International Journal of Research in Engineering, Science and Management
Volume 7, Issue 2, February 2024
https://www.ijresm.com | ISSN (Online): 2581-5792

Reassessment of Software Development Life Cycle Models

Kusum Lata Dhiman*
Assistant Professor, Department of Computer Science and Engineering, Parul University, Vadodara, India

***Abstract*: In this research paper we explained the various software development models and their advantages and disadvantages. This paper includes all information related to the SDLC models and their methodology and techniques to do project work. This paper mainly surveys the Waterfall Model, iterative waterfall Model, V-Shaped Model, RAD Model, Evolutionary Model, Prototyping Model, Spiral Model, Agile Model, Incremental Model and Build and Fix Model.**

***Keywords*: SDLC Models, RAD Model, methodology, projects.**

1. Waterfall Model

The Waterfall Model was first Process Model to be introduced. It is very simple to understand and use. In a Waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases. The waterfall model is the earliest SDLC approach that was used for software development. In "The Waterfall" approach, the whole process of software development is divided into separate phases. The outcome of one phase acts as the input for the next phase sequentially [2]. The waterfall model is a sequential design process in which progress is seen as flowing steadily downwards (like a waterfall) through the phases of Conception, Feasibility Study, Analysis, Design, Construction, Testing, Production/Implementation and Maintenance [13].

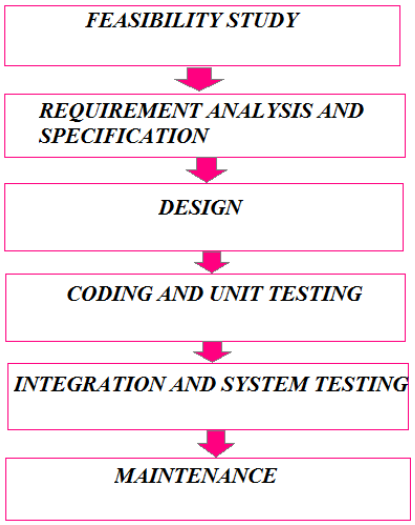

Fig. 1. Waterfall model

- Communication: Establishes expectations of stakeholders
- Planning: Develop a well-defined plan
- Modeling: Develop a model
- Construction: Build actual projects
- Deployment: Delivery of project to the customer and its maintenance

A. Advantages

- Classical model and easy to understand and implement
- Widely used in government projects
- Works well on mature projects and weak team

B. Disadvantages

- It is difficult to estimate time and cost for each phase of the development process.
- Once an application is in the testing stage, it is very difficult to go back and change something that was not well-thought-out in the concept stage.
- Not a good model for complex and object-oriented projects.
- Not suitable for projects where requirements are at a moderate to high risk of changing.

2. Iterative Waterfall Model

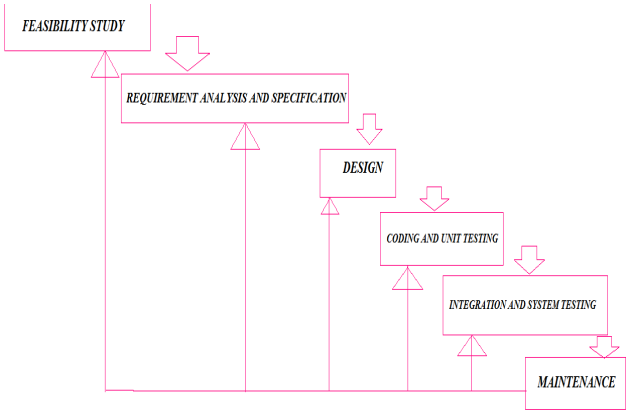

Fig. 2. Iterative waterfall model

The iterative waterfall model provides feedback paths from every phase to its preceding phases, which is the main difference from the classical waterfall model. When errors are

*Corresponding author: kusumdhiman212@gmail.com

detected at some later phase, these feedback paths allow for correcting errors committed by programmers during some phase. It is good to detect errors in the same phase in which they are committed. It reduces the effort and time required to correct the errors [3].

A. Advantages

- Iterative waterfall model feedback path from one phase to its preceding phase allows correcting the errors that are committed and these changes are reflected in the later phases.[2]
- The iterative waterfall model is very simple to understand and use.

B. Disadvantages

- there is no scope for any intermediate delivery. So, customers must wait a long forget the software.[3]
- Projects may suffer from various types of risks. But Iterative waterfall model has no mechanism for risk handling.

3. V-Shaped Model

The V-model is a type of SDLC model where the process executes sequentially in a V-shape. It is also known as the Verification and Validation model. It is based on the association of a testing phase for each corresponding development stage. The next phase starts only after completion of the previous phase i.e. for each development activity, there is a testing activity corresponding to it [11].

Verification**:** It involves a static analysis technique (review) done without executing code. It is the process of evaluation of the product development phase to find whether specified requirements are met [1].

Validation*:* It involves dynamic analysis techniques (functional, non-functional), and testing done by executing code. Validation is the process of evaluating the software after the completion of the development phase to determine whether the software meets the customer's expectations and requirements.

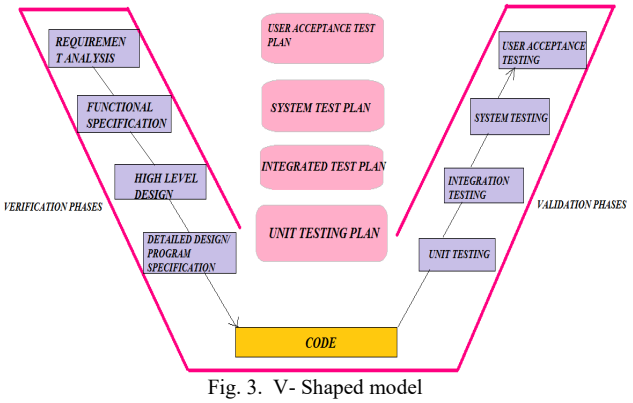

Fig. 3. V- Shaped model

A. Advantages

- This is a highly disciplined model and Phases are completed one at a time [12].
- V-Model is used for small projects where project requirements are clear.
- Simple and easy to understand and use.

B. Disadvantages

- High risk and uncertainty.
- It is not good for complex and object-oriented projects.
- It is not suitable for projects where requirements are not clear and contain a high risk of changing.

4. RAD Model

RAD or Rapid Application Development process is an adoption of the waterfall model; it targets developing software in a short period. It focuses on the input-output source and destination of the information.[10] It emphasizes delivering projects in small pieces; the larger projects are divided into a series of smaller projects. The main feature of the RAD model is that it focuses on the reuse of templates, tools, processes, and code [12].

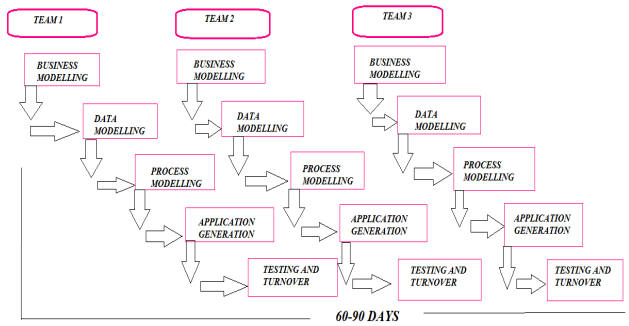

Fig. 4. RAD model

A. Advantages

- Flexible and adaptable to changes.
- It is useful when you must reduce the overall project risk.
- It is adaptable and flexible to changes.
- Due to code generators and code reuse, there is a reduction of manual coding.

A. Disadvantages:

- It cannot be used for smaller projects.
- Not all applications is compatible with RAD.
- When technical risk is high, it is not suitable.
- Requires highly skilled designers or developers.

5. Evolutionary Model

The evolution model divides the development cycle into smaller, “Incremental Waterfall Models” in which users can get access to the product at the end of each cycle. The users provide feedback on the product for the planning stage of the next cycle and the development team responds, often by changing the product, plans, or process [9]. The evolution model is based on the initial implementation will result in the user comments it can be repaired through many versions until an adequate system can be developed. In addition to having separate activities, this model provides feedback to developers [4].

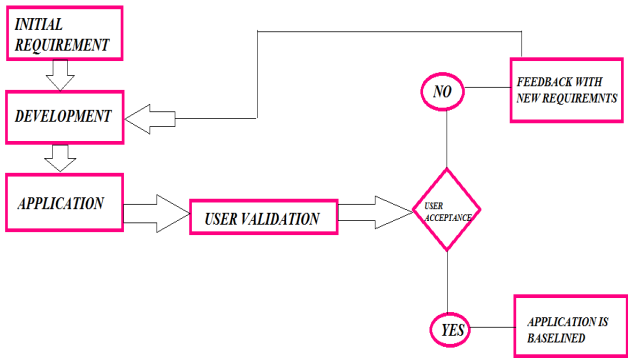

Fig. 5. Evolutionary model

A. Advantages

- Error reduction: The version is tested with the customer which reduces the error thoroughly.
- User satisfaction: The user gets satisfied and he gets the full chance of experimenting partially developed system.
- Business benefit: Successful use of this model can benefit not only business results but marketing and internal operations as well.
- High quality: As you should be satisfied with every version, it produces a high-quality product.

B. Disadvantages

- Uncertain nature of customer needs: A confused user has uncertainty over his requirements, so giving him several versions may change his requirements rapidly.
- Time and Cost: As this model reduces "Time and Cost" but the requirement is not gathered correctly. It will subsequently time, cost, and effort.
- Confusion by several versions: A user might get "confused by several versions of the software. It will affect the final product.

6. Prototyping Model

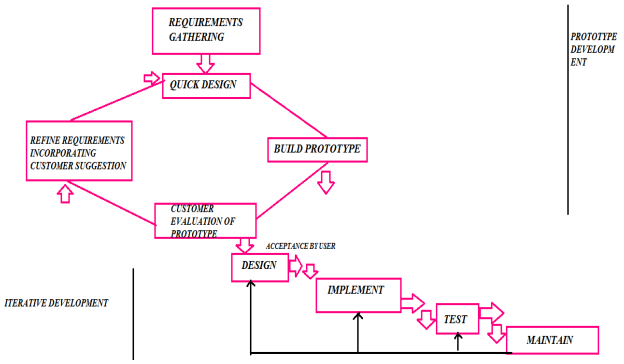

Fig. 6. Prototyping model

The Prototyping Model is a systems development method (SDM) in which a prototype (an early approximation of a final system or product) is built, tested, and then reworked as necessary until an acceptable prototype is finally achieved from which the complete system or product can now be developed [8]. This model works best in scenarios where not all of the project requirements are known in detail ahead of time [9]. It is an iterative, trial-and-error process that takes place between the developers and the users.

A. Advantages

- The customers get to see the partial product early in the life cycle. This ensures a greater level of customer satisfaction and comfort [1].
- New requirements can be easily accommodated as there is scope for refinement.
- Missing functionalities can be easily figured out.
- Errors can be detected much earlier thereby saving a lot of effort and cost, besides enhancing the quality of the software.
- The developed prototype can be reused by the developer for more complicated projects in the future.
- Flexibility in design.

B. Disadvantages

- Costly concerning time as well as money.
- There may be too much variation in requirements each time the prototype is evaluated by the customer.
- Poor Documentation due to continuously changing customer requirements.
- It is very difficult for the developers to accommodate all the changes demanded by the customer.
- Developers in a hurry to build prototypes may end up with sub-optimal solutions.

7. Spiral Model

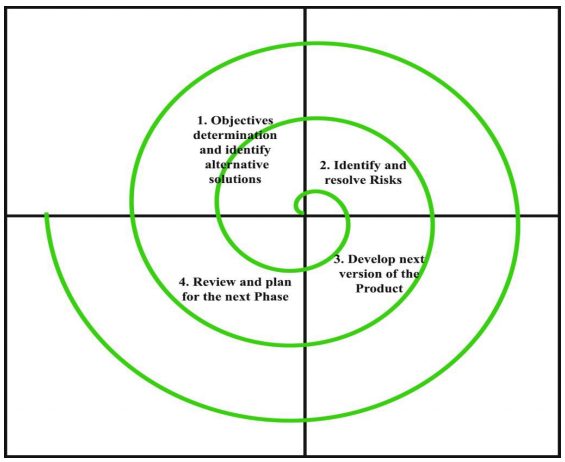

Fig. 7. Spiral model

The spiral model is one of the most important Software Development Life Cycle models, which provides support for Risk Handling [8]. In its diagrammatic representation, it looks like a spiral with many loops. The exact number of loops of the spiral is unknown and can vary from project to project. Each loop of the spiral is called a Phase of the software development process. The exact number of phases needed to develop the product can be varied by the project manager depending upon the project risks.

A. Advantages

- Measuring progress by the amount of completed work.

- Continually seeking excellence
- Harnessing change for competitive advantage
- Simplicity
- Self-organizing team to come out with the best architectures, requirements, and designs.
- Regular adaptation to changing circumstances more effectively.

B. Disadvantages

- Can be a costly model to use.
- Risk analysis requires highly specific expertise.
- The project's success is highly dependent on the risk analysis phase.
- Does not work well for smaller projects.
- It is not suitable for low-risk projects.

8. Agile Model

Agile development methodology attempts to provide many opportunities to assess the direction of a project throughout the development life cycle. Agile methods break tasks into small increments with minimal planning and do not directly involve long-term planning [8]. Iterations are short time frames that typically last from one to four weeks. Each iteration involves a cross-functional team working in all functions: planning, requirements analysis, design, coding, unit testing, and acceptance testing [7]. At the end of the iteration, a working product is demonstrated to stakeholders. This minimizes overall risk and allows the project to adapt to changes quickly.

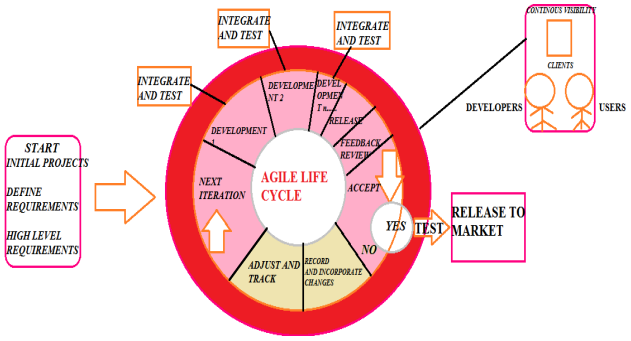

Fig. 8. Agile life cycle model

A. Advantages

- Working software is delivered frequently (weeks rather than months).
- Face-to-face conversation is the best form of communication.
- Close daily cooperation between business people and developers.
- Continuous attention to technical excellence and good design.
- Regular adaptation to changing circumstances.
- Even late changes in requirements are welcomed.
- Customer satisfaction by rapid, continuous delivery of useful software.

B. Disadvantages

- There is a lack of emphasis on necessary designing and documentation.
- The project can easily get taken off track if the customer representative is not clear about what outcome they want [1].
- In the case of some software deliverables, especially the large ones, it is difficult to assess the effort required at the beginning of the software development life cycle.

9. Incremental Model

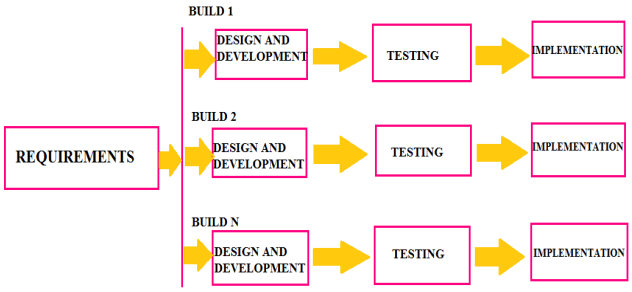

Fig. 9. Incremental model

A. Advantages

- Generates working software quickly and early during the software life cycle.
- This model is more flexible – less costly to change scope and requirements.
- It is easier to test and debug during a smaller iteration.
- In this model customers can respond to each build.
- Lowers initial delivery cost.

B. Disadvantages

- Needs good planning and design.
- Needs a clear and complete definition of the whole system before it can be broken down and built incrementally.
- Total cost is higher than waterfall.

10. Build and Fix Model

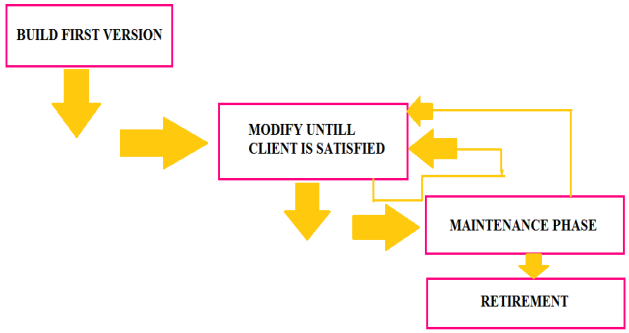

Fig. 10. Build and Fix model

In this most simple model of software development, the product is constructed with minimal requirements, and generally no specifications not any attempt at design, and testing is most often neglected. This is a representation of what

is happening in many software development projects [6].

This process goes on until the user feels that the software can be used productively. However, the lack of design requirements and repeated modifications result in a loss of acceptability of software. Thus, software engineers are strongly discouraged from using this development approach [5].

A. Advantages

- Cost efficient for very small projects of limited complexity

B. Disadvantages

- Unsatisfying approach for products of reasonable size.
- Cost is higher for larger projects
- Product will not be delivered on time most of the time.
- Often results in a product of overall low quality.
- No documentation is produced.
- Maintenance can be extremely difficult without specification and design document

11. B Shaped Model

It is used to increase the features of the waterfall model. B Model is devised by Birrell. It is represented by a vertical line and leads to the maintenance cycle at the bottom. Each step is important and there is no interdependency on other stages. In B shaped model we use a double arrow between all the phases. The b model starts with inception [4]. It is suitable in category 1 software life cycle. It is used to ensure that constant improvement of software and systems can become the part of development stage.

A. Advantages

- B shaped model is flexible in its interaction with system requirements.
- We can make one change by moving through the maintenance cycle at the same time.
- We can take the possibility while or by using reverse to the previous phase if any error occurs.

B. Disadvantages

- Changes are slow in the shaped Model.
- For modification purposes we have another design and analysis phase we have new requirements associated with changes which lead to the understanding and investigation again.

12. Wheel and Spoke Model

Wheel and Spoke models are a bottom-up approach. It is beneficial in the design and prototyping stage. Wheel And Spoke Model is a development model which parallelly sequential in nature [3]. It is essentially a modification of the spiral model that is designed to work with smaller initial teams, which then scale upwards and build value faster.

- The wheel and spoke are best used in an environment where several projects have a common architecture or feature set that can be abstracted by an API.

A. Advantages

- Less risk at the initial stage: If one is developing a small-scale prototype as compared to a full-blown development effort, then several programmers are needed at the start.
- The core team developing the prototype gains experience from each successful program that adapts the prototype and sees an increasing number of bug fixes and a general rise in code quality.

13. Unified Process Model

The Unified Process is based on the enlargement and refinement of a system through many iterations, with cyclic feedback and adaptation [6]. The system is developed incrementally over time, and no. of iterations and thus this approach is also known as incremental and iterative software development. The iterations are spread over four phases where each phase consists of one or more iterations.

The steps described by Unified Process are as follows:

- Business modeling
- Requirements
- Analysis and design
- Implementation
- Test
- Deployment

A. Phases in Unified Process

Unified process is dividing the development process into five phases that are mentioned below:

- Inception
- Elaboration
- Conception
- Transition
- Production

14. JAD (Joint Application Development)

This methodology involving the client and end-user in the development process results in a project that is more aligned and suitable to their needs [5].

A. Advantages

- Improved Delivery Time: The time needed to develop a product by using the JAD model is short and more effective than that of other basic models.
- Cost Reduction: Efficiently finding out the requirements and facts with business executives and stakeholders will make a low effort to create the system.
- Better Understanding: The team members who can professionally interact with each other better usually helps to understand product development easier.
- Improved Quality: Since all the key decision-makers and stakeholders of the project are involved in the implementation of the project, so there is the less chance of error, and hence the product quality becomes effective and more accurate [8].

- Resolve difficulties more easily and invent error free product
- Lower Risk
- Faster Progress

B. Disadvantages

- It requires specific commitment.
- Difficult to achieve a goal because there are lots of options within a team

15. XP (Xtreme Programming)

Extreme programming (XP) is the most important software development part of Agile models. It is used to enhance software quality and responsive to customer requirements. The extreme programming model recommends taking the best practices that have worked well in the past in program development projects to extreme levels [1].

It includes the following things that are mentioned below:

- Code review
- Testing
- Incremental Development
- Simplicity
- Design
- Integration Testing

A. Applications

- The XP model is very helpful in small projects consisting of small teams as face-to-face meetings are beneficial to achieve.
- This type of project faces changing requirements rapidly and technical issues. So, the XP model is used to complete this type of project.

16. Scrum

In the Scrum framework, agile methodology is used. In the scrum model, iterative and incremental process is used. The project is divided into several phases, each of which results in a ready-to-use product.

In the scrum model, we use some roles that are explained below:

- The product owner takes care of the user's requirements
- The Scrum master coordinates the complete development methodology.
- The Scrum team create the product. Its main motives are programming and developing, analysis, testing, etc.

17. Big Bang Model

This Big Bang model is a software development model that does not follow any kind of protocol, and hence its planning needs are minimum This model holds no planning or analysis, so this model involves many risks than other SDLC models.[2]

Big Bang model can be partitioned into the following terms:

- In this development model, modules get integrated as all the individual modules are entirely built and are not integrated separately or individually.
- Each module is tested separately to look for error or defects.
- In case there is any bug or error in any module, the module is disintegrated, and the root cause of the problem is found.

A. Advantages

- This is quite a simple model.
- Implementing this model makes managing tasks easier.
- It required no planning.
- Developers have large flexibility in developing the entire product.
- Resources necessary for developing the product is very less.

B. Disadvantages

- This model is not suitable for big or complex projects.
- Very meagre model for building long and constant projects.
- Holds very high risk.

18. Lean Development

The Lean model for software development is followed by "lean" manufacturing work and principles. It is also called LSD (Lean Software Development). It is based on the agile concept. The lean approach is also called MVP (Minimum Viable Product). Seven lean principles are mentioned below:

- eliminate waste
- amplify learning
- decide as late as possible
- deliver as fast as possible
- empower the team
- build in integrity
- see the whole.

A. Advantages

- Streamlined concept is helpful in functionality to be delivered in short time.
- It deletes unwanted activity, and as a result, we can reduce the cost.
- It encourages the development team to make decisions, which can also boost morale.

B. Disadvantages

- It depends on the team involved, making it not as perfect as other frameworks
- Depends on good quality documentation, and failure to do so can result in development mistakes

19. Conclusion

In this paper, software development life cycle and SDLC models are defined. Every model has its own advantages and disadvantages and every model came into existence to cope with the problems of existing model of that time. In now a day's

waterfall and spiral are the most used in the software development process and the other models are used according to the requirements of the software products. The software developers use the models according to the size of the software that is to be developed. SDLC is helpful for users to get a high-quality product within time and budget. So that the user can select the best-suited model as per his requirements.

References

[1] Barry Boehm, "Spiral Development: Experience, Principles, and Refinements", edited by Wilfred J. Hansen, 2000.

[2] Roger Pressman, Software Engineering: A Practitioner's Approach, Sixth Edition, McGraw-Hill Publication.

[3] K. K. Aggarwal, Yogesh Singh Software Engineering 3rd Edition.

[4] Karlm, "Software Lifecycle Models', KTH, 2006.

[5] Vanshika Rastogi, "Software Development Life Cycle Models-Comparison, Consequences," International Journal of Computer Science and Information Technologies, vol. 6(1), 2015, 168-172.

[6] Sanjana Taya, "Comparative Analysis of Software Development Life," 2011.

[7] C. Senthil Murugan and S. Prakasam, "A Literal Review of Software Quality Assurance", International Journal of Computer Applications, vol. 78, no. 8, September 2013.

[8] Anu Gupta, "Quality Assurance and Its Standards: Importance in Various SDLC Models", International Journal of Advanced research in Computer Science and Management studies, vol. 2, no. 12, December 2014.

[9] Chaitali Roy and Mousumi Saha, "The Realm of Software Quality Assurance", International Journal of innovations and Engineering and Technology.

[10] Shubham Dwivedi, "Software Development Life Cycle Models - A Comparative analysis", International Journal of Advanced Research in Computer applications.

[11] Nayan B. Ruparelia, "Software Development Life Cycle Models," vol. 35, no. 3, May 2010.

[12] Jacobson I., Booch G. & Rumbaugh J. (1999): The unified software development process; Addison-Wesley, Reading, Massachusetts. Raymond, Eric (2001): Cathedral and the Bazaar, 1st Edition.

[13] Mooz, H and Forsberg, K., A visual explanation of the development methods and strategies including the waterfall, spiral, vee, vee+, and vee++ models, pp. 4-6, 2001.