2012

A Petri-Net Based Approach of Software Visualization for Software Customization

Vida Sadri
University of Windsor

A Petri-Net Based Approach of Software Visualization for Software Customization

by

Vida Sadri

A Thesis
Submitted to the Faculty of Graduate Studies
through the School of Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Master of Science at the
University of Windsor

Windsor, ON, Canada

2012

A Petri-Net Based Approach of Software Visualization for Software Customization

by

Vida Sadri

APPROVED BY:

____

External Reader
Dr. Gokul Bhandari
Odette School of Business

____

Internal Reader
Dr. Luis Rueda
School of Computer Science

____

Advisors
Dr. Jessica Chen
Dr. Xiaobou Yuan
School of Computer Science

____

Chair of Defense
Dr. Boubakeur Boufama
School of Computer Science

DECLARATION OF ORIGINALITY

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

ABSTRACT

Different from the traditional approach of software development from scratch, Software Product Line (SPL) allows software customization. When further supported by Service-Oriented Architecture (SOA), SPL offers unprecedented advantages for reusing software artifacts in mass customization of software applications, leading to radically reduced time, cost, and effort of software development. Accordingly, an interactive dialogue-based system for ontology-based requirement elicitation has been developed previously, in our research group, by Zhang [19].

This thesis works on enhancement of the prior work by introducing software visualization to the process of interactive requirement elicitation. A research was conducted for choosing the most suitable visualization method for the existing text-based software. For this purpose, a layered structure for SOA visualization with support of Petri Nets is chosen. Accordingly, this method was implemented and a usability study was done to validate improvements in comprehension of the end-user in visualized version comparing to the previous version of requirement elicitation system.

DEDICATION

To my beloved parents for their unconditional love and support.

ACKNOWLEDGEMENTS

My ultimate gratitude goes to my supervisors, Dr. Xiaobu Yuan and Dr. Jessica Chen, for their continuous advise and support throughout my thesis. Their insightful feedback and instructions made it possible for me to accomplish this work.

I would like to acknowledge Dr. Luis Rueda and Dr. Gokoul Bhandari whose suggestions and recommendations greatly improved the quality of this work.

My special thanks goes to my parents and my family for their patience and love they provided to me during all times.

I express my deep appreciation to my friend Navid, for his encouragement, motivation and moral support he provided during all stages of my higher education. He made this work so meaningful to me.

Many thanks to my good friends Ahmad Tavakoli and Ali Karaki for their valuable and helpful ideas on this work.

I would like to thank students of Computer Science and Business departments who participated in the study of my thesis. I am deeply grateful for the time and effort they spent on the test.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

1.1. Introduction

In software development process there is a need to cut down the cost, effort and market of software products. The size and complexity of software systems are increasing day by day and it is essential that the developed software be flexible enough to adapt to market changes and new technologies [1]. Therefore, improving reusability and hence quality will increase the productivity of software teams and reduce the time and cost to market new software products. Product assembly from components, reducing labour intensive tasks with automation setting up the software product lines and supply chains and standardizing the interfaces, architectures and processes can bring the software industry to a much higher productivity [2]. This can be done by Software Product Line, which produces a set of distinct but similar products, and Service Oriented Architecture, which solve the integration and interoperability problems, together. They both promise the development of flexible, cost-effective and reusable software systems [3].

On the other hand, to make the process of software development less complex and more understandable for software developers it is essential to use some tools for supporting the tasks which are included in software development process specially for the most curtail phase which is the requirement elicitation phase. One possible aspect for such a support is Software Visualization (SV). Recently so many software visualization techniques and tools are available but it is critical to choose the most suitable one for a suitable activity in software development process to do the most effective visualization for a specific software system [4].

1.2. Problem Statement

It is discussed in many papers [1, 2, 5, 6, 7], that when SOA concept is used in Software Product Line, it will make a mass customization of software application by reusing software artefacts which can be very beneficial, specially time-wise and effort-wise. Based on this fact, beforehand, an interactive text-based dialogue interface has been developed in our research group [19], which interacts with the user who wants to do software development, in natural language and does the requirement elicitation process automatically based on the ontology behind it. The ontology represents the knowledge of the product features as well as their business logic. It represents the commonalities and variabilities among a group of related artefacts and in this way it directs the dialogue system to perform requirement elicitation [19].

The problem here is, the mentioned idea seems beneficial in theory but it is not proved that its usage in industry can be advantageous or not. It is needed to be evaluated in practice on different real users, to be formally certified as an applicable, usable technique.

On the other hand, the text-based dialogue system can be enhanced. The software developer needs to have an overall overview of the system that the text-based system lacks in having this feature. Also, the user needs to know what steps are already done and what steps are left. Reading all long comments on the dialogue system and understating all of them and keeping them all in mind in order to do software development can be a tough and time consuming job. This process will need several activities such as reading, reasoning and constructing mental models. Consequently, there is a need for a more optimal solution.

As described earlier, software visualization can be noticeably beneficial when it is used in different phases of the process of software development. Based on [31] graphical representation of information comparing to other presentation methods consumes user's information-processing capabilities much more efficiently. If the graphic presentation is used properly, it reduces the need for mental information recording and reorganization and also it decreases the memory load. On the other hand, the interaction between human and the machine would be much faster and simpler and on the whole appealing for the users [31]. Therefore, different software visualization techniques should be studied and the best one for this type of system should be chosen. The chosen, proposed method can be implemented as an additional user-interface.

At the end, another evaluation should be conducted in order to justify the usability of the visualized part of the project to prove if it really enhances the comprehension of the system for the users and even advances in usability factors which are, learnability, efficiency, error rate and user satisfaction should be investigated.

1.3. Contribution

This thesis, presents a visualized user-interface for the implemented dialogue-based, interactive requirement elicitation system, with the purpose of enhancing the understanding of the user of the dialogue system and reducing time, cost and effort spent on working with the previous system and in overall increasing the usability of the system. Because both dynamic flow and concurrency of the distributed services was aimed to be illustrated in the graphical visualization, a two layered interface which shows both services in one layer and the workflow of the system in another system have been chosen. For visualizing the workflow layer, Petri-net, which is a very well known formalism for modeling the behavior of concurrent systems [25], has been decided to be used as the most suitable technique for designing the visualization of this type of system.

Both, previous text-based system and the proposed, implemented system with the graphical user interface should be evaluated for justification of, first of all, if the ontology-based requirement elicitation approach is acceptable by real users and second, if the graphical user interface improves the usability of the previous text-based system. Therefore, a usability study will be conducted at the end on real users to see if all the expected benefits will be met or not.

1.4. Structure Of the Thesis

The structure of the remainder of this thesis is as follows. Chapter II is about Software Customization, which SPL and SOA concepts will be discussed in detain in this chapter. In Chapter III, Software Visualization will be discussed. Chapter IV will report the proposed Petri Net visualization method followed by implantation and usability testing and the results of the thesis in chapter V. The final chapter of this thesis puts this project into perspective, by discussing the major results and their impact, and by providing an outlook into future work.

CHAPTER II

SOFTWARE CUSTOMIZATION

2.1. Software Product Line

Traditionally, the software used to be applied on products was very small and simple. In order to modify and produce a new product, it used to be much easier and cheaper to copy, transport or replace the software than the hardware. The main focus of generating a product was on the hardware and software did not used to play a key role in product generation [8].

However, now, software plays a very critical role in any system. The reason for that is the flexibility of software in modifying the system and also software's strength in adding a new functionality to the system, which perhaps it would be difficult to be performed without it and only by means of modifying the hardware. Therefore, in order to make the system production's process much more efficient, the concept of Software Product Line Engineering will be addressed [8].

Software Product line is a paradigm to develop software applications and software products, by building reusable parts and reusing them. For this purpose mass customization is being used which means large production of goods with taking into account the customer's individual requirements. For this purpose, we should focus on commonalities and differences in the applications (in terms of requirements, architecture, components and test artifacts) of the product line to be modeled in a common way [8]. By using SPL, some advantages can be gained such as reduction of development cost and time, enhancement of quality, coping with evolution and complexity and etc [8].

Software Product Line Engineering Paradigm consists of two processes: Domain

Engineering and Application Engineering.

Domain Engineering establishes a platform and defines commonalities and variabilities of the product line. Our main focus in this thesis is on domain engineering process [8].

Application Engineering derives the application from the platform, which is built by domain engineering [8].

Although lots of research has been conducted on benefits of using Software Product Lines for software development and how to scope and define and develop product lines but only few approaches and tools are available for product derivation and the way utilize the product line [9].

Another concept that currently gets a lot of attention in researches and in many papers is brought with the concept of SPL is Service Oriented Architecture. In some research papers [1, 2, 5, 6, 7] it is discussed about combining SPL and SOA together to improve the practical value of SPL, to make the software development process more efficient and to improve the quality of the developed software. It is believed that this combination can decrease development costs and effort, improve time to market, application customized to specific customers or market segment needs and competitive advantages [1]. The reasons for these claims will be discussed later.

2.2. Service Oriented Architecture

"Service Oriented Architecture is an information technology architectural approach that supports the creation of business processes from functional units defined as services" [10].

Services are modules of business or application functionality. SOA consists of services, which are shared and reusable on an IT network and they communicate with each other. This communication can either be held by data passing between services or by coordination of two or more services for doing a common activity [11].

The basic SOA is based on interaction between three software agents, which are called *service provider, service consumer* and *service registry.* The three operations, which are being conducted in this architecture, are find, bind and publish. Service provider develops and publishes services' descriptions and their access information in service registry. Service requester tries to find the most suitable service in the service registry and by means of available access information in service registry, will bind the required service to the service provider to invoke required services [12]. SOA is both a business strategy and a software architectural principle [39].

Service Oriented Architecture is a rapid, low cost and easy composition of distributed applications, which is the best paradigm to minimize business environment complexity and maximize the productivity [6].

The advantages of service orientation are loose coupling, abstraction and reusability of business functionalities [13].

2.3. Integration

However, SOA lacks in supporting high customization and systematic planned reuse. It means that it is possible to use certain services for software development but if any changes happen to the order or participants of service composition services, which are not designed to be highly customizable and reusable, would not support variability. Thus SPL engineering, which basically has the principle of variability, customization and systematic planned reuse, can be used to aid SOA for better functionality and achieve these benefits [1]. Furthermore, the integration of SPL and SOA concepts give the ability of reusing existing services instead of continuously developing them from scratch [44].

As a conclusion for this part the concepts of SPL and SOA are in no way mutually exclusive and where they differ they act as each other’s complement [2].

2.4. Requirement Elicitation

Traditionally the process of software development used to have little or no dependency on business processes. Programmers used to sit and write a code, which is supposed to be useful. But this method will not be useful for the larger and more complex systems. Therefore the system life cycle has been broken into smaller parts, which are called phases. Requirement engineering is the earliest phase in this process, which is typically proceeded by business planning [14].

"Requirements Engineering (RE) is the systematic process of developing requirements through an iterative cooperative process of analyzing a problem, documenting the resulting observations in a variety of representation formats, and checking the accuracy of the understanding gained." [15].

One of the essential tasks of RE during software engineering is Requirement Elicitation. Researches show that a major cause of problems in software projects is inadequate requirement engineering [8]. Consequently, the basic prerequisite of software product line, which is a software developments paradigm, is requirement elicitation process, which shows the commonalities and differences of the requirements [8].

There are different techniques that can be used for requirement elicitation. These techniques are either conversational which is mainly conducted by interviews with two or more people, observational which can be done by observing people when they are carrying out their routine job, analytic which means exploring existing documentation or knowledge gained from either conversation or observation and synthetic which is combining conversation, observation and analytic methods into a single method. In practice these techniques are not adequately applicable [16].

In [17] it is mentioned that useful, useable and desirable software products are created using interaction design. Software developers do not benefit from interaction design though. The tools that software developers use for developing are insufficient and not appealing for them. Although the importance of using Human-Computer Interaction (HCI) concept in Software Development Process (SDP) is not very clear for many software developers, HCI experts have been tried to show that the integration between these two, can cause better user satisfaction derived from a user-centered SDP [18]. However, conducting an interactive software engineering paradigm is still an issue.

One possible idea is to take advantage of both SOA and SPL concepts. SOA can be used in order to makes it easier for the software engineers by introducing services as loosely coupled software functionalities eliminating the lower-level complexity. On the other hand SPL is useful for managing the variable software engineering. In interactive software engineering, machines can be used to guide the users to select reusable software assets and implementing the candidate application by composing the ordered services [19].

The previous thesis from the same research group, which has been conducted by Zhang [19], is titled as "An Interactive Approach of Ontology-Based Requirement Elicitation". In that project a requirement elicitation approach has been proposed for SOA-based SPL engineering as a programming model for realizing the interactive requirement engineering [19].

The proposed interactive model is a dialogue-based system, which interacts with users in a natural language. The way dialogue system works is, it extracts and analyses the expressions produce by human-beings users in order to accomplish a task and generates an expression in a natural langue for the user accordingly. Therefore, dialogue system can be a convenient way for human-machine interaction.

In the previous proposed dialogue system, slot-filling tasks is considered for the requirement elicitation process, in which the user knows about the goals and has the information about doing the task. These tasks will be done based on knowledge base of the dialogue system. It is claimed in the previous proposed dialogue system that ontology-based requirement engineering is the most popular technique among all the other knowledge-driven requirement engineering techniques. Ontology represents the common knowledge within a domain. It means that it provides shared vocabulary to construct the concepts, objects and their properties and relations of a domain or a task, which can cause common understanding of the structure of information between people or software agents [40]. By using ontology, the common concepts of a domain can be defined by experts and the knowledge can be used by people with any background and without professional training [19].

To develop ontology, the concepts in the domain should be defined, and a hierarchical order should be arranged between them. The slots and the allowed clauses for those slots also should be defined. At the end the instances and the values for slots of instances should be filled [41].

The model developed in the previous project, integrates the requirement engineering knowledge with service-oriented knowledge. Since SOA encapsulates application functionalities into loosely coupled services, software applications can be implemented by discovering, composing and invoking services in SOA. The ontology of services makes automatic service discovery and composition possible [42]. In ontology there exists a class called ServiceProfile, which contains the characteristics of services and is used to match with the client's requests. It happens in this way that for the reason of discovering services, the ServiceProfile of the requestor automatically will be matched with the provider's ServiceProfile through semantic capability matching [43] and if the matching succeeds the desired services are found.

In the domain of requirement elicitation the requirements can be classified into three categories of *function, quality* and *softgoal.* Each of these categories have different roles in the system and also for all of them another factor called *rank* is defined which is needed to direct the requirement elicitation process and is expressed in the ontology model. Functions are the functionalities in the system that the user can order. Quality is a non-functional constraint that imposed on a function. Softgoals are non-functional constraints impose on the whole system environment. In between each of these three types of requirements, some relationships exist such as *generalize, decompose, rely, contradict, associate, hasRank* and *invalid*. These relationships will be discussed briefly as follows [19]:

- Generalize relationship is defined to show that an instance of function, quality and Softgoal is also an instance of requirement.
- When requirement *x* decomposes to *y, y* is a less complex requirement of the same type as *x*.
- Requirement *x* relies on requirement *y* it means that realization *x* relies on implementation of *y*.
- When requirement *x* and requirement *y* contradict it means they are not supposed to be realized with each other in the product software at the same time.

- Function *x* associates with quality *y*.
- HasRank relationship shows that requirement *x* has a unique rank *r*.
- Invalid relation ship shows that there is an invalid relationship between requirements *x* and *y*.

For instantiating the ontology model, first all the these relationships should be established between the available requirements and the following procedure will show the instantiation of the ontology [19]:

1. The main functions which are the roots of the decomposition tree will be identified
2. If any children of the root contribute to the composition with their parent, they should be decomposed by the Decompose relationship and if the children of children are also decomposable the same story should be repeated on them till there is no composition between parents and children.
3. All the quality constraints should be found and the associate relationship between children and the corresponding function should be established
4. Sofgoals should be identified and decomposed.
5. Rely and contradict relationships should be established
6. A rank should be assigned to each of the requirements based on their importance.

Based on what has been discussed a graph as Figure 2.1 will be produced.

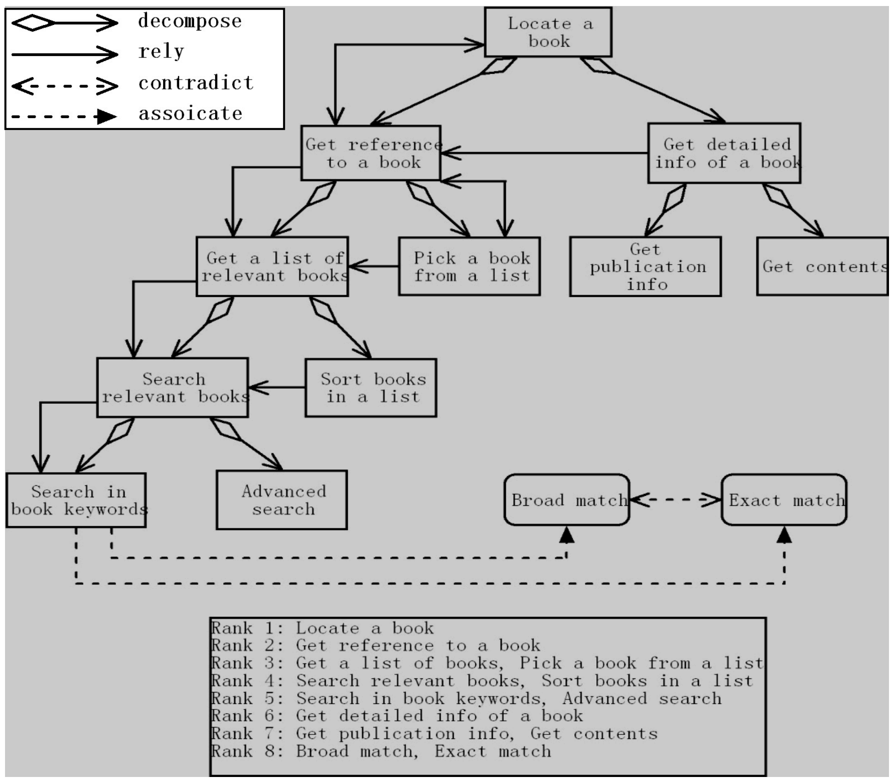

decompose
rely
contradict
assoicate
Locate a book
Get reference to a book
Get detailed info of a book
Get a list of relevant books
Pick a book from a list
Get publication info
Get contents
Search relevant books
Sort books in a list
Search in book keywords
Advanced search
Broad match
Exact match
Rank 1: Locate a book
Rank 2: Get reference to a book
Rank 3: Get a list of books, Pick a book from a list
Rank 4: Search relevant books, Sort books in a list
Rank 5: Search in book keywords, Advanced search
Rank 6: Get detailed info of a book
Rank 7: Get publication info, Get contents
Rank 8: Broad match, Exact match

Figure 2.1. Requirement model instantiated with book locating service [19]

The way the interactive requirement elicitation works is a follows:

The requirements will be offered to the user one by one based on the rank is assigned to them and the user should choose from them. If the requirement is essential it will be picked automatically and regardless user's opinion. The functions will be evaluated first and after that all the qualities and evaluation of softgoals will be followed. All the requirements will be met by the dialogue system. If the user decides to drop a requirement the requirement which has the rely relation ship with that requirement will be dropped as well. If a requirement decided to be picked by the user and another requirement has the contradict relationship with that requirement will be dropped and the requirement with the rely relationship will be picked as well [19].

CHAPTER III

SOFTWARE VISUALIZATION

3.1. Software Visualization

Software visualization means providing the image of existing software using visual objects. Software visualization might visualize different aspects of the software, such as software structure, components and even the runtime behavior of the software. It is proven that appropriate visualization can significantly reduce the effort spent on different phases of software development. By means of visualization, developers and stakeholders can obtain an overall point of view of the software structure, software logic or explain and communicate with the development process [4]. Generally, software visualization is mainly used for program behavior exhibition, logical debugging and performance debugging but it fundamentally is concerned with software comprehension [20]. By providing a good graphical representation in order to visualize the software, a better user understanding of the system can be more promising than textual representation of the software [21].

By graphical presentation of information the capabilities of user's information processing would be utilized much more effectively than other presentation methods. If a suitable graphical representation tool is chosen properly, there would be less need for perceptual and mental information recording and it would reduces the memory loads. By providing graphical interfaces, there would be a faster information transfer between computer and people. Because it has been proven that symbols can be recognized and classified faster and more precise than text by users. Also, because of its simplicity, graphics will remain in casual users' minds much easier. It also gives a better feeling of control to users when they can see objects on the screen. In overall graphical visualization can provide so many other benefits faster learning, faster use and problem solving, more charming and etc. [31].

There are so many software visualization tools and techniques available [4]. Visualization techniques consist of collection of elements such as points, lines, shapes, texts and textures which each of these elements illustrates an entity or an attribute from a dataset, which is going to be visualized. In some cases more than one visualization technique can be applied for a system [22].

Software visualization techniques can be categorized from motion perspective into two groups of static and dynamic visualization. An example of static visualization is view of the source code with colors [20]. Dynamic visualization is based on information from the analysis of execution of a program [22] and the data generated at the runtime such as data flow or control flow [20].

With regard to dimension, visualization can have either two or three dimensions. Two-dimensional SV tools mainly involve graph or treelike representations, which may contain many nodes and arcs [23]. For some systems with too much information to be visualized, using two-dimensional technique may cause confusion. Therefore, in some papers the need of extra spatial dimension is suggested, which may make it more possible for the designer to describe more aspects of the system [23].

To choose the best visualization technique for the existing software, first of all, the reason and goal of the visualization should be clear. Then the group of users and their level of knowledge and experience with computer systems should be defined. Also, all the existing objects and elements in the system and all the relationships between them should be detected and it should be decided that what aspects of the system are going to be presented. The usages and limitations of the existing system, which is going to be visualized, should also be investigated [20]. On the other hand the current software visualization techniques need to be evaluated. At the end the technique that mostly meet the requirements of the system will be chosen and would be implemented.

3.2. Software Visualization Techniques Analysis

The main reason for this visualization is to make the text-based system comprehensible for users. In this case, users would spend less time to have a more clear and precise point of view of the system. It will happen in this way that instead of reading the comments and memorizing the structure of the system, users will see the flow of the system dynamically while working with the text-based system and have an overview of the system in a big scale in front of them. The dialogue-based software is used in requirement elicitation phase of software development process. Therefore the main group of people who are going to take advantage of this visualization should be software developers. However, it is a good idea to make it also easy for people with business background to use this software in order to develop their required systems by themselves. In this thesis, the main focus is to limit the visualization to the people with computer background specially software developers. It is a difficult job to keep both groups with diverse expectations from the system satisfied. The system has been tried to be designed in a way that, working with it, be as easy as possible even for people with no specific experience in working with computers. Usability testing will validate how useable the system is. It will be discussed later.

Furthermore, since the system is SOA-based in many papers [1, 27, 28, 10, 29] it is discussed that the appropriate approach for SOA visualization is a layered approach because the concept of SOA has a layered structure. It is one of the SOA's advantages that multiple perspectives within an organization can be taken into account [30] since basically SOA consists of both technical and functional aspects. Functional perspective is mostly related to business people while technical perspective deals mostly with IT people. In order to make it understandable for both groups of users, an SOA based system should be visualized in a way to demonstrate both aspects. In the first layer, the flow of the activities, which are being processed in the system, can be shown. The next layer can visualize the services and the relationships between them. Even more layers such as application layer which shows the implementation of the functionalities provided by services in the service layer in more details, can be used depending on the level of abstraction and the type of users [27].

Consequently, the required visualization method should be dynamic in order to show the flow of the system. Also, because in some parts of the system some services have the same rank to be evaluated the chosen visualization technique should be able to show the concurrency and parallelism. Because SOA is used in this system, then it should provide a layered design for visualization. For choosing the number of dimensions for the system, both two and three-dimensional can be chosen depending on the level of details needed to be illustrated. The main objects, which should be visualized, are few tasks such as Evaluation, Pre-Evaluation, Picking (Yes) and Abandoning (No), that are repeatedly being performed in the system. There is a flow in the system, which shows the order of firing of the tasks in the system. This flow should be clearly presented to the users. Also existing services, which are the very requirements that are going to be elicited, should be depicted.

Many graphical visualization techniques exist that can depict the concept and the workflow of the interactive requirement elicitation system. As it is mentioned, the workflow process determines that which tasks need to be executed in which order and by whom [26]. A list of some of the most popular and suitable techniques, which can be used to visualize the workflow of software systems along with their advantages and limitations, is shown in Table 3.1.

	Description	Advantages	Drawbacks
Flow Chart	Views the paths of a code fragment [22]	Is a generic concept/ Applicable in every programing language [22]	Its representation lies in the code abstraction [22]/ Does not need explicit events. Transition occurs automatically upon completion of activities [45].
State diagram	Illustrates process states and transitions among the states [22].	Event-based / Gives an abstract description of the behaviour of the system [45].	Does not allow arcs to flow from any number of states to any number of states [49].
Activity diagram	Shows the overall flow of control and objects [46].	Supports iteration and concurrency [47].	They can get very big and incomprehensible [48].
Petri nets	A graphical tool for description and analysis of concurrent processes [53].	Represents process features such as parallelism, synchronisation and conflicts [50]/ Very powerful and flexible for both logical and quantitative modeling[51]/ Allow arcs to flow from any number of states [49]	The model is very flexible but its flexibility results in loss of focus for users who are less interested in formal analysis [52].

Table 3.1. Comparison of different workflow visualization techniques

After reviewing all the mentioned visualization techniques in table 3.1, it can be concluded that the most suitable technique, which is both appropriate for visualizing the workflow of the system as well as illustrating concurrency of the tasks, is petri net. In the rest of this chapter more details about petri-nets will be addressed and different aspects of applying it as a graphical visualization technique for the text-based system will be discussed.

3.3. Petri Nets

"Petri nets are used to construct a formal model for a Graphical User Interface (GUI). Petri net is a type of visual communication tool same as flow chart or other software development diagrams but the main advantage of petri net is, it can be used to analyze and simulate the concurrent and dynamic activities of systems" [24].

Petri nets are a very well known formalism technique for demonstrating the workflow behavior of the system. Petri-net for the first time was presented by C.A.Petri in 1962 and since then lots of researches focused on petri nets. The ability to clearly represent the concurrency related concerns like parallelism, synchronization and etc. in a graphical way is one of the best advantages of petri nets. [25].

Petri-net is a special type of directed graph with the initial marking M_0 and two types of nodes called places and transition, which are illustrated by circles and rectangles respectively. An arc will connect each place to a transition and each transition to a place. A marking is assigned to each place demonstrates the number of tokens existing in that place. If marking of a place is zero, it means that place is empty.

There are some rules, which are known as *firing rues* and are applicable to a petri-net and change the marking of the petri-net. These rules are as follows:

1. A transition t is called enabled when there is at least one token in each input place p of t.
2. An enabled transition t will be fired when its associated event occurs.
3. The firing of enabled transition t removes one token from each input place p of t and adds one token to each output place p of t [24].

A petri net is a 3-tuple <P, T, W> where:

- $P = \{p1, p2, p3, \dots, pm\}$ is a finite set of places
- $T = \{t1, t2, t3, \dots, tn\}$ is a finite set of transitions
- $F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs from a place to a transition or from a transition to a place (flow relationship) [24].

Figure 3.1 depicts a sample of Petri net.

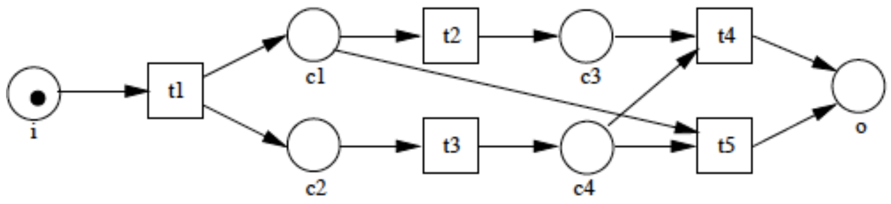


Figure 3.1. An example of a Petri net [26]

Several reasons exist for using Petri nets for workflow modeling. Some of these reasons are as follows:

- *Formal semantics:* Because the semantic of the Petri net has been defined formally, a workflow process specified in terms of a petri net has a clear and precise definition.

- *Graphical nature:* Because Petri net is a graphical language, it is very easy to understand and very suitable to use as a visualization tool for communicating with the end-users.
- *Expressiveness:* Petri net has got all the primitive requirements to model a workflow process.
- *Vendor independent:* Petri net is a tool-independent framework for modeling and analysing processes [26].

Some of the workflow routing constructs, which are needed to be represented in this system, are sequential routing, parallel routing and conditional routing. In sequential routing, tasks are executed sequentially if the execution of one task should happen after execution of the previous task. In parallel routing two tasks should be executed at the same time or in any order. In conditional routing, one task will be executed between two or more alternatives. It depends on the decision made by the system [26]. All these routings are shown in Figure 3.2.

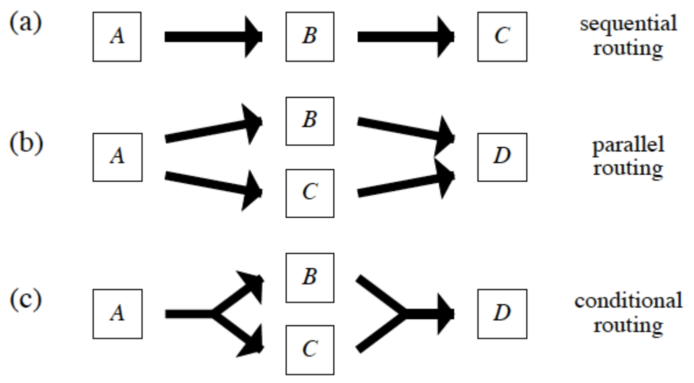


Figure 3.2. Different routings in Petri nets [26]

CHAPTER IV

PROPOSED PETRI-NET BASED SOA VISUALIZATION METHOD

4.1. Introduction

Previously, in the prior work was conducted by Zhang [19] a dialogue-based system was developed. It automates the requirement engineering process. It manages the knowledge related to SPL requirement engineering by defining an ontology for the system and also it generates service-oriented outputs for the automation of system implementation. In this system some questions such as whether the user needs a specific requirement is generated and the user will respond to these questions. At the end, based on the services the user has ordered, the system generates a service discovery and composition.

An improvement that can be applied on this dialogue-based system is graphical visualization of the behavior of the system. A petri-net based SOA visualization is presented in this thesis, which visualizes the dialogue-based interactive requirement elicitation system. It is used for eliciting user's requirements based on human-machine interaction. As it is mentioned in the previous chapter, for SOA visualization, layered approach can be the most promising for better understanding, because, it can illustrate the structure and the concept of the SOA-based system much more precise.

For this reason, this proposed visualization method shows the requirements/services on the background as the on the service layer and on the top of them there is a petri-net, which shows the flow of the dialogue system which is on the business layer. Because the visualization is decided to be two-dimensional the business layer and the service layer overlaps. It is also possible to show the dependencies between the services on the service layer. In order to apply this idea, this design should be altered to a three-dimensional plan instead of a two-dimensional. But there is a risk of complexity increment that can cause comprehension decline. The goal of this visualization is to give the user, which is mainly a software developer, a better understanding of the system under development in order to save time, energy and cost.

4.2. The Structure Of The Proposed Method

The visualization should somehow be included as a part of the dialogue system. The frame-based dialogue system designed in the previous research consists of four components: Interface, I/O controller, Dialogue manager and Knowledge base. The visualization component will be added between interface and I/O controller Figure 4.1.

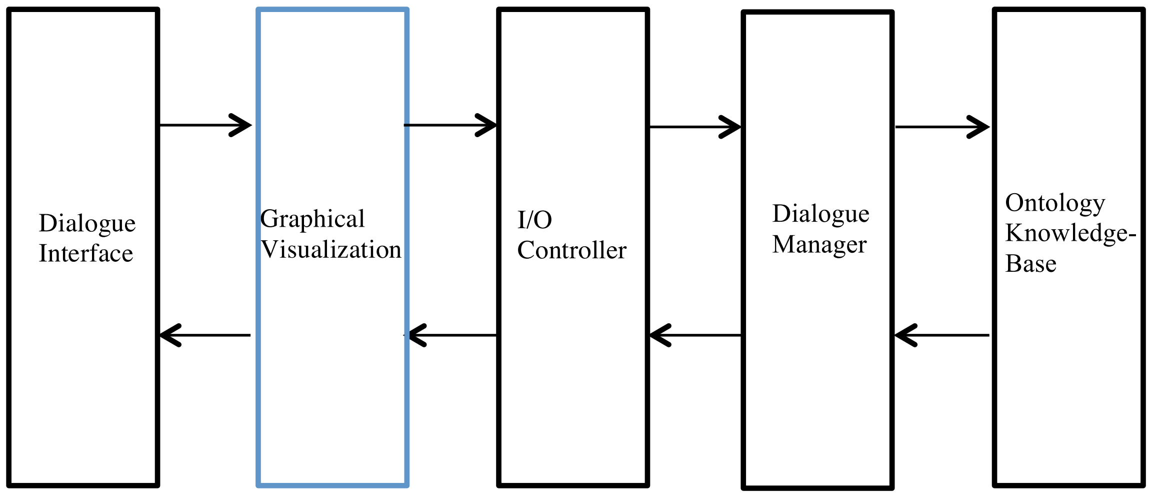
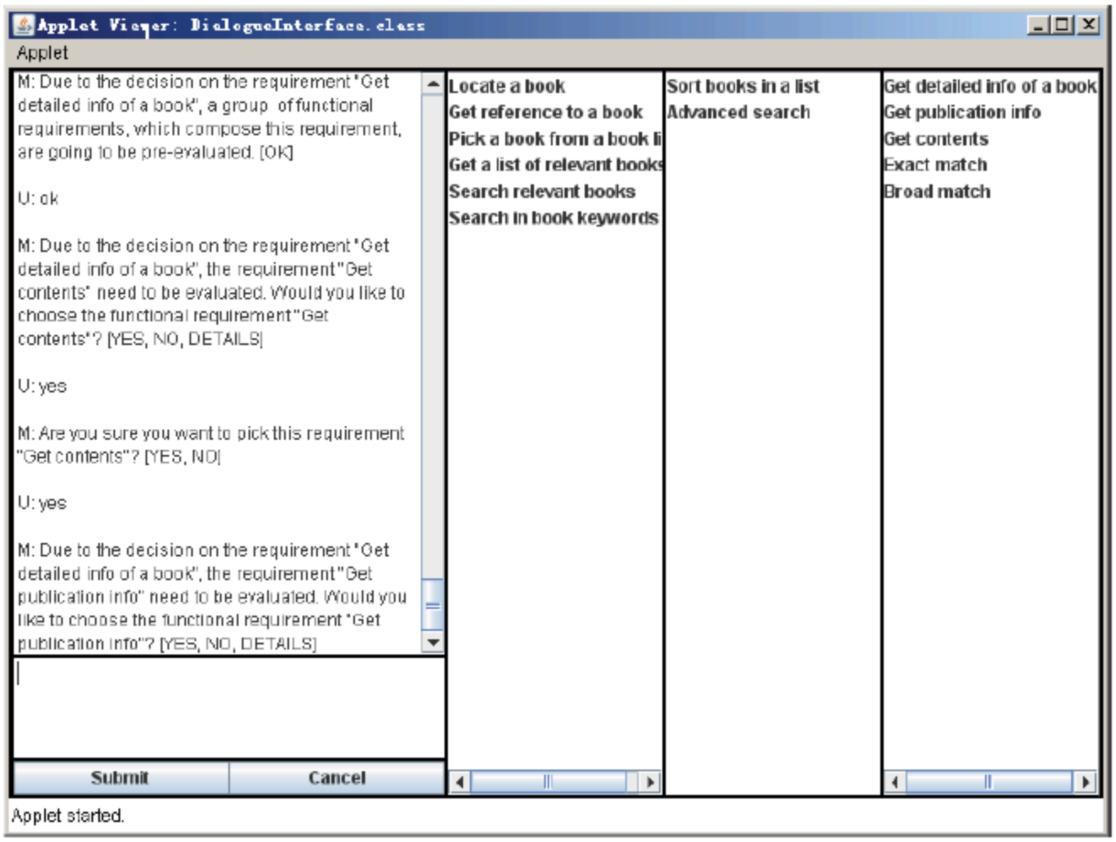


Figure 4.1. Architecture of the modified system

It works in this way that, the dialogue interface will display the questions generated by the machine and the user will respond in the provided slot on the interface. In the previous system the answer used to go directly to the I/O controller to be matched with the predetermined answers in the system Figure 4.2. This time in the new system with graphical user interface, the answer will be passed to the visualization component and then to the I/O controller. It means that all the inputs and outputs of the system should pass through the visualization component at least once. If the user’s answer matches with the saved answer options in the system, it will pass to the next stage which is the dialogue manager otherwise the user will be asked to enter the correct answer. The user’s answer will be converted to the format processable for the machine and will be passed to the dialogue manager. The dialogue manager will consult the ontology knowledge base and will generate an answer subsequently. This answer will be passed to the I/O controller and visualization components and the user can read the answer in the dialogue interface and also observes the changes occur in the system on the graphical interface. These changes will be shown by token moves and color changes in the petri-net and background services.


Applet Viewer: DialogueInterface.class

Applet

M: Due to the decision on the requirement "Get detailed info of a book", a group of functional requirements, which compose this requirement, are going to be pre-evaluated. [OK]

U: ok

M: Due to the decision on the requirement "Get detailed info of a book", the requirement "Get contents" need to be evaluated. Would you like to choose the functional requirement "Get contents"? [YES, NO, DETAILS]

U: yes

M: Are you sure you want to pick this requirement "Get contents"? [YES, NO]

U: yes

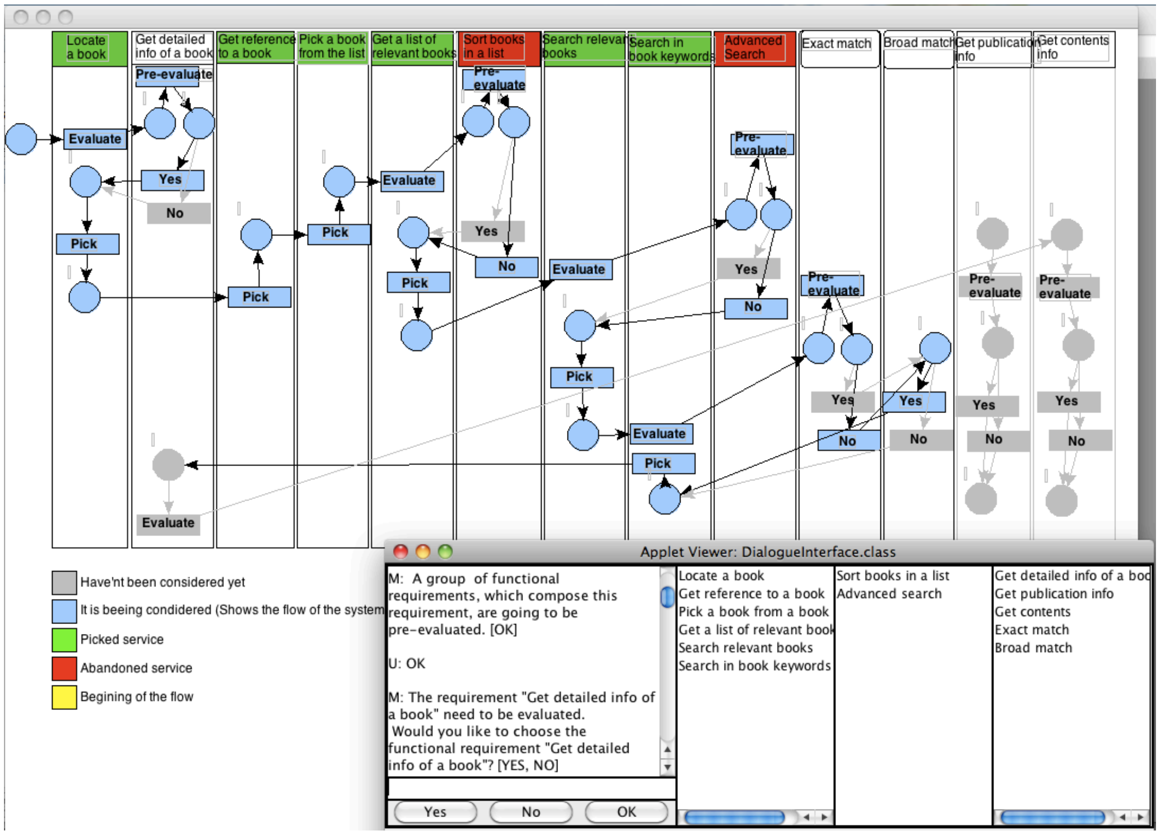
M: Due to the decision on the requirement "Get detailed info of a book", the requirement "Get publication info" need to be evaluated. Would you like to choose the functional requirement "Get publication info"? [YES, NO, DETAILS]

Locate a book	Sort books in a list	Get detailed info of a book
Get reference to a book	Advanced search	Get publication info
Pick a book from a book li		Get contents
Get a list of relevant books		Exact match
Search relevant books		Broad match
Search in book keywords		

Figure 4.2. The text-based user interface [19]

4.3. A Modified Algorithm

As is mentioned in previous section, there will be two user interfaces. One is text-based and the other one is the graphical representation of what is happening in the requirement elicitation process. As it is shown in Figure 4.3, all the services, which are being used as the requirements of the software system, are being shown on the background of the interface.



M: A group of functional requirements, which compose this requirement, are going to be pre–evaluated. [OK]

U: OK

M: The requirement "Get detailed info of a book" need to be evaluated.
Would you like to choose the functional requirement "Get detailed info of a book"? [YES, NO]

Locate a book
Get reference to a book
Pick a book from a book
Get a list of relevant book
Search relevant books
Search in book keywords

Sort books in a list
Advanced search

Get detailed info of a boo
Get publication info
Get contents
Exact match
Broad match

Figure 4.3. The enhanced text-based system with the graphical interface

This design meets the qualifications of the SOA visualization. Regarding to many papers [27, 28, 10, 29], a suitable visualization for SOA systems is a layered design, which in this case the graphical interface, demonstrates a somehow layered approach. It has got two layers, which are service layer in the background, and business layer on the top of service layer. Since the design decided to be two-dimensional, therefore these two layers overlap. To emphasize what is happening in the visualization in the beginning, all the services and the petri-net on the top are in a faded color. As the dialogue interface goes on, the color of each service, which is being evaluated by the system or the user will be highlighted. Based on the petri-net rules whenever a transition's associated event occurs that transition will fire. Thus, regarding dialogue manager's decision a change in color will happen to the petri-net and each fired transition and its input place and arc and its output arc will turn to blue. In this way, the user can follow up the workflow of the system and he will know which step he is in. So that, all the traversed path will be in blue and all the remained paths will be faded. Similarly, all the selected or dropped services, either by user’s decision or by ontology knowledge base will be turned to green or red respectively after each picking or abundance. The pseudo code of this system is shown in Figure 4.4. In this code, all the black lines are from the previous system and blue lines are related to the enhancement done on the system by graphical visualization.

```
1.  For each requirement R to be evaluated
2.      IF R is essential for the system
3.                  CALL performRequirementSelecting with R
4.                  t = Pick
5.                  SET R green
6.  Label requirements that have two directional rely relationship with the essential
    requirements as "essential"
7.      ELSE IF R is pre-selected THEN
8.            t = Evaluate
9.            CALL performRequirementSelecting with R
10.     ELSE IF R is pre-dropped THEN
11.           t = Abandon(No)
12.           CALL performRequirementDropping with R
13.     ELSE
14.           CALL evaluateRequirement with R
15.           t = pre-evaluate
16.           IF R is to be selected THEN
17.                 SET  $t$ = Pick (Yes)
18.                 CALL performRequirementSelecting with R
19.                 SET R green
20.           ELSE
21.                 SET $t$ = Abandon (No)
22.                 CALL performRequirementDropping with R
23.                 SET R red
24.           END IF
25.     END IF
26. END FOR
27.  performRequirementSelecting with R
28.           CALL selectRequirement with R
29.           IF t is enabled THEN
30.                 $.t \xrightarrow{T} t.$ (move token from the input place of $t$ to its output
                    place)
31.                 SET $t$ and $.t$ and its input and output arcs and its input place
                    blue
32.           ELSE wait until $t$ is enabled and goto 30
33.           CALL preselectRequirement with the requirement R relies on
34.           CALL preDropRequirement with the requirements R contradicts
35.           CALL preEvaluateRequirement with the requirements R decomposes
              into
36.           IF R is a function THEN
37.                 CALL preEvaluateRequirement with the qualities R is
                    associated with
38.           END IF
```

```
39. perfromRequirementDropping with R
40.         CALL dropRequirement with R
41.         IF $t$ is enabled THEN
42.                 $.t \xrightarrow{T} t.$ (move token from the input place of $t$ to its output
                    place)
43.                 SET $t$ and $.t$ and its input and output arcs blue
44.                 SET R red
45.         ELSE wait until t is enabled and goto 44
46.         CALL preDropRequirementwith the requirements that relies on R
```

Figure 4.4. The modified pseudo code of the system

Here is the explanation of the above algorithm. The following cases may happen in the system.

1. If requirement R is essential to the system, PerformRequirementSelecting will be called.
2. Task t will be set as “Pick” in the graphical interface and requirement R in the service layer will turn green.
3. If the requirement R is non-essential and pre-selected, PerformRequiremnetSelecting will be called.
4. Tasks t will be labeled as “Evaluate”.
5. If the requirement *R* is non-essential and pre-dropped, *preformRequirementDropping* will be called.
6. If the requirement *R* is non-essential and has not been pre-selected or pre-dropped, *evaluateRequirement* will be called to have *R* evaluated by users. Then if users choose to select *R*, actions for selecting a requirement will be performed. Otherwise, actions for dropping a requirement will be performed. In the graphical interface if the answer of the user is yes then task t will be labeled as “Yes” and requirement R will turn green on the background. If the user drops the

requirement, the label of t will be set as “No” and the requirement on the background will turn red.

- PerformRequirementSelecting contains SelectRequirement R, which makes R to be selected in the system, PreSelectRequirement to pre-select the requirements that relies on R, Pre-DropRequirement to pre-drop the requirements that contradict with R and PreEvaluateRequirement to pre-evaluate the qualities that associate with requirement R. In the graphical interface if the labeled task is enabled, a token will move from the input place to the output place of the labeled task. Then the task and its corresponding input place and arc and output arc will turn blue.
- performRequirementDropping contains dropRequirement R, which abandons R and preDrops the requirements that rely on R. In the graphical interface if the labeled task is enabled, a token will move from the input place to the output place of the labeled task. Then the task and its corresponding input place and arc and output arc will turn blue.

CHPTER V

IMPLEMENTATION AND USABILITY STUDY

5.1. Implementation

In this thesis graphical interface implementation is done by Java 6.0 on Mac OS X operating system. For coding and debugging Eclipse IDE (3.6) is being used. A GUI simulator called "Rakiura JFern" which is a Java-based framework is used to design the petri-net in the project.

The graphical visualization works along with the modified version of the text-based system. The user should do software customization by interacting with the dialogue-based system and checks the flow of the process in the graphical interface. The whole system is shown in Figure 4.3. As it is illustrated, all the requirements are in some rectangles in the background of the visualized system and the petri-net is on top of it. As it is mentioned before, because this visualization is in the category of two-dimensional visualization, then it seems that these two layers overlay.

Some of the features of the pervious text-based system have been changed based on some standards in usability engineering. For example, because the answers consist of, "Yes", "No" or "OK", three buttons are added instead of user typing a word in the dialogue manager, in order to reduce the time and effort for typing and user can accomplish the task only by clicking by the embedded buttons. Based on [31], button is convenient operable control, which is used for frequently used actions that are specific to a window.

Another modification to the dialogue-based system was shortening the dialogues. In [32] it is mentioned that dialogues should be relevant to the users and give them exactly needed information. The previous dialogue-based system had some information that was mostly suitable for software developers and not ordinary users. Even for software developers there were so long, repetitive and not very useful. Therefore, the dialogues were modified in a way to give the users the most brief and important and at the same time complete information.

The graphical interface is added to the text-based system in order to show the user the flow of the development process by the text-based system. The basic actions are done in the ontology of the text-based system are evaluating, pre-evaluating, picking and selecting or abandoning the services. Actions are represented by transitions. Whenever each of these actions takes place, the transition related to that task will be fired and the color of that transition and its input arc and place and its output arc will turn to blue. In this way the flow of the system will be presented by color changing. Each picked requirement in the system will turn to green and each abandoned one will turn to red as soon as the dialogue-based system announces that respectively it has picked or abandoned that service.

5.2. Usability Study

Usability is a quality characteristic, which evaluates some main attributes during software development process. It is believed that usability attributes are some accurate and measureable components of an abstract concept, which is usability [33]. These attributes are:

- Learnability - How quickly and easily users can perform a productive work with a new system and how easily they can remember the way the system operates after not using the system for a while.
- Efficiency – The number of tasks can be done by the user in a specific time

interval.

- Reliability – The error rate using the system and time it takes to recover from the errors.
- Satisfaction – The level of user satisfaction after working with the system.

The way that these attributes can be measured is by observing the users while they are working with the system and have an interview with them after they accomplish a task with the system [33].

In issues that human interacts with technology, the analytical research paradigm is not sufficient. Therefore, empirical studies in software engineering are getting more acceptable continuously [34]. Usability is about how the system interacts with the user [35]. Usability engineering defines the final usability level and ensures that the software under usability testing reaches that level [35].

Usability study can be done by different methods. These methods are divided in two general groups as empirical and analytical methods. In the projects that human interacts with machines empirical studies are very useful [34]. On the other hand analytical studies can give early feedback about the design of the interactive system to software testers. Analytic method consists of two classes of methods, which are *usability inspection* and *cognitive walkthroughs.* Usability inspection, involves systematic inspection of the design by means of some factors for a practical, good design.

One example of usability inspection method is *heuristic evaluation,* which is an informal usability testing method. In this method based on general-purpose design guidelines the evaluator, will inspect the proposed design in order to check whether the usability principals have been followed in the design [36].

In [36] nine heuristics are proposed: *simple and natural dialogue, speak the user's language, minimise user memory load, be consistent; provide feedback; provide clearly marked exits; provide short cuts; good error messages, and prevent errors*.

The other method, which is the cognitive walkthroughs uses more explicit, detailed procedure and conducts a more work-based usability analysis by testing real users when faced with the system. It analyses the quality of the interface in directing the user to accomplish a task by asking three simple questions: *Will the correct action be made sufficiently evident to users?; Will users connect the correct action's description with what they are trying to achieve?; Will users interpret the system's response to the chosen action correctly?.* Whenever there is a "no" answer to any of these questions, problems may occur [36].

A usability engineering model presented by Gould and Lewis and is called "famous rules". These rules are: early focus on users, user participation in the design, Coordination of the different parts of the interface, empirical user testing and iterative revision of designs based on the test results [37].

For usability engineering there is a term called usability engineering life cycle, which means not only how the current interface design is satisfying but also whether it is modifiable for future interface. This life cycle has three stages as follows:

❖ Predesign stage:

The main goal of this step is to know about the target user and the task he is going to accomplish. The more it is done in this stage it would be more cost effective for the whole testing, because most probably the number of changes that should be done in the future will be reduced.

1. Early focus on the user: The first step in usability testing is to know about the user

and his exact needs from the system. For example, knowing about user's work experience, educational level, age, level of computer experience will help to anticipate user's problems and consequently will help to design a better interface with considering user's learning difficulties. Also, the weaknesses of the current system should be found out. It should be clear that in the current system what obstacles the user has on his way to achieve his goals, or what is making the user to spend lots of time or makes the user uncomfortable with the system.

2. Setting usability goals: In usability testing, the four usability characteristics should be met. Obviously for each system the priority of each characteristic would be different. But on the whole because all of them are related, getting a good result in any of them can be satisfying.

❖ Design stage:

The main goal of this phase is to reach a useable implementation that is suitable to be released. For this reason first, based on the usability principles we have to provide a prototype of the final system and then test the prototyped system with real users to make sure the design will meet our goal.

3. User participation in the design.
4. Coordination of the different parts of the interface
5. Empirical user testing: This step is very beneficial and it has two basic forms. The first one, which is mostly quantitative will check if the usability goal has been achieved or not and the second one which is more qualitative, the reason of the parts of the interface which are wrong and the amount of their wrongness will be figured out. Different testing methods can be used in this step like user

observation while working with the system and asking questions from the user about his experience with the system.

6. Iterative revision of design based on the test results: Based on the empirical testing stage we can redesign the interface and again do the testing on the new interface.

❖ Postdesign stage:

This stage is the follow up study of product use in the field. The same task, which is done in design stage, which was revising the design and retesting it repeatedly, can be done with the final product with considering the final product as a prototype of future products [38].

5.3. Proposed Usability Testing Method

Based on what is discussed in usability section of this research, both analytical and empirical testing has been conducted in this thesis. For analytical testing, a combination of usability inspection and cognitive walkthroughs methods has been used along with famous rules, as much as it was applicable and practicable with the available feasibilities, in different phases of designing the system. The main focus was on meeting usability attributes as much as possible in the design. The text-based system modified by applying relevant principles in the heuristic evaluation checklist presented in the previous section. The most related principles are simple and natural dialogues by shortening the comments of the dialogue-based system, speaking the user's language in the dialogues by changing the dialogues in a way that non-software developers can understand the concept, minimize the user's memory load by adding *provided requirements, picked requirements and abandoned requirements* titles on top of each section of kept requirements in the dialogue-based interface (also, in the visualized interface, colors are used for picked and abandoned services in order to reduce users memory load), consistency already exists in the system by the same messages from the dialogue-based system, the visualized interface provides feedback for the user by changing the color of the nodes and keeps the user informed about what is happening in the system. For usability inspection method, the dialogue-based system has already provided error messages for the user when he types an irrelevant word or presses an irrelevant button, for error prevention, dialogue based system gives the user options for the appropriate response in brackets after each question. Even adding three buttons in the graphical interface will prevent some typing errors in text-based system and cases less errors happen. On the other hand the system is designed in a way to responses positively to all the three questions in cognitive walkthrough method.

For the empirical testing, both systems has been tested by users from both, computer science and business departments. A questionnaire was provided to the users for asking their idea about the system for both improving the user interface usability and comparing the modified system with the previous system. Also, in some special cases the idea of some of the users with high experience in software development were asked and applied to the system as much as applicable.

Chapter VI

RESULTS

Both text-based system and graphical interface system were subjected to usability evaluation by two groups of users with varying levels of software development expertise. The first group included 20 students with very little or no experience in software development skills from business department. The second group consisted of 50 computer science students. It was assumed that in general, computer science students have a higher experience in software development than business students, which are not expected to have any experience with this field.

	Computer Students		Business Students	
	Text-Based	Visualized	Text-Based	Visualized
Undergrad (1st yr)	1	1	1	0
Undergrad (2nd yr)	3	6	0	2
Undergrad (3rd yr)	3	1	3	3
Undergrad (4th yr)	2	2	4	6
Master's	12	12	1	0
PhD	4	2	0	0

Table 6.1 – Distribution of participants in the test according to their academic level

6.1. Briefing

The order of the tasks that participants were asked to do is as follows. First, They were asked to listen to a brief description of the differences between classic software development process (SDP) and software product line (SPL), which were symbolized with a very well known and simple concept of Lego. For this reason, few slides were provided for the participants, and they were asked to compare the way a city can be built by basic, cubic Lego pieces to make the city by pre-made Lego accessories such as doors, windows, characters and vehicles. In the second way, instead of making each unit of the city by putting basic building blocks one by one together, a city can be made using pre-made pieces. This concept can be generalized to the concept of classic Software Development Process versus Software Product Line. The users were explained that SDP is like building a city with basic Lego pieces because in this process the code should be written from scratch. On the other hand SPL can be the same as the process of constructing the city with putting pre-made pieces of accessories together. Because SDP is based on customization and reusing of existing software components [1], this exemplification can be illustrative for participant with any level of knowledge about computers and specifically software development.

6.2. Requested Task

After briefing, a task sheet had been handed to them, describing the task they were requested to complete. The task sheets were different for the users who were testing the text-based system and the users who ware testing the graphical interface one. In this task sheet, the details of how each system functions, was described. The test they were requested to complete was to choose only three requirements from 7 optional requirements that the system offers to them. In both systems, the requirements, "Get detailed info of a book", "Sort books in a list", "Advanced search", "Exact match", "Broad match", "Get publication info" and "Get contents" were the variable requirements that the user was able to pick or abandon any of them based on his will.

In the task sheet [Appendices B1, B2], they were asked to pick only three of the services, which give the online book shopping service the following abilities, 1. To sort the search results, 2. To search for a book based on the exact word that is entered to the system and 3. To show the user the information about the contents of the book he has searched for.

These three descriptions are corresponding to three requirements “Sort books in a list”, “Exact match” and “Get contents of a book” respectively.

The user was supposed to pick only these three requirements and abandon four other variable ones. As there are only seven variable requirements, after the users accomplished working with the system, a score out of 7 was given to each of them based on the number of correct selections of the requested requirements, hence, each correct picking and abandoning has one point of the total score. In this way the error-rate of the system, which is one of the usability factors can be measured by subtracting the score from the complete score, which in this case is 7.

The start time were recorded for each user once he/she started to work with the system and the end time was noted as soon as the user was done with choosing all the requested requirements and dropping the rest of the variable requirements. The duration of the time the user spent to finish the tasks with the system was calculated in order to investigate, on average, how long it takes for the users to finish the task. This time will be needed to compute the efficiency of the system, which is about the number of the tasks that can be done by the user in a time interval, and it is one of the main factors in usability testing.

By having the score and time results of the study for both text-based system and graphical system, these two systems can be compared from efficiency and error-rate perspectives. In order to calculate other factors in usability study such as learnability, efficiency and user satisfaction, the participants were asked to feel up a questionnaire [Appendix A1, A2] and according to their answers, other usability factors will be analysed.

6.3. Scores

A series of statistical tests have been done on the scores' results based on a similar study done in [54] about Software Visualization Tool through Usability Study. As we encountered different mean scores for graphical interface group and text-based interface group, which can be seen in Table 6.2, series of test had been conducted to evaluate statistical significance of acquired results for. The Kolmogorov-Smirnov Normality test was performed on the scores for graphical interface and text-based groups, with respect to acquired p-value of 0.4254; there was no real evidence that the scores for graphical interface and text-based system do not belong to the same continuous distribution. Then we can assume that scores are normally distributed.

Consequently, the F-test had been conducted on the scores data for both groups. The F-test test statistic, which is the two groups' variance ratio, has been calculated as F=0.8351 with p equals to 0.5997. It can be concluded within the aforementioned level of significance, that these groups come from distributions with the same variances. Having assumed that the two groups belong to normal distribution and have equal variances regarding to KS test and F-test results, t-test can be used to analyze the mean scores of these two groups.

	Graphical Interface	Text-Based Interface
Mean	5.67	5.00
Standard Deviation	1.331	1.456
Variance	1.771	2.121

Table 6.2 – Statistical parameters for the sample populations

We performed the t-test on the data after validating that the two groups do not have different variances. We were not able to verify the null hypothesis of t-test which is that random samples are from normal distributions with equal means at the standard 5%

significance level, meaning that means of two score groups belonging to graphical interface and text-based interface are statistically different.

In other words with acquired significance level p = 0.0494, so the chance that means of these two groups are equal, is under 5% level of significance. The 95% confidence interval for the test was acquired on interval of [0.0018,1.3315] difference between the means.

6.4. Time

A similar study to what have been done on the score results and based on [54] and [55], was done for the measured time of the graphical-based group comparing to text-based group. For the time, chi-squared goodness of fit test have been undertaken to check for normality of the sample distributions. The difference between maximum and minimum of measured time for graphical-based group and text-based group is respectively 8 and 9, hence 9 have been chosen as the number of bins for chi-square test. The p-value or significance level for chi-square test for graphical is 0.0515 and for text-based group is 0.1641. Both numbers satisfies the standard significance level of 5% and means that null hypothesis of normality for these distributions can not be rejected at that significance level.

Considering the results of the previous test, F-test has been done to study variances of those groups, similar to the study done on the scores. At the same 5% level of significance, variances of these two groups were not statistically different. Following the F-test, we have done the Two sample t-test to analyse the means of the two groups for time. With acquired p of 0.0034, the result was that we can reject the null hypothesis that at 5% level there is no significant difference between the means of time of these two groups. Also at 95% confidence interval, the difference between two means falls in the interval between 0.6089 and 2.9368.

6.5. Questionnaire

The questions of the questionnaire were designed to address the main factors of usability, which are learnability, efficiency, error rate and satisfaction. In this part of the thesis each question and the purpose of designing them for evaluating each factor will be discussed and all the results gained from the users' opinions will be evaluated.

6.5.1. Learnability

The first two questions are aimed to evaluate learnability of the systems. As it is mentioned before, learnability is about how fast and easy users can work with a new system for the first time. This factor can address easiness of the system based on user's opinion as well as the user's opinion about his/her understanding of the system.

6.5.1.1. Easiness

In the first question, users had been asked to rank the level of easiness of working with the system based on their experience. The answer of the users to this question represents their personal impression about their experience of working with the system. The results of the first question for the text-based system are shown in Figure 6.2. It can be seen that average easiness point for graphical interface group, which is 7.91 out of 10, is higher than average point for the text-based group, which is 5.85. This is also confirmed by the higher average score acquired from graphical interface group, that is 5.66 and is 0.66 higher than the average for text-based group 5.00.

In Figure 6.1, average easiness points and average scores have been categorized for the business students and computer science students groups. On the whole, on average both score and easiness given by the graphical interface group for both computer an business students were higher than the score and the suggested easiness rank by all the users from the text-based group. However, in the text-based system, although the average score attained from computer science group, which is 5.28, is higher than business students group's score, which is 4.22, the business students gave a higher easiness point than computer science students. This difference can either be related to the unequal number of participants from each field or it can be because of difference in educational background of participants that can give them various standards for defining easiness.

In case of graphical interface group, both the average score and average easiness point for computer science student is higher than business students.

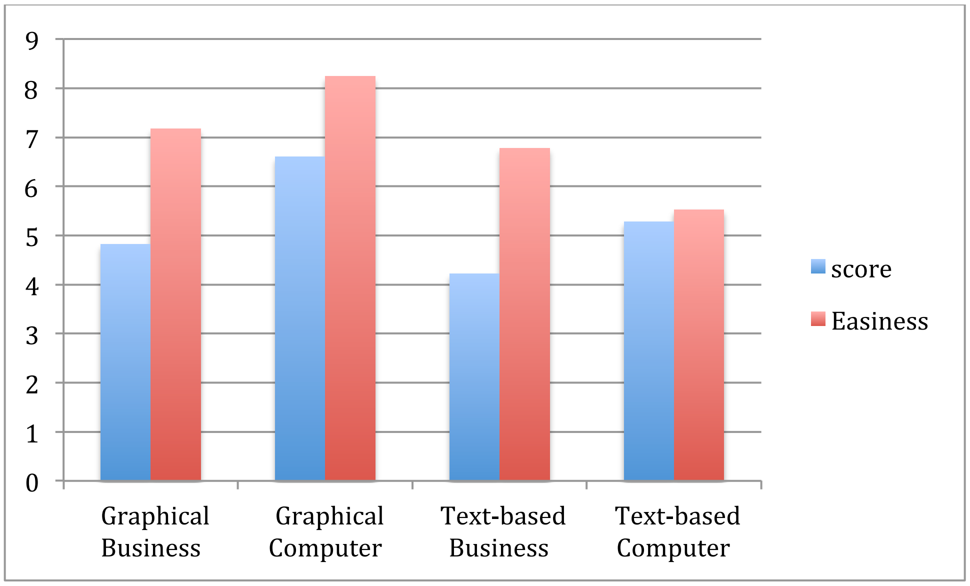


Figure 6.1 - Comparison of average Easiness point to average Score

To sum up, as it was expected the average easiness point given by the users for the graphical interface group was higher than average point for the text-based group.

<u>6.5.1.2 Understandability</u>

Understandability is another factor that was being evaluated in this study. Users were asked to rank their understanding of the system from 1 to 10. Since this point was given to the users by their own, we need to modify this rank based on how successful they had done the tasks and how fast they finished the tasks. Therefore, the following formula, which satisfies these needs, is suggested.

$$Understanding = \frac{User\_rank \times Score}{Time}$$

In this formula, score represents how successful users were in performing the tasks. Therefore a high score shows the subject who had better performance (picked and abandoned correct requirements in less time) had understood the systems better and therefore his suggested score for his understanding is more valid and would have a higher weight and vice versa.

In Figure 6.2 a comparison between Understandability and easiness values is made. It can be seen that Understandability value, in graphical interface is higher than in text-based interface for both user groups from computer department and business department.

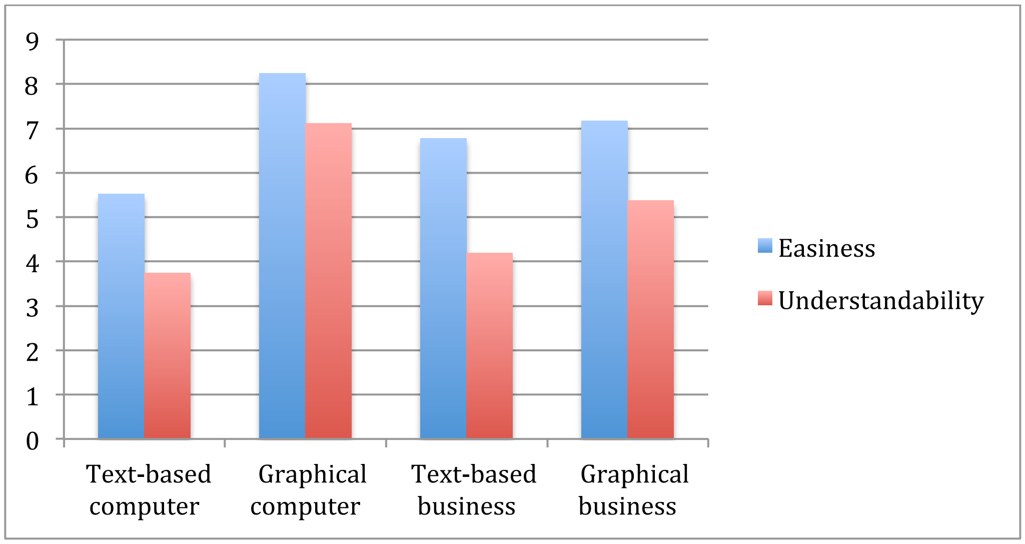


Figure 6.2 – Comparison of Easiness point to the estimated Understandability point

Also as it was mentioned before easiness factor for graphical interface is higher than text-based interface. Since both easiness and Understandability for graphical interface are higher than text-based interface, it can be conclude that learnability of graphical interface is higher for both computer users and business users.

	Text-based Computer	Graphical Computer	Text-based Business	Graphical Business
Easiness	5.52	8.24	6.78	7.17
Understandablity	3.75	7.12	4.19	5.38

Table 6.3 – Easiness and understanding points for different groups

6.5.2. Efficiency

As it is mentioned before, efficiency factor represents number of tasks that can be done by a user in a specific time unit. Thus efficiency can be calculated from dividing score by the time users need to complete the tasks. It should be noted that score is calculated based on number of tasks users had done successfully.

As it can be seen in Figure 6.4 that average of efficiency in graphical interface is 0.81 whereas in text-based interface average is 0.56. In other words, efficiency in graphical interface is about %44 higher than efficiency in text-based interface which means, in the same time unit, users can perform %44 more tasks correctly when users use graphical interface instead of text-based interface.

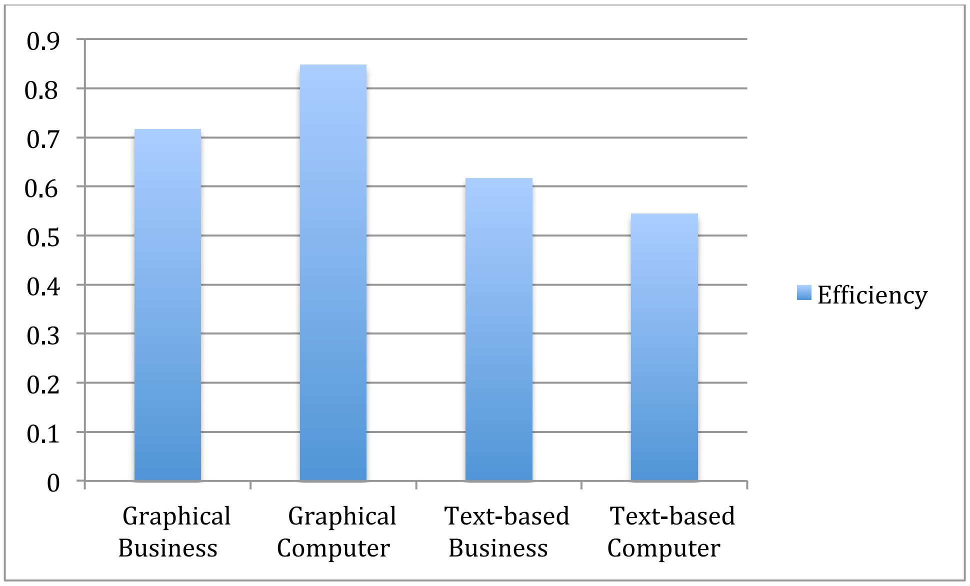


Figure 6.3 – Comparison of Efficiency between different groups

In Figure 6.3, efficiency for different users based on their level of expertise in computer issues, is categorized into four groups. By looking at the figure, it can be seen that efficiency of the users who used graphical interface, no matter whether they were from computer department or business department, is higher than users who used text-based interface. In particular, efficiency of computer students who used graphical interface is higher than other users.

By looking at this factor, we can conclude that having a graphical user interface helps users work with the system faster and more accurately.

6.5.3. Error rate

The next usability factor that should be examined is error-rate. For evaluating the error-rate of each system, the score of each participant should be considered. Since the score was calculated out of seven, the error each user makes is the complement of the score he got. Also, in the questionnaires there was a question that asked the participants about their personal opinion that the error they have made, how much was related to the design of the interface in the text-based interface. Besides, to prove how much the visualization can be effective for the users in order not to make mistakes in the graphical interface questionnaire their opinion was asked. With all the mentioned information, the error-rate and users' opinions about the reason of the error will be brought as follows.

The error-rate in the text-based system for the computer science students was 1.72 out of 7 while the same rate for the visualized interface was 0.94. It shows that for the graphical interface, computer students made fewer errors than for the text-based system. The same data in the text-based system for business students was 2.78 which is much higher than both computer students groups and for the business students who tested visualized interface it was 2.18 which is still higher than computer students results for both systems but it shows that business students who worked with the visualized interface had less error-rate than who worked with the text-based system. For overall comparison between the error-rate of text-based system and the system with graphical interface, all the students who used text-based system made more errors (with having the average error-rate of 2.0) than the all the users who used graphical interface (with having the average error-rate of 1.33).

Since the measurement scale in two questionnaires were different, first we should make these two scales even. In the text-based questionnaire one question asks about the relation between the design of the system and the error that the user makes. While in the graphical interface questionnaire this question changes to the rate the user estimates that the design of the interface prevents him from making mistakes. To make theses two data alike, to be able to compare them with each other, we calculated the complement of the rate the users gave to the graphical interface out of 10. For the text-based system, in average computer students thought that with the rate of 6.88 out of 10 the error they made was related to the way the system interface was designed. Business students on average gave the score, 6.11 out of 10 to the relation of the way the system was designed and the error they made. On the other hand, for the system with the graphical interface, prevented error making with the rate of 8.16 out of 10 in the opinion of computer students, which with the applied changes this number converts to 2.84. It means that the graphical interface with the rate of 2.84 out of 10 directs users to make mistakes. Business users gave the system with graphical interface, rate of 8 out of 10 for error prevention factor. This rate changes to 3.00 out of 10 to mean the same as user error-making relation for text-based interface. It means that 3 out of 10 it was the design of interface's fault that the user made mistakes, in business students' opinions. To make all comparisons more clear all the statistics mentioned above illustrated in Figure 6.4.

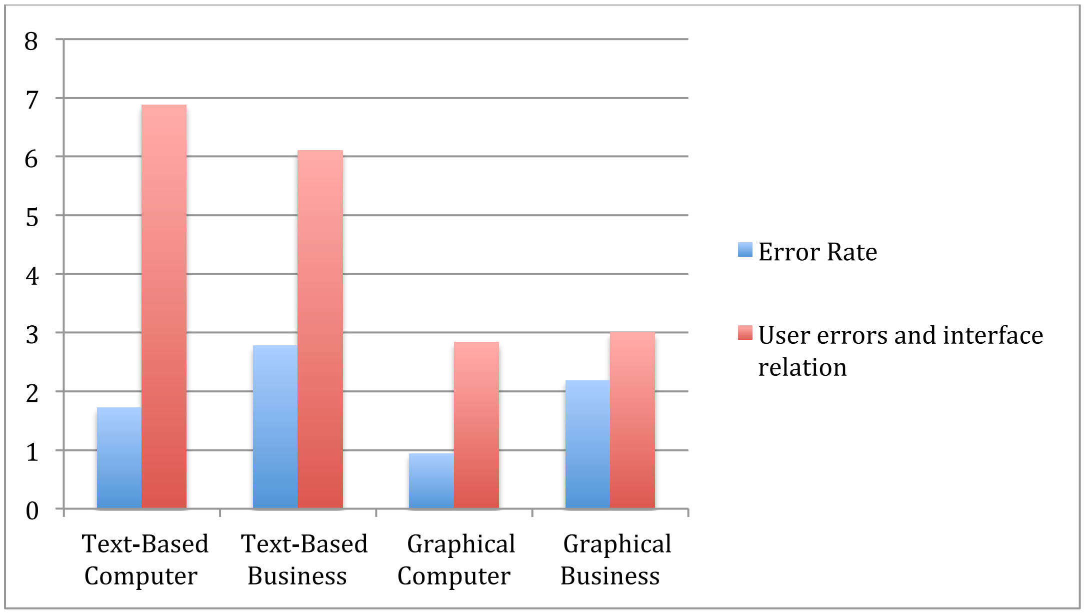


Figure 6.4 – Comparison of users' actual error-rate with claimed relation of users' error-making and interface

Overall, the users of the text based system had 28.6% error-rate and believed that the error they made was 67% related to the design of the system and users of the system with the graphical interface had 19% error-rate and assumed that the visualization 81% prevents them to make errors.

6.5.4. User Satisfaction

The last usability factor, which was evaluated in the study, was user satisfaction. One of the questions in the questionnaires was considered to evaluate the overall satisfaction of the user of working with each system. In this question, it was asked the participants to score their overall satisfaction of the system out of 10. The results were as follows.

The average score that the computer students gave to the text-based system was 6.22. This score, increases to 8.0 for the business students who worked with the same system. For the system with graphical interface, computer students were 8.08 satisfied while business students' satisfaction level was 7.55. These discussed statistics can be illustrated in a chart as shown in Figure 6.5. In overall all users of the text-based system gave the score of 6.69 to their satisfaction of the system and users of the graphical interface system determined their level of with the score 7.92 out of 10.

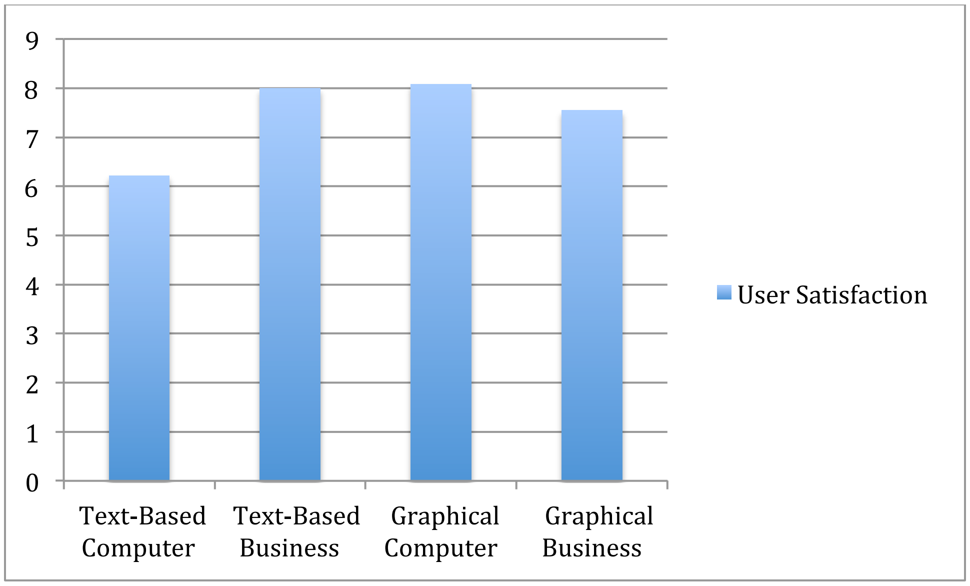


Figure 6.5 - User overall satisfaction of the system for different groups

For the purpose of comparison between two systems, in overall the participants of the visualized interface were more satisfied with this system than the users of text-based. The only exception is the score of satisfaction of business students for these two systems.

On average, the score of user satisfaction for business students who worked with the text-based system was higher than the business students who worked with the graphical interface, but their standard deviation was also higher. It means that although the number of business students with higher satisfaction than the graphical interface users were higher, the ratio of users of the text-based system with lower satisfaction is also higher than the graphical interface users.

In order to prove the claim that the chosen graphical interface enhances the interface of the text-based system in different ways, other than the questions about usability factors some other questions were added in the questionnaires such as users' opinions about what changes can be done in each system and the necessity of visualization for them and etc. These questions are going to be discussed as follows.

6.6. Necessity

One of the important questions was if the users think that the visualization is necessary for the text-based system or not. This question was only asked of the students who worked with the visualized system. Basically, this question can show that how users found the visualization useful and informative. The answer options for this question were "necessary" for the users who thought that it was helpful, "no difference" for the students that do not look at the graphical interface and prefer to read the comments of the dialogue interface and "not necessary" for the students that think that it can be confusing and distracting. The results of this question were as follows.

72% of the computer students, 63% of business students, and on the whole 69.5% of all students believed that this visualization is necessary for the system. 8% of computer students and 27% of business students and on the whole 16.7% of all the students found the visualization not useful enough and there was no difference for them for the visualization to exist.

The percentage of computer students who found the visualization not necessary and probably more confusing for working with the text-based system was 16% and this number for business people was 9% and for all the students participated in testing the system with graphical interface was 14%. Therefore, the majority of the students were willing to have the visualization as a help for a better understanding and point of view of the system.

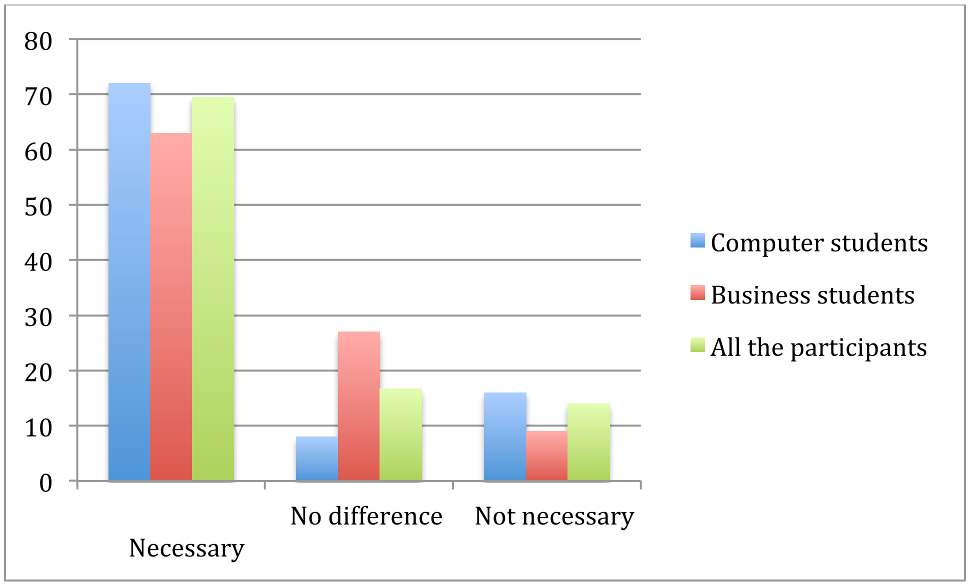


Figure 6.6 – Necessity (percentage) of visualization according to users (graphical group)

6.7. Problems

In text-based questionnaire there is a multiple-choice question for the students who worked with this system, which gives the users the option to add their opinion about the problems that the text-based system has. The three options which were very noticeable based on the usability standards for designing user interface were given as options to the users and they were also asked to add their own opinion if they find more problems in the system. The three options were given to them were as follows:

a) The current state of the user is not clear

b) The comments are very long and not understandable enough

c) Interaction with the system is time consuming and not very convenient

Between these three choices, the percentage of computer students who selected choice “a” was 80%. 60% agreed to choice “b” and 56% choice “c”. Between business students, choice “a” was chosen by 55.5% of the participants, choice “b” by 33.3 % and choice “c” by 44.4% of them. Figure 6.7 illustrates these percentages in a chart.

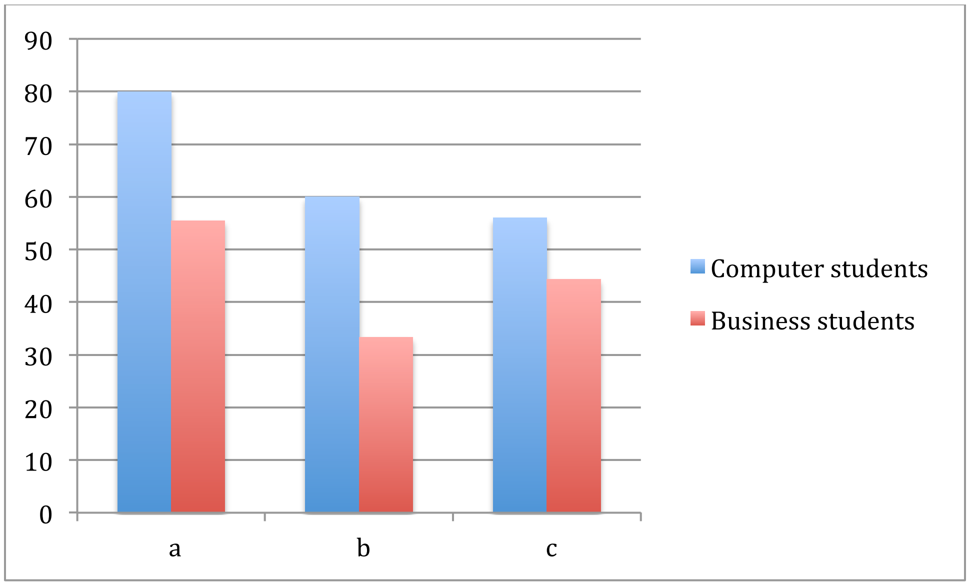


Figure 6.7 – Percentage of users that acknowledged problems (a, b and c) for text-based system

Also the opinion of the users of the graphical interface was also asked about how the system can be improved. There existed two options for them to choose from and also they could add any additional opinion.

The choices for this question were:

a) Having direct interaction with the visualized interface

b) Removing the comments

Between these two options the 84% computer students who were the users of the graphical interface approved that they agree with option ”a” and 100% of the business students had the tendency to directly interact with the graphical interface (Figure 6.7). These ideas can be used for further enhancements of the next versions of the graphical interface.

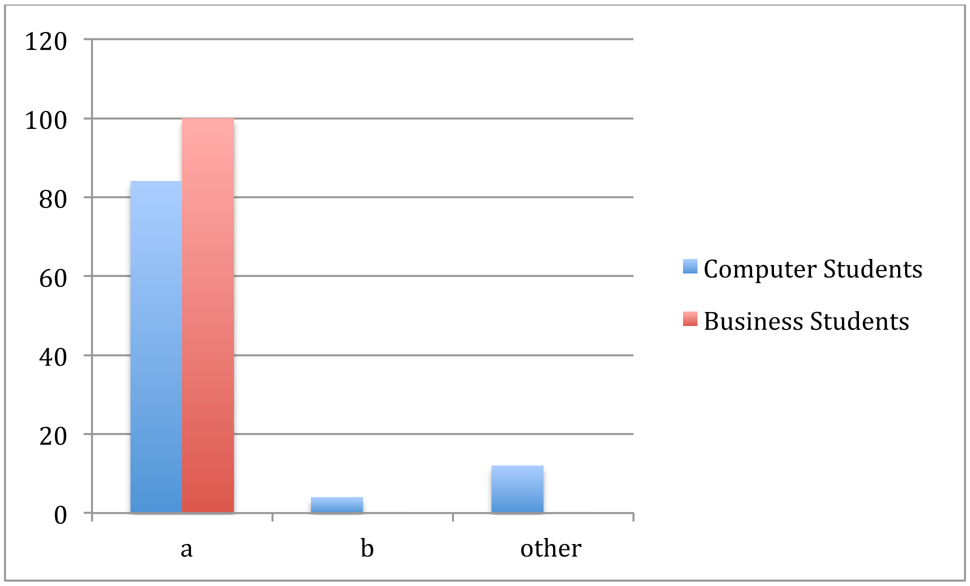


Figure 6.8- Percentage of users that acknowledged problems (a, b and c) for graphical system

Some participants added their own idea about the system. Some of the added comments where as follows:

∗ Text-based interface, Computer:

1. Vague comments.
2. Needs visual aid, lacks in control.
3. Confusing about which step I am in.

∗ Graphical interface, Computer:

1. Drag/Drop required
2. Add audio after each selection
3. Very exciting

∗ Graphical interface, Business

1. SPL idea is very advanced

6.8. Overall Opinion

There are two questions in both questionnaires that ask about the whole idea of ontology-based interactive requirement elicitation. The purpose of this question is to find out whether participants are content in overall with the idea of software customization using software product line.

For this reason users are asked that based on the descriptions that has been given to them before working with the systems, and also based on their experience with the system and their previous experiences do they have any preference on choosing the classical software development process or choosing the software product line. Also, they are asked that how much they think that software product line can improve the software development process.

On the whole, from 70 participants 67% of them claimed that using SPL could be beneficial for software development.

In the last question in this category, overall opinion of the users about the system had been asked from them. The users were asked to give a score from 1 to 10 to their assumption of the level of improvement made by software product line to the software development process. Average improvement point for graphical based interface is 7.83, for the text-based group is 7.26 and for the whole population is 7.55. The details of this part can be seen in Figure 6.9.

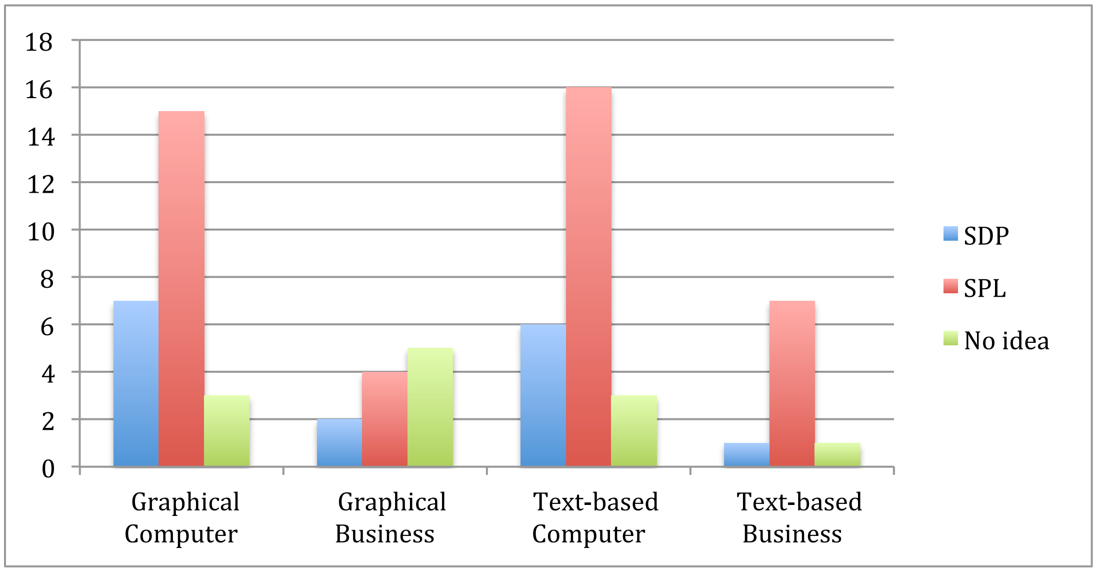


Figure 6.9- Overall participants' ideas about the whole idea of SDP and SPL

Average results for the whole usability study are shown in Table 6.4. . As it had been discussed earlier, in all the parameters including calculated parameters (time and score) and user-stated parameters, graphical interface group acquired a better average.

	Text-Based		Visualized	
	AVG	STDV	AVG	STDV
Easiness	5.85	2.653	7.92	1.730
Understandable	6.94	2.014	8.11	1.304
Satisfaction	6.69	2.535	7.92	1.610
Error	6.68	2.293	2.89	1.879
Improvement	7.26	1.928	7.83	1.813
Time	9.41	2.743	7.64	2.113
Score	5.00	1.456	5.67	1.331
Error Rate	2.00	1.456	1.33	1.331

Table 6.4 – Average results for Text-based and visualized interfaces

In Table 6.5, average results for users have been divided and shown based on their academic background, which is either computer science or business field.

	Computer Students				Business Students			
	Text-Based		Visualized		Text-Based		Visualized	
	AVG	STDV	AVG	STDV	AVG	STDV	AVG	STDV
Easiness	5.52	2.584	8.24	1.589	6.78	2.774	7.17	1.888
Understandable	6.92	1.956	8.36	1.114	7.00	2.291	7.55	1.572
Satisfaction	6.22	2.525	8.08	1.730	8.00	2.179	7.55	1.293
Error	6.88	2.068	2.84	1.886	6.11	2.892	3.00	1.949
Improvement	7.12	2.027	7.96	2.031	7.67	1.658	7.55	1.214
Time	10.24	2.505	7.96	2.354	7.11	2.028	6.91	1.221
Score	5.28	1.400	6.06	1.338	4.22	1.394	4.82	0.874
Error Rate	1.72	1.400	0.94	1.338	2.78	1.394	2.18	0.874

Table 6.5 – Average results for all measured parameters for Comp. and Business students

To sum up all the results collected from business and computer participants for both text-based and graphical interfaces a brief description will be as follows.

Business users of the text-based system on the whole spent less time than all computer users of both text-based and graphical interface to finish the test and had more errors than all the other users of any groups. On the other hand their estimation on easiness and understandability of the system was higher than computer users of text-based interface estimates and they were more satisfied than computer users of the text-based system. It can be concluded that the reason is either because of small number of participants in business group or business participants' misunderstanding in specific of the text-based. On the whole, based on all the collected results from both computer and business groups, it can be debated that the answers of computer students are more realistic, because all the factors in computer groups answers matches with each other as it was expected at the time of design of the system. It can be because participants with computer experience can have more realistic exceptions of a user interface than people with no computer background. Therefore, to make the system more suitable for business group more changes have to be done and more analysis should be conducted.

CHAPTER VII

CONCLUSION AND FUTURE WORK

7.1. Conclusion

In this research, a study has been carried out to conclude when the chosen graphical visualization technique and further enhancements apply on the previously developed text-based requirement elicitation system, it gives better understanding to users of the system and reduces the time and effort they need to spend on eliciting desired requirements.

This research has been accomplished in a number of steps. Initially, the previous text-based, interactive requirement elicitation system was studied and based on basic concepts in design of that system the most suitable graphical interface from both semantic and usability points of view has been designed and implemented. Furthermore, a usability study was conducted on a group of students with different academic backgrounds to justify that the proposed design can positively improve the usability of the user interface.

The results of the study show that, in overall users had a positive opinion about using both graphical-based and text-based interface. However, based on users opinions on average all usability parameters, which are Learnability, Efficiency, Error-rate and User satisfaction have been improved comparing to text-based interface system. Besides, on average users of graphical interface had accomplished the same tasks faster and more accurately than the users of the text-based system. On the whole the majority of the users of both systems prefer Software Automation concept, which is the basis of both systems. over the classical Software Development Process.

Since we found some contradictions in business users scores and their estimation of easiness and their understanding of the system, it can be concluded that further analysis is required. Also, from the beginning the system was designed for people with some experience in software development, which normally business people are not included in this group. Therefore, it was expected initially that the results of computer students be more consistent than business students, which can be considered as a positive outcome based on our primary goal.

7.2. Future Work

Despite the fact that the graphical visualization improved the usability and quality of the user interface, the results of the usability study show that there is still a lot of space for improvement. First of all, the interface can be designed in a way that users can have a direct interaction with the graphical interface instead of indirect dialogue-based communication. It would give the users a better feeling of control over the system. In addition, different graphical interfaces can be designed for people with different expectations and expertise in the field of software development. It can be as easy as drag and drop for users with basic or no knowledge of computer or as sophisticated as a layered three dimensional graphical interface which even be able to illustrates the details about the relationships between services and system architecture.

APPENDICES

Appendix A

Questionnaires of the Usability Study

The following figures illustrate the questionnaires, which were required to be filled by the participants after working with each of the implemented systems for the purpose of usability investigation

Appendix A1

Questionnaire for Visualized Interface:

Name:	**Department:**
Graduate	**Undergraduate (Year:)**

E-mail (for participation in the draw) :

Any previous experience in software development? if yes please specify your level of expertise from 1 to 5(5 is expert) ? Yes / No

Please try to answer all the following questions based on the investigator's description and your personal experience with the software you just worked with:

1. **From 1 to 10, how easy it was for you to work with the system and customize an online book shopping service in overall (1 hardest, 10 easiest)?**

2. **How many points are you going to give to your overall understanding of the system? (From 1 to 10)**

3. **Do you think the visualization was necessary for the text-based system?**
   a. Necessary b. No difference c. Not necessary

4. **What score are you going to give to your overall satisfaction of working with the system? (From 1 to 10)**

5. **How effective do you think visualization is in preventing the user from making mistakes? (Give a score from 1 to 10, more is better)**

6. **What changes do you think will make the visualized interface more useful? Add your opinion if you prefer.**
   a. **Having direct interactions with the visualized interface**
   b. **Removing the comments**

7. **If you were supposed to develop a similar software, or were in a situation to choose a software to be developed for your business, which approach you would prefer?**
   a. **Classical Software development process**
   b. **Software product line**
   c. **No idea**

8. **How much do you think Software Product Line improved the Software development process? (Give a score from 1 to 10)**

Appendix A2

Questionnaire for Text-based Interface:

Name:	**Department:**
Graduate	**Undergraduate (Year:)**

E-mail (for participation in the draw) :

Any previous experience in software development? if yes please specify your level of expertise from 1 to 5(5 is expert) ? Yes / No

Please try to answer all the following questions based on the investigator's description and your personal experience with the software you just worked with:

1. **From 1 to 10, how easy it was for you to work with the system and customize an online book shopping service in overall?(1 hardest, 10 easiest)**

2. **How many points are you going to give to your overall understanding of the system? (From 1 to 10)**

3. **What score are you going to give to your overall satisfaction of working with the system?(Out of 10)**

4. **How much do you think error-making in the system is related to the design of the interface? (Score from 1 to 10, 10 means they are very related)**

5. **Which of the following you consider as a problem of the system (choose more than one if applicable and add your idea if you wish)?**
   a. **The current state of the user is not clear (The user should know how many steps are done and which steps are left in order to complete the whole task)**
   b. **The comments are very long and not understandable enough**
   c. **Interaction with the system is time consuming and not very convenient.**

6. **If you were supposed to develop a similar software, or were in a situation to choose a software to be developed for your business, which approach you would prefer?**
   a. **Classical Software development process**
   b. **Software product line**
   c. **No idea**

7. **How much do you think Software Product Line improved the Software development process? (give a score from 1 to 10)**

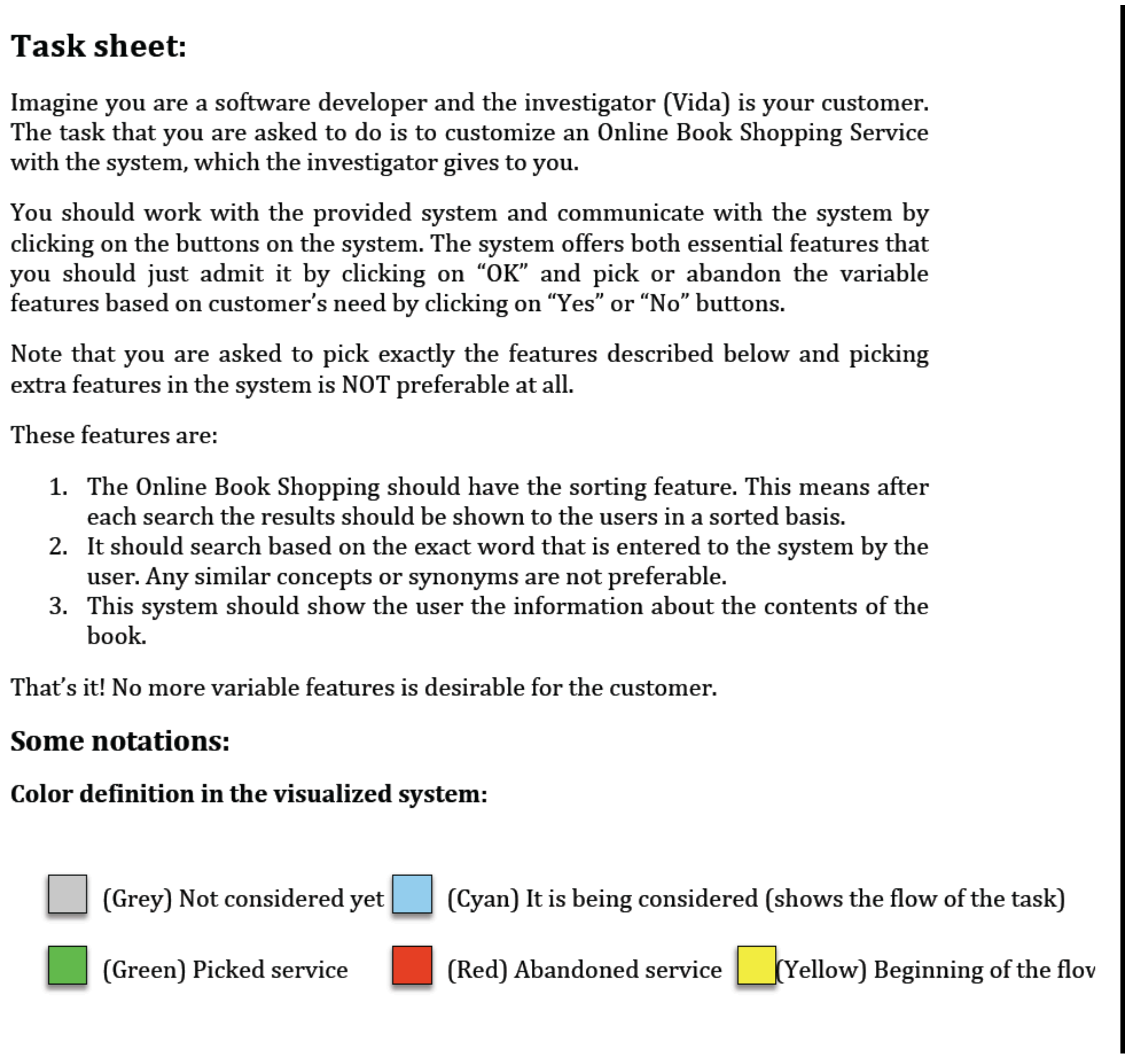
Appendix B

Task sheet of the usability study

The following figures illustrate the task sheets, which were required to be read by the participants before working with each of the implemented systems for the purpose of usability investigation.

Appendix B1

Task sheet for the system with graphical Interface



Task sheet:

Imagine you are a software developer and the investigator (Vida) is your customer. The task that you are asked to do is to customize an Online Book Shopping Service with the system, which the investigator gives to you.

You should work with the provided system and communicate with the system by clicking on the buttons on the system. The system offers both essential features that you should just admit it by clicking on "OK" and pick or abandon the variable features based on customer's need by clicking on "Yes" or "No" buttons.

Note that you are asked to pick exactly the features described below and picking extra features in the system is NOT preferable at all.

These features are:

1. The Online Book Shopping should have the sorting feature. This means after each search the results should be shown to the users in a sorted basis.
2. It should search based on the exact word that is entered to the system by the user. Any similar concepts or synonyms are not preferable.
3. This system should show the user the information about the contents of the book.

That's it! No more variable features is desirable for the customer.

Some notations:

Color definition in the visualized system:

⬜ (Grey) Not considered yet	🟦 (Cyan) It is being considered (shows the flow of the task)	
🟩 (Green) Picked service	🟥 (Red) Abandoned service	🟨 (Yellow) Beginning of the flov

Appendix B2
Task sheet for the system with text-based Interface

Task sheet:

Imagine you are a software developer and the investigator (Vida) is your customer. The task that you are asked to do is to customize an Online Book Shopping Service with the system, which the investigator gives to you.

You should work with the provided system and communicate with the system by typing a suitable answer and submitting it. The system offers both essential features that you should just admit it by typing "OK" and pick or abandon the variable services based on customer's need by typing "Yes" or "No" or "Details" if you want some information about the inner relations between services. The answer choices for each question are written in brackets after each question.

Note that you are asked to pick exactly the features described below and picking extra features in the system is NOT preferable at all.

These features are:

1. The Online Book Shopping should have the sorting feature. This means after each search the results should be shown to the users in a sorted basis.
2. It should search based on the exact word that is entered to the system by the user. Any similar concepts or synonyms are not preferable.
3. This system should show the user the information about the contents of the book.

That's it! No more variable features is desirable for the customer.

REFERENCES

[1] F. M. Medeiros, E. S. de Almeida, and S. R. de Lemos Meira, “Towards an approach for service-oriented product line architectures”, In Proc. of the 3rd international workshop on Service-Oriented Architectures and Software Product Lines, 2009.

[2] Altintas, N. Ilker and Cetin, Semih and Dogru, Ali H., “Industrializing software development: the "factory automation" way”, Springer-Verlag, pp 54-68, 2007.

[3] Kang, Dongsu and Baik, Doo-Kweon, “Bridging Software Product Lines and Service-Oriented Architectures for Service Identification Using BPM and FM”, *IEEE Computer Society*, pp. 755-759, 2010.

[4] S. Bassil and R. K. Keller., “Software visualization tools: Survey and analysis”, *In Proceedings IWPC 2001*, pp. 7 – 17, 2001.

[5] J. Lee, D. Muthig, and M. Naab, “An approach for developing service oriented product lines,” inSPLC ’08: 12th International Software Product Line Conference, pp. 275–284, IEEE Computer Society, 2008

[6] D. Kang, C. yang Song, and D.-K. Baik, “A method of service identification for product line,” in ICCIT ’08: 3rd International Conference on Convergence and Hybrid Information Technology, vol. 2, pp. 1040– 1045, 2008.

[7] S. Trujillo, C. Kastner, and S. Apel, “Product Lines that Supply Other Product Lines: A Service-Oriented Approach,” in SPLC Workshop: Service-Oriented Architectures and Product Lines - What is the Connection?, Sep. 2007.

[8] K. Pohl, G. Böckle, and F. van der Linden, “Software Product Line Engineering:Foundations, Principles, and Techniques”. Berlin: Springer, 2005.

[9] R. Rabiser, P. Grunbacher, and D. Dhungana. “Requirements for product derivation support: Results from a systematic literature review and an expert survey”, Information and Software Technology, 52(3), 2010.

[10] Zhang, L.-J. and Zhou, N. and Chee, Y.-M. and Jalaldeen, A. and Ponnalagu, K. and Sindhgatta, R. R. and Arsanjani, A. and Bernardini, F. , “SOMA-ME: A

platform for the model-driven design of SOA solutions”, IBM Systems Journal, Vol. 47, pp 397 – 413, 2008.

[11] Eric A. Marks, Michael Bell, “Service Oriented Architecture (SOA): A Planning and Implementation Guide for Business and Technology”, Willey, 2006.

[12] Rolland, C., Kirsch-Pinheiro, M., Souveyet, C., “An Intentional Approach to Service Engineering”, IEEE Transactions on Service Computing, Vol. 3 , pp. 292—305, 2010.

[13] Erl, T., “Service-oriented Architecture: Concepts, Technology, and Design”, Prentice Hall PTR, Upper Saddle River, New Jersey, Munich, 2005.

[14] Goguen, J. & Linde, C., Techniques for Requirements Elicitation, 1st IEEE International Symposium on Requirements Engineering, San Diego, pp. 152-164, 1993.

[15] Paech B and Kohler K, “Usability Engineering integrated with Requirements Engineering”, in *Bridging the Gaps between Software Engineering and Human-Computer Interaction*, IEEE CS Press, 2003.

[16] Diaper, D., “Integrating HCI and Software Engineering Requirements Analysis”, SIGCHI Bulletin 29, 1, 41-50, 1997

[17] C. Y. Knaus, "Feature - Interaction design for software engineering: Boost into programming future," *Interactions*, 15(4), 71-74, 2008.

[18] Sousa, K., Furtado, E., “RUPi—A unified process that integrates human-computer interaction and software engineering”, In: Proceedings of the International Conference on Software Engineering (ICSE), pp. 41– 48, 2003.

[19] Xieshen Zhang, “An Interactive Approach of Ontology-based Requirement Elicitation for Software Customization”, M.S. thesis, CS. Dept., UWindsor, Windsor, ON, 2011.

[20] Petre, M., and de Quincey, E., “A gentle overview of software visualization*”, The Computer Society of India Communications (CSIC) ,PPIG newsletter*, 2006.

[21] TEYSEYRE, A. AND CAMPO, R. M., “An overview of 3d software visualization”, *IEEE TVCG,* Vol.15, pp. 87–105, 2009.

[22] Juergen Rilling and S.P. Mudur, “3D visualization techniques to support slicing-based program comprehension”, *Computers &amp; Graphics*, Vol. 29, pp 311-329, 2005.

[23] Gracanin, D., Matkovic, K., and Eltoweissy, M., “Software visualization’, Innovations in Systems and Software Engineering”, *A NASA Journal*, Volume 1, pp 221-230, 2005.

[24] LI, X., AND MUGRIDGE, R. 1994, “Petri net based graphical user interface specification tool*”, In Software Education Conference,* 1994.

[25] Palanque Ph., Bastide R., “Petri net based Design of User-driven Interfaces Using Interactive Cooperative Object Formalism”, In proceedings of 1st Eurographics Workshop on Design, Specification and Verification of Interactive Systems - F. Paterno (Ed.) - Carrara, Italy - 8-10 June.1994.

[26] W.M.P. van der Aalst, “The application of petri nets to workflow management,” *The Journal of Circuits, Systems and Computers*, vol. 8, no. 1, pp. 21–66, 1998.

[27] Eicker, S. and Spies, T. and Kahl, C. “Software Visualization in the Context of Service-Oriented Architectures”. *Visualizing Software for Understanding and Analysis, 2007. VISSOFT 2007,* Vol. 0, pp. 108 -111, June 2007.

[28] C. Lin, S. Lu, Z. Lai, A. Chebotko, X. Fei, J. Hua, F. Fotouhi, “Service-oriented Architecture for VIEW: A Visual Scientific Workflow Management System”, *Proc. of the International Conference on Services Computing (SCC)*, pp. 335–342, 2008.

[29] Mansukhani, M., “Service oriented architecture—White Paper”. *Hewlett-Packard Corp.*, 2005.

[30] Eicker, S., Jung, R., Schwittek, W., Spies, T., “SOA Generic Views - In the Eye

of the Beholder”, *IEEE Computer Society*, pp. 479–486, 2008.

[31] GALITZ, W. O., “The Essential Guide to User Interface Design”, *Wiley*, 1997.

[32] Molich, R., and Nielsen, J., “Improving a human–computer dialogue”, *Comm. ACM*, pp. 338-348, 1990

[33] E. Folmer, J. v. Gurp, and J. Bosch., “Scenario-Based Assessment of SoftwarenArchitecture Usability”, In the Proceedings of Workshop on Bridging the Gapsn Between Software Engineering and Human-Computer Interaction, ICSE, 2003.

[34] P. Runeson and M. Host., “Guidelines for conducting and reporting case study research in software engineering”, *Empirical Software Engineering,* 14(2), pp 131–164, 2009.

[35] X. Ferre, N. Juristo, H. Windl, L. Constantine, “Usability Basics for Software Developers”, *IEEE software*, pp. 22–30, 2001.

[36] J. C. Campos and M. D. Harrison, “From HCI to Software Engineering and back”. *ICSE* ',pp 49-56, 2003.

[37] Gould, J. D., C. Lewis, “Designing for usability: Key principles and what designers think”, *Comm. ACM*, Vol.28(3), pp 300–311, 1985.

[38] Nielsen, J. , “The usability engineering life cycle”, *IEEE Computer*, Vol. 25(3), pp12–22, 1992.

[39] DERLER, P. AND WEINREICH, R., “Models and Tools for SOA Governance.”, *International Conference on Trends in Enterprise Ed. Springer,* 2006.

[40] F. Arvidsson and A. Flycht-Eriksson, "Ontologies I," 2008. [Online]. Available: http://www.ida.liu.se/~janma/SemWeb/Slides/ontologies1.pdf.

[41] N. F. Noy and D. L. McGuinness, "Ontology Development 101: A Guide to Creating Your First Ontology," Stanford Knowledge Systems Laboratory, Tech. Rep. KSL-01-05, 2001.

[42] D. Martin, et al., "OWL-S: Semantic Markup for Web Services," 2004. [Online]. Available: http://www.w3.org/Submission/OWL-S.

[43] D. Martin, et al., "Bringing Semantics to Web Services: The OWL-S Approach," in *Proceedings of the 1st International Workshop on Semantic Web Services and Web Process Composition*, 2004, pp. 26-42.

[44] Ribeiro, Heberth Braga G. and de Almeida, Eduardo Santana and de Lemos Meira, Silvio R., "An approach for implementing core assets in service-oriented product lines", *Proceedings of the 15th International Software Product Line Conference,* Vol. 2, pp. 17:1 -17:4, 2011.

[45] M. Samek. , "Practical UML Statecharts in C/C++Event-Driven Programming for Embedded Systems", *Newnes*, 2008.

[46] R. Lenz, K.A. Kuhn, "Towards a continuous evolution and adaptation of information systems in healthcare**",** *Int. J. Med. Inform*, Vol. 73, pp. 75–89, 2004.

[47] R. Bastos, D. Dubugras and A. Ruiz, B., "Extending UML Activity Diagram for Workflow Modeling in Production Systems", *in 35th Annual Hawaii International Conference on System Sciences, IEEE*, 2002.

[48] J. Helldahl, U. Ashraf, "Use Case Explorer-A Use Case Tool", M.S. thesis, CS and Engineering Dept, Chalmers Univ. , Göteborg, Sweden, 2009.

[49] Amelia Ritahani Ismail, "FINAL QD On the Use of Modelling and Simulation for Granuloma Formation", M.S. thesis, CS Dept, York Univ., 2008.

[50] E. Villani, J.C. Pascal, P.E. Miyagi, R. Valette, "A Petri-net based object-oriented approach for the modelling of hybrid productive systems", *Pergamon/Elsevier Science,* Vol. 68, pp. 1394-1418, 2005.

[51] Wakefield, R. R., “Application of Extended Stochastic Petri Nets to Simulation and Modeling of Construction Systems”, *Civil Engineering and Environmental Systems*, pp. 1-22 1998.

[52] L.Hardman, G. van Rossum, and D. Bulterman., “Structured multimedia authoring”, *In ACM, Multimedia’93. ACM Press*, 1993.

[53] Sun, P., Wang, J., Li, X., Jiang, C., “Performance analysis of workflow model with resource constraints”, *In: Proceedings of the First International Multi Symposiums on Computer and Computational Sciences*, vol. 1, pp. 397–401, 2006.

[54] Marcus, A., Comorski, D., and Sergeyev, A., "Supporting the Evolution of a Software Visualization Tool through Usability Studies", *in Proceedings International Workshop on Program Comprehension*, St. Louis, MO, pp. 307-316, 2005.

[55] Abdinnour-Helm, S.F., Chaparro, B.S. & Farmer, S.M., “Using the end-user computing satisfaction (EUCS) instrument to measure satisfaction with a web site”, *Decision Sciences*, Vol. **36**, 341–364, 2005

VITA AUCTORIS

NAME	Vida Sadri
PLACE OF BIRTH	Tehran, Iran
YEAR OF BIRTH	1985
EDUCATION	Bachelor of Computer Engineering, Azad University, South Tehran Branch, 2004 - 20