EXTENDED ABSTRACT

1) This paper proposes to introduce a modified software engineering methodology to enhance the Software Development Life Cycle that is used as the primary mode for software development.

2) This method uses the fact that every software project that is not having a novel definition will be having some common phases, modes of development, processes, information or input/output details that were used by earlier or other projects.

3) This method uses the principles of inheritance and reusability of Object-Oriented Analysis & Design.

4) In this method, we first create a repository of information, processes, phases and their details, that were not used earlier.

5) In other words, create a record of all the details of novel and unique phases and processes that were involved.

6) Then, whenever successive phases of any project are to be executed, records for similar types of phases and processes are checked and if found, are retrieved.

7) This information is used for execution of current project.

8) This cuts down time, costs and error problems.

9) In this method, any changes or replacements from the record database can be made as and when required.

INTRODUCTION

As we all know that the traditional SDLC cycle has many shortcomings. One of them is the ignoring of repeated processes being executed repeatedly, causing money and time loses. Also, the knowledge of executing previous SDLC cycles is not utilized (except a little in Requirement Analysis and System Analysis stage). Thus, it causes non-uniformity and error-prone behavior in systems. Thus, we need a method or adjustment to the current SDLC cycle that will allow us to correct such problems. The traditional SDLC cycle has these stages:

1.) Problem Definition
2.) Requirement Analysis
3.) System Analysis
4.) System Design
5.) System Coding
6.) Testing
7.) Implementation
8.) Maintenance & Follow-Up.

In each of these phases, we do a fresh execution, without utilizing past experience and know-how. We can change these phases for better execution and follow up of phases in SDLC.

METHODOLOGY USED

This paper has used the traditional method of Scientific Proving using facts and figures. There was no need for tables to be made. However, we have utilized the works of previous authors for analyzing them (the works), finding any problems to them and then refining them to obtain a better solution. This is proposed method, and is completely original and developed by the author of this paper (Jitesh Dundas) himself, under the guidance of the director of his college. The observations were made from facts, the current scenario in the IT Industry and in the education institutions. Also, logical and scientific thinking has been used to record the observations made in this paper.
CONCEPT USED

This improved cycle is called Object-Oriented System Development Life Cycle (OO-SDLC). This method is used the traditional SDLC cycle with some changes. At every stage, we put in place a checkpoint, when we try to compare the current phase with those phases that have been executed earlier.

Here, we combine the principles of SDLC, OOAD and DBMS for the evolvement of the new cycle. Specifically, we have used reusability, inheritance and Traditional SDLC.

ALGORITHM USED

The OO-SDLC has the following steps:

a.) To create record database of phases that are new and distinct.(not recorded before)

b.) For each project (SDLC cycle), do the following:

   b.1) Select the phase to be executed in current project.

   b.2) Compare it with those in database. If found similar or same , then use the phase information (that is needed) of previous projects from the record database.

   b.3) Using these details and know-how, execute the phase of current project at hand.

   b.4) Once done, move to next phase and repeat the process.

   c.) If the information and know-how of current phase execution was better, then replace or update the changes in the record database.

   d.) Record any details or problems encountered in current project. Also, look for solutions from the database.

   e.) Thus in this way, repeat the process for each project

DESCRIPTION

According to the algorithm, we first understand the project phase at hand. Next we compare it to the records in the database to find if any previous project’s know-how and information can be utilized here. If found, then that record and information is fetched and studied by the project manager. We then analyze this information (project and process information) and try to inherit this in our current phase execution. Next, we record the results and report new problems that may have occurred. Also, we update or replace any changes that may be needed to the record database. Again, we repeat the same steps for the other project phases. Once project is executed, we can repeat the process for each new project that comes.

Clearly, this process is just providing an updation to the traditional SDLC cycle.

NOTE:-
The SDLC life cycle is famous all around the world and used by everyone for software development. Thus, there is no need to explain the SDLC in this paper. One can refer the books given in the reference section of this paper to know more about the traditional SDLC.

Also, as we stress on reusing and inheriting the knowledge and data that exists with us, we can conclude that we are using Object-Oriented Concepts in the SDLC Life Cycle. Thus, it has been called Object-Oriented SDLC.
**Record Database**

It contains all the information regarding previous projects, its phases, and processes involved in them. Also, it has solutions, updates and expected problems in SDLC.
FUTURE SCOPE OF THE SDLC LIFE CYCLE
CONCLUSION
Thus it is clear that the way in which software is developed has gone a lot of change in the past. Better technologies, better platforms, better computers have made software development better, easier and faster. It can be concluded that the software development life cycle will become, as time goes, shorter, complex and time-saving. However, it also depends upon us on how we can best manage software development.

FUTURE SCOPE

( Please see the diagram on Page-4 for the section below )
The details given in this section try to reflect or expect the changes in the SDLC cycle in the coming years. This has been done by observation and facts from studying the techniques and technologies used in the world today.

The phase of problem definition and requirement analysis will remain as such or (upto a minor extent changed). They will keep doing their activities as specified earlier.

The analysis part may see a more flattened approach.
Currently, the development cycle till now is more vertical than horizontal. In short, when one phase completes for the entire system, the next phase commences. This implies:
1) each phase is dependent on its previous phase
2) It is time-consuming

As time goes and development cycle shortens, the lower phases will tend to be executed in parallel.
For e.g.) When a module of system is designed, the coding phase for that module will start followed by Unit Testing. Thus, the cycle will become more and more horizontal rather than parallel.

Also, the coding phase can be expected to be automated with the help of IDE’s and Code Generators. Also, unit testing (and white-box testing techniques) can be expected to follow the same. They will be merged into Design Phase.
Also, the Design phase will emphasize the study and design of integration issues of the system.

For eg.) Lotus Notes Domino Designer (used for workflow application development ) is more of a Design Tool and not a coding tool. Also, some IDE tools for Assembly language and embedded systems programming are addition to it.

Next, the Design phase will be followed by the Implementation phase.
Thus, the implementation phase would include the white-box testing and others.

Thus testing here will concern with performance and user-acceptance issues.
The implementation will move in paralled with UAT too. As requirements change or during the implementation stage, the user asks for minor changes , then UAT will be changed accordingly.
The Training & Review Stages can be put next in line of execution. The system after training will undergo reviews for bug-fixing, enhancements and improvements.
The Support(Maintenance) stage for a system is the longest of all stages. All the change requests, bug-fixing and improvements come under it. They will be recorded under the Review stage.
Each change request in this stage for the current system will have its own method of execution :
1) The review stage gives the requirements.
2) Then, impact analysis of the system is done to find and analyze the bug or improvements.
3) A strategy or blue-print is made to solve the problem (Like the design stage)
4) Implementation and Testing will be done. And so on.

Thus, we can say that the Maintenance stage for the existing system is comprised of its own process of execution and instead of creating this stage, we could execute each issue with an SDLC cycle. Thus, the Maintenance stage can be removed and the feedback stage can be given a link back to the top. Thus each feedback will be shaped as a requirement.

In all the Stages of SDLC, there will be an input and output movement of knowledge from the Centralized Knowledge Database.
If similar systems exist, then we can pull out or inherit that knowledge from it and move ahead. In case there is better knowledge, then we can update it into the database. This activity is done before that phase execution.
Also, external Reviewers or people outside the project scope with the expected system knowledge can be called to contribute to this system.

Thus, the SDLC will become shorter, faster and automated. Also, it is not correct that requirements can be frozen. System may get updates or issues at any stage and so must get updated. Thus the need for a centralized database arises.
REFERENCES

