Agile Software Testing:

Agile Testing is a type of software testing that follows the principles of agile software development to test the software application. All members of the project team along with the special experts and testers are involved in agile testing. Agile testing is not a separate phase and it is carried out with all the development phases i.e. requirements, design and coding, and test case generation. Agile testing takes place simultaneously throughout the Development Life Cycle. Agile testers participate in the entire development life cycle along with development team members and the testers help in building the software according to the customer requirements and with better design and thus code becomes possible. The agile testing team works as a single team towards the single objective of achieving quality. Agile Testing has shorter time frames called iterations or loops. This methodology is also called the delivery-driven approach because it provides a better prediction on the workable products in less duration time.

- Agile testing is an informal process that is specified as a dynamic type of testing.
- It is performed regularly throughout every iteration of the Software Development Lifecycle (SDLC).
- Customer satisfaction is the primary concern for agile test engineers at some stage in the agile testing process.

Features of Agile Testing

Some of the key features of agile software testing are:

- **Simplistic approach:** In agile testing, testers perform only the necessary tests but at the same time do not leave behind any essential tests. This approach delivers a product that is simple and provides value.
- **Continuous improvement:** In agile testing, agile testers depend mainly on feedback and self-learning for improvement and they perform their activities efficiently continuously.
- **Self-organized:** Agile testers are highly efficient and tend to solve problems by bringing teams together to resolve them.
- **Testers enjoy work:** In agile testing, testers enjoy their work and thus will be able to deliver a product with the greatest value to the consumer.
- **Encourage Constant communication:** In agile testing, efficient communication channels are set up with all the stakeholders of the project to reduce errors and miscommunications.
- **Constant feedback:** Agile testers need to constantly provide feedback to the developers if necessary.

Agile Testing Methodologies

Some of the agile testing methodologies are:

Test-Driven Development (TDD): TDD is the software development process relying on creating unit test cases before developing the actual code of the software. It is an iterative approach that combines 3 operations, programming, creation of unit tests, and refactoring.

Behavior Driven Development (BDD): BDD is agile software testing that aims to document and develop the application around the user behavior a user expects to experience when interacting with the application. It encourages collaboration among the developer, quality experts, and customer representatives.

Exploratory Testing: In exploratory testing, the tester has the freedom to explore the code and create effective and efficient software. It helps to discover the unknown risks and explore each aspect of the software functionality.

Acceptance Test-Driven Development (ATDD): ATDD is a collaborative process where customer representatives, developers, and testers come together to discuss the requirements, and potential pitfalls and thus reduce the chance of errors before coding begins.

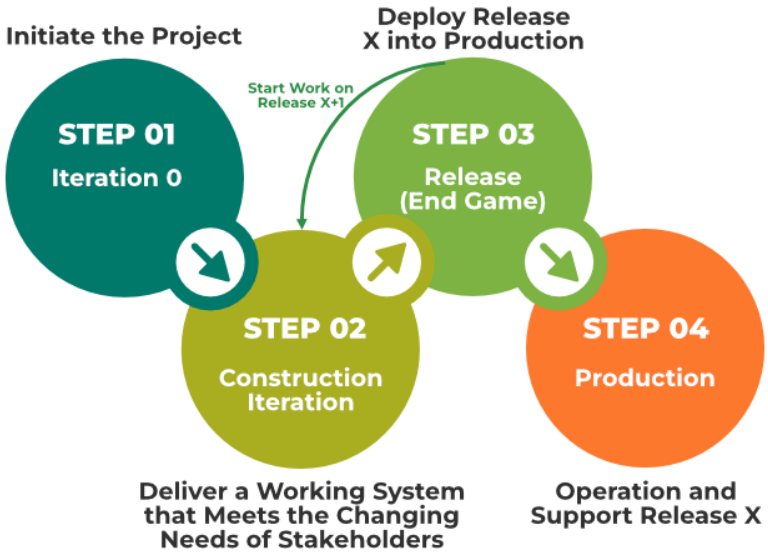
Extreme Programming (XP): Extreme programming is a customer-oriented methodology that helps to deliver a good quality product that meets customer expectations and requirements.

Session-Based Testing: It is a structured and time-based approach that involves the progress of exploratory testing in multiple sessions. This involves uninterrupted testing sessions that are time-boxed with a duration varying from 45 to 90 minutes. During the session, the tester creates a document called a charter document that includes various information about their testing.

Dynamic Software Development Method (DSDM): DSDM is an agile project delivery framework that provides a framework for building and maintaining systems. It can be used by users, developers, and testers.

Crystal Methodologies: This methodology focuses on people and their interactions when working on the project instead of processes and tools. The suitability of the

crystal method depends on three dimensions, team size, criticality, and priority of the project.



Agile Testing Life Cycle

The agile testing life cycle has 5 different phases:

Impact Assessment: This is the first phase of the agile testing life cycle also known as the feedback phase where the inputs and responses are collected from the users and stakeholders. This phase supports the test engineers to set the objective for the next phase in the cycle.

Agile Testing Planning: In this phase, the developers, customers, test engineers, and stakeholders team up to plan the testing process schedules, regular meetings, and deliverables.

Release Readiness: This is the third phase in the agile testing lifecycle where the test engineers review the features which have been created entirely and test if the features are ready to go live or not and the features that need to be sent again to the previous development phase.

Daily Scrums: This phase involves the daily morning meetings to check on testing and determine the objectives for the day. The goals are set daily to enable test engineers to understand the status of testing.

Test Agility Review: This is the last phase of the agile testing lifecycle that includes weekly meetings with the stakeholders to evaluate and assess the progress against the goals.

Agile testing lifecycle

Agile Test Plan

An agile test plan includes types of testing done in that iteration like test data requirements, test environments, and test results. In agile testing, a test plan is written and updated for every release. The test plan includes the following:

Test Scope.

Testing instruments.

Data and settings are to be used for the test.

Approaches and strategies used to test.

Skills required to test.

New functionalities are being tested.

Levels or Types of testing based on the complexity of the features.

Resourcing.

Deliverables and Milestones.

Infrastructure Consideration.

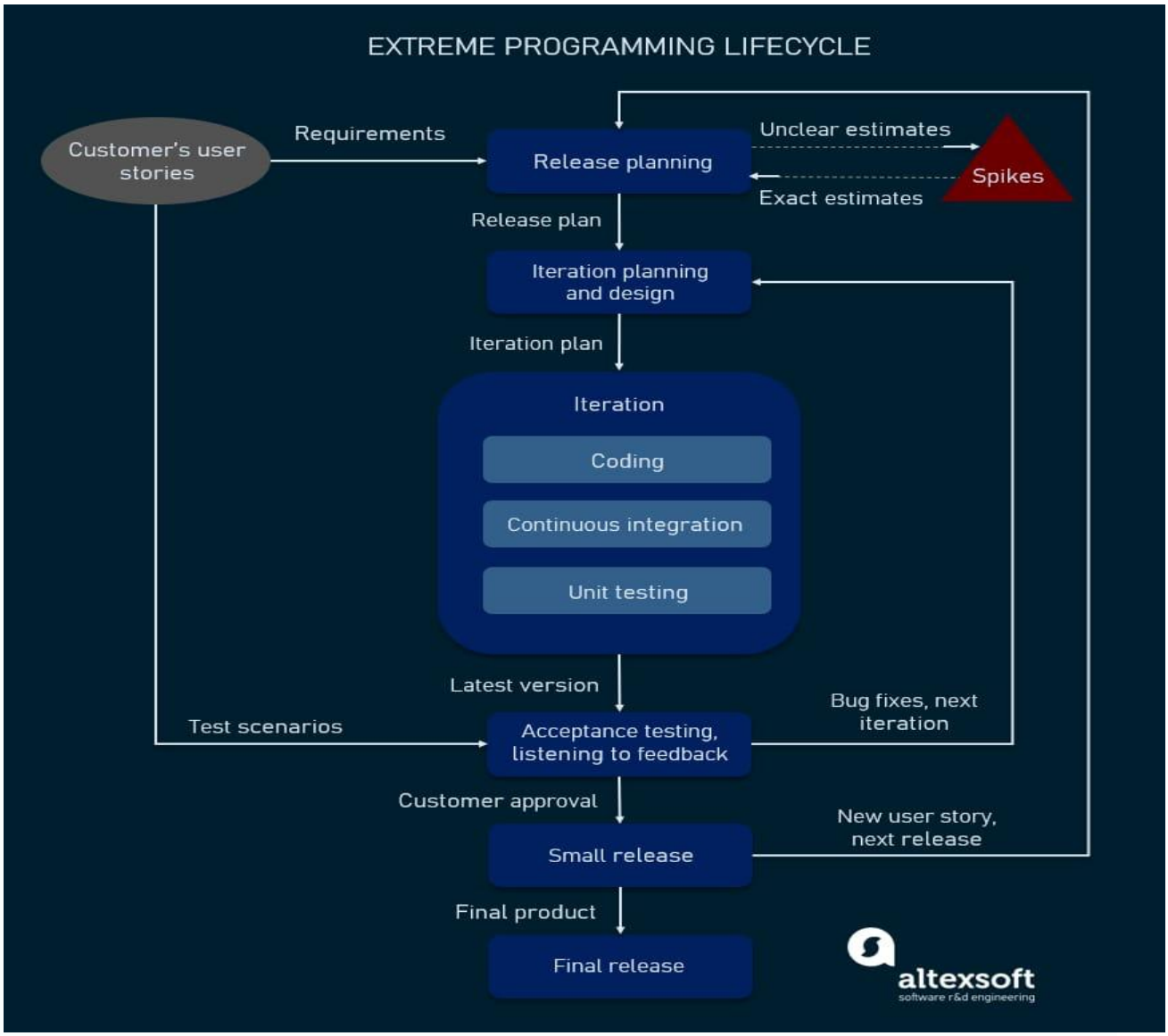
Load or Performance Testing.

Mitigation or Risks Plan.

Extreme programming:

The XP framework normally involves 5 phases or stages of the development process that iterate continuously:

1. **Planning,** the first stage, is when the customer meets the development team and presents the requirements in the form of user stories to describe the desired result. The team then estimates the stories and creates a release plan broken down into iterations needed to cover the required functionality part after part. If one or more of the stories can't be estimated, so-called *spikes* can be introduced which means that further research is needed.
2. **Designing** is actually a part of the planning process, but can be set apart to emphasize its importance. It's related to one of the main XP values that we'll discuss below -- simplicity. A good design brings logic and structure to the system and allows to avoid unnecessary complexities and redundancies.
3. **Coding** is the phase during which the actual code is created by implementing specific XP practices such as coding standards, pair programming, continuous integration, and collective code ownership (the entire list is described below).
4. **Testing** is the core of extreme programming. It is the regular activity that involves both unit tests (automated testing to determine if the developed feature works properly) and acceptance tests (customer testing to verify that the overall system is created according to the initial requirements).
5. **Listening** is all about constant communication and feedback. The customers and project managers are involved to describe the business logic and value that is expected.



XP lifecycle in a nutshell

Such a development process entails the cooperation between several participants, each having his or her own tasks and responsibilities. Extreme programming puts people in the center of the system, emphasizing the value and importance of such social skills as communication, cooperation, responsiveness, and feedback. So, these roles are commonly associated with XP:

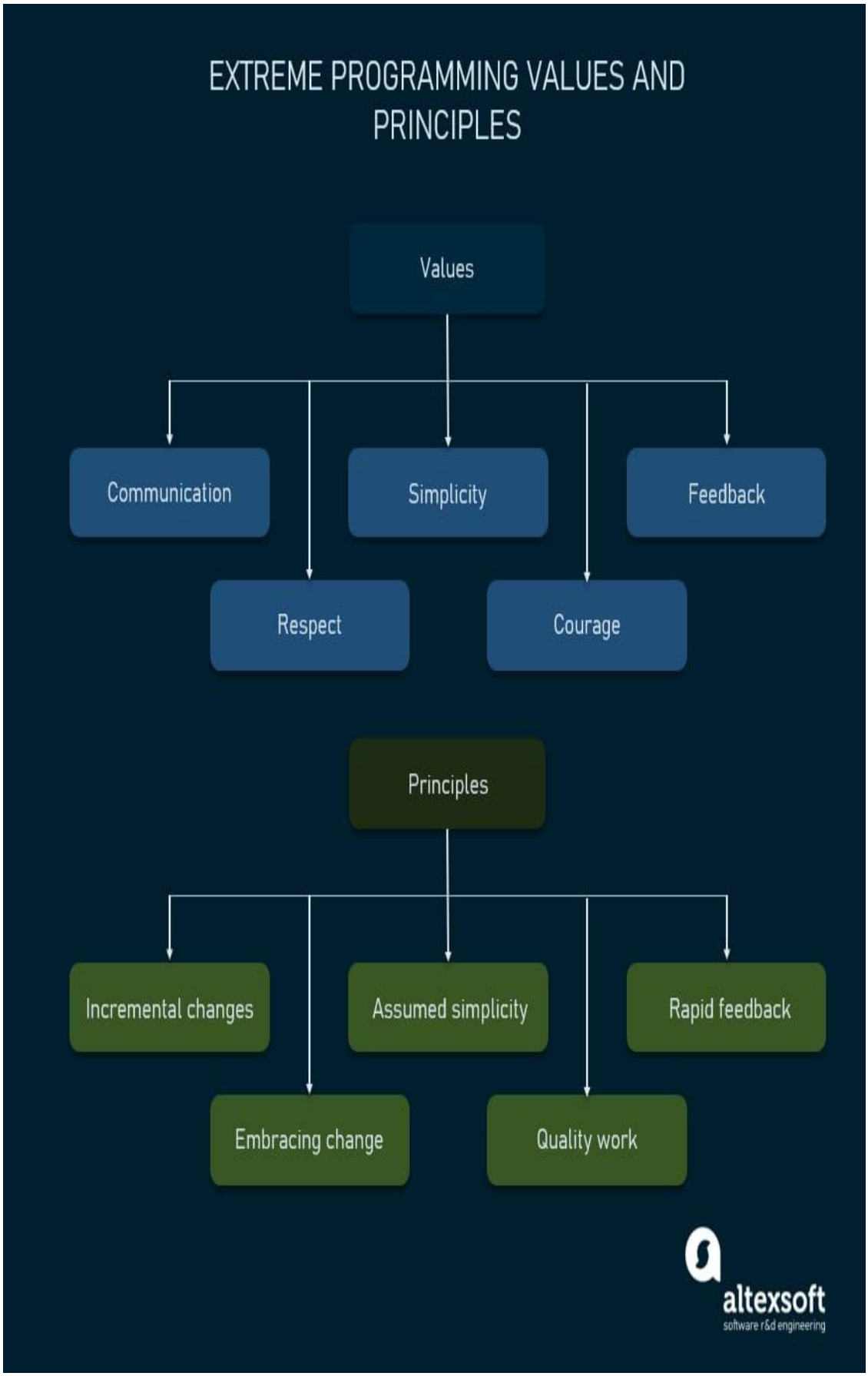
1. **Customers** are expected to be heavily engaged in the development process by creating user stories, providing continuous feedback, and making all the necessary business decisions related to the project.
2. **Programmers or developers** are the team members that actually create the product. They are responsible for implementing user stories and conducting user

tests (sometimes a separate **Tester** role is set apart). Since XP is usually associated with cross-functional teams, the skill set of such members can be different.

3. **Trackers or managers** link customers and developers. It's not a required role and can be performed by one of the developers. These people organize the meetups, regulate discussions, and keep track of important progress KPIs.
4. **Coaches** can be included in the teams as mentors to help with understanding the XP practices. It's usually an outside assistant or external consultant who is not involved in the development process, but has used XP before and so can help avoid mistakes.

Values and principles of extreme programming

In the late 90s, Ken Beck summarized a set of certain values and principles that describe extreme programming and lead to more effective cooperation within the team and, ultimately, higher product quality.


EXTREME PROGRAMMING VALUES AND PRINCIPLES
Values
Communication
Simplicity
Feedback
Respect
Courage
Principles
Incremental changes
Assumed simplicity
Rapid feedback
Embracing change
Quality work
altexsoft
software r&d engineering

Values and principles of XP

Values of extreme programming

XP has simple rules that are based on 5 values to guide the teamwork:

1. **Communication.** Everyone on a team works jointly at every stage of the project.
2. **Simplicity.** Developers strive to write simple code bringing more value to a product, as it saves time and effort.
3. **Feedback.** Team members deliver software frequently, get feedback about it, and improve a product according to the new requirements.
4. **Respect.** Every person assigned to a project contributes to a common goal.
5. **Courage.** Programmers objectively evaluate their own results without making excuses and are always ready to respond to changes.

These values represent a specific mindset of motivated team players who do their best on the way to achieving a common goal. XP principles derive from these values and reflect them in more concrete ways.