[Xray] TDD vs ATDD

TDD

TDD is a practice derived from Extreme Programming, that aims to provide the ability to make refinements on the code (i.e. refactoring) with confidence so that internal methods/functions/interfaces maintain the input/output logic.

In TDD (test-driven development), the process can be described in high-level as:

- mostly unit tests (and possibly integration tests) are created first, by the developer
- tests are added to the CI pipeline and start running
- implementation of feature goes on until tests pass

TDD's focus is on how features are implemented.

In Xray, a possible approach for TDD would be:

1. define the story/requirement and add it to the project's backlog
    a. even though this is not specific to TDD itself, it's important to clarify the story/requirement with the team and get an agreement, including on the acceptance criteria (the workflow status can be used to signal it)
2. developers scratch the skeleton of the code
3. developers start implementing automated checks (i.e. "automated tests") validating functions/methods of the code that will be implemented; these include unit-level tests and, optionally upper levels (e.g. integration tests)
4. automated tests are added to the CI pipeline and run frequently
    a. some of these test results can be submitted to Xray
        i. we don't recommend submitting unit-level test results though; these should be tracked and monitored in the CI tool. Unit tests are quite dynamic and would pollute Jira easily (they may not even be clearly related with a requirement/story)
5. developers implement code of feature until tests pass

In sum,

- automated unit tests are written first, then executed in CI; implementation of feature goes on until all test are passing

Note that pure TDD diz not tell you if some feature delivers the expected behavior; its purpose is to assure that you build right a piece of software (and not on building the right software).

ATDD

ATDD (acceptance test-driven development) is an extension to TDD but more focused on the agreed behavior that a given feature must provide (i.e. assuring that you build the right software).

In Xray, a possible approach for ATDD would be:

1. define the story/requirement and add it to the project's backlog
2. clarify the story/requirement with the team and get an agreement, including on the acceptance criteria (the workflow status can be used to signal it)
    a. Related to each acceptance-criteria, acceptance-tests will be depicted and written in an executable format (i.e. their specification can be interpreted and executed by an automation framework)
        i. You can use built-in capabilities of Xray for Gherkin based tests (e.g. Cucumber/SpecFlow/Behave scenarios/scenario outlines) and create those in Xray
        ii. You can also use other frameworks (e.g. Robot framework, Fitnesse); in that case, you would define tests elsewhere (i.e. outside of Xray, storing them in the SCVS)
            1. if using Robot, you can specify the issue key of the requirement/story in Jira right in the specification of the acceptance test so that whenever results are imported, the corresponding Test would automatically be linked to the requirement /story
3. Unit testing would proceed as usual
    a. developers scratch the skeleton of the code
    b. developers start implementing automated checks (i.e. "automated tests") validating functions/methods of the code that will be implemented (unit tests)
4. Acceptance-tests will be implemented (before further development of the feature goes on)
5. automated tests are added to the CI pipeline and run frequently
    a. unit tests => we don't recommend submitting unit-level test results to Xray
    b. acceptance tests
        - these can be submitted to Xray and tracked in Test Executions and Test Plans

        - Test entities will be auto-provisioned, if necessary, and linked back to the respective requirement/story
    - can also be tracked from the covered item perspective (i.e. from requirements/story) taking advantage of Xray coverage capabilities, including the Overall Coverage report
6. developers implement code of feature until all tests (e.g. unit, acceptance tests) pass

In sum,

- if using supported Gherkin-based frameworks, such as Cucumber, Tests are written in Xray, implemented, executed in the CI and results reported back to Xray
- if using other frameworks (e.g. Robot framework), Tests are specified elsewhere, implemented, executed in the CI and results reported back to Xray; Xray will auto-provision them and link the results to the stories/requirements (if possible)