Example test plan

```
{
    "name": "mastodon-follow-local",
    "session_template": {
        "tests": [
            {
                "name": "system2.follow_receive::FollowReceiveTest"
            },
            {
                "name": "system2.like_single_hop::LikeTest"
            },
            {
                "name": "system2.reply_single_hop::ReplyTest"
            }
        ]
    },
    "constellations": [
        {
            "name": "Mastodon vs Mastodon",
            "roles": {
               "leader_node": {
                   "nodedriver": "MastodonUbosNodeDriver"
               },
               "follower_node": {
                   "nodedriver": "MastodonUbosNodeDriver"
               }
            }
        },
        {
            "name": "Mastodon vs WordPress+plugins”,
            "roles": {
               "leader_node": {
                   "nodedriver": "MastodonUbosNodeDriver"
               },
               "follower_node": {
                   "nodedriver": "WordPressPlusPluginsUbosNodeDriver"
               }
            }
        }
    ]
}
```

Example test class

```
@test
class SendNoteToTest:
   def __init__(self,
        sender_node: FediverseNode,
        receiver_node: FediverseNode
    ) -> None:
        self.sender_node = sender_node
        self.sender_actor_acct_uri = None

        self.receiver_node = receiver_node
        self.receiver_actor_acct_uri = None

        self.note_content =  f"Testing sender_creates_note {datetime.now()}"
        self.note_uri = None


    @step
    def provision_actors(self):
        self.sender_actor_acct_uri = self.sender_node.obtain_actor_acct_uri()
        self.receiver_actor_acct_uri = self.receiver_node.obtain_actor_acct_uri()


    @step
    def sender_creates_note(self):
        self.note_uri = self.sender_node.make_create_note(
            self.sender_actor_acct_uri,
            self.note_content,
            deliver_to=[ self.receiver_actor_acct_uri ])
        assert self.note_uri


    @step
    def wait_until_note_received(self):
        received_content = poll_until(
            lambda: self.receiver_node.actor_has_received_object(self.receiver_actor_acct_uri, self.note_uri))

        assert_that(received_content, contains_string(self.note_content), SpecLevel.MUST, InteropLevel.PROBLEM)
        assert_that(received_content, equal_to(self.note_content), SpecLevel.IMPLIED, InteropLevel.DEGRADED)
```