PRACTICAL FILE Of
SOFTWARE ENGINEERING LAB

SUBMITTED BY:	SUBMITTED TO:
GAURAV BHARDWAJ	*Dr. KULDEEP KUMAR*
18103034	*ASSISTANT PROFESSOR*
GIRISH	*CSE DEPARTMENT*
18103035	
HARDIK PUNJ	
18103039	
CSE/5th SEMESTER	
GROUP: G2	

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

DR. B.R. AMBEDKAR NATIONAL INSTITUTE OF TECHNOLOGY JALANDHAR

Table of Contents

Table of Contents (FSR)

Document Review History

S. No.	Version History	What's New?	Recent submit
1.	Version alpha(α)	Simple form filling **UI,** full documentation	18/09/20
2.	Version Beta(β)	Improved Documentation and references, included demonstration	29/11/20
3.			
4.			
5.			
6.			
7.			

Group Member Details

Hardik Punj:

18103039

hardikp.cs.18@nitj.ac.in

Girish Kumar:

18103035

girishkk.cs.18@nitj.ac.in

995-152-7282

Gaurav Bhardwaj:

18103034

gauravb.cs.18@nitj.ac.in

628-391-3449

Title:

Automatic Resume Building Software

Abstract:

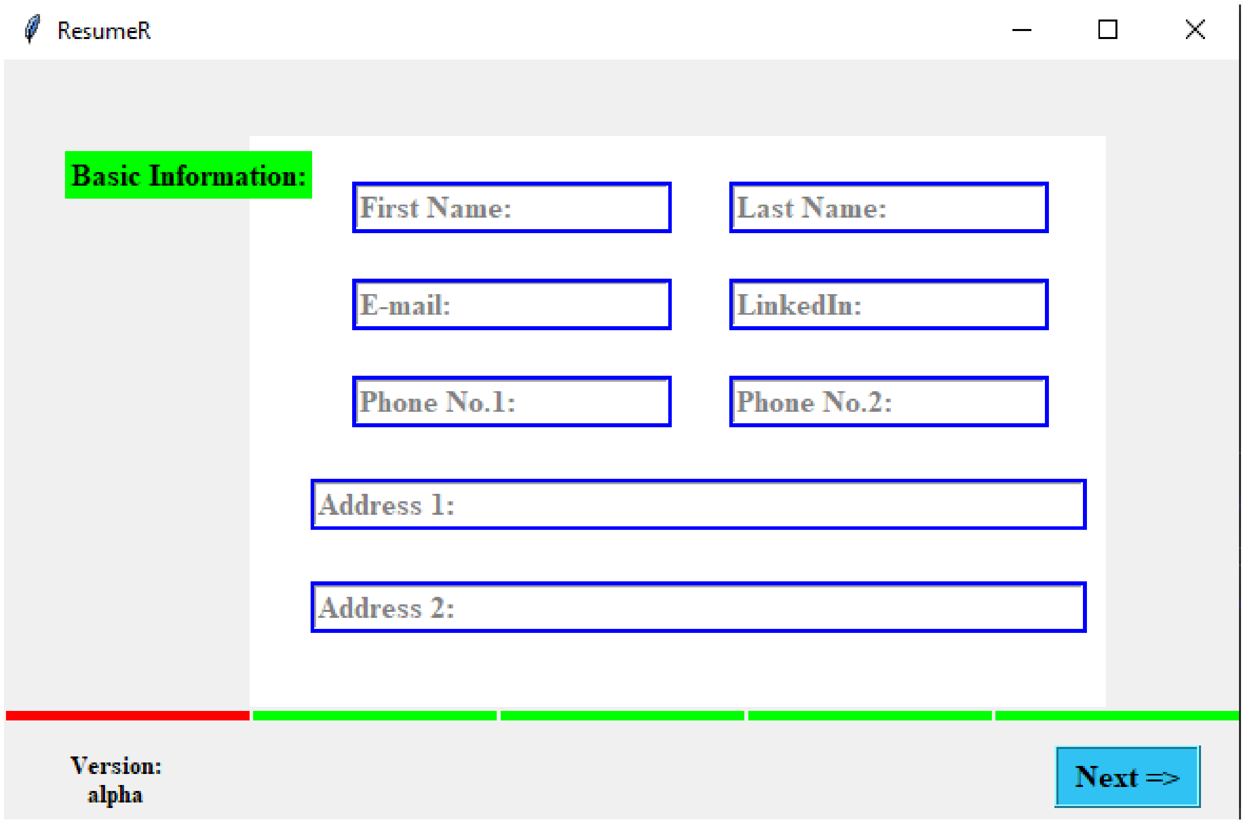
A resume is a document that contains a summary or listing of relevant job experience and education. The résumé or CV is typically the first item that a potential employer encounters regarding the job seeker and is typically used to screen applicants, often followed by an interview, when seeking employment. In the job search process, a well-written and well-designed résumé is essential. Our program does the thinking and writing for you. Resume Builder simplifies the work of finding the job by providing intelligent and user-friendly software Resume Builder can display the resume structure in a user-friendly format, so that you can choose which sections of the resume to publish. You still have to input your information onto the page or software for the resume builder to be able to complete your resume.

Keyword:

Resume, API, Database, Form, GUI, Python, Tkinter, modules, fonts, templates[5], editor, file formats.

Description of the project:

As you have known of Résumé[1]/ Portfolio[1] documents so you also know document builders[2] helps you create these perfect documents by asking your details. It is a great way to develop the documents with options for further modification as the software captures the information one time and reflects it on various designed templates. Most software are made online to create Resume but these do not allow us to make further editing, and features like printing the documents as PDF[5]/Word format[5] are not made free. Now, every Location does not have good internet connection, so may be the accessibility of the web sites would be less over there. Keeping these cases in mind, we'll try to provide an application that will be fully free of cost and with all text processing actions and with tons and tons of samples templates[6] of your interests. We plan to provide the services to save your data and also printing options with your preferred formats for free of cost.



First Name:

Last Name:

E-mail:

LinkedIn:

Phone No.1:

Phone No.2:

Address 1:

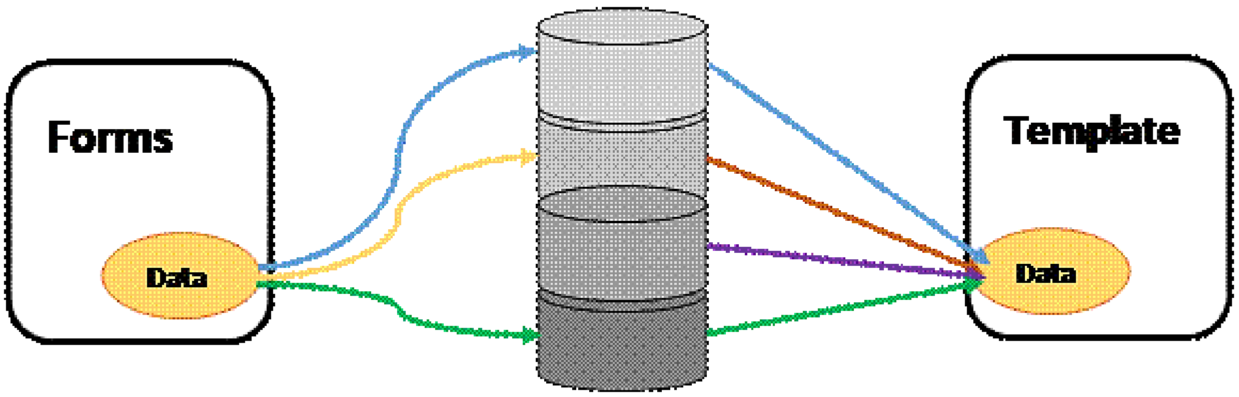
Address 2:

Version:
alpha

Form filling interface of the software

The application will be written and designed in **Python**[3] Language, **SQL**[4] database. With the help of modules like **Tkinter**[3], **tkinter-fonts**[3], etc. to prepare form like structure for entering your data and buttons for saving and navigation across forms and converting code files to execution files by using **Pyinstaller**[3]. Output will be shown as editable document. To explain its operation the forms are input taking widget in which are stored into a variable first and then are shifted to a data base in your system. In case of showing document, the file will get the data from database to fit in templates. Final feature will be you are able to choose from different templates, using different fonts sorted by different regions, languages, and job titles.

Saving and Retrieving Data form disk database



[6]

Previous work:

Some builders are purely 100% online and don’t have the functionality to allow the user to save their resume to other word processing software such as Microsoft Word, and often you can only print out what you have completed online. Some of the resume builders only provide you with a PDF copy of your resume which again restricts you for further adaption or updating.
Till now we have come across only MS Word in Windows line-ups for this purpose, But the problem is that only few templates are free and in-build, and new templates need to download and to pay some cost for the templates. And you need put your information from start every-time you change a template, which is frustrating. There is full freedom to design your own template.

Project scope and Contribution:

As placement season of most of the colleges is going to start, making Resume is a very hectic work for all the students. Also, many companies judge the candidature of a student just by his/her Resume. So it is necessary for the student to think beyond the third dimension while making the Resume.

- **Automated Resume Builder** is the Application which helps students to get their resume in hand just by filling up a simple form where important credentials need to be filled.
- The resume is downloadable in PDF format. Also, the user can log in again to access the previous resume that he had made.

Contribution in project:
In team of 3 people: one person will be following more than one jobs, eg: Coder + Documentation, Tester + Project designer, Tester + Resource Manager, Release Manager + UI Designer.

- 〉 In this project, I (Gaurav) will be project **coder + documentation** manager.
- 〉 I (Hardik) will be incharged of **Tester + resource manager + release manager**
- 〉 I (Girish) will be mainly in region of **Tester + sources of reference + release manager**

Technical Feasibility:

The project is technically feasible as described plan in work schedule. We start by building basic form layout in separate modules (files) and then merging theses files, adding database technicality using **pymysql/sqlite3**[4] modules and end result will be output on simple scratched Text editor for saving and printing options. The system will be sharable by creating .exe (executional) file. Later on more features like language support, suggestions for templates acc. to job title and regions.

Risk analysis:

Inputs: Since application contains form filling so it is not allowed to enter wrong inputs. In order to solve, we will check if entered data is of respective types before adding to database and to move to next sections.

Work Plan/Schedule:

S. No.	Schedule	Time Period	Status
1.	Planning	1 Sept. - 15 Sept.	●●⟩
2.	Building a Software	10 Sept. – 30 Sept.	●●⟩
3.	Creating forms into modules	10 Sept. – 20 Sept.	●●⟩
4.	Combining modules using buttons	20 Sept. – 15 Nov.	●●⟩
5.	**Testing of software**	24 – 25 Sept.	●●⟩
6.	Create a database	24 – 28 Sept.	●●⟩
7.	**Testing of database**	29 Sept. – 30 Sept.	●●⟩
8.	Integrating database to application	1 Oct. – 30 Oct.	●●⟩
9.	Building a Template builder System	1 Oct. – 10 Oct.	●●⟩
10.	**Testing of Template editor**	11 Nov. – 15 Nov.	●●⟩
11.	Integrating Template builder to software	26 Oct.	●●⟩
12.	**Combining all modules into an application**	27 Oct. - 3 Nov.	●●⟩
13.	Add Printing and file conversion features	4 – 10 Nov.	
14.	**Testing of System**	11 – 12 Nov.	
15.	**Product Release**	15 Nov.	
16.	Collecting Sample templates files	21 – 22 Nov.	

17.	*Adding language support*	23 Nov. – 12 Dec.	
18.	*Adding sort by region and job*	13 – 23 Dec.	
19.	**Project Deployment**	24 Dec.	
20.	**Maintaince**	→	

Note:

- A: A will take more time than others
- *A:* A will take more time
- **A: Special need of checking performance**

Suggested deliverables:

A software for filling information into forms that automatically generates a resume file with options to edit, save, and print the document.

Any other details:

Not Applicable

Reference:

[1] Resume document: Wikipedia
[2] Document Builder: Java, Google APIs.
[3] Python Language: Python Documentation, Tkinter Modules, Pyinstaller Module.
[4] Databases: https://en.wikipedia.org/wiki/Database, SQLite3, PymySQL
[5] Formats: File Formats Wikipedia, Microsoft Word Format
[6] Templates: Resume-Templates, Downloadable Templates

Git/Github:

Git or github is a great source to get collective information on previously made projects. Reference:

- https://github.com/sramezani/resume-builde
- https://github.com/AmruthPillai/Reactive-Resume
- https://github.com/nitish6174/resume-generator

Table of Contents (SRS)

Document Revision History

S. No.	Version History	What's New?	Recent submit
1.	Version alpha(α)	Basic documentation	29/11/20
2.			
3.			
4.			
5.			
6.			
7.			

1. The Software Requirements Specification Outline

1. Introduction

1.1 Purpose and Scope

Our Plan is to provide you with a software to help your profile look the best of a kind, With help of an interactive Form filling experience. This software have functionalities like easy forms, various templates, one can create their own templates, once information is fed, data will stored in disk, give suggestion for best resume style sorted according to preferences of the jobs, companies (in which you want to apply), support for different languages for people around the globe, it is desktop system so network is not required, and since data will be stored one time so no need to put it again and again, of-course data updating will be there if needed.

1.2 Document Conventions

--None--

1.3 Definitions, Acronyms, and Abbreviations

None. List Definitions, Acronyms, and Abbreviations used in the SRS here.

1.4 Intended Audience and Reading Suggestions (Overview)

The intended Audience include all types of people like client, developers, testers, etc.

For client: This software is created for client to help them create resume document. The client is suggested to read the manual of the software before actually using the software to get familiarize with the user interface and working.

For developers: The future developers are suggested to read the manual for understanding the working, the developer should have through knowledge of Python Language as well as of its user interface modules for updating the software in future. Since software is free and open source so experienced developer can modify the source code for their own working.

For Testers: The testers should be having through knowledge of Python and GUI.

For marketing staff: They should read the manual of the software.

1.5 References

- https://python-docx.readthedocs.io/en/latest/
- https://py-googletrans.readthedocs.io/en/latest/
- https://pyinstaller.readthedocs.io/en/stable/

2. Overall Description

2.1 Product Perspective

This project of making a resume builder is to provide a software to help the user to create wonderful portfolios to help him/her get through the interview section when applying for a job in a company. One can use web-based builder software to get the job done but the issue in this approach:

- May be internet connections are weak in certain region.

- The data used in template are stored in web site's server database from where it may get leaked
- May be the template structures are fixed i.e. data can be updated only not the arrangement of the sections cannot be changed.

In context of on disk software, till now Microsoft's Word tool is only providing this functionality. The only problem in this software is that only templates are provided and the user has to put all data by themselves, and any time you want to create new document, data is to be updated every time, plus only few templates are provided and are free, if other template is required which are downloadable are sometimes requires payment.

2.2 Product Functions

The software will come with UI to get your information and store it for next step, i.e to create a CV document, user can give information in some languages If they want to, save the build doc into pdf, docx files and able to print them in instant.

2.3 Operating Environment

We are creating a Desktop application so, we have planned Windows platform to be the first environment for this software. Every member is having Windows Operating System, so it will be easy to transfer data to other members. Later on, we will try to provide this software for other operating systems.

2.4 User Documentation

The user guide will be provided by the Feasibility-Study-Report.

2.5 General Constraints, Assumptions and Dependencies

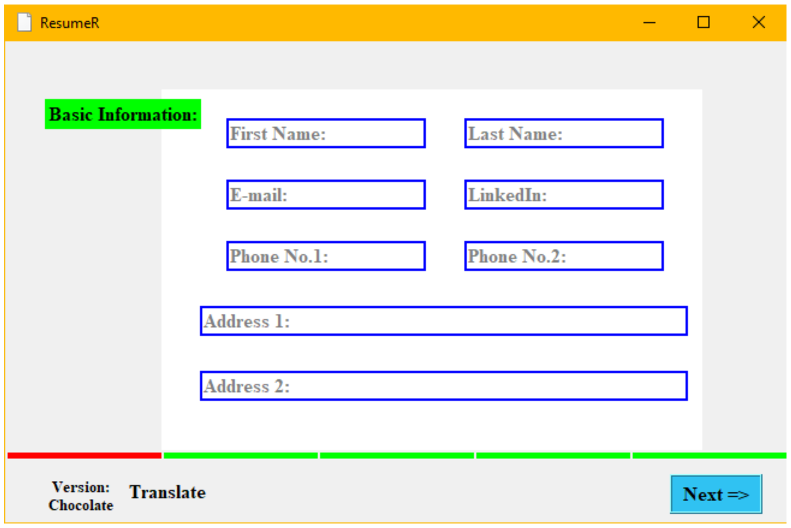
For changing language feature we use Google translate APIs i.e. **googletrans** we are restricted by the language support which this module has. we are assuming that module will work efficiently with this constraint. Similarly, for the other modules as well.

3. Specific Requirements

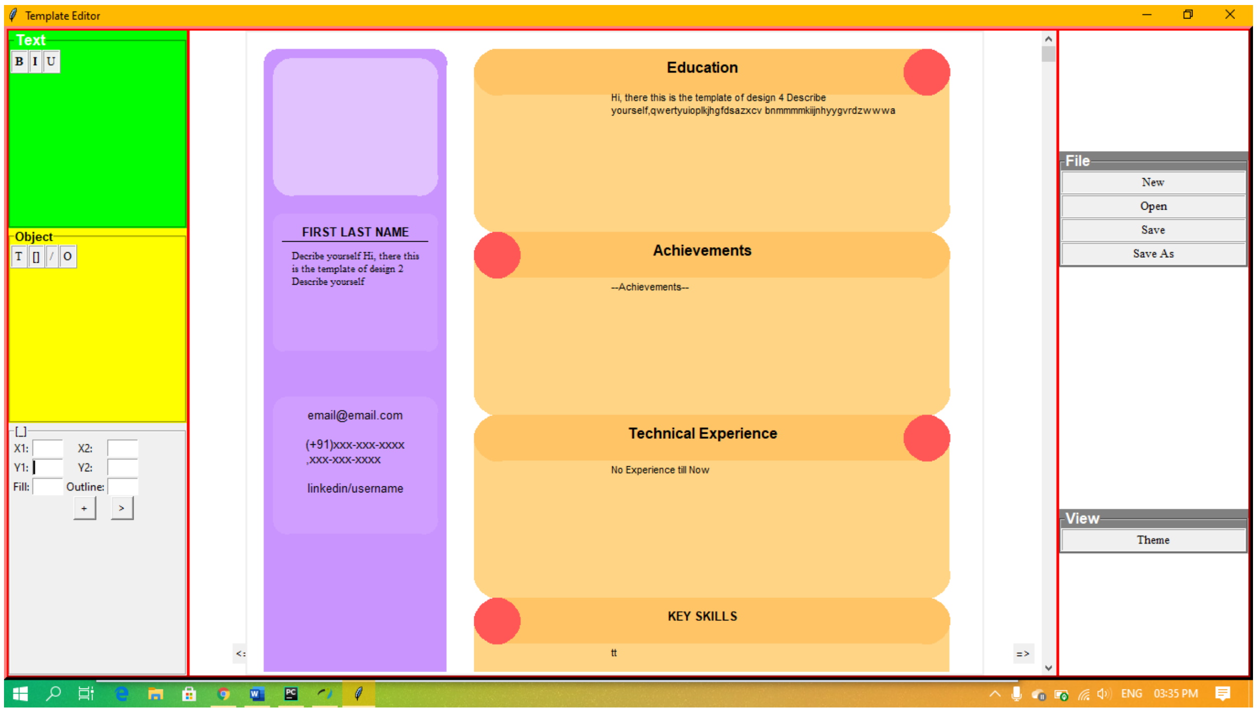
3.1 External Interface Requirements

3.1.1 User Interfaces

The first half of system is data collecting input boxes. The input in the boxes varies for every input box in the application like, name field => single-line input, phone no. => 10-digit number, address fields => text boxes having multi-line inputs, etc. The data is taken on instance when button is clicked. These input boxes are placed inside frames as shown in picture below.



The second half of the system is Editing UI, The window is divided into 3 sections: Panel 1, Panel 2, Panel 3. Panel 1 consists of options/objects/tools that help you create the templates on plane canvas. Panel 2 contains the main workspace consisting the canvas (1200 x 800)px with slider on Left end. Panel 3 contains other main options for non-functionality requirements. The Current UI is shown below:



Decribe yourself Hi, there this is the template of design 2 Describe yourself

Hi, there this is the template of design 4 Describe yourself,qwertyuioplkjhgfdsazxcv bnmmmmkiijnhyygvrdzwwwa

--Achievements--

No Experience till Now

tt

3.1.2 Hardware Interfaces
The user is required to have a PC for using this software, in addition, the user can also use a printer for printing purposes.

3.1.3 Software Interfaces
We are building this product using Python language and it's modules(libraries) like tkinter, python-docx, mail-merge, etc. In detail-

-	Python Language	3.9
-	Tkinter	8.4
-	Python-docx	0.8.10
-	Sqlite3 Database Interface	2.0
-	Pyinstaller	4.0
-	googletrans	0.3.11

We are using Python in Windows 10 Operating System. The standalone product is built with tkinter module in Pycharm Editor. **Sqlite3** module provides the interface to connect to database for storing and retrieving your entered data.

Docx in **python-docx** module provide interface for creating or updating Ms Word .docx file.

3.1.4 Communications Interfaces
--None—

3.2 Functional Requirements

3.2.1 Form like UI
The form ui consists of various input taking Entry boxes taking data in different type of formats. Some of that are required fill field like first name, contact number, etc..

Purpose: To take correct information form the user to store in the database and to show on CV.

Inputs: Most of the fields take string input, some fields like phone number, scores, etc. required to be filled with numeric inputs only.

Output: Information is stored in a database table.

3.2.2 Supports different languages
This features will help to convert the English filled data to convert to specified language data.

Purpose: There may be possibility that a company takes individuals whose documents are written in specific language or a case where the user want to convert the language to be changed to other language.

Input: Select the Source-Destination language, by default Source language is English.

Output: Specified language

3.2.3 Template builder
This feature plays a main role with building a document Template from scratch, or outputting the whole document. It comes with various options like creating text objects, creating shapes, etc.

Purpose: Extending above part.

Input: for a object, its attributes as text/numeric/colour fields. To create the objects command given as button click.

Output: Showing various styles on the canvas.

3.2.4 Document Converter

To output the created document as **.pdf/.docx** files by using modules/library files of FPDF/python-docx.

Purpose: to directly help you to get various types of files of different types.

Input: file name and Button Click

Output: Saves the file as the given file name.

3.3 Design and Implementation Constraints

Using Python + Tkinter limits the developer to create static shaped applications i.e. it has same appearance when it is full screened or the original size. The developer is not able to assign icons/ pictures in developing the application. The syntax of sqlite3 database should be proper.

3.4 Other Requirements

--NONE—

Identifying the Requirements from Problem Statements

Introduction

Requirements identification is the first step of any software development project. Until the requirements of a client have been clearly identified, and verified, no other task (design, coding, testing) could begin. Usually business analysts having domain knowledge on the subject matter discuss with clients and decide what features are to be implemented.

In this experiment we will learn how to identify functional and non-functional requirements from a given problem statement. Functional and non-functional requirements are the primary components of a Software Requirements Specification.

Objectives

After completing this experiment you will be able to:

- Identify ambiguities, inconsistencies and incompleteness from a requirements specification
- Identify and state functional requirements
- Identify and state non-functional requirements

Requirements

Sommerville defines "requirement" as a specification of what should be implemented. Requirements specify how the target system should behave. It specifies what to do, but not how to do. Requirements engineering refers to the process of understanding what a customer expects from the system to be developed, and to document them in a standard and easily readable and understandable format. This documentation will serve as reference for the subsequent design, implementation and verification of the system.

It is necessary and important that before we start planning, design and implementation of the software system for our client, we are clear about it's requirements. If we don't have a clear vision of what is to be developed and what all features are expected, there would be serious problems, and customer dissatisfaction as well.

Characteristics of Requirements

Requirements gathered for any new system to be developed should exhibit the following three properties:

Unambiguity: There should not be any ambiguity what a system to be developed should do. For example, consider you are developing a web application for your client. The client requires that enough number of people should be able to access the application simultaneously. What's the "enough number of people"? That could mean 10 to you, but, perhaps, 100 to the client. There's an ambiguity.

Consistency: To illustrate this, consider the automation of a nuclear plant. Suppose one of the clients say that it the radiation level inside the plant exceeds R1, all reactors should be shut down. However, another person from the client side suggests that the threshold radiation level should be R2. Thus, there is an inconsistency between the two end users regarding what they consider as threshold level of radiation.

Completeness: A particular requirement for a system should specify what the system should do and also what it should not. For example, consider a software to be developed for ATM. If a customer enters an amount greater than the maximum permissible withdrawal amount, the ATM should display an error message, and it should not dispense any cash.

Categorization of Requirements

Based on the target audience or subject matter, requirements can be classified into different types, as stated below:

User requirements: They are written in natural language so that both customers can verify their requirements have been correctly identified

System requirements: They are written involving technical terms and/or specifications, and are meant for the development or testing teams

Requirements can be classified into two groups based on what they describe:

Functional requirements (FRs): These describe the functionality of a system -- how a system should react to a particular set of inputs and what should be the corresponding output.

Non-functional requirements (NFRs): They are not directly related what functionalities are expected from the system. However, NFRs could typically define how the system should behave under certain situations. For example, a NFR could say that the system should work with 128MB RAM. Under such condition, a NFR could be more critical than a FR.

Non-functional requirements could be further classified into different types like:

Product requirements: For example, a specification that the web application should use only plain HTML, and no frames

Performance requirements: For example, the system should remain available 24x7

Organizational requirements: The development process should comply to SEI CMM level 4

Functional Requirements

Identifying Functional Requirements

Given a problem statement, the functional requirements could be identified by focusing on the following points:

Identify the high level functional requirements simply from the conceptual understanding of the problem. For example, a Library Management System, apart from anything else, should be able to issue and return books.

Identify the cases where an end user gets some meaningful work done by using the system. For example, in a digital library a user might use the "Search Book" functionality to obtain information about the books of his interest.

If we consider the system as a black box, there would be some inputs to it, and some output in return. This black box defines the functionalities of the system. For example, to search for a book, user gives title of the book as input and get the book details and location as the output.

Any high level requirement identified could have different sub-requirements. For example, "Issue Book" module could behave differently for different class of users, or for a particular user who has issued the book thrice consecutively.

Preparing Software Requirements Specifications

Once all possible FRs and non-FRs have been identified, which are complete, consistent, and non-ambiguous, the Software Requirements Specification (SRS) is to be prepared. IEEE provides a template [iv], also available here, which could be used for this purpose. The SRS is prepared by the service provider, and verified by its client. This document serves as a legal agreement between the client and the service provider. Once the concerned system has been developed and deployed, and a proposed feature was not found to be present in the system, the client can point this out from the SRS. Also, if after delivery, the client says a new feature is required, which was not mentioned in the SRS, the service provider can again point to the SRS. The scope of the current experiment, however, doesn't cover writing a SRS.

Statement :- New users can register to the system through an online process. By registering a user agrees to abide by different pre-defined terms and conditions as specified by the system. Any registered user can access the different features of the system authorized to him / her, after he authenticates himself through the login screen. An authenticated user can put items in the system for auction. Authenticated users users can place bid for an item. Once the auction is over, the item will be sold to the user placing the maximum bid. Payments are to be made by third party payment services, which, of course, is guaranteed to be secure. The user selling the item will be responsible for it's shipping. If the seller thinks he's getting a good price, he can, however, sell the item at any point of time to the maximum bidder available.

Learning Objectives:

Learn about the three different aspects that have to be taken care of while writing requirements specification

Limitations:

1. There's no specification when an auction gets over -- **Ambiguitiy**
2. An item is said to be sold to the max bidder after auction is over; it can also be sold before the auction is over – **Inconsistency**
3. **Incompleteness -**
    - No mention of how a new user registers

- No mention of any dispute regarding the sold product
- No mention of what kind of products could be put on auction

Bibliography

- Requirements Engineering: A Good Practice Guide, Ian Sommerville, Pete Sawyer, Wiley India Pvt Ltd, 2009
- Fundamentals of Software Engineering, Rajib Mall, Prentice-Hall of India, 3rd Edition, 2009

Webliography

- Lecture on "System Analysis and Design", NPTEL
- When Telepathy Won't Do: Requirements Engineering Key Practices
- Requirements Analysis: Process of requirements gathering and requirement definition
- IEEE Recommended Practice for Software Requirements Specifications
- Requirements Trace-ability and Use Cases

Aim: Introduction to Papyrus & Eclipse and Step by Step guide to install it on windows 10.

Theory:

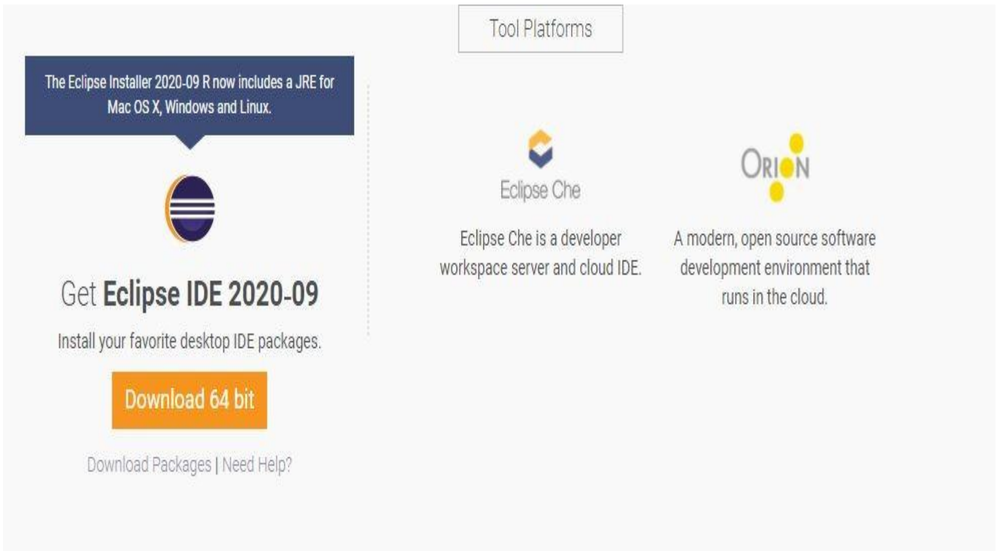
What is Eclipse?

Eclipse is an integrated development environment (IDE) for Java and other programming languages like C, C++, PHP, and Ruby etc. Development environment provided by Eclipse includes the Eclipse Java development tools (JDT) for Java, Eclipse CDT for C/C++, and Eclipse PDT for PHP, among others.

Procedure For Downloading Eclipse:

We can download eclipse from http://www.eclipse.org/downloads/

We can download any version of it as it is free.



Tool Platforms

The Eclipse Installer 2020-09 R now includes a JRE for Mac OS X, Windows and Linux.

Get **Eclipse IDE 2020-09**

Install your favorite desktop IDE packages.

Download 64 bit

Download Packages | Need Help?

Eclipse Che

Eclipse Che is a developer workspace server and cloud IDE.

ORION

A modern, open source software development environment that runs in the cloud.

The capabilities of each packaging of eclipse are different. Java developers typically use Eclipse Classic or Eclipse IDE for developing Java applications.

The drop down box in the right corner of the download page allows you to set the operating system on which eclipse is to be installed. You can choose between Windows, Linux and Mac. Eclipse is packaged as a zip file.

Procedure for Installing Eclipse:

To install on windows, you need a tool that can extract the contents of a zip file. For example you can use –

- Win RAR
- PeaZip
- IZArc

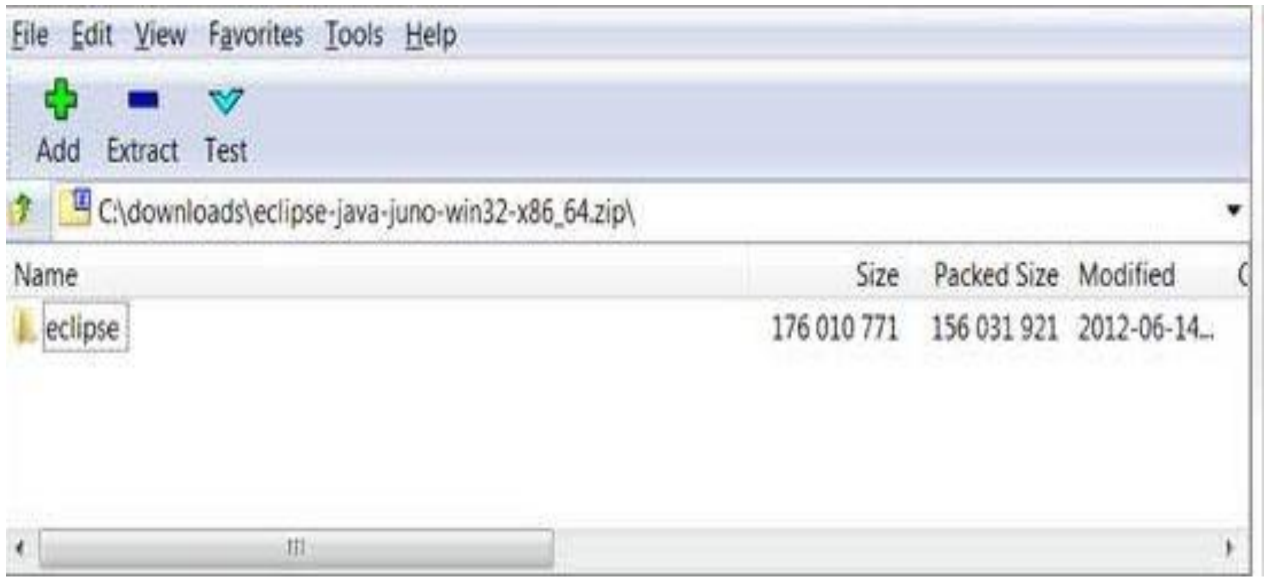
Using any one of these tools, extract the contents of the eclipse zip file to any folder of your choice.

How To Launch Eclipse:

On the windows platform, if you extracted the contents of the zip file to c:\, then you can start eclipse by using c:\eclipse\eclipse.exe

When eclipse starts up for the first time it prompts you for the location of the workspace folder. All your data will be stored in the workspace folder. You can accept the default or choose a new location.

If you select "Use this as the default and do not ask again", this dialog box will not come up again. You can change this preference using the Workspaces Preference Page. See the Preference tutorial page for more details.

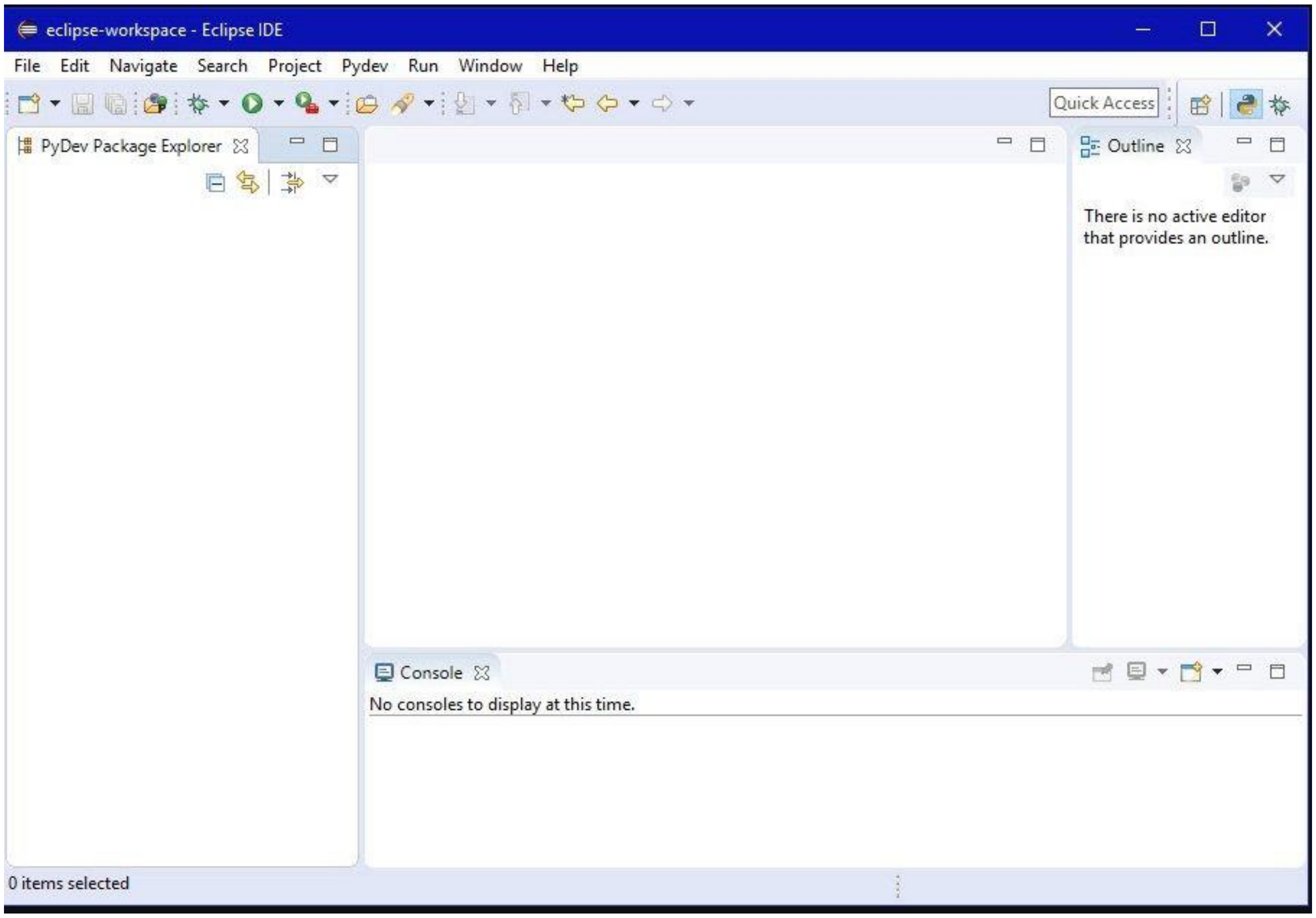


C:\downloads\eclipse-java-juno-win32-x86_64.zip\

Name	Size	Packed Size	Modified
eclipse	176 010 771	156 031 921	2012-06-14...

Parts of an Eclipse Window

- Editors (all appear in one editor area)
- Menu Bar
- Toolbar


eclipse-workspace - Eclipse IDE
File Edit Navigate Search Project Pydev Run Window Help
Quick Access
PyDev Package Explorer
Outline
Console

There is no active editor that provides an outline.

No consoles to display at this time.

0 items selected

Papyrus:

Papyrus is built on the extensible Eclipse framework and is an implementation of the OMG (Object Management Group) specification Unified Modeling Language (UML) version 2.4. 1. Papyrus is a comprehensive UML modeling environment, where many diagrams can be used to view different aspects of a system.

How To Install:

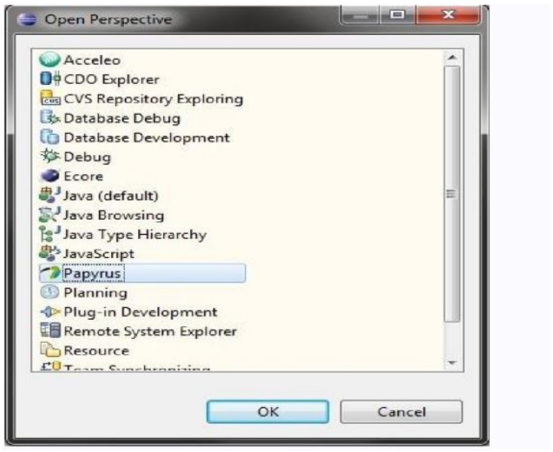
If you do not have Eclipse installed, you may want to

https://www.eclipse.org/papyrus/?P=114&L=EN&ITEMID=16#2the stand-alone version of Papyrus (102 MB.)

If you already have Eclipse installed, you are only required to add(http://www.papyrusuml.org/scripts/home/publigen/content/templates/show.asp?P=114&L=E N&ITEMID=16#3) Papyrus to your configuration.

Initial Setup

To begin using Papyrus, launch Eclipse (or stand-alone Papyrus) and navigate to Window -> Open Perspective -> Other, select Papyrus and press OK.



To create a new Project, navigate to File -> New -> Papyrus Project. Enter your project name and, if desired, change the default location. Advance to the next screen, check the UML box, and press Finish.

Hence, we are ready to use Papyrus.

Unified Modelling Language (UML)

The **Unified Modeling Language** (**UML**) is a general-purpose, developmental, modeling language in the field of software engineering that is intended to provide a standard way to visualize the design of a system. The creation of UML was originally motivated by the desire to standardize the disparate notational systems and approaches to software design.

UML has been evolving since the second half of the 1990s and has its roots in the object-oriented programming methods developed in the late 1980s and early 1990s.It is originally based on the notations of the Booch method, the object-modeling technique (OMT) and object-oriented software engineering (OOSE), which it has integrated into a single language.

UML 2.0 major revision replaced version 1.5 in 2005, which was developed with an enlarged consortium to improve the language further to reflect new experience on usage of its features. UML offers a way to visualize a system's architectural blueprints in a diagram, including elements such as:

- any activities (jobs);
- individual components of the system
- and how they can interact with other software components;
- how the system will run;
- how entities interact with others (components and interfaces);
- external user interface.

Although originally intended for object-oriented design documentation, UML has been extended to a larger set of design documentation, and been found useful in many contexts.

The following lists of UML diagrams and functionality summaries enable understanding of UML applications in real-world examples.

Structure diagrams and their applications

Structuring diagrams show a view of a system that shows the structure of the objects, including their classifiers, relationships, attributes and operations:

- Class diagram
- Component diagram
- Composite structure diagram
- Deployment diagram
- Object diagram
- Package diagram
- Profile diagram

Behaviour diagrams and their applications

Behaviour diagrams are used to illustrate the behavior of a system, they are used extensively to describe the functionality of software systems. Some Behaviour diagrams are:

- Activity diagram
- State machine diagram
- Use Case Diagram

Interaction diagrams and their applications

Interaction diagrams are subset of behaviour diagrams and emphasize the flow of control and data among the things in the system being modelled:

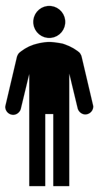
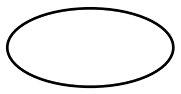
- Communication diagram
- Interaction overview diagram
- Sequence diagram
- Timing diagram

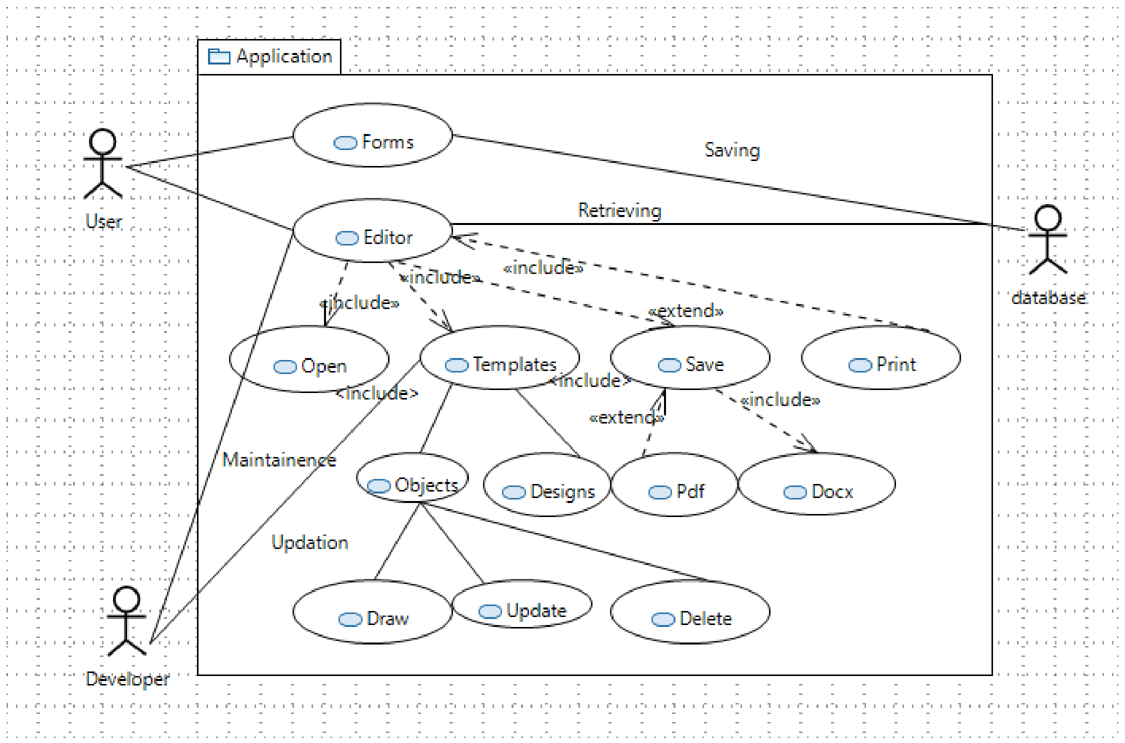
Some of these Diagrams are drawn for representing our project:

Use Case Diagram:

Use cases are a set of actions, services, and functions that the system needs to perform. The purpose of a use case diagram in UML is to demonstrate the different ways that a user might interact with a system.

Notations:

Actor: The users that interact with a system. An actor can be a person, an organization, or an outside system that interacts with your application or system. They must be external objects that produce or consume data. There are 2 types of actors **Primary Actors** – who initiate a use case. E.g. – a person who withdraw money from the ATM **Secondary Actors** – who has involvement in a use case	
Use Case: Horizontally shaped ovals that represent system function (process - automated or manual) Each Actor must be linked to a use case, while some use cases may not be linked to actors.	
Communication Link: A line between actors and use cases, it represents which actor is interacting with which use case.	
Boundary of system: Draw your system's boundaries using a rectangle that contains use cases. Place actors outside the system's boundaries.	
Extends: It shows optional behaviour.	<<extends>>
Include: It shows mandatory behaviour.	<<include>>

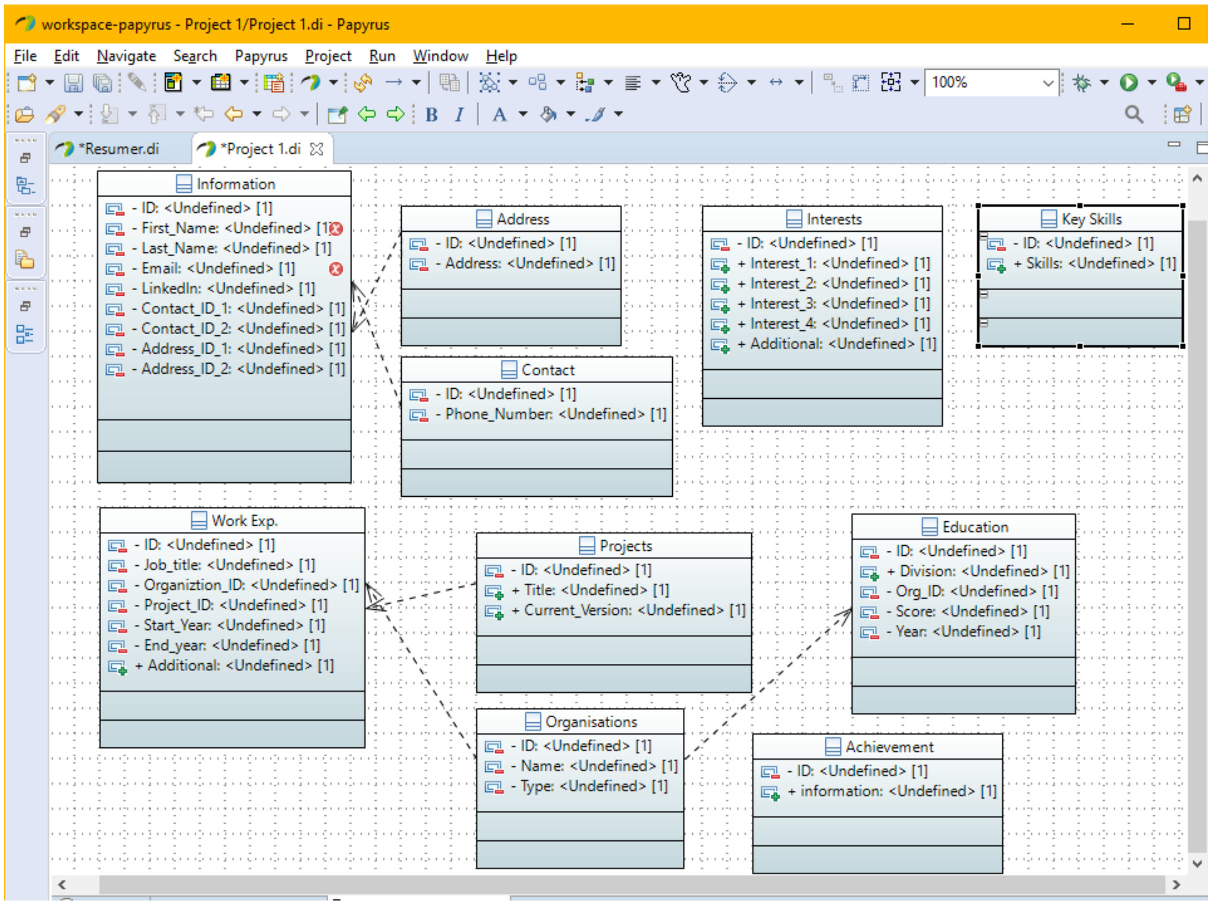


Use Case Diagram

UML Class Diagram:

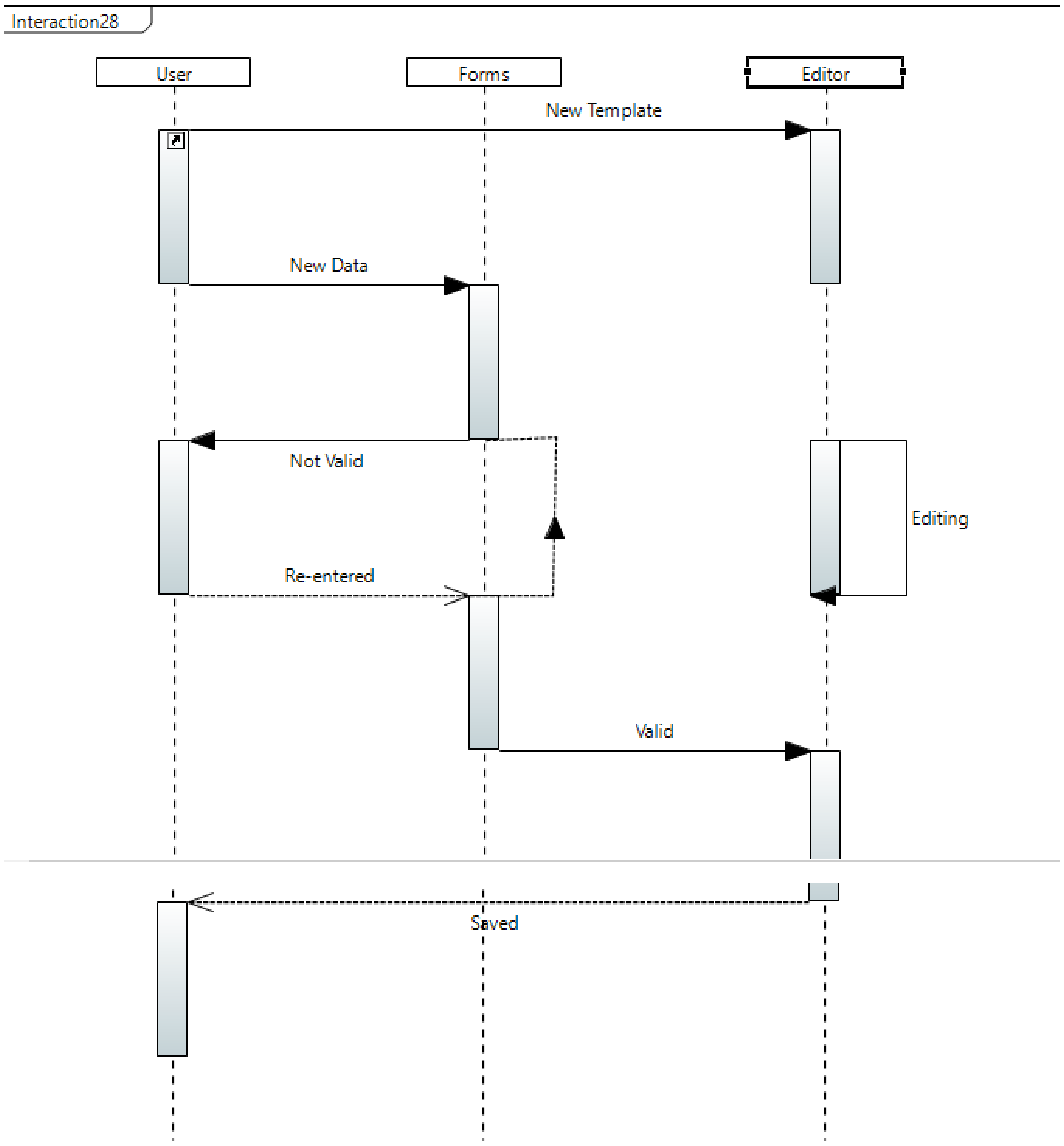
Class Diagram gives an overview of a software system by displaying classes, attributes, operations, and their relationship. Class Diagram gives the static view of an application. A class diagram describes the types of objects in the system and the different types of relationships that exist among them. This modelling method can run with almost all Object-Oriented Methods. A class can refer to another class. A class can have its objects or may inherit from other classes.

The Database/Model Diagram:


workspace-papyrus - Project 1/Project 1.di - Papyrus
File Edit Navigate Search Papyrus Project Run Window Help
100%
*Resumer.di
*Project 1.di
Information
- ID: <Undefined> [1]
- First_Name: <Undefined> [1]
- Last_Name: <Undefined> [1]
- Email: <Undefined> [1]
- LinkedIn: <Undefined> [1]
- Contact_ID_1: <Undefined> [1]
- Contact_ID_2: <Undefined> [1]
- Address_ID_1: <Undefined> [1]
- Address_ID_2: <Undefined> [1]
Address
- ID: <Undefined> [1]
- Address: <Undefined> [1]
Interests
- ID: <Undefined> [1]
+ Interest_1: <Undefined> [1]
+ Interest_2: <Undefined> [1]
+ Interest_3: <Undefined> [1]
+ Interest_4: <Undefined> [1]
+ Additional: <Undefined> [1]
Key Skills
- ID: <Undefined> [1]
+ Skills: <Undefined> [1]
Contact
- ID: <Undefined> [1]
- Phone_Number: <Undefined> [1]
Work Exp.
- ID: <Undefined> [1]
- Job_title: <Undefined> [1]
- Organiztion_ID: <Undefined> [1]
- Project_ID: <Undefined> [1]
- Start_Year: <Undefined> [1]
- End_year: <Undefined> [1]
+ Additional: <Undefined> [1]
Projects
- ID: <Undefined> [1]
+ Title: <Undefined> [1]
+ Current_Version: <Undefined> [1]
Education
- ID: <Undefined> [1]
+ Division: <Undefined> [1]
- Org_ID: <Undefined> [1]
- Score: <Undefined> [1]
- Year: <Undefined> [1]
Organisations
- ID: <Undefined> [1]
- Name: <Undefined> [1]
- Type: <Undefined> [1]
Achievement
- ID: <Undefined> [1]
+ information: <Undefined> [1]

UML Case Diagram

UML Sequence Diagram:

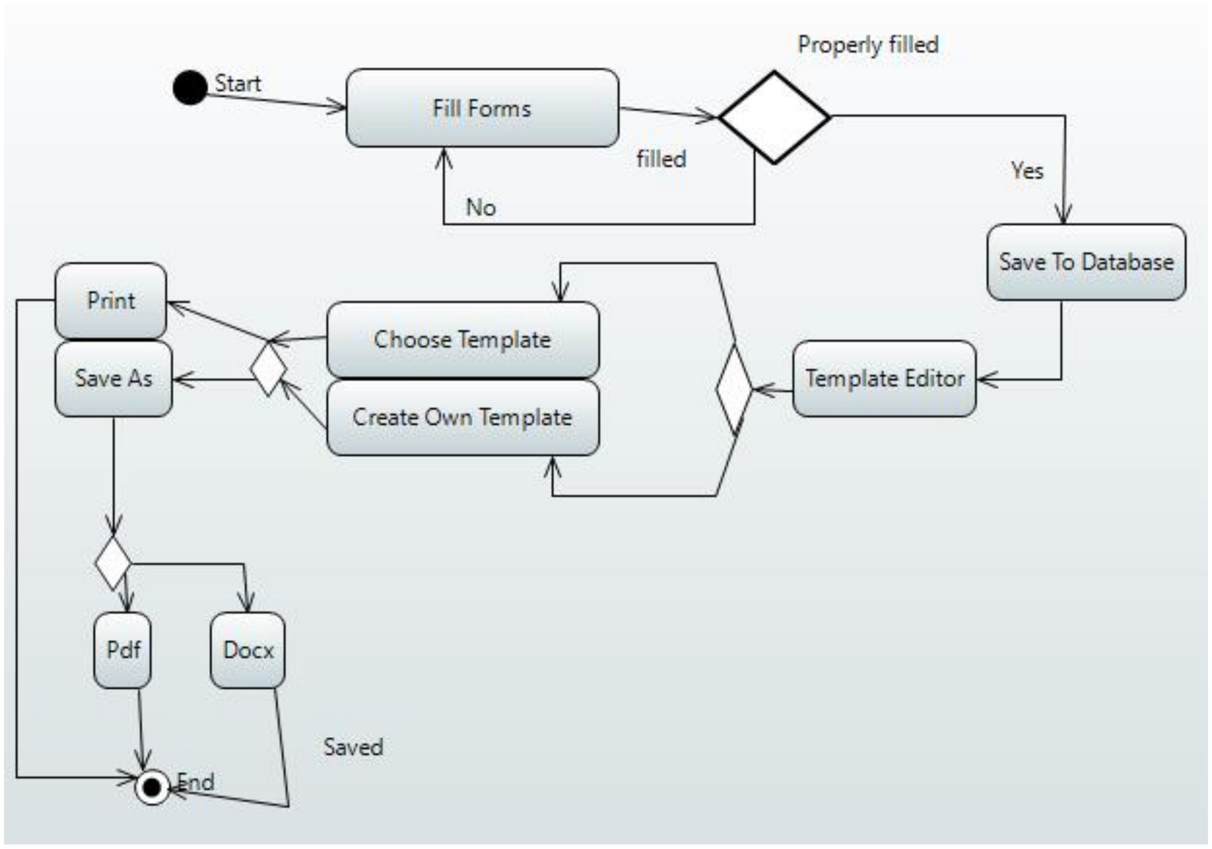


Sequence Diagram

UML Activity Diagram:

Activity diagram describes the dynamic aspects of the system. It is the flowchart to represent the flow of various connected activities.

The Activity of our project is as shown.



UML Activity Diagram

Estimation of Project Metrics

Introduction

After gathering the entire requirements specific to software project usually we need to think about different solution strategy for the project. Expert business analysts are analyzing their benefits and as well as their shortcomings by means of cost, time and resources require to develop it.

In this experiment, we will learn how to estimate cost, effort and duration for a software project, and then select one solution approach which will be found suitable to fulfill the organizational goal.

Objectives

- Categorize projects using COCOMO, and estimate effort and development time required for a project
- Estimate the program complexity and effort required to recreate it using Halstead's metrics

Project Estimation Techniques

A software project is not just about writing a few hundred lines of source code to achieve a particular objective. The scope of a software project is comparatively quite large, and such a project could take several years to complete. However, the phrase "quite large" could only give some (possibly vague) qualitative information. As in any other science and engineering discipline, one would be interested to measure how complex a project is. One of the major activities of the project planning phase, therefore, iss to estimate various project parameters in order to take proper decisions. Some important project parameters that are estimated include:

- **Project size:** What would be the size of the code written say, in number of lines, files, modules?
- **Cost:** How much would it cost to develop a software? A software may be just pieces of code, but one has to pay to the managers, developers, and other project personnel.
- **Duration:** How long would it be before the software is delivered to the clients?
- **Effort:** How much effort from the team members would be required to create the software?

In this experiment we will focus on two methods for estimating project metrics: COCOMO and Halstead's method.

COCOMO

COCOMO (Constructive Cost Model) was proposed by Boehm. According to him, there could be three categories of software projects: organic, semidetached, and embedded. The classification is done considering the characteristics of the software, the development team and environment. These product classes typically correspond to application, utility and system programs, respectively. Data processing programs could be considered as application programs. Compilers, linkers, are examples of utility programs. Operating systems, real-time system programs are examples of system programs. One could easily apprehend that it would take much more time and effort to develop an OS than an attendance management system.

The concept of organic, semidetached, and embedded systems are described below.

- **Organic:** A development project is said to be of organic type, if
    - The project deals with developing a well understood application
    - The development team is small
    - The team members have prior experience in working with similar types of projects
- **Semidetached:** A development project can be categorized as semidetached type, if
    - The team consists of some experienced as well as inexperienced staff
    - Team members may have some experience on the type of system to be developed
- **Embedded:** Embedded type of development project are those, which
    - Aims to develop a software strongly related to machine hardware
    - Team size is usually large

Boehm suggested that estimation of project parameters should be done through three stages: Basic COCOMO, Intermediate COCOMO, and Complete COCOMO.

Basic COCOMO Model

The basic COCOMO model helps to obtain a rough estimate of the project parameters. It estimates effort and time required for development in the following way:

$$\text{Effort} = a * (\text{KDSI})^b \text{ Person-Months}$$

$$\text{Tdev} = 2.5 * (\text{Effort})^c \text{ Months}$$

where

- KDSI is the estimated size of the software expressed in Kilo Delivered Source Instructions
- a, b, c are constants determined by the category of software project
- Effort denotes the total effort required for the software development, expressed in person months (PMs)
- Tdev denotes the estimated time required to develop the software (expressed in months)

The value of the constants a, b, c are given below:

Software project	*a*	*B*	*c*
Organic	2.4	1.05	0.38
Semi-detached	3.0	1.12	0.35
Embedded	3.6	1.20	0.32

Intermediate COCOMO Model

The basic COCOMO model considers that effort and development time depends only on the size of the software. However, in real life there are many other project parameters that influence the development process. The intermediate COCOMO take those other factors into consideration by defining a set of 15 cost drivers (multipliers) as shown in the table below. Thus, any project that makes use of modern programming practices would have lower estimates in terms of effort and cost. Each of the 15 such attributes can be rated on a six-point scale ranging from "very low" to "extra high" in their relative order of importance. Each attribute has an effort multiplier fixed as per the rating. The product of effort multipliers of all the 15 attributes gives the **Effort Adjustment Factor (EAF).**

Cost Drivers	**Very Low**	**Low**	**Nominal**	**High**	**Very High**	**Extra High**
Product attributes						
Required software reliability	0.75	0.88	1.00	1.15	1.40	
Size of application database		0.94	1.00	1.08	1.16	
Complexity of the product	0.70	0.85	1.00	1.15	1.30	1.65
Hardware attributes						
Run-time performance constraints			1.00	1.11	1.30	1.66
Memory constraints			1.00	1.06	1.21	1.56
Volatility of the virtual machine environment		0.87	1.00	1.15	1.30	
Required turnabout time		0.87	1.00	1.07	1.15	
Personnel attributes						
Analyst capability	1.46	1.19	1.00	0.86	0.71	
Applications experience	1.29	1.13	1.00	0.91	0.82	
Software engineer capability	1.42	1.17	1.00	0.86	0.70	
Virtual machine experience	1.21	1.10	1.00	0.90		

Cost Drivers	Very Low	Low	Nominal	High	Very High	Extra High
Programming language experience	1.14	1.07	1.00	0.95		
Project attributes						
Application of software engineering methods	1.24	1.10	1.00	0.91	0.82	
Use of software tools	1.24	1.10	1.00	0.91	0.83	
Required development schedule	1.23	1.08	1.00	1.04	1.10	

EAF is used to refine the estimates obtained by basic COCOMO as follows:

$$\text{Effort}_{|\text{corrected}} = \text{Effort} * \text{EAF}$$

$$\text{Tdev}_{|\text{corrected}} = 2.5 * (\text{Effort}_{|\text{ corrected}})^{c}$$

Complete COCOMO Model

Both the basic and intermediate COCOMO models consider a software to be a single homogeneous entity -- an assumption, which is rarely true. In fact, many real life applications are made up of several smaller sub-systems. (One might not even develop all the sub-systems -- just use the available services). The complete COCOMO model takes these factors into account to provide a far more accurate estimate of project metrics.

To illustrate this, consider a very popular distributed application: the ticket booking system of the Indian Railways. There are computerized ticket counters in most of the railway stations of our country. Tickets can be booked / cancelled from any such counter. Reservations for future tickets, cancellation of reserved tickets could also be performed. On a high level, the ticket booking system has three main components:

- Database
- Graphical User Interface (GUI)
- Networking facilities

Among these, development of the GUI is considered as an organic project type; the database module could be considered as a semi-detached software. The networking module can be considered as an embedded software. To obtain a realistic cost, one should estimate the costs for each component separately, and then add it up.

Advantages of COCOMO

COCOMO is a simple model, and should help one to understand the concept of project metrics estimation.

Drawbacks of COCOMO

COCOMO uses KDSI, which is not a proper measure of a program's size. Indeed, estimating the size of a software is a difficult task, and any slight miscalculation could cause a large deviation in subsequent project estimates. Moreover, COCOMO was proposed in 1981 keeping the waterfall model of project life cycle in mind. It fails to address other popular approaches like prototype, incremental, spiral, agile models. Moreover, in present day a software project may not necessarily consist of coding of every bit of functionality. Rather, existing software components are often used and glued together towards the development of a new software. COCOMO is not suitable in such cases.

COCOMO II was proposed later in 2000 to many of address these issues.

Halstead's Complexity Metrics

Halstead took a linguistic approach to determine the complexity of a program. According to him, a computer program consists of a collection of different operands and operators. The definition of operands and operators could, however, vary from one person to another and one programming language to other. Operands are usually the implementation variables or constants -- something upon which an operation could be performed. Operators are those symbols that affects the value of operands. Halstead's metrics are computed based on the operators and operands used in a computer program. Any given program has the following four parameters:

- n1: Number of unique operators used in the program
- n2: Number of unique operands used in the program
- N1: Total number of operators used in the program
- N2: Total number of operands used in the program

Using the above parameters one compute the following metrics:

- **Program Length:** N = N1 + N2
- **Program Vocabulary:** n = n1 + n2
- **Volume:** V = N * lg n
- **Difficulty:** D = (n1 * N2) / (2 * n2)
- **Effort:** E = D * V
- **Time to Implement**: T = E / 18 (in seconds)

The program volume V is the minimum number of bits needed to encode the program. It represents the size of the program while taking into account the programming language.
The difficulty metric indicates how difficult a program is to write or understand.

Effort denotes the "mental effort" required to develop the software, or to recreate the same in another programming language.

Statement: Considering your immense expertise in software development, The Absolute Beginners Inc. has recently allotted you a mega project. The goal of the project is to create a database of all Hindi films released since 2000. The software would allow one to generate a list of top ten hit films, top ten flop films, best comedy films, and so on. Using your prior experience you have decided the approximate sizes of each module of the software as follows:

- Data entry (0.9 KDSI)
- Data update (0.7 KDSI)
- Query (0.9 KDSI)
- Report generation and display (2 KDSI)

Also take into consideration the following cost drivers with their ratings:

- Storage constraints (Low)
- Experience in developing similar software (High)
- Programming capabilities of the developers (High)
- Application of software engineering methods (High)
- Use of software tools (High)

Applying intermediate COCOMO estimate the minimum size of the team you would require to develop this system. Assuming that your client would pay Rs. 50,000 per month of development, how much would be the likely billing?

Learning Objectives:

- Identify type of a project as per COCOMO
- Prepare an estimate of required effort and cost

Calculation:

Product size: 0.9 + 0.7 + 0.9 + 2 = 4.5 KLOC = KDSI

EAF = 0.91 x 0.91 x 0.95 x 0.91 => 0.7 (taken from considerations)

Hence, the table can be formed:

Project Type	**a**	**b**	**c**
Organic	2.4	1.05	0.38
Project size (in KDSI)	4.5		
Effort (in PM)	11.64		
T_{dev} (in month)	6.35		
Effort Adjustment Factor (EAF)	0.7		

Project Type	**a**	**b**	**c**
$\text{Effort}\|_{\text{corrected}}$ (in PM)	8.15		
$T_{dev}\|_{\text{corrected}}$ (in month)	5.55		
No. of developers	2		