**Software Development Lifecycle Models**

**Introduction**

Software development lifecycle models are a series of distinct stages in software engineering that divide the work into distinct categories. They emerged in the 1960s

In this research paper the following software development lifecycle models will be discussed:

1. Classic
2. Incremental
3. Evolutionary
4. Spiral

A brief description of each strengths and weaknesses, the types of projects that are suitable for each, the factors for successful application and a graphical representation of each will be provided.

|  |  |
| --- | --- |
| 1. Classic (Waterfall) | |
| **Description** | |
| The waterfall model was one of the first project management methodologies that came into being. Originating in manufacturing and construction, the model also inherited the highly structured approach and rigidity of these industries. It is broken down into six stages and each individual stage must be completed before moving on to the next. The six stages are:   1. Requirement Analysis 2. Design 3. Implementation 4. Verification 5. Integration 6. Maintenance   The Waterfall method suits large scale fixed requirement development projects. | |
| **Applicable Projects** | |
| Mission critical projects:  Payment Gateways  E-commerce supply chains  Anti-virus software  Flight Control  Traditionally used in construction and manufacturing | |
| **Critical factors for success** | |
| Each stage must be completed before the next phase begins.  The requirements must be fully understood.  The plan must be completed before any development work begins. | |
|  | |
| **Advantages** | **Disadvantages** |
| Adapts to shifting teams  Forces structured organisation  Allows for early design changes  Suited for milestone-focused development  Verification at each stage ensures early detection of errors/misunderstanding. | Non-adaptive design constraints  Ignores mid-process user/client feedback  Delayed testing period  Assumes that the requirements of a system can be frozen.  Very difficult to go back to any stage after it finished. |
| **Useful Links** | |
| <https://melsatar.blog/2012/03/15/software-development-life-cycle-models-and-methodologies/>  <http://www.diva-portal.org/smash/get/diva2:3207/FULLTEXT02.pdf> | |
| 2. Incremental or Iterative | |
|  | |
| **Description** | |
| Incremental Model is a process of software development where requirements are broken down into multiple standalone modules of software development cycle. Each iteration passes through the requirements, design, coding and testing phases. | |
| **Applicable Projects** | |
| IBM Project Management  NASA Space Shuttle software programs  US Department of Defence | |
| **Critical factors for success** | |
| User/Client feedback is essential  Know when MVP is reached | |
| (Courtesy of Microsoft MSF) | |
| **Advantages** | **Disadvantages** |
| Generates working software quickly and early during the software life cycle.  More flexible – less costly to change scope and requirements.  Easier to test and debug during a smaller iteration.  Easier to manage risk because risky pieces are identified and handled during its iteration.  Each iteration is an easily managed milestone. | Each phase of an iteration is rigid and do not overlap each other.  Problems may arise pertaining to system architecture because not all requirements are gathered up front for the entire software life cycle. |
| **Useful Links** | |
| <https://www.testingexcellence.com/incremental-model/>  <https://www.projectsmart.co.uk/which-life-cycle-is-best-for-your-project.php> | |

|  |  |
| --- | --- |
| 3. Evolutionary | |
| **Description** | |
| Evolutionary model is a combination of iterative and incremental approach to software development. Delivering your system in a big bang release, delivering it in incremental process over time is the action done in this model. Some initial requirements and architecture envisioning need to be done. | |
| **Applicable Projects** | |
| UML | |
| **Critical factors for success** | |
| Ideal for software products that have their feature sets redefined during development because of user feedback and other factors. The Evolutionary development model divides the development cycle into smaller, incremental waterfall models in which users are able to get access to the product at the end of each cycle.  Feedback is provided by the users on the product for the planning stage of the next cycle and the development team responds, often by changing the product, plans, or process. | |
|  | |
| **Advantages** | **Disadvantages** |
| Continuous development | Never finished |
| **Useful Links** | |
| <https://www.tutorialride.com/software-engineering/evolutionary-process-models-in-software-engineering.htm> | |

|  |  |
| --- | --- |
| 4. Spiral | |
| **Description** | |
| One of the most flexible SDLC methodologies, the Spiral model takes a cue from the Iterative model and its repetition; the project passes through four phases (planning, risk analysis, engineering and evaluation) over and over in a “spiral” until completed, allowing for multiple rounds of refinement.    The Spiral model is typically used for large projects. It enables development teams to build a highly customized product and incorporate user feedback early on in the project. Another benefit of this SDLC model is risk management. Each iteration starts by looking ahead to potential risks and figuring out how best to avoid or mitigate them. | |
| **Applicable Projects** | |
| Medium to high risk projects  When the user is not sure of their needs  Where the requirements are complex and heavy | |
| **Critical factors for success** | |
| When project is large  When releases are required to be frequent  When creation of a prototype is applicable  When risk and costs evaluation is important  For medium to high-risk projects  When requirements are unclear and complex  When changes may require at any time  When long term project commitment is not feasible due to changes in economic priorities | |
| (courtesy of guru99.com)  What is Spiral Model? When to Use? Advantages & Disadvantages | |
| **Advantages** | **Disadvantages** |
| Additional functionality or changes can be done at a later stage  Cost estimation becomes easy as the prototype building is done in small fragments  Continuous or repeated development helps in risk management  Development is fast and features are added in a systematic way  There is always a space for customer feedback | Risk of not meeting the schedule or budget  It works best for large projects only also demands risk assessment expertise  For its smooth operation spiral model protocol needs to be followed strictly  Documentation is more as it has intermediate phases  It is not advisable for smaller project, it might cost them a lot |
| **Useful Links** | |
| <https://searchsoftwarequality.techtarget.com/definition/spiral-model> | |