Chapter 9

A Comparative Analysis of BRIDGE and Some Other Well Known Software Development Life Cycle Models

9.1 Introduction

The rapid development in the hardware technology has made modern processors very efficient and powerful. Hence, the expectations from the software have gone to zenith. But the complexity of the modern software are much complex as compare to those of earlier. Development cost, time and quality of the modern software are in crisis. A software (SW) project, irrespective of its size, goes through certain defined stages, which together, are known as the Software Development Life Cycle (SDLC). Life Cycle refers to the different phases involved starting from the project initiation to project retirement. For better understanding and implementation of the various phases of software development, different software development models have been developed and proposed so far. There are several Software Development Life Cycle (SDLC) Models i.e. Classical Waterfall, Spiral, Prototype, V-Model, evolutionary model etc. All these SDLC models have several advantages as well as some limitations. It is pre established that different SDLC models have different capabilities and limitations. Hence, selecting suitable SDLC model for any project is quite crucial as not all process models are good for any type of project. Hence, analyzing the different SDLC model is significant and helps one to select the appropriate model for a project. Recently a few more new process models are

Based on the publication in the "*International Journal of Computer Science & Engineering Technology(IJCSET)*", ISSN: 2229–3345, 5(3), 196–202, 2014.

proposed with the well known traditional models to accommodate the new industrial needs.

9.2 Software Development Approach, Process and Process Model

It is really tough to draw a sharp line between software development approaches and SDLC process models. In many literature of software engineering, these terms are used interchangeably or confusedly. So, before we begin the details discussion of the topic, let us somehow draw the boundary line between software development approaches and SDLC process models. Defining these two terms are beyond the scope in this context. Here we just try to explain both only to establish the differences from our point of view. SW development process or simply process typically defines the set steps to be carried out during the development of the system. SW development life cycle (SDLC) is the time from the concept development to the product retirement i.e. the time of SW process. SW development life cycle (SDLC) process model typically depicts the fashions in which the SW process to be carried out i.e. which steps/phases to be done before or after another step/phase. In general all the process models do cover all distinct phases defined by SW process, but in different manner or sequence- which makes one process model differ from the other. In other words, a software development process model is an approach to the Software Development Life Cycle (SDLC) that describes the sequence of steps to be followed while developing software projects (7, 8). We consider Agile, incremental, extreme and iterative as approach or philosophy to software development rather than as process model which can be implemented following other process models i.e. Waterfall, RAD, Spiral, Prototyping or alike.

9.3 Parameter Selection for Comparative Analysis

Below we enlist and discuss briefly the parameters alphabetically those we have considered for the purpose of comparative analysis:

- **Adaptability:** This is the ability to react to operational changes as the project is developed. Change orders are easily assimilated without undue project delay and cost increases.

- **Budget:** Budget remains one of the most significant crisis for software development projects. Some process model like Spiral and Prototyping increases the project cost as compared to others. Hence a SDLC process model has great impact on software development cost or budget.

- **Changes Incorporation:** Change is unavoidable in software development. Managing change is a critical component of any SDLC model. Change Management and SLDC are not mutually exclusive. Change management occurs throughout the development life cycles which need to be incorporated in the system development.

- **Complexity of the Process Model:** Different SDLC process model have varying degree of complexity. Some are easy to use and implement while others are not.

- **Documentation:** Documentation of software development process is very important but time consuming and expensive. To reduce development time and cost, agile philosophy recommends less document which remains one of the most important critic of agile philosophy. Documentation plays vital roles in system development, implementation, maintenance, and project management. But, not all process models facilitate and recommend adequate and sufficient documentation.

- **Expertise Required:** Although some process models are better over others, but need some kind of expertise during its use and implementations in various phases at varying degrees. To avail the advantages of some process models i.e. Spiral, BRIDGE(62) and others which supports reusability - the software engineers required certain level of expertise.

- **Flexibility:** The freedom afforded to software architects, analysts or developers to tailor the software development process according to business needs and project characteristics is a crucial factor in successful project completion. The software development organization often can benefit from introducing flexibility into their software development methodologies (170).

- **Guarantee of Success:** This is really crucial to measure whether any process model will guarantee success or not. If so, up to what extent wills the process model guarantees the success is a big question need to be explored. As the project success depends upon many other constraints and parameters, but given the other parameters as desired, project success may vary from following one SDLC model to another.

- **Integrity & Security:** Including security early in the system development life cycle (SDLC) will usually result in less expensive and more effective security than adding it to an operational system. To be most effective, information security must be integrated into the SDLC from its inception (171).

- **Maintenance:** Systems are dynamic and the model offers the ability to produce a final project that is inherently designed for maintenance. This includes such items as cumulative documentation.

- **Management Control:** Management will have the ability to redirect and if necessary redefine the project once it is begun. A key phrase is 'incremental management control', with each step under tight management control. Management control has great impact on project success.

- **Overlapping Phases:** Each step of the project is to be completed before another is begun. Project modules are distinct and easily identifiable.

- **Parallel development:** Parallel development support, if possible to employ may increase productivity and reduce development time while optimally utilizing the resources.

- **Productivity:** The SDLC must ensure that the expected return on investment (ROI) for each project is well defined. The SDLC must minimize the unnecessary rework. It must be designed in such a way as to take maximum advantage of the computer assisted software engineering (CASE) tools (119). At the same time the SDLC must utilize the resources most effectively and efficiently to improve the productivity.

- **Progress Measurement:** Progress measurement allows development team as well as the project management team to determine how well tasks were estimated, how well they were defined, and whether items are completed on-time and within budget. Any SDLC process model should provide the facility to measure the progress during the system development.

- **Quality Control:** Each module of the project can be thoroughly tested before another module is begun. Project requirements are measured against actual results. Milestones and deliverables can be used for each step of the project.

- **Requirements Specification:** Depending on the project nature, the requirement may be identified at the very beginning of the project development or may be discovered during the development process. But, not all process model supports requirement discovery over the development process. Hence, requirement specification may be static or may be dynamic. Any SDLC process model should take into account the issue of requirement specification.

- **Requirements Understanding:** Some process model needs the requirement must be well understood before the development process stated, while other may allow understanding the requirements over the development process. One may start with the initial understanding of the requirement and during the development the requirement understanding increases gradually.

- **Reusability:** Reusability is one of the most significant and efficient attribute of any SDLC process model these days. Reusability helps to improve system productivity, reduce cost and system delivery on-time. The degree of reusability support may vary from one process model to another.

- **Risk Involvement:** The risk involvement may vary from one model to another depending on the nature of requirement understanding capability support by the process model. Apart from this, there may be several other sources of risks involvement.

- **Risk Management:** Different types of risks are implicit part of any project. Levels of risk are identifiable and assessment strategies available. Strategies are proved for overall and unit risks.

- **Simplicity:** Any model need is easy to understand and to implement. Simplicity of any process model reduces the burden of expertise and improves productivity while reduce development cost and project risk.

- **System Delivery:** The system may be delivered either partially as individual operational module wise or as the complete system with full functionality at once.

- **Time:** Time is actually referred to as Time Horizon because we are interested in knowing the projected completion of the project. The development time may vary from one process to another.

- **Understandability and Implementation:** Different process model may need varying level of expertise. Simple and better understandable process model are always easy to implement.
- **User Involvement:** Any model lends itself to strong and constant end user involvement. This includes project design as well as interaction during all phases of project development.

9.4 Comparative Analysis

The different process models are discussed in section 2.5 and BRIDGE process model is discussed earlier in Chapter 6. The comparisons among different SDLC models in respect to the features discussed in the previous section are illustrated in Table 9.1 (105, 172, 173, 174, 175, 176, 177, 178, 179, 180). From the above comparative analysis, it is established that the BRIDGE process model possesses many suitable features in comparison to the other process model.

9.5 Conclusion

There exist several well known SDLC process models. One process model has different comparative advantages from the others in many respects. But no process model is just good for any type of project. So it is not blindly recommended to choose any process model for any project! The above comparative study shows that overall the BRIDGE process model has several competitive advantages over the other existing well known process models. As BRIDGE model has excellent adaptability, supports process tailoring and other attributes, we recommend this SDLC process model to be used for any types of software development projects.

Table 9.1: Comparison of Different SDLC Process Model

Models / Parameters	BRIDGE	Waterfall	Prototype	Evolutionary	Spiral	RAD	V-Shape
Adaptable	Excellent	Limited	Good	Good	Excellent	Limited	Limited
Budget	Low	Low	High	Low	High	Low	Low
Changes Incorporate	Easy	Impossible / Difficult	Easy	Easy	Easy	Easy	Difficult
Complexity	Medium	Simple	Moderate	Complex	Complex	Medium	Simple
Documentation	Yes	Strong	Weak	Moderate	Moderate	Poor/ Limited	Yes
Expertise Required	Medium	Low	Low	Low	High	Medium	Medium
Flexibility	Flexible	Inflexible	Highly Flexible	Highly Flexible	Flexible	High	Rigid
Guarantee of Success	High	Less	Good	Good	High	Good	High
Integrity and Security	High	Vital	Weak	Weak	High	Vital	Limited
Maintenance	Easily Maintained	Least Glamorous	Routine Maintenance	May be overlooked	Typical	Easily maintained	Lest
Management Control	Yes, Dedicated	No	No	Weak	Moderate	Weak	Weak
Overlapping Phases	May be	No	Yes	Yes	Yes	No	No
Parallel Development	Supported	No	No	Limited	Limited	No	Limited
Productivity	Highest	High	Improved	Improved	High	Improved	Improved
Progress Measurement	Measurable	Easily Monitored	Measurable	Measurable	Measurable	Measurable	Measurable
Quality Control	Very Good	Poor	Moderate	Good	Good	Adequate	Moderate
Requirements Specification	Adaptable/ Dynamic	At the Beginning	Frequently Changed	Frequently Changed	At the Beginning	Time-box Release	At the Beginning
Reusability	Excellent	Limited	Poor	Poor	Moderate	Moderate	Moderate
Risk Involvement	Low	High	Low	Moderate	Low	Little	Low
Risk Management	Highly Supporter	Not Considered	Moderate	Good	Highly Supporter	Poor	No
Simplicity	Intermediate	Simple	Simple	Intermediate	Intermediate	Very Simple	Simple
System Delivery	Early and periodic partial operational system	At the end of the system development	At the end of the system development	Early and periodic partial operational system	At the end of the system development	At the end of the system development	At the end of the system development
Time	Shortest	Short	Long	Long	Long	Short	Short
Understandability and Implementation	Moderate	Easy	Easy	Moderate	Complex	Moderate	Easy
User Involvement	Throughout Process	At the beginning	High/Up to design phases	Throughout Process	High	Throughout Process	At the beginning