability', 'IT staff availability' and 'Complexity of patch installation process'.
- More system administrators reported that there was a delay in the patch installation on their production systems due to 'Skepticism' and 'Downtime availability' when compared to the number of system administrators who reported that there was a delay due to 'IT staff availability' and 'Complexity of patch installation process'.
- More system administrators reported that the delay due to 'Skepticism' and 'Downtime Availability' was more than a week when compared to the number of system administrators who reported that there was a delay due to 'IT Staff availability' and 'Complexity of patch installation process'.

Chapter 6: Implications, Future work and Summary

In this chapter we conclude the thesis by discussing the implications of research findings, suggest some future work in this area, and lastly give a conclusion to the thesis.

6.1 Implications of Research Findings

The findings of our study can have potentially significant implications in the following areas of Software Engineering:

- Maintenance processes in the software industry.
- Customer Technical Support processes in the software industry.
- Development of framework for measuring the impact of rediscovery causes.
- Patch management policies of software users.

We now discuss each of these points in this order.

6.1.1 Maintenance processes in the software industry

The findings presented in section 5.2.1.7 show that 48% of the rediscoveries occur due to the delay on the part of the software provider to provide the patch to the software user. Also, delay in defect diagnosis causes 1.1%[3] of the rediscoveries and delay in fix release causes 7.1%[4] of them. However, the majority of rediscoveries, i.e., 39.8%[5], are caused due to delay in defect fixing. Hence, although diagnosis technologies and preventive maintenance policy have the potential to reduce some of the rediscoveries, i.e., 8.2% in total, to bring down the number of rediscoveries significantly the delay in defect fixing needs to be reduced by a considerable amount. This can be achieved by various measures like employing more developers, training, adopting better development processes, etc.

[3] 2.3% of rediscoveries caused due to delay on the software providers' side to provide the patch.
[4] 14.8% of rediscoveries caused due to delay on the software providers' side to provide the patch.
[5] 82.9% of rediscoveries caused due to delay on the software providers' side to provide the patch.

6.1.2 Customer Technical Support processes in the software industry

The findings presented in section 5.2.2.5.8 show that 'Skepticism' and 'Downtime availability' are the causes which are more significant than others on the software users' side which cause rediscoveries. These causes need to be addressed to reduce the delay on the software users' side in installing the software patch once it is made available by the software provider. Some of the potential solutions to address 'Skepticism' have been mentioned in chapter 2.

6.1.3 Development of framework for measuring the impact of rediscovery causes

The measurement of the impact of various rediscovery causes can be helpful to cost-effectively address the rediscovery causes to reduce the overall cost due to rediscoveries. The rediscovery cause taxonomy presented in chapter 4 together with research methodology GQM presented in section 5.1.2 can be used as a basic framework to measure the impact of various rediscovery causes. Although, this basic framework has been successfully used by us in our case studies, it is necessary to enhance the framework to make it more usable by appropriate tool support.

6.1.4 Patch management policies of software users

The findings presented in section 5.2.1.7 show that 52% of the rediscoveries occur due to the delay on the part of the software user to install the patch once it is made available by the software provider. This information can be useful to the software users to evaluate the impact of their patch management policies on the overall rediscovery cost. This evaluation will help them further to cost-effectively address the problem due to rediscoveries by making appropriate changes in their patch management policies.

6.2 Future Work

There are several suggestions listed below for future work that arose while conducting the study, they are:

- The replication of the case studies. The case study to achieve research objectives on the software provider's side was done on a single software product. However, currently we have a very diverse population of software

products in existence and hence drawing long reaching conclusions from a single case study may not be prudent. Hence, more case studies in this regard can be beneficial. Also, the case study to achieve objectives on the software user's side involved 100 system administrators whereas the actual population of system administrators in the IT industry is much larger. Also, the patch management policies of the system administrators may vary with the characteristics of the software product. An extended study involving more number of system administrators working with more diverse software products can be beneficial in this regard.

- The mentioned in section 6.1.2, the information provided to the system administrators by the software providers is not provided in a user-friendly way. The software users have to manually delve to extract the information relevant to them. If this process can be automated to some extent wherever possible it could prove to be very beneficial to the software users. More research in this direction is necessary.
- A generic framework with appropriate tool support to measure the impact of various causes for rediscoveries needs to be developed. This will very helpful to replicate the case studies like the ones presented in this thesis.

6.3 Conclusion

In this work, our research goal was to identify the various causes for software rediscoveries, create the taxonomy of these rediscovery causes and establish the significance of these causes. We identified the various causes for software rediscoveries from the literature and created taxonomy of these causes. To establish the significance of each of these causes for rediscoveries we undertook two case studies. Each step in our research process was inspected and validated, leading to results that we feel are sincere and valid.

Our findings suggest that the delay on the software providers' side to provide the software users the patch to fix a defect in the software product contributes to approximately 50% of the rediscoveries; whereas, the delay on the software users' side

to install that patch contributes to approximately 50% of the rediscoveries. On the software providers' side, the delay in diagnosis of the defect and the delay in packaging and releasing of the defect contribute to approximately 1% and 7% of the total rediscoveries respectively. The delay due to the design of the fix alone contributes to about approximately 40% of the rediscoveries (see section 5.2.1.7). On the software users' side skepticism about the patch causing issues with functionality and/or performance, and non-availability of system downtime were identified to be the causes with relatively higher significance than other causes, for software users to delay the installation of a patch which causes rediscoveries (see section 5.2.2.5.7).

These findings have potential significance in industry and research. Maintenance and customer technical support processes in the software industry and development of framework to measure impact of rediscovery causes can all potentially benefit from the results presented.

We suggested further research in related areas, and also that replication of this study be done to further strengthen the findings.

Glossary

Defect diagnosis: The process of analyzing of a failure to identify the defect [Lee Iyer 2000].

Issue Tracking System: An issue tracking system (also called trouble ticket system or incident ticket system) is a computer software package that manages and maintains lists of issues, as needed by an organization. Issue tracking systems are commonly used in an organization's customer support call center to create, update, and resolve reported customer issues, or even issues reported by that organization's other employees. An issue tracking system often also contains a knowledge base containing information on each customer, resolutions to common problems, and other such data. An issue tracking system is similar to a "bugtracker", and often, a software company will sell both, and some bugtrackers are capable of being used as an issue tracking system, and vice versa [Wiki].

Patch: A patch is a small piece of software designed to update or fix problems with a computer program or its supporting data. This includes fixing bugs, replacing graphics and improving the usability or performance [Wiki].

Patch Management: The process of controlling the deployment and maintenance of interim software releases into operational environments [NISCC 2006].

Security Patch: If a patch is a piece of data used to update a software product, then a security patch is a change applied to an asset to correct the weakness described by a vulnerability. This corrective action will prevent successful exploitation and remove or mitigate a threat's capability to exploit a specific vulnerability in an asset [Wiki].

Software Defect: A type of change request that identifies an anomaly or flaw in a work product [IBMT]; Any flaw or imperfection in a software work product [Florac 1992]. A software work product is any artefact created as part of the software process including computer programs, plans procedures, and associated document and data [CMU/SEI 1991].

Software Failure: The inability of a system or component to perform its required functions within specified performance requirements [IBMT]; Deviation of the delivered service from compliance with the specification [Laprie 1992].

Software defect discovery: A single software defect can cause multiple software failures. The very first failure due to a defect is called a software defect discovery [Adams 1982].

Software defect rediscovery: A single software defect can cause multiple software failures. The very first failure due to a defect is called a software defect discovery. All subsequent failures due to the defect are called software defect rediscoveries [Adams 1982].

Exploratory case study: A case study where we do not begin the study with a theory but instead conduct the study to develop a theory which may be tested by another study [Yin 1993].

Bibliography

[Adams 1984] Adams, E., "Optimizing Preventive Service of the Software Products", *IBM J. Research and Development,* vol. 28, no. 1, pp. 2-14, January 1984

[Altekar 2005] Altekar, G.; Bagrak, I.; Burstein, P.; Schultz, A., "OPUS: Online patches and updates for security", *Proceedings of 14th Conference on USENIX Security Symposium 2005*, pp. 19-19, August 2005

[Ballintin 2005] Ballintijn, G., "A case study report on the development, release and deployment processes of chipsoft", *Technical Report SEN-E0506, Centrum voor Wiskunde en Informatica (CWI), Amsterdam,The Netherlands*, April 2005

[Basili 1994] Basili, V.; Caldiera, G.; Rombach, H., "The goal question metric approach", *Encyclopedia of Software Engineering*. John Wiley: New York 1994; pp. 528–532, 1994

[Baumann 2004] Baumann, A.; Appavoo, J.; Da Silva, D.; Krieger, O.; Wisniewski, R., "Improving operating system availability with dynamic update", *Proceedings of the 1st Workshop on Operating System and Architectural Support for the On-Demand IT Infrastructure 2004 (OASIS 2004)*, pp. 21-27, October 2004

[Baumann 2005] Baumann, A.; Heiser, G.; Appavoo, J.; Da Silva, D.; Krieger, O.; Wisniewski, R. W.; Kerr, J.; "Providing dynamic update in an operating system", *Proceedings of the Annual Conference on USENIX Annual Technical Conference 2005,* pp. 32-32, April 2005

[Beattie 2002] Beattie, S.; Arnold, S.; Cowan, C.; Wagle, P.; Wright, C. "Timing the application of security patches for optimal uptime", *Proceedings of LISA '02: 16th Systems Administration Conference 2002*, pp. 233-242, 2002

[Brodie 2005] Brodie, M.; Sheng Ma; Lohman, G.; Mignet, L.; Modani, N.; Wilding, M.; Champlin, J.; Sohn, P., "Quickly Finding Known Software Problems via Automated Symptom Matching", *Proceedings of Second International Conference on Autonomic Computing, 2005(ICAC 2005)*, pp. 101-110, June 2005

[Campbell 1975] Campbell, D.; "Degrees of freedom and the case study", *Comparative Political Studies,* vol. 8, pp. 178-193

[Carmines 1991] Carmines, E. G. & Zeller, R.A. *"Reliability and validity assessment"*. Newbury Park: Sage Publications, 1991

[CMU/SEI 1991] Paulk, Mark C.; Curtis, Bill; Chrissis, Mary Beth; *Capability Maturity Model for Software (CMU/SEI-91-TR-24, ADA 240603)*, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 1991

[Cobb 1992] Cobb, P.; Lennon, C.; Long, K., "System and Method for Software Early Error Detection and Data Capture", *U.S. Patent no. 5119377*, 1992

[Cooke 1979] Cook, T. D., & Campbell, D. T. (1979). *Quasi-experimentation: Design and analysis issues for field settings.* Boston: Houghton Mifflin.

[Crameri 2007] Crameri, O.; Knezevic, N.; Kostic, D.; Bianchini, R.; Zwaenepoel, W., "Staged deployment in mirage, an integrated software upgrade testing and distribution system", *SIGOPS Operating Systems Review*, vol. 41, no.6 ,pp. 221-236, October 2007

[CSU] Colorado State University's: "Writing@CSU: Writing Guide", http://writing.colostate.edu/guides/research/content/

[Dungan 2004] Dunagan, J.; Roussev, R.; Daniels, B.; Johnson, A.; Verbowski, C.; Wang, Y., "Towards a self-managing software patching process using black-box persistent-state manifests", *Proceedings of International Conference on Autonomic Computing, 2004*, pp. 106-113, May 2004

[Florac 1992] Florac W.A., "Software quality measurement: a framework for counting problems and defects", *CMU/SEI-92-TR-22*, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 1992

[Gerace 2005] Gerace, T.; Cavusoglu, H., "The critical elements of patch management", *Proceedings of the 33rd Annual ACM SIGUCCS Conference on User Services (SIGUCCS 2005)*, pp. 98-101, November 2005

[Gkantsidis 2006] Gkantsidis, C.; Karagiannis, T.; Vojnovic, M., "Planet scale software updates", *SIGCOMM Computer Communication Review*, vol. 36, no. 4 ,pp. 423-434, August 2006

[IBMT] IBM Terminology
http://www-306.ibm.com/software/globalization/terminology/index.jsp

[Jansen 2005] Jansen, S., "Alleviating the release and deployment effort of product software by explicitly managing component knowledge", *Proceedings of the Workshop on Development and Deployment of Product Software. US Education Service 2005*, pp. 21–30, 2005

[Laprie 1992] Laprie J.C.; Avizienis A.; Kopetz H.; Kopetz H., "*Dependability: Basic Concepts and Terminology*", Springer-Verlag New York, Inc., 1992

[Lee Iyer 2000] Lee, I.; Iyer, R., "Diagnosing Rediscovered Problems Using Symptoms", *IEEE Transactions on Software Engineering*, vol. 26, no. 2, pp. 113-127, February 2000

[Lee McRee Bartlett 1996] Lee, I.; McRee, R.; Bartlett, W., "On-line recovery for rediscovered software problems", *Proceedings of IEEE International Computer Performance and Dependability Symposium 1996*, pp. 78-87, September 1996

[Lee Pitt Iyer 1996] Lee, I.; Pitt, G.; Iyer, R., "Efficient service of rediscovered software problems", *Proceedings of Annual Symposium on Fault Tolerant Computing 1996*, pp. 348-352, June 1996

[MBSA] Microsoft Baseline Security Analyzer, www.microsoft.com/technet/security/tools/mbsahome.mspx.

[Mullen Gokhale 2005] Mullen, R.; Gokhale, S., "Software defect rediscoveries: a discrete lognormal model", *Proceedings of 15th International Symposium on Software Reliability Engineering*, pp. 203-212, November 2005

[NISCC 2006] "Good Practice Guide - Patch Management", *National Infrastructure Security Co-ordination Centre*, October 2006 http://www.cpni.gov.uk/Docs/re-20061024-00719.pdf

[Pasala 2006] Pasala, A.; Rao, S.; Gunturu, S; Sinha, P., "An Approach Based on Modeling Dynamic Behavior of the System to Assess the Impact of COTS Upgrades", *Proceedings of 13th Asia Pacific Software Engineering Conference 2006. (APSEC 2006)*, pp. 19-26, December 2006

[Pasala 2008] Pasala, A.; Lew Yaw Fung, Y.L.H.; Akladios, F.; Appala Raju, G.; Gorthi, R., "Selection of Regression Test Suite to Validate Software Applications upon Deployment of Upgrades", *19th Australian Conference on Software Engineering, 2008 (ASWEC 2008)*, pp. 130-138, March 2008

[Rumsey] Rumsey D., "Intermediate Statistics for Dummies", *ISBN 0470045205*, Wiley Publishing, 2007.

[SME] European Commission (2003-05-06), *Recommendation 2003/361/EC: SME Definition*

[SMS] Microsoft SMS, www.microsoft.com/smserver

[TechSupport] PLM Technical Support Process – At a Glance, http://www-1.ibm.com/support/docview.wss?uid=swg27005882&aid=1

[Thornton Quema 2005] Thornton J.; Quema V., "Evaluating Patch Safety: Configuration Sharing for Problem Avoidance", *TR-05-2, Palo Alto Research Center*, February 2005

[Tiv] Tivoli, http://www.tivoli.com

[Wiki] Wikipedia, http://en.wikipedia.org

[Wood 2003] Wood A., "Software reliability from the customer view", *Computer* , vol. 36, no. 8, pp. 37-42, August 2003

[WU] Microsoft Windows Update http://windowsupdate.microsoft.com

[Yin 1993] Yin R., "Applications of Case Study Research", 1st edition, Sage Publications, 1993.

[Yin 2003] Yin R., "Case Study Research: Design and Methods", 3rd edition, *Applied social research methods series*, vol. 5, Sage Publications, 2003.

Appendices

Appendix A

Questionnaire: Case Study – Software User

aSCREENER1: Which title best describes your position?

1. Owner / president / CEO
2. C–Level officer
3. VP–level
4. Director–level
5. Manager
6. Team lead / supervisor
7. Team member
8. Contractor
9. Consultant
10. Intern or Co–op student
11. Student

aSCREENER2: What percentage of your time at work is spent in system administration tasks (installing/upgrading software, etc)?

1. None – My job does not include system administration tasks
2. 1-10 %
3. 11-25 %
4. 26-50 %
5. 51-100 %

aSCREENER3: How would you best describe your job function?

1. IT focused
2. IT and business focused, but more IT focused
3. IT and business focused, but more business focused
4. Business focused

aSCREENER4: What best describes the scope of the organization for which you are answering this questionnaire? Please answer all questions in the questionnaire for this organization.

1. Department
2. Business Unit
3. Enterprise

aDM1: What kind of software systems does your organization generally administer, or has administered in the past 24 month?

aDM1_A1: Middleware (Eg. Application servers – Websphere, Sun Java Application Server, WebLogic, JBOSS, etc) – Yes / No

aDM1_A2: Web servers (Eg. Apache, Microsoft IIS, etc) – Yes / No

aDM1_A3: Database Management Systems (Eg, DB2, MySQL, Oracle, etc) – Yes / No

aDM1_A4: Email servers (Eg. Microsoft Exchange Server, IBM Lotus Domino, etc) – Yes / No

aDM1_A5: CRM/ERP softwares (Eg. SAP, Oracle, etc) – Yes / No

aDM1_A6: Content / Data Management Applications (Eg. Alfresco, Apache Lenya, Joomla, etc) – Yes / No

aDM1_A7: System software (Eg. Operating systems, etc) – Yes / No

aDM1_A8: Other – Please mention here

cDM2: What best describes your organization's industry?

1. Education
2. Healthcare
3. Wholesale / Retail
4. Transport / Utilities / Communication
5. Manufacturing
6. Government
7. Financial Services
8. Business Services
9. Other – Please mention here

bDM3: What is your organization's requirements regarding availability of systems?

bDM3_1: Days per week – 1 to 7

bDM3_2: Hours per day – 1 to 24

bDM4: How many full-time equivalent employees work in your organization?

1. 1 – 25
2. 26 – 150
3. 151 – 800

4. 801 – 176,000

bDM5: How many full-time equivalent IT employees work in your organization?

1. 1 – 2
2. 3 – 7
3. 8 – 100
4. 101 – 50,000

bDM6: How many years of work–experience do you have in an IT related role?

1. 2 – 11 years
2. 12 – 14 years
3. 15 – 20 years
4. 21 – 40 years

aSP1a: Please indicate your agreement with the following statements on the following scale.

1. Strongly Disagree
2. Disagree
3. Somewhat Disagree
4. Neutral
5. Somewhat Agree
6. Agree
7. Strongly Agree

aSP1a_1: In general, your organization's software vendors provide you with timely notices about new defects and patch releases.
aSP1a_2: In general, your organization's software vendors provide you with relevant information regarding new defects to help you analyze whether your systems are under risk.
aSP1a_3: In practice, your organization would install a software patch even if you have not experienced a defect, which the patch is purported to fix.
aSP1a_4: Your organization has adequate automation to make patch installation a straightforward task.
aSP1a_5: In practice, your organization installs all the patches released by your software vendors.
aSP1a_6: Your organization has adequate IT staff resource to timely address all potential patch installations.

aSP1b: Do you have any comments or rationale about the previous statements?

aSP2a: What was the average delay in the installation of a patch on your systems in the last 24 months?

1. Less than 1 hour

2. More than 1 hour but less than 1 day
3. More than 1 day but less than 1 week
4. More than 1 week but less than 1 month
5. More than 1 month but less than 3 months
6. More than 3 months but less than 6 months
7. More than 6 months

aSP2b: Do you have any comments or rationale about the previous statement?

aSP3a: How important were the following factors to your organization's IT department's decision to delay a patch installation on the following scale?

1. Not Important
2. Slightly Important
3. Somewhat Important
4. Moderately Important
5. Important
6. Very Important
7. Extremely Important

aSP3a_1: There was a delay in testing the patch before installation on production systems.
aSP3a_2: The systems were functioning normally and we had not experienced any defect which the patch was known to fix.
aSP3a_3: There was a delay due to lack of available system downtime.
aSP3a_4: There was a delay due to lack of available IT staff to handle the patch installation process.
aSP3a_5: There was a delay due to lack of adequate automation or non–usage of patch management tools, to handle the patch installation.

aSP3b: Do you have any comments about the factors listed above that influence patch installation delay?

aSP4a: What was the average delay in patch installation on production systems at your organization in the last 24 months due to the following factors?
Options:

1. Less than 1 hour
2. More than 1 hour but less than 1 day
3. More than 1 day but less than 1 week
4. More than 1 week but less than 1 month
5. More than 1 month but less than 3 months
6. More than 3 months but less than 6 months
7. More than 6 months

aSP4a_1: Testing the patch before installation.
aSP4a_2: Lack of available system downtime.
aSP4a_3: Lack of available IT staff to install the patch.

aSP4a_4: Lack of adequate automation of the patch installation process.

aSP4b: Do you have any comments or rationale about the patch installation delay due to various factors?

aSP5a: What was the percentage of patches from your software vendors that your organization's IT department decided not to install, in the last 24 months?

1. None – We installed all patches from our software vendor
2. 1 – 10 %
3. 11 – 25 %
4. 26 – 50 %
5. 51 – 100 %

aSP5b: Do you have any comments or rationale about the previous statement?

aSP6a: Has your organization cancelled the installation of a patch from a software vendor in the last 24 months? – Yes / No

aSP6b: Do you have any comments or rationale about the previous statement?

aSP7a: How important were the following factors to your organization's IT department's decision to cancel a patch installation on the following scale?

1. Not Important
2. Slightly Important
3. Somewhat Important
4. Moderately Important
5. Important
6. Very Important
7. Extremely Important

aSP7a_1: We keep the changes made to our systems minimal.
aSP7a_2: The systems were functioning normally and we had not experienced any defect that the cancelled patch was known to fix.
aSP7a_3: The lack of available system downtime limited the number of patch installations.
aSP7a_4: The lack of available IT staff limited the number of patch installations.
aSP7a_5: The lack of adequate automation of the patch installation process or non–usage of patch management tools, limited the number of patch installations.

aSP7b: Do you have any comments or rationale about the previous statements?