COMP 3211

Software Engineering

Zhu Jin Shun

22101071d

Assignment 1

Answers

Question 1: ACM/IEEE SE Code of Ethics and Professional Practice

Clause 1 (Principle 2 Client and Employer)

2.02. Not knowingly use software that is obtained or retained either illegally or unethically.

Software engineers may inadvertently violate the ethical clause against using software obtained or retained illegally or unethically by treating their developed software as personal property. For instance, they might keep original copies to claim ownership and reuse code from prior projects in new developments for convenience, which violates intellectual property rights. To mitigate this risk, organizations should establish clear policies regarding software ownership and usage. Upon project completion, all code and documentation should be transferred to the client, who can acknowledge the engineer's contributions through a certificate. Regular training on ethical standards and intellectual property rights will further reinforce compliance among engineers.

Clause 2: (Principle 7 Colleagues)

7.02. Assist colleagues in professional development.

This clause may be violated when a senior engineer is asked to assist a colleague with less experience or knowledge. For instance, a senior engineer might refuse to mentor junior team members or provide unprofessional answers for asked

questions to avoid creating competition for their future. To prevent this situation, organizations should cultivate a culture of collaboration and knowledge-sharing. Grouping engineers by skill level to promote a fair and supportive environment for professional development. Encouraging experienced engineers to mentor others through workshops or pair programming will enhance skills across the team and ensure compliance with this ethical obligation, ultimately benefiting the entire organization.

Question 2: Software Process

I worked on a WeChat Mini Program designed to assist with tasks like item pickup, express delivery, and takeout for my friend's college. I would consider using an agile method better due to the flexibility needed among our student team from a perspective view now. However, we ultimately opted for a plan-driven approach based on the waterfall model. This allowed us to follow a structured plan, completing each stage before moving next. For example, we finished half of the development before seeking school permission and conducted a survey to gather student opinions on necessary features before finalizing our plans.

Question 3: Manifesto for Agile Software Development

4 Situations

1. Agile software development values individuals and interactions over processes and tools
2. Working software over comprehensive documentation:
3. Customer collaboration over contract negotiation:
4. Responding to change over following a plan

Situation 1:

Excessive focus on relationships and interactions within a group can result in poor understanding of requirements and development processes, ultimately leading to low-quality work that fails to meet project expectations and fulfill necessary requirements.

Situation 2:

Prioritizing working software over documentation can hinder new members' understanding and contributions, while existing members may forget crucial details, resulting in inconsistencies and potential issues when changes are needed.

Situation 3:

Excessive focus on customer collaboration over contract negotiations may lead to scope creep, resulting in an incomplete product that fails to meet customer expectations due to constantly changing requirements not documented in the contract.

Situation 4:

When a team excessively prioritizes responding to changes and abandons the initial project plan, it can lead to disarray. If only one developer implements changes without team coordination, the overall development process becomes chaotic and ineffective, ending in a totally messed up development process for the overall software development.

Question 4: eXtreme Programming

The first issue addresses the inadequate consideration of all stakeholders' perspectives in software requirements, which leads to rework. It emphasizes the importance of incorporating both end-user and client viewpoints upfront, a reminder that is often necessary for students. The second issue highlights that quality is typically recognized only when absent, complicating late-stage issue resolution. Students need a solid understanding of quality techniques, although these methods can seem excessive for smaller projects, potentially leading to unsuitable processes. Both issues are discussed in Section 3.1 at the software requirements and software quality sub-section.

From the discussion on stakeholders' perspectives, I learned that involving both end-users and clients early in the project is crucial to minimizing rework. If I apply XP to a future course project, this lesson will be invaluable. By prioritizing

collaboration and maintaining frequent communication with stakeholders, I can ensure that requirements are clear and agreed upon from the start. Regular feedback loops, such as user stories and continuous integration, will help refine our understanding of requirements and allow for quick adjustments, ultimately leading to a more successful project outcome.

Regarding software quality, I learned that quality is often only noticed when lacking, complicating late-stage issue resolution. While some quality techniques may appear excessive for smaller projects, appropriately adapting them can foster a culture of quality, enabling the team to identify and address potential issues early on, ultimately leading to a more robust final product. Additionally, I recognize that starting and testing with smaller projects before progressing to larger ones can enhance efficiency and improve development outcomes. Applying this lesson in future XP projects will be beneficial. By integrating quality practices from the outset such as unit testing, continuous feedback, and pair programming, I can uphold high standards throughout development.

Question 5: Requirements Specification

Functional system requirements:

The system shall allow users to schedule a meeting using Microsoft Teams or Zoom by providing a title, date, time, duration, and invitees, ensuring no conflicts with existing meetings and sending confirmation emails to all

participants.

Non-Functional system requirements:

The system should be usable by users with varying technical skills, enabling them to schedule a meeting within three steps or fewer, ensuring 90% can complete the task independently.