2014

An Interactive Approach to Software Visualization for Customization

Manpreet Singh Kaler
University of Windsor

An Interactive Approach to Software Visualization for Customization

By

Manpreet Singh Kaler

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

2014

An Interactive Approach to Software Visualization for Customization

by

Manpreet Singh Kaler

APPROVED BY:

_____

A. Azab
Dept. of Industrial and Manufacturing Systems Engineering

_____

L. Rueda
School of Computer Science

_____

X. Yuan
School of Computer Science

17 April 2014

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

The Software Product Line (SPL) provides software customization by composing several different web services together. 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. A Petri-Net based visualization system for the software customization has been developed in our research group.

This thesis works on enhancement of the prior work by introducing an interactive approach of software visualization for software customization. The proposed approach segregates the users based on their interaction with the system and the best suited visualizations are selected and displayed for the users. In this thesis an interactive framework based on Contextual Control Model has been proposed. A usability study has been conducted to validate the improvements in the usability of the proposed system compared to the existing system.

DEDICATION

To my beloved mother who taught me the most important lessons of life.

ACKNOWLEDGEMENTS

I would like to thank and express my sincere gratitude to my supervisor Dr. Xiaobu Yuan, for his support, guidance and encouragement. His valuable hints and stimulating suggestions has always helped me proceed with this thesis. Without his support and guidance, this work would be impossible.

I would also like to thank Dr. Ahmed Azab and Dr. Luis Rueda, for being in the thesis committee and spending their valuable time in providing me with encouraging feedback.

My special thanks goes to my parents and my sister for their patience and love they provided to me during all times. Finally I would like to thank my friends Akansha, Arshdeep, Harsh, Pawan, Ravinder, Ranjeet and Sahil for their encouragement and moral support.

I would like to thank University of Windsor students 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 1

Introduction and Motivation

1.1 Introduction

During the initial stages of the evolution of software development there was not much emphasis on the reusability of the software artefacts. The cost of the software for a computer system was negligible compared to the cost of the hardware used. But with the evolution of the more and more complex software systems the reusability of the software artefacts has become much more important consideration.

The software has become much more critical part of any computer 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. Due to this increase in the size and complexity of the software system there is a need to cut down the cost and effort involved in deepening and customizing these systems. Therefore, in order to make the system production's process much more efficient, the concept of Software Product Line Engineering is used [22].

Using the Software Product Line different software applications and other software products can be developed by building reusable software artefacts and reusing these software artefacts. 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 [23].

There is also a need to make this development process less complex and more user friendly. The users using these systems have varying levels of knowledge and can be from different domains. Software Visualizations can be used to improve the understanding of the system and thus improving the overall usability of the system. 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 [21].

1.2 Motivation

In last few decades the software development process has changed radically. The software development process has come long way from the software systems with a few lines of code to the software systems with millions of lines of codes. Although the software systems have grown bigger and bigger and are now gigantic but as we can see in most cases there are a lot of software systems that are not so different from others when we see them at lower levels. These similarities at smaller level can be used to develop the reusable software components or artefacts which can be used to combine and develop a more complex software system. Software product line (SPL) engineering is a paradigm to develop software applications with reusable software assets, which are tailored to individual customers' needs [18]. This approach helps in reducing the development costs of the software systems as we do not have to reinvent the wheel every time.

By reusing services, and adopting SOA-based methods in SPL engineering, especially the Semantic Web Service techniques (e.g. automatic service discovery and composition)

[22], the goal of automating software development could be achieved. A system based on this approach has been developed in our research group which uses Software Visualizations to help the user interact better with the system. But this does not help all users as all the users can never have same level of understanding of the system they are trying to develop. This motivates me to conduct a research on this particular technology and solve the challenges associated with an interactive human computer interaction.

1.3 Problem Statement

In software industry due to the increased complexity and the large size of the software systems now a day, the emphasis is on the reusability of the code. The code written for one software system or an application is reused for a similar application and code is not completely written from scratch.

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. In Software Product Line the concept of Service Oriented Architecture is used to develop automated software system for selection and matching of reusable services to create new applications. The services are loosely coupled in order to allow them to communicate with each other. The services can be composed based on the selection, so that the best possible services.

Based on this fact, beforehand, Petri-Net based interface has been developed in our research group [17], which interacts with the user who wants to do software customization, in natural language and does the requirement elicitation process automatically based on the ontology behind it. The visualization system uses Petri-Net to

provide the user with the visualizations. 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 [17].

The problem here is that the system can be used by a wide variety of users and amount of knowledge of these users vary widely. The visualization system based solely on Petri-Net is not very helpful for the users with lesser technical knowledge or the users who are more interested in the business aspect of the software. The level of expertise, knowledge and domain of interest vary widely across these users. But Software visualization for all these users are same and reveal same sort of information. Therefore different types of users should be studied and as per the needs of these users different visualization methods best suited to their level and domain knowledge should be chosen. This research identifies the need to study different types of users and the best suited visualization methods for these users.

In the end a usability study of the system should be conducted to study the usefulness and efficiency of the system and to justify the use of new method. The system should be investigated to check if it improves the learnability, efficiency, and user satisfaction of the system and reduces the error rate.

1.4 Contributions

This thesis presents an interactive approach of software visualization for software customization. The purpose of this method is to enhance the usability of the system, and improve the performance of the system by reducing time, cost and effort spent on

developing the software systems using the traditional development techniques. The introduction of the interactive approach will reduce the effort spent on working with the system which further leads to reducing the time and cost spent. The interactive software visualization will be implemented in the graphical interface of the system, thus improving the usability of the system.

The users are segregated into four groups based of their level of knowledge about the software development and software customization process. The Contextual Control Model (COCOM) is used for the classification of the users based on their actions. These users will be provided different types of visualization interfaces as per their knowledge.

The software customization based on Software Product Line is still in initial stages and the effort involved in the process is huge. The interactive user based approach of software visualization for the software customization will decrease the effort, time and cost of the process of software customization. It will allow the users with lesser knowledge of software development process to customize software for them or to understand the flow of the software being developed.

The Interactive approach of software visualization improves the human computer interaction by providing different visualizations to different users for the better understanding. It provides users a framework which is more friendly and interactive in nature, thus improving the overall user experience.

A usability study will be conducted on real users to compare the existing system and the proposed system, and to check whether the proposed system enhances the user learnability for the real users.

1.5 Structure of the Thesis

The aim of this study is to outline the interactive approach of software visualization for software customization. In chapter 2, a literature review and survey is presented on software customization. Traditional Software engineering methodology, Software Product Line, Service Oriented Architecture, integration of SOA and SPL along with Requirement Engineering and Requirement Elicitation is discussed.

Chapter 3 discusses the central elements of Software Visualization. Visual Representations, Software Visualizations Techniques and Different Methods of Visualizations are described in detail. Chapter 4 describes the proposed Interactive Approach of Software Visualization for Software Customization. The structure of the approach is also discussed. Contextual Control Model and segregation of users on the basis of this model is described in detail along with the algorithm for the interactive approach.

In Chapter 5 implementation and proposed usability study for the verification of the system is discussed. Chapter 6 details the results and analysis of the usability study conducted in order to check the usability of the interactive visualization system. Chapter 7 conclude the thesis and proposes some avenues of future work.

Chapter 2

Software Customization

2.1. Traditional Software Engineering Methodology

No matter how software development is performed or what approach is taken the essential task involved is problem solving. The way developers solve problems is generally the same no matter what the problem is or the approach taken. Problem solving involves four essential activities: requirements - gathering and documenting details about the problem; analysis - understanding the problem in enough detail to ensure a correct solution; design - finding and specifying an optimal solution to the problem; and implementation (if needed) - implementing the solution in whatever form it takes [57].

The essential problem in software development is how to implement, using certain technologies and within certain constraints, a particular information processing system. Although there are associated problems of understanding the domain these are generally non-software related. It can be argued that no matter what paradigm or approach is taken to software development each of the problem solving activities has to be undertaken to some extent. In essence, every developer goes through the requirements, analysis, design, and implementation cycle, be it over an extended period, a week, a day, an hour or minutes, and whether or not they document the results, discuss them with others on a whiteboard, or just consider them informally within their head. There is no escaping these activities [57].

A software development lifecycle (SDLC) gives a high level perspective of how the different problem-solving activities may be worked through in phases by an individual or

team doing software development [57]. The most popular traditional model used is Waterfall Model. The Waterfall SDLC was presented by Dr. Winston W. Royce as a method for software development. It involves sequentially completing each phase in full and then moving on to the next phase. In case of this Waterfall Model the progress is often seen as flowing steadily downwards and hence it is named waterfall method.

The waterfall model is the most commonly used traditional model for the software development and software customization. The figure 2.1 shows the waterfall model structure:

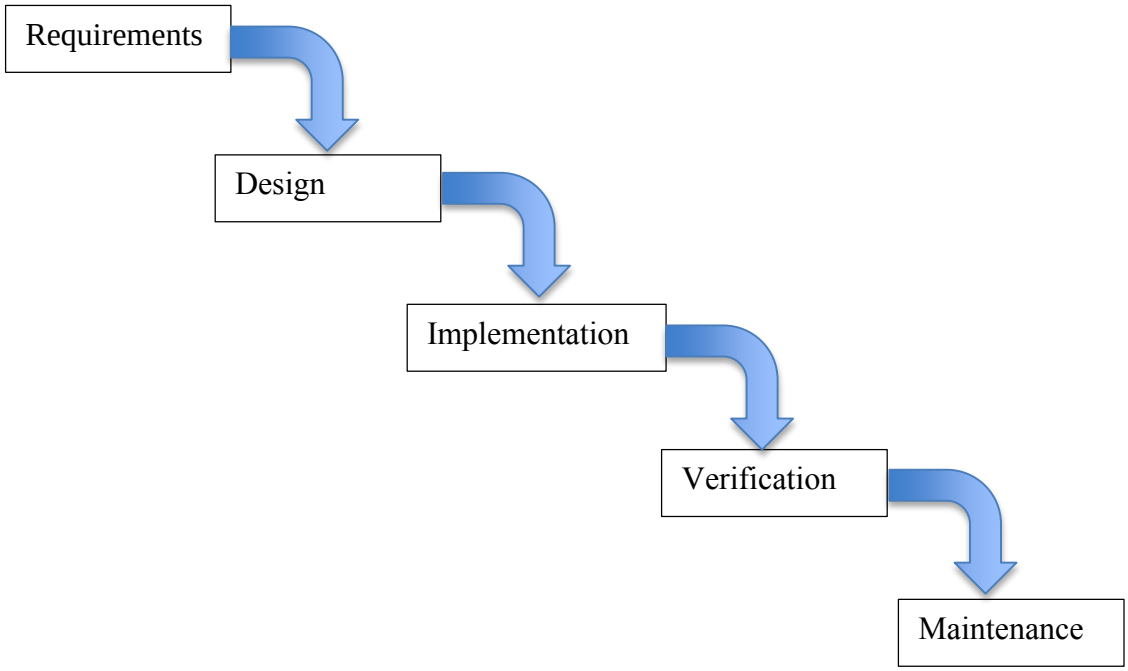


Figure 2.1. The Waterfall Model

The sequential phases in Waterfall model are [57]:

- Requirement Gathering and analysis: All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification doc.
- System Design: The requirement specifications from first phase are studied in this phase and system design is prepared. System Design helps in specifying hardware and system requirements and also helps in defining overall system architecture.
- Implementation: With inputs from system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality which is referred to as Unit Testing.
- Integration and Testing: All the units developed in the implementation phase are integrated into a system after testing of each unit. After the integration the entire system is tested for any faults and failures.
- Maintenance: There are some issues which come up in the client environment. To fix those issues patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

2.2. Software Product Line

During the initial stages of the evolution of software development there was not much emphasis on the reusability of the software artifacts. The cost of the software for a computer system was negligible compared to the cost of the hardware used. But with the evolution of the more and more complex software systems the reusability of the software artefacts became more and more important consideration.

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 [26].

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 [25].

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 [25]. 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 [25].

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 Domain engineering is the process of software

product line engineering in which the commonality and the variability of the product line are defined and realized. [25].

Application Engineering derives the application from the platform, which is built by domain engineering. Application engineering is the process of software product line engineering in which the applications of the product line are built by reusing domain artefacts and exploiting the product line variability [25].

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 [26].

Compared to the effort spending on developing and modeling the software product lines, little support is available for enhancing their utilization in practice,. Without effective approaches to utilize the product lines, particularly the automated approaches, SPL could not be widely accepted in industry. In other words, they will be of more academic value than practical value [23].

2.2. Service Oriented Architecture

Service Oriented Architecture is a software architectural model used for automation of service composition. Service Oriented Architecture separates single business software automation logic into several smaller units of logic. These smaller units are simpler, distinct and distributed in nature. Loose coupling, abstraction and reusability of business functionalities are the major advantages of the 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.” [27].

In Service Oriented Architecture the reusability of code is considered of great importance and thus all the smaller individual units of logic can be reused in several different applications. A single or a group of these smaller units of logic which can work independently are termed as Services. Service Oriented Architecture provides several techniques for composing these services to build a complete business process. Service Composition is the process of combining services based on the service selection.

Services are modules of business or application functionality. Service Oriented Architecture 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 [28].

In Service Oriented Architecture the reusability of code is considered of great importance and thus all the smaller individual units of logic can be reused in several different applications. A single or a group of these smaller units of logic which can work independently are termed as Services. Service Oriented Architecture provides several techniques for composing these services to build a complete business process. Service Composition is the process of combining services based on the service selection. Generally most of the approaches use SOAP.

The Service Oriented Architecture consists of three types of agents. These are:-

1. Service Provider

The Service Provider is the component responsible for creating and publishing a service to a registry. It also makes the service available to the other components through the internet.

2. Service Requestor

The Service Requestor performs service discovery on the service registry to find the needed service and then access that service.

3. Service Broker

The Service Broker component aides service providers and service requestors to find each other by acting as the registry of services.

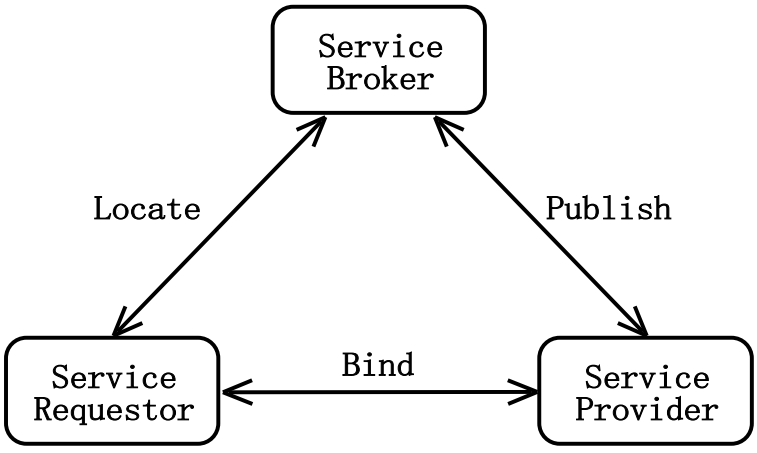


Figure 2.2 Basic Components of SOA

These SOA agents perform find, bind and publish operations. 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 [30]. Service Oriented Architecture helps in achieving loose coupling between the services, abstraction and reusability of business functionalities [30].

2.3 Integrating SOA and SPL

One of the major benefits claimed for SOA is the flexible building of IT solutions that can react to changing business requirements quickly and economically. SOA promises a vision where service providers offer their services and service requesters search and discover these services based on their business needs. In SOA, service providers are usually decoupled from service requesters, thus requesters and providers can change independently of each other. In addition, application development is usually done by assembling services rather than developing components and code. Further, inter-organization collaboration can occur in a decentralized and highly distributed manner. As a result, variability in SOA has different challenges to deal with than in non-SOA systems [24].

Although, there are many differences between typical software product lines and service-oriented architectures, SPL concepts can be used to model SOA variability. Since services in SOA could be used by different clients with varying functionality, SOA variability modeling can benefit from SPL variability modeling techniques. Service

oriented systems can be modeled as service families, similar to the concept of SPL. The main goal of SPL is the reuse-driven development of SPL member applications by using reusable assets from all phases of the development life cycle. This goal is similar to the goal of SOA where flexible application development is a common theme [26].

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 [18]. Furthermore, the integration of SPL and SOA concepts give the ability of reusing existing services instead of continuously developing them from scratch [27].

Combining Service Oriented Architecture and Software Product Line improves the practical implementations of the Software Product Line. It makes the software development process more efficient and improves the quality of the final software product. It decreases development costs and effort, improve time to market, application customized to specific customers or market segment needs and competitive advantages [18]. The integration of SPL and SOA concepts give the ability of reusing existing services instead of continuously developing them from scratch [27].

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 [19].

2.4. Requirement Engineering

Requirement engineering is recognized as the most critical part of the entire software development process [32]. Typically, over 40% of errors in a software project are from requirement, while they need 10 more times of costs to repair than other errors. “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.” [31].

Traditional process-based or scenario-based requirement engineering methods predefine a group of processes and corresponding guidelines. And the requirement engineering activities and deliverables are carried out following the guidelines [31]. However, it is very often that when the processes are ongoing, some important information is not available yet. So, engineers have to repeat the processes, which results in project delay and additional cost [32].

The process based requirement engineering methods predefine a group of processes and corresponding guidelines, and the requirement engineering activities and deliverables are carried out following these guidelines. But in most of the cases the requirements cannot be frozen before the initiation of the development phase. Moreover very often during these processes some important information is not available. In these cases the process is required to repeat when this information is available, this increases the cost of development and also delays the project.

Distinguished from traditional process-driven requirement engineering, knowledge-driven requirement engineering, as a novel requirement engineering paradigm, is

performed under the direction of domain knowledge. As a result, hidden information can be retrieved and used to direct the requirement engineering process. The outcome is expected to be more mature and complete, and rework can be dramatically reduced [32]. Ontology based requirement engineering is a knowledge-driven requirement engineering method which has following major advantages:

1. It provides formal representation for both requirement documents and knowledge
2. It describes the problem domain with varying degrees of formalization and expressiveness
3. It is well suited as an evolutionary approach
4. It is used to support requirements management and improve requirement artefacts' traceability [33].

The previous thesis from the same research group, which has been conducted by Zhang [15], 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 [15].

2.5. Requirement Elicitation

One of the essential tasks of Requirement Engineering during software engineering is Requirement Elicitation. Researches show that a major cause of problems in software projects is inadequate requirement engineering [26]. 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 [26].

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 [32].

In [33] 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 [34]. 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 make 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 [15].

The previous thesis from the same research group, which has been conducted by Zhang [15], 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 [15].

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 language 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. 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. 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 [15].

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 [50]. 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 [50] 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 [15]:

- 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 these relationships should be established between the available requirements and the following procedure will show the instantiation of the ontology [15]:

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.

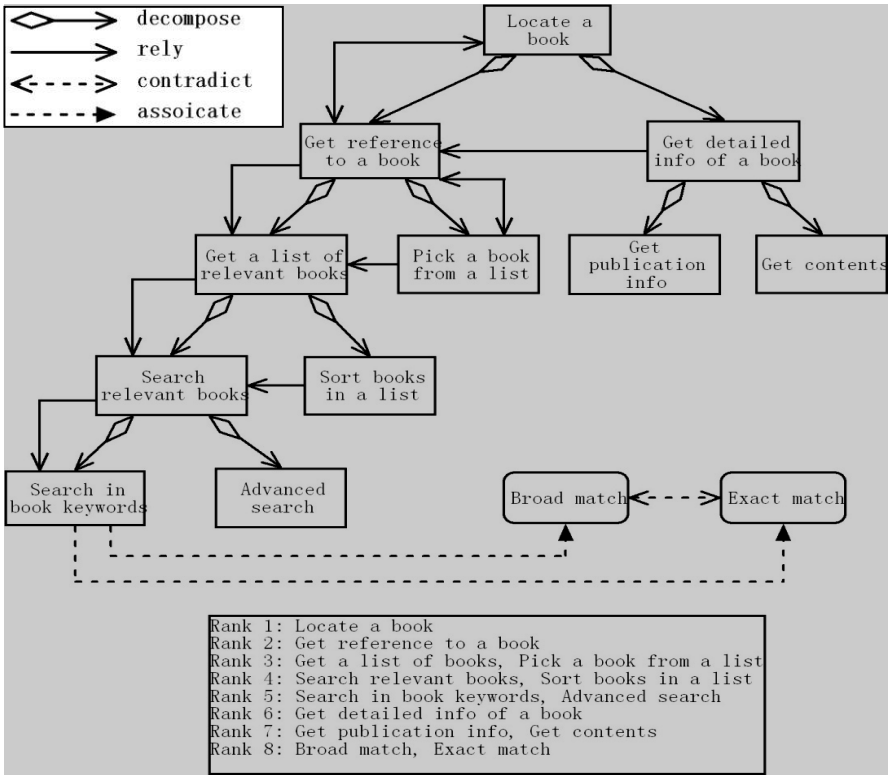


Figure 2.3. Requirement model instantiated with book locating service [19]

The requirements will be offered to the user one by one based on the rank 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 ‘A’, the requirement ‘B’ which has the *rely* relationship with the requirement ‘A’ will be dropped as well. If a requirement ‘C’ is decided to be picked by the user and another requirement ‘D’ has the contradict relationship with the requirement ‘C’ will be dropped and the requirement with the *rely* relationship will be picked [15].

Chapter 3

Software Visualization

3.1. Software Visualization

Card et al. define visualization as “the use of computer supported, interactive, visual representations of data to amplify cognition” [5]. In simple words Visualization is the graphical/visual representation of any knowledge to make the understanding easier. Visualization aids in simplifying the complex ideas for better understanding.

Software Visualization is the process of producing the visual image of a software system for the better understanding. There are different aspects of a software system which can be visualized, generally the structure of the software, major algorithm, simpler components of the software or the runtime behavior are visualized.

Software Visualization can be defined as “a representation of computer programs, associated documentation and data that enhances, simplifies and clarifies the mental representation the software engineer has of the operation of a computer system” [36]. The use of software visualization helps in the easier understanding of the software system and thus it reduces the effort and time spent on different phases of the software development and software customization. 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 [39]. Generally, software visualization is mainly used for program behavior exhibition, logical debugging and performance debugging but it is fundamentally concerned with software comprehension [37]. 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 [39].

Software visualization needs not be attractive or impressive but it should evoke visual understanding for better software comprehension. Software Visualization facilitates the human understanding and effective use of computer programs by relying on the crafts of typography, graphic design, animation, cinematography, and interactive computer graphics [36]. In overall graphical visualization can provide so many other benefits faster learning, faster use and problem solving, more charming etc. [37].

The major problem in the implementation of the software visualization a wide variety of applications is the lack of scalability and flexibility. Most of the visualizations implemented these days are specific to the needs of that particular project or software. The software visualizations implemented for a specific scenario are not easily transferrable to another scenario.

The visualization of a software deals with the representation of the software component in some visual form. For a different application the visualization needs to be changed according to the needs of the new systems. To solve this problem a few generalized software visualizations have been created so that they can be used across different areas. But in case of these generalized software visualizations, generally the visualization is too general to be practically implemented in a certain area of application. In order to use these generalized visualizations a lot changes are required to use these in a specific area. The efforts needed to be put into this are generally way too much compared to the efficiency of these visualizations

Software is an intangible object which cannot be physically seen or felt. Software Visualization deals with the representation of the intangible software entities in the form of something tangible [10]. In software system we have a large amount of information, and it is hard to decide what information software visualization needs to represent in order to make the software comprehension easier.

The Software Visualization should be designed in such a way that it can provide all the required information in such a visual format which makes the information easy to understand. The level of detail should be decided such that the information provided through the visualization should neither be too general nor too specific.

3.2. Visual Representations

Software is an intangible object which cannot be physically seen or felt. Software Visualization deals with the representation of the intangible software entities in the form of something tangible [10]. In software system there is large amount of information, and it is hard to decide which information software visualization needs to represent in order to make the software comprehension easier.

The Software Visualization should be designed in such a way that it can provide all the required information in such a visual format which makes the information easier to be understood. The level of detail should be decided such that the information provided through the visualization should neither be too general nor too specific. Generally a single visualization is used for different users, but this provides information which is too general for a few and very specific for a few.

There are number of visualization techniques available for implementing software visualization. Different types of visual elements like points, lines, shapes, trees, graphs, texts, textures can be used to represent different entities and attributes. In some cases more than one visualization technique can be applied for a system [9].

There are different perspectives of the visual representations on the basis of which we can categorize different software visualization techniques. The visualizations can be static or dynamic based on the nature of the visual technique used. The static visualization do not comprehend the changes on runtime. An example of static visualization is view of the source code with colors [9]. Dynamic visualization changes over every time based on information from the analysis of execution of a program [10] and the data generated at the runtime such as data flow or control flow [7].

On the basis of the number of dimensions most commonly visualizations are Two Dimensional or Three Dimensional. Two-dimensional software visualization tools mainly involve graph or treelike representations, which may contain many nodes and arcs. In some cases there can be too much information which cannot be easily represented in two dimensions, in these cases the need of extra spatial dimension is required, which makes the visual representation of the information much easier [56].

The best suited visualization technique should be chosen keeping mind the target users for the visualization, the complexity of the visualization, and goal of the visualization. The level of knowledge and experience of the users should be considered while using a particular type of visual representation. The usages and limitations of the existing system, which is going to be visualized, should also be investigated [56]. The technique that mostly meets the requirements of the system should be implemented.

3.3. Software Visualization Techniques

The main purpose of using visualization in a system is to make it more comprehensible and easy to use for the users. The visualizations help users to have a more clear and precise point of view of the system while spending lesser time on 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. With the use of the visualizations users have an overview of the system in front of them. Different visualizations can provide different information about the users or they can provide the same information in different ways. These different visualizations can be used for different types of users to provide them the information they need and which helps them using the system more efficiently and effectively while decreasing the number of errors or mistakes made by the users while using the system.

The dialogue-based software is used in requirement elicitation phase of software development process. In a dialogue-based software system the user interacts with the software by responding to the software with the best possible inputs. The system is SOA –based and so in order to make the requirement elicitation process easier for the user with the help of visualization, the visualization should be selected keeping in mind that it should be well suited for the SOA-based systems. Service Oriented Architecture (SOA) has a layered architecture so in many papers [18, 27, 28, 29] it is discussed that the appropriate approach for SOA visualization is a layered approach. 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. 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 [9].

The required visualization methods 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.

Complexity is another major factor while choosing a suitable visualization technique. The complexity of the used visualization should be such that the visualization should be self-explanatory and the process of understanding the visualization should not be time consuming or should not require too much effort on the part of the user. But the complexity can be relative in case of human computer interactions. Different users with varying levels of knowledge can regard a visualization as very simple or very complex

based on their expertise. So the visualizations should be chosen in such a way that they are suitable for a large group of users.

Many graphical visualization techniques exist that can depict the concept and the workflow of the interactive requirement elicitation system. A list of the most 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**
Petri nets Based	A graphical tool for description and analysis of concurrent processes	Represents process features such as parallelism, synchronisation and conflicts / Allow arcs to flow from any number of states	The model is very flexible but its flexibility results in loss of focus for users who are less interested in formal analysis
Directed Graphs Based	Shows overall flow of the system	Less complicated than Petri Nets, helpful in retaining user interest due to simplicity, Provides the required details for analysis	Although it is simpler than Petri Nets still it can be very complicated for users with lesser knowledge
Requirement Model Based	Describes the requirement model of the system	Based on requirement model of the system, thus helping in the correct requirement elicitation	Does not provide the flexibility, concurrent processes are harder to analyse
Block Based Visualization	Simple representations of the requirements	Useful for users with almost no knowledge of the system as it provides a very general overview	Provides no detail about the flow among different requirements.

Table 3.1 Comparison of different visualization techniques

3.4. Different Types of Visualizations

3.4.1. Petri Net based Visualization

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].

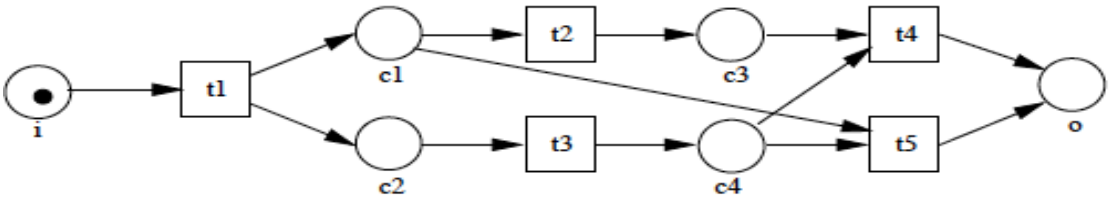


Figure 3.1. An example of a Petri net [26]

Petri-net has 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 rules 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:

- $P = \{p_1, p_2, p_3, \dots, p_m\}$ is a finite set of places
- $T = \{t_1, t_2, t_3, \dots, t_m\}$ 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].

3.4.2. Directed Graph Based Visualization

A graph is defined as a representation of a set of objects where some pairs of these objects are connected to each other. The interconnected objects are known as vertices and the connections between these vertices are called edges. A graph in which each graph edge is replaced by a directed graph edge is called directed graph. The directed graphs do not have multiple edges or loops.

An arc e = (x,y) is considered to be directed from x to y; y is called the head and x is called the tail of the arc; y is said to be a direct successor of x, and x is said to be a direct predecessor of y. If a path made up of one or more successive arcs leads from x to y, then y is said to be a successor of x, and x is said to be a predecessor of y. The arc (y,x) is called the arc (x,y) inverted.

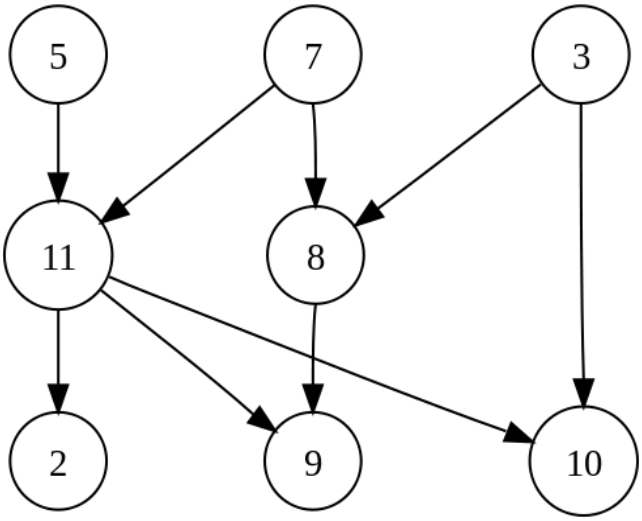


Figure 3.2. An example of a Directed Graph

3.4.3. Requirement Model Based Visualization

The requirement model provides the overview and structure of the requirement elicitation process in this system. The requirement model can be used for explaining the requirement elicitation process graphically. So a requirement based visualization method can used to provide the overview of the requirements and their relationships with each other.

The Requirement Model Based Visualization provides an effective method for the graphical visualization of the system. This visualization very helpful for the users to understand the requirements of the system and they can easily keep track of all the requirements and their relationships (which requirement is dependent upon other requirement or which requirement is essential or which requirements have to be selected in tandem). It gives an easier overview of the requirements and makes it easier for the user to know which requirements have been selected and which requirements have been dropped.

3.4.4. Block Based Visualization

Simple visualization methods can provide the user with the basic information about the system while hiding the complexity of the system from the user. A visualization can be regarded as simple if it is self-explanatory to most of the users. The visualization should fundamentally provide a user an overview of the system and the current status or progress that has been made at that point of time.

The Block based visualization can be used for this purpose. In case of the Block Based visualization the basic operations or actions are represented as simple blocks. In case of the requirement elicitation system, a block based visualization method can be used such that the requirements are represented by these simple blocks. The block based visualization provides an overview of all the requirements and does not deal with the connection and links between these requirements. The user is given an overview of which requirement has been selected and which requirement has been dropped.

Chapter 4

Proposed Interactive Approach of Visualization

4.1. Introduction

In any Software Development Life Cycle or Software Product Line Engineering there are many users working on a single software system. These users work in different groups or teams, each team has an assigned work, and members of a team have similar work. These users work on different aspects of the software systems. In the Software Product Line System based on SOA the mass software customization is made easier by using the smaller software artefacts as the building blocks for the software system.

In the previous research conducted by Vida Sadri Petri Net Based Visualization is used in the system to help the users interact with the dialogue-based system. As mentioned in the [16] 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." So in the Petri-Net Based Visualization system the main emphasis is on the users who have knowledge about the software systems as well as some experience with the development of software. It was evident from the usability study conducted for the above mentioned

research that the system was not as useful for the people with lesser knowledge about the software development process.

4.3. The Structure of the Proposed Method

The interactive visualization approach has to be included as the part of the previously developed system. The Petri Net based system based system designed previously by Vida Sadri [17] consists of five components: Dialogue Interface, Graphical Visualization, I/O Controller, Dialogue Manager and Knowledge Base. The Interactive Visualization component will be added to the system between Graphical Visualization and I/O Controller components.

In the previous system the answers were passed from the Graphical Visualization component to the I/O Controller. But in the new system the answer will be passed through the Interactive Visualization component. The Interactive Visualization component decides the best suitable visualization for the user. The graphical visualization component will provide the user with that particular visualization. The answer form the Interactive visualization will be passed to the I/O controller. The user's answer is accepted and passed to Dialogue Manager if it matches with the saved answer options, if not user is asked to enter correct answer. The user's answer will be converted to the format that can be processed by 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 visualizations and background services.

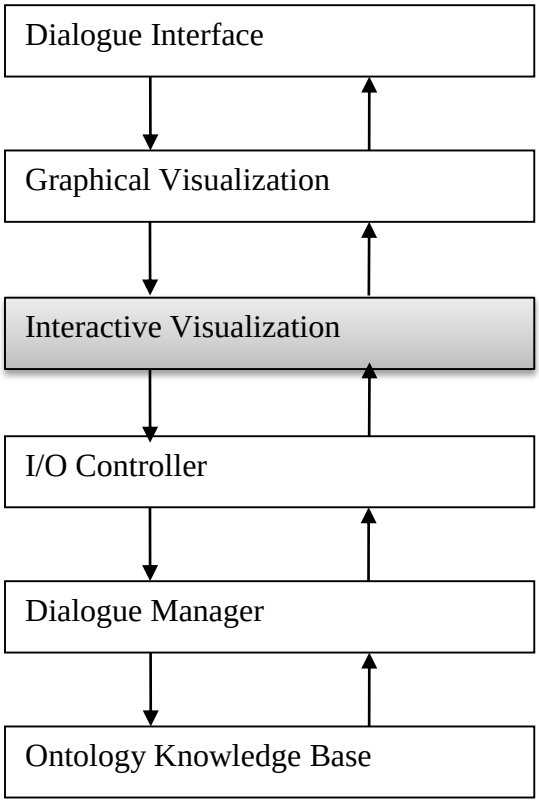


Figure 4.1. Structure of Proposed Method

4.4. Contextual Control Model

Contextual Control Model was developed by Hollanagel to control and analyze team behavior based on cognitive modes. In this model the system decides what action to take next according to the context of the situation. The behavior is analyzed at macro level instead of the micro level [56].

A contextual control model implies that actions are determined by the context rather than by any inherent relations between them. The focus of a contextual control model is therefore on how the choice of next action is controlled rather than on whether certain sequences are more proper or likely than others.

There are four different modes for the control. These are:-

1. **Scrambled**

   The choice of next action is completely unpredictable or haphazard. Scrambled control characterizes a situation where there is little or no thinking involved in choosing what to do.

2. **Opportunistic**

   The next action is chosen from the current contest alone based on the salient features rather than on more durable intentions or goals. The person does very little planning or anticipation, perhaps because the contest is not clearly understood.

3. **Tactical**

   Performance is based on some kind of planning, hence more or less it follows a known procedure or rule. Planning is however of limited scope and the needs taken into account may sometimes be *ad hoc.*

4. **Strategic**

   The person considers the global context, thus uses a wider time horizon and looks ahead at higher level goals. The strategic level should provide a more efficient and robust performance, and therefore be the ideal to strive for.

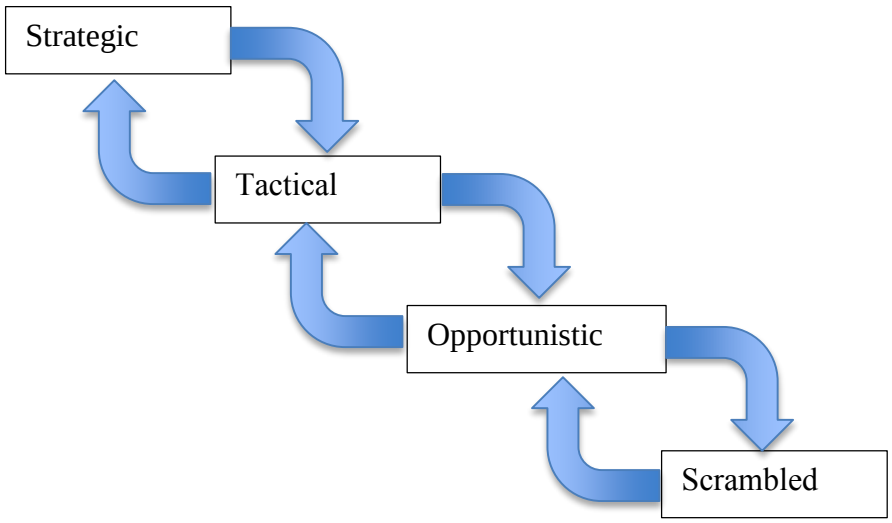


Figure 4.2: Internal structure of Contextual Control Model [56]

In the dynamic system, the individual's transition between COCOM controls modes to maintain the control over the dynamic condition, which in turn depends on the current context of the situation. It provides a useful framework to view the changes in cognitive work in response to contextual features such as time limit and information availability. Control in this model is conceptualized as planning what to do in the short-term and within the time horizon of the system with which the human is interacting.

4.5. Different Types of Users

As discussed in previous chapters' for a software system there are users with different level of understanding and knowledge. These users have different domains of expertise and even in case of users from same domain the level or expertise varies widely. Based on the four contextual control modes users can be divided into four categories considering their level of knowledge and expertise. These users directly relate to the different contextual control modes discussed above. The categorization of the user is discussed below:

1. **Experts**

   The Expert user is the one who has highest level knowledge about the software development and software customization. These users are well versed with software concept like classes, objects, class relations, functions and function calls and other programming related stuff. They are more concerned about the details of stuff and how it works.

   Expert Users favors the more detailed information in the visualization and is able to understand more complicated and technical types of visualizations. The decision making in case of these expert users relates to the Strategic contextual control mode [56]. When these users are working on customization of a software system they have long term goals related to the selection and rejection of the required services. The interaction with the dialogue manager for these users usually takes lesser and more consistent time. The error rate of their responses is far less than other types of users with lesser knowledge.

2. **Professionals**

These users are those people who have the knowledge about the target system, system needs and requirements, but they may or may not have any programming experience or not. These users are very clear about what the system should do but they have lesser knowledge about how it should be done. The planning for these users is of limited scope and the needs taken into account may sometimes be ad hoc.

The decision making in case of these users is tactical in nature when referenced with the contextual control model [56]. These users will interact swiftly with the dialogue manger or the software system and more often than not the responses will be relevant and correct with context of the end goals or final requirements.

3. **Amateurs**

The Amateur Users have almost no knowledge about the software customization or software development. They know what to expect from the customized software system but they don't have any idea about the working of the system. These users know about certain procedures or rules but they lack the in-depth knowledge about the system.

The decision making in case of these users relates to the Opportunistic contextual control mode [56]. There is very little planning to execute the required tasks and very little anticipation regarding the future requirements. The context of the problem is not very clearly understood and the decisions are made after analyzing

each and every requirement at that particular time. These users take longer in responding to certain requirement needs and the results are not consistent.

4. **Novice**

These are the users who lack technical knowledge as well as the business knowledge of the software system. The human computer interaction for these kinds of users is completely unpredictable or haphazard. There is little or no knowledge about the visualization about the technical aspects of the system.

These types of users relate to Scrambled contextual control mode which characterizes a situation where there is little or no thinking involved in choosing what to do [56]. This type of users make random guesses and they are much more prone to make mistakes during the requirement elicitation process.

4.6. Proposed Algorithm

Figure 4.3 provides an overview of the architecture of the proposed system. The Visualization platform is responsible for displaying the visualizations to the user. The Requirement elicitation processor processes the responses and keep track of elicitation process. COCOM mode selector chooses the contextual control model as per the user response history and then the visualization selector selects the visualizations based on the COCOM mode, and then these are displayed by visualization platform. Response History Store and Visualization History Store keeps track of response history and visualization history respectively.

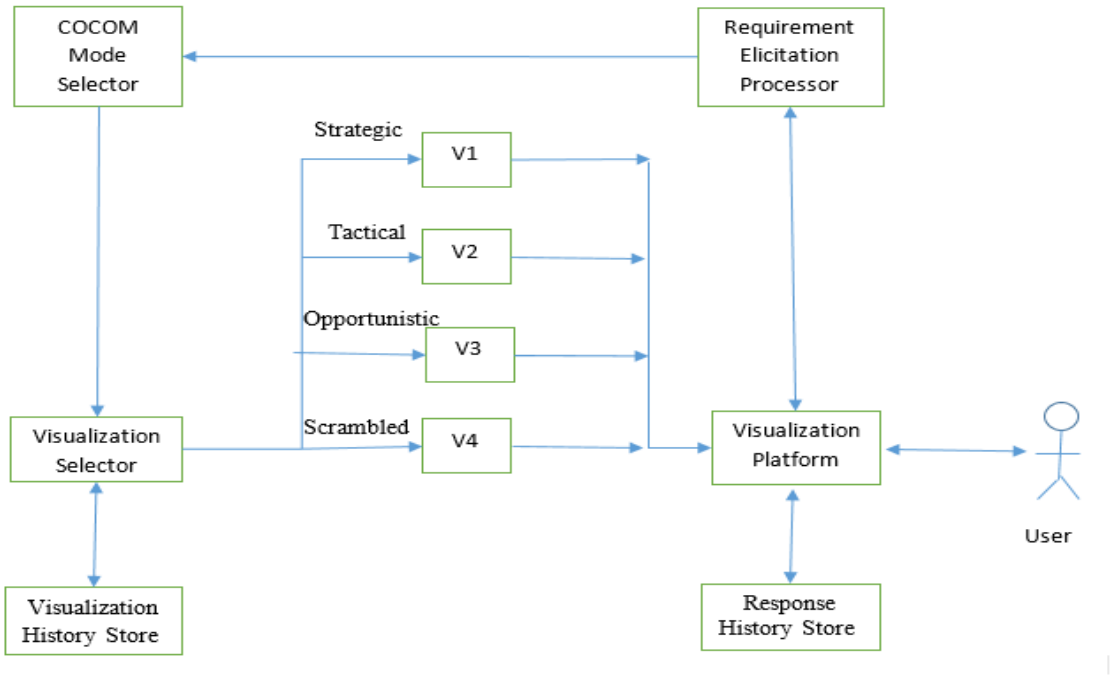


Figure 4.3. Proposed System Architecture

```
1.  FOR each user response RESP
2.      Process_User_Response(RESP)
3.      Response_History = Get_Response_History()
4.      Visualization_History = Get_Visualization_History( )
5.      COCOM_Mode = Check_COCOM_Mode(Response_History, RESP)
6.      IF COCOM_Mode = strategic
7.          Updated_Pref_Visualization = Select_Visualization(Policy_Strategic,
        Visualization_History)
8.      ELSE IF COCOM_Mode =tactical
9.          Updated_Pref_Visualization = Select_Visualization(Policy_Tactical,
        Visualization_History)
10.     ELSE IF COCOM_Mode =opportunistic
11.         Updated_Pref_Visualization =
        Select_Visualization(Policy_Opportunistic, Visualization_History)
12.     ELSE IF COCOM_Mode =scrambled
13.         Updated_Pref_Visualization = Select_Visualization(Policy_Scrambled,
        Visualization_History)
14.     IF Current_Visualization  == Updated_Pref_Visualization
15.         Update_Visualization(Current_Visualization)
16.         Display_Visualization(Current_Visualization)
17.     ELSE
18.         Pass_token(Current_Visualization, Updated_Pref_Visualization)
19.         Update_Visualization(Updated_Pref_Visualization)
20.         Display_Visualiation(Updated_Pref_Visualization)
21.         Set_Current_Visualization(Updated_Pref_Visualization)
22.     Append_To_Visualization_History(Current_Visualization)
```

Figure 4.4. Pseudo Code of the System

The pseudo code executes for each and every response entered by the user. *Process_User_Respnse* is called to process the user response in context to the requirement being considered. If the response is valid then the requirement can be evaluated, pre-evaluated, picked, or dropped. In case the response is not valid user is asked for a valid response. User's response history and visualization history are retrieved by the system. The contextual control mode is computed based on the user's response history and the last response. Based on this Contextual Control mode most preferred visualization is selected for the user.

If the preferred visualization is same as the current visualization being displayed then the visualization is updated. In case the preferred visualization is different from the current visualization being displayed a token is passed from the current visualization to preferred visualization. This preferred visualization is updated as per the user response and then it is displayed to the user. Visualization history is updated with the latest preferred visualization.

Chapter 5

Implementation and Usability Study

The main objective of this thesis research is to provide a visualization system which can cater to the needs of different types of users based on their interaction with the software customization system. In this chapter, the implementation of the interactive visualization system in a software customization system will be discussed. This system has been implemented to improve the human computer interaction and to make the software customization process easier and understandable to users with lesser knowledge about the software systems and software development. This chapter will also discuss a usability study to verify whether the proposed interactive system is actually helpful to the real users.

5.1. Implementation

In this thesis interactive graphical interface implementation is done by Java 7.0 on Windows 8.1 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 used to design the visualizations used in the implementation.

The interactive visualizations systems works along with the text based system. The user should work on the software customization by interacting with the dialogue-based system and checks the flow of the process in the interactive graphical interface. All the requirements are shown in some rectangles in the background of the visualized system.

The interactive approach has been implemented in the system by creating some new methods and by modifying some existing methods being used for the requirement elicitation, text based dialogue interface and Petri Net based dialogue interface. The DialogueInterface.java class is responsible for the main interface of the system. Most of the work related to the implementation of the interactive system has been carried out in this class. The figure below shows the list of methods in DialogueInterface.java class.

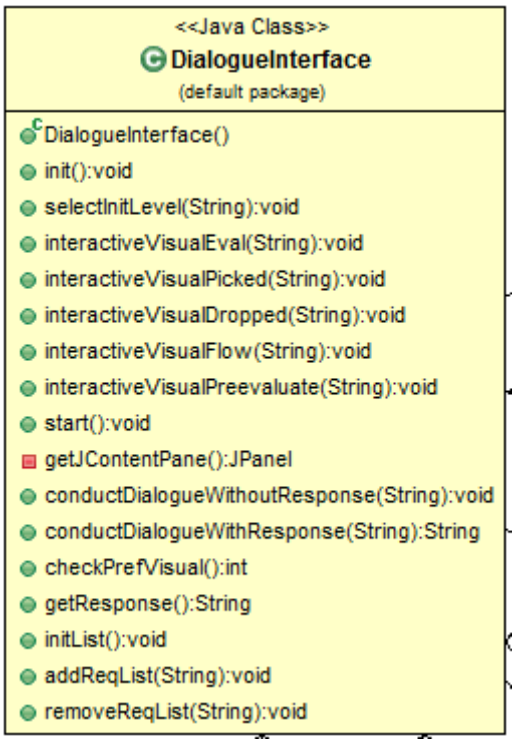


Figure 5.1. DialoguedInterface.java Class

Following new methods have been created:

void interactiveVisualEval(String) : This method is used to perform the visualization for all evaluation actions in the requirement elicitation process.

void interactiveVisualPicked(String) : This method is used to update the visualizations when a requirement is picked by the user.

void interactiveVisualDropped(String) : This method updates all the visualizations when any requirement is dropped by the user.

void interactiveVisualPreevaluate(String) : This method updates the visualizations when a requirement is pre-evaluated.

void interactiveVisualFlow(String) : This method controls the flow of control in the visualizations.

void selectInitLevel(String) : This method is used initialize the preferred visualization method for a user.

int checkPrefVisual() : This method checks the preferred visualization method for a particular user and returns a value based on that.

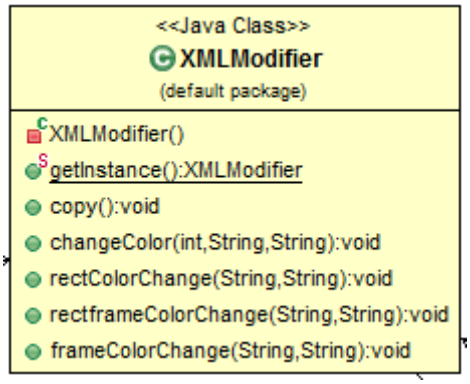


Figure 5.2. XMLModifier.java Class

In XMLModifier.java Class all the methods were updated to include the logic for interactive visualization approach. The methods were updated so that they can be used for

creating four different visualizations and in such a way that all the four visualizations are updated at the same time when a user responds to the dialogue interface.

There are changes made to the other classes such as Requirement.java, InteractiveRE.java, BookShoppingNet.java. All these classes are modified and updated to accommodate the new interactive approach.

There are four different visualizations used in the interactive system, but one time only one of these is displayed on the screen to the user. The best suited visualization for any particular user is deduced by system based on the algorithm discussed earlier in this thesis. The interactive graphical interface helps the user in knowing and understanding the ongoing software customization process. The best suitable visualization is shown to the user based on the user's responses and interaction with the system. The graphical user interface provides the users with an option to choose another visualization or to view other visualization and revert to their preferred visualization method.

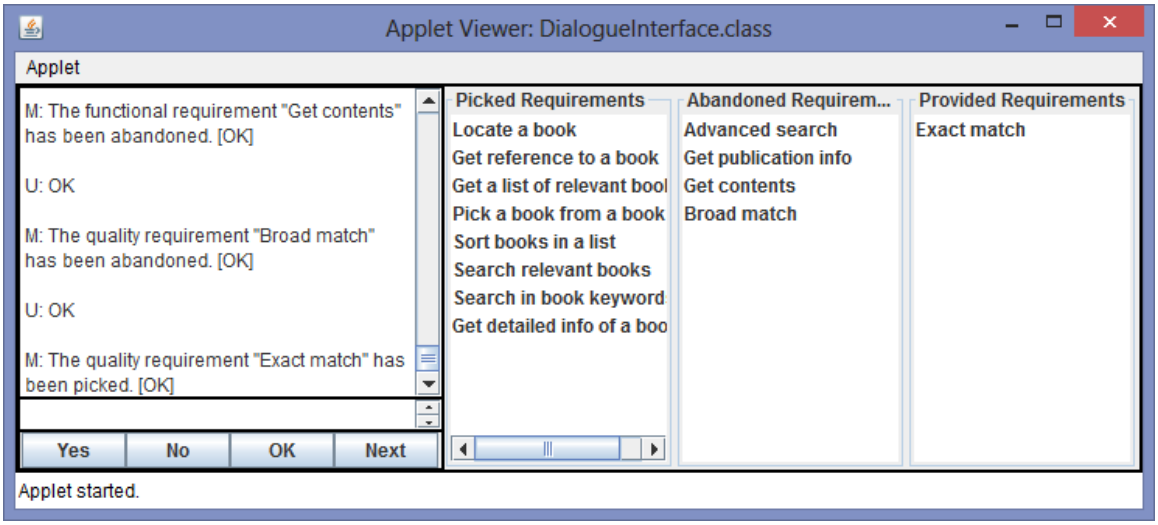


Figure 5.3. Dialogue Manager Interface

The user interacts with the system to for achieve the software customization goals. This interaction can take place is few different ways based on the input methods used by the user. User can input his/her responses in the textbox field, and then can view computer's response in the panel itself. The dialogue manager for the requirement elicitation process has been developed in such a way that the answers consist of, "Yes", "No" or "OK". On the other hand the user can just use the buttons provided in the graphical user interface to respond to the dialogue manager, system's responses will be displayed in the panel. For this purpose three different buttons are provided in the interactive system, these buttons are "Yes", "No" and "OK". There is another button embedded in the user interface "Next", this button is used to change the current visualization displayed to the user. In the interactive system we have four different visualizations, sometimes a user may want to check some other visualization, and this can be done by clicking "Next" button.

The interactive framework proposed in this thesis segregates the users into four groups based on the contextual control model. The user responses are studied and then the best possible visualization for a particular user is displayed. For this purpose four different visualizations has been implemented. These are:

1. Petri Nets Based Visualization
2. Directed Graph Based Visualization
3. Requirement Model Based Visualization
4. Block Based Visualization

Figure 5.4 shows Petri Net Based Visualization. . 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.

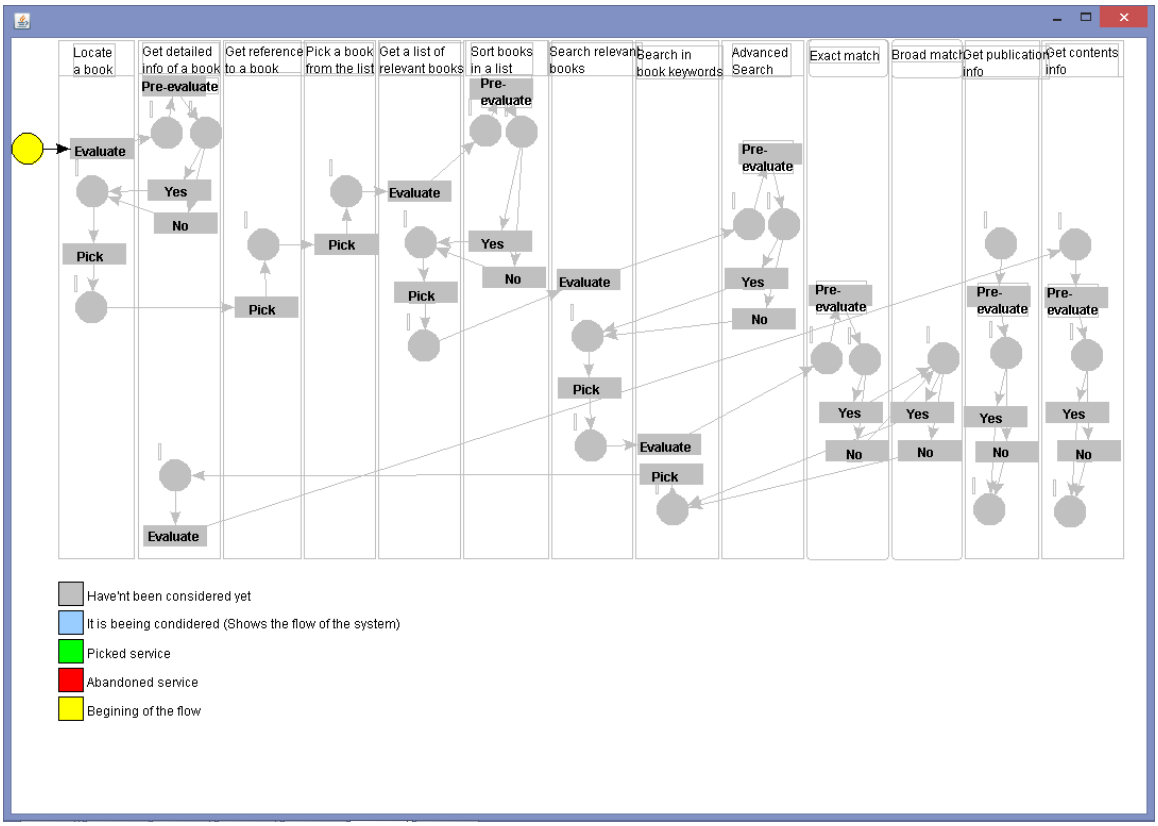


Figure 5.4. The Petri Net Based Visualization

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.

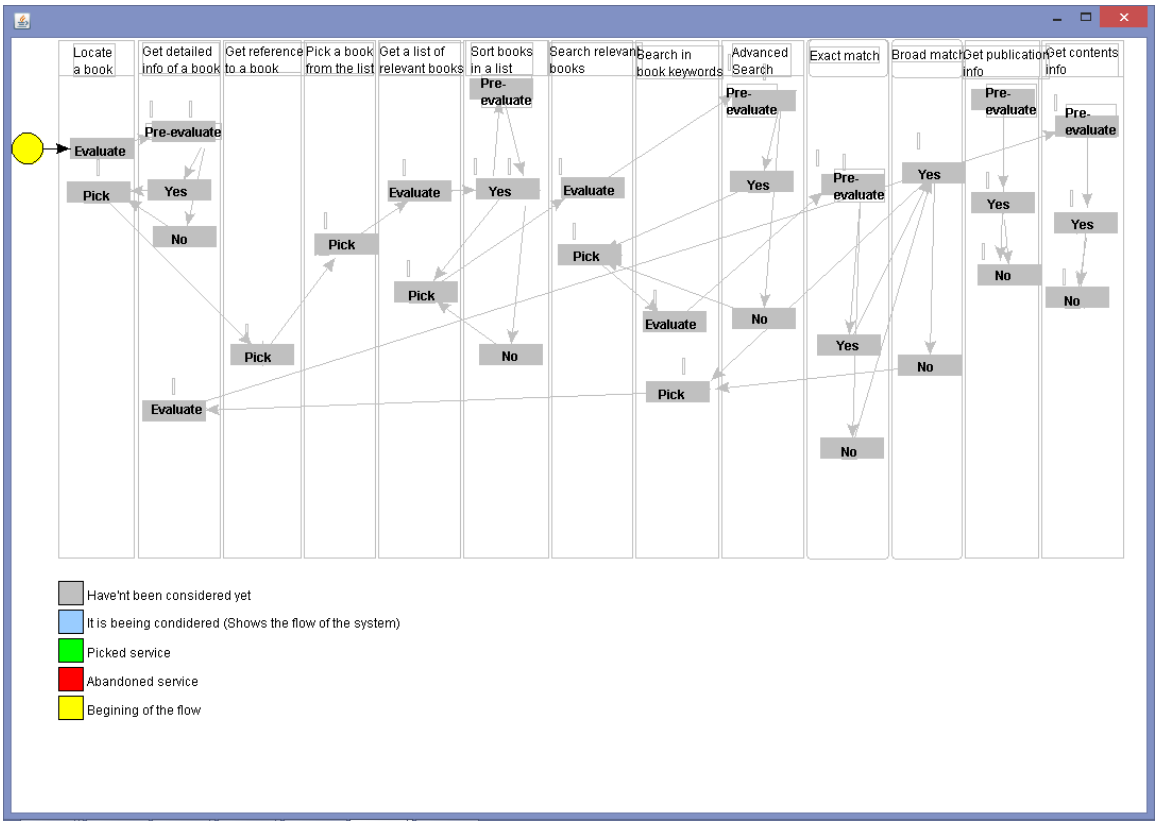


Figure 5.5. The Direct Graph Based Visualization

Figure 5.5 shows Directed Graph Based Visualization. In case of Directed Graph Based Visualization all the requirements are shown as rectangles in the background of the visualized system and the directed graph is on top of it. Due to the two dimensional nature of the visualization it seems that two layers overlay.

Actions are represented by vertices of the directed graph. Whenever each of these actions takes place, the transition related to that task will be fired and the color of that transition will turn to blue and the flow of the system will be presented by color changing. The requirements are visualized in the exactly same manner as in the Petri Net Based visualization, the selected requirements are marked by green whereas dropped ones are colored red.

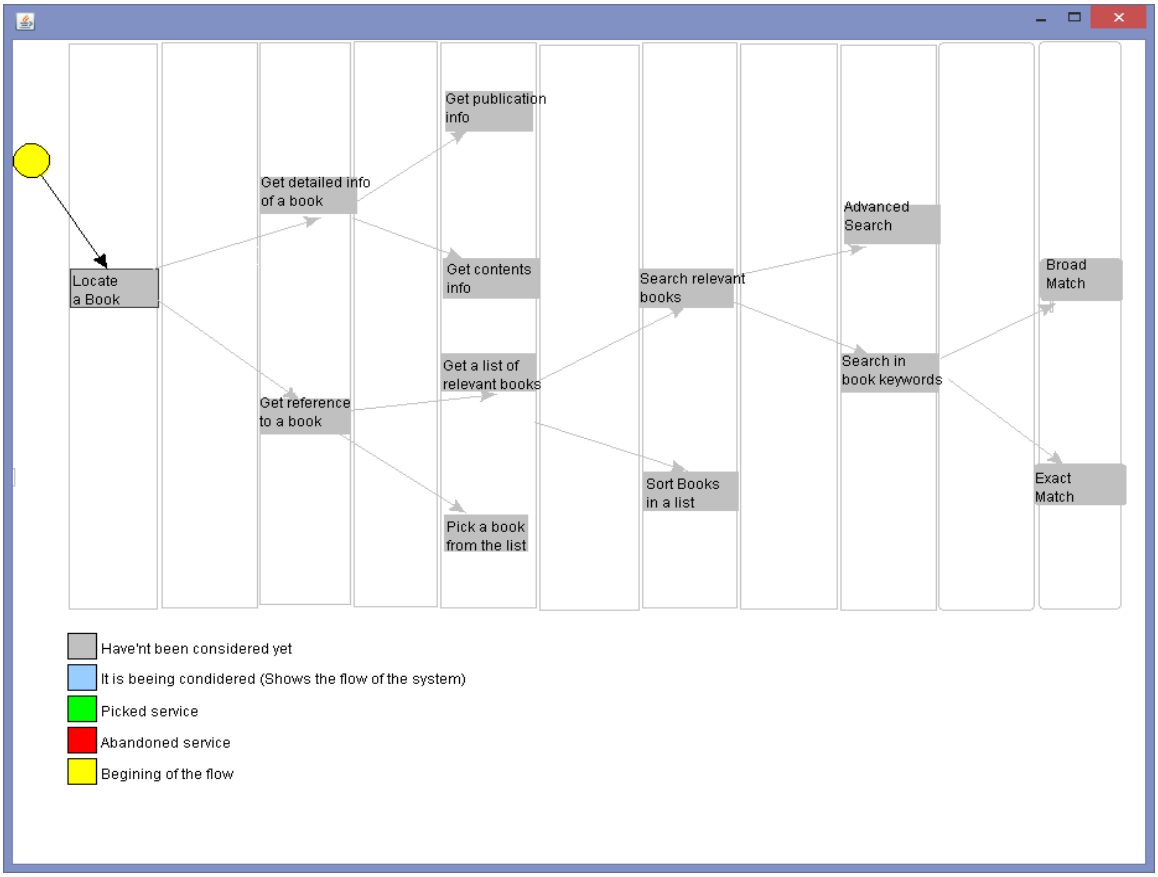


Figure 5.6. Requirement Model Based Visualization

The Requirement Model Based visualization does not visualize the actions taken at each step. This reduces the complexity of the visualization and helps the users concentrate on the requirements. The requirement model depicts requirements and their relationships between each other. When a requirement is selected the block representing that particular requirement turns green, whereas when a requirement is dropped the block turns red.

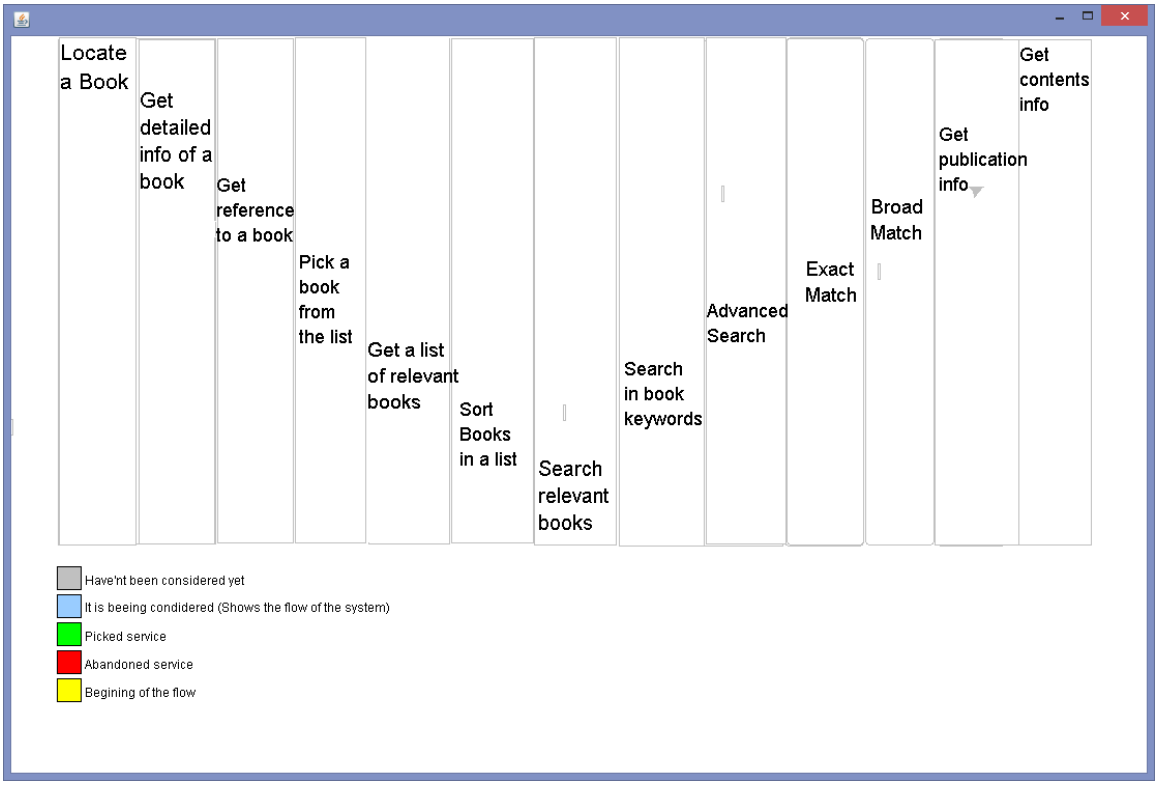


Figure 5.7. Block Based Visualization

The Block Based Visualization is the simplest among all the visualizations implemented and it only visualizes the requirements of the requirement elicitation process. In case of this visualization even the relationships among the requirements is not considered. The block representing a particular requirement turns green when the requirement is picked or it turns red when the requirement is dropped.

5.2. Usability Study

Usability can be defined as the ease with which a user can learn to operate, prepare inputs for, interprets outputs of a system or component [46]. The usability of an interface is a measure of the effectiveness, efficiency and satisfaction with which specified users can achieve specified goals in a particular environment with that interface [47].

Usability concerns how easy computer systems are to use. Usability is often distinguished from utility, which more concerns functionality. As the definitions above make clear, usability covers many aspects of the use of a system. For example, we must consider ease of learning, ease of regular use, memorability, error handling, and even subjective satisfaction. We must also consider the kinds of users who are likely to use the system, and what kind of goals they will be trying to achieve. Once we determine the aspects of usability and users that are most of interest, we can then conduct usability studies accordingly [47].

The usability study is conducted to study the usability of a system. The usability is a quality characteristic of the system which deals with the system`s interaction with humans. It helps in knowing the practical efficiency of the system from the usability point of view. Usability is very abstract concept and the usability study helps in studying the abstract concept of usability with the help of some usability attributes which are accurate and measurable in nature. Following are the usability attributes used in the usability study:

- 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. [46]

These attributes can be measured by observing the users when they are working on the system. An interview can also be conducted with the participants to get feedback from them after they have completed a task using a particular system. An interview also provides us a way to get feedback from the user, what they feel about the system and their reaction after using a new system.

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 [46]. Usability is about how the system interacts with the user [47]. Usability engineering defines the final usability level and ensures that the software under usability testing reaches that level [47].

The usability attributes can be measured by observing the users when they are working with the system or by having interviews and questionnaires after they used the system. The interview and questionnaires should have the questions related to the interaction between the user and the machine. In [47] Gould and Lewis has proposed "Famous Rules" for a usability engineering. These famous rules are:

- Early focus on the users
- User participation in the design

- Coordination of the different parts of the interface
- Empirical user testing and iterative revision of designs

In [46] nine heuristics are proposed: simple and natural dialogue, speak the user's language, minimize user memory load, be consistent; provide feedback; provide clearly marked exits; provide short cuts; good error messages, and prevent errors.

There is another method cognitive walkthroughs uses more explicit, detailed procedure and conducts a more work-based usability analysis by testing real users when faced with the system. In order to analyze the quality of the interface in directing the user to accomplish a specific task following three simple questions are asked:

- 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?

The answer to all these questions should be a "yes", in case there is a "no" answer to any of these questions, problems may occur [46].

The usability engineering life cycle has three stages:

- **Predesign Stage**

  During the predesign stage of the usability engineering life cycle the emphasis is on the target user and the tasks that the end user will perform. In this stage the focus should be on the user, the nature and needs of the users' needs to be understood. There can be several different criteria which can provide us with useful information about the user. For example user's experience with similar

systems can be a very important factor in most cases. Usability goals should be set during this stage.

- **Design Stage**

  In the design stage the emphasis is on the proper implementation. The released system should be useable and useful for the user. Usually a prototype is designed based of the usability principles and the needs of the users. This prototype is tested with the real users, and feedback from these users is used to analyze if the design will meet required goals.

- **Post design Stage**

  The post design stage is the study of the product to be used in the field. The testing is conducted on the real system with the real users. In this stage the design can be revised and retested for the future versions of the product.

5.3. Proposed Usability Testing Method

In this research for the purpose of the usability testing of the system, both the analytical as well as empirical testing was conducted. During the different stages of the system development usability inspection and the cognitive walkthrough methods were used. The famous rules were used wherever they were applicable and feasible in the system. The main purpose was to improve the usability of the system by enhancing the usability attributes.

To improve the usability of the system, the interactive software visualization approach has been implemented. It is expected to improve the usability of the system by introducing different types of visualizations for different types of users based on the knowledge and understanding of these users. The existing system has been improved and enhanced. The user can change visualization being shown to him/her. The system tries to check the level and preferences of the user and provide the user with the best suitable visualization based on the responses of the user. In order to check if the proposed approach produced the expected level of results a usability study was conducted.

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. Different visualizations has been designed in a way such that they are consistent with the overall design of the system. The error messages and the instructions to the users has been changed and more self-explanatory in the proposed system.

For the empirical testing of the usability of the system, both the modified and the original system will be tested by users with different level of knowledge from computer science and other departments. A questionnaire will be prepared to understand the user`s interaction with the system. It will contain questions related to comparison between the systems. The questionnaire will include questions for providing new ideas and suggestions for improving the usability of the system.

Chapter 6

Usability Study Results

6.1. Introduction

In order to check the usability of the system, both existing and proposed system were subjected to usability evaluation by users with varying levels of software development experience. It was assumed that in general, computer science students have a higher experience in software development compared to the students from other departments. Moreover the level of expertise and knowledge of the software development among the computer science varies based on the level of study. Graduate students from the computer science department are assumed to have the highest level of expertise.

	Computer Students		Other Departments	
	Existing	Proposed	Existing	Proposed
Undergrad (1st yr)	1	0	4	1
Undergrad (2nd yr)	1	2	2	3
Undergrad (3rd yr)	7	5	3	4
Undergrad (4th yr)	2	6	2	1
Graduate Students	7	5	10	12

Table 6.1 – Distribution of participants according to their academic level

6.2. Task Description

The participants had to follow few steps and guidelines while working with the system. The purpose of the study is explained to the user. Also the general functionality of the computer software for online book shopping and the the concept of software requirements and the process of requirement elicitation. Classic software development process (SDP) and software product line (SPL) were explained with a very well-known and simple concept of Lego.

The participants 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.

6.3. Requested Task

After explaining the few major concepts related to the system, the users were provided a task sheet. This task sheet described the task they were requested to complete. There were two different task sheets, one each for the existing and the proposed system. The functioning of the systems was explained in these task sheets.

The task for the participants 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" are the optional requirements. Any user can choose to pick or not to include these requirements in the system.

In the task sheet [Appendices B1, B2], the participants 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.
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.

So the task requires the participants to pick only these three requirements and abandon other four other optional requirements. Considering all the requirements that a participant can choose to include in the system, it was decided to score each participant's work out of 7. So it one point was allocated to each correct picking and similarly one point for abandoning a requested requirement. The variance between the maximum and minimum

score will be 7. In case a participant picks only the required 3 requirements, 3 points will be added to the base score 0, and another 4 points will be added for abandoning all the requirements that are not needed to be picked. In this case the participant will score maximum 7 points. On the other hand if the participant picks all the not required requirements and abandon all the required ones, the score will be 0.

The time taken by the participants were also recorded. The start time being the time when the participant actually started using the system and the end time being when the participant was done with selecting all the requirements. The duration of the time the participant spent to finish the tasks with the system was calculated in order to investigate, on average, how long it takes for the participants to finish the task. This time is used to calculate the efficiency of the system.

Thus the efficiency of both the existing and the proposed systems can be calculated from the time results found during the study. Similarly the scores can be used to calculate the error rate of the system. Participants were asked to fill questionnaire, the responses to the questions in the questionnaire were used to calculate the other important factors such as learnability, efficiency and user satisfaction.

6.4. Questionnaire

The purpose of the questionnaire was to get feedback from the participants so that we can analyze the usability of the system. The questions of the questionnaire were designed in a way that they provide information about the efficiency, error rate, satisfaction and issues related to the system. The questionnaire had 7 questions in all. The results gained from the participants' opinion about the system are used to analyze the usability.

6.4.1 Learnability

Learnability is how fast and easily new users can work with a new system for the first time. The learnability of the system is analyzed with the users' opinions about easiness to use the new system and user's understanding of the system. The first two questions in the questionnaire are used to check the learnability of the system. These questions were:

1. From 1 to 10, how easy it was for you to work with the system and customize an online book shopping service? (1 hardest, 10 easiest)
2. From 1 to 10, how many points are you going to give to your overall understanding of the system? (1 minimum, 10 maximum)

6.4.1.1. Easiness

The first question in the questionnaire asked the participants to rank the level of easiness of working with the system. The participants provide a rank based on their experience with the system. They were asked to rank it on the scale of 10, 10 being the easiest and 1 being the hardest.

In Figure 6.1, average easiness points and average scores have been categorized for the computer science students and the students from the departments groups. The average of both the score and the easiness rank are higher in case of the proposed system.

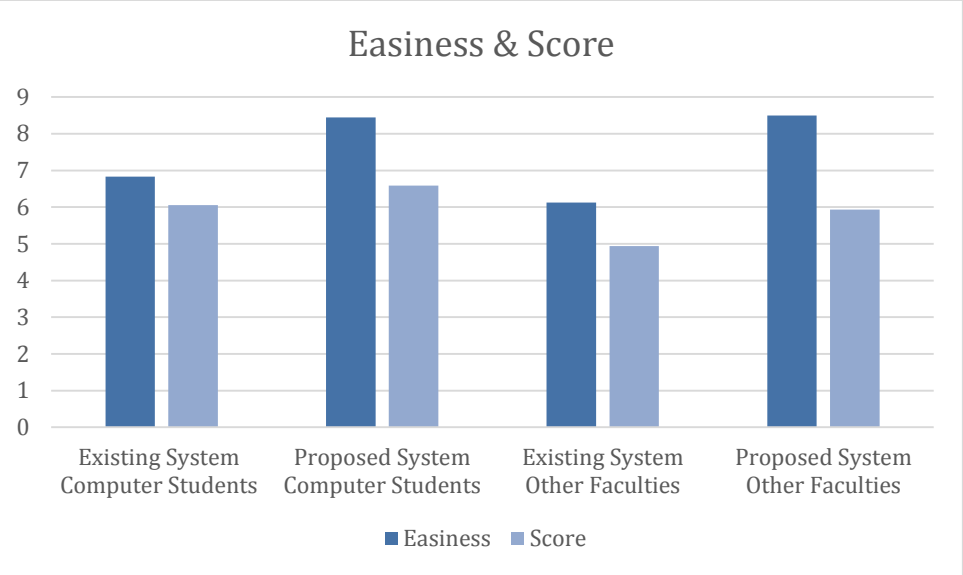


Figure 6.1 Comparison of Easiness and Score

6.4.1.2 Understandability

The second question on the questionnaire dealt with the understandability of the system. The participants were asked for their opinion about their understanding of the system. Similar to the first question the participant had to rank on a scale of 10, 10 being the highest and 1 being lowest.

As the feedback to the question only tells us about the participant's opinion about his understanding of the system it does not provide a complete picture. In order to make this more rank more meaningful 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. [VS]

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

Understanding will be greater if a participant makes correct decisions working on the system and achieves the goals in lesser time. The ‘Score’ represents how successful a participant was in achieving the required goals. Similarly ‘Time’ is the time taken by the participant to finish the task.

Figure 6.2 highlights the comparison between Understandability and Easiness values for both the systems. The Understandability value in the interactive system is greater than in existing system for both the groups. Since both easiness and understandability for the proposed system is greater than the existing system, it can be logically concluded that the learnability of proposed system is higher.

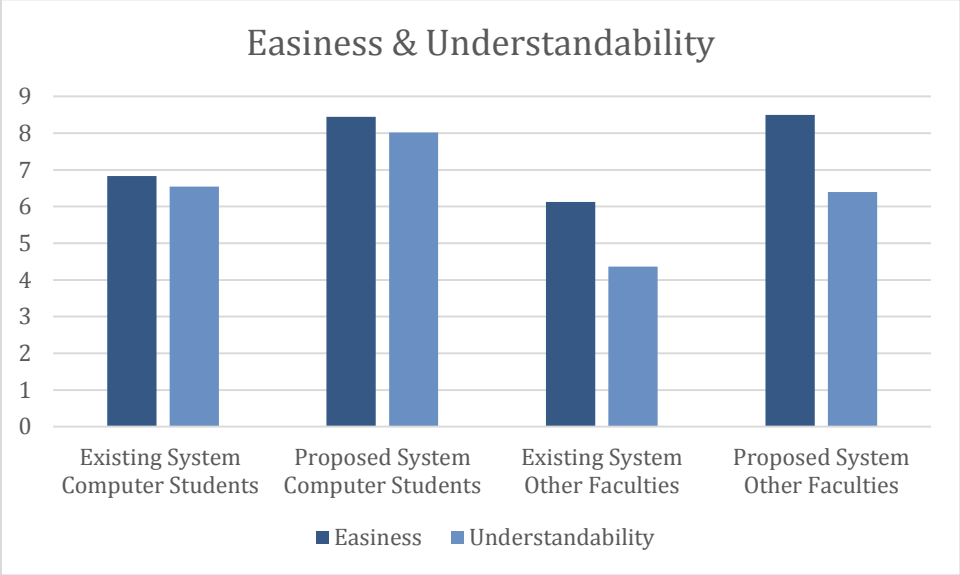


Figure 6.2 Comparison of Easiness and Understandability

6.4.2. Efficiency

The next factor in the usability study is efficiency. Efficiency can be defined as the number of tasks successfully completed by the user in a specific time interval. In our study every participant was asked to do perform the same tasks to achieve same goals.

The number of successfully completed tasks in this case is same as the score of the participant. So the efficiency is calculated by the using the score of the participant and the time taken by the participant to complete the required tasks.

Figure 6.3 provides the comparison between efficiency scores for all the users for the existing system as well as the interactive system. The efficiency of the users who used the interactive method irrespective of their level of knowledge was higher than the users who used the Petri Net based visualization system. So it can be concluded that the proposed system helps users become more efficient, more accurate and faster.

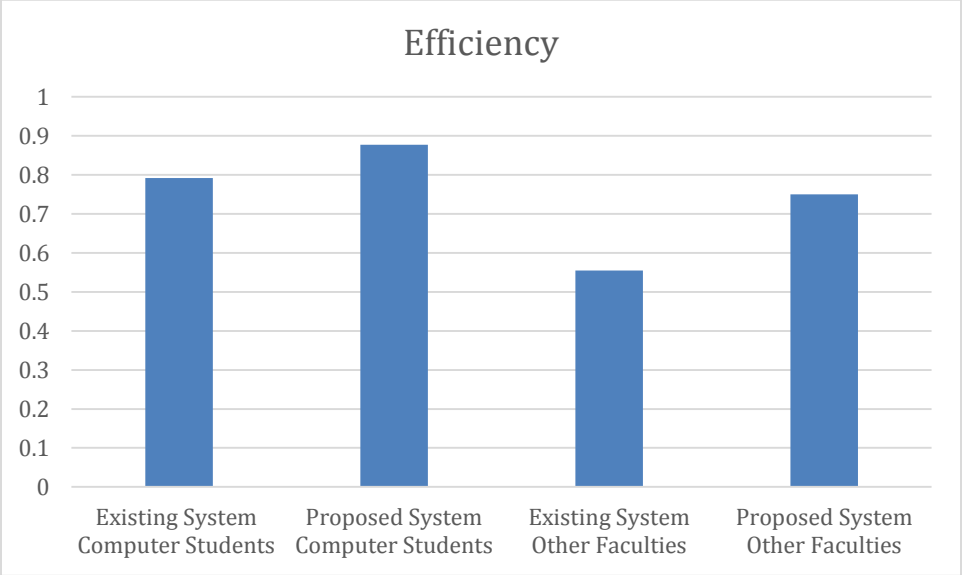


Figure 6.3 – Comparison of Efficiency between different groups

6.4.3. 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. Error rate is the percentage of errors a user make while working with the system.

The average error rate for the computer students using the existing system is 13.44%, it comes down to 5.88% for the computer students using the proposed system. Similarly the average error rate for the students from other departments is 29.46% when using existing system. Whereas while using interactive system the average error rate for students from other departments comes down to 15.18%.

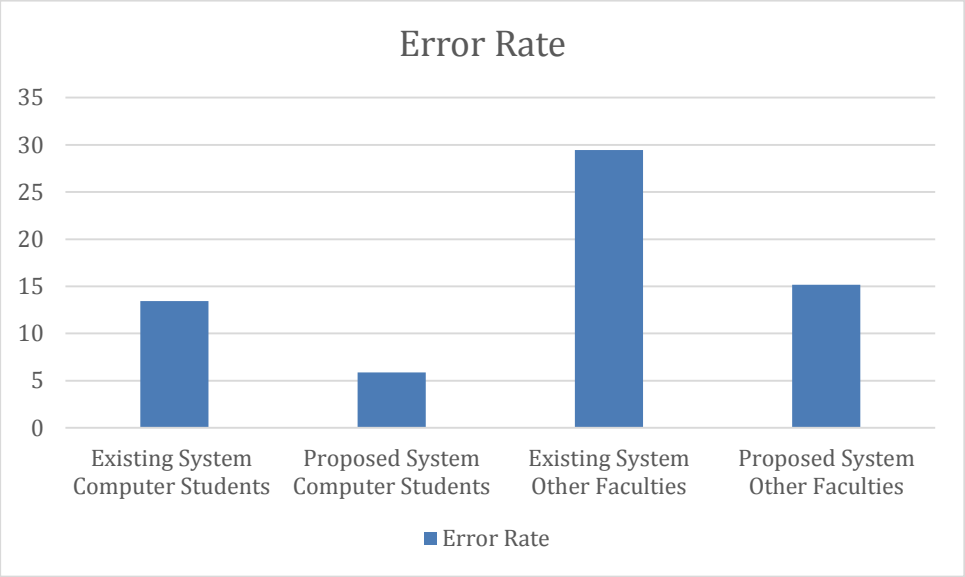


Figure 6.4 Error Rate Comparison

6.4.4. User Satisfaction

Another major factor in the usability study is the user satisfaction. As the name suggests it tells us how satisfied a participant was after working with the system. The third question in the questionnaire deals with the user satisfaction. The participant needs to rate the satisfaction level with the system on a scale of 10, where is 10 is completely satisfied and 1 not at all satisfied.

The average score given by computer science students to the existing system is 7.22, this increases to 8.11 for the interactive system. Similarly average satisfaction score for the existing petri net based visualization system for the students from other departments is 7.43, and this increases to 8.37 out of 10 in case of the interactive visualization system. When we compare the satisfaction levels of both the systems, the satisfaction is higher for the proposed interactive method.

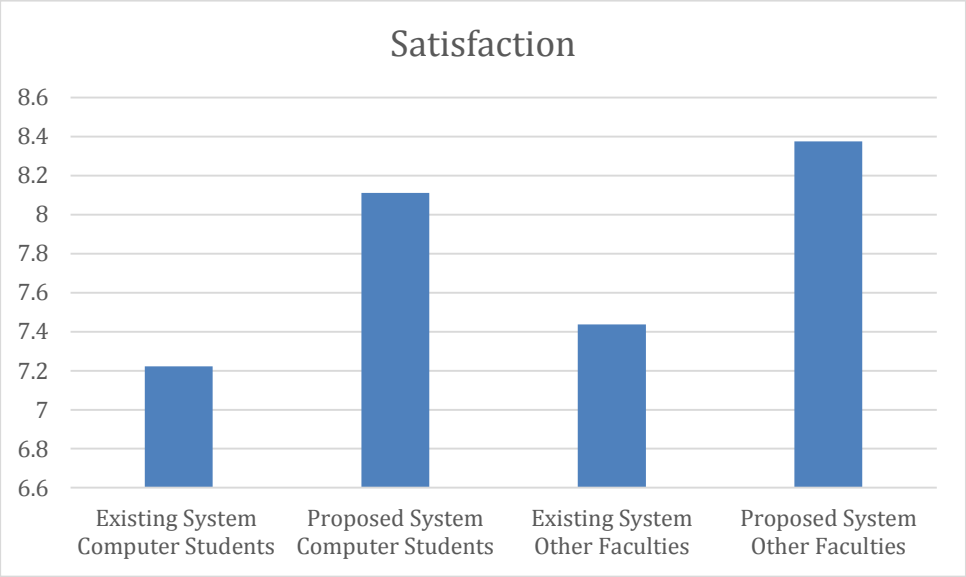


Figure 6.5 Comparison of Satisfaction

6.4.5. Necessity

There was one particular question which was just asked to the users who used the interactive system. This question was related to the users' opinion about the necessity of the interactive visualization. The question was:

Do you think the interactive visualization was necessary for the system?

a. *Necessary* *b. No difference* *c. Not Necessary*

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.

83% of the computer students, 81% of the students from other departments, and on the whole 82% of all students believed that the interactive approach of visualization is necessary for the system. 11% of computer students and 14.2% of the students from other departments and on the whole 12.8% of all the students found that the interactive approach of visualization is not useful enough and there was no difference for them for it to exist.

The percentage of computer students who found the interactive approach of visualization not necessary and probably more confusing for working with the system was 5.5% and this number for students from other departments was 4.7% and the overall figure was 5.1%. Therefore, a vast majority of the students preferred using the interactive system and believe it helps in better understanding of the system.

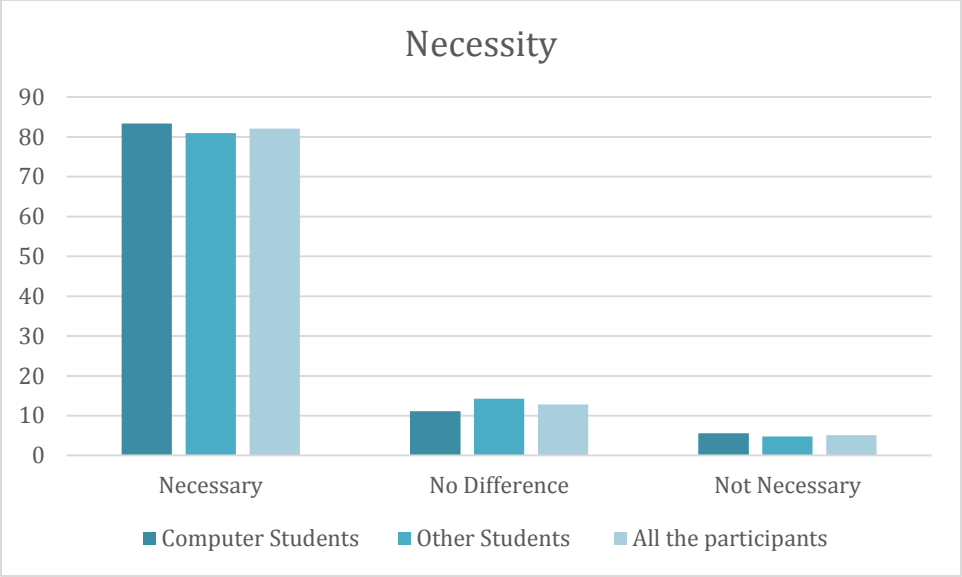


Figure 6.6 Necessity Comparison

6.4.6. Problems

In the questionnaire for the existing Petri Net based system there was a question related to the problems with the system. The users were asked about their opinion regarding what they consider as a problem with the system. 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. Comments are long and not self-explanatory.

b. Petri Net Visualization is complicated

c. Lack of alternative visualizations

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

Existing System, Computer Students:

1. Lacks in control.
2. System is complicated.

Interactive interface, Computer:

1. Drag/Drop will make system even better.
2. Direct Interaction with the visualization elements.
3. Exciting

6.4.7. 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. Overall 78.2% of the users believed that the SPL could be beneficial to the software development and customization.

The last question in both the questionnaire asked the users their overall opinion about the system. 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 interactive system is higher than the improvement score for the existing system.

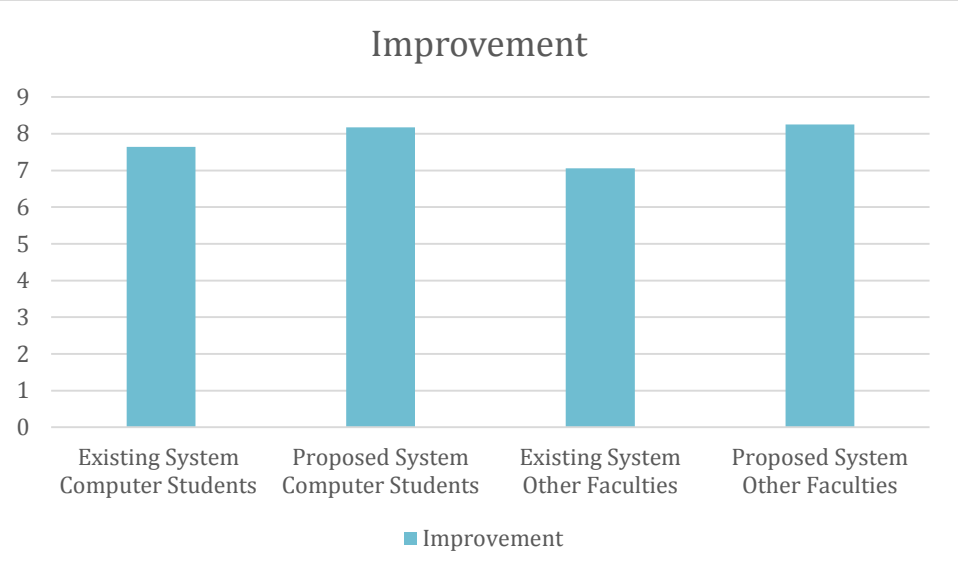


Figure 6.7 Improvement Comparison

In Table 6.2, average results for users have been divided and shown based on their academic background, which is either computer science or business field.

	Computer Students				Others			
	Existing		Proposed		Existing		Proposed	
	AVG	STDV	AVG	STDV	AVG	STDV	AVG	STDV
Easiness	6.83	1.12	8.44	0.83	6.13	1.29	8.50	1.05
Understandable	6.54	2.20	8.01	1.58	4.36	1.23	6.40	1.90
Satisfaction	7.22	1.27	8.11	1.33	7.44	1.18	8.38	1.05
Improvement	7.65	1.06	8.18	1.23	7.06	1.04	8.25	1.05
Time	8.33	1.94	7.89	1.76	9.19	1.91	8.13	1.13
Score	6.06	0.97	6.59	0.59	4.94	1.26	5.94	0.99
Error Rate	0.94	0.97	0.41	0.59	2.06	1.26	1.06	0.99

Table 6.2 Average results for all the students

To sum up all the results collected from participants for both interactive system and Petri Net based system:

Time spent on the interactive system is less as compared to the Petri Net based system for both the groups. There has been a rise in the easiness and understandability score both groups. The understandability score for non-computer science users is 4.36 for the Petri Net based system where as for the Interactive system it is 6.40. Satisfaction score for non-computer science students is higher than computer science users for both the systems. This can be due to different levels of expectations in case of two different groups, or it can be said that students with computer experience can have more realistic exceptions of a user interface than people with no computer background. The satisfaction score for both groups of users were higher for the Interactive system. Both the groups considered that SPL has improved software development and software customization process. The error rate for both computer science students and non-computer science students was reduced by to almost halves. Thus it can be concluded that the Interactive system has been successful in improving the user experience with the help of different visualizations and by choosing the best suitable visualization for a user.

On the whole, based on the collected results from both computer science and non–computer science students, it can be concluded that the interactive system improves the efficiency of the software customization process. Also it makes the system much more attractive for the users with lesser knowledge about the software development process.

.

CHAPTER 7

Conclusion and Future Work

7.1. Conclusion

In this research, a study has been carried out to conclude that when Interactive Approach of Software Visualization is applied on the previously developed 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 was conducted in a number of steps. Initially the existing text based requirement elicitation system and the Petri Net based Visualization systems were studied. The usability study conducted by Vida Sadri for these systems was studied which motivated to develop an interactive approach of visualization for the system. A usability study was conducted on a group of students from different departments and diverse academic backgrounds to justify that the proposed design can improve the usability of the system.

The results of the study shows that overall users had a positive opinion about using both Interactive system and Petri Net based visualization system. However, based on users opinions on average all usability parameters, which are Learnability, Efficiency, Error-rate and User satisfaction have been improved compared to Petri Net Based Visualization interface system. Besides, on average users of interactive 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.

7.2. Future Work

Despite the fact that the interactive approach has improved the usability and quality of the software customization system, the results of the usability study show that there is still a lot of scope for improvement. In fact, the proposed framework can be considered as an early attempt of interactive interfaces exclusively in the field of software customization, future work will need to implement this approach in different research areas. Currently there are four types of visualizations used for different users, further work should be done to look for more visualizations which can further enhance the usability of the system. Morphing can also be implemented in future versions of this system, where a visualization gradually changes to produce other visualization.

During the usability study it was found out that the users are interested in using an interface in which they can have a direct interaction with the graphical interface instead of indirect dialogue-based communication. So a drag and drop kind of system can be researched where the user can interact and choose the required services by simply dragging and dropping them.

APPENDICES

Appendix A

Questionnaire for Interactive Visualization Interface

Questionnaire for Interactive Visualization Interface

Name: Department:

Graduate: Undergraduate (Year:)

Email (for participation in the draw):

Do you have 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? (1 hardest, 10 easiest)

2. From 1 to 10, how many points are you going to give to your overall understanding of the system? (1 minimum, 10 maximum)

3. Do you think the interactive visualization was necessary for the system?
   a. Necessary b. No difference c. Not Necessary

4. What score will give to your overall satisfaction of working with the system? (1 minimum, 10 maximum)

5. How effective do you think is the interactive visualization in preventing the user from making mistakes? (1 minimum, 10 maximum)

6. What changes do you think will make the visualized interface more useful? If possible add comments
   a. Having direct interactions with the visualized interface
   b. Removing all the comments

7. Which approach will you prefer for software development?
   a. Classical Software Development Process
   b. Software Product Line
   c. No Idea

8. How much do you think the Software Product Line improved the software development process? (1 minimum, 10 maximum)

Appendix B

Questionnaire for Existing Interface

Questionnaire for Existing Interface

Name: Department:

Graduate: Undergraduate (Year:)

Email (for participation in the draw):

Do you have 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? (1 hardest, 10 easiest)

2. From 1 to 10, how many points are you going to give to your overall understanding of the system? (1 minimum, 10 maximum)

3. What score will give to your overall satisfaction of working with the system? (1 minimum, 10 maximum)

4. How much do you think the error making in the system is related to the design of the interface? (1 minimum, 10 maximum)

5. Which of the following do you consider as a problem of the system? Choose more than one if applicable and add your ideas if you wish?
   a. Having direct interactions with the visualized interface
   b. Comments are long and complicated
   c. Interface is complicated
   d. Interaction with the system is time consuming

6. Which approach will you prefer for software development?
   a. Classical Software Development Process
   b. Software Product Line
   c. No Idea

7. How much do you think the Software Product Line improved the software development process? (1 minimum, 10 maximum)

Appendix C

Task sheet for Existing System

The following figure illustrates the task sheet, which was required to be read by the participants before working existing system for the purpose of usability investigation.

Task Sheet:

Imagine as a part of your job requirement you are asked to work on the development of a software system. The task you are asked to do is to customize an Online Book Shopping Service with the system.

You should work with the provided system and communicate with it by clicking on the buttons on the system. The system offers both essential features that you should just admit by clicking "OK" and pick or abandon the variable features based on the 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 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 concept or synonyms are not preferable.
3. This system should sow the user the information about the contents of the book.

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

Appendix D

Task sheet for Interactive System

The following figure illustrates the task sheet, which was required to be read by the participants before working existing system for the purpose of usability investigation.

Task Sheet:

Imagine as a part of your job requirement you are asked to work on the development of a software system. The task you are asked to do is to customize an Online Book Shopping Service with the system.

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

The Interactive system analyzes the level of knowledge of the user and provides the best suited visualization for the user. The user can change these visualizations according to his/her needs and system, system takes into consideration these preferences when analyzing the preferred visualization for the user.

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:

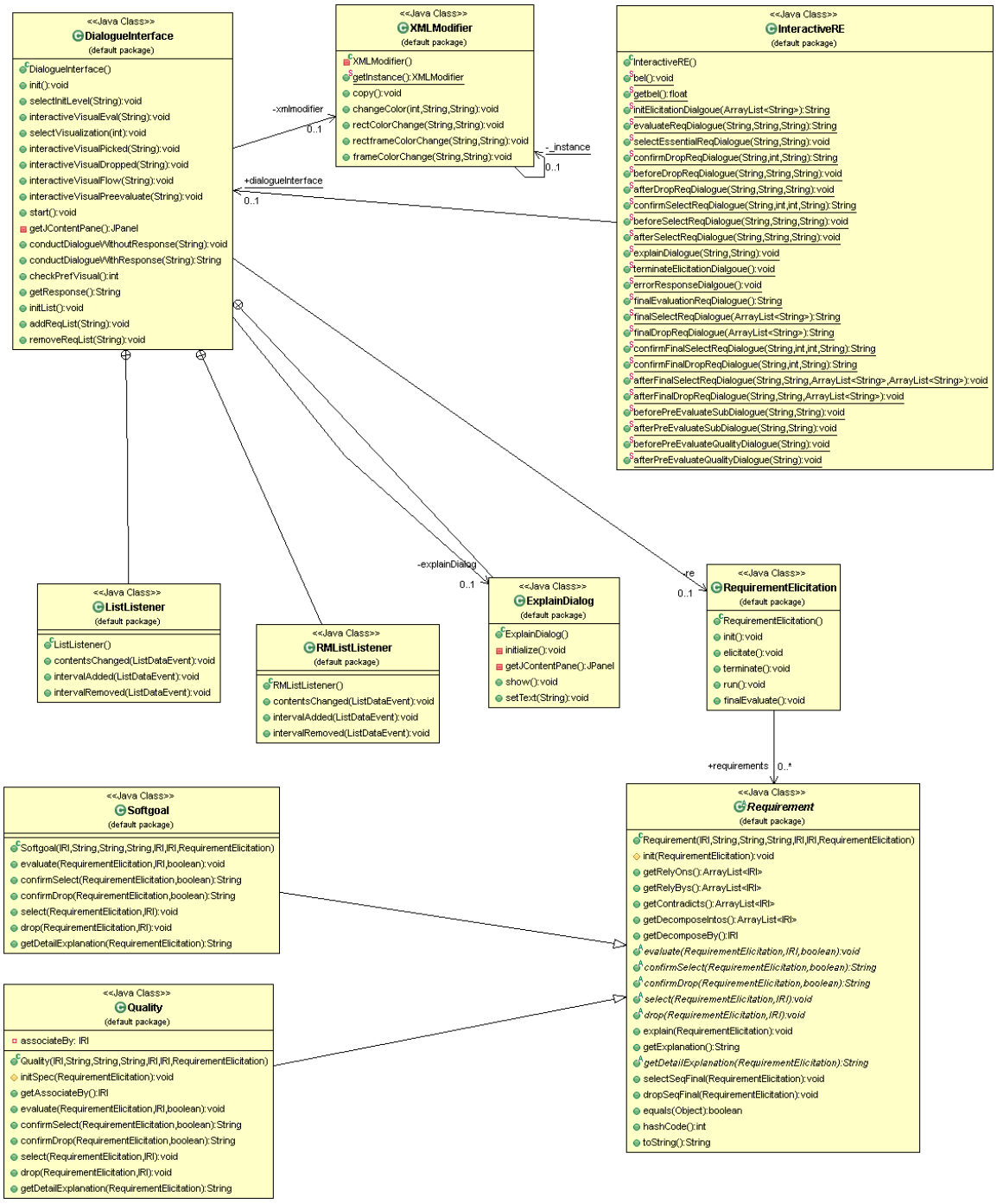
1. The Online Book Shopping should have the sorting feature. This means after the results should be shown to the users in a sorted basis.
2. It should search based on the exact word that is entered to the system by the user. Any similar concept or synonyms are not preferable.
3. This system should sow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

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

Appendix E

System Class Diagram

The following figure illustrates the class diagram for the implementation.



REFERENCES

[1] S. Eicker, T. Spies, C. Kahl. "A Virtual Reality Application for Software Visualization," in Proc. 4th IEEE International Workshop on Visualizing Software for Understanding and Analysis, 2007, pp 108-111.

[2] M. J. Pacione. "Software visualization for object-oriented program comprehension," in Proc. 26th International Conference on Software Engineering, 2004, pp. 63-65.

[3] Chao Ma, Yanxiang He. "An Approach for Visualization and Formalization of Web Service Composition," in Proc. International Conference on Web Information Systems and Mining, 2009, pp 342-346.

[4] A.R. Teyseyre, M.R. Campo. (2008, Jul.). "An Overview of 3D Software Visualization," IEEE Transactions on Visualization and Computer Graphics. 2005, pp 64-69.

[5] Rick Rabiser, Paul Grünbacher, Deepak Dhungana. (2009, Nov.). "Requirements for product derivation support: Results from a systematic literature review and an expert survey." Information and Software Technology. [Online]. 52(8), pp. 324-346. Available: http://www.sciencedirect.com/science [Feb. 14, 2011].

[6] Bowen Hui. "Automatic Software Customization: A Methodology for Learning Individual Preferences." Internet: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.159.1955, [Feb. 14 ,2010].

[7] H.M. Kienle, H.A. Muller. "In Proc. 4th IEEE International Workshop on Visualizing Software for Understanding and Analysis, 2007, pp 2.

[8] Michael P. O'Brien. "Software Comprehension – A Review & Research Direction." Internet: www.st.cs.uni-saarland.de/edu/empirical-se/2006/PDFs/brien03.pdf, Nov, 2003 [Apr. 14, 2011].

[9] Denis Gracanin, Kresimir Matkovic, Mohamed Eltoweissy. (2005, Sept.). "Software Visualization." Innovations in Systems and Software Engineering, A

NASA Journal. [On-line]. 1(2), pp. 221-230. Available: www.cg.tuwien.ac.at/research/publications/2005/gracanin-2005-soft/gracanin-2005-soft-PDF.pdf [Feb, 2011].

[10] Alfredo R. Teyseyre, Marcelo R. Campo. "An Overview of 3D Software Visualization." IEEE Transactions on Visualization and Computer Graphics, vol. 15, pp. 87-105, July. 2008.

[11] Juergen Rilling, S.P. Mudur. "3D visualization techniques to support slicing-based program comprehension." Computers & Graphics 29, vol. 29, pp. 311-329, June. 2005.

[12] B.A. Price, I.S. Small, R.M. Baecker. "A Taxonomy of Software Visualization." Proceedings of the Twenty-Fifth Hawaii International Conference on System Sciences, 1992, pp 597-606.

[13] Jiming Liu, Chi Kuen Wong, Ka Keung Hui. "An Adaptive User Interface Based On Personalized Learning." IEEE Intelligent Systems. 2003, pp 52-57.

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

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

[16] Rajaprabhu Dhanapal, "An Approach for Contextual Control in Dialogue Management with Belief State Trend Analysis and Prediction", M.S. thesis, CS. Dept., UWindsor, Windsor, ON, 2012.

[17] Vida Sadri, "A Petri-Net Based Approach of Software Visualization for Software Customization", M.S. thesis, CS. Dept., UWindsor, Windsor, ON, 2012.

[18] 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.

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

[20] 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.

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

[22] 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.

[23] 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.

[24] 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.

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

[26] 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.

[27] Zhang, N. Zhou, Y. Chee, A. Jalaldeen, K. Ponnalagu, R. Sindhgatta, A. Arsanjani, F. Bernardini, “SOMA-ME: A platform for the model-driven design of SOA solutions”, IBM Systems Journal, Vol. 47, pp 397 – 413, 2008.

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

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

2010.

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

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

[32] 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.

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

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

[35] 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.

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

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

[38] Li, X., and Mugridge, R. 1994, “Petri net based graphical user interface specification tool”, In Software Education Conference, 1994.

[39] 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.

[40] 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.

[41] 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.

[42] 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.

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

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

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

[46] Nielsen, J. , "The usability engineering life cycle", IEEE Computer, Vol. 25(3), pp12–22, 1992.

[47] 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.

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

[49] 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.

[50] 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.

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

[52] 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.

[53] 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.

[54] 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.

[55] Teyseyre, A and Campo, R. M., “An overview of 3d software visualization”, IEEE TVCG, Vol.15, pp. 87–105, 2009.

[56] N A Stanton, M J Ashleigh, A D Roberts, F Xu, “Testing Hollnagel's Contextual Control Model: Assessing team behavior in a human Supervisory control task”, International Journal of Cognitive Ergonomics, 2001.

[57] R. Dąbrowski, K. Stencel and G. Timoszuk and I. Crnkovic, V. Gruhn, M. Book and Eds. “Software is a directed multigraph” ECSA, ser. Lecture Notes in Computer Science, vol. 6903, pp. 360-369, 2011, Springer

[58] Ashley Aitken, Vishnu & Ilango,”A Comparative Analysis of Traditional Software Engineering and Agile Software Development”, 46th Hawaii International Conference on System Sciences (HICSS), 2013.

[59] G. Timoszuk, R. Dabrowski, K. Stencel, C. Bartoszuk, “Magnify - A new tool for software visualization”, Federated Conference on Computer Science and Information Systems, pp. 1485-1488, 2013.

[60] A. Barbar, A. Ismail, “A framework for autonomic software customization”, International Conference on Electronics, Computer and Computation, pp. 334-338, 2013.

[61] X. Yuan, X. Zhang, “An Interactive Approach of Online Software Customization via Conversational Web Agents”, IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing, pp. 327-334, 2013.

[62] A .Assila, Houcine Ezzedine & M.S. Bouhlel, “A web questionnaire generating tool to aid for interactive systems quality subjective assessment”, International Conference on Control, Decision and Information Technologies, pp. 815-821, 2013.

VITA AUCTORIS

Name	Manpreet Singh Kaler
Place of Birth	Amritsar, India
Year of Birth	1986
Education	Bachelor of Technology, Computer Science & Engineering, Guru Nanak Dev University, India. Master of Science, Computer Science, University of Windsor, Canada.