R1: Process Assignment

1. **What do you think are the top three best ideas of Agile software development, and why?**

One of the strongest ideas of Agile software development is iterative and incremental delivery. Agile focuses on delivering work in small, manageable pieces rather than waiting until the end of a project to release a finished product. This approach allows teams to receive early feedback and correct issues before they become costly or complex. Iterative delivery reduces risk and improves quality by allowing continuous refinement throughout the development process (Atlassian, n.d.; Martin, n.d.).

Another key idea of Agile is customer and stakeholder collaboration. Agile prioritizes ongoing communication with stakeholders to ensure that development remains aligned with user needs. Rather than relying solely on initial requirements, Agile teams actively seek feedback during each iteration. This practice helps reduce wasted effort and improves overall satisfaction because the final product reflects real expectations (Asana, n.d.; Microsoft, n.d.).

A third important Agile idea is embracing change instead of resisting it. Agile frameworks are designed to accommodate changing requirements, even late in the project lifecycle. This flexibility is critical in software development environments where technologies, user needs, and priorities frequently evolve. Agile views change as an opportunity to improve value rather than as a disruption (Martin, n.d.; Microsoft, n.d.).

2. **What do you think are three Agile practices that humans find the hardest to do well, and why?**

One Agile practice that many people struggle with is self-organizing teams. Agile expects teams to manage their own work, distribute responsibilities, and make collective decisions. This can be difficult for individuals who are accustomed to traditional hierarchical management structures. Effective self-organization requires trust, accountability, and strong interpersonal skills, which take time and experience to develop (IEEE, n.d.; Atlassian, n.d.).

Another challenging Agile practice is continuous communication and feedback. Agile relies on frequent meetings such as daily stand-ups, sprint reviews, and retrospectives. While these practices promote transparency, they can be uncomfortable for individuals who are hesitant to speak openly or provide honest feedback. Ineffective communication can reduce the benefits of Agile and create misunderstandings within the team (Atlassian, n.d.; Asana, n.d.).

A third difficult practice is maintaining a sustainable pace of work. Agile emphasizes consistent productivity without burnout, but teams often overestimate how much they can complete in a sprint. Overcommitment can lead to stress, missed deadlines, and reduced morale. Learning to balance ambition with realistic planning is a skill that many teams find difficult to master (Asana, n.d.; Microsoft, n.d.).

3. **What parts of Agile (at least two) would work well for a course project with 4-5 team members, and what parts would not work well (at least two)?**

For a course project with 4–5 team members, short iterations or sprints would work very well. Weekly or biweekly sprints allow students to break the project into smaller tasks that can be managed alongside other academic responsibilities. Short iterations improve focus and make progress easier to track, which is especially helpful in a time-limited academic setting (Microsoft, n.d.; Atlassian, n.d.).

Another Agile practice that fits well in a course project is regular stand-up meetings. Short check-ins help team members share progress, identify challenges, and stay accountable. In small student teams, stand-ups improve coordination and reduce the risk of uneven workload distribution (Atlassian, n.d.).

However, some Agile practices are less effective for course projects. One limitation is continuous customer involvement. In academic projects, instructors typically serve as evaluators rather than active stakeholders, which limits opportunities for real-time feedback. This reduces the effectiveness of one of Agile's core principles (Martin, n.d.). Another Agile aspect that does not translate well to coursework is minimal documentation. Agile prioritizes working software over documentation, but course projects usually require written reports, reflections, and formal

submissions for grading. As a result, teams must adapt Agile practices to meet academic documentation requirements (Martin, n.d.; IEEE, n.d.).