SWE201c

**Agile:**

3.

1. **Requirement characteristics:**

* **Reliability**:

With all the reliability, this project are suitable for the Agile model because:

* Instead of relying solely on documentation, functioning software demos are regarded as the greatest way to communicate with consumers in the working software stage.
* Because the requirements cannot be fully acquired at the outset of the project for a variety of reasons, customer collaboration is critical to obtaining accurate product specifications.
* The final stage of Agile model development, Responding to change, emphasises rapid response to change and continual improvement.
* **Types and number of requirements:**
* The most effective means of communication is face-to-face. Co-operation on a daily basis between business people and programmers.
* Any demand adjustments, no matter how last-minute, will be accommodated. Maintaining a focus on both technical prowess and aesthetic appeal. Adaptation to changing conditions on a regular basis.
* **How often the requirements can change:**
* Agile models' flexibility allows for rapid shifts in needs. A realistic approach to software development, agile approaches encourage teamwork and cross-training, as well as rapid creation and demonstration of functionality, because it can handle both static and dynamic requirements,
* The agile methodology is able to produce early iterations of working solutions. A good model is chosen by the administration when dealing with situations that are both dynamic and manageable.

* **Can the requirements be defined at an early stage**
* As a result of its limited rules and ease of use, the Agile model allows for continuous development and delivery within an overall planned context. Agile is superior to Waterfall or V-model because it gives developers more freedom.

1. **Development team:**

* **Team size:**
* **Level of understanding of user requirements by the developers:**
* Due to the fact that Agile is a more team-based model than Waterfall, it is ideal for improving cross-training and interdepartmental cooperation and, as a result, fits the team well. Decisions that have a significant impact on the company's future are still made by the team lead.

1. **User involvement in the project: (Small/Average/Large):**

* Throughout the development process, customers should be actively involved. It is their job to provide, prioritise, and assess the new system needs and iterations.
* It's not uncommon for the project's requirements to shift on a regular basis. If a customer is eager to meet with a software development team at any time, the highly qualified and experienced Agile team is always available. This Agile project necessitates a small team and therefore a modest project.

1. **Conclusion**:

* In light of the foregoing evaluation, I feel that the Agile model is the most appropriate approach in this situation due to the frequent supply of requirements. Since the requirements system is flexible and subject to change at any time, it is appropriate to deploy the system first rather than developing it in stages.

**V-model:**

1. **Requirement characteristics:**

* **Reliability:**

With all the reliability, this project are suitable for the V-model because:

* The Verification and Validation model (V-model) is the name given to this model. The V-Shaped life cycle follows a sequential course, much like the waterfall model. Before moving on to the next level, each one must be completed.
* The product will be tested concurrently with the development of a new version.
* Waterfall was the original inspiration for the V-model, a software development methodology. Requirements, specifications, design, and implementation are all part of the process, which includes a verification and validation step. This is done by unit testing, integration testing, system testing, and finally acceptance testing, which ensures that the requirements have been met.

* **Types and number of requirements:**
* In the V-model, the project team identifies the high-level and detailed design phases of the project based on the project's needs. The requirements get more and more specific as each level is accomplished.
* **How often the requirements can change**
* The V-shaped architecture is best suited for small to medium projects with well-defined and fixed requirements. It's best to go with a V-Shaped design if you have the necessary technological resources and expertise.
* **Can the requirements be defined at an early stage**
* Activities such as test planning and design take place long before code is written. This is a time-saver. As a result, the waterfall paradigm has a lower success rate.

1. **Development team:**

* **Team size:**
* **Level of understanding of user requirements by the developers:**
* Unit tests, integration tests, system tests, and acceptance tests are all included in the V-Model. The unit and integration tests ensure that the system's design is implemented in the code through the use of automated testing.
* Customer expectations are met through system and acceptability testing. Various components of the software are tested at various levels, each of which is performed independently of the others. To begin testing a new level, the V-model requires that the preceding level must be finished first.

1. **User involvement in the project: (Small/Average/Large)**
2. **Conclusion:**

* To summarise, I believe that the V-model is best suited to this situation, as the criteria are well defined and unlikely to alter. Because of the limited size of the group, each member should concentrate on a single stage at a time. Development in stages is unnecessary because the system's requirements don't specify which parts of the system should be deployed first.

**Waterfall**:

1. **Requirement characteristics:**

* **Reliability:**

With all the reliability, this project are suitable for the Waterfall because:

* Process Models were first introduced in the Waterfall Model. The term "linear-sequential life cycle model" is also used to describe this concept. It's a breeze to learn and master.
* Using a waterfall paradigm, each stage must have been finished in full before the next stage can be started. Projects that are less than a year old and have no ambiguous requirements are ideal candidates for this paradigm of software development.
* It is only appropriate to apply this paradigm if the needs are well-known, clearly defined, and unchanging.
* **Types and number of requirements:**
* The business analyst gathers the requirements and the team analyzes them in this phase. During this stage, requirements are written down and more explanations can be obtained.
* The requirements are documented by the Business Analysts once they have spoken with the customer. The project team requires answers to the following questions, which were not included in the requirements paper, after going over and analysing the requirements.
* **How often the requirements can change:**
* This is due to the stringent regulations and standards that must be adhered to.
* As a result, the requirements for these projects are well-known in advance, and contracts stipulate precisely what must be delivered when the project is completed.
* Design modifications can be implemented earlier in the Waterfall approach, which is advantageous because later design changes are difficult to implement. Easy.
* In order to make this change, there is no need for any code or implementation. - Appropriate for projects with a tight deadline: The Waterfall model's sequential nature lends itself nicely to projects involving teams and organisations that thrive when given deadlines.
* Team members will be able to readily grasp and follow the timeline if it is broken down into particular times. Furthermore, the researcher's job is made easier by having a timeline for the entire process and by establishing a few particular deadlines or milestones for each stage.
* **Can the requirements be defined at an early stage:**
* As a result of the model's rigidity, it's straightforward to keep track of the project's progress.
* Phases are done sequentially under this model. Do not mix phases.
* For smaller projects with well-defined and well-understood needs, the waterfall paradigm works well.

1. **Development team:**

* **Team size:**
* **Level of understanding of user requirements by the developers:**
* The project's high- and low-level software architecture is the responsibility of the architect and other senior members of the team. To ensure that the banking application is always accessible, it has been decided that the system should have redundant backup and failover capabilities. High-level and low-level design documents are drawn out by the architect.

1. **User involvement in the project: (Small/Average/Large)**
2. **Conclusion:**

* To summarise, I believe that a waterfall approach would be preferable because the project's requirements are well defined and unlikely to change. Waterfall was also utilised in banking, healthcare, nuclear facilities, space shuttles and many other areas. Development in stages is unnecessary because the system's requirements don't specify which parts of the system should be deployed first.