# DOES DOMAIN KNOWLEDGE INCREASE CREATIVITY DURING REQUIREMENTS DEVELOPMENT: AN EMPIRICAL INVESTIGATION

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

By

Sonu Sharma

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

Major Department:
Software Engineering

March 2014

Fargo, North Dakota

North Dakota State University
Graduate School

**Title**

Does Domain Knowledge Increase Creativity during Requirements Development: An Empirical Investigation

**By**

SONU SHARMA

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. Kenneth Magel

Dr. Kendall E Nygard

Dr.Liming Zhang

Approved by Department Chair:

04/10/2014
Date

Dr. Brian M. Slator
Signature

# ABSTRACT

To design and build a system/software, we need to understand the business of the organization, so understanding the business is very important for requirement analysis of such system. This requires an effective and efficient use of domain knowledge. In this paper we discuss the results of the empirical study that was carried out to understand the influence of domain knowledge on the creativity during requirement development phase. And then discuss our results, it's relevance to software organization and some additional findings from the study.

# ACKNOWLEDGEMENTS

I would like to express my gratitude to my supervisor Dr. Gursimran Walia for the useful comments, remarks and engagement through the learning process of this master paper. Also, I like to thank the participants in my survey, who have willingly shared their precious time during the process of study. I would like to thank my wife, Aditi, for her love, kindness and support she has shown during the past one year it has taken me to finalize this paper. Furthermore I would also like to thank my parents for their endless love and support. Lastly, I offer my regards and blessings to all of those who supported me in any respect during the completion of the project.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# 1. INTRODUCTION

Creativity refers to "the ability to produce new and original ideas and things" [1]. Scientific creativity can be characterized as a process toward achieving an outcome recognized as innovative by the relevant community. Creativity does not happen inside one person's head, but in the interaction between a person's thoughts and a socio-cultural context [3]. The creative process requires experimentation, the exploration of variations, and the continual evaluation of one's progress [2]. According to Brett Rolfe, interaction with individuals, analysis of different processes and resources leads to creative solution [7].

Historically, creativity has been defined by many psychologists. One of the most influential definitions was provided by psychologist Joy Paul Guilford, who proposed the concept of *Divergent Thinking* (i.e., the ability to create multiple solutions to the problem) as the main ingredient of creativity model. Guilford has conducted a multitude of studies and tests, to show that creativity cannot be evaluated through the standardized intelligence (or IQ) tests. Guilford, in fact showed that, the people who are able to generate a large number of solutions (some of which are original) are more creative and generally score lower on the IQ due to their divergent approach to the problems (as opposed to converging to a single solution).

Other than Guilford, other authors consequently characterized creativity as bringing into being something new, producing new knowledge or generating multiple solutions that are new, original and unique [8].

Generally, creativity is considered to be a factor in the fields of art and literature, where artistic innovation is considered to be a major criterion. Creativity is also associated with the fields related to design and architecture. However, fields related to science and technology (e.g.,

Software Industry), do not seems to exploit the creative abilities of individuals (e.g., software professionals) for technological innovations.

A part of a reason is due to the fact that technological industries (like software companies) follow strict development processes (devoid of effective brainstorming) and contains hierarchal structure that imposes top-down information flow which suppresses room for discussion and originality. Another reason is that, individuals working in a company are managed in a way that does not encourage creativity. For example, in most software companies, the requirements are finalized by management and are handed out to the developers (that does not leave room for discussions) to develop a working product that does something technical.

To that end, software development is the process of creating software solutions that are original and useful to end users. Software development is knowledge intensive and problem-solving activity often complicated by changing needs that require creativity and improvisation [3]. Therefore, creativity has an important place in the software development process. Researchers have begun to realize the importance of creativity in software development [3], especially during the later stages (e.g., design, implementation) of the software process [3]. Majority of creativity research has focused on developing computer-based tools for promoting creativity during the design phase of software development [3].

## 1.1. Problem statement

There is lack of research results on the creativity during the requirements phase of software process. Generally, requirements engineering is not considered as creative process because of the notion that requirements are gathered from stakeholder and written in a particular notation (e.g., IEEE requirements standard) and that all the creativity happens in the design phase. However, the invention of new software systems and products (e.g., smartphones) reveal that the

process of discovering what the system will and would not do (i.e., requirements) require creativity. Furthermore, the design of each innovative product has a requirements stage where the stakeholders create, invent, ideas which form the requirements for the software system to be developed [15].

In addition, software requirements development is more than just asking stakeholders what they want. It has been well documented that, stakeholders generally lack the technical expertise to specify the innovative requirements [11]. As eloquently put by Maiden et al. [11] and Mich et al., [11], the requirement engineering phase involves different stakeholders that have varying backgrounds and conflicting ideas that need to collaborate to arrive at a list of requirements that would lead to the development of an innovative product. Thus, requirement engineering phase requires creative people to innovate good ideas that would later be implemented.

Other researchers have also started to envision requirements development as a creative process [5]. While there has been multitude of empirical studies that have been conducted to evaluate the factors that affect creativity during the design stage [5], there has been very little work that has been done to study creativity during the requirements stage. One of the work was use of Divergent thinking [3] to come up with alternative possible responses and ideas during requirement gathering and analysis stage.

Some researchers try to quantify creativity, Jennifer Wiley try to quantify effects of Domain knowledge in creativity but that focus on fixation of ideas i.e.; they did not show the effect of domain knowledge in creative problem solving but demonstrated that how domain expertise can constrain the general solution in problem solving for experts [10].

## 1.2. Motivation

There are multiple factors (e.g., process, technology, domain etc.) that can affect the creativity during different stages (e.g., gathering, analysis, specification, management etc.) of the requirements engineering process. We are trying to investigate the effect of "*domain knowledge"* on the creativity during the requirements gathering process, and is discussed as follows.

To succeed during requirements elicitation, we should obtain knowledge of target business area to specify requirements [11]. In other words, incorrect or incomplete knowledge lead an analyst to maze of complex business. We call such necessary knowledge as *Domain Knowledge*, and call target business area as problem domain. In this paper we are trying to find out whether Domain knowledge plays a key role to come up with creative ideas during problem solving. As such, the effective and efficient usage of domain knowledge [11] is also important in requirements gathering and analysis. Invention of creative requirements can be established if an individual have strong domain knowledge.

The motivation behind this research is to examine the effect of domain knowledge on the creative thought process during software requirement gathering. To accomplish that, we compared the creative solutions (i.e., requirement gathered) prepared by domain experts in their familiar domain and unfamiliar domain. This paper reports the results from an empirical study that evaluates the creativity of requirements developers by giving them two problems; one from the domain they are familiar with, and the other one from an unfamiliar domain. They were asked to come up with set of requirements based on problem statement and then another set of requirements based on detailed use case. The correctness of the solution/use case was evaluated by a Software Engineering Graduate professor at NDSU and a Graduate student having an industry experience of 4 years in Software industry. Then there set of requirements on each

domain were plotted in a detailed concept map for each domain and each requirement was given a score based on a scoring technique (later defined in this paper). The cumulative score of all requirements for each participant is considered their creativity score.

# 2. BACKGROUND AND RELATED WORK

In this section we discuss about impact of creativity in Software Engineering and then discuss about studies conducted on creativity in Requirement Engineering.

## 2.1. Impact of creativity in software engineering

Software Engineering is performed by creative, knowledgeable people who should work with mature software process [9]. Creative people should reorganize, recombine or reinterprate any number of entities, concepts or ideas either by modifying, removing, or adding new elements to the existing processes in new and different ways. As stated in the above definition, software development needs creative people as they not only create results that are unique but also useful as they focus their effort on bringing something of value to the market. Creativity involves a dynamic balance between convergent and divergent thinking [4]. Divergent thinking is the ability to generate a set of possible responses, ideas, options, or alternatives in response to an open-ended question, task, or challenge; and convergent thinking involves narrowing this set to one alternative, and then implementing this alternative by empirically testing and communicating it to the scientific community [3]. In software development, the balance between convergent and divergent thinking is achieved by putting more emphasis on divergent thinking in idea- finding phase and then using convergent thinking by collecting the most promising, inviting or intriguing ideas. Researchers have conducted several studies of creativity in software engineering, in software industry designers face multiple challenges and problems during a problem solving. In that scenario designers expressed that generating multiple ideas helps them understand the problem better, prevents them from being reluctant to change, trigger new thoughts, and creates a rich set of possibilities. Interestingly, most designers considered ideas relatively easy to come

by, it was choosing the most promising ideas that required the most work and exercising creative talent [5]. Creative thinking leads to improved productivity as software engineers can refine and improve their performance by solving problems creatively in ever changing circumstances [3]. Software engineers have utilized the concepts from psychology to improve their thinking process and to be more creative and productive. For example; In the 1880s, Ringlemann examined the effects of working collectively on a rope-pulling task (tug-of-war) and noted a decrease in individual performance with increasing group size. Furnham applied the same concept for brainstorming sessions in software engineering and suggested that the group size should be about five to seven people. If there are too few people, not enough suggestions are generated. If too many people participate, the brainstorming session becomes uncontrollable [4].

## 2.2. Empirical studies on creativity in requirements engineering

Generally Requirements engineering is not recognized as a creative process. But there are some studies that proved that stakeholders increasingly create and invent ideas and express them as requirements [15]. This paper [15] reports techniques that were applied to encourage creative thinking during the requirements. It describes the use of unusual theories, such as analogical reasoning from cognitive science, to show the use of these techniques.

The study they conducted was to gather requirement for a system that will provide computer-based assistance to air traffic controllers to resolve potential conflicts between aircraft. The system was called CORA-2 system. The CORA-2 team consisted of one manager, 2 requirements engineers, 2 air traffic controllers who acted as domain experts, 1 human factors expert and 1 technical expert. Three one-day creativity workshops were conducted over a two-month period to generate requirements and design ideas for CORA-2. Each workshop involved between 16 and 20 team members and stakeholders (managers, air traffic controllers and

technology experts). The first workshop was designed to encourage explorative creativity, in which people explore possible ideas to create new ones. As it seems very similar to brainstorming, their innovation was to encourage analogical reasoning – common in creative domains – to generate new ideas. The last 2 workshops were designed to encourage combinatorial creativity, which is the creation of new ideas from a combination of existing ideas. And decompose requirements to make them more precise and easier to understand. The last 2 workshops were also designed to encourage transformational creativity. During transformational creativity people change the solution space in a way that things that were considered impossible are now possible (Boden 1990), for example by challenging pre-conceived constraints and exploring new solutions to existing problems.

There is another study [10] which shows that domain knowledge helps in eliciting requirements for software development.

Three major factors which influence creativity are:

1. ***Process Knowledge***: Understanding the proper way of documenting and maintaining documents. Like not only making a textual requirement document but building the Use Case and other diagrammatic document. Using a tool to manage requirements, so that all the related requirements are clubbed together and are easy to trace.
2. ***Domain Knowledge***: Domain Knowledge is very critical for Software development, having strong domain knowledge gives you advantage to think beyond the requirements and understand the complications and challenges pre hand. It saves a lot of planning and implementation time.

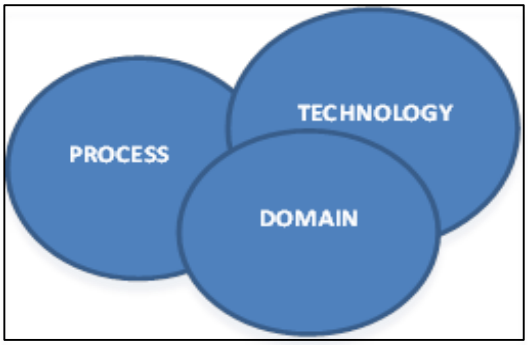


Figure 1. Factors influencing creativity

3. ***Technology***: If you understand a technology well. You can make correct decisions/suggestions at early phase of requirement gathering.

In this paper our main goal is to find out and evaluate the increase/decrease in creativity of a domain experts by giving them two problems one from the domain they are familiar with and the other from the domain (unfamiliar) they are not familiar with.

# 3. QUANTIFYING CREATIVITY

Existing researches on creativity in requirement phase is very subjective. And most of them talks about the process used to come up with creative solutions like using different requirement gathering and analyzing skills (eg, combinatorial creativity) [15]. Other research try to quantify effects of Domain knowledge in creativity but that focus on fixation of ideas i.e.; they did not show the effect of domain knowledge in creative problem solving but demonstrated that how domain expertise can constrain the general solution in problem solving for experts [10]. In this paper we try to quantify creativity of domain experts in requirement gathering phase and compare their creativity score in familiar and unfamiliar domain.

In this section we will describe our approach to quantify creativity in requirement phase. Creativity according to this paper is picking more specific requirement instead of abstract requirements. To specify requirement we divided a system/domain requirements in a tree structure using concept maps. In order to understand the creativity scoring we used, we need to first understand concept map and why we used it. A concept map is a type of graphic organizer used to help organize and represent knowledge of a subject. Concept maps begin with a main idea (or concept) and then branch out to show how that main idea can be broken down into specific topic. Concept maps represents knowledge/requirements in graphs. Concept maps consist of nodes (points/vertices) and links (arcs/edges). Nodes is a concept/requirement. Concept maps represent a hierarchical structure.

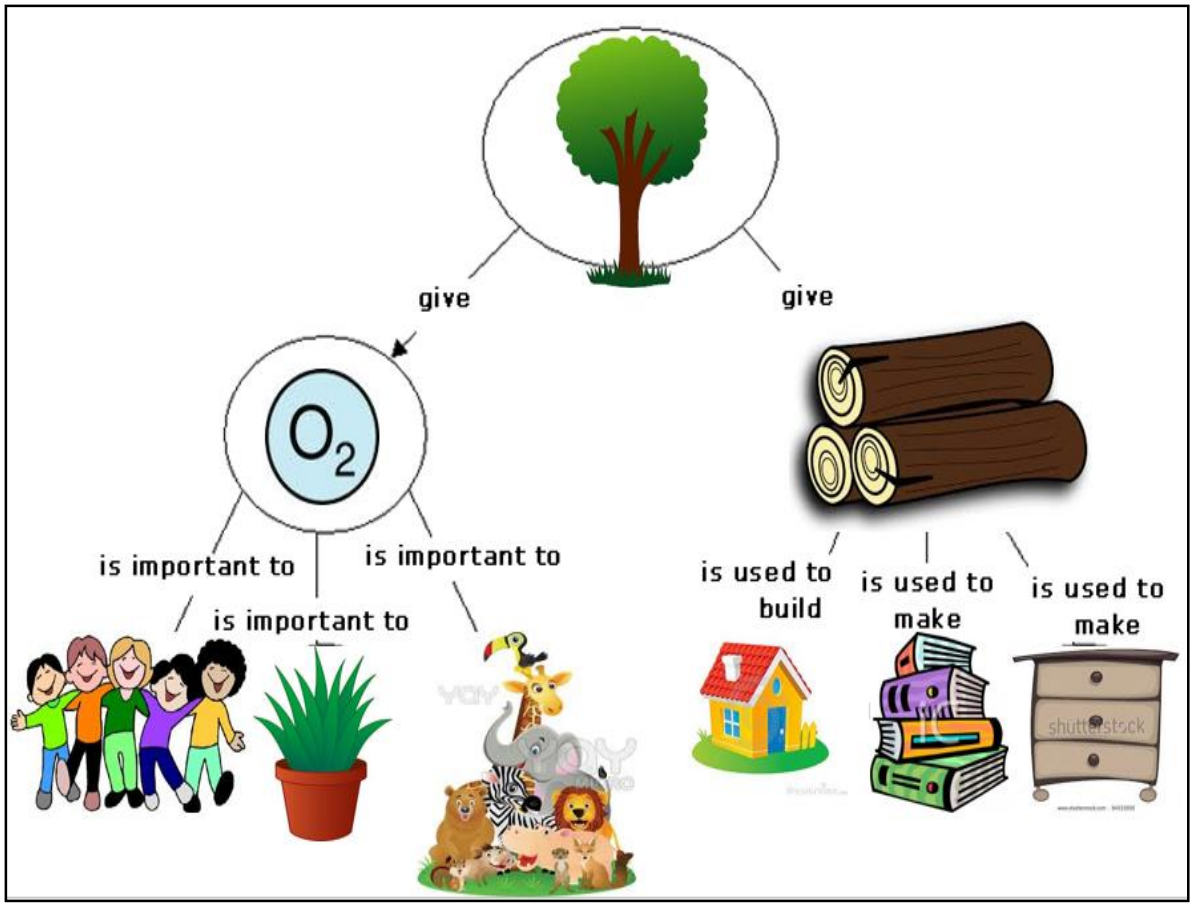

Figure 2. Mental map of a tree

Concept map helps understand the bigger picture and the problem to be solved. Concept maps helps identify missing requirements, identify errors. Concept maps helps understand what customers wants. They are Simple and easy to understand and create.

Concept maps used for familiar and unfamiliar domain in this paper represents requirements in each domain. Each node of the concept map represents a requirement. The top node represent an abstract requirement and the requirements gets more and more specific as we reach lower node in the tree. For example, consider the concept map of an ATM system (Figure 3) and two requirements as below

Requirement 1: ATM machine should support Multiple Languages

Requirement 2: ATM machine should support English language

Requirement 2 is more specific than Requirement 1. Requirement 2 gives a very clear picture which can be really helpful in later stages of software development. It removes ambiguity and assumptions from the system.

The concept map below, also shows that requirement for a domain are hierarchical and the child requirements are more specific than their parents. As lower levels are derived from their parents they require more thinking than the upper levels. The Lower level node in Tree are more specific than higher level node. But it does not imply that if some requirement is covering lower level node the requirement also cover its upper level node. To cover the parent node the requirement should cover all child nodes. For ex, for below concept map, if the Requirement says "ATM machine should support English and Spanish languages". It is covering all the child nodes of the parent node ("Language"). So the stakeholder thought about alternative solution which makes the requirement more specific than requirement 2. So this requirement should have higher creativity score than Requirement 2.

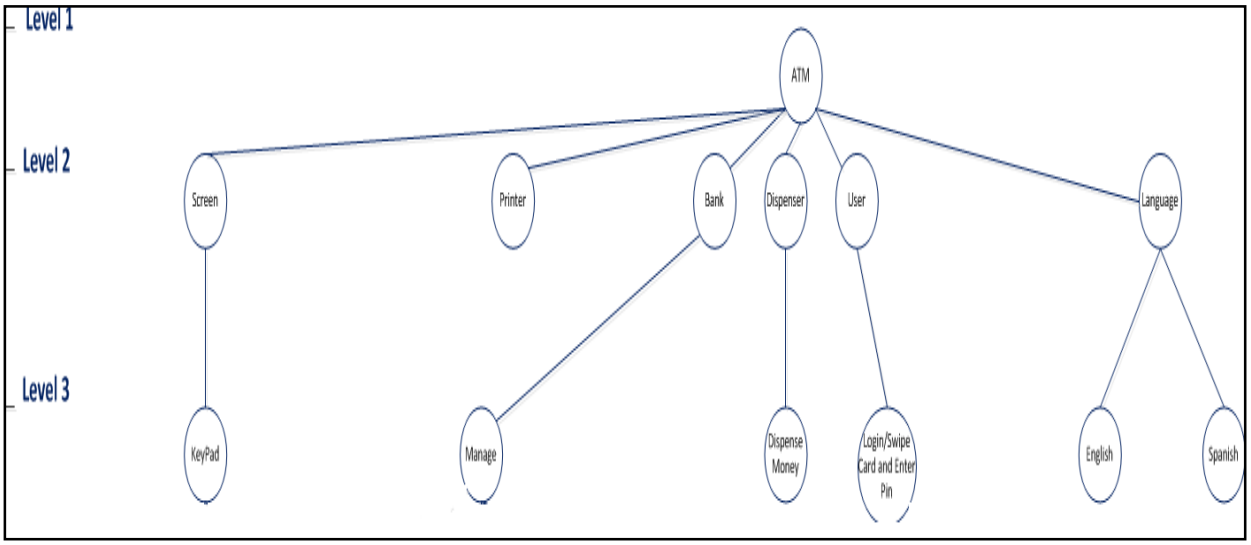


Figure 3. A portion of mental map of an ATM system

## 3.1. Calculation

The requirement gets more specific as the level increase, this means Creativity is directly proportional to the depth of concept map. To keep the scoring simple we kept value of each node at same level equal to level value. Top level has value 1 and the level value increment by 1 at each level. Creativity score for each participant in each domain (familiar and unfamiliar) is calculated in two Phases.

In order to describe C precisely, we introduce the following definitions and notations. Let M = (R, E) represent a mental map tree, with r ∈ R representing the set of requirements and $e_{ij}$ ∈ E, represents the hierarchical relationship between $r_i$ and $r_j$. Let $R_s \subseteq R$ define the set of nodes covered by requirements by a participant.

$$C = \sum_{R_s} depth(r)$$

Where depth(r) represents the depth of node r. Note that depth of root node is 1.

Let $R_{PD}$ be the set of requirements identified using problem description. Let $R_{UC}$ be the set of requirements identified using use cases.

For Phase I (gathering requirements by reading problem description), the score is computed as

$$C_{PD} = \sum_{R_{PD}} depth(r)$$

For Phase II (gathering additional requirements by using detailed use cases), the score is computed as

$$C_{UC} = \sum_{R_{UC}} depth(r)$$

The requirement gets more specific as the level node increase, this means Cr ∞ Ln (Cr → Creativity Score, Ln → Level of node). To keep the scoring simple we kept value of each node at same level equal to level value. Top level has value 1 and the level value increment by 1 at each level. Creativity score for each participant in each domain (familiar and unfamiliar) is calculated in two Phases.

Below is an example showing one of the detailed concept map used for Familiar Domain (ATM) to plot user requirement and calculate creativity score. Each node of the detailed concept map represents a requirement for the system. ATM's detailed concept map has 10 levels. Each node filled with Orange color represents requirement gathered by reading problem description. And the node filled with Blue color represents requirement gathered by detailed use case.

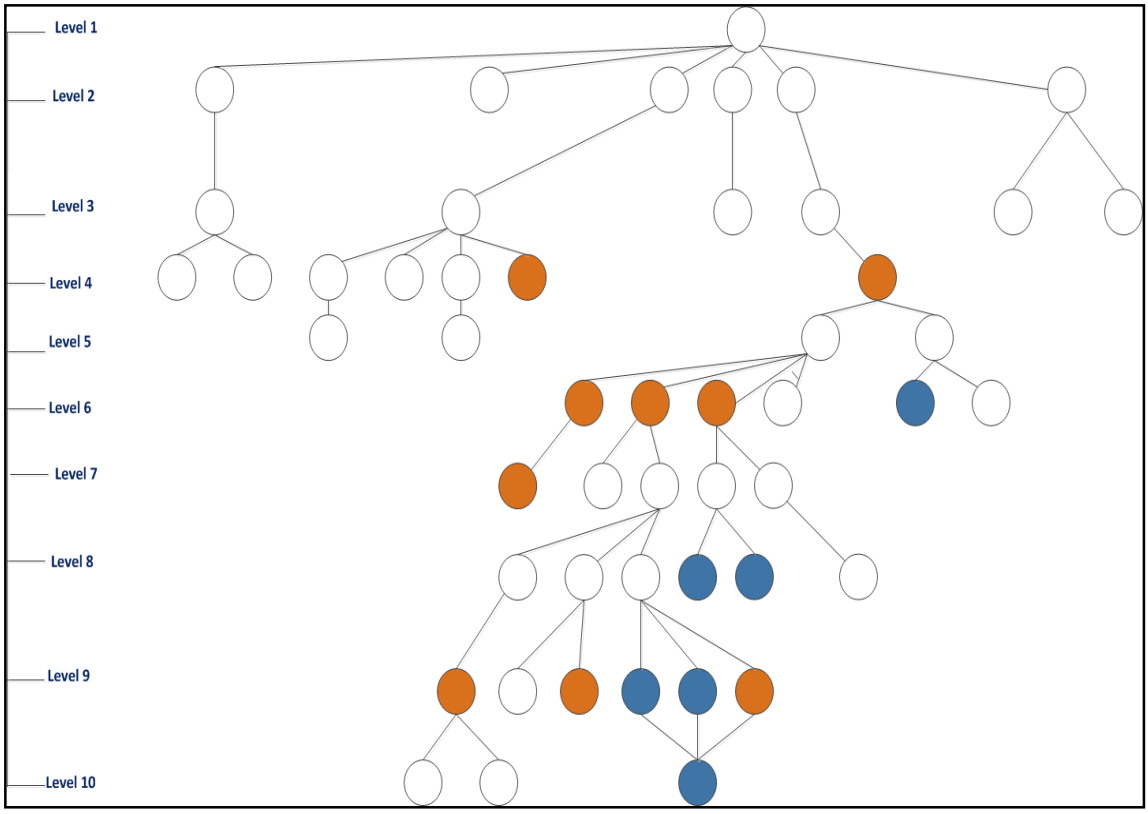


Figure 4. Calculation on mental map of an ATM system

So the scores from Figure 4 in previous page, came out as in Table 1 below.

Table 1. Creativity score calculation

| Phase I Score $C_{PD} = \sum_{R_{PD}} depth(r)$ | Phase II Score $C_{UC} = \sum_{R_{UC}} depth(r)$ | Phase I Number of valid Requirements | Phase II Number of valid Requirements |
|---|---|---|---|
| 4+ 4 + 6 + 6 + 6 +7 + 9 + 9 + 9 = 60 | 6 + 8 + 8 + 9 + 9 + 10 = 48 | 9 | 6 |

In this way each participant's set of requirements were scored and a detailed analysis of the score is done in section 5.

# 4. EXPERIMENT DESIGN

The main goal of this study is to compare the creativity of solution of domain experts in gathering requirements. Software problem can mean lot of different things in Software Engineering. For this research we have considered requirement gathering as a software problem. To properly focus on the study, a set of research questions were needed.

The three research questions used for this research are mentioned below. The research question is focused with a part of study to check the creativity of domain experts in software problem solving in their own domain and then compare the creativity of their solution in other domain. The research question focuses on a part of study that will be conducted to measure the approach domain experts take to come up with creative solutions.

***Research Question 1 (RQ 1)***: Does a domain expert come up with more creative requirements by reading problem description in unfamiliar domain than familiar domain?

***Research Question 2 (RQ 2)***: Does a domain expert come up with more creative requirements by using detailed Use Case in familiar domain than unfamiliar domain?

***Research Question 3 (RQ 3)***: Does experts needs some tools/process to come up with creative solution?

## 4.1. Experiment procedure

We decided to utilize Concept Maps to judge the creativity in finding solution. A very detailed concept map showing requirement flow is created by a thorough discussion between a Graduate Professor at NDSU and a student with 4 years of Software industry experience. Graduate professor and student with 4 years experience are experts in both the system used for case study in this paper. To quantify creativity each node of the detailed concept map is given a score based

on the level they are in. Each participant's sets of requirements are plotted into the detailed concept. And the cumulative score of each participants set of requirement on the concept map is considered their creativity score. The higher a score the higher is considered the creativity of requirements. Still even if two participant have same number of requirements their creativity score can differ. The lower a node in concept map a requirement fit in higher is its creativity score. So if a requirement satisfies a node at level 9 and another requirement satisfies node at level 5. The one at level 9 is considered more complex, because to reach that node you need to think about alternative and exception scenarios, so higher is the creativity score. Creativity refers to "the ability to produce new and original ideas and things" [1]. In a software industry to come up with creative solution(s) you have to consider all the scenarios. By all the scenarios I mean not only looking at what needs to be done\achieved but also looking at alternative solution(s), exception flow(s).

In software and systems engineering, a use case is a list of steps, typically defining interactions between a role (known in UML as an "actor") and a system, to achieve a goal. The actor can be a human or an external system.

Each use case describes one way the system is used, but one of the biggest benefits of use case modeling is that it also describes all of the things that might go wrong. Identifying exceptions to a successful scenario early in the project saves a lot of time during later stages of Software Development Life Cycle.

Finally, once a use case model has been developed, it can be used to drive many other aspects of software development, including project planning (cost, complexity and timing estimates), object models, test case definitions, and user documentation. Properly (considering exception and alternative flows) defined Use Cases help you define the right functionalities.

## 4.2. Participating subjects

Nine (9) Computer Science graduate students enrolled in the Software Design course at North Dakota State University in spring 2011 participated in this experiment. These students were predominantly masters and PhD computer science students and had taken requirement engineering course prior to this study.

## 4.3. Artifact

During this two days experiment, the participants were given a problem description on ATM machine and Communication process software and were asked to write functional requirements. Details of the study is under Appendix section (9.1 and 9.2) of this paper.

## 4.4. Procedure

Each individual was given a set of requirements from familiar domain (ATM) and an unfamiliar domain. And were asked to write the functional requirements. Each participant was required to list all different use cases and the requirements related to each of those use cases. Then, analyze each use case (in a rough corner) to come up with additional set of requirements. Detailed use case approach was used to help participant generate more requirements. Because during detailed use case creation Alternative, Exception scenarios are also considered and that leads to more requirements. The correctness of the solution/requirements was evaluated by a Software Engineering Graduate professor at NDSU and a Graduate student having an industry experience of 4 years in Software industry. All the participants' students knew Use Case text very well.

Each participant's requirements were placed in a detailed concept map to find the nodes of concept map covered by each participant's set of requirements. The detail map was created by a thorough discussion between a Graduate Professor at NDSU and a Graduate student with 4 years of industry experience. The professor and student were expert in both 'ATM' system and 'Production Verification Request system'. Once the detailed concept map was ready for each system. A score was provided to each node of concept map. The topmost node had value 1. And for each level the value of the node was increased by 1. Each node of the concept map represents a requirement for the system. The lower node in the concept map represents more specific requirements that their parents. So if a participants has two requirements and if one requirement satisfies node at level 10 and the other satisfies node at level 5, so the total score the participant gets is 15. In duplicate cases, where two requirements from same participant satisfies same node (say node 5). Participant gets the score 5. One assumption was made that all node at the same level have same level of importance/complexity. So, each node at level 5 has value 5.

The scoring of requirement for each participant was divided into two Phases. In Phase I, all the requirements gathered from problem description were plotted into each node of the concept map. And a score was calculated as summation $\sum$ (of each covered node value). In Phase II, the requirements gathered from detailed Use Case text were also plotted into the concept map and the score was calculated as summation $\sum$ (each node value covered by requirement created by use case). Some participant had few incorrect/ambiguous requirement. Those requirements were not considered for scoring.

So higher the cumulative score on concept map, higher is the user creativity. So, for software requirement gathering we consider creativity as coming up with lot of requirements to

cover all obvious scenarios as well as alternative and exception scenarios. Because anyone can come up with obvious requirements but the one who comes up with all obvious requirements and alternative and exception requirements is the one considered creative. Figure 5 and Figure 6 shows the detailed concept map used to plot each participant's set of requirements in familiar and unfamiliar domain.

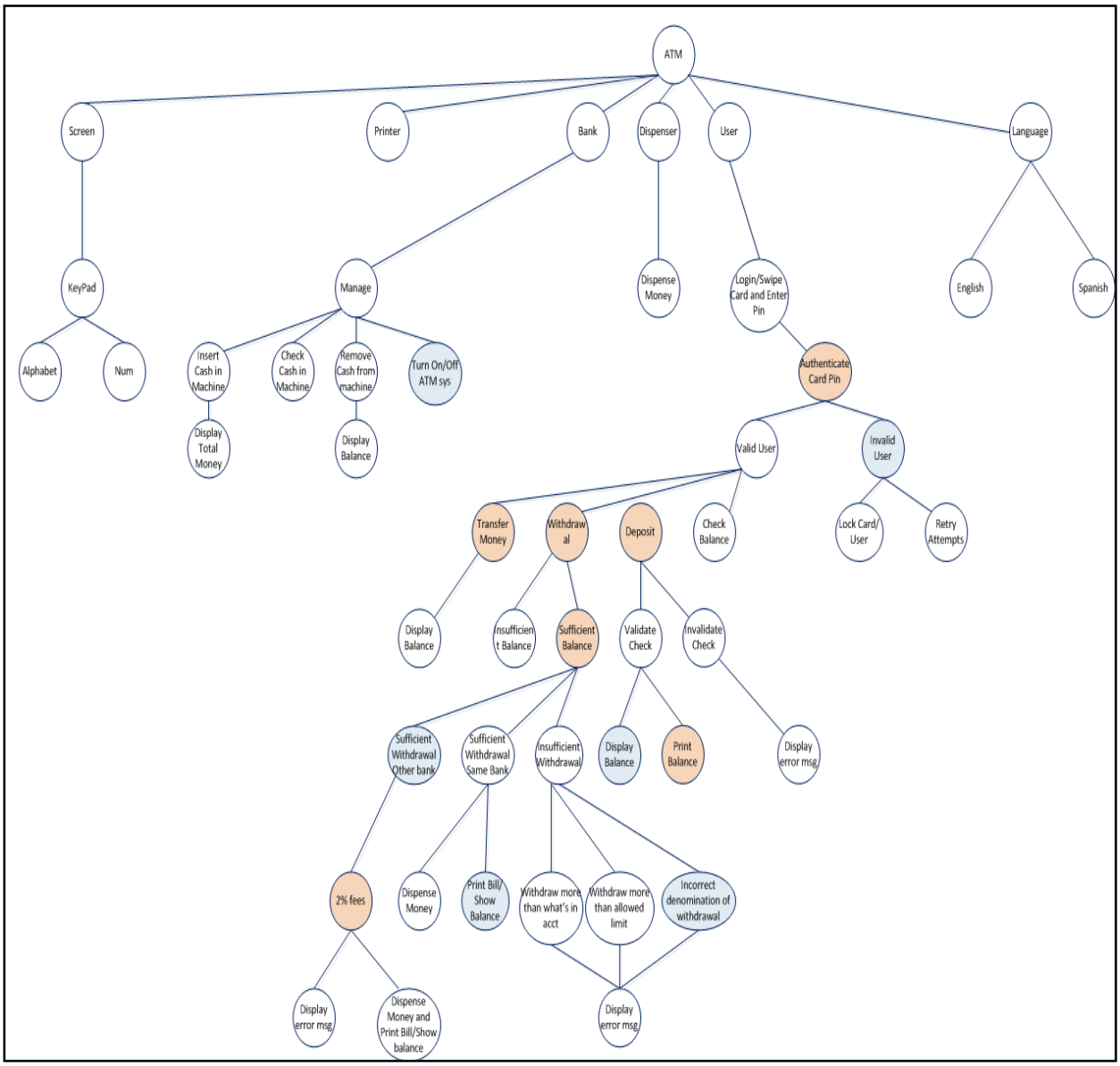


Figure 5. Detailed mental map of a familiar domain system

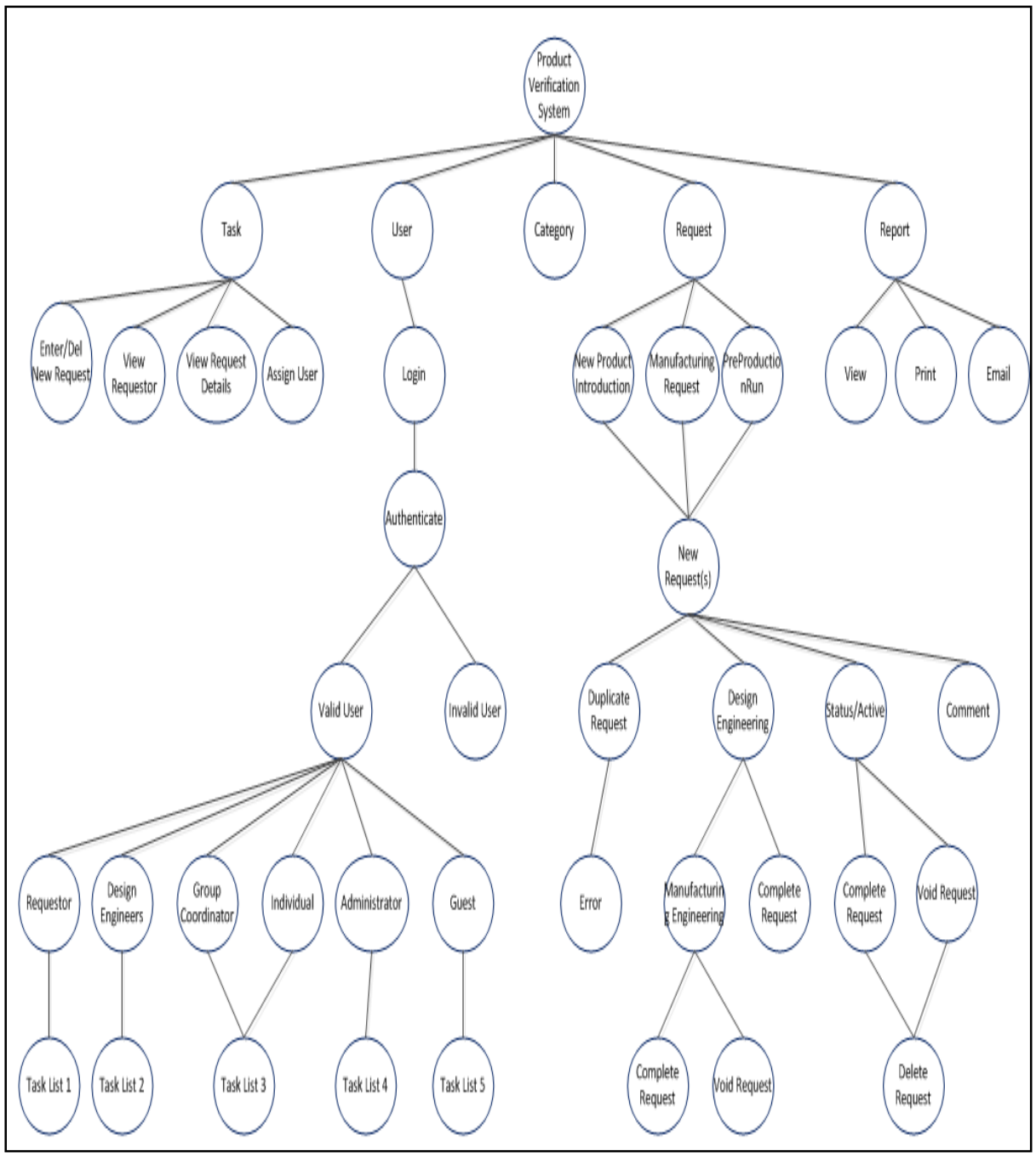


Figure 6. Detailed mental map of unfamiliar domain system

The below test lists gives details about the task lists used in Figure 6.

**Task List 1**

- Enter new request.
- Delete Request.
- View Requestor.
- View Request task details.
- View all request status.
- Add comment to any task.
- Send Email about Request.
- View/Print/Email applicable reports.

**Task List 2**

- Task List 1.
- Mark task needed or not needed.
- Assign responsibility to a task and notify individuals about the responsibility.
- Add new Design Engineering task(s) to a request.
- Maintain design engineering task lists.

**Task List 3**

- Task List 1 and Task List 2.
- Maintain their task group status information.
- Maintain their group task list information.

**Task List 4**

- Task List 1, Task List 2 and Task 3.
- View as Requestor or Individual.
- Main view same as Group Coordinators.

**Task List 5**

- View request status.
- View request task details.
- Add comments to any task.
- View/print/email applicable reports

# 5. RESULTS

In the below tables, Table 2 contains familiar domain results and Table 3 contains the unfamiliar domain results (number of requirements gathered).

Table 2. Number of requirements in familiar domain

| Student | Number of Requirements from Problem description | Number of Requirement from Use Case |
|---|---|---|
| 1 | 7 | 6 |
| 2 | 10 | 3 |
| 3 | 14 | 2 |
| 4 | 8 | 3 |
| 5 | 10 | 9 |
| 6 | 10 | 4 |
| 7 | 7 | 5 |
| 8 | 8 | 3 |
| 9 | 8 | 2 |
| Total | 82 | 38 |
| FINAL Requirements 82+38 = 120 | | |

Table 3. Number of requirements in unfamiliar domain

| Student | Number of Requirements from Problem description | Number of Requirement from Use Case |
|---|---|---|
| 1 | 7 | 4 |
| 2 | 10 | 2 |
| 3 | 8 | 2 |
| 4 | 7 | 1 |
| 5 | 16 | 2 |
| 6 | 8 | 3 |
| 7 | 6 | 3 |
| 8 | 7 | 1 |
| 9 | 10 | 3 |
| Total | 79 | 21 |
| FINAL Requirements 79+21= 100 | | |

If we look at the total number of requirement, for familiar domain the total was 120 and for unfamiliar domain it was 100. The number of requirements gathered reading problem description is almost similar for both familiar and unfamiliar domain. The interesting part of this data evolves when we look at requirements gathered with use case techniques. After applying the use case technique the number of requirements gathered by each participant is much greater in case of familiar domain compare to unfamiliar domain. For detailed Use Case, stakeholder has to consider all the Alternative, Exception scenarios. It's easier to cover these scenarios once you are familiar and experienced in a domain. The plot below shows a graphical representation of number of requirements gathered in familiar and unfamiliar domain.

The below image shows "Requirement by Problem Description". Each dot in the graph represents number of requirement gathered by each participant. The x axis represents Number of Participants and y axis represents Number of Requirements. The orange plots represent Familiar domain data and blue plots on the graphs represents unfamiliar domain.

. The data on Figure 7 shows that the number of requirements gathered for familiar domain are either equal or more of unfamiliar domain. Except one participant all rest 8 participants they gathered more requirement in familiar domain than unfamiliar.

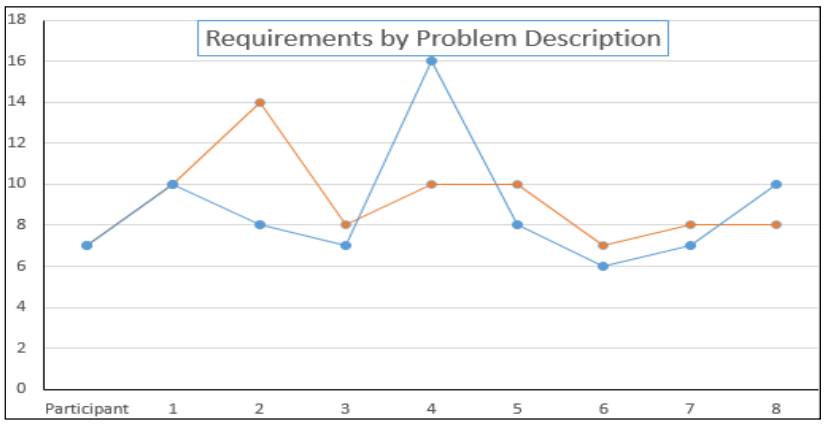


Figure 7. Requirements by problem description

The below image shows “Requirement by Use Case” graph. But Figure 8 shows the requirements gathered by each participant was almost double than what each participant gathered for unfamiliar domain. Clearly this signifies that in familiar domain the use case technique does really help to come up with lot more requirements.

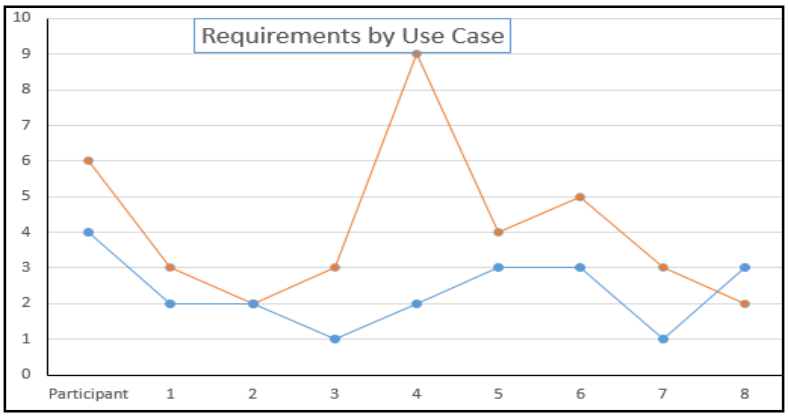


Figure 8. Requirements by Use Case

Tables 4 and 5 shows the creativity score provided for each for both familiar and unfamiliar domain. Table 4 contains familiar domain results.

Table 4. Creativity scores in unfamiliar domain

| Student | Score (Phase I Problem Description) | Score (Phase II Use Cases) |
|---|---|---|
| 1 | 60 | 108 |
| 2 | 48 | 64 |
| 3 | 44 | 50 |
| 4 | 19 | 40 |
| 5 | 44 | 65 |
| 6 | 25 | 61 |
| 7 | 17 | 44 |
| 8 | 36 | 59 |
| 9 | 26 | 53 |
| Total | 319 | 582 |
| FINAL Score 319+582 = 901 | | |

Table 5 below contains the unfamiliar domain results.

Table 5. Creativity scores in familiar domain

| Student | Score (Phase I Problem Description) | Score (Phase II Use Cases) |
|---|---|---|
| 1 | 27 | 37 |
| 2 | 49 | 56 |
| 3 | 52 | 60 |
| 4 | 49 | 49 |
| 5 | 46 | 51 |
| 6 | 62 | 69 |
| 7 | 30 | 40 |
| 8 | 23 | 23 |
| 9 | 40 | 44 |
| Total | 378 | 429 |
| FINAL Score 378+429= 807 | | |

Looking at the Final creativity score for each phase (I and II) in each domain (familiar and unfamiliar). It clearly shows that the requirements that user chose on Unfamiliar domain have more creative score, thus they cover more complex scenarios. Even if Table 1 and Table 2 shows the number of requirements gathered following problem description is more in case of familiar domain, still the creative score is low compared to creative score of unfamiliar domain. But when we consider the detail Use Case approach it clearly show the number of requirement and the creative score is much higher in familiar domain compare to unfamiliar domain. For detailed use case approach the number of requirements gathered for familiar domain became 38 whereas it is much less 21 for unfamiliar domain. And the same pattern is seen for creativity score.

The below two graphs shows the creativity score graph for both Phase I and Phase II of familiar and unfamiliar domain. The orange dots signifies creativity score in familiar domain and the blue dot is for unfamiliar domain.

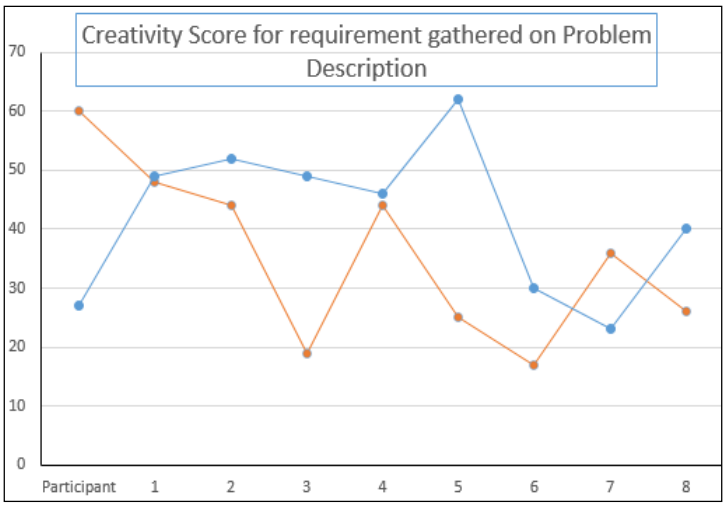

Figure 9. Score by problem description

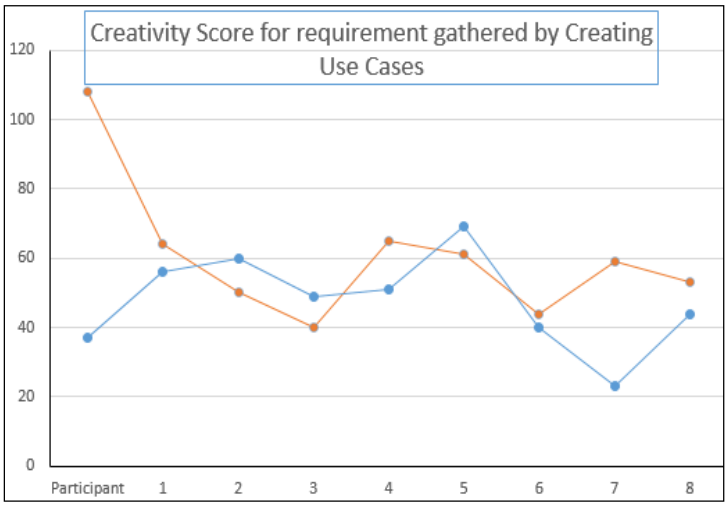

Figure 10. Score by use case

So the creativity score graph for requirements gathered on problem descriptions shows that out of 9, 6 participants scored higher in creativity of finding requirements. So, using problem description technique, 66.66% of participants had higher creative score in the domain they were

not familiar with. But, with inclusion of detailed use case technique 77.77% of participants had higher creative score in familiar domain.

## 5.1. Anova analysis

### 5.1.1. Phase I

A one-way ANOVA was conducted (Table 6 and Table 7) to compare if there was a significant difference between familiar problem description creativity score (FAM_PD) and unfamiliar problem description creativity score (UNFAM_PD). Similar analysis was conducted (Table 8 and Table 9) to compare the creativity score as a result of utilizing the use cases between familiar (FAM_UC) and unfamiliar domains (UNFAM_UC).

The analysis for FAM_PD and UNFAM_PD revealed that people in unfamiliar group exhibited statistically higher significant score (M=20.58, SD=6.36) when compared to scores obtained in familiar domain (M=12.57, SD=5.19) with F (16) =8.58 at p=0.01 (<0.05 level). Detailed statistics are provided in Table 1 and Table 2.

Table 6. Anova analysis showing mean value for familiar domain

| Category | N | Mean | Std. Deviation | Std. Error |
|---|---|---|---|---|
| Familiar PD | 9 | 12.5690 | 5.19389 | 1.73130 |
| Unfamiliar PD | 9 | 20.5882 | 6.36312 | 2.12104 |
| Total | 18 | 16.5786 | 6.98366 | 1.64606 |

Table 7. Anova analysis showing p value for familiar domain

| Score | Sum of Squares | Degree of Freedom | Mean Square | F | P Value |
|---|---|---|---|---|---|
| Between Groups | 289.390 | 1 | 289.390 | 8.579 | 0.010 |
| Within Groups | 539.726 | 16 | 33.733 | | |
| Total | 829.116 | 17 | | | |

So the statistical data shows an interesting result that people can be creative in unfamiliar domain. This can be advantageous in software industry. This can imply that if people from different domain are involved in software gathering phase of a system/software, they will come up with more creative requirements than people familiar to the domain. This data can lead to another research topic of identifying how the group creativity will be when people from familiar and unfamiliar domain worked together in requirement gathering.

The statistical data also answers the first research question we have "Does a domain expert come up with more creative requirements by reading problem description in unfamiliar domain than familiar domain?" With the data it clearly shows that domain experts come up with more creative requirements by reading problem description in unfamiliar domain.

### 5.1.2. Phase II

One-way ANOVA between FAM_UC and UNFAM_UC revealed that people in familiar group exhibited statistically higher significant score (M=8.86, SD=4.22) when compared to scores obtained in unfamiliar domain (M=2.77, SD=1.85) with F (16) =15.67 at p=0.001 (<0.05 level). Detailed statistics are provided in Table 3 and Table 4.

Table 8. Anova analysis showing mean value for unfamiliar domain

| Category | N | Mean | Std. Deviation | Std. Error |
|---|---|---|---|---|
| Familiar UC | 9 | 8.8652 | 4.22566 | 1.40855 |
| Unfamiliar UC | 9 | 2.7778 | 1.85045 | 0.61682 |
| Total | 18 | 5.8215 | 4.45237 | 1.04943 |

Table 9. Anova analysis showing p value for unfamiliar domain

| Score | Sum of Squares | Degree of Freedom | Mean Square | F | P Value |
|---|---|---|---|---|---|
| Between Groups | 166.758 | 1 | 166.758 | 15.672 | 0.001 |
| Within Groups | 170.243 | 16 | 10.640 | | |
| Total | 337.001 | 17 | | | |

When detailed use case technique is used to come up with more requirements the stakeholders have to consider alternative and exception scenarios. And the result implies that the expertise in a domain helped stakeholders to consider those scenarios to come up with more creative requirements. Domain experts creative requirements significantly improved using detailed use case. The statistical data also implies that using detailed use case tool can be helpful to come up with creative requirements. For this paper, to keep our focus on quantifying creativity, we only used detailed use case tool. So these results also motivate to compare stakeholder's creative requirement gathering using different tools and find out which tool can be best to use to come up with creative requirements.

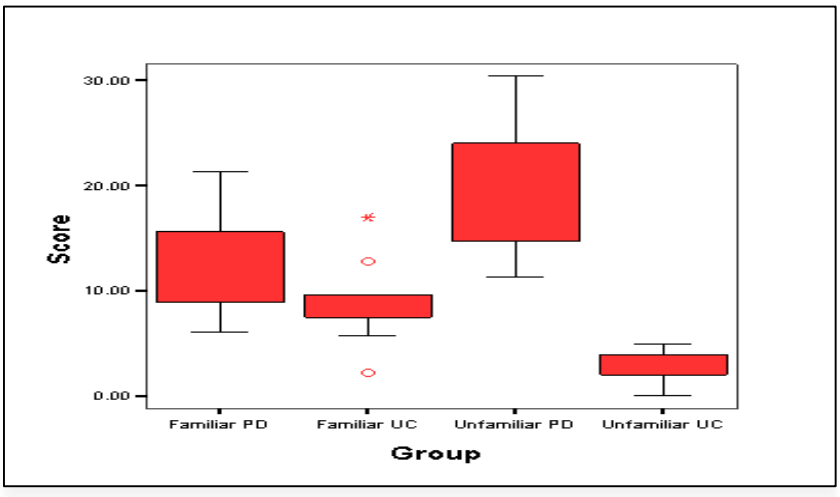


Figure 11. Data distribution chart

The statistical data also answers the second research question we have above “Does a domain expert come up with more creative requirements by using detailed Use Case in familiar domain than unfamiliar domain?” With the data it clearly shows that domain experts come with more requirements in familiar domain.

# 6. RELEVANCE OF RESULT FOR SOFTWARE ORGANIZATION

This study gives us a fair idea of how the creativity of an individual varies between the domains he/she is familiar with. The number of requirements gathered by reading problem description were more in case of familiar domain than unfamiliar domain, still the creativity score was much less in familiar domain. So only number of requirements gathered cannot always be considered as creativity parameter. But the Data shown in Table 4 and Table5 clearly shows the importance of detailed use case in requirement phase. It can really help consider gathering the alternative and exception scenario requirement and hence help a software organization be more creative in gathering requirements. This research also shows that even if a person is an expert in a domain if they follow detailed use case, they can greatly increase their creativity in requirement phase. So this study can lead to another study to find out if an organization switch people among different domains will they be more successful with innovation/creativity or not. Will it be beneficial for the organization or not.

# 7. DISCUSSION OF FINDINGS

This study do shows that domain experts can do a better job of finding requirements in their domain compare to unfamiliar domain. This study also showed that using detailed use case approach which makes sure, one is considering alternative and exceptional scenarios for each use case did help in finding lot of new requirements for both familiar and unfamiliar domain. Still the number of requirements gathered in familiar domain using use case template approach was almost the double of unfamiliar domain. This result also shows a greater importance and impact of detailed use case to gather more requirements.

# 8. CONCLUSION

So to answer the research question in Section 3 we can certainly say that for research question 1 , yes domain experts comes up with more creative solution/requirements in unfamiliar domain. And for research question 2, if domain experts use tools/technique like detailed use case they can certainly come up with lot more complex requirements which if left in initial stages can be very costly during later stages of software development. And for research question 3, using detailed use case tool really helped in both familiar and unfamiliar domain. Specially for familiar domain the creativity of requirements significantly improved. So this research also depicts the impact detailed use case can have in requirements gathering phase of software development.

# REFERENCES


1. N. Bonnardel. *Creativity in Design Activities: The Role of Analogies in a Constrained Cognitive Environment*. 1999.
2. M. Terry, E.D. Mynatt. *Recognizing Creative Needs in User Interface Design*. 2002.
3. U. Farooq, J.M. Carroll, C.H. Ganoe. *Supporting Creativity in Distributed Scientific Communities*. 2005.
4. A. Furnham. *The Brainstorming Myth*. 2000.
5. M.Bailey, C. Coats, K. Hamilton. *Understanding Knowledge Management Practices for Early Design Activity and Its Implications for Reuse*. 2009.
6. L. Gabora. *Cognitive Mechanisms Underlying the Creative Process*.
7. E.L. Santanen, R.O. Briggs, G. de Vreede. *A Cognitive Network Model of Creativity: A Renewed Focus on Brainstorming Methodology*.
8. B. Rolfe. *On the Production of Creative Subjectivity*. 2009.
9. R. Pressman. *Software Engineering: A Practitioners Approach (4th edition).*
10. Jennifer Wiley. *Expertise as mental set: The effects of domain knowledge in creative problem solving , Pennsylvania 1998,26 (4), 716-740*
11. The role of domain knowledge representation in requirements elicitation
12. J. Kato, M. Saeki, A. Ohnishi, M. Nagata, H. Kaiya, S. Komiya, S. Yamamoto, H. Horai, and K. Watahiki. PAORE: Package Oriented Requirements Elicitation. In Proceedings of 10th Asia-Pacific Software Engineering Conference (APSEC 2003), pages 17–26, Chiang Mai, Thailand, Dec. 2003. IEEE Computer Society Press.
13. A. Osada, D. Ozawa, H. Kaiya, and K. Kaijiri. Modeling Software Characteristics and Their Correlations in A Specific Domain by Comparing Existing Similar Systems. In
14. K.-Y. Cai, A. Ohnishi, and M. F. Lau, editors, QSIC 2005, Proceedings of The 5th International Conference on Quality Software, pages 215–222, Melbourne, Australia, Sep. 2005.IEEE Computer Society.
15. Neil Maiden, Suzanne Robertson & Alexis Gizikis Centre for HCI Design, City University, London Atlantic Systems Guild, London Provoking Creativity: Imagine What Your Requirements Could be Like.

# APPENDIX

## A.1. Case study 1

Please go through the problem description below and write a set of functional requirements for the system below. Then think (draw some use cases in rough) about the different use cases for the system and add more requirements that came to your mind while writing use case(S). While writing use case (see a detailed use case template in Appendix below) we consider alternative and exception flows and that can lead to more requirements.

Alternative flows example: Amount Exceeds Withdrawal Limit, Amount Exceeds Daily. Withdrawal Limit

### A.1.1. Problem description

***ABC*** Bank wants to develop software for operating ATM Machine in 10 different States in United States. Customers of different banks can do a transaction using this ATM machine. ATM machine will only let one customer do a transaction at a time. However the customers of other banks (other than ABC Bank) would be charged 2% transaction fees of the transaction amount. ATM machine will have a display screen for customer interaction. It will also have a magnetic stripe so that customer can swipe his/her debit card/credit card. It will have a keypad to input the choices and the amount for the transaction. It will also have a dispenser for withdrawal of cash (in multiples of 20). There will be a printer to print the transaction details. Customers can receive the receipts for transactions done.

Customer will have to insert valid card and personal identification number (PIN) for authorization purposes. Once the customer is authorized as a valid customer then he/she can

perform as many transactions as he can. ATM Machine will return card back to customer only when customer indicates that no further transactions are required.

Different Entities involved in the system are:

1. Customer – Any person who would interact with ATM Machine for performing transactions.
2. Bank- Bank is financial entity that validates the customer. It also validates each transaction with regards to sufficient funds to perform transaction.
3. Operator- Operator is a person who would interact with ATM Machine to start and stop ATM Machine. He/she is responsible to make sure there is sufficient cash in ATM Machine. If there are insufficient funds he reloads the machine with cash. He also removes all the deposit envelops from ATM Machine.

### A.1.2. Tasks

1> Create a set of Functional requirements on the basis of above problem description. Please follow the below template to record the requirements.

| Req. ID | Description |
|---|---|
| 0100 | A customer must be able to make a transfer of money between any two accounts linked to the card. |

2> Among the number of new requirements you Added

a> List the requirements you created/added from your Understating of ATM machines.

b> List the requirements you created/added while you were writing use case(for example, while writing Alternative flow or Exception flow you realize a

scenario that can happen and you realized that there is no requirements for this scenario ).

## A.2. Case study 2

Please go through the problem description below and write a set of functional requirements for the system below in a similar fashion as used for the ATM system.

### A.2.1. Problem description

The main goal of this project is to create a central repository for all communication about a product or process development so that comments, task status and work order status is available to all stakeholders involved.

The new Production Verification Request system project will assist Daktronics' design, production and process development team members in making all of the product and process development tasks performed in the design, pre-production and testing of packet assemblies transparent. It will assist the employee communication from initial design to the development process initial, through preproduction, engineering and ultimately to the stage where the assembly is fully qualified for standard production.

Current project and process communication is inconsistent. The communication that exists today is via emails, phone calls and meetings. Not all the people that need this communication are included in the emails, phone calls or meetings. Also, it is time consuming to hunt for emails, make multiple fact finding phone calls or attend multiple update meetings. Many projects are emergencies or late because of this lack of communication. If this system is developed, it will no doubt benefit entire organization.

*Business Requirement*

There will be six categories of users of this system: Requestors, Design Engineers, Group Coordinators, Individuals, Administrators and Guests. The users in all categories will be required to securely logon to the system.

Requestors (Project Coordinators) are allowed to enter new requests, maintain request information and status, delete requests, view the status of all requests in the system, view the request task details, add comments to any task, send emails about the requests and request tasks, and view/print/email applicable reports.

Design Engineers are allowed to enter new requests, maintain request information and status, delete requests, view the status of all requests in the system, mark tasks and needed and not needed within their group, assign responsibility to tasks and notify individuals of their responsibility, view the request task details, add new design engineering tasks to a request after it has been released, maintain their own task status information, send emails about the requests and request tasks, add comments to any task, maintain the Design Engineering task list information, and view/print/email applicable reports.

Group Coordinators are allowed to view the status of all requests in the system, mark tasks and needed and not needed within their group, assign responsibility to tasks and notify individuals of their responsibility, view the request task details, add new group specific tasks to a request after it has been released, maintain their own task status information, maintain their task group's tasks status information, send emails about the requests and request tasks, add comments to any task, maintain their own group's task list information, and view/print/email applicable reports.

Individuals are allowed to view the status of all requests in the system, mark any tasks assigned to them as not needed, view the request task details, maintain their own task status

information, maintain their task group's task status information, send emails about the requests and request tasks, add comments to any task, and view/print/email applicable reports.

Administrators are allowed to do anything that a Requestor, Design Engineer, Group Coordinator, or Individual is allowed to do. They are allowed to view the system as a Requestor would and as an Individual would. Their main view of the system should be the same as a Group Coordinator. They are also allowed to maintain users, locations, business units, groups and task groups.

Guests are allowed to view the status of all requests in the system, view the request task details, add comments to any task, and view/print/email applicable reports.

The Production Verification Request system will allow new pre-production work order requests to be entered into the system. These requests will be categorized as one of the following: a new product introduction (NPI), a manufacturing request for a mature product (MR – PCA needed for engineer testing, MN – PCA design needs to change a component to a new non-existing Daktronics part, or MD - PCA design needs to change a component to a different existing Daktronics part), a pre-production run (PPR) or a production verification (PV). This system will not be used for standard production work orders. Any request activity (newly entered, change of information, deletion of a request, Design Engineering task release or Manufacturing Engineering task release) will prompt the system to ask the user if they would like to send a message to the Group Coordinators. A new request will automatically have a status of active and will remain active until a Requestor, Design Engineer or Administrator deletes the request, or changes the status to Void or Complete. Once a request has been released to manufacturing, it cannot be deleted – only voided or completed. The request must be released to Design Engineering before it can be released to Manufacturing Engineering. When a request is released to Design

Engineering, all tasks are populated to the request and the tasks appropriate to the type of request for Design Engineers are marked as needed. When a request is release to Manufacturing Engineering, the tasks appropriate to the type of request for all manufacturing groups are marked as needed.

Once a request's tasks have been released the request tasks are available to be maintained in the system.

### A.2.2. Tasks

1> Create a set of Functional requirements on the basis of above problem description. Please follow the below template to record the requirements.

| Req. ID | Description |
|---|---|
| 0100 | Shall implement a secure system logon |

2> Among the number of new requirements you Added

a> List the requirements you created/added from your Understating of ATM machines.

b> List the requirements you created/added while you were writing use case(for example, while writing Alternative flow or Exception flow you realize a scenario that can happen and you realized that there is no requirements for this scenario ).

## A.3. Detailed use case template

### A.3.1 Use case

| | | | |
|---|---|---|---|
| **Use Case ID:** | **Enter a unique numeric identifier for the Use Case. e.g. UC-1.2.1** | | |
| **Use Case Name:** | **Enter a short name for the Use Case using an active verb phrase. e.g. Withdraw Cash** | | |
| **Created By:** | | **Last Updated By:** | |
| **Date Created:** | | **Last Revision Date:** | |
| **Actors:** | **[An actor is a person or other entity external to the software system being specified who interacts with the system and performs use cases to accomplish tasks. Different actors often correspond to different user classes, or roles, identified from the customer community that will use the product. Name the actor that will be initiating this use case (primary) and any other actors who will participate in completing the use case (secondary).]** | | |
| **Description:** | **[Provide a brief description of the reason for and outcome of this use case.]** | | |
| **Trigger:** | **[Identify the event that initiates the use case. This could be an external business event or system event that causes the use case to begin, or it could be the first step in the normal flow.]** | | |
| **Preconditions:** | **[List any activities that must take place, or any conditions that must be true, before the use case can be started. Number each pre-condition. e.g.**<br><br>1. **Customer has active deposit account with ATM privileges**<br>2. **Customer has an activated ATM card.]** | | |
| **Postconditions:** | **[Describe the state of the system at the conclusion of the use case execution. Should include both minimal guarantees (what must happen even if the actor's goal is not achieved) and the success guarantees (what happens when the actor's goal is achieved. Number each post-condition. e.g.**<br><br>**Customer receives cash**<br>**Customer account balance is reduced by the amount of the withdrawal and transaction fees]** | | |

| | |
|---|---|
| **Normal Flow:** | **[Provide a detailed description of the user actions and system responses that will take place during execution of the use case under normal, expected conditions. This dialog sequence will ultimately lead to accomplishing the goal stated in the use case name and description.**<br><br>**Customer inserts ATM card**<br>**Customer enters PIN**<br>**System prompts customer to enter language performance English or Spanish**<br>**System validates if customer is in the bank network**<br>**System prompts user to select transaction type**<br>**Customer selects Withdrawal From Checking**<br>**System prompts user to enter withdrawal amount**<br>**…**<br>**System ejects ATM card]** |
| **Alternative Flows:**<br>**[Alternative Flow 1 – Not in Network]** | **[Document legitimate branches from the main flow to handle special conditions (also known as extensions). For each alternative flow reference the branching step number of the normal flow and the condition which must be true in order for this extension to be executed. e.g. Alternative flows in the Withdraw Cash transaction:**<br><br>**4a. In step 4 of the normal flow, if the customer is not in the bank network**<br>**System will prompt customer to accept network fee**<br>**Customer accepts**<br>**Use Case resumes on step 5**<br><br>**4b. In step 4 of the normal flow, if the customer is not in the bank network**<br>**System will prompt customer to accept network fee**<br>**Customer declines**<br>**1. Transaction is terminated**<br>**2. Use Case resumes on step 9 of normal flow**<br><br>**Note: Insert a new row for each distinctive alternative flow. ]** |
| **Exceptions:** | **[Describe any anticipated error conditions that could occur during execution of the use case, and define how the system is to respond to those conditions.** |

| | |
|---|---|
| | **e.g. Exceptions to the Withdraw Case transaction**<br><br>**2a. In step 2 of the normal flow, if the customer enters and invalid PIN**<br>**Transaction is disapproved**<br>**Message to customer to re-enter PIN**<br>**Customer enters correct PIN**<br>**Use Case resumes on step 3 of normal flow]** |
| **Includes:** | **[List any other use cases that are included ("called") by this use case. Common functionality that appears in multiple use cases can be split out into a separate use case that is included by the ones that need that common functionality. e.g. steps 1-4 in the normal flow would be required for all types of ATM transactions- a Use Case could be written for these steps and "included" in all ATM Use Cases.]** |
| **Frequency of Use:** | **[How often will this Use Case be executed. This information is primarily useful for designers. e.g. enter values such as 50 per hour, 200 per day, once a week, once a year, on demand etc.]** |
| **Special Requirements:** | **[Identify any additional requirements, such as nonfunctional requirements, for the use case that may need to be addressed during design or implementation. These may include performance requirements or other quality attributes.]** |
| **Assumptions:** | **[List any assumptions that were made in the analysis that led to accepting this use case into the product description and writing the use case description.**<br>**e.g. For the Withdraw Cash Use Case, an assumption could be:**<br>**The Bank Customer understands either English or Spanish language.]** |
| **Notes and Issues:** | **[List any additional comments about this use case or any remaining open issues or TBDs (To Be Determined) that must be resolved. e.g.**<br><br>**1. What is the maximum size of the PIN that a use can have?]** |