Question Bank

12 agility principles
Human factors affecting agile
Explain a) agile b) scrum

Explain principles that guide process

Explain principles that guide practice
Explain communication principles

12:46 PM

Agility principles for those who want to achieve agility:

1. Our highest priority is to **satisfy the customer** through early and continuous delivery of valuable software.
2. **Welcome changing requirements**, even late in development. Agile processes harness change for the customer's competitive advantage.
3. **Deliver working software frequently**, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers **must work together** daily throughout the project.
5. Build projects around **motivated individuals**. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team **is face-to-face conversation**.
7. **Working software** is the primary measure of progress.
8. Agile processes promote **sustainable development**. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to **technical excellence and good design** enhances agility.
10. **Simplicity**—the art of maximizing the amount of work not done—is essential.
11. The best architectures, requirements, and designs emerge from **self– organizing teams**
12. At regular intervals, the team reflects on how to become more ***effective***, then tunes and adjusts its behavior accordingly.

1.3.3 Human Factors

Agile development focuses on the talents and skills of individuals, molding the process to specific people and teams." The key point in this statement is that *the process molds to the needs of the people and team.*

1. **Competence**. In an agile development context, "competence" encompasses innate talent, specific software-related skills, and overall knowledge of the process that the team has chosen to apply. Skill and knowledge of process can and should be taught to all people who serve as agile team members.

2. **Common focus**. Although members of the agile team may perform different tasks and bring different skills to the project, all should be focused on one goal—to deliver a working software increment to the customer within the time promised. To achieve this goal, the team will also focus on continual adaptations (small and large) that will make the process fit the needs of the team.

3. **Collaboration**. Software engineering (regardless of process) is about assessing, analyzing, and using information that is communicated to the software team; creating information that will help all stakeholders understand the work of the team; and building information (computer software and relevant databases) that provides business value for the customer. To accomplish these tasks, team members must collaborate—with one another and all other stakeholders.

4. **Decision-making ability.** Any good software team (including agile teams) must be allowed the freedom to control its own destiny. This implies that the team is given autonomy—decision-making authority for both technical and project issues.

5. **Fuzzy problem-solving ability.** Software managers must recognize that the agile team will continually have to deal with ambiguity and will continually be buffeted by change.

6. **Mutual trust and respect.** The agile team must become what DeMarco and Lister call a "jelled" team. A jelled team exhibits the trust and respect that are necessary to make them "so strongly knit that the whole is greater than the sum of the parts."

7. **Self-organization**. In the context of agile development, self-organization implies three things: (1) the agile team organizes itself for the work to be done, (2) the team organizes the process to best accommodate its local environment, (3) the team organizes the work schedule to best achieve delivery of the software increment. Self-organization has a number of technical benefits, but more importantly, it serves to improve collaboration and boost team morale.

Scrum

Scrum is an agile software development method that was conceived by Jeff Sutherland and his development team in the early 1990s. Scrum principles are consistent with the agile manifesto and are used to guide development activities within a process that incorporates the following framework activities: requirements, analysis, design, evolution, and delivery. Within each framework activity, work tasks occur within a process pattern called a sprint. The work conducted within a sprint is adapted to the problem at hand and is defined and often modified in real time by the Scrum team. The overall flow of the Scrum process is illustrated in following figure.

1.3 WHAT IS AN AGILE PROCESS?

Any agile software process is characterized in a manner that addresses a number of key assumptions about the majority of software projects:

1. It is difficult to predict in advance which software requirements will persist and which will change. It is equally difficult to predict how customer priorities will change as the project proceeds.
2. For many types of software, design and construction are interleaved. That is, both activities should be performed in tandem so that design models are proven as they are created. It is difficult to predict how much design is necessary before construction is used to prove the design.
3. Analysis, design, construction, and testing are not as predictable.

Given these three assumptions, an important question arises:

1) How do we create a process that can manage unpredictability?

It lies in process **adaptability**. **An agile process, therefore, must be adaptable.** But continual adaptation without forward progress accomplishes little. Therefore, an agile software process must adapt incrementally.

To accomplish incremental adaptation, an agile team requires customer feedback. An effective catalyst for customer feedback is an operational prototype or a portion of an operational system. Hence, an incremental development strategy should be instituted. Software increments must be delivered in short time periods so that adaptation keeps pace with change.

This iterative approach enables the customer to evaluate the software increment regularly, provide necessary feedback to the software team, and influence the process adaptations that are made to accommodate the feedback.

1.8.1 Principles that Guide Process

The following set of core principles can be applied to the framework, and by extension, to every software process.

Principle 1: Be agile. Whether the process model you choose is prescriptive or agile.

1. keep your technical approach as simple as possible
2. keep the work products you produce as concise(short) as possible
3. Make decisions locally whenever possible.

Principle 2. Focus on quality at every step. For every process activity, action, and task should focus on the quality of the work product that has been produced.

Principle 3. Be ready to adapt. adapt your approach to conditions imposed by the problem, the people, and the project itself.

Principle 4. Build an effective team. Build a self-organizing team that has mutual trust and respect.

Principle 5. Establish mechanisms for communication and coordination. Projects fail because stakeholders fail to coordinate their efforts to create a successful end product.

Principle 6. Manage change. The methods must be established to manage the way changes are requested, approved, and implemented.

Principle 7. Assess risk. Lots of things can go wrong as software is being developed.

Principle 8. Create work products that provide value for others. Create only those work products that provide value for other process activities, actions and tasks.

1.8.2 Principles That Guide Practice

- Software engineering practice has a single goal i.e.., to deliver on-time, high quality, operational software that contains functions and features that meet the needs of all stakeholders.
- To achieve this goal, should adopt a set of core principles that guide the technical work.
- The following **set of core principles are fundamental to the practice of software engineering**:

Principle 1. Divide and conquer. A large problem is easier to solve if it is subdivided into a collection of elements (or modules or components). Ideally, each element delivers distinct functionality that can be developed.

Principle 2. Understand the use of abstraction. At its core, an abstraction(overview) is a simplification of some complex element of a system used to communicate meaning in a single phrase.

Principle 3. Strive for consistency. Whether it's creating a requirements model, developing a software design, generating source code, or creating test cases. All these are consistent so that the software is easier to develop.

Principle 4. Focus on the transfer of information. Software is about information transfer—from a database to an end user, from an operating system to an application,

Principle 5. Build software that exhibits effective modularity. Modularity provides a mechanism for any complex system can be divided into modules (components).

Principle 6. Look for patterns. Brad Appleton suggests that: The goal of patterns within the software community is to create a body of literature to help software developers resolve recurring problems encountered throughout all of software development process.

Principle 7. When possible, represent the problem and its solution from a number of different perspectives. When a problem and its solution are examined from a number of different perspectives(ways).

Principle 8. Remember that someone will maintain the software. software will be corrected as defects are remove, adapted as its environment changes, and enhanced as stakeholders request more capabilities.

1.9.1 Communication Principles

Customer requirements must be gathered through the **communication activity**. Communication has begun.

Principle 1. Listen.

Try to focus on the speaker's words, rather than formulating your response to those words. Ask for clarification if something is unclear, but avoid constant interruptions. Never become contentious in your words or actions (e.g., rolling your eyes or shaking your head) as a person is talking.

Principle 2. Prepare before you communicate. Spend the time to understand the problem before you meet with others. If necessary, do some research to understand business domain jargon. If you have responsibility for conducting a meeting, prepare an agenda in advance of the meeting.

Principle 3. Someone should facilitate the activity. Every communication meeting should have a leader (a facilitator) to keep the conversation moving in a productive direction, (2) to mediate any conflict that does occur, and (3) to ensure than other principles are followed.

Principle 4. Face-to-face communication is best. But it usually works better when some other representation of the relevant information is present. For example, a participant may create a drawing or a "strawman" document that serves as a focus for discussion.

Principle 5. Take notes and document decisions. Things have a way of falling into the cracks. Someone participating in the communication should serve as a "recorder" and write down all important points and decisions.

Principle 6. Strive for collaboration. Collaboration and consensus occur when the collective knowledge of members of the team is used to describe product or system functions or features. Each small collaboration serves to build trust among team members and creates a common goal for the team.

Principle 7. Stay focused; modularize your discussion. The more people involved in any communication, the more likely that discussion will bounce from one topic to the next. The facilitator should keep the conversation modular, leaving one topic only after it has been resolved

Principle 8. If something is unclear, draw a picture. Verbal communication goes only so far. A sketch or drawing can often provide clarity when words fail to do the job.

Principle 9. (a) Once you agree to something, move on. (b) If you can't agree to something, move on. (c) If a feature or function is unclear and cannot be clarified at the moment, move on. Communication, like any software engineering activity, takes time. Rather than iterating endlessly, the people who participate should recognize that many topics require discussion (see Principle 2) and that "moving on" is sometimes the best way to achieve communication agility.

Principle 10. Negotiation is not a contest or a game. It works best when both parties win. There are many instances in which you and other stakeholders must negotiate functions and features, priorities, and delivery dates. If the team has collaborated well, all parties have a common goal. Still, negotiation will demand compromise from all parties.