DEVELOPING A SOFTWARE TOOL TO ENHANCE THE CREATIVITY DURING SOFTWARE DEVELOPMENT USING THE RESULTS FROM THE LITERATURE REVIEW

A Paper
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Reshma Hegde

In Partial Fulfillment of the Requirements
For the Degree of
MASTER OF SCIENCE

Major Department:
Computer Science

July 2013

Fargo, North Dakota

North Dakota State University
Graduate School

Title

Developing a Software Tool to enhance the Creativity during Software Development Using the Results from Systematic Literature Review

By

Reshma Hegde

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Dr. Gursimran Walia
Chair

Dr. Kendall E. Nygard

Dr. Simone Ludwig

Dr. Limin Zhang

Approved:

July 31, 2013
Date

Dr. Brian Slator
Department Chair

ABSTRACT

Success during software development depends on the creativity of software engineers. Knowledge plays a very important role in enhancing the creativity of software developers. Knowledge is available in different forms like repository knowledge (experiences of past projects) and community knowledge (gained through communication among software engineers). I have developed a tool which could help software engineers be more creative and successful. In order to develop this tool, a systematic literature review was undertaken to find how knowledge influences creativity and the key features required in the tool. The systematic literature review reports the various knowledge sources and how these can be accessed by developers to be more creative, and the methods used to access the knowledge sources. This paper describes the motivation for the tool, features of the tool and improvements for the next version of the tool.

ACKNOWLEDGEMENTS

I'm grateful to my adviser Dr. Gursimran Walia for his continuous help, support, patience and guidance in the development and completion of this paper. He has been a helpful advisor who has motivated me at every step and guided me during my master's program.

I'm grateful to Dr. Kendall Nygard for always being helpful with various stages in the progress of my Master's program and for taking out the time to be a part of my supervisory committee.

I'm grateful to Dr. Simone Ludwig for taking out the time to be a part of my supervisory committee.

I'm grateful to Dr. Limin Zhang for taking out the time to be a part of my supervisory committee.

I'm grateful to the Computer Science department faculty and staff in all the ways I could use their help in the progress and completion of my Master's program.

Finally, I'm grateful to my parents, sister and fiancé for being supportive all the way.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

Knowledge plays a significant role in the creativity of software engineers, and the creativity of software engineers defines the success of projects. This paper presents the development of a software tool which would promote the creative thinking process among software engineers. To identify the list of creativity-enhancing features for this tool, it was necessary for us to understand the way knowledge influences the creativity of software engineers. To understand the influence of knowledge over creativity of software engineers, a systematic literature review was conducted. The results of the systematic literature review are the foundation for the tool development and are discussed in the later sections.

Creativity refers to "the ability to produce new and original ideas and things" [1]. Graham Wallas described the stages of the creative process as the *preparation stage*; where data is collected and an attempt is made to solve a problem; the *incubation stage*; where the brain works unconsciously on the problem with the knowledge of the individual; the *illumination stage*; where ideas start emerging and links are established between known facts; and the *verification stage*; where ideas are picked up individually and worked on so that it can be communicated to others [8]. The creative process is described as rapid idea generation phase followed by periods of incubation and reflection before these ideas are refined or new ideas are created [16].

Software systems are developed by a team of people with complementary skills. Software development is conceptually a complex, knowledge intensive and cognitive activity. Effective software development relies on the knowledge collaboration and on the creativity of software developers [9]. Creative thinking requires the ability to integrate internal (stored in head) and external knowledge (stored across different artifacts and developers) for performing a task.

Creative thinkers search for new ideas by manipulating existing knowledge to see different problems, opportunities and solutions [4]. On that end, previous researchers have conducted studies to analyze the role of various forms of knowledge (prior knowledge, analogies etc.) on the creativity during software development. This includes case studies [6, 13], experiments [12, 7], surveys and interviews [10] with students and professionals from various domains. Their research has found that knowledge plays a positive role and helps professionals to be more creative in their work. Knowledge in different forms helps software engineers to come up with as many ideas as possible to solve a problem. The results show that prior knowledge helps engineers see the previous solutions and get new ideas and inspiration for the current problem [5].

Further, knowledge in the form of examples helps engineers with a thorough understanding of the problem and the kinds of solutions that already exist in the market. This helps engineers to add innovation to their solution [10]. Similarly, knowledge in the form of analogies helps engineers to see the problem from different viewpoints. This helps the engineers generate more ideas to solve their problem. An experiment was conducted to prove that designers came up with more ideas with the help of analogies as compared to designers without any help from analogies [7]. Similarly, sharing of knowledge among peer engineers makes the engineers see a problem in different perspective thereby triggering creative ideas. An experiment was conducted with 50 students of software engineering and social worker degree. In the experiment students were asked to create an innovative product or service from an existing object. Students were provided with tools to share their ideas and connect with other peers. The results showed that students who shared their knowledge generated more ideas for the innovative product [17].

While the above results provide evidence that using "knowledge" can improve creativity, a subset of previous research [5,7,10,17] narrowed down the focus on different forms of knowledge (e.g., prior knowledge or analogies) that had an impact on the creativity of software engineers. It was found that information was scattered among different papers and each provided detailed information on only one form of knowledge, how that knowledge type can be accessed to enhance creativity, and what problems are faced when accessing that information [5]. Hence, there was a necessity to collect all this scattered information and organize them in a single study to be able to better understand the effect of knowledge (in different forms) on creativity of software engineers. To realize this, a systematic literature review was conducted.

A systematic literature review is a formalized, repeatable process in which researchers systematically search a body of literature to document the state of knowledge on a particular subject. The benefit of performing a systematic review, as opposed to using the more common ad hoc approach, is that it provides the researchers with more confidence that they have located as much relevant information as possible. This approach is more commonly used in other fields such as medicine to document high-level conclusions that can be drawn from a series of detailed studies [3]. To be effective, a systematic review must be driven by an overall goal. The goal of this research is to: *Identify all the sources and forms of knowledge which helps improve creativity among software engineers, the various ways that the knowledge is accessed, and the limitations in the way that they are accessed*.

The remainder of the paper is structured as follows: Chapter 2 presents related background work and motivations for undertaking a systematic literature review. The methods of the systematic literature review are detailed in Chapter 3. Chapter 4 reports all the findings of the literature survey, and Chapter 5 discusses the results of the literature review in light of the

developing the software tool for enhancing the creativity of software engineers. Chapter 6 explains the tool and its features. Finally, Chapter 7 has the conclusion and future enhancements to the tool.

2. BACKGROUND AND RELATED WORK

This section outlines the motivation for conducting a systematic literature review and describes relevant background work to help provide context for the research presented in the remaining sections of this document.

2.1. Motivation

Many software companies these days want their employees to be as creative as possible and come up with new innovations. Creativity among software engineers has become very important for the long-term success of the company. My Master's paper in an effort to help software engineers by providing them with a tool which could enhance their creativity and make them more successful. In order to understand what triggers or enhances creativity among software engineers, an ad hoc literature review was conducted. During the ad hoc literature review, it was mentioned that among several factors (i.e., intellectual abilities, knowledge, styles of thinking, personality, motivation and environment in which the engineer worked) that drove creativity among software engineers, ***knowledge*** has the most impact [13].

Software engineering is a knowledge intensive and collaborative task. Its success depends on the collaborative knowledge and skills of the software engineers involved in the process (as opposed to their individual knowledge) [9]. Software engineering knowledge is scattered across different resources (e.g., artifacts, code, documents, peers, lessons learned etc.) and continually flows through the entire software development ecosystem. From this perspective, knowledge plays an important role in triggering the creativity of software engineers. This paper reports the results from the systematic literature review that was undertaken to better understand the influence of knowledge on the creativity in software engineering and to use those insights during the tool development. The main motivation was to help software engineers understand the

importance of knowledge and properly utilize the knowledge to have a creative edge. This work also intended to help software engineers understand the benefits of different forms of knowledge (e.g., prior knowledge, examples and analogies) and include them in their day to day software development activities, and to help motivate the software developers to store and share their experiences with others as it can help their peers generate creative ideas to solve their problems.

2.2. Related work on role of knowledge in creativity

The idea of examining knowledge to investigate its effect on the creativity of software engineers is not new. I found various sources during the ad-hoc literature review that concentrated on knowledge and creativity, but almost all the sources concentrated on a different type of knowledge [5, 7, and 10]. A brief review of these different knowledge sources and its impact on the creativity is discussed follows.

Prior knowledge is a collection of information about the previous solutions for a problem. It can be sketches, diagrams, documents collected while working on a problem. Studies have shown that prior design knowledge is highly valued and useful during early phases of design. Generating new ideas requires background research and designers stated that they usually look at other designers work on similar projects or their own previous work to get inspiration. Designers used the artifacts from the previous projects, coupled with their previous experiences for generating and evaluating solution possibilities. Despite knowing that prior knowledge helps in generating new ideas, people are not able to use if effectively because they are not able to access the prior knowledge. Searching is very cumbersome as the volume of data is very large. Many things are missing in the prior knowledge, such as the reason for choosing one design over another [5]. Our literature survey identified these limitations and provides suggestions to overcome these limitations and allow knowledge access to be seamless and help creativity.

Another mechanism that can contribute to the emergence of new ideas is ***analogy making.*** This means new ideas can be inspired by previous situations and objects which may or may not belong to the same application domain. This was evaluated through an experiment that had ten art designers design a new kind of chair, and they were given some sources to make analogies. It was seen that the group which was guided with sources to make analogies came up with more ideas than the unguided group. This study showcased that analogies helps in being creative [7]. Another study showed that examples were very useful in the early design as they help in the idea generation phase. They inspire new ideas and make the designer creative. The example helps the designers understand what is already available in market and hence provides them with an opportunity to add new features that do not already exist and be creative with their design. Designers search for examples in magazines, books, and the internet. Examples are stored in the form of photo copies, clippings of magazines, digital images, web blogs, etc. [10].

There are various tools like CAD which provide dynamic knowledge to engineers on the fly and help them be more creative. Tools store knowledge within them and help users apply this knowledge practically to solve problems. In this way, users will spend all their effort in solving the problem as the tool takes care of providing the right information as needed. Analysis of the studies of knowledge and creativity can help us understand the role played by knowledge in triggering the creative ideas among software engineers.

3. SYSTEMATIC LITERATURE REVIEW

This section describes the process used for performing a systematic literature review of the role played by knowledge in enhancing the creativity of software professionals. This includes a description of the review protocol, which describes the high-level research questions, the sources to be included in the literature review, various criteria used for conducting the study, and the data that was extracted from each research paper included in the review.

3.1. Research approach

The systematic review is based on guidelines established by Kitchenham in "Procedures for Undertaking Systematic Reviews" [2, 3]. The purpose of performing a systematic literature review is similar to that of performing any scientific experiment. Procedures are established, followed, and reported on so that other researchers are capable of replicating the work. Following a systematic review process also provides a high degree of control over the type and quality of reference works that will be included in the review, and helps to provide support for the conclusions of the literature review.

In accordance with the guidelines for a systematic literature review established by Kitchenham [2], the following steps were implemented:

1. Formulate a review protocol.
2. Execute the review based on the established protocol (identify the primary studies, evaluate those studies, extract and synthesize data from those studies).
3. Analyze the results of the review.
4. Disseminate the results of the review.
5. Discuss the findings of the review.

The review protocol specifies the research questions to be addressed, establishes a list of databases, conference proceedings, journals, etc. from which primary sources will be selected, and establishes criteria for including sources and evaluating their quality. The subsequent steps closely mirror those of any other experiment in that the protocol is executed, the results of the review are analyzed to address the research questions, and the results are presented and discussed.

3.2. Research questions

A high-level research question ("What is the role played by knowledge in creativity of professionals in software development organization?") was decomposed into three more specific research questions and related sub-questions. A list of these research questions and the motivation for those questions is available in Table 1.

Table 1. Research Questions and Motivation

Research Question	Motivation
RQ1. Is there any evidence that knowledge influences creativity in software engineering? **RQ1.1. If knowledge does influence creativity, in which form does it influence?**	Determine the extent to which the knowledge influences creativity in software engineering.
RQ2. Is there any evidence that the knowledge access methods identified in literature enhances the creativity of software professionals? **RQ2.1. Which are the knowledge access methods that influence creativity in software engineering and what are the limitations of these knowledge access methods?**	Determine if there are any evidences on the knowledge access methods which prove that they improve the creativity of software professionals and also determine the limitations of those methods.
RQ3. What are the ways to improve the current knowledge access methods so that it can support the creative thinking process of software professional?	Determine the ways to improve the current knowledge access methods.

Research question 1 gathers and analyzes the evidence from the literature to investigate the extent to which the knowledge has an effect on creativity.

Research question 2 gathers and analyzes different ways in which knowledge is accessed and utilized to be creative. It also gathers all the shortcomings of the knowledge access methods.

Research question 3, a meta-question, gathers additional evidence along with the information from question 1 and 2 to suggest improvements of the knowledge access methods to support creativity in software development.

3.3. Source selection and search

Initially, an ad hoc review was performed in order to assist with the development of search strings and to provide a list of potential conference proceedings and journals to be manually searched. Using the results of the ad hoc review as a guideline, selection criteria were developed to establish a list of initial databases to be searched and more relevant conference proceedings or journals to be searched manually. References from primary sources were also included if they were relevant.

Before conducting the search, all the databases and conference to be searched must be listed to ensure that I find the most complete and relevant set of primary study materials. The source list which were searched included these Databases: ACM Digital Library, IEEE Explorer, APA PsycINFO, ScienceDirect

The master search string mentioned below was used to search the above databases. These search strings were manipulated as required to suit the database to be searched.

((creative OR creativity OR imagination OR imaginative OR thought OR think OR thinking cognitive OR intuition OR memory) AND (software OR web OR package OR computer system OR product OR system OR software system) AND (design OR develop OR development OR engineering OR engineer OR test OR testing OR plan OR planning OR originate OR formulate OR technology))

3.4. Study inclusion and exclusion criteria

The database search resulted in an extensive list of papers, some of which were clearly not related to the research questions. To narrow down the results from the search, a set of inclusion and exclusion criteria were developed in order to assist the selection of appropriate papers to be included in the literature review. These criteria were applied in multiple steps, starting with using the title to exclude papers not related to our research focus, then proceeding based on the papers' abstracts, and finally concluding with the papers' contents in their entirety. A list of the criteria used can be found in Table 2 and is discussed in this section.

Table 2. Study Inclusion and Exclusion Criteria

Inclusion Criteria	Exclusion Criteria
1. Publications that are related to knowledge and creativity in software engineering.	Publications that are not in English.
2. Publications that reported the results on creativity in Software Engineering.	Publications that were short or mini reports.
3. Publications that contained empirical results.	Publications that were not related to any of our research questions.
4. Publications that are related to creativity and knowledge which were not software engineering, but could be applied to software engineering.	Publications that contain unclear or ambiguous results.

This search resulted in a list of 6696 potential papers from which only most relevant had to be chosen. A lot of papers were removed from the list based on irrelevant titles and abstracts. After applying this process, 43 papers remained. Each of these 43 papers was read in entirety. A set of inclusion /exclusion rules were created so that only the most relevant studies were considered for the systematic review (as shown in Table 2).

By applying these rules on each of the 43 papers, only 16 remained. The included papers were published in the proceedings of the ACM conferences of Creativity and Cognition, Human

Factors in Computing System, Foundations of software engineering, WikiSym, IEEE international conference of Industrial engineering and engineering management, Technological management for the global future conference, Science direct Design studies, ACM Computers and Education and IEEE engineering management journals.

3.5. Data extraction and synthesis

Once all of the papers had undergone the filtering based on inclusion and exclusion criteria, data extraction was performed on all papers. A data extraction form was developed to ensure consistent extraction across all papers. Table 3 lists the fields of the data extraction form as well as a description of each field.

Table 3. Data Extraction Form

Data	Description
Reference information	The complete title of the paper with authors.
Main research question	The list of research question that the papers provide answers for.
Motivated by previous study	The previous study from which the study was inspired, if any.
Type of study	The type of study (e.g., survey, controlled experiment etc.).
Findings	The main findings of the paper
Validity threats	Threats in the findings or study techniques.
Why is the paper interesting?	Highlights of the paper/most interesting finding or technique.
Knowledge management technique	The knowledge management or access technique discussed in the paper.
Improvement	Improvements for the current study if applicable.
Empirical evidence	The empirical evidence for the research question(s).
Relation between creativity and knowledge	Any information that indicates that knowledge improves or influences creativity.

4. REPORTING THE REVIEW

This section reports the results organized by each research question and sub questions using the information extracted and synthesized from all the papers.

4.1. Question 1: Is there any evidence that knowledge influences creativity in software engineering?

A review of the literature indicates that knowledge has an important relationship with the creativity during software construction. Software engineering knowledge (available in various forms) helps software engineers come up with larger number of solutions to their problems (some of which may be original). While most of the empirical studies focused on the relationship between design knowledge and the creativity, the relationship and the resulting effects can be applied to any development stage since each stage involves knowledge transfer of some sort. Furthermore, this section also includes the previous studies from Psychology and Human Cognition, whose findings and results can be generalized to the software engineering domain.

The review identified many forms of knowledge that can influence the creativity of software professionals. The main forms of knowledge that were reported in literature include:

1. ***Prior knowledge*** – Includes useful information (ideas that could be extracted from design documents, magazines, sketches, notes, lessons learnt etc.) from previous projects. The previous projects could be one's own projects or other's projects.
2. ***Analogies*** – The entities which look similar to the entity under construction. This could belong from the same domain or other domains.
3. ***Dynamic knowledge in tools*** – Pertains to the knowledge that is embedded into the tool and is available to the engineers at any time. The user need not spend time remembering or searching for this knowledge. as it is already stored and available in the tools; he/she

can concentrate on the actual problem and get the information from the tool. This is useful information about the current problem which is given when required to the user or when user demands for it.

A brief discussion of these knowledge forms and their effect on individual developers' creativity follows:

Cognitive Psychology has often cited the use of the ***prior knowledge*** as an important ingredient to advance the human cognitive ability while trying to create novel solutions. Humans are able to use the prior information (e.g., examples of other's work) to learn and expand the prior knowledge to produce new ideas and solutions. Likewise, prior knowledge is one of the commonly reported forms of knowledge that helped software engineers to generate more new ideas or be creative. Specifically, software engineers during the early phase of design heavily depend on the prior knowledge (form their own and their peer's prior work) in order to produce new ideas. Prior knowledge represents conceptual ideas, lessons and representations captured or collected when solving a problem Bailey et al., reported the results from interview with 14 design engineers and an online survey with 17 questions collecting responses from 28 additional participants. Participants were from various expert domains like industrial graphics, interaction design, and mechanical design. All the participants had at least five years of professional experience. The results from interview and survey showed that the reuse of prior knowledge is highly valued and the most important step during early phases of design.

In early phases, people are still looking for directions rather than actual solutions and analysis of the prior work provides a starting point or direction for new solutions. They provide inspiration to new and creative ideas. A few examples of prior knowledge can be well-defined solutions to recurring problems, case-based solutions, design decisions and the reasons behind

those decisions, and the design history for a particular item. Results collected from the study showed that designers want to use theirs and other designer's artifacts and reflect on the lessons, decisions, and overall process to generate new ideas. Generating new ideas to solve a particular problem is an important step, and generating new ideas without any help is difficult and designers tended to access previous work or project to get inspiration. The study also showed that designers stored artifacts like notes and sketches from their current project, as they were sure that they will need them in their future projects; however designers mostly prefer to analyze other designer's work more than their own past work to gain new ideas and inspiring solutions [5].

Examples are another source of prior knowledge that is helpful in enhancing the creativity of software engineers. Examples are any kind of material, product, prototype, or digital artifact that contributes directly or indirectly to the design. Examples can come from the designer's own work or external sources like other designer's work, web, blogs, magazines, etc. Herring et al. conducted an interview with eleven designers to understand the use of examples in creative design. The results from the interview pointed out that examples helped designers in the idea generation phase. Examples also helped the designers to understand the problem better and helped them compare similar products to see the novelty in their idea (so that it is not just an extension of the previous work). During the evaluation phase, examples were used as references to see how their design has evolved from where they started [10]. Based on these results, prior knowledge plays a very important role in the initial phases of software designs. It provides a starting point and inspiration for generating new ideas for a particular problem.

Analogies are another form of knowledge which helped designers to come up with creative and innovative solutions by analyzing the design solution from a different domain.

Analogies help compare two similar entities, find similar feature and integrate them. This can lead to creative solutions. As pointed out by Johan Hoorn, "Creativity is to put two familiar entities in an unusual combination," which explains that people come up with creative solutions when they combine similar entities together, compare them, and look closely to their intersecting features and integrate them. He explains the creative process as entities being associated, selected, integrated and adapted until an optimal solution is found [11].

Nathalie Bonnard reported the results from two experiments that show that creativity during the design phase can be improved with the help of analogies. In the first experiment, designers were asked to design a chair for cyber café. This was conducted with 10 applied arts students. Designers were provided with sources to make analogies. The two analogies provided belonged to the same domain and were nomadic stool and rocking chair. The other two were from a different domain and were canoe and logotype. The results showed that the group that used analogies was able to draw features from many more analogies than the unguided group, who did not have sources to make analogies. The guided group came up with 26 sources to make analogies, whereas the unguided group came up with 6. The more number of sources they came up with, the more ideas they were able to generate [7].

During the second experiment, professional designers and students were asked to design a renewed torch. Out of six volunteers that participated in the experiment, two were professional designers and four were design students of applied arts. They were asked to find a new gesture to use the torch and move away from the traditional switch. Very minimal information about the problem was given to the participants, but additional information or analogies were provided upon request. The results showed that professional designers (and more experienced) asked more questions to understand more details about the problem in hand compared to the students, and the

questions were targeting the audience who will be using the torch. This helped them understand the problem better and visualize the problem from various perspectives. Professional designers applied these various viewpoints into their problem solving at an early stage. If it was discovered later, then it would be impossible to adapt into the design. Overall, these results showed that analogies can help designers to extract more details about the problem in hand and in coming up with more ideas to solve the given problem [7].

The engineer's knowledge base consists mainly of scientific principles, formulae, and rules. The scientific principles are linked to the real world, where creative products will be used. This linking or applying the principles to the real world is done by the engineer. If we want to enhance the creativity and creative output of an engineer, we should be able to build the capability of applying the scientific principles to the real problems. The tool should be capable of using the scientific rules to narrow down a search of problem space with well-defined reasoning. The information technology (IT) tools can gather this knowledge and devise rules and make it available to the engineers during their work of solving problems. This is called the ***dynamic knowledge***, which is available in the tool and can help designers concentrate completely on the solution.

Thomas A. Kappel and Albert Rubenstein have given various examples to demonstrate that IT helps in creativity. Example of tools like CAD helps designers analyze and design solutions. It helps in rapid prototyping. This benefits not only engineers but even manufacturing and sales. Simulation tools help engineers run their design in its environment and predict how it behaves, identify any shortcomings and fix them. This helps the engineer in trying different things and be creative. Expert systems are another tool that can interact with the engineer. It encourages an engineer to come with alternative designs by making suggestion to his or her

existing design, providing some useful information from engineer's previous designs or works, evaluating the design against requirements etc. Tools with knowledge built into them can help engineers be creative [14].

4.2. Question 2: Is there any evidence that the knowledge access methods identified in literature enhances the creativity of software professionals?

The literature review provided evidence regarding the knowledge access methods that provide the users with the knowledge they need and help them produce creative solutions to their problems.

The review identified two major categories of knowledge access that can influence creativity and are listed below.

1. ***Repository*** – Repository is a centralized place where all prior knowledge is stored. This knowledge can be pulled by users as needed or can be provided to the users dynamically with the help of tools, as they work on problems. In this category, searching relevant information is very challenging as there is huge amount of data and search mechanisms offered by repositories are not very efficient and is often the reason that users avoid using repositories. The search mechanism used in the repository is usually a basic search offered by the tool in which the knowledge is stored or provided by the operating system. These simple search mechanisms are not efficient at finding the relevant information thereby inhibiting the creativity of software engineers.
2. ***Collaboration*** – Software engineers exchange and share their ideas and information with other engineers in an organization. Knowledge and idea sharing helps software engineers to explore areas which they are otherwise unaware of, and helps them to see different viewpoints, opens a new channel of thought, and provides feedback to improve on one's

idea and knowledge. The main problem during the collaboration is finding the right people and tools to collaborate and share knowledge with.

Knowledge sharing through collaboration was the most commonly reported method that software engineers use to access the knowledge which in turn triggers creative ideas and their development. Ardiaz et al. performed a study with students using the tools named ***Wiki ideas*** and ***Creativity Connector*** for knowledge sharing and collaboration. Wiki ideas are a web based tool where users can share ideas and conduct brain storm sessions. The tool stores the complete history of how the idea has evolved. Creativity connector is a social networking tool which allows people from same and different backgrounds and skill sets to participate in brain storm sessions. These tools allow large number of users to participate in brainstorming sessions to share their knowledge and provide feedback which can result in greater number of ideas. The feedback that the users get when they share their thoughts with other users helps them refine their ideas. The fifty participating subjects in the study were drawn from a public university of Navarra and were enrolled in social worker and software engineering degrees. The students were divided into two equal groups, one composed of people from similar backgrounds and the other group had people from different backgrounds based on the degrees that students were enrolled in. The goal for the groups was to create an innovative product or service from an existing object. The preliminary results from the study showed that these tools helped users to generate a large number of ideas by sharing knowledge and through the feedback provided by other users. It also provided user interface to enable users to see the ideas online and provide feedback to them. The results also showed that the heterogeneous group produced greater number of ideas as compared to homogeneous group [17]. Based on these results, collaboration between peers (irrespective of the peers' background) can help software engineers to be more creative and

generate a larger number of ideas. Therefore, it is deemed important for software engineers to collaborate with other knowledgeable peers and exchange feedback and suggestions.

YunWen Ye further explains that knowledge collaboration in software development happens in two axes, namely the technological axis and the social axis. In the ***technological axis***, tools like books, manuals and repositories are used to enhance or add on to the insufficient knowledge in the heads of software developers. In the ***social axis***, collaboration happens with other knowledgeable peers to share their proposed ideas and improve their solutions. Engineers collaborate to solve a problem by filling in the missing knowledge in each other's head. When other knowledgeable peers review the ideas from different perspectives, multiple ideas can emerge or the existing ideas could evolve.

Computers play two roles to support knowledge collaboration: a) they act like repositories and provide useful information to the developers on their work, and b) they provide a platform for software developers to share new ideas and solutions [9]. Kao et al. conducted an experiment to show that people came up with new ideas and understood concepts better when they share their ideas and concepts with others, as opposed to working alone. The experiment was conducted with thirty two students of a computer hardware class. During the experiment, students initially formed their concept maps (which consisted of the concepts and links between different concepts that they had learnt in the class). The concept map represented how well they understood the concepts in the class. Students exchanged these maps with other students, reviewed them and noted down any information (or creative cross links) that was missing in their maps. Next, the students reworked their concept maps by incorporating the information gained from other concept maps. They either added or deleted some concepts in their maps. The results from the experiment showed that the revised maps were better than the original ones in the sense

that individual students might aim only at certain points/details/areas to such an extent that they might miss out few important details. Furthermore, when individuals reviewed and shared their concepts and ideas with other students, it gave them a new perception to concepts and pushed them beyond their traditional concept maps and helped them discover ideas which they could not find on their own [12]. Based on these results, during collaboration software engineers explore the unaware zone and generate more creative ideas compared to what they would have done when working alone.

Tarik Baykara also reports that among several factors that can drive creativity among the people in a software organization (e.g., intellectual abilities, knowledge, thinking styles, personality, and motivation), the ability to provide a dynamic work environment for the engineers to share their ideas and knowledge is most relevant factor. He came to this conclusion by performing qualitative evaluation and through assessment of creativity observed during all the project phases on ten completed research and development projects in Turkish's scientific and technological council. Through extensive interviews and questionnaires with project directors, key researchers and managers, it was concluded that knowledge sharing drives creativity in three types of software projects. ***Type A*** is idea driven which focuses on new idea creation. While, this heavily depends on individual knowledge, the ability to generate new and more ideas are important and require creative thinking. ***Type B*** is product driven that needs creative design and prototyping and depends on the entire team pouring in creative ideas in timely fashion. ***Type C*** is problem driven where creation of new ideas are needed to solve a problem in critical situation and requires both individual and team effort. The results from the case study showed that there is a strong correlation between knowledge sharing and the creativity in all three project types.

Based on these results, it is important for an organization to support and promote knowledge sharing between their employees to allow them be more creative [13].

Searching and finding knowledge in repositories, tools, internet, books and magazines is another way of accessing knowledge and being creative. Edmonds et. al., reports the results from a retrospective study on Mike burrow's bicycle design which showed that for a designer to be creative, he should have access to diverse and dynamic knowledge. Mike Burrow followed a. Design b. Make c Use and d. iterate approach. He designed the bicycle, built the model, used the bicycle. He iterated through these steps till he got a perfect bicycle. Based on the results from the study, it was concluded that dynamic knowledge should be made available to the designer depending on the problem he/she is facing.

The authors proposed a system called KSS (Knowledge Support System) that provides the knowledge or information to the designers during the design process and provides the ability to the designer to be able to make changes or improvements to the knowledge as the design progresses. The KSS also includes the design history in the form of predecessor designs, design rules and knowledge editors, and provides access to all the information and expertise. This system was proposed after concluding that embedding knowledge into the tools can provide knowledge dynamically to the user [6].

Similar to the above study, Bales et al. conducted an observational study to see how designers organize tangible information in their work space and how they utilize this information in their work. The results from the study showed that the prior knowledge and the domain knowledge are keys to creative and successful design. They also observed the designers individual and collaborative work spaces and showed that the tangible information is moved between these workspaces as design evolves and refines. Based on these results, the authors

concluded that, for any tool to support creativity, it has to provide multiple workspaces and ability to move the information between these work spaces to share their ideas with others and get feedback to refine their solutions [15].

On similar lines Joshua et al., conducted a study to see how software tools can help the designers in being creative. The study was conducted with six groups of three designers each. The participating subjects were drawn from UI design and architecture domain. The tool used in the study provided personal and collaborative workspaces for sharing ideas and work. People were asked to work on individual ideas, and then move it to the collaboration space and work collaboratively on the idea. The tool also allowed them to work on multiple ideas in parallel by providing multiple work spaces. The results of the study showed that the designers liked using this tool as it helped them have multiple solutions for the problem and also work on them in parallel. The tool also helped the teams share their ideas with others, get feedback and work in collaboration which led to creative solutions [16].

Based on these results, there is ample evidence to show that the knowledge access and sharing methods help improve the creativity of software engineers, especially during the software design. However, there were few limitations with the knowledge access method that inhibits the creative power of software engineers. For example, when using the repositories, designers need to search for prior knowledge which was very cumbersome due to plethora of information stored in them. It is challenging to finding the right information efficiently. Software engineers also expect to have a visual search to help them locate the right information easily. Knowledge users also felt that the information was stored in an inconsistent manner and wanted some kind of consistency so that they can easily make their way to the right information they are looking for [5]. Similarly, during the knowledge collaboration, the most difficult problem is to

find the right people to collaborate with. It is mostly seen that people are not always aware of whom to contact to get help when they are in problem or get feedback from. At times when people find the right people to collaborate with, others may be busy with their work and would not completely participate in collaboration or they might also completely deny from participating [9]. Therefore, it is important for an organization to create work environment for the engineers to be able to share their ideas and knowledge and collaborate in order to support creative software development. Searching seemed to be a big problem which could be preventing users from accessing the repository. Hence in my tool I have tried to provide information in different views and tried to reduce search to an extent. Data is provided to user in an easy way to avoid making the user to search for it. Collaboration is also implemented in the tool to encourage knowledge sharing among the users of the tool.

4.3. Question 3: What are the ways to improve the current knowledge access methods so that it can support the creative thinking process of software professional?

During the systematic review, I came across multiple studies to suggest that the knowledge access methods need to be improved for supporting creativity [5, 4, 9, and 18]. Multiple studies reported knowledge stored in repositories as one of the main sources of knowledge used by software engineers to solve problems. While prior knowledge gained from repositories is very useful, it is not used effectively as searching is very difficult and in few occasions important information from the data is missing from these repositories. The search mechanisms used in existing repositories are not very powerful thereby making searching for the right things extremely difficult. This is the reason why software engineers seldom use prior knowledge stored in repositories.

One way to improve this could be to make search more powerful with introducing visual search. In this kind of search, a user can visually see the data through which he is searching, can modify the search based on his needs, and can see how the data is organized so that he/she can search the data in an effective way. Once the search provides the right information in an efficient manner, software engineers can start using the repositories to enhance their creative thought process. Regarding the incompleteness of the information stored in the repositories, it is important to collect the story behind the knowledge being stored. For example, when a design was chosen over alternate designs, storing the reason as to why it was chosen over others is a good approach. This would help the person looking at this knowledge better understand the original thought behind the decision and help relate it to his or her current problem [5].

Based on the results reported in section 4.1, ***analogies*** are an important form of knowledge which triggered creativity among software engineers. Maiden et al., [4] conducted a workshop to show that using analogies during the requirements engineering generated more creative ideas but had inherent limitations. The workshop targeted collecting requirements for a future air traffic management system for managing departures from major European airports. Twenty one professionals from the departure management and scheduling departments in the UK and France attended the workshop. One of the main lessons learned from this workshop was the realization that, in order to support the use of analogies during requirement stage, people need to be trained on how to use analogies and to be able to extract useful information from analogies to transfer knowledge from one domain to generate new ideas and use it in the targeted domain. Therefore, usage of analogies should be supported by providing relevant trainings so that people know how to extract knowledge from analogies and generate creative ideas. While this workshop

was focused on the requirements phase, the results are equally applicable at other phases of software development.

I also came across a proposal for tool that can provide access to knowledge and help in collaboration with knowledgeable peers. As the developer types the APIs, the tool would be able to provide suggestions and comments for the method signature. When the developer hovered over the method there would be information about what the method is used for and the purpose of the arguments. For more understanding, the developer could click on the link which would take him to the detailed explanation of the method with examples of code. If the developer wanted deeper understanding of the method the page also contained discussion forums where one could see what kind of problems people had and the answers to the questions. The developer could also post questions and reply to questions posted by other users in the forum [9]. This tool, if developed, would provide dynamic knowledge to software engineers on fly, help enhance their knowledge and would help them become more creative and efficient in their work. The idea behind the development of such tools is to be able to provide dynamic knowledge when needed to the software engineers, so that they can concentrate on the real problems rather than expecting them to remember all the pertinent information they would need to solve a particular problem.

Based on the results reported in section 4.2, sharing knowledge or collaboration is one of the common ways to promotes creativity. Andrawina [18], reports the results from a survey that organizational factors like top management support to knowledge sharing and technology factors like computer network and electronic bulletin boards, discussion groups to collaborate and share knowledge influence positively and encourage people to share knowledge and apply them to be innovative. The survey recommended that organizations should have management who will appreciate and reward knowledge sharing behaviors among their employees and provide

technology help required for knowledge collaboration. This would help organization and its people be innovative and creative [18].

Additionally, the results in section 4.2 also reported that the two major knowledge access methods were repository and collaboration. Both of these methods have their own advantages of usage. Repository approach is completely at the control of the user. Knowledge users can use it as and when they require it, whereas collaboration needs the other person's time and readiness to collaborate and share their knowledge. The data in the repository does not contain dynamic contextual data whereas the collaboration gives access to dynamic and contextual knowledge.

The best approach would be to integrate both types of access methods into a single system by providing access to both the repository knowledge and the contextual knowledge. Software tools that can combine both these approaches would act as single point of contact for a software developer to search for information when he/she wants feedback or help with his or her problem. Such tools would increase the knowledge access and in turn help software engineers be more creative [9]. I decided to use this tip and in the tool that I will be presenting in the coming section, I have combined the repository approach and collaboration approach of knowledge access to gain the benefits of both the access methods.

5. DISCUSSION OF FINDINGS FROM LITERATURE SURVEY

This section provides a summary of the principal findings of the systematic literature review discusses the strengths and weaknesses of the gathered evidence and relates how this information can be used by software professionals. These are the findings that helped me in developing the tool that could help software engineers be more creative. I have utilized most of the findings here to come up with the features of my tool.

5.1. Principal findings

The purpose of this literature review was to identify if knowledge improves the creativity of software professionals. To ascertain this information, a systematic literature review was designed and executed on popular publication databases, as well as a wide variety of journals and conference proceedings related to the topic area. The identified data were analyzed and categorized to explore trends in the data. The principal findings of this review are as follows:

I saw in the literature review that knowledge (available in different forms) helps software professionals to become more creative. Additionally, the knowledge types reported in literature that impacts the creativity of software engineers were broadly classified as: prior knowledge, analogies, and knowledge available in tools. Prior knowledge provided useful information about previous projects and helped software engineers in the initial stages of idea generation. They provided a starting point or motivation to start with ideas to solve the problem. Analogies helped software engineers link two objects from same or different domain and create new ideas. Tools provided knowledge on the fly when required by the software engineer and help software engineers concentrate on the original problem.

Knowledge is accessed using different ways and the most common knowledge access techniques that I found during the review were searching for prior knowledge in repositories,

internet, wikis, books and magazines and the second technique was collaboration, where software engineers shared theirs ideas and thoughts with others. They build on each other's ideas and provide constructive feedback. They covered each other's knowledge gap as a group.

Knowledge is very useful for software engineers and it makes them more creative and still I saw that not all software engineers were utilizing the available knowledge. The reasons for this were that searching for the right information in large repositories was cumbersome and when the information was found relevant information but would be missing in the repositories. Searching huge database and finding the right information was difficult as the search techniques were not very powerful. Gaining new knowledge via collaboration was difficult too, as finding the right experts to share the ideas and get input was difficult and when the right people found that they were busy in their work and had minimal or no time to collaborate.

There were few limitations in knowledge access methods, but I did see a few suggestions to improve these shortcomings in the literature review. To improve searching and make it easier, the knowledge should be stored in a consistent way, searching techniques should be improved, and powerful visual techniques should be added. We should try to combine repository knowledge with collaboration tools so that software engineers can find all the knowledge in one location. Software engineers should be trained on analogy making and utilizing analogies to extract useful information and come up with new ideas. Management of software industries should support and enhance knowledge sharing behavior among employees.

This paper targets the tool for capstone projects and it can be adapted to any group easily. In the development of the tool, the repository and collaboration approach has been combined. The repository will contain all the data from previous capstone projects providing the prior knowledge. The tool will have a discussion board which will provide a platform for capstone

projects to collaborate and share knowledge. The data will be organized in a specific order so that it is easy to search.

5.2. Strengths and weaknesses

This section discusses the strengths and weaknesses of the evidence collected in the literature review by examining the source selection, source quality, and validity of the evidence.

5.3. Source selection

A wide variety of sources were selected from in order to produce a large list of candidate papers for inclusion in the literature review. Multiple literature databases covering relevant journals, proceedings, and other publications were searched and a manual search of conference proceedings and journals associated with the topic area was also conducted. Unfortunately, there were not a large number of empirically validated sources restricted to the precise topic area of this report. In order to increase the number of available sources and evidence of knowledge as a factor to improve creativity, some constraints of the inclusion/exclusion criteria were relaxed. Specifically, sources from the information technology and information systems fields were considered, especially if they made some indication that the research was focused on knowledge and creativity.

5.4. Source quality

To ensure that only quality evidence was included for consideration in this literature review, I ensured that all sources considered had some kind of experiment, survey, case study to claim what they state. In general all sources can be considered to have a quality, at least insofar as the data in the sources is based on an empirical study. Due to the low number of sources

immediately related to the topic area, it was necessary to relax constraints to improve the overall number of sources. But then I still ensured that they talk about creativity and knowledge.

5.4.1. Validity of evidence

Stringent inclusion and exclusion criteria were used to ensure that only appropriate papers were included in this literature review. To reduce potential bias, data extraction forms were used to ensure that the same information was extracted from each source. After extracting relevant information from all sources, the researchers of this work compared a subset of selected papers to ensure that the data being extracted was similar and consistent. When possible, researchers attempted to apply the same extraction method (e.g. only extract results that were reported as statistically significant.) across sources.

5.4.2. Threat to validity

One of the threats is that most of the sources found during the review focused on the relationship between knowledge and creativity in the requirements and design stages of software engineering. I did not find any source that focused on the creativity during the testing stage of software development. I anticipate that this relationship (knowledge vs. creativity) would hold true for other stages of software development, due to the nature of testing as a knowledge-intensive activity. However, there is no empirical evidence that was reported in literature to underscore this claim. Another validity threat is that the sources which reported the effect of knowledge on the creativity during the design included designers that were not limited to software engineering background. However, the results were consistent across different studies.

6. RESEARCH TOOL – KNOWLEDGE REPOSITORY AND DISCUSSION FORUM FOR CAPSTONE PROJECTS

From the literature survey, it was revealed that knowledge does increase creativity of software engineers. The results provided evidence regarding various forms of knowledge that influences and increases the creativity of software engineers. The two primary ways knowledge is accessed are repository and collaboration. **Repositories** are mostly used to access pre-knowledge or information from past experiences or previous projects. Pre-knowledge was one of the forms of knowledge identified in the literature review. Pre-knowledge is any kind of information from the previous work of solving a problem. It could be sketches of design, documents, clippings from magazines etc. Pre-knowledge gives a starting point to software engineers to proceed towards the solution. It might also give information about the dos and don'ts for certain design approaches. Pre-knowledge inspires new ideas to solve the current problem in hand. **Collaboration** was the second approach where knowledge spreads and develops. Collaboration or knowledge-sharing fills one another's knowledge gaps, opens new channels of thought, and improves one's idea and helps him be creative. Collaboration gives access to dynamic knowledge.

I decided to combine these two approaches into a single tool. I thought that combining repository knowledge and collaboration into one tool will bring the benefits of both. This tool is specific to capstone projects currently but is equally applicable to elsewhere. For the repository, I used the data from previous capstone projects. The different information captured is explained in detail in the later section. For collaboration, I plugged in a discussion board with the repository. When the repository cannot provide the knowledge that the user is looking for, he or she can utilize the discussion board to find the right information and fill in the knowledge gap. Searching

in huge repositories can be cumbersome, hence I have organized the data in particular structure which will help the end users. I have provided various types of reports that the users can look into and gain more knowledge and insights about the projects. I provide a discussion board where the users can share their knowledge or post questions about their problems. This platform will enable the students to share their knowledge and gain new knowledge.

6.1. Repository: Pre-knowledge reports from previous projects

I can generate multiple reports to pull information about previous projects. The reports are classified into mainly seven buckets: 1) Project type; 2) Project team size; 3) Project status; 4) Project scale; 5) Project grade; and 6) Project effort deviation type.

The type of reports present in every bucket is as follows: 1) Good features; 2) Bad features; 3) Lessons learnt; 4) Effort deviation reasons; and 5) Recommendations.

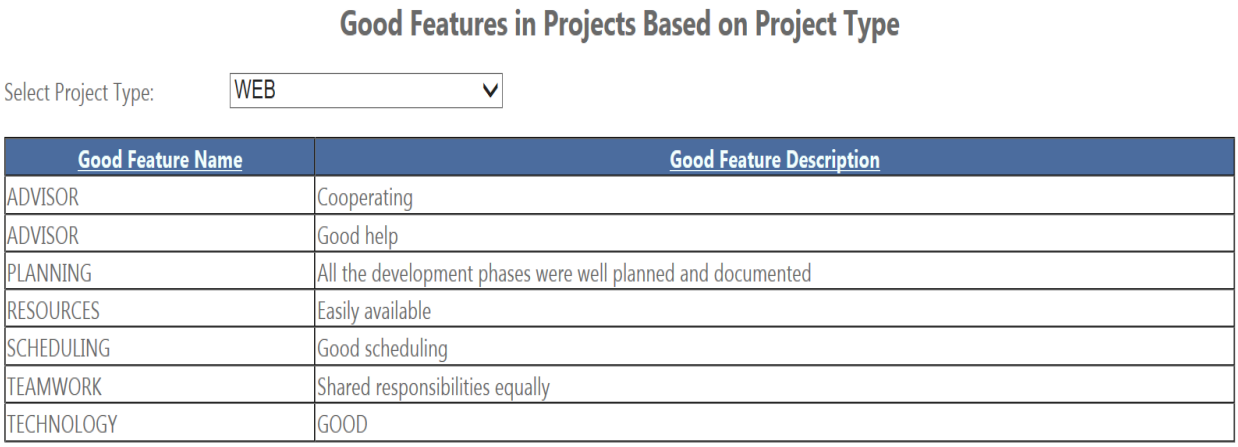
In this section I will discuss every bucket in detail.

6.1.1. Project type

The reports are divided based on the type of project. Figure 1 is a screenshot representing the types of projects. The types of projects could be web, desktop, etc.

Figure 1. Project types

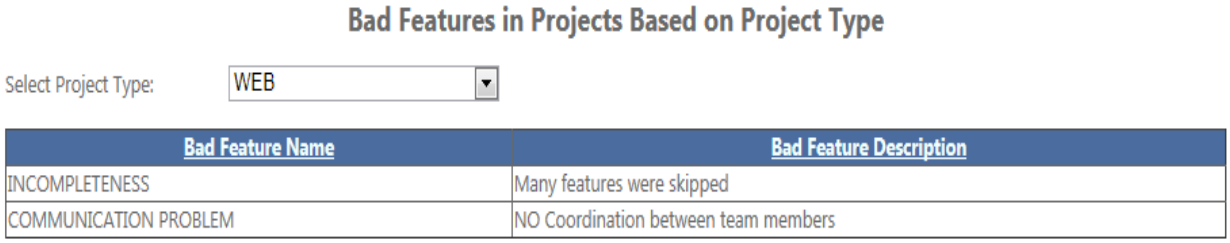
This report provides all the good features of a particular type of project with description. This can help the current project teams to incorporate these features, if they are applicable to their project. Figure 2 represents a screenshot of good features of all the web projects.



Good Feature Name	Good Feature Description
ADVISOR	Cooperating
ADVISOR	Good help
PLANNING	All the development phases were well planned and documented
RESOURCES	Easily available
SCHEDULING	Good scheduling
TEAMWORK	Shared responsibilities equally
TECHNOLOGY	GOOD

Figure 2. Good features from web projects

This report also provides all the bad features of a particular type of project. This can help the current project teams to avoid such features in their team. Figure 3 represents a screenshot of bad features of web projects.



Bad Feature Name	Bad Feature Description
INCOMPLETENESS	Many features were skipped
COMMUNICATION PROBLEM	NO Coordination between team members

Figure 3. Bad features from web projects

This report can provide the lessons learnt from a particular project type. This can help the current project teams to avoid the learning curve and utilize the information without undergoing the actual development cycle. They can utilize these lessons in their project. Figure 4 represents a screenshot of lessons learnt from web projects.

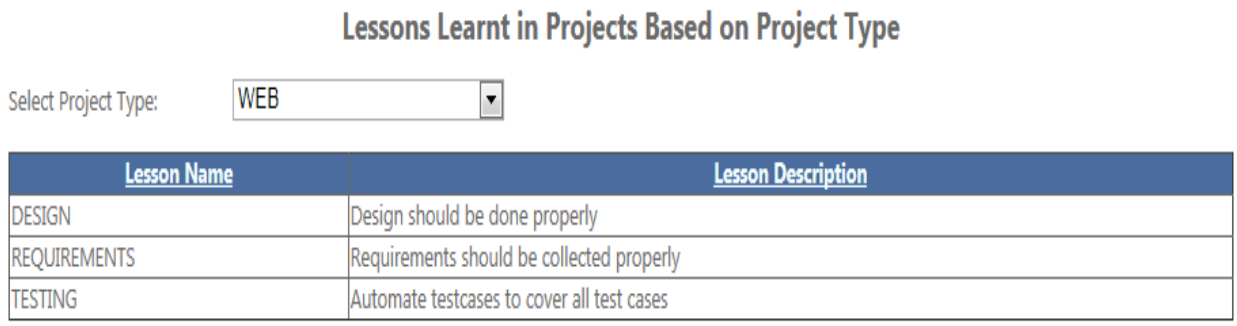


Figure 4. Lessons learnt from web projects

This report provides all the recommendations from the previous teams belonging to a particular project type. These are the collection of lessons that the teams would have done better if they would do it better second time. This information can be helpful for the current project teams to not make the same mistakes and improvise. Figure 5 represents a screenshot of recommendations from web projects.

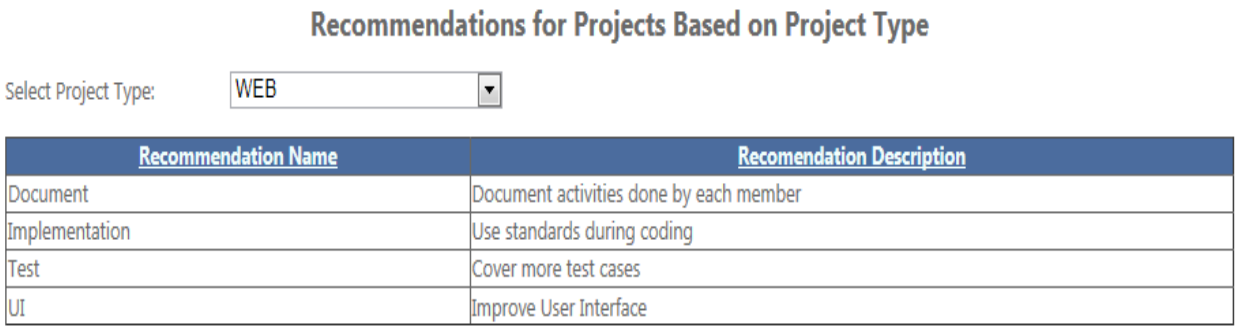


Figure 5. Recommendations from web projects

This report provides the effort deviation information and the reasons behind the deviation. This information can help the current projects to be aware of the deviation reasons and avoid them. Figure 6 represents a screenshot of effort deviations and reasons from web projects.

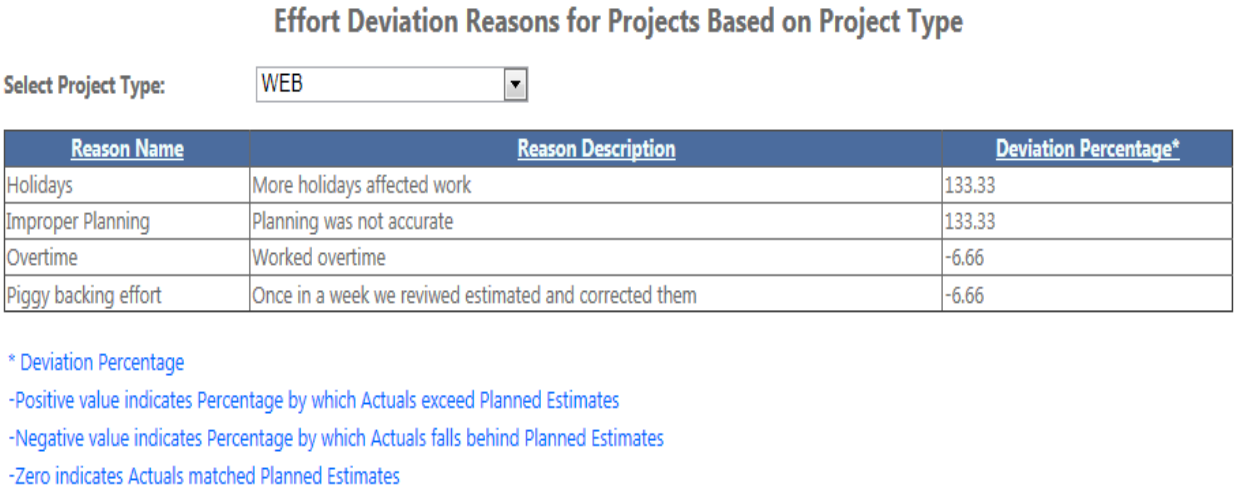


Figure 6. Effort deviations and reasons for web projects

6.1.2. Project team size

The reports are divided based on the project team size. Figure 7 represents a screenshot of project team sizes.

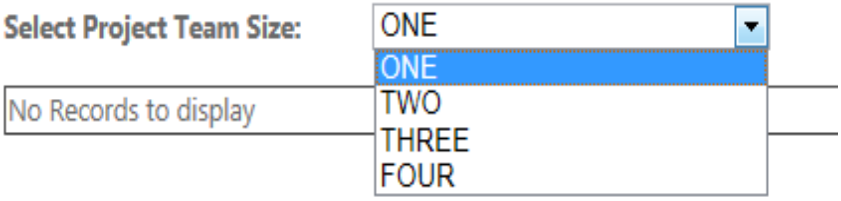


Figure 7. Project team sizes

This report provides all the good features of a particular team sized projects with description. This can help the current project teams to incorporate these features, if they are applicable to their project. Figure 8 represents a screenshot of good features from all the projects with team size of four.

Figure 8. Good features from the projects with a team size of four

This report provides all the bad features of a particular team sized projects. This can help the current project teams to avoid such features in their team. Figure 9 represents a screenshot of bad features from all the projects with team size of four.

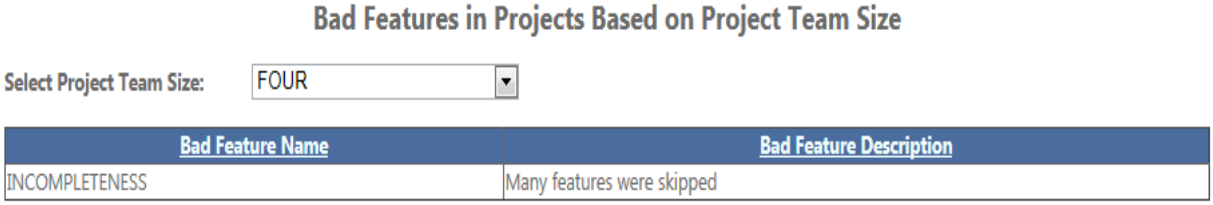


Figure 9. Bad features from all the projects with a team size of four

This report provides the lessons learnt from a particular team sized projects. This can help the current project teams to avoid the learning curve and utilize the information without undergoing the actual development cycle. They can utilize these lessons in their project. Figure 10 represents a screenshot of lessons learnt from all the projects with team size of four.

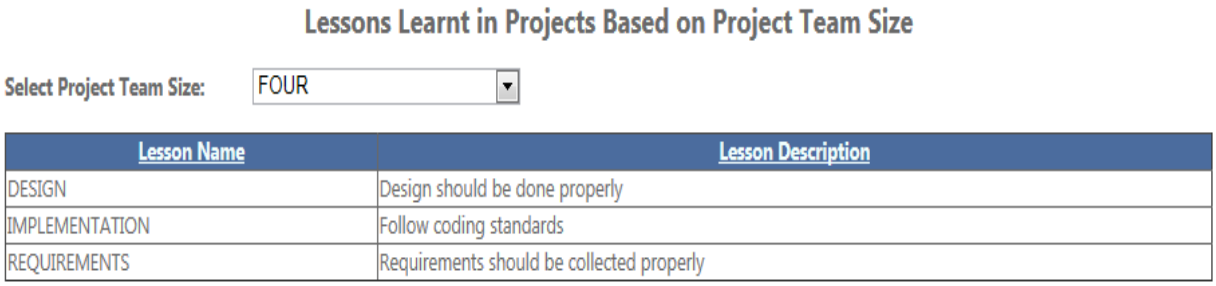
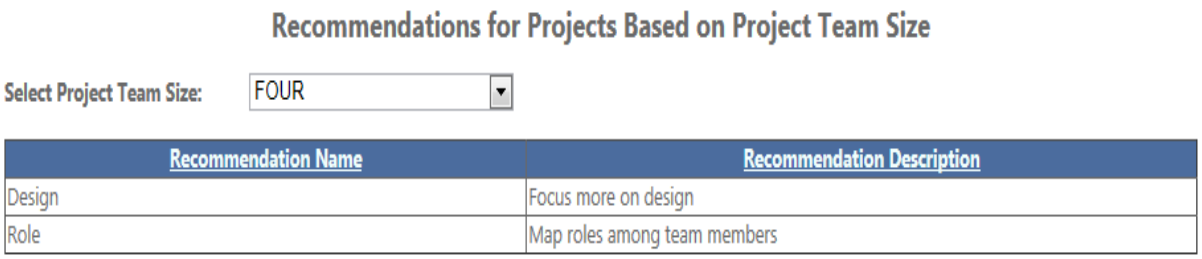


Figure 10. Lessons learnt from the project teams with a team size of four

This report provides all the recommendations from the previous teams with a particular team size. These are the collection of lessons that the teams would have done better if they would do it better second time. This information can be helpful for the current project teams to not make the same mistakes and improvise. Figure 11 represents a screenshot of recommendations from the projects with team size of four.



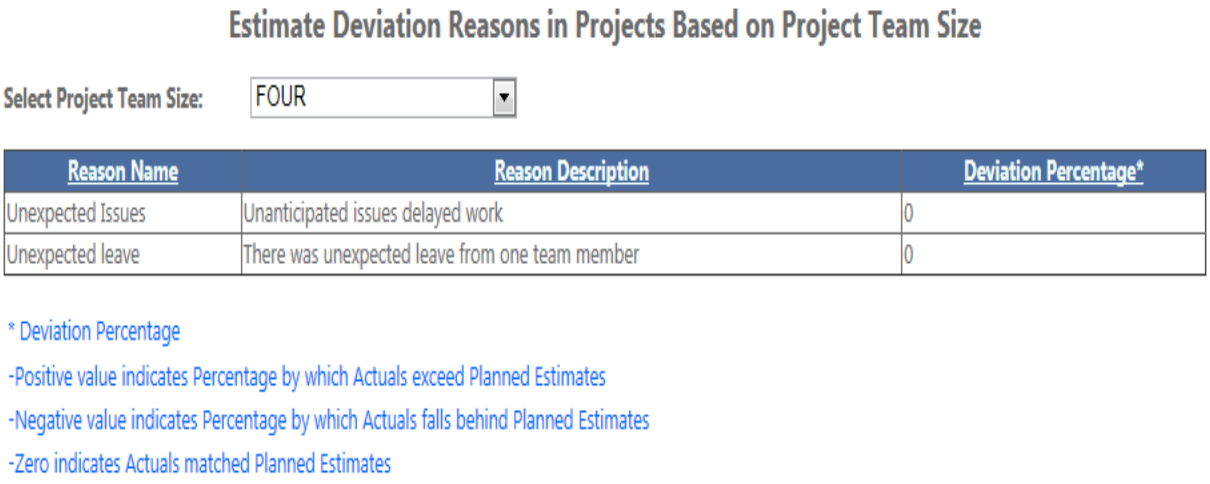
Recommendations for Projects Based on Project Team Size

Select Project Team Size: FOUR

Recommendation Name	Recommendation Description
Design	Focus more on design
Role	Map roles among team members

Figure 11. Recommendations from the projects with a team size of four

This report provides the effort deviation information and the reasons behind the deviation. This information can help the current projects to be aware of the deviation reasons and avoid them. Figure 12 represents a screenshot of effort deviations and reasons from the projects with team size of four.



Estimate Deviation Reasons in Projects Based on Project Team Size

Select Project Team Size: FOUR

Reason Name	Reason Description	Deviation Percentage*
Unexpected Issues	Unanticipated issues delayed work	0
Unexpected leave	There was unexpected leave from one team member	0

* Deviation Percentage

-Positive value indicates Percentage by which Actuals exceed Planned Estimates

-Negative value indicates Percentage by which Actuals falls behind Planned Estimates

-Zero indicates Actuals matched Planned Estimates

Figure 12. Effort deviations and reasons from all projects with a team size of four

6.1.3. Project scale

Reports are divided based on the scale or the size of the project. Figure 13 represents the screen shot of various project scales.

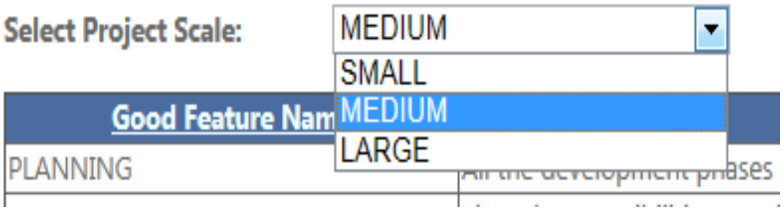


Figure 13. Project scales

This report provides all the good features from projects belonging to a particular scale with description. This can help the current project teams to incorporate these features, if they are applicable to their project. Figure 14 represents a screenshot of good features of all the medium scale projects.

Figure 14. Good features from medium scale projects

This report provides all the bad features projects belonging to a particular scale. This can help the current project teams to avoid such features in their team. Figure 15 represents a screenshot of bad features of medium scale projects.

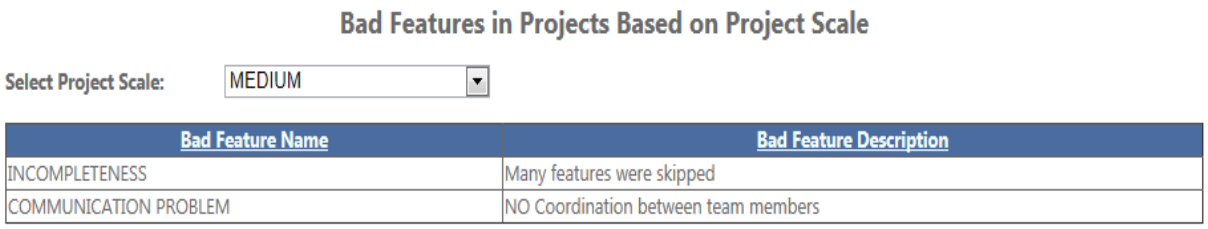


Figure 15. Bad features from medium scale projects

This report provides the lessons learnt from projects of a particular scale. This can help the current project teams to avoid the learning curve and utilize the information without undergoing the actual development cycle. They can utilize these lessons in their project. Figure 16 represents a screenshot of lessons learnt from medium scale projects.

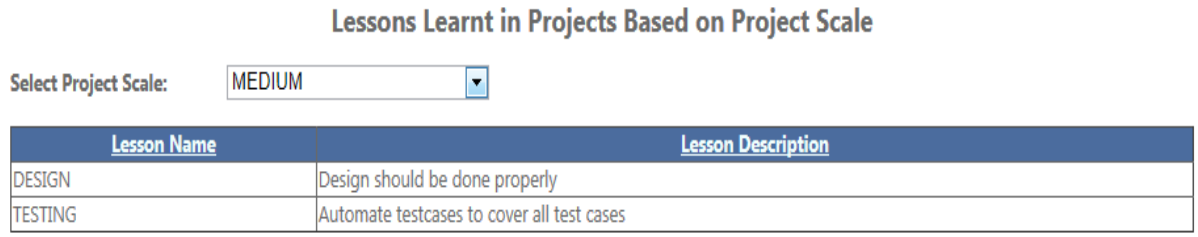


Figure 16. Lessons learnt from medium scale projects

This report provides all the recommendations from the previous projects belonging to a particular project scale. These are the collection of lessons that the teams would have done better if they would do it better second time. This information can be helpful for the current project teams to not make the same mistakes and improvise. Figure 17 represents a screenshot of recommendations from medium scale projects.

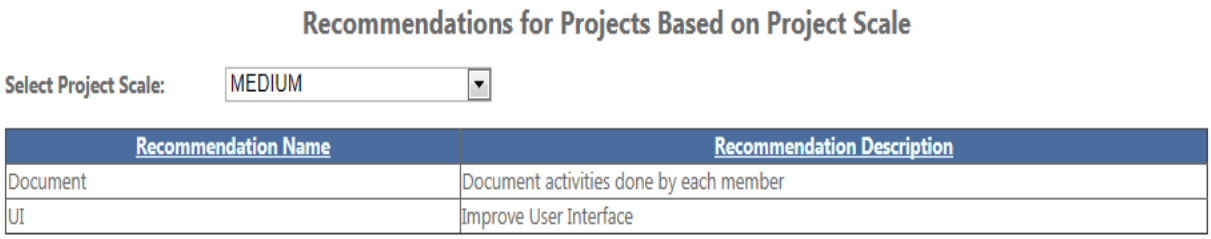


Figure 17. Recommendations from medium scale projects

This report provides the effort deviation information and the reasons behind the deviation. This information can help the current projects to be aware of the deviation reasons and avoid them. Figure 18 represents a screenshot of effort deviations and reasons from medium scale projects

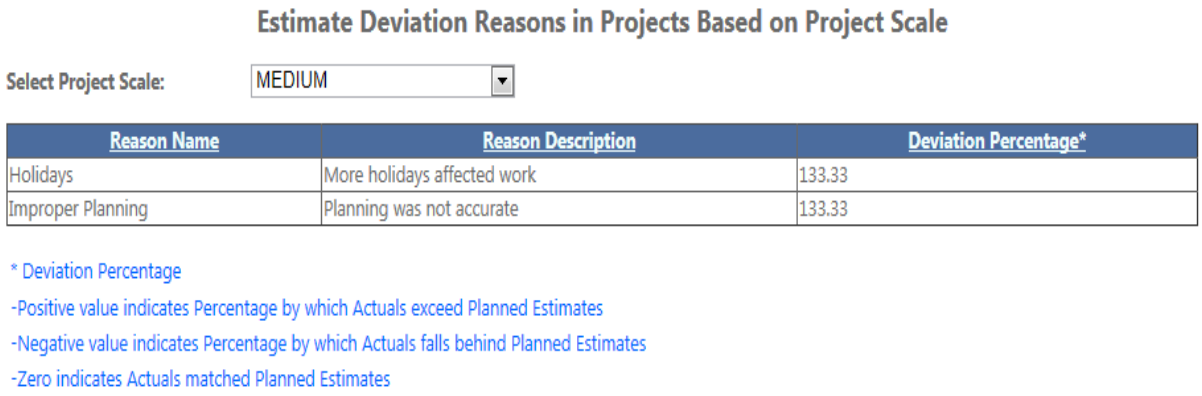


Figure 18. Effort deviations and reasons from medium scale projects

6.1.4. Project grades

Reports are divided on the grades that the project team obtained. Figure 19 represents a screenshot of all the project grades.

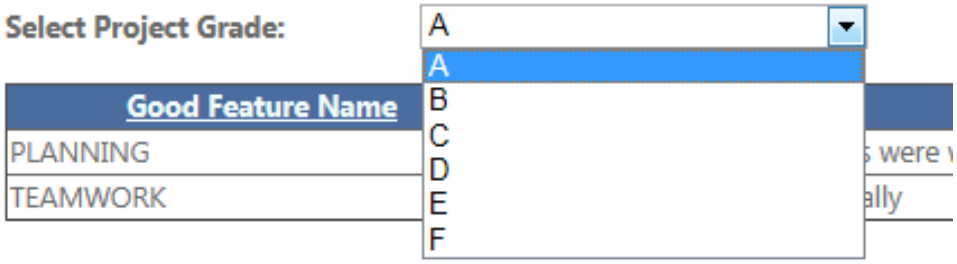


Figure 19. Project grades

This report provides all the good features from projects with a particular grade. This can help the current project teams to incorporate these features, if they are applicable to their project. Figure 20 represents a screenshot of good features of all the projects with a grade.

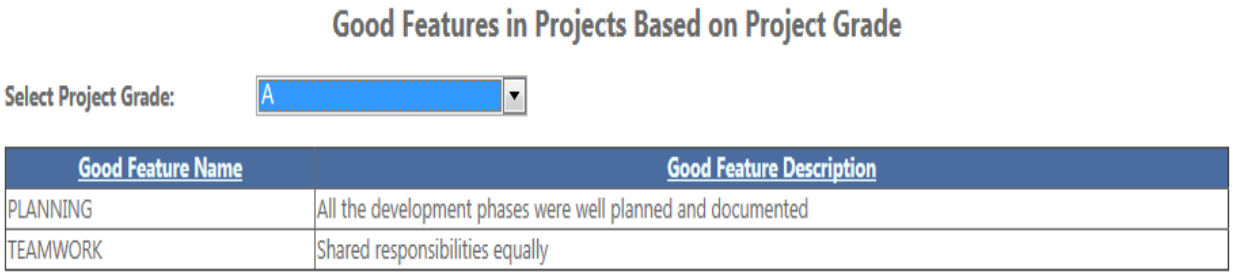


Figure 20. Good features from projects with a grade

This report provides all the bad features from projects with a particular grade. This can help the current project teams to avoid such features in their team. Figure 21 represents a screenshot of bad features of all the projects with a grade.

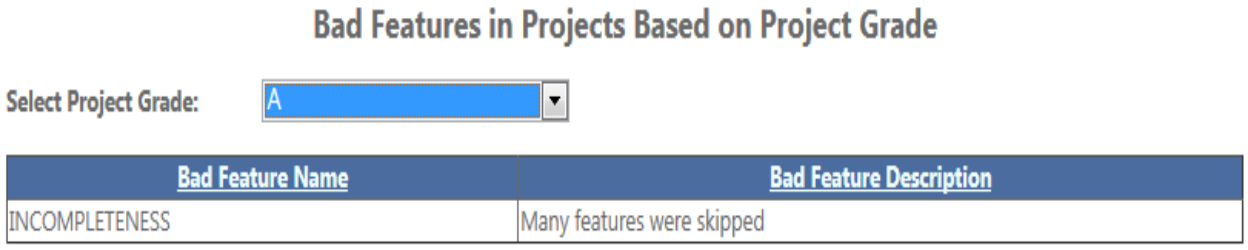


Figure 21. Bad features from projects with a grade

This report provides the lessons learnt from a particular project type. This can help the current project teams to avoid the learning curve and utilize the information without undergoing the actual development cycle. They can utilize these lessons in their project. Figure 22 represents a screenshot of lessons learnt from all the projects with a grade

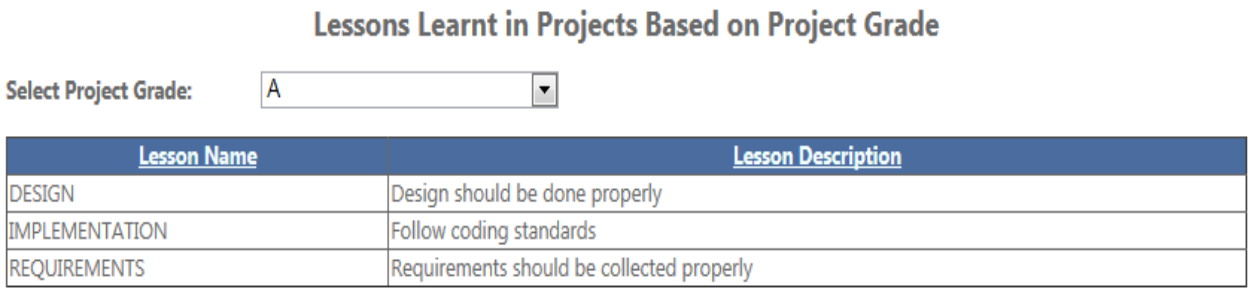


Figure 22. Lessons learnt from projects with a grade

This report provides all the recommendations from the projects with a particular grade. These are the collection of lessons that the teams would have done better if they would do it better second time. This information can be helpful for the current project teams to not make the same mistakes and improvise. Figure 23 represents a screenshot of recommendations from all the projects with a grade.

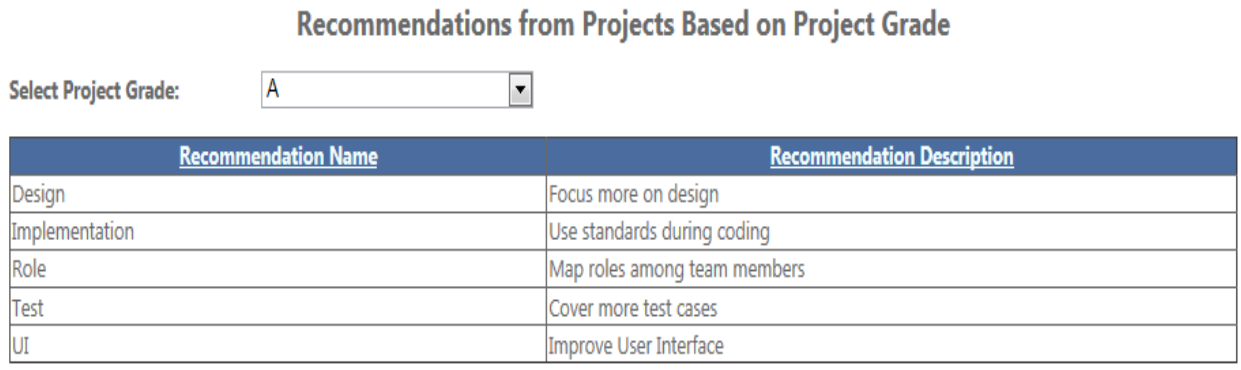
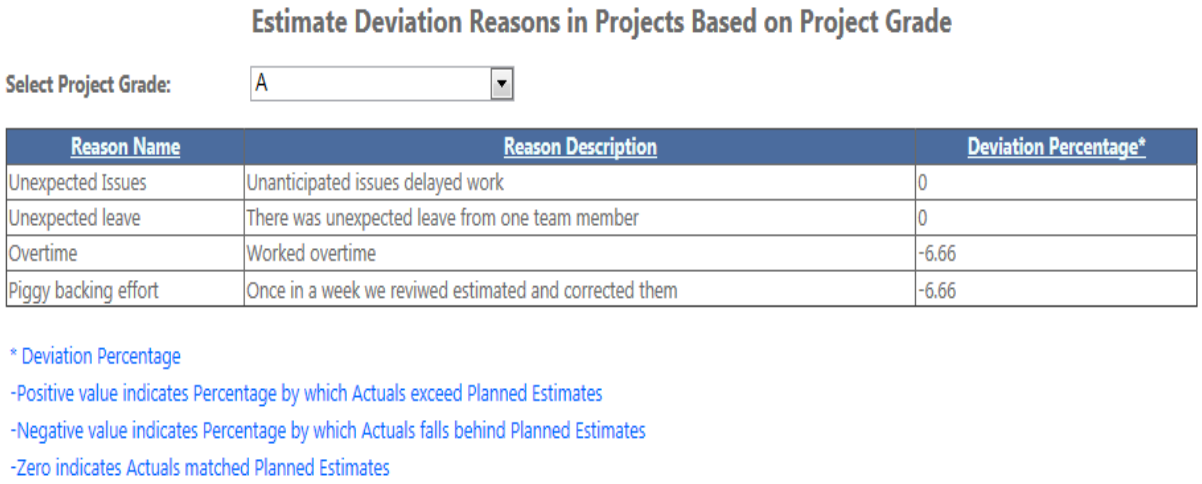


Figure 23. Recommendations from projects with a grade

This report provides the effort deviation information and the reasons behind the deviation. This information can help the current projects to be aware of the deviation reasons and avoid them. Figure 24 represents a screenshot of effort deviations and reasons from all the projects with a grade.



Estimate Deviation Reasons in Projects Based on Project Grade

Select Project Grade: A

Reason Name	Reason Description	Deviation Percentage*
Unexpected Issues	Unanticipated issues delayed work	0
Unexpected leave	There was unexpected leave from one team member	0
Overtime	Worked overtime	-6.66
Piggy backing effort	Once in a week we reviwed estimated and corrected them	-6.66

* Deviation Percentage
-Positive value indicates Percentage by which Actuals exceed Planned Estimates
-Negative value indicates Percentage by which Actuals falls behind Planned Estimates
-Zero indicates Actuals matched Planned Estimates

Figure 24. Effort deviation and reasons from projects with a grade

6.1.5. Project requirements

Reports are divided based on the requirements. Figure 25 represents the type of requirements.

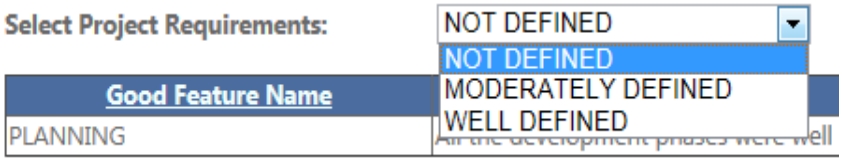


Figure 25. Project requirements

This report provides all the good features of a particular type of project requirements. This can help the current project teams to incorporate these features, if they are applicable to

their project. Figure 26 represents a screenshot of good features of all the projects with well-defined requirements.

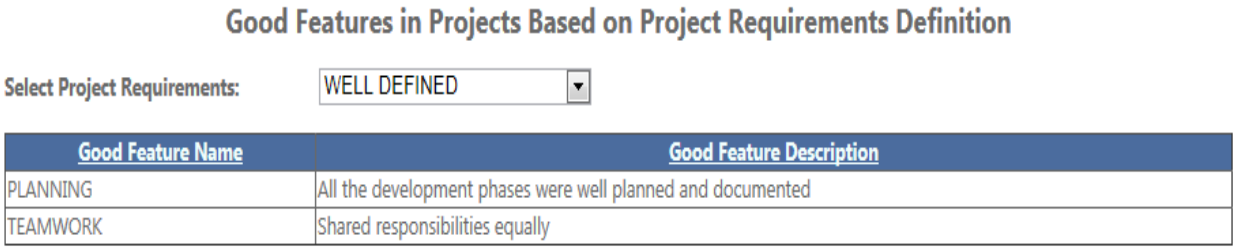


Figure 26. Good features from projects with well-defined requirements

This report provides all the bad features of a particular type of project. This can help the current project teams to avoid such features in their team. Figure 27 represents a screenshot of bad features of the projects with well-defined requirements.

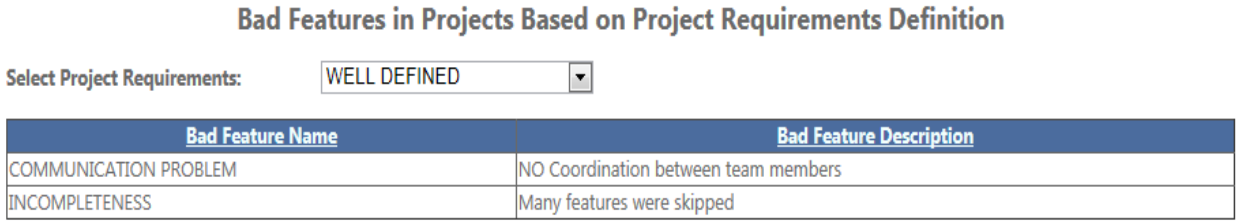


Figure 27. Bad features from projects with well-defined requirements

This report provides the lessons learnt from a particular project type. This can help the current project teams to avoid the learning curve and utilize the information without undergoing the actual development cycle. They can utilize these lessons in their project. Figure 28 represents a screenshot of lessons learnt from the projects with well-defined requirements.

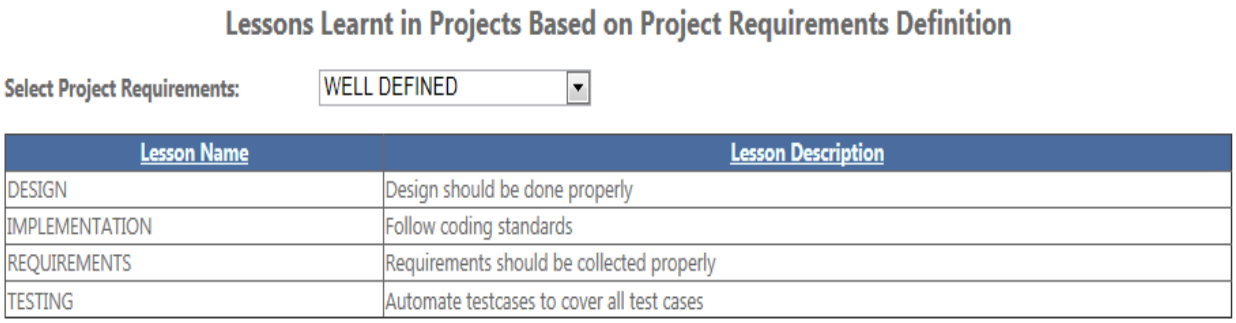


Figure 28. Lessons learnt from projects with well-defined requirements

This report provides all the recommendations from the previous teams belonging to a particular project type. These are the collection of lessons that the teams would have done better if they would do it better second time. This information can be helpful for the current project teams to not make the same mistakes and improvise. Figure 29 represents a screenshot of recommendations from the projects with well-defined requirements.

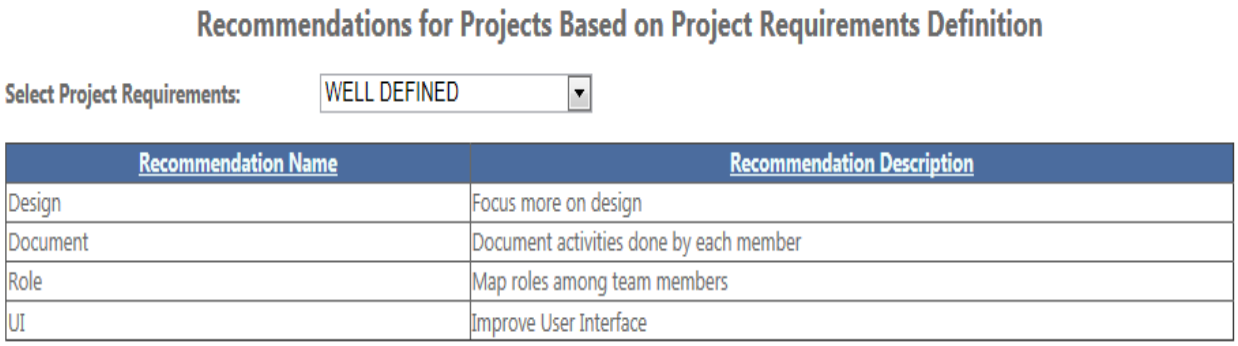


Figure 29. Recommendations from projects with well-defined requirements

This report provides the effort deviation information and the reasons behind the deviation. This information can help the current projects to be aware of the deviation reasons and avoid them. Figure 30 represents a screenshot of effort deviations and reasons from the projects with well-defined requirements.

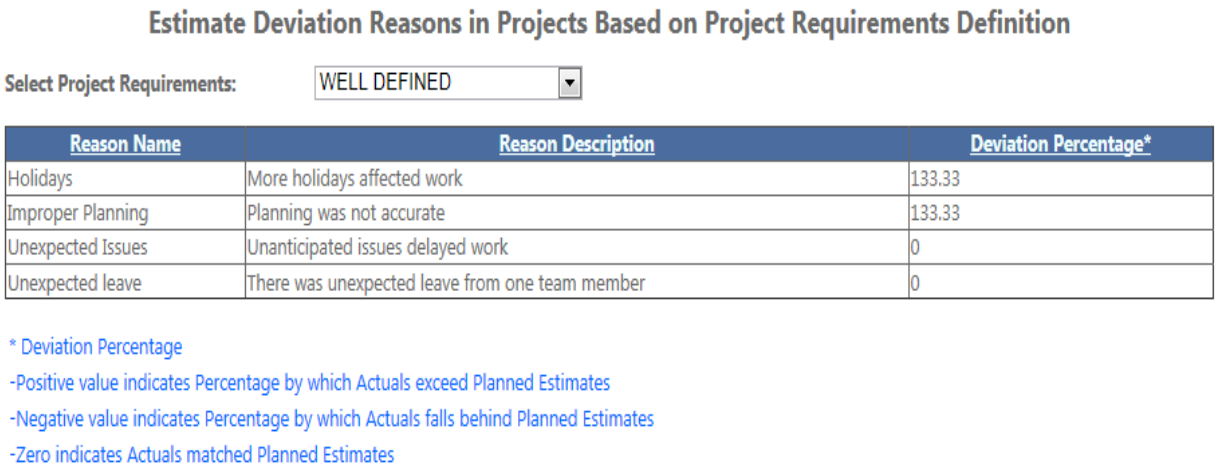


Figure 30. Effort deviations and reasons from projects with well-defined requirements

6.1.6. Project status

Reports are divided based on project status. Figure 31 represents a screenshot of various project states.

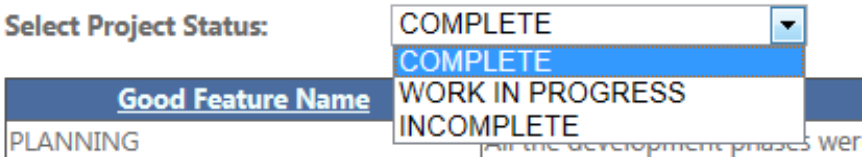


Figure 31. Project status

This report provides all the good features from projects which belong to a particular state. This can help the current project teams to incorporate these features, if they are applicable to their project. Figure 32 represents a screenshot of good features of all the completed projects.

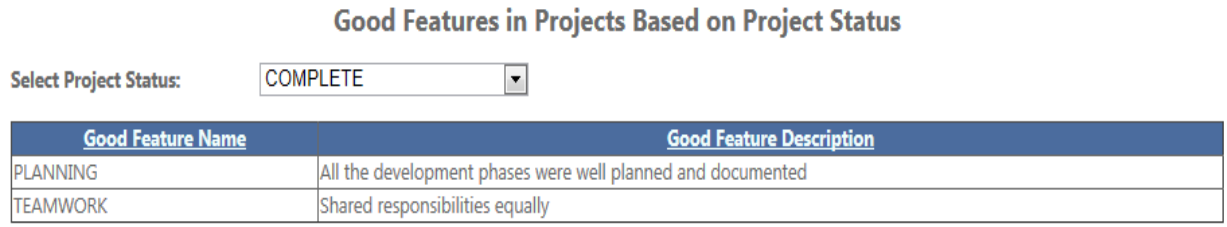


Figure 32. Good features from complete projects

This report provides all the bad features from projects which belong to a particular state. This can help the current project teams to avoid such features in their team. Figure 33 represents a screenshot of bad features of completed projects.

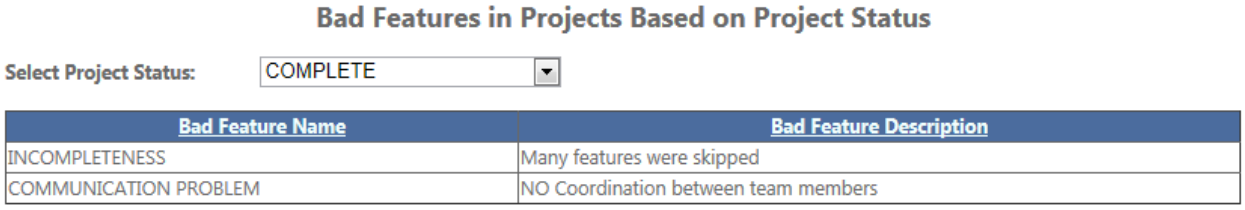


Figure 33. Bad features from complete projects

This report provides the lessons learnt from projects which belong to a particular state. This can help the current project teams to avoid the learning curve and utilize the information without undergoing the actual development cycle. They can utilize these lessons in their project. Figure 34 represents a screenshot of lessons learnt from completed projects.

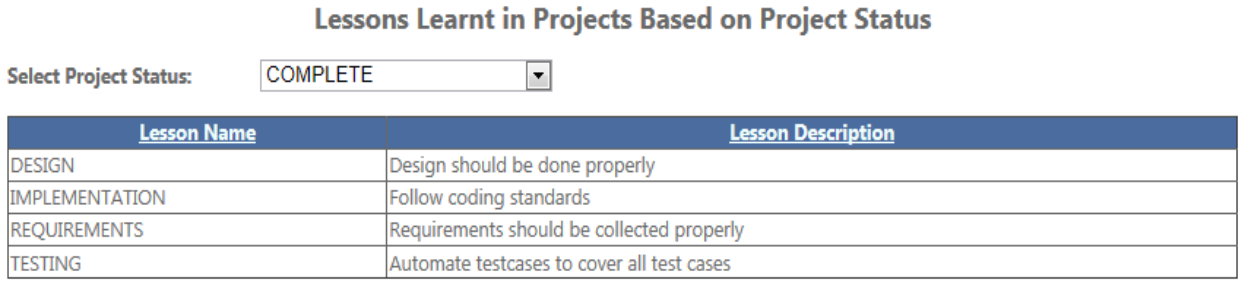
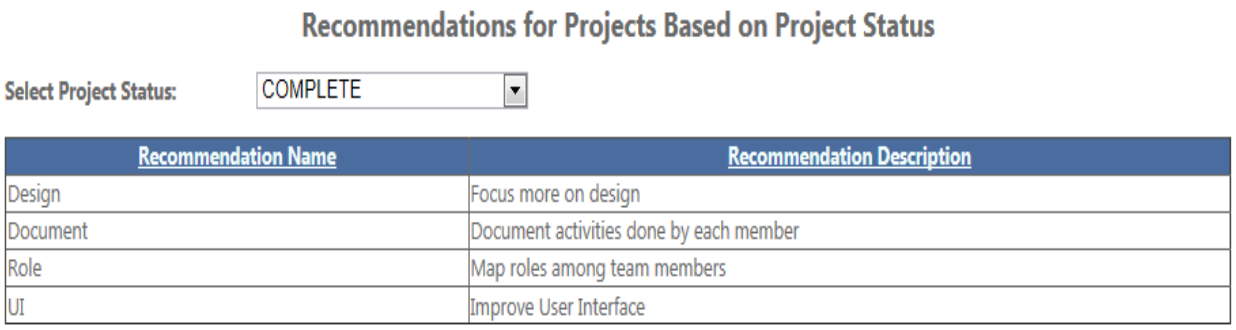


Figure 34. Lessons learnt from complete projects

This report provides all the recommendations from projects which belong to a particular state. These are the collection of lessons that the teams would have done better if they would do it better second time. This information can be helpful for the current project teams to not make the same mistakes and improvise. Figure 35 represents a screenshot of recommendations from completed projects.



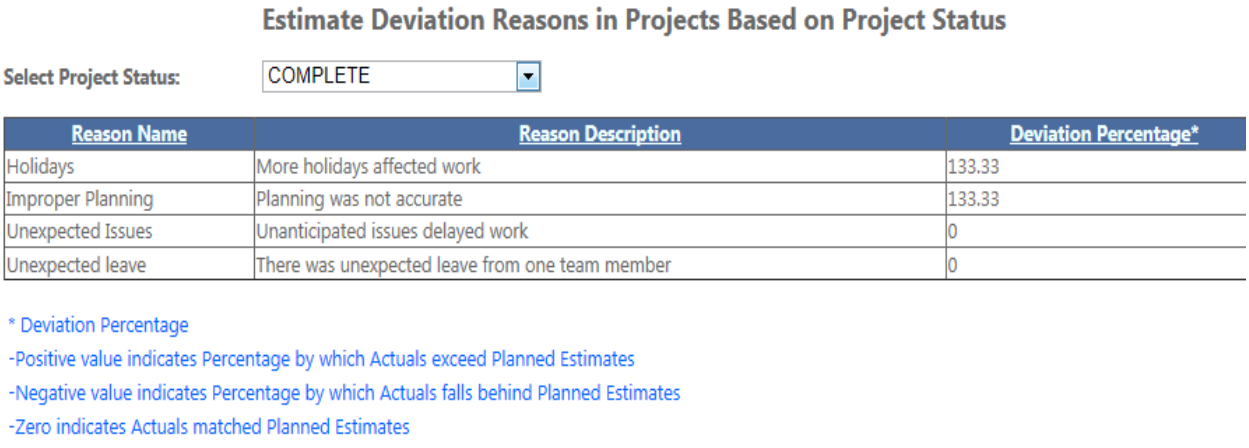
Recommendations for Projects Based on Project Status

Select Project Status: COMPLETE

Recommendation Name	Recommendation Description
Design	Focus more on design
Document	Document activities done by each member
Role	Map roles among team members
UI	Improve User Interface

Figure 35. Recommendations from complete projects

This report provides the effort deviation information and the reasons behind the deviation. This information can help the current projects to be aware of the deviation reasons and avoid them. Figure 36 represents a screenshot of effort deviations and reasons from completed projects.



Estimate Deviation Reasons in Projects Based on Project Status

Select Project Status: COMPLETE

Reason Name	Reason Description	Deviation Percentage*
Holidays	More holidays affected work	133.33
Improper Planning	Planning was not accurate	133.33
Unexpected Issues	Unanticipated issues delayed work	0
Unexpected leave	There was unexpected leave from one team member	0

* Deviation Percentage
-Positive value indicates Percentage by which Actuals exceed Planned Estimates
-Negative value indicates Percentage by which Actuals falls behind Planned Estimates
-Zero indicates Actuals matched Planned Estimates

Figure 36. Effort deviations and reasons from complete projects

6.1.7. Estimate deviation

Reports are divided based on the estimate deviation types. Figure 37 represents a screenshot of various estimate deviation types.

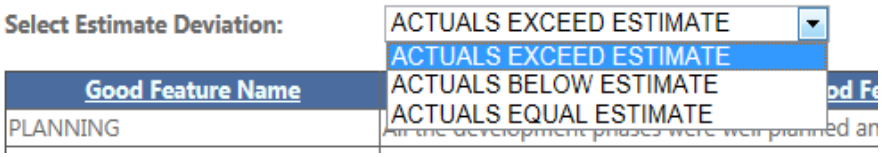


Figure 37. Estimated deviation

This report provides all the good features from projects with certain type of estimate deviation. This can help the current project teams to incorporate these features, if they are applicable to their project. Figure 38 represents a screenshot of good features of all the projects with actuals exceeding the estimation.

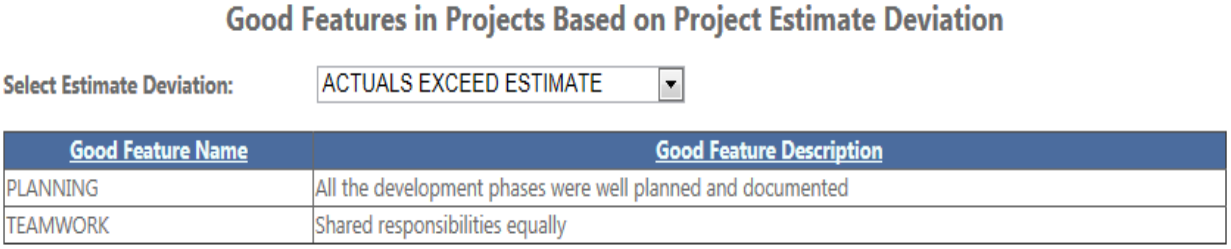


Figure 38. Good features of projects whose estimates were exceeded

This report provides all the bad features from projects with certain type of estimate deviation. This can help the current project teams to avoid such features in their team. Figure 39 represents a screenshot of bad features of projects with actuals exceeding the estimation.

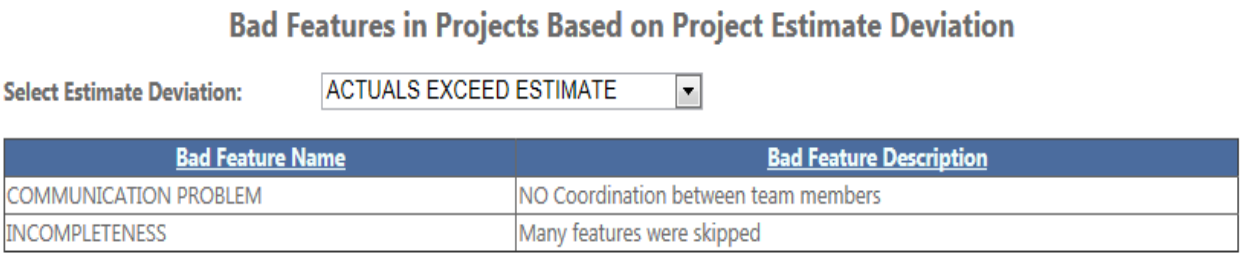


Figure 39. Bad features of projects whose actuals exceeded estimates

This report provides the lessons learnt from projects with certain type of estimate deviation. This can help the current project teams to avoid the learning curve and utilize the information without undergoing the actual development cycle. They can utilize these lessons in their project. Figure 40 represents a screenshot of lessons learnt from projects with actuals exceeding the estimation.

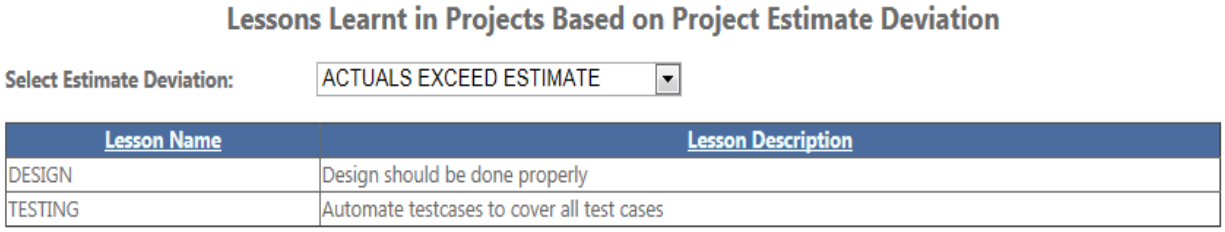


Figure 40. Lessons learnt from projects whose actuals exceeded estimates

This report provides all the recommendations from projects with certain type of estimate deviation. These are the collection of lessons that the teams would have done better if they would do it better second time. This information can be helpful for the current project teams to not make the same mistakes and improvise. Figure 41 represents a screenshot of recommendations from projects with actuals exceeding the estimation.

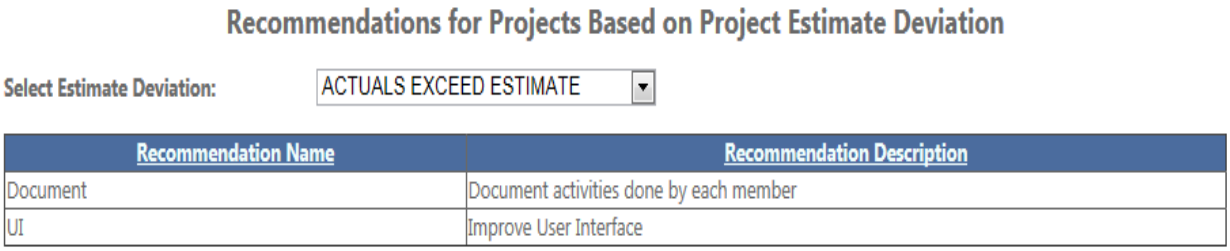


Figure 41. Recommendations from projects whose actuals exceeded estimates

6.2. Discussion forum

The discussion forum (Figure 42) lets users log in and post new topics for discussions and share the knowledge or initiate a good conversation which can pull out knowledge from individuals. Students can also post the problems or road blocks that they are facing in the forum and can get quick help. Students can help others in their problems by answering to others question or problems.

During the literature survey, the results showed that it was very difficult to find a person and his or her time to get their feedback. In this kind of setting it is easy to find answers for the problem areas or unknown areas easily. This forum currently is available to just students of capstone projects, but it can be exposed to all students or can be linked with other domain discussion forum. This is because; I saw in the survey that people from other domain can give good feedback as they see the idea with different viewpoint.

The top section is where the user can post a fresh topic or his or her problem in their current project. The second section is a list of all topics currently being discussed. The user can click on any of the topic and the comments for that topic appear in the third section. The user can reply to that topic on the fourth section.

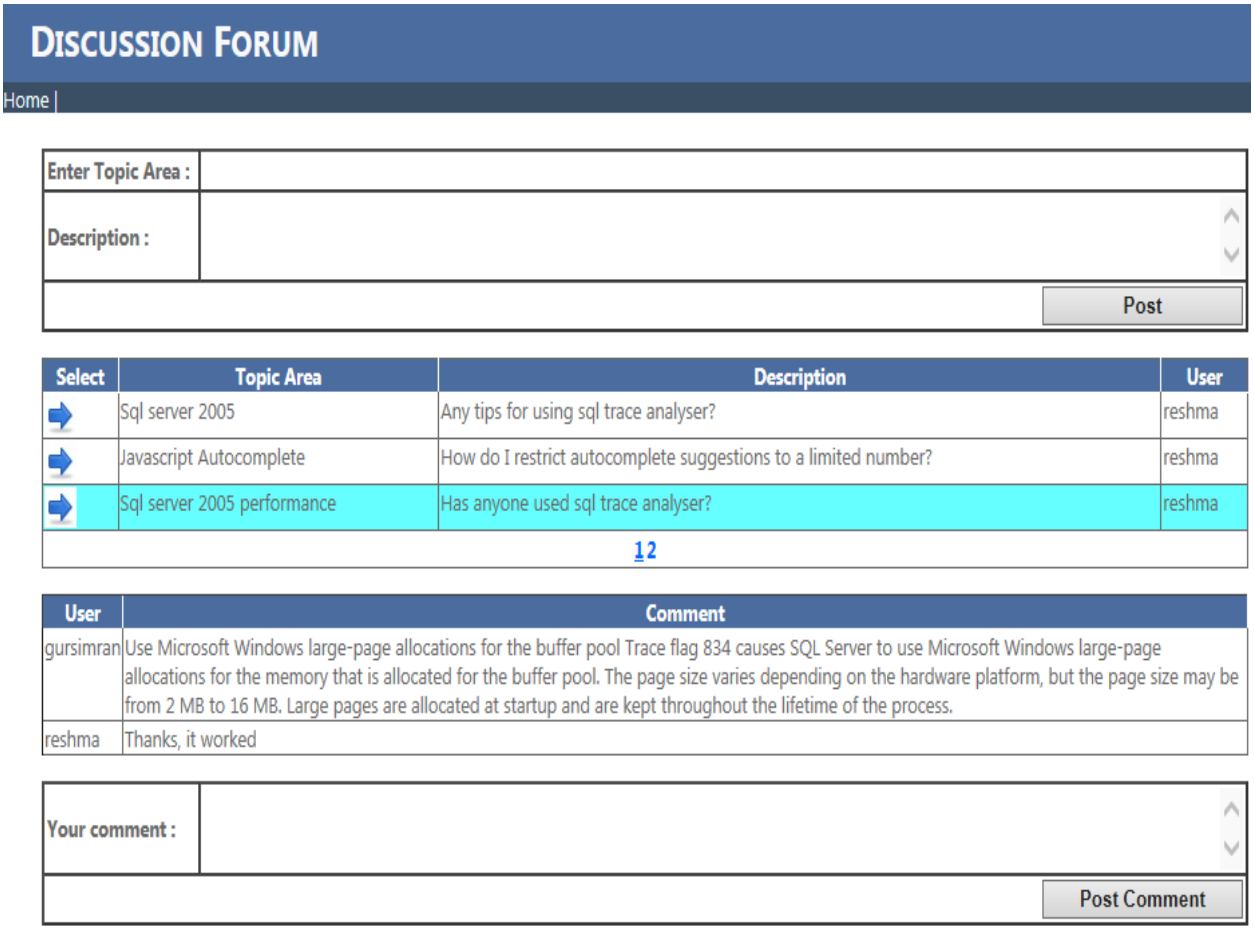


Figure 42. Discussion forum

7. CONCLUSION

For future studies I can improve and update the repository knowledge, and provide features so that the user can maintain the repository. This tool also can be utilized by capstone projects in the future semesters and feedback can be collected from students to understand how useful the tool was. We can also make changes based on the feedback or suggestions. I can also add features to update and maintain the repository. I can make the collaboration more interactive and try to involve people from various domains participate in the collaboration and sharing of knowledge. Future research into this area would be necessary to find more details about the testing stage of software engineering. I can also study the knowledge forms individually and develop tools to improve the user experience with knowledge access.

This literature review reports the findings of an investigation of the influence of knowledge on creativity of software engineers. The results of this review indicate that knowledge influences creativity positively among software professionals. Knowledge is available in various forms and is accessed by software professional in various ways. Knowledge in all forms helps software engineers come up with more ideas to solve their problems. I also saw how different forms of knowledge help or motivate software engineers during idea generation phase. Few of the limitation in the literature review were that, I was not able to gather any results specific to testing phase and not all the sources having information with respect to design were from software engineering field. The literature had subjects from various backgrounds including software engineering. The literature review helps software engineers and management of software industries to support and motivate knowledge sharing and accessing techniques. The literature review also points to all the ways knowledge can be accessed and about various forms

in which knowledge is available. This can help software engineers know where to find the information that is lacking during their regular work to help their solutions be more creative.

8. REFERENCES

1. A. K. Aurum, F. Daneshgar and J. Ward, “Investigating Knowledge Management Practices in Software Development Organizations – An Australian Experience. Information and Software Technology”, Information and Software Technology, Vol. 50, Issue. 6, pp. 511–533, May 2008.
2. B. A. Kitchenham, “Procedures for Undertaking Systematic Reviews”, Technical report, Computer Science Department, Keele University, 2004.
3. B. A. Kitchenham and S. Charters, “Guidelines for Performing Systematic Literature Reviews in Software Engineering.”, EBSE Technical Report, Keele University and Durham University Joint Report, July 2007.
4. N. Maiden, S. Manning, S. Robertson and J. Greenwood, “Integrating Creativity Workshops into Structured Requirements Processes”, Designing interactive systems, 2004.
5. M. Sharmin, B.P. Bailey, C. Coats and K. Hamilton. “Understanding Knowledge Management Practices for Early Design Activity and Its Implications for Reuse”, Conference on Human Factors in Computing Systems, pp. 2367-2376, 2009.
6. L. Candy and E. Edmonds, “Creative Design of the Lotus Bicycle: Implications for Knowledge Support Systems Research”, Design Studies, Vol. 17, Issue, 1, pp. 71-90, 1996.
7. N. Bonnardel, “Creativity in Design Activities: The Role of Analogies in a Constrained Cognitive Environment”, Creativity and Cognition conference, pp. 158-165, 1999.
8. L. Gabora, “Cognitive Mechanisms Underlying the Creative Process”, Creativity and Cognition conference, pp. 126-133, 2002.
9. Y. Ye, “Supporting Software Development as Knowledge –Intensive and Collaborative Activity”, Foundations of Software Engineering Conference, pp. 15-22, 2006.

10. S. Herring, C. Chang, J. Krantzler and B. Bailey, “Getting Inspired! Understanding How and Why Examples are Used in Creative Design Practice”, Conference on Human Factors in Computing Systems, pp. 87-96, 2009.
11. J.Hoorb, “A Model for Information Technologies that can be Creative”, Creativity and Cognition conference, pp. 186-191, 2002.
12. G. Kao, S. Lin and C. Sun, “Breaking Concept Boundaries to Enhance Creative Potential: Using Integrated Concept Maps for Conceptual Self-Awareness”, Computers & Education, Vol. 51, Issue. 4, pp. 1718-1728, 2008.
13. T. Baykara, “Dynamics of ‘Technological Creativity’ as a Decision in Knowledge Creation Process”, Technology Management for the Global Future, Vol. 2, pp. 951-956, July 2006.
14. T. Kappel, “Creativity in Design: The Contribution of Information Technology”, Engineering Management, 1999.
15. C. J. Bales and E. Y. Do, “Managing Information in a Creative Environment”, Creativity and Cognition conference, pp. 353-354, 2009.
16. J. Hailpern, E. Hinterbichler. C. Leppert, D. Cook and B. Bailey, “TEAM STORM: Demonstrating an Interaction Model for Working with Multiple Ideas during Creative Group Work”, Creativity and Cognition conference, pp. 193-202, 2007.
17. O. Ardaiz, M. L. Acedo and M. T. Acedo, “Wikiideas and Creativity Connector: Supporting Group Ideational Creativity”, Article No. 31, WikiSym, 2008.
18. L. Andrawina, “Relationship between Knowledge Sharing and Absorptive Capacity Moderated by Organizational and Technology Factors: A Conceptual Model”, IEEE International Conference of Industrial Engineering and Engineering Management, pp. 1865-1869, December 2009.