ASSIGNMENT 2

Software Methodologies

1.

- Classical Waterfall Model
  - SDLC model very that is very simple and easy to understand and basis for other models. In this model, the flow is, input of the next phase is the output of the previous phase.
- Iterative Waterfall Model
  - in this model provides feedback path from phase to its preceding phases, when errors detected at some later phase the feedback paths allow correcting or reworked it.
- Incremental Model
  - requirements of software are first broken down into several modules that can incrementally constructed and developed. Development team first undertakes to develop core features of the system, once fully developed, then these are Regine to increase levels of capabilities by adding new functions in successive version.
- Prototyping Model
  - a prototype of the end product is first developed, tested and refined as per customer feedback repeatedly until a final acceptable prototype is achieved.

- Spiral Mode
    - each loop of the spiral is called phase of the software development process and it has capability to handle risks.
- Rapid Application Development (RAD) Model
    - this model prioritizes rapid prototyping and quick feedback over long-drawn-out development and testing cycles.
- Agile Model
    - developed in incremental, rapid cycles and small incremental releases with each release building on previous functionality more so tested to ensure software quality is maintained.
- Scrum Model
    - works very well for innovative and complex product development projects.

2.

Software Methodology	Strengths	Drawbacks/Weaknesses
Classical Waterfall Model	• easy to understand • phases are processed one at a time • works well for smaller projects	• no feedback paths • no overlapping of phases • difficult to accommodate change request
Iterative Waterfall Model	• has feedback path • simple • cost-effective • well-organized	• difficult to incorporate change request • incremental delivery not supported • overlapping of phases not supported • risk handling not

		supported
Incremental Model	• Flexible and less expensive to change requirement and scope. • easy to identify errors • customers can respond to each building	• requires good planning designing • problems may cause due to system architecture • each iteration phase is rigid and no overlapping
Prototyping Model	• flexible in design • easy to detect errors • can find missing functionality • has scope refinement	• costly • poor documentation (continuous changing customer requirements) • rush delivery of products (customers' demand) • May increase system complexity
Spiral Model	• software produced early in SLC • flexibility in requirement • capable of handling risks • good for large and complex projects	• not suitable for small projects • expensive • process is complex • dependable on Risk Analysis • requires highly specific expertise
Rapid Application Development	• requirements can be change at any time • encourages and priorities customer feedback • reviews are quick • development time is reduced • more productivity with	• need strong team collaboration • cannot work large teams • need highly skilled developers • need user requirement throughout the cycle • suitable for projects

	fewer people	with small time
Agile Model	• people and interactions are emphasized • working software is delivered frequently • regular adaptation to changing circumstances • late changes in requirement are welcome	• lack of emphasis on designing and documentation • only senior programmers are capable of taking decisions required
Scrum Model	• complete project deliverables quickly and effectively • effective use of time and money • large projects divided into easily manageable sprints	• often leads to scope creep • chances of project failure are high • adopting framework in large teams is challenging