A Case study based Software Engineering Education using Open Source Tools

Sowmya B J
Dept. of CSE
M. S. Ramaiah Institute of Technology
sowmyabj@msrit.edu

Srinidhi Hiriyannaiah
Dept. of CSE
M.S. Ramaiah Institute of Technology
srinidhih@msrit.edu

K.G. Srinivasa
Dept. of CSE
M.S. Ramaiah Institute of Technology
kgsrinivas@msrit.edu

ABSTRACT
Software engineering is a course for undergraduate computer science students that comprises of principles of engineering in a software development. In this course, students learn about typical phases of software that involves requirement analysis, planning and scheduling, design and coding, testing, deployment and management on different case studies. In this paper, we investigated a practical approach for learning software engineering through open source tools for different phases of the software on different case studies they have chosen as their problem statement.

Keywords
Software Engineering, Waterfall model, Open source tools for Software Engineering.

1. INTRODUCTION
Software Engineering describes about various phases involved in a software project such as requirements and analysis, design and coding, testing, deployment and maintenance. In this paper we discuss the lab process that was implemented to provide a conducive and a near software firm environment for students to provide a better understanding of Software engineering principles, different phases of the software and finally the principles of management through Open source tools. The paper is further organized as follows. Section 2 discusses about a brief introduction to the process model and Case Studies categories followed for the projects in the lab. Section 3 describes about course overview, tasks carried out in each week and the tools used, section 4 discusses about survey conducted for the curriculum and its results based on the course outcomes of the introduction of a practical way to teach software engineering with tool.

2. PROCESS AND PHASES
Software engineering consists of process models such as waterfall model, incremental process, prototyping model, spiral model, scrum model and several other models [1] [2]. Out of these models, waterfall process model is one of the basic models that is used for developing a typical software in a industry [1]. It consists of phases requirement analysis, planning and scheduling, design, development, testing, deployment and maintenance.

In requirement analysis phase, the requirements are gathered for the project based on the features and goals listed down for a project. It may include functional requirements such as authentication of a user login and non-functional requirements such as reliability, performance. During planning phase the overall estimate or listing the various tasks to be carried out carries out the schedule of the project and resources are assigned to the tasks identified. In the system design phase, various modules are identified that defines the features and requirements identified in the phase 1. The modules identified are developed using suitable implementation language during development phase and tested with appropriate test cases either manually or automated during testing. There are some disadvantages with waterfall model compared to other software processes such as early frozen requirements, no feedback from the user and so on as discussed in [1][2].

Compared to other models we have followed a waterfall model approach with some modifications in the phase of requirements analysis and the feedback. During requirement analysis phase a SMART matrix approach was followed to frame the objectives and goals of a software project. For a feedback analysis in each phase a weekly status reports were designed that are discussed in the upcoming sections.

3. COURSE OVERVIEW
The course of Software Engineering is conducted for undergraduate students for Computer Science and Engineering as per ACM guidelines for Software Engineering education [7] in the third year of their curriculum having 4 credits with lectures and a practical lab. In the lab a team of 3 -4 students were formed where each one will be playing a different role in each phase. In this section we discuss the approach and the phases that we followed for our practical approach to teach Software Engineering using Open source tools. The different Case studies based on the problem statements chosen by the students were identified as shown in the table 1.

Table 1. Case Studies

Type	Case Studies	Example
A	E-commerce applications	e-banking, Online shopping & Logistics
B	Social related applications	Smart city, e-Voting, Remote health monitoring
C	Management application	Cab reservation, Hotel management system, Just Dial application

In the first week, a **problem statement** was designed and given to the students for applying Software Engineering techniques to it.

This problem statement was then formulated into SMART (Specific, Measurable, Achievable, relevant and Time Bound) matrix that helps in coming up with Goals and Objectives of the problem to be solved. Based on these goals and objectives, features are listed down for the software to be developed for the defined problem statement.

Now based on the features, **requirements elicitation** is carried out to create the Software Requirement Specification (SRS) using the tool called OSRMT (Open Source Requirements Management Tool) [8]. It provides a GUI interface for specifying the requirements, add dependencies between them. Once the final requirements are ready, a report on the requirements can be exported using the tool. Based on the categories of projects as identified in table we have identified top requirements for each category as shown in the table 2.

Table 2. Top requirements for the Case Studies

Case Study Classification	Top Requirements
Type A	User Info database, User Accounts, Session Allotment, SMS and e-mail alert, Easy to use User interface
Type B	User Info database, User Accounts, Travel Guide, Emergency Services, Smart communication, Data Access manger, fast Disaster force, Smart Health care,
Type C	User Info database, User Accounts, Area and Time of booking, Queries ON

In the next phase, **planning** is carried out for the Case study using ProjLibre [9]. With the help of this tool, team of students identifies the different type of activities and their dependencies that need to be carried out during the project and their roles in each activity. At the end of this phase, a project plan or schedule generically called as Gantt chart is prepared.

Various cost drivers such as application experience; required reliability and so on drive a software project. In this phase, **effort estimation** is carried out for the project based on COCOMO model using tool available by University of Southern California [10]. The selection of cost drivers is estimated based on the features, requirements and planning schedule as carried out in the previous phases. At the end of this phase, effort required for the project is calculated in person months with number of resources required at each phase. The next phase of the project for the case study is **Design** where the features and requirements are now represented schematically using StarUML tool [11]. We instructed the students to design the following representations for the project.

- **Use case diagram**
  It represents the summary level scenario of all the features in a project. It consists of actors, preconditions involved in the scenario.
- **Class diagram**
  It represents various classes, its attributes and methods, association between the classes for the actors considered in the use cases.
- **Sequence diagram**
  It represents the dynamic behaviour of the system or software with respect to the objects of the classes and its functions.

At the end of this phase, the team will be able to identify the modules that can be implemented. The modules can be identified based on the class diagrams and the sequence diagrams.

A software project involves risks such as personnel shortfalls, wrong software function implementation and so on. The next phase involved **risk analysis and mitigation** using risk management toolkit [12]. A risk matrix was prepared with XLS sheet with common fields as shown in the table 3 with an example of risk Personnel shortfalls. Once the risks are listed down, the priority and the rank of the risks is maintained with mitigation steps to control it.

Table 3. Risk template

Risk no	Risk name	Probability of occurence	Severity	Risk mitigation
1	Personnel shortfalls	85%	H	Provide necessary training on the required implementation language (C, Java)

Implementation or Coding was carried out in the next phase. During coding, we also followed **unit testing** approach where the students carried out unit testing with JUnit framework [13]. **Cyclomatic complexity** of a module was calculated using the McCabe formula, $V(G)=e-n+p$ by representing unit tested modules as a graph G with n nodes and e edeges [14]. Depending on the range of the cyclomatic complexity, those modules were further broken down if needed. Finally, during **testing** phase, we followed manual testing approach where test cases were designed with a template as shown in the table. These test cases were run against the features designed during the first phase and test report was generated.

To keep track of their activities the team had to submit the **Weekly Status Report** as shown in the fig 1, which has the information about their role and tasks accomplished, major decision that were part of the plan and the milestones achieved. The activities carried out in each week are summarized in the table 4 and tools used for each phase and tasks carried out as described above are shown in the table 5.

Table 4. Activities in each week

Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9

Problem Statement, Objectives, and Goals	Features and Requirements	Project planning and scheduling	Effort estimation using COCOMO model	Software Design	Risk analysis and mitigation	Coding and Unit Testing	Cyclomatic complexity	Testing

Table 5. Tools used for phases

Tool	Phase	Tasks carried out
OSRMT	Requirements and Analysis	It provides a GUI for providing the requirements in a list with dependencies between them.
Project Libre	Planning and scheduling	A gantt chart is prepared using this tool, that gives the overall schedule and resource planning for the project
COCOMO (online)	Effort estimation	Using this tool, effort in terms of person months is estimated with staffing profile required for each phase as shown in the fig.
Star UML	Design	Different sequence and collaboration diagrams, use case diagrams are drawn based on the features and requirements enlisted using OSRMT.
Risk matrix chart	Risk analysis and mitigation	A risk matrix of different risks and their probabilities, severity and mitigation steps to control the risk is prepared
JUnit	Unit Testing	A unit test for each module is designed and executed.

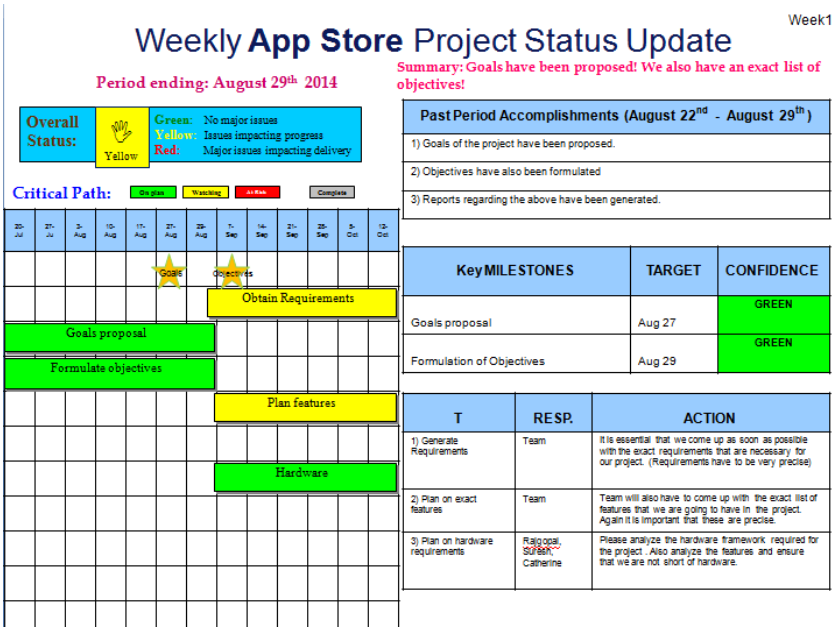


Fig 1.Weekly Status Reports

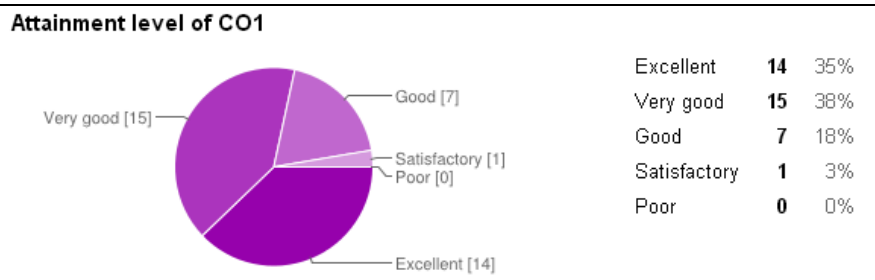
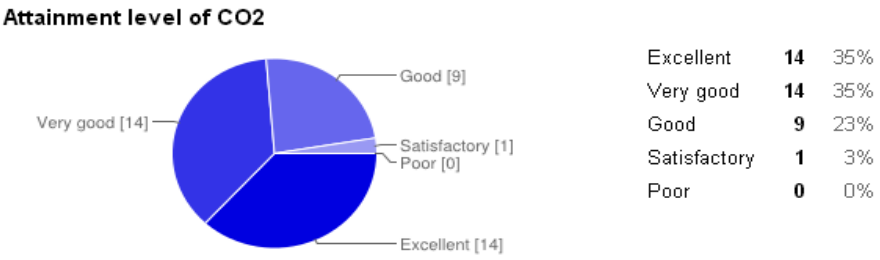
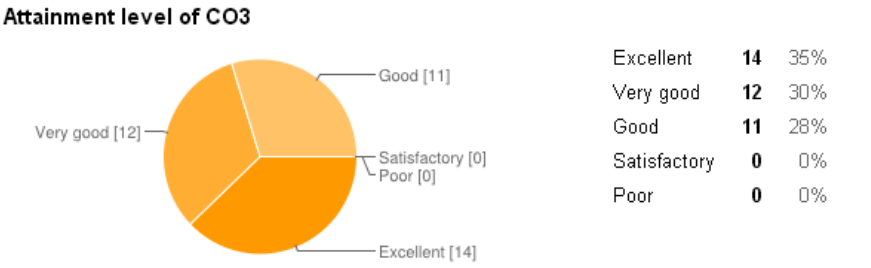
4. COURSE SURVEYS AND RESULTS

In this course of Software Engineering, we conducted surveys by designing course outcomes that can be satisfied by the students after learning this course. The main aim of these surveys is to improve the curriculum of Software Engineering course using the feedback from the students. The analysis and the results are as shown in the table 6 and table 7. Table 6 lists the course outcomes and their results. Table 7 lists the survey questions formed and its results. From these survey results and feedback from the students we infer the following results.

- A consistent SRS is prepared using the principles of requirement management.
- Ability to plan, schedule and estimate the effort for various activities in the project to accomplish its goals and objectives.
- Design and identify the modules of the system for problem using different UML diagrams.
- Implement the specification and testing effectively.

- Inculcate project management principles in a team and as an individual efficiently.

Table 6. Course outcomes and results

Sl.no	Course Outcome	Results of Survey
1	Identify a problem statement, trace the requirements and write unambiguous, correct and consistent SRS	
2	Prepare a project plan and estimate effort required for the project	
3	Identify, analyze and develop a risk management plan for the potential risks in the project	
4	To create a specification of a software artifact intended to accomplish goals	

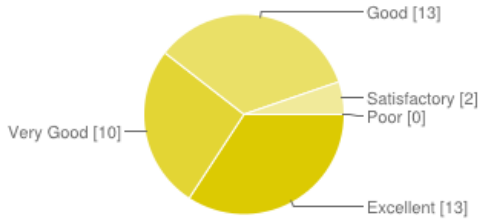
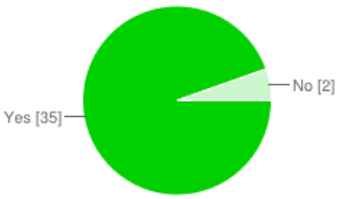
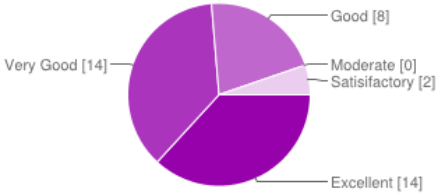
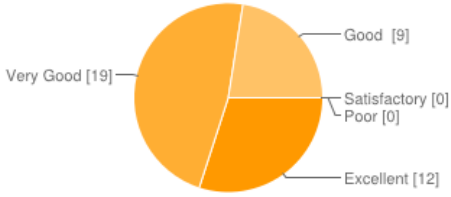
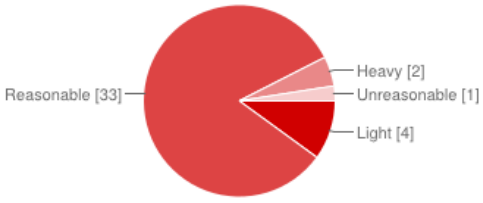
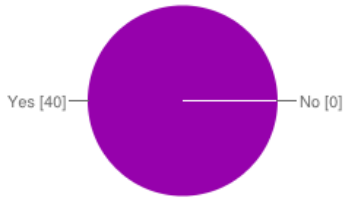
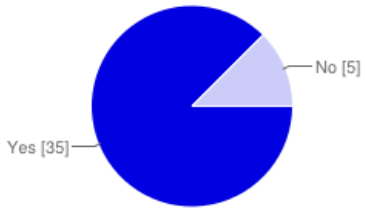
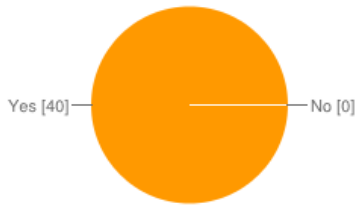
5	To perform testing of the code using Junit testing and test cases	**Attainment Level of CO5**  Excellent 13 33% Very Good 10 25% Good 13 33% Satisfactory 2 5% Poor 0 0%

Table 7. Survey and results

Sl.no	Survey questions	Results
1	Use of Tools and practical approach to understand the Software principles and development process	**Was the Lab Component helpful in understanding Software Engineering Principles**  Yes 35 88% No 2 5% **Attainment Levels of Tools explored in the Lab**  Excellent 14 35% Very Good 14 35% Good 8 20% Moderate 0 0% Satisifactory 2 5%
2	Quality of Course Content	**Quality of the course content**  Excellent 12 30% Very Good 19 48% Good 9 23% Satisfactory 0 0% Poor 0 0%
3	Course workload for the number of credits	**For the number of credits, the course workload was**  Light 4 10% Reasonable 33 83% Heavy 2 5% Unreasonable 1 3%

4	Organization of Lectures and lessons for the course	**Were the lectures clear/well organized and presented at a reasonable pace ?** Yes **40** 100% No **0** 0% 
5	Class sessions and lectures stimulate and helpful in understanding the course	**Did the lectures stimulate you intellectually?** Yes **35** 88% No **5** 13%  **Did the class sessions increase your understanding of the course?** Yes **40** 100% No **0** 0% 

5. COMPARISON WITH VIRTUAL LABS

The similar approach has been adopted in virtual lab developed for teaching software engineering at IIT Kharagpur [15], The main difference with our approach and virtual labs is we are using exhaustive open source tools. We have aligned our complete theory course structure mapping to the tasks they perform during SDLC. The team based learning and role based learning are also exploited during this course delivery.

While calculating the metrics in planning phase ie, **LOC**, **Effort, Schedule and COST** in virtual labs they have used the COCOMO model, using the cost drivers they estimated the metrics in virtual labs. In our practical approach, we have estimated effort using **COCOMO Model II** tool and we have estimated the schedule using Gnatt Chart and Pert Chart using the **ProjectLibre** tool. In the next phase, designing their problem statement using Use-case, Collaboration and Sequence diagram in virtual labs, but we have designed our problem statement using Use-case diagram, The class diagram to identify the classes and modules, identifying the aggregation and association. The dynamic behavior using sequence diagram using **StarUML** tool. During the process of identifying the cyclomatic complexity we have assigned them to implement the prime modules and to identify the **cyclomatic complexity** metrics for the implemented modules. Finally, in virtual labs they developed the test cases for the modules, but in our practical approach we have we generated the automated unit test cases using **Junit** Testframework and manual testing for the different modules.

6. CONCLUSION

Software engineering education plays a key role for the students to understand the principles of software development and its practices in software industry. The practical tool based approach for Software engineering followed helps the students to understand practically about the principles, phases and practices about software development in real time environment. The approach we have introduced in the course is in par with virtual labs introduced in [15]. The mapping of categories of the experiments followed in the virtual labs is same as the different tasks carried out in our approach as shown in the table 4.

Initially, the course was designed with only lectures based on the contents in the syllabus. Based on the students opinion to improve the course content through a practical approach, the course was modified with new syllabi and lab. Since, the project was carried out in a team, the students learned about principles of management by participating individually and collaborating with each other in the team. The proposed Case based approach for

Software Engineering made students aware of the course contents and the principles and techniques of the course.

7. REFERENCES

[1] Jalote, P. (2008). *A concise introduction to software engineering*. Springer.

[2] Pressman, R. S., & Jawadekar, W. S. (1987). Software engineering. *New York 1992*.

[3] Pankaj Jalote: A Concise Introduction to Software Engineering , Springer, 2008 *(Chapters: 1-4, 6-8)*

[4] David Gustafson: Software Engineering, Schaum's Outline Series, McGraw Hill, 2002 *(Chapters: 6)*

[5] Emilia Mendes, Nile Mosley: Web Engineering, Springer, 2006 *(Chapter: 1)*

[6] Roger S. Pressman: Software Engineering A Practitioner's Approach, 7th Edition, McGraw Hill, 2010

[7] ACM, IEEE. (2008). *Computer science curriculum 2008, An interim revision of CS 2001*. Retrieved March 14, 2012 from http://www.acm.org/education/curricula/ComputerScience2008.pdf

[8] http://sourceforge.net/projects/osrmt/

[9] http://www.projectlibre.org/

[10] http://csse.usc.edu/tools/COCOMOII.php

[11] http://staruml.io/

[12] http://www2.mitre.org/work/sepo/toolkits/risk/ToolsTechniques/RiskNav.html

[13] http://junit.org/

[14] McCabe, T. J. (1976). A complexity measure. *Software Engineering, IEEE Transactions on*, (4), 308-320.

[15] http://virtual-labs.ac.in/cse08/

Appendix

***Course Title*: Software Engineering**	***Course Code*: CS515**
Credits (L:T:P): 3:0:0	***Core/ Elective:* Core**
Type of Course: Lecture	**Total Contact Hours: 42 hrs**

Prerequisites: Nil

Course Objectives

Objectives of the course are to:

1. Provide an understanding of the principles of software engineering in a broader system context and the notions of software engineering process and management.
2. Identify the processes, techniques and deliverables that are associated with requirement engineering including system requirement and system modeling
3. Analyze the various steps involved in the design process and the different design approaches which include function-oriented design and object-oriented design
4. Identify the importance of testing in assuring the quality of software with an understanding of managing risks during the progress of the project
5. Appreciate the need for web engineering

The Software Problem & Processes: Cost, Schedule & Quality, Scale & Change, Software Processes: Process & Project, Component Software Processes, Software Development Process Models, Project Management Process

Requirements Analysis & Project Planning: Requirements Analysis & Specification: Value of a Good SRS, Requirements Process, Requirements Specification, Functional Specification with Use Cases, Other Approaches for Analysis, Planning a Software Project: Effort Estimation, Project Schedule & Staffing, Quality Planning, Risk Management Planning, Project Monitoring Plan

Design, Coding & Unit Testing: Design: Design Concepts, Function-oriented Design, Object-oriented Design, Detailed Design, Metrics, Coding & Unit Testing: Programming Principles & Guidelines, Incrementally Developing Code, Managing Evolving Code, Unit Testing, Code Inspection, Metrics

Testing & Risk Management: Testing Concepts, Testing Process, Black-box Testing, White-box Testing, Metrics, Risk Analysis & Management: Introduction, Risk Identification, Risk Estimation, Risk Exposure, Risk Mitigation, Risk Management Plans

Web Engineering: The Need for Web Engineering: Introduction, Web Applications vs Conventional Software, The Need for an Engineering Approach, Empirical Assessment, Conclusions

Textbooks

1. Pankaj Jalote: A Concise Introduction to Software Engineering , Springer, 2008 *(Chapters: 1-4, 6-8)*
2. David Gustafson: Software Engineering, Schaum's Outline Series, McGraw Hill, 2002 *(Chapters: 6)*
3. Emilia Mendes, Nile Mosley: Web Engineering, Springer, 2006 *(Chapter: 1)*

Reference Books

1. Roger S. Pressman: Software Engineering A Practitioner's Approach, 7th Edition, McGraw Hill, 2010

Course Delivery

The course will be delivered through task and role based team learning concepts

Course Assessment and Evaluation

	What		To Whom	When/ Where (Frequency in the course)	Max Marks	Evidence Collected	Contribution to Course Outcomes
Direct Assessment Methods	CIE	Internal Assessment Test	Students	Thrice (Average of the best two will be computed)	30	Blue Books	1, 2, 3, 4 & 5
	SEE	Standard Examination		End of Course (Answering 5 of 10 questions)	100	Answer scripts	1,2,3,4 & 5
Indirect Assessment Method	Students Feedback		Students	Middle of the course	-	Feedback forms	1, 2, 3 Delivery of the course
	End of Course Survey			End of the course	-	Questionnaire	1, 2, 3, 4 & 5 Effectiveness of Delivery of instructions & Assessment Methods

Course Outcomes

At the end of the course the students should be able to:

1. Demonstrate an understanding of the principles and techniques of Software Engineering
2. Understand the activities in project management, requirement engineering process and to identify the different types of system models
3. Apply the knowledge of design engineering in software development
4. Formulate different testing methods and tools
5. Recognize the need for web engineering

Mapping Course Outcomes with Programme Outcomes

Course Outcomes	Programme Outcomes											
	1	2	3	4	5	6	7	8	9	10	11	12
1. Demonstrate an understanding of the principles and techniques of Software Engineering	X			X								
2. Understand the activities in project management, requirement engineering process and to identify the different types of system models		X	X	X	X		X		X			X
3. Apply the knowledge of design engineering in software development					X				X			
4. Formulate different testing methods and tools		X		X	X		X		X			
5. Recognize the need for web engineering		X							X			

Course Title: **Software Engineering Lab**	*Course Code*: **CSL515**
Credits (L:T:P) 0:0:1	*Core/ Elective:* **Core**
Type of Course: Practical sessions	**Total Contact Hours: 28 hrs**

Prerequisites: Nil

Course Objectives

Objectives of the course are to:

1. Study and apply principles of engineering to the design, development, and maintenance of software
2. Implement the processes, techniques and deliverables that are associated with requirement engineering including system requirement and system modeling
3. Apply the knowledge, skills and techniques of project management to execute projects effectively and efficiently
4. Provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation

Course Contents

1. Introduction
2. Requirements Engineering
3. Project Management
4. Metrics
5. Risk Management
6. Analysis & Design
7. Testing
8. Quality Assurance

Textbooks

1. Pankaj Jalote: A Concise Introduction to Software Engineering , Springer, 2008
2. David Gustafson: Software Engineering, Schaum's Outline Series, McGraw Hill, 2002
3. Emilia Mendes, Nile Mosley: Web Engineering, Springer, 2006

Reference Books

1. Roger S. Pressman: Software Engineering A Practitioner's Approach, 7th Edition, McGraw Hill, 2010

Course Delivery

The course will be delivered through practical sessions in the laboratory.

Course Assessment and Evaluation

	What		**To Whom**	**When/ Where (Frequency in the course)**	**Max Marks**	**Evidence Collected**	**Contribution to Course Outcomes**
Direct Assessment Methods	CIE	Internal assessment test	Students	Lab test: Once	30	Data sheets	1, 2, 3
		Mini project		Demonstration of techniques learnt: Once	20	Document	1,2, 3, 4, 5
	SEE	Standard Examination		End of the course: Once	100	Answer scripts	1,2, 3, 4, 5
Indirect Assessment Method	Students Feedback		Students	Middle of the course	-	Feedback forms	1, 2, 3 Delivery of the course
	End of Course Survey			End of the course	-	Questionnaire	1, 2, 3, 4 & 5 Effectiveness of Delivery of instructions & Assessment Methods

Course Outcomes

At the end of the course the students should be able to

1. Identify a problem statement, trace the requirements and write unambiguous, correct and consistent SRS
2. Prepare a project plan and estimate effort required for the project
3. Identify, analyze and develop a risk management plan for the potential risks in the project
4. To create a specification of a software artifact intended to accomplish goals
5. To perform exhaustive testing of the code

Mapping Course Outcomes with Programme Outcomes

Course Outcomes	Programme Outcomes											
	1	2	3	4	5	6	7	8	9	10	11	12
1. Identify a problem statement, trace the requirements and write unambiguous, correct and consistent SRS	X	X		X		X	X		X			X
2. Prepare a project plan and estimate effort required for the project	X			X	X							
3. Identify, analyze and develop a risk management plan for the potential risks in the project	X			X								
4. To create a specification of a software artifact intended to accomplish goals	X	X	X	X						X		
5. To perform exhaustive testing of the code	X			X								