Evolving A New Model (SDLC Model-2010) For Software Development Life Cycle (SDLC)

PK.Ragunath, S.Velmourougan #, P. Davachelvan * ,S.Kayalvizhi, R.Ravimohan†

Department of Bioinformatics, Sri Ramachandra University, Porur, Chennai , India,
Centre For Reliability (CFR), STQC Directorate (Govt. of India),Chennai. India
*Pondicherry University , Pondicherry, India

Summary
Structured project management techniques (such as an SDLC) enhance management's control over projects by dividing complex tasks into manageable sections. A software life cycle model is either a descriptive or prescriptive characterization of how software is or should be developed. But none of the SDLC models discuss the key issues like Change management, Incident management and Release management processes within the SDLC process, but, it is addressed in the overall project management. In the proposed hypothetical model, the concept of user-developer interaction in the conventional SDLC model has been converted into a three dimensional model which comprises of the user, owner and the developer. In the proposed hypothetical model, the concept of user-developer interaction in the conventional SDLC model has been converted into a three dimensional model which comprises of the user, owner and the developer. The "one size fits all" approach to applying SDLC methodologies is no longer appropriate. We have made an attempt to address the above mentioned defects by using a new hypothetical model for SDLC described elsewhere. The drawback of addressing these management processes under the overall project management is missing of key technical issues pertaining to software development process that is, these issues are talked in the project management at the surface level but not at the ground level.
.

Key words:
Sofware Development ,SDLC Model-2010,
Project Management, SDLC models, SDLC Phases.

1. Introduction

PROJECT MANAGEMENT IN SDLC

Organizations may employ an SDLC model or alternative methodology when managing any project, including software development, or hardware, software, or service acquisition projects. Regardless of the method used, it should be tailored to match a project's characteristics and risks. Boards, or board-designated committees, should formally approve project methodologies, and management should approve and document significant deviations from approved procedures.

Structured project management techniques (such as an SDLC) enhance management's control over projects by dividing complex tasks into manageable sections. Segmenting projects into logical control points (phases) allows managers to review project phases for successful completion before allocating resources to subsequent phases. The number of phases within a project's life cycle is based on the characteristics of a project and the employed project management methodology. A five-step process may only include broadly defined phases such as prepare, acquire, test, implement, and maintain. Typical software development projects include initiation, planning, design, development, testing, implementation, and maintenance phases. Some organizations include a final, disposal phase in their project life cycles. The activities completed within each project phase are also based on the project type and project management methodology. All projects should follow well-structured plans that clearly define the requirements of each project phase. Iteration enhances a project manager's ability to efficiently address the requirements of each party (end users, security administrators, designers, developers, system technicians, etc.) throughout a project's life cycle. Iteration also allows project managers to complete, review, and revises phase activities until they produce satisfactory results (phase deliverables).[1]

What is SDLC

A software cycle deals with various parts and phases from planning to testing and deploying software. All these activities are carried out in different ways, as per the needs. Each way is known as a Software Development Lifecycle Model(SDLC)[2].A software life cycle model is either a descriptive or prescriptive characterization of how software is or should be developed. A descriptive model describes the history of how a particular software system was developed. Descriptive models may be used as the basis for understanding and improving software development processes or for building empirically grounded prescriptive models. [3]

Manuscript received January 5, 2010
Manuscript revised January 20, 2010

SDLC models

* **The Linear model (Waterfall)**
- Separate and distinct phases of specification and development.
- All activities in linear fashion.
- Next phase starts only when first one is complete.

* **Evolutionary development**
- Specification and development are interleaved (Spiral, incremental, prototype based, Rapid Application development).
- Incremental Model (Waterfall in iteration),
- RAD(Rapid Application Development) - Focus is on developing quality product in less time,
- **Spiral Model** - We start from smaller module and keeps on building it like a spiral. It is also called Component based development.

* **Formal systems development**
- A mathematical system model is formally transformed to an implementation.

* **Agile Methods.**
- Inducing flexibility into development.

* **Reuse-based development**
- The system is assembled from existing components.

The General Model

Software life cycle models describe phases of the software cycle and the order in which those phases are executed. There are tons of models, and many companies adopt their own, but all have very similar patterns. The general, basic model is shown below:

General Life Cycle Model

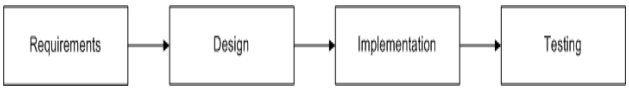

Fig 1 General Life Cycle Model

Each phase produces deliverables required by the next phase in the life cycle. Requirements are translated into design.[4] Code is produced during implementation that is driven by the design. Testing verifies the deliverable of the implementation phase against requirements.

Waterfall Model

The waterfall model is a sequential software development process, in which progress is seen as flowing steadily downwards (like a waterfall) through the phases of Conception, Initiation, Analysis, Design (validation), Construction, Testing and maintenance. Small to medium database software projects are generally broken down into six stages:

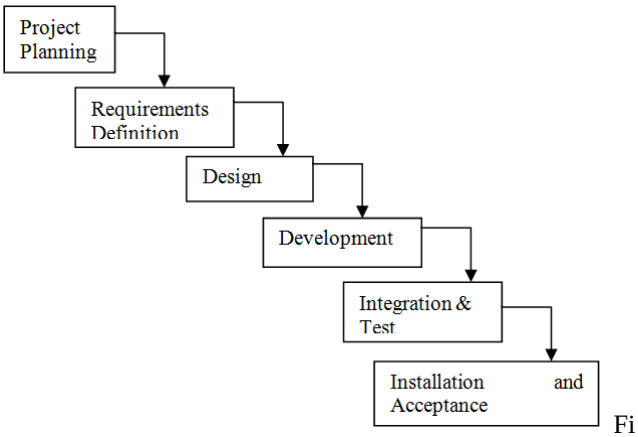

Fig 2. Waterfall Model

The unmodified "waterfall model". Progress flows from the top to the bottom, like a waterfall. The waterfall development model has its origins in the manufacturing and construction industries; highly structured physical environments in which after-the-fact changes are prohibitively costly, if not impossible. Since no formal software development methodologies existed at the time, this hardware-oriented model was simply adapted for software development.

The first formal description of the waterfall model is often cited to be an article published in 1970 by Winston W. Royce (1929–1995),[5] although Royce did not use the term "waterfall" in this article. Royce was presenting this model as an example of a flawed, non-working model (Royce 1970). This is in fact the way the term has generally been used in writing about software development—as a way to criticize a commonly used software practice.[6]

Waterfall Lifecycle Model

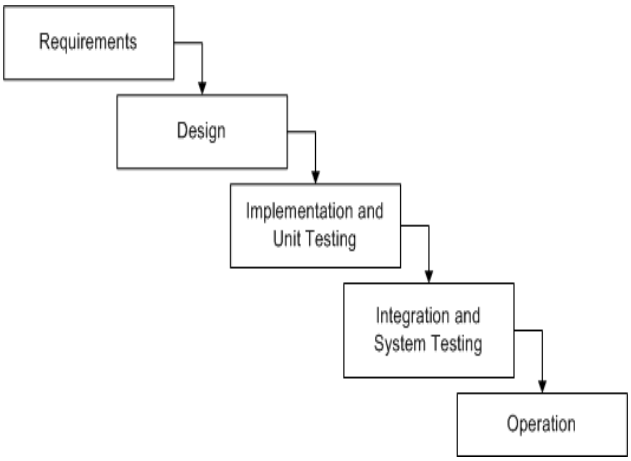

Fig 3 Waterfall Lifecycle Model

Advantages

a) Simple and easy to use.
b) Easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.
c) Phases are processed and completed one at a time.
d) Works well for smaller projects where requirements are very well understood.

Disadvantages

a) Adjusting scope during the life cycle can kill a project
b) No working software is produced until late during the life cycle.
c) High amounts of risk and uncertainty.
d) Poor model for complex and object-oriented projects.
e) Poor model for long and ongoing projects.
f) Poor model where requirements are at a moderate to high risk of changing.

V-Shaped Model

Just like the waterfall model, the V-Shaped life cycle is a sequential path of execution of processes. Each phase must be completed before the next phase begins. Testing is emphasized in this model more so than the waterfall model though. The testing procedures are developed early in the life cycle before any coding is done, during each of the phases preceding implementation. Requirements begin the life cycle model just like the waterfall model. Before development is started, a system test plan is created. The test plan focuses on meeting the functionality specified in the requirements gathering.

The high-level design phase focuses on system architecture and design. An integration test plan is created in this phase as well in order to test the pieces of the software systems ability to work together. The low-level design phase is where the actual software components are designed, and unit tests are created in this phase as well. The implementation phase is, again, where all coding takes place. Once coding is complete, the path of execution continues up the right side of the V where the test plans developed earlier are now put to use. [7]

V-Shaped Life Cycle Model

Advantages

a) Simple and easy to use.
b) Each phase has specific deliverables.
c) Higher chance of success over the waterfall model due to the development of test plans early on during the life cycle.
d) Works well for small projects where requirements are easily understood.

Disadvantages

a) Very rigid, like the waterfall model.
b) Little flexibility and adjusting scope is difficult and expensive.
c) Software is developed during the implementation phase, so no early prototypes of the software are produced.
d) Model doesn't provide a clear path for problems found during testing phases.

Incremental Model

In the incremental model, you construct a partial implementation of a total system. Then you slowly add increased functionality. The incremental model prioritizes requirements of the system and then implements them in groups. Each subsequent release of the system adds function to the previous release, until all designed functionality has been implemented. [8]

Incremental Life Cycle Model

Advantages

a) Generates working software quickly and early during the software life cycle.
b) More flexible – less costly to change scope and requirements.
c) Easier to test and debug during a smaller iteration.
d) Easier to manage risk because risky pieces are identified and handled during its iteration.
e) Each iteration is an easily managed milestone.

Disadvantages

a) Each phase of an iteration is rigid and do not overlap each other.
b) Problems may arise pertaining to system architecture because not all requirements are gathered up front for the entire software life cycle.

Spiral Model

The spiral model was defined by Barry Boehm in his 1988 article A Spiral Model of Software Development and Enhancement. This model was not the first model to discuss iterative development, but it was the first model to explain why the iteration matters. As originally envisioned, the iterations were typically 6 months to 2 years long. Each phase starts with a design goal and ends with the client (who may be internal) reviewing the progress thus far. Analysis and engineering efforts are applied at each phase of the project, with an eye toward the end goal of the project. [9]

The spiral model is similar to the incremental model, with more emphases placed on risk analysis. The spiral model has four phases: Planning, Risk Analysis, Engineering and Evaluation. A software project repeatedly passes through these phases in iterations (called Spirals in this model). The baseline spirals, starting in the planning phase, requirements are gathered and risk is assessed. Each subsequent spiral

builds on the baseline spiral. Requirements are gathered during the planning phase. In the risk analysis phase, a process is undertaken to identify risk and alternate solutions. A prototype is produced at the end of the risk analysis phase. Software is produced in the engineering phase, along with testing at the end of the phase. The evaluation phase allows the customer to evaluate the output of the project to date before the project continues to the next spiral. In the spiral model, the angular component represents progress, and the radius of the spiral represents cost.

Spiral Lifecycle Model

Advantages

a) High amount of risk analysis.
b) Good for large and mission-critical projects.
c) Software is produced early in the software life cycle.

Disadvantages

a) Can be a costly model to use.
b) Risk analysis requires highly specific expertise.
c) Project's success is highly dependent on the risk analysis phase.
d) Doesn't work well for smaller projects.

Disadvantages of existing SDLC Model

But none of the SDLC models discuss the key issues like Change management, Incident management and Release management processes within the SDLC process, but, it is addressed in the overall project management.

a) The drawback of addressing these management processes under the overall project management is missing of key technical issues pertaining to software development process that is, these issues are talked in the project management at the surface level but not at the ground level.

Example: For the functional requirement changes and its associated software requirement, design, code and default maintenance are addressed in the project level change management will be at document level not the technical aspects which are essential to the implementation of the functions or the module.

b) Each and every incident/failure/bugs that come across during the development are not mostly recorded and escalated to the project
management until and unless otherwise it results in major mission losses.

c) There is a lack of understanding that these are vital data may be used to establish performance and reliability of the product being developed.

d) Finally the release management is at present looked from the project management angle, but there are weaknesses due to non-linking of certain issues which are driven to rip during development and only the surface level issues like configuration, setup creation, registration of components are addressed but the compatibility between existing components and newer components added after installation and default screens (hidden) and hidden parameters, test stubs used during testing, components unused for future modification and relevant vulnerabilities are not addressed.

We have made an attempt to address the above mentioned defects by using a new hypothetical model for SDLC described elsewhere.

Material and Methods

New features proposed in the SDLC model-2010:

a) The hypothetical model proposed addresses all these issues by embedding the core control which is mapped to the project management and traceable to the surface level data existing to the project management and the ground level technical data additionally present to address quality attributes such as security, safety and performance and installability etc, issues during or after the deployment of the software.

b) In addition to this it has modified the concept of user-developer interaction in the conventional model to a three dimensional model which comprises of the user, owner and developer. It also establishes a clear guideline to address who has to do what in the various stages in SDLC? and when in SDLC?, so that the conflicts among the developer and user is removed also enables the timely completion of projects by removing the hidden overheads that occur due to conflicts or lack of knowledge.

c) Moreover the outer circle of the model depicts the various quality attributes pertaining to the various actions and the activities to be done by everyone involved in the development activities from requirement gathering to maintenance in the SDLC phases.

d) In the proposed hypothetical model, the concept of user-developer interaction in the conventional SDLC model has been converted into a three dimensional model which comprises of the user, owner and the developer.

e) It also defines clearly a set of guidelines as to who has to do what in SDLC? and when in the various stages in SDLC? Or in other words the roles and responsibilities are clearly defined in these guidelines.

This ensures that conflicts among the developer and the user are eliminated or reduced to a manageable level. Hence, timely completion of software projects within the set timelines is made possible by removing the hidden overheads that arise due to lack of knowledge.

f) To ensure that the quality attributes of a software project are adhered to, an outer circle has been added to ensure that these quality attributes are embedded in the various activities and actions done by everyone involved in the development activities from requirement gathering to maintenance in various phases of SDLC.

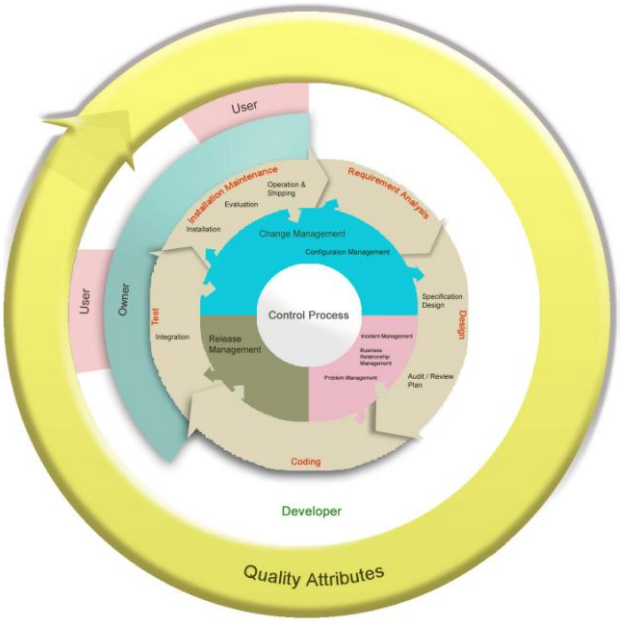

Fig 4 SDLC Model-2010.

Lightweight methodologies were developed to efficiently manage software projects subjected to short timelines and excessive uncertainty and change. Reducing the time-to-market is a way of life for most companies. [10] Shrinking cycle times are commonplace in the software industry. Web development efforts are an excellent example of reduced cycle timing. [11] They differ from the development of traditional applications in many ways. Lightweight SDLC methodologies seek to balance these two extremes.[12] They are an example of the application of the risk-reward approach to investing time and resources in various development activities. Lightweight methodologies explore the line of bare sufficiency. [13] A recent study by the Cutter Consortium found that traditional SDLC methodologies "fall short in the new e-business environment. They are unable to keep up with fast-paced, ever-changing e-business projects". [14] Perhaps the greatest strength of the new lightweight methodologies is that they provide a palatable alternative to the code and fix mentality that permeates today's environment. [15] On the other hand, one of the biggest limitations of lightweight SDLCs is their inability to handle large development teams. [16] As expected, lightweight methodologies are strong in some areas and weak in others. On the negative side, when a project falls short of being barely sufficient, failure occurs. [17] On the positive side is the example of the Chrysler Compensation System discussed earlier. After a 26-person development team failed to complete what was considered a large system that required heavy SDLC, an eight-person team using XP successfully completed the project in one year. [18] Key to success when employing lightweight methodologies is their application to projects with one or more of the following enablers.

a) Small co-located development teams (i.e. two to eight people in a room).
b) On-site customer or usage expert.
c) Short increments between deliverables.
d) Fully automated regression tests.
e) Experienced developers.

As the number of the above characteristics increase, so does the probability of success for a project employing lightweight SDLCs.

The "one size fits all" approach to applying SDLC methodologies is no longer appropriate.[19] Each SDLC methodology is only effective under specific conditions. Traditional SDLC methodologies are often regarded as the proper and disciplined approach to the analysis and design of software applications. [20] Examples include the code and fix, waterfall, staged and phased development, transformational, spiral, and iterative models. Lightweight methodologies on the other hand are a compromise between no process and too much process.

The following details explain the applicability of the new proposed SDLC model. The details given below explain how the newly proposed SDLC model has the strength of improving Reliability of the software. This new SDLC Model has been applied to both these software.

(a). Spare Parts Cost Optimisation Software.

(b). Academic HR Manager.

Implementation of the' Proposed SDLC Model-2010':-
Software 1(Spare Parts Cost Optimisation Software)

This software is used for optimising the cost of parts for manufacturing. The language used was Microsoft Visual Basic 6.0 as front end and Microsoft Access as the database. This software is a tool to estimate the optimum number of spare parts required to be stocked in order to improve the availability of the electronic systems, cost effectively, based on scientific approach. Fig.5 and Fig.6 illustrate the logical diagram of the software and flow diagram respectively.

The estimated reliability of software developed was compared with and without the use of the proposed new SDLC model and the results are compared and have been tabulated below (Table 1 and Table 2).While designing this software, the new hypothetical SDLC model has been used.

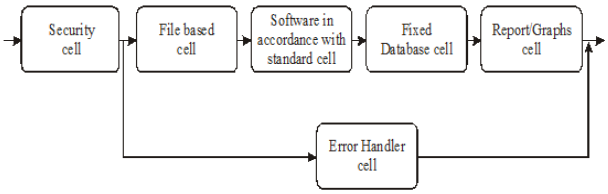

Fig. 5.Logical flow diagram of the software

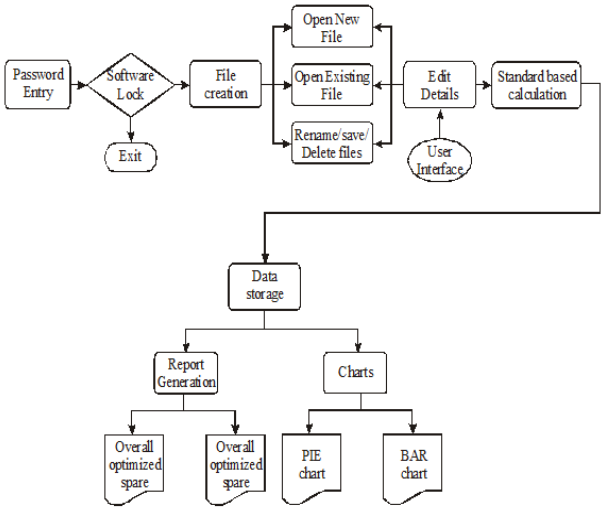

Fig. 6. Functional diagram of Spare Parts Software

Software 2 (Academic HR Manager)

The Department of Bioinformatics (which is an ISO 9001:2000 certified) in Sri Ramchandra University (www.srmc.edu), Chennai, India have developed a new Human Resource software. (Academic HR Manager)

This software helps in

A) Workload allocation of academic staff in the department,

B) Monitoring of workload allocation of academic staff in the department, and

C) Analysis of total work carried out by the academic staff of the Department.

While designing this software, the new hypothetical SDLC model has been used.

RESULT

This SDLC model-2010 has been applied while developing both the software. viz., Spare parts cost Optimisation and Academic-HR Manager. The following are the results of applying the new software model to the following software. viz.,

a)Spare parts cost Optimisation and

b)Academic-HR Manager.

The estimated reliability of software developed was compared with and without the use of the proposed new SDLC model and the results are compared and have been tabulated below (Table 1 and Table 2).While designing this software, the new hypothetical SDLC model has been used.

TABLE 1 THE FAILURE RATE WITH THE EXISTING SDLC MODEL

Name of the cell	Projected no. of failure (r)
Security cell	4
File based cell	6
S/w in accordance with std.	6
Fixed database	4
Report/Analysis	3
Error Handler	7
Repetitive failures	13
Configuration errors	23
Version conflict errors	3
Failures due to lack infrastructure	7
Awareness Failure	3
Total No . of failure	79

(f=R/r*15) (15 similar test benches were created to test the product)

TABLE 2 THE FAILURE RATE WITH THE NEW SDLC MODEL

Name of the cell	No. of failures (N)
Security cell	0
File based cell	0
S/w in accordance with std.	1
Fixed database	0
Report/Analysis	0
Error Handler	1
Repetitive failures	0
Configuration errors	2
Version conflict errors	0
Failures due to lack infrastructure	1
Awareness Failure	1
Total No . of failure	6

(f=R/r*15) (15 similar test benches were created to test the product)

Discussion

But none of the SDLC models discuss the key issues like Change management, Incident management and Release management processes within the SDLC process, but, it is addressed in the overall project management.

The drawback of addressing these management processes under the overall project management is missing of key technical issues pertaining to software development process

that is, these issues are talked in the project management at the surface level but not at the ground level.

This drawback in the existing SDLC model is rectified by the hypothetical new SDLC model by embedding the core control which is mapped to the project management and traceable to the surface level data existing to the project management and the ground level technical data additionally present to address quality attributes such as security, safety and performance and installability etc, issues during or after the deployment of the software.

Each and every incident/failure/bugs that come across during the development are not mostly recorded and escalated to the project management unless otherwise it results in major mission losses.

To ensure that the quality attributes of a software project are adhered to, an outer circle has been added to ensure that these quality attributes are embedded in the various activities and actions done by everyone involved in the development activities from requirement gathering to maintenance in various phases of SDLC.
We have modified the user-developer interaction in the existing SDLC model, from a two dimensional one to a three dimensional model in the new proposed SDLC model which comprises of the user, owner and developer. It also establishes a clear guideline to address who has to do what in SDLC? and when in the various stages of SDLC?

There is a lack of understanding that these are vital data may be used to establish performance and reliability of the product being developed.

This defect of the existing SDLC model has been addressed in the proposed new SDLC model by eliminating the conflicts among the developer and user and also enables the timely completion of projects by removing the hidden overheads that occur due to conflicts or lack of knowledge.

Advantages of the New Model

a) The hypothetical model proposed addresses all these issues by embedding the core control which is mapped to the project management and traceable to the surface level data existing to the project management and the ground level technical data additionally present to address quality attributes such as security, safety and performance and installability etc, issues during or after the deployment of the software.
b) In addition to this it has modified the concept of user-developer interaction in the conventional model to a three dimensional model which comprises of the user, owner and developer. It also establishes a clear guideline to address who has to do what in SDLC? and when in the different stages of SDLC?. So that the conflicts among the developer and user is removed also enables the timely completion of projects by removing the hidden overheads that occur due to conflicts or lack of knowledge.
c) Moreover the outer circle of the model depicts the various quality attributes pertaining to the various actions and the activities to be done by everyone involved in the development activities from requirement gathering to maintenance in the SDLC phases.
d) In the proposed hypothetical model, the concept of user-developer interaction in the conventional SDLC model has been converted into a three dimensional model which comprises of the user, owner and the developer.
e) Very clear definition of roles and responsibilities ensures that conflicts among the developer and the user are eliminated or reduced to a manageable level. Hence, timely completion of software projects within the set timelines is made possible by removing the hidden overheads that arise due to lack of knowledge.
f) To ensure that the quality attributes of a software project are adhered to, an outer circle has been added to ensure that these quality attributes are embedded in the various activities and actions done by everyone involved in the development activities from requirement gathering to maintenance in various phases of SDLC.

Scope for further study

The dependent processes like incident management, configuration management and release management could be explored within the scope of development and maintenance. Further one to one mapping between the phases of SDLC with these processes can be established.

Bibliography

[1] following standards were used as guides to develop this SDLC description. The standards were reviewed and tailored to fit the specific requirements of small database projects.
- ANSI/IEEE 1028: Standard for Software Reviews and Audits
- ANSI/IEEE 1058.1: Standard for Software Project Management Plans
- ANSI/IEEE 1074: Standard for Software Lifecycle Processes
- SEI/CMM: Software Project Planning Key Process Area.

[2] Raymond Lewallen - CodeBetter.Com - Stuff you need to Code Better! Published 08-01-2008 10:27 AM.
[3] Craig Larman, Victor R. Basili, "Iterative and Incremental Development: A Brief History," Computer, vol. 36, no. 6, pp. 47-56, June 2003, doi:10.1109/MC.2003.1204375.
[4] www.coders2020.com/what-is-sdlc-what-are-the-various-sdlc-models-explain-them.
[5] Curtis, Krasner, Iscoe, 1988.
[6] Dr. Winston W. Royce(1929 - 1995) at www.informatik.uni-bremen.de. Retrieved 27 Oct 2008.

[7] Conrad Weisert, Information Disciplines, Inc., Chicago, 8 February, 2003
[8] Craig Larman, Victor R. Basili (June 2003). "Iterative and Incremental Development: A Brief History.
[9] Using Both Incremental and Iterative Development.Dr. Alistair Cockburn, Humans and Technology, Crosstalk May 2008.
[10] Sims, D. (1997). Vendors struggle with costs, benefits of shrinking cycle times. IEEE Computer, 30(9), 12-14.
[11] Strigel, W. (2000, February). Is Web Development a New Creature? Software Productivity Center. Retrieved March 11, 2001, from the World Wide Web: http://www.spc.ca/resources/essentials/feb2300.htm#three.
[12] Yourdon, E. (2000, October). The Emergence of "Light" Development Methodologies. Software Productivity Center. Retrieved March 11, 2001, from the WorldWideWeb: http://www.spc.ca/resources/essentials/oct1800.htm#3.
[13] Cockburn, A. (2001). Just-In-Time Methodology Construction. Humans and Technology. Retrieved March 11, 2001, from the World Wide Web: http://www.crystalmethodologies.org/articles/jmc/justintime methodologyconstruction.html
[14] Cutter. (2000, October). Light Methodologies Best for E-business Projects. Cutter Consortium. Retrieved March 11, 2001, from the World Wide Web: http://cutter.com/consortium/research/2000/crb001003.html.
[15] Fowler, M. (2000, December). Put Your Process on a Diet. Software Development Online. Retrieved March 11, 2001, from the World Wide Web: http://www.sdmagazine.com/articles/2000/0012/0012a/0012a .htm.
[16] Fowler, M. (2000, November). The New Methodology. ThoughtWorks. Retrieved March 11, 2001, from the World Wide Web: http://www.martinfowler.com/articles/newMethodology.html.
[17] Cockburn, A. (2000, July/August). Selecting a project's methodology. IEEE Software, 17(4), 64-71.
[18] Cockburn, A. (2000, September). Balancing Lightness with Sufficiency. American Programmer. Retrieved March 11, 2001, from the World Wide Web: http://www.crystalmethodologies.org/articles/blws/balancingl ightnesswithsufficiency.html.
[19] Lindvall, M., & Rus, I. (2000, July/August). Process diversity in software development. IEEE Software, 17(4), 14-18.
[20] Rothi, J., & Yen, D. (1989). System Analysis and Design in End User Developed Applications. Journal of Information Systems Education. Retrieved April 7, 2001, from the WorldWideWeb:http://www.gise.org/JISE/Vol1-5/SYSTE MAN.htm.

Dr.PK.Ragunath, Professor and Head of Department of Bioinformatics(ISO 9001:2000), Sri Ramachandra University , Chennai , India. He has done his PhD in Madras Medical College, Chennai in Patho-informatics of Human Lymphoma. Moreover, he has completed several IT courses and having rich experience in Software Project Management.

Ravimohan Rajmohan received the B.Com degree from Loyola College in 1986 and M.B.A. degree in Business Administration from PSG College of Technology, Coimbatore in 1988. During 1988-1997, he has been working in various sales and marketing positions with various companies in Tamilnadu. From 1997 onwards he switched over to a career in IT in the specialised field of ERP solutions and has worked with various MNC's like HP and Accenture. He has a flair for teaching and has been handling classes in the subject of Hospital Management Information Systems for MBA students. He is also a member of the Board of Studies, in the Department of Bioinformatics, Sri Ramachandra University.

S. Velmourougan is presently working as Scientist at Centre For Reliability (CFR-Chennai), STQC Directorate, Ministry of Information Technology, Govt. of India.ty Analysis, Failure Analysis, and Reliability development/Growth testing. He is a "Certified Ethical Hacker (CEH)" certified by Eccouncil, USA, Qweb Lead assessor, IQNET, ISMS-Lead Auditor, IRCA-UK also he is a "Certified Reliability Professional (CRP)" and "Certified Software Test Manager (CSTM)".