ACCEPTANCE CRITERIA

Outcome

Acceptance criteria are conditions that a feature or product must satisfy in order to be accepted by a user or customer.

Function

Writing acceptance criteria for customer requirements helps to define the boundaries of the change, gives the team and customer a shared definition of done and provides a basis for acceptance testing.

The format that acceptance criteria take depends to a large extent on other considerations, such as how requirements are captured (user stories versus use cases, for example) and a team's development and testing practices. However, all acceptance criteria should have the following in common:

- The customer owns the acceptance criteria, but they are generated and refined in close collaboration with the development team (including business analysts, testers and developers).
- They are written in the business domain language.
- They will be refined while the requirement is still being discussed but the team should have a high degree of confidence about them before starting development.
- They should serve as the foundation for acceptance testing.
- They are often backed up with examples.

Benefit

Implementation

Who

Embed acceptance criteria into your requirements at the following points in your development process:

When a requirement is first discussed with a customer

Ensure that some high-level rules are captured about the requirement. If you are using user stories, you can write these on the back of the card.

When analysis or design work is being done

Talk to the customer and get specific examples. Get quick feedback: Are there more rules? Can we get more examples for the rules? Can we be more specific about the examples? What tests would she run to check that the product delivered matches her expectation?

User Story

In order to be refunded or to avoid a charge, as a travel site user, I want to cancel a reservation.

Acceptance criteria

A user who cancels more than twenty-four hours in advance gets a complete refund.

A user who cancels less than twenty-four hours in advance is refunded all except a £25 cancellation fee.

A cancellation code is displayed on the site and is emailed to the user.

Example of acceptance criteria after first discussion

Scaling Factors

Stop writing acceptance criteria when additional criteria do not clarify the requirement (e.g. user story or use case) and its intent.

Difficulty

Involve tester in order that testing could be outlined and because testers will ask the right questions. What would success/failure look like? Happy Flow? Exceptions? etc.

Before starting development

Review the acceptance criteria with the customer. Have we understood this? Go into more detail the nearer you are to developing it... Make sure testers and developers are involved at this stage as questions are better asked now.

Potential Pitfalls

- Don't use the acceptance criteria to define the solution. For example, "customer selects payment type from dropdown menu" would be supplying the implementation. Instead, the criteria should look at customer outcome, "customer can select payment type".
- Don't view acceptance criteria as complete when raising requirement. It's OK to refine them before development; in fact they *should* be reviewed and refined before development.
- If you do not involved testers during the analysis phase, your acceptance criteria may not capture the necessary tests.
- Do not confuse acceptance criteria with functional, user interface and unit tests. These tests are derived from the acceptance criteria, but they will go much deeper in testing all the possible flows (especially the exceptional and error flows), their detailed steps and boundary conditions. Conveniently, existing tools can help convert the acceptance criteria into functional tests (also known as acceptance tests) and integrate them into a continuous integration build cycle.

If you want to learn more, consider reading:
User Stories Applied: For Agile Software Development by Mike Cohn
Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise by Dean Leffingwell