**Acceptance test–driven development (GITHUB) (test)**

Acceptance test–driven development nedir

**Acceptance test–driven development** (**ATDD**) is a [development](https://www.wikiwand.com/en/Software_development) methodology based on communication between the business customers, the developers, and the testers.[[1]](https://www.wikiwand.com/en/Acceptance_test%E2%80%93driven_development#citenotePugh111) ATDD encompasses many of the same practices as [specification by example](https://www.wikiwand.com/en/Specification_by_example) (SBE),[[2]](https://www.wikiwand.com/en/Acceptance_test%E2%80%93driven_development#citenote2)[[3]](https://www.wikiwand.com/en/Acceptance_test%E2%80%93driven_development#citenote3) [behavior-driven development](https://www.wikiwand.com/en/Behavior-driven_development) (BDD),[[4]](https://www.wikiwand.com/en/Acceptance_test%E2%80%93driven_development#citenote4) example-driven development (EDD),[[5]](https://www.wikiwand.com/en/Acceptance_test%E2%80%93driven_development#citenote5) and support-driven development also called story test–driven development (SDD).[[6]](https://www.wikiwand.com/en/Acceptance_test%E2%80%93driven_development#citenote6) All these processes aid developers and testers in understanding the customer's needs prior to implementation and allow customers to be able to converse in their own domain language.

ATDD is closely related to [test-driven development](https://www.wikiwand.com/en/Test-driven_development) (TDD).[[7]](https://www.wikiwand.com/en/Acceptance_test%E2%80%93driven_development#citenote7) It differs by the emphasis on developer-tester-business customer collaboration. ATDD encompasses [acceptance testing](https://www.wikiwand.com/en/Acceptance_testing), but highlights writing acceptance tests before developers begin coding.

### **Overview**

Acceptance tests are from the user's point of view – the external view of the system.[[1]](https://www.wikiwand.com/en/Acceptance_test%E2%80%93driven_development#citenotePugh111) They examine externally visible effects, such as specifying the correct output of a system given a particular input. Acceptance tests can verify how the state of something changes, such as an order that goes from "paid" to "shipped". They also can check the interactions with interfaces of other systems, such as shared databases or web services. In general, they are implementation independent, although automation of them may not be.[[8]](https://www.wikiwand.com/en/Acceptance_test%E2%80%93driven_development#citenote8)[[9]](https://www.wikiwand.com/en/Acceptance_test%E2%80%93driven_development#citenote9)

### **Creation**

Acceptance tests are created when the requirements are analyzed and prior to coding.[[1]](https://www.wikiwand.com/en/Acceptance_test%E2%80%93driven_development#citenotePugh111) They can be developed collaboratively by requirement requester (product owner, business analyst, customer representative, etc.), developer, and tester. Developers implement the system using the acceptance tests. Failing tests provide quick feedback that the requirements are not being met. The tests are specified in business domain terms. The terms then form a ubiquitous language that is shared between the customers, developers, and testers.[[10]](https://www.wikiwand.com/en/Acceptance_test%E2%80%93driven_development#citenote10) Tests and requirements are interrelated.[[11]](https://www.wikiwand.com/en/Acceptance_test%E2%80%93driven_development#citenote11) A requirement that lacks a test may not be implemented properly. A test that does not refer to a requirement is an unneeded test. An acceptance test that is developed after implementation begins represents a new requirement.[[12]](https://www.wikiwand.com/en/Acceptance_test%E2%80%93driven_development#citenote12)

### **Testing strategy**

Acceptance tests are a part of an overall testing strategy. They are the customer tests that demonstrate the business intent of a system. Component tests are technical acceptance tests developed by an architect that specify the behavior of large modules. Unit tests are created by the developer to drive easy-to-maintain code.[[13]](https://www.wikiwand.com/en/Acceptance_test%E2%80%93driven_development#citenote13) They are often derived from acceptance tests and unit tests. Cross-functional testing includes usability testing,[[14]](https://www.wikiwand.com/en/Acceptance_test%E2%80%93driven_development#citenote14) exploratory testing,[[15]](https://www.wikiwand.com/en/Acceptance_test%E2%80%93driven_development#citenote15) and property testing (scaling and security).[[16]](https://www.wikiwand.com/en/Acceptance_test%E2%80%93driven_development#citenote16)

## **Acceptance criteria and tests**

Acceptance criteria are a description of what would be checked by a test. Given a requirement such as "As a user, I want to check out a book from the library", an acceptance criterion might be, "verify the book is marked as checked out". An acceptance test for this requirement gives the details so that the test can be run with the same effect each time.

### **Test format**

Acceptance tests usually follow this form:[[1]](https://www.wikiwand.com/en/Acceptance_test%E2%80%93driven_development#citenotePugh111)

**Given (setup)**

A specified state of a system

**When (trigger)**

An action or event occurs

**Then (verification)**

The state of the system has changed or an output has been produced

Also, it is possible to add Statements that start with **AND** in any of the sections below (Given, When, Then).

For the example requirement, the steps could be listed as:

Given Book that has **not** been checked **out**

**And** User who **is** registered **on** the system

**When** User checks **out** a book

**Then** Book **is** marked as checked **out**

### **Complete test**

The previous steps do not include any specific example data, so that is added to complete the test:

**Given:**

Book that has not been checked out

|  |  |
| --- | --- |
| ***Books*** |  |
| Title | Checked out |
| Great book | No |

User who is registered on the system

|  |
| --- |
| ***Users*** |
| Name | Sam |

**When:**

User checks out a book

|  |
| --- |
| ***Checkout action*** |
| User | Sam | Checks out | Great book |

**Then:**

Book is marked as checked out

|  |
| --- |
| ***Books*** |
| Title | Checked out | User |
| Great book | Yes | Sam |

### **Test examination**

Examination of the test with specific data usually leads to many questions. For the sample, these might be:

* What if the book is already checked out?
* What if the book does not exist?
* What if the user is not registered on the system?
* Is there a date that the book is due to be checked-in?
* How many books can a user check out?

These questions help illuminate missing or ambiguous requirements. Additional details such as a due-date can be added to the expected result. Other acceptance tests can check that conditions such as attempting to check out a book that is already checked out produces the expected error.

### **Another test example**

Suppose the business customer wanted a business rule that a user could only check out one book at a time. The following test would demonstrate that:

**Scenario:** Check that checkout business rule is enforced

**Given:**

Book that has been checked out

|  |
| --- |
| ***Books*** |
| Title | Checked out | User |
| Great book | Yes | Sam |
| Another great book | No |  |

|  |
| --- |
| ***Users*** |
| Name |
| Sam |

**When:**

User checks out another book

|  |
| --- |
| ***Checkout action*** |
| User | Sam | Checks out | Another great book |

**Then:**

Error occurs

|  |
| --- |
| ***Error occurred*** |
| Description |
| Violation of checkout business rule |

### **Project acceptance tests**

In addition to acceptance tests for requirements, acceptance tests can be used on a project as a whole.[[1]](https://www.wikiwand.com/en/Acceptance_test%E2%80%93driven_development#citenotePugh111) For example, if this requirement was part of a library book checkout project, there could be acceptance tests for the whole project. These are often termed [SMART objectives](https://www.wikiwand.com/en/SMART_criteria). An example test is "When the new library system is in production, the users will be able to check books in and out three times as fast as they do today".

-

-

-