RAMAIAH UNIVERSITY OF APPLIED SCIENCES

ASSIGNMENT

Course Code	19CSC212A
Course Name	Software Development Fundamentals
Programme	B. Tech
Department	Computer Science & Engineering
Faculty	Faculty of Engineering Technology

Name of the Student	SUBHENDU MAJI
Reg. No	18ETCS002121
Semester/Year	4th / 2020
Course Leader/s	Ms Sahana P. Shankar

Declaration Sheet				
Student Name	SUBHENDU MAJI			
Reg. No	18ETCS002121			
Programme	B. Tech	Semester/Year	4th / 2020	
Course Code	19CSC212A			
Course Title	Software Development Fundamentals			
Course Date		To		
Course Leader	Ms. Sahana P. Shankar			

Declaration

The assignment submitted herewith is a result of my own investigations and that I have conformed to the guidelines against plagiarism as laid out in the Student Handbook. All sections of the text and results, which have been obtained from other sources, are fully referenced. I understand that cheating and plagiarism constitute a breach of University regulations and will be dealt with accordingly.

Signature of the Student		Date	
Submission date stamp (by Examination & Assessment Section)			

Signature of the Course Leader and date	Signature of the Reviewer and date

Contents

Engineering and Technology			
Ramaiah University of Applied Sciences			
Department	Computer Science and Engineering	Programme	B. Tech.
Semester/Batch	4th/2019		
Course Code	19CSC212A	Course Title	Software Development Fundamentals
Course Leader(s)	Ms. Sahana P. Shankar, Ms. Supriya and Ms. Prakash P		

Questions		Marking Scheme	Marks		
			Max Marks	First Examiner Marks	Moderator
1					
	1.1	Problems with the existing software process models for the current and future software development requirements	3		
	1.2	Design of a software process model which suits the current and future software development requirements	5		
	1.3	Justification of the designed model with an Example	2		
		Question 1 Max Marks	**10**		
		Total Assignment Marks	**10**		

Course Marks Tabulation				
Question	**First Examiner**	**Remarks**	**Moderator**	**Remarks**
1				
Marks (Max 10)				

Signature of First Examiner **Signature of Moderator**

Question No. 1

Solution to Question No. 1:

Now a day, computers running with special purpose application software are being used as an extensive aid to solve complex problems almost each and every place starting from gaming to engineering, industries applications, scientific research and different allied fields. This special purpose software is sometimes unique and distributed in nature with higher degree of complexity. Developing such complex software is not so easy because of the different constraints. Our existing software models do not provide adequate flexibility to be applied for such large and complex projects. So, we must have a better software development process model that will help to overcome these challenges.

ACTIVITIES OF SOFTWARE DEVELOPMENT

Problem solving in software development consists of the following activities:

i. Understanding the problem
ii. Deciding a plan for the solution
iii. Coding the planned solution
iv. Testing the actual program

These activities may be very complex for large systems.
So, each of the activity has to be broken into smaller sub activities or steps. These steps are then handled effectively to produce a software project or system.
The basic steps involved in software project development are:

i. Requirement analysis
ii. Design
iii. Coding
iv. Testing

In addition, there is a fifth step, maintenance that consists of maintaining the system after deployment, i.e. delivery to the customer. Unlike hardware, software does not wear out. But it is very likely that some errors of the system, which were not found during the software testing phase, may be found by the customer. These errors or bugs need to be reported and resolved immediately. Also, over time, as newer technologies and platforms are developed, system starts becoming outdated. It is important to provide new features to the system after intervals and make it compatible with various latest platforms.

A1.1 Problems with the existing software process models for the current and future software development requirements

The earlier software projects were of limited scope with relatively less complexity and smaller size. In contrast, the modern software has wider scope, higher degree of complexity and larger size with better quality, portability and scalability requirements. Sometimes, the modern software has to work with some existing legacy system. Developing such system are more challenging because of the interoperability and dependency factors. The modern real-time systems have lots of critical issues such as time and space complexity require to be addressed.

Recently, lots of new approaches are being used at practice to overcome the modern software crisis.
Some recent trends in modern software developments are listed below:

- Component based software development
- Software reuse
- Aspect oriented software development
- Service oriented software development
- Multi-Tired Software Design
- Object Oriented Software Development
- Standards practices

Problems with some existing software development models:

1. Waterfall Model
    - Real projects rarely follow the sequential approach
    - There is uncertainty at the beginning of the project regarding requirements and goals. This model does not accommodate these uncertainties very well.
    - It does not yield a working version of the system until late in the process
2. incremental model
    - User community needs to be actively involved in the project. This demands on time of the staff and add project delay
    - Communication and coordination skills take a center stage
    - Informal requests for improvement for each phase may lead to confusion
3. spiral model
    - Can be a costly model to use
    - Risk analysis requires highly specific expertise
    - Project's success id highly dependent on risk analysis phase
    - Doesn't work well for smaller projects

model /Features	Requirement Specifications	Cost	Resource Control	Simplicity	Risk Analysis	User Involvement	Flexibility	Reusability
Waterfall	Beginning	Low	Yes	Simple	Only at beginning	Only at beginning	Rigid	Limited
Incremental	Beginning	Low	Yes	Intermediate	No risk analysis	Intermediate	Less Flexible	Yes
Spiral	Beginning	Expensive	Yes	Intermediate	Yes	High	Flexible	Yes
Agile	Frequently changed	Very High	No	Complex	Yes	High	Highly Flexible	Use Case reuse
RUP	Beginning	Expensive	Yes	Simple	Yes	Only at beginning of last phase	Considerable	Supports reusability of existing classes

Reasons for failure of Traditional Models:

1. Non-involvement of the client over the entire project development
2. Lack of better understanding of the system requirements
3. Lack of communications among the team members
4. Lack of project management controls over the entire development period
5. Overlooking verification activity
6. Insufficient documentations
7. Lack of configuration management
8. Non importance to component-based software development and
9. Poor support of component reusability

A1.2 Design of a software process model which suits the current and future software development requirements

After analyzing the importance of all the recent software development trends, a rather new and novel software development process model that adopts the modern software development trends and practices is "**WALTER**".
The schematic diagram of the WALTER model is given in Figure 1.

WALTER Process Model Description:
Unlike the other process models, the WALTER model consists of several phases with distinguished objectives that are discussed in the following section briefly:

Phase 1: Requirement Analysis
The objective of this phase is to identify the exact requirements from the client using different techniques and to specify them in a document for future use after verification. During requirement gathering, the analyst extracts the system requirements from the client. The gathered requirements required to be analyzed for removing the redundancy, incompleteness, inconsistencies, anomalies etc. This phase is often called the requirement analysis phase. Finally, the verified requirements are to be specified in a document called Software Requirements Specification (SRS) and stored for future use.

Phase 2: Feasibility Analysis & Risk Analysis
The objective of this phase is to analyze the suitability of the project in respect to different project attributes to check the different suitability aspects among the alternatives. After carrying out the analysis, the optimal solution is selected. At this stage the project cost estimation has to carry out. The different feasibility i.e. economic feasibility, technical feasibility, operational feasibility has to carry out to manage the different system constraints. Sometimes, the result of the different feasibility analysis may contradict. In such cases, necessary changes, modification and/or negotiation may have to do in the project upon consulting the client. Finally, after verification the result of the feasibility analysis has to be specified in a document called feasibility report and to be kept for future reference. Beside feasibility analysis, at this phase the different project risks have to be identified, analyzed and specified in the risk specification document.

Phase 3: Software Architecture Design
Once the project is confirmed, we must design the software architecture. Software architecture design is a high-level design activity and relatively a recent trend in industries after understanding its importance. We may consider software architecture as abstract design of the complete system. The objective of software architecture design is to identify the subsystems, building blocks or the components of the system along with their communication interfaces expressing their external behavior to improve the project understandability and to communicate with the different stakeholders. The architecture design should reflect the functional requirements specified in the SRS document.

Phase 4: Patterns Identification & Component Search
In general, a system consists of a set of subsystems, so called components. If we analyze any problem, we may find some components common in different projects representing some general structures of a system. These common components are sometimes patterns. The objective of this phase is to identify these patterns. But, to use these predeveloped components efficiently in our system, the system must be designed keeping this objective in mind and the designer should be well aware of the available components in the component library.

Phase 5: Standard Coding & Unit Testing
All the components identified during the last phase may not be available in the component library. The objective of this phase is to write program code for the unmatched components. Often, a few unmatched components may work as desired just with a suitable added interface. In those cases, the benefit analysis must be done to take the decision whether to develop the interface only or the unmatched components from the scratch. The unmatched modules must be coded properly following the standard coding guidelines and practices laid down by the organization itself or the available standard conventions as per the organization interest.

Phase 6: System Building: Component Integration & System Testing
Once all the individual components are gathered, it's the time to integrate these to build the whole system preferably following the bottom up approach. Hence, the objective of this phase is to build the whole system by integrating all the components. However, it is not necessary that, after integrating the pre-tested components successfully, the integrated system will work correctly. Various types of problems such as type mismatch, number of parameter mismatch, return type mismatch etc. may arise. Hence, there is a need to test the integrated system at different level of integration. This is called integration testing.

Phase 7: System Validation
By successful verification of the system, we can only ensure that whatever the functions are implemented in the developed system do work correctly. The objective of this phase is to check whether all the functional requirements as specified in the SRS document specified by the client are exactly included the system or not.

Phase 8: System Deployment & Implementation

Once the system is validated, now it's the time to deliver the system to the client and implement the system at client site. Again, some more changes may be required to accommodate and adjust for proper functioning of the system. Delivering the system to client should not be taken as a formality. Ultimately users are going to use the system.

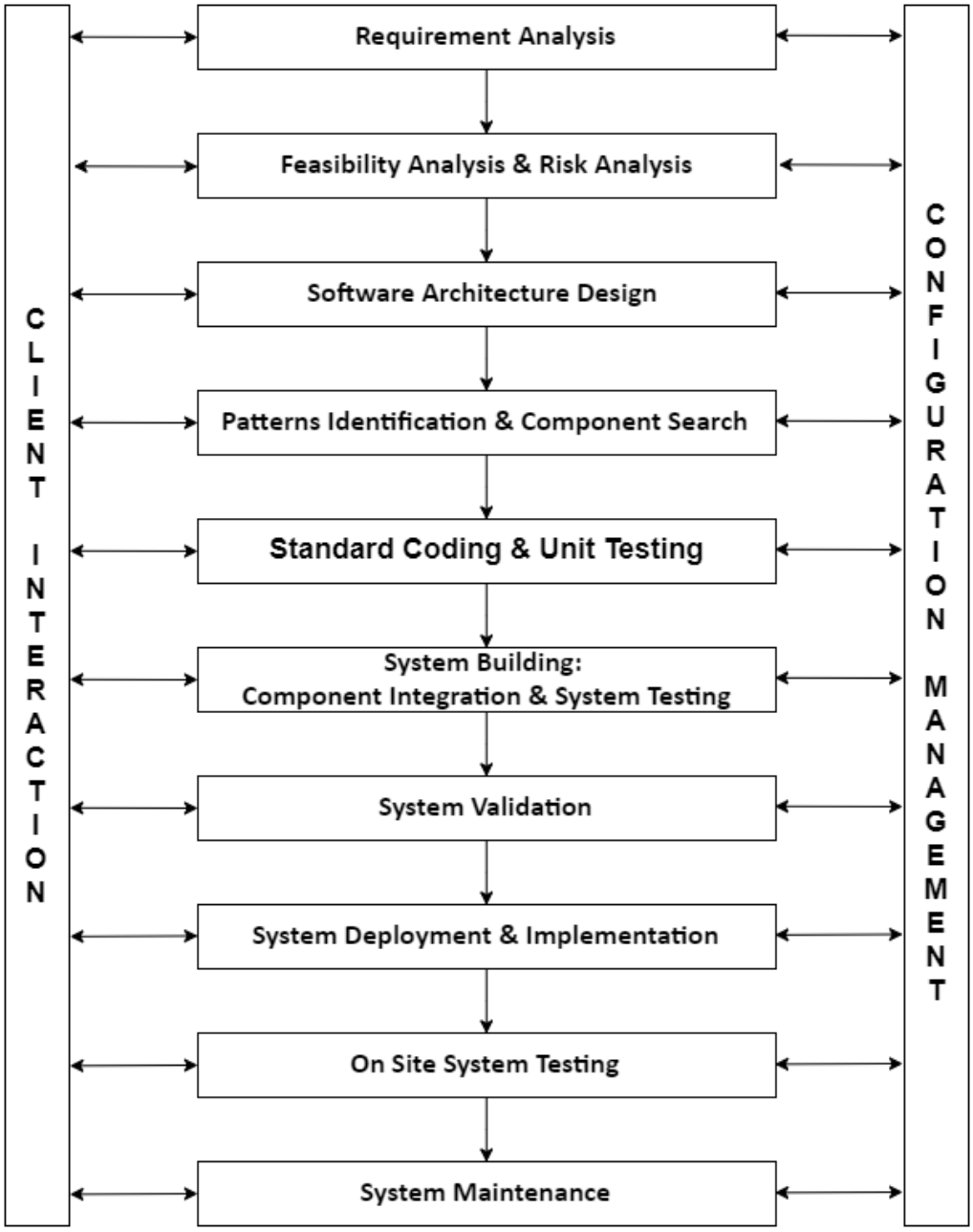

Figure 1 WALTER Model

Phase 9: On Site System Testing
Although, system testing is completed prior to system implementation, but due to different environmental changes and other reasons, the system may not function correctly at the work site. Hence, after implementation, the system needs to be tested at work site too. This testing is called on-site system testing.

Phase 10: System Maintenance
Software Maintenance denotes any changes made to a software product after it has been delivered to the client. Maintenance is a continuous process over the software life cycle. The objective of this phase is to provide the post delivery services to the system for its desirable functioning.

A1.3 Justification of the designed model with an Example

Advantages of WALTER model:

i) It involves the client over the entire development life cycle activities.
ii) It keeps continuous communication with the project management team.
iii) Its explicit verification of individual phases.
iv) Separate software architecture design phase.
v) Separate system deployment phase.
vi) Separate on-site system testing phase.
vii) Supports components-based software development.
viii) It emphasizes on standard coding.
ix) It considers configuration management as a separate activity.
x) It forces to specify all the phase deliverables.
xi) It explicitly instructs to validate the system.

This model can be used to both simple systems as well as complex systems. It supports the object oriented, component-based software development paradigm. By process tailoring, this model also can be applied to develop any software projects that are directly unfit to the actual model.

After the complete analysis, it can be concluded that if the WALTER model is followed to any software project development, most of the software crisis may be overcome up to great extent delivering the fully functional system with better quality within time and budget achieving the true goal of any software project development.

Bibliography

- https://www.researchgate.net/publication/316510707_Analysis_of_various_Software_Process_Models
- https://shodhganga.inflibnet.ac.in/bitstream/10603/172044/16/16_chapter%206.pdf
- https://medium.com/omarelgabrys-blog/software-engineering-software-process-and-software-process-models-part-2-4a9d06213fdc