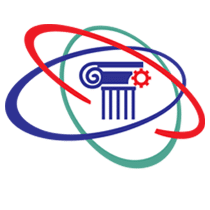
**ACROPOLIS INSTITUTE OF TECHNOLOGY & RESEARCH**



A Software Enginerring Project

**VISUALISER**

Submitted in Partial Fulfillment for the award of

Bachelor of Technology

In

Computer Science

( 2019 – 20 )

Submitted To: Submitted By:

* Nidhi Nigam -Gourav Jaiswal

Software Engineering Approach

**Introduction**

"Paradigm", a Greek word meaning "example", is commonly used to refer to a category of entities that share a common characteristic. Software engineering paradigms are also known as Software engineering models or Software Development Models.

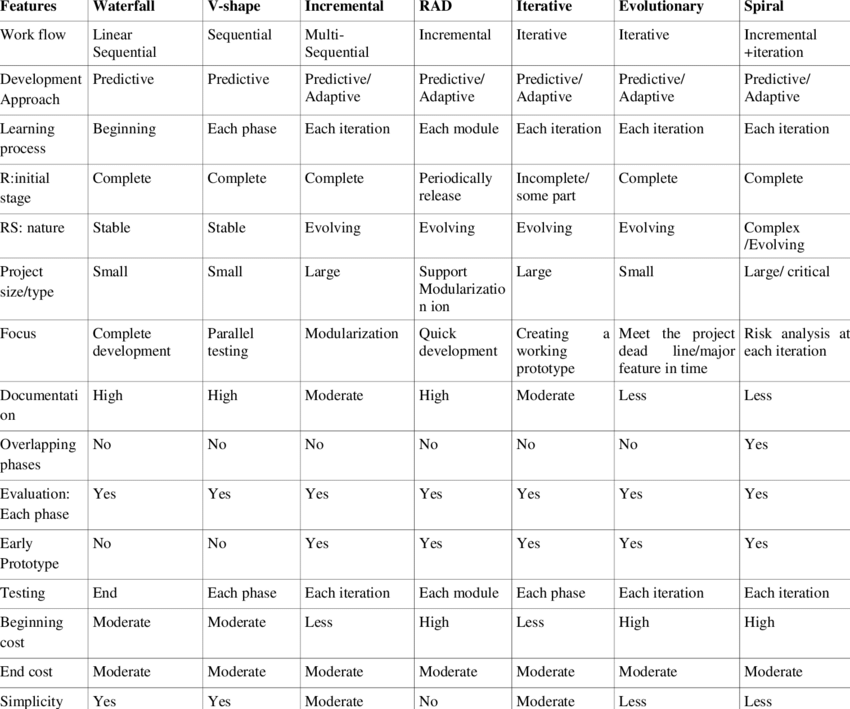
In order to reduce the potential chaos of developing software applications and systems, we use software process models and paradigms that describe the tasks that are required for the building of high-quality software systems. The specific process model or paradigms used to develop a given system depends heavily on the nature of the target system. Use of software paradigms in the development of the software processes has many benefits, including supporting systematic approach and the use of standard approaches and methodologies.

The software engineering paradigm which is also referred to as a software process model or Software Development Life Cycle (SDLC) model is the development strategy that encompasses the process, methods and tools. SDLC describes the period of time that starts with the software system being conceptualized and ends with the software system been discarded after usage.

**Software Engineering Paradigm Applied-**

Software Engineering is the establishment and use of sound engineering principles in order to obtain economically software that is both reliable and works efficiently on real systems. To solve actual problems in an industry or organization setting, such as Central or State Govt. Sectors, Public and Private Sectors, Colleges, Schools, etc.

**Comparing All Software Devlopment Approachs**

****

**Prototyping Approach**

**Definition**  
“Prototyping Approach is the development of a working model which may then be developed further into a fully functioning solution".( Wilson .C, (2001) *Software Design and Development: The Preliminary Course.*)  
  
**Information**  
Prototyping is an engineering technique that has been transferred to computer systems development. The developer makes a small scale model of the proposed program so that users can give feedback and ensure it meets their needs.  
  
**Characteristics**

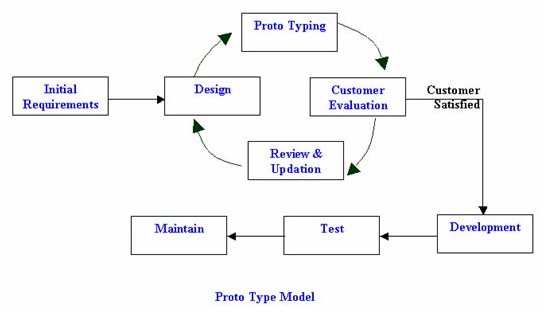
* Prototypes focus on what the users interacts with, the input and output requirements only
* Prototypes are not interested in the code behind the interfaces
* It is a non-formal approach
* It encourages end user participation
* It improves communication between programmer/s and end user/s
* The developer can develop typical screens and then quickly modify them after user feedback
* They give immediate feedback, because the changes can be completed quickly and/or in situ
* You can quickly create the look and feel for the proposed/final product in comparison to the other approaches. Mind you, the final product might take some time to build.
* They typically results in a lower required budget and shorter development time
* They are a good approach for small scale projects
* They can be incorporated within the structured approach
* Prototypes ignore program controls, control structures and error handling because they focus on the user interface.
* Prototypes can evolve into the final product.

**Use**

* multimedia
* website's
* online enquiry

**Personnel**

* analysis
* programming team



**Reasons To Use To Prototype Model-**

Prototype model is used when the desired system needs to have a lot interaction with the end users. Typically, online ,systems, web interfaces have a very high amount of interaction with end users, are best suited for Prototype model. It might take awhile for a system to be built that allow ease of use and needs minimal training for the end users. Prototyping ensures that the end users constantly work with the system and provide a feedback which is incorporated in the prototype to result in useable system. They are excellent for designing good human computer interface systems. This is the reason why we have used this model in our project.

**Twelve Principles of Agile Manifesto**

* **Customer Satisfaction** − Highest priority is given to satisfy the requirements of customers through early and continuous delivery of valuable software.
* **Welcome Change** − Changes are inevitable during software development. Ever-changing requirements should be welcome, even late in the development phase. Agile processes should work to increase customers' competitive advantage.
* **Deliver a Working Software** − Deliver a working software frequently, ranging from a few weeks to a few months, considering shorter time-scale.
* **Collaboration** − Business people and developers must work together during the entire life of a project.
* **Motivation** − Projects should be built around motivated individuals. Provide an environment to support individual team members and trust them so as to make them feel responsible to get the job done.
* **Face-to-face Conversation** − Face-to-face conversation is the most efficient and effective method of conveying information to and within a development team.
* **Measure the Progress as per the Working Software** − Working software is the key and it should be the primary measure of progress.
* **Maintain Constant Pace** − Agile processes aim towards sustainable development. The business, the developers, and the users should be able to maintain a constant pace with the project.
* **Monitoring** − Pay regular attention to technical excellence and good design to enhance agility.
* **Simplicity** − Keep things simple and use simple terms to measure the work that is not completed.
* **Self-organized Teams** − An agile team should be self-organized and should not depend heavily on other teams because the best architectures, requirements, and designs emerge from self-organized teams.
* **Review the Work Regularly** − Review the work done at regular intervals so that the team can reflect on how to become more effective and adjust its behavior accordingly.

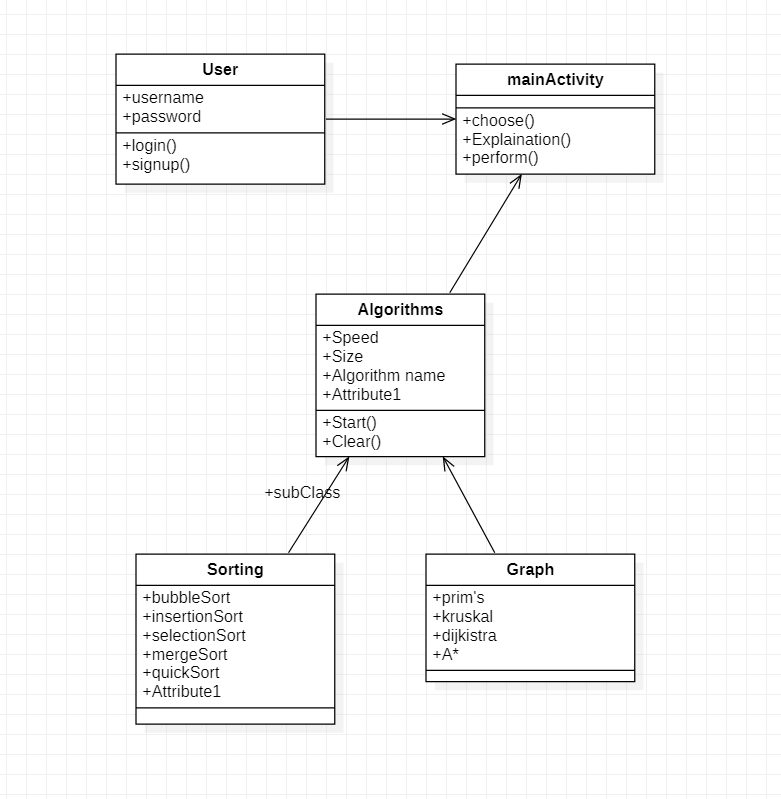
Problem Statement

|  |
| --- |
|  |
|  | Students have difficulties in understanding Graphs and various Sorting algorithms. Various sorting techniques have many linked methodologies which can be remembered but the concepts get faded in the memory after some time. Same thing happens with the concepts of graphs, also graphs require precision, the more precise graphs are made the more accurate results are formed. Manually such accuracy cannot be achieved at all the points.  Solution  Fading of once learned concepts can be forgotten if things are learned visually.  When we speak of graphs and sorting methods, both require precision and good  Visualization . VISUALISER enables the students to learn the concepts of both  graphs and sorting methods by visualizing them instead of mugging up the process. |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |



Diagram

**Class Diagram**



**Activity Diagram**

