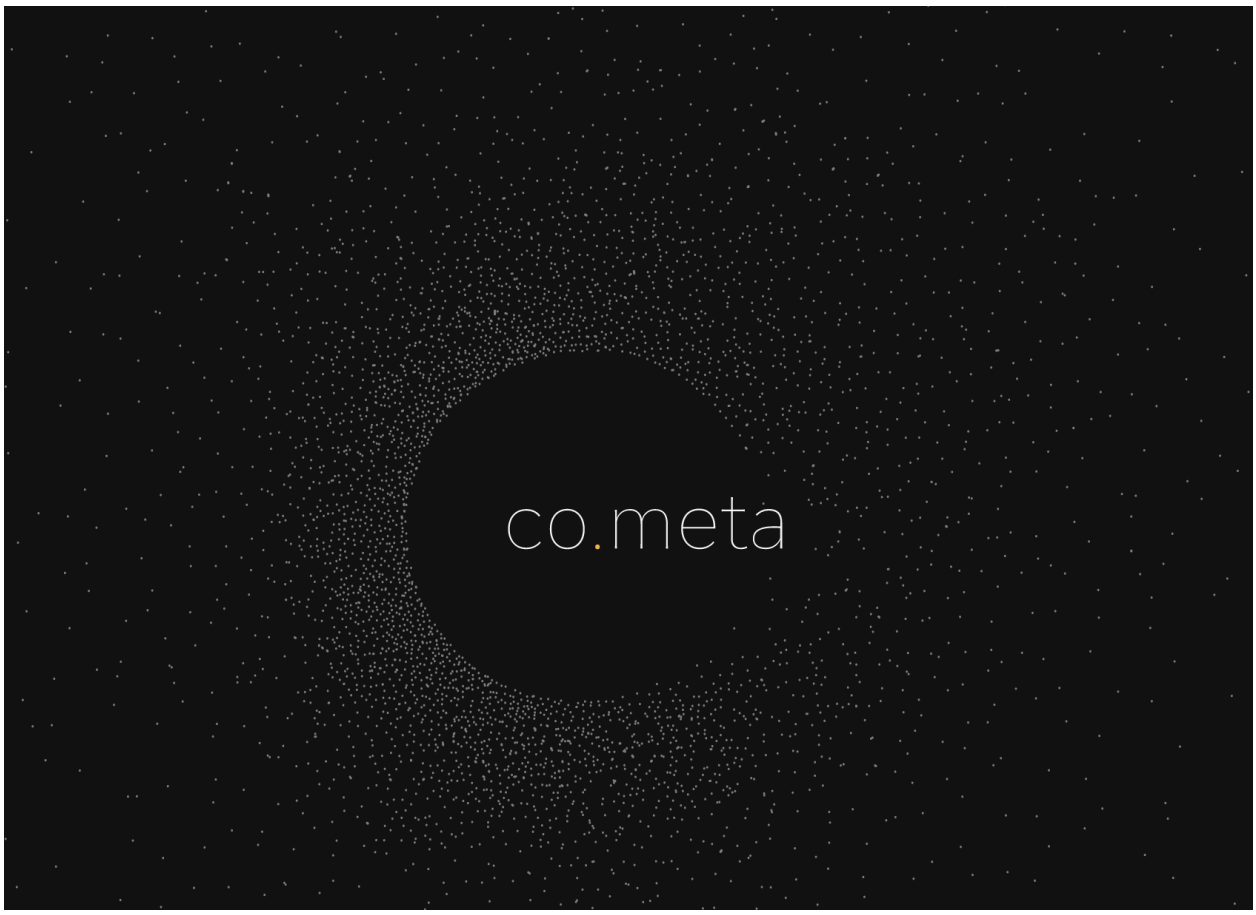
Tea-time with Testers in association with Amvara, presents

Rejoicing Poetry

Floating deep in the co.meta galaxy



The big purpose

The purpose of this report is to provide professional assessment of the co.meta tool so its stakeholders can decide if it is ready for the market.

This assessment has been performed by Rohit Kadam and Lalitkumar Bhamare on behalf of Tea-time with Testers magazine. Their assessment and opinions written in this report are with the sole intention of providing honest, authentic and professional opinions aimed at supporting co.meta to succeed in the market.

The overall impression

We feel positively overwhelmed with the wide range of features and numerous possibilities co.meta has to offer to solve various problems related to automation in testing.

Writing UI automation is very easy with cometa particularly because of its no-code offerings and a lot of thought given to different possibilities by the makers of the tool.

We are generally impressed with the capability this tool offers. Dig deeper and you will always find something more to explore.

Here is the highlight of some of the cool features we liked the most:

1. Once onboarded with a tool, one can write tests within minutes
2. Selection of OS/Browser combination and parallel run support
3. Visualization of features when tests are running
4. Quick video access after test run next to feature
5. Screenshot readily available next to step when running tests
6. In built cron support
7. Feature steps reusability

In order to find the answer to the question if co.meta is ready for market, we decided to take a deeper dive into its multicolored universe. We liked what we came across, in some cases we felt delighted. And there were some things we felt could be thought upon. The detailed assessment below primarily discusses our experience with the tool and things that we would like to bring on the table for the discussion.

Detailed assessment

The detailed assessment has been done considering key elements listed below which we think decide about overall quality and sustainability of the products we test i.e. Features, Complexity, Claims, Configurations, User Personas, Scenarios,Variability, Interoperability, Data, and Structure. We have clubbed together some criterions where we found it necessary.

Features:

We took a tour of various features co.meta offers and here is how we feel about certain things and our recommendations around those:

1. Landing page/UI talks about “folders/features” etc. There are folders and also features. Then inside folders, there are features.

   It would be useful to have a well defined structure and hierarchy to make it more intuitive for the users. This way they can map the structure of the tool with their mental model of how they want to organise their tests/suit. For example, what does “feature” mean in Co.meta name space? We assume it is a TDD/BDD terminology of a feature file that co.meta wants to emphasise on?

2. It would be nice to add a short “onboarding video or step by step walk-through” for first time login or keeping it available on-demand

3. Step definitions lack the “Given, When, Then” structure which can help users establish a logical flow while reading the test and also for deciding the next steps.

4. Download PDF is actually a feature report download which is impressive. Could be named accordingly for better understanding of the feature. Could be considered as a “Feature” for bug reporting i.e. When a test fails, the feature report (with all steps and screenshots) can be readily used for bug reporting by integrating with bug-management tools.

5. Playing the recorded video option is very useful and very impressive.

Claims:

Purpose of this tour was to verify the claims co.meta makes so that we can know if it is positioning itself in the market to compliment it all

1. Co.meta claims to be a Complete Meta Test Automation but there is no explanation anywhere why it is so, what it means and how the tool really delivers it.

2. Co.meta claims to be no-code, low-code which it fulfills but the structure of the tool reflects a lot of TDD/BDD influence in its design which might be confusing for people who are not familiar with it e.g. POs or Project Managers.

   Following the structure and operations of the tool becomes easy for those who are familiar with TDD/BDD but for non-technical people, this might become a learning curve and may deter them from using the tool.

   Operations:

   1. Copy pasting with the keyboard (Ctr V) does not work.
   2. Maybe creating a new feature from inside some project folder should have that default project selected?

3. Intermediate steps which are apparently re-usable steps, are displayed as ”Run Javascript function” which is counter-intuitive and not very inline with no-code ideology.

Complexity:

Tools with a lot of technical depth and innovative offerings tend to become complex without the makers realising it. We tried to find out how co.meta delivered on our complex tour.

1. Co.meta is loaded with extraordinary features that enable users to achieve a lot with bare minimum coding efforts. However, the structure of the tool is complex and can be simplified further so testers can know its real capability and how to use it.

   Our current impression is that there are lots of possibilities that we could explore and we are not sure if I have explored this tool enough since we feel there is still more to discover. This feeling can be overwhelming for the first time users.

   Co.meta is a very powerful and very elegant tool, like a giant intelligent elephant. It would be useful to slice this elephant down for easy consumption for day to day usage.

2. Local installation seems equally complex and requires some technical onboarding but we would leave it out of scope since co.meta brands itself as a no-code tool.

3. The dashboard with lines on the X-Y graph looks beautiful and it has useful information but the plain “graph-like” structure adds to its complexity. To make it less-complex, it could be made more readable by presenting it differently e.g. bar

charts? The suggestion is to present the dashboard in such a way that a person does not need to dig deeper to synthesise it, and understand it as it is being seen.

4. There is some redundancy of features which can be easily utilised to show more capabilities of the tool. For example, there is a video icon in seperate column and again “show video” is an option under … menu.

5. Too technical terminology of features adds more to the complexity of the usage. For example, “Pixel difference” column implies that there are differences in images. How major or critical are these differences? Does higher numbers mean bigger risk? How exactly does this information translate to business risk? Could it be presented in a less technical way and more “business-language” way..?

6. Scheduling feature is great but when editing it is not very clear that edit more is already on.

Configurations:

Tools become more useful if they have the possibility to change the configurations where needed. Particularly in case of no-code/low-code tools, finding the right balance through configurability becomes even harder.

Currently there are not many options for changing configurations in co.meta to serve various objectives. Only option we could find was about scheduling which is a very useful feature.

We think these configuration options could be made more prominent and visible.

Users/Persona:

We tested the tool with three key personas in mind:

1. Automation Engineer
2. Test Manager
3. DevOps Engineer

The tool offers a number of great possibilities and flexibility for the automation engineer. However, we think there is a scope of improvement to make co.meta more useful from a test-management point of view.

Features like ***improved dashboard/reporting and tying it with business-risk*** can be a unique feature for co.meta (which most of the automation tools miss out on).

Similarly for DevOps Engineer, it would be helpful to have more visibility into integrating the cometa tests in CI/CD flow, visualising it, mapping it with particular PRs, easy of debugging the failed cases and root cause analysis of the same etc. would be nice to have and if it is there, Cometa may want to make it more visible to the users.

Testability:

The tool has been very neatly developed and we did not encounter major problems while testing it per say.

Visualisation of tests while they are being run adds great advantage to the testability of Cometa.

Interfaces:

This aspect was kept out of the scope of review since it would require more time and infra support for meaningful evaluation of cometa integrations with external apps.

But looking at the strong technical foundation of the tool, we believe it will excel on the integration parts.

Structure:

This is where we spent most of our time. In the structure tour we tried to focus on everyday use of the tool, code of the tool, how it caters to various use cases, interfaces and scenarios. Below are the observations/open questions/recommendations we have listed and we would be happy to elaborate them all on the live-walk through of this report.

1. Tooltips for icons on websites would be helpful (had difficulty in figuring out from where to add a feature file)
2. Showing suggestions when using env vars would be helpful.
3. Substring matching of the step definitions from the whole step (e.g. I ... filter)

4. How do we add my own step definition? E.g. If I need to write a very unique product specific step which could be - `When I select 3rd element from the list with css selector and store it as $VAR `? (current way is not easy to scale)

5. In search functionality we tried to assert that there should be 2 occurrences of given cssselector? Which step can we use?

6. When you're on a Feature view, how do you navigate to your space? E.g. from https://prod.cometa.rocks/#/Default/Test/315. I used the browser back button.

7. Bit of struggle with structuring the tests. When developing tests for a website, we structure tests with specific sections, common steps, helpers etc. and as we're heavy IDE users, we had to struggle a bit.

8. Copy/Duplicate step when writing feature file. In some cases users might want to click on 4 different buttons and can use the same step with different css. Copy functionality would be useful.

9. File upload support?

10. If there are limitations on the user's staging server where we can run only `x` number of tests and we choose to run tests in `n` batches. How can we combine reporting of those `n` number of runs?

11. Is there a rerun facility only on the failed combination? In rerun we would expect minimal manual intervention.

12. Would it make sense to have some reusable data structure to store locators instead of storing it as ENV var? E.g. in page object model we define locators and page actions in certain classes and it is easy to find for future usage. With Big/Complex websites having 100 Env vars would make it hard to find? Or is it already possible and I am missing something?

13. Can we use cucumber features like Example Scenarios where we can run the same scenario on different data? E.g. If we are checking Search functionality with different inputs and relevant outputs.

14. Sometimes, you want to try different ways to interact with an element or test with your identifier (e.g. Move the mouse and then click or use tabs and then click etc.) If this step is way down in your feature file, you have to rerun all steps and it is time consuming. Something to play with interactively would be nice.

15. “Click on element with text step” missing? XPath can be used.

16. Search text on the whole page with scrolling?

17. How is the version control system integrated if some team wants to use it?

18. It is difficult to visualize how development of the tests by multiple people, execution of the tests on multiple environments and respective reports will look? Users might expect easy, transparent navigation though all of the points mentioned above.

19. On the fly data creation support(e.g. API calls, shell script execution)

20. How about importing some third party libraries?

Guess we found some bugs on the fly:

1. Running Feature without any test and downloading pdf returns 500 error - https://prod.cometa.rocks/#/Default/Test/315/run/30316/step/44679

2. When the “Edit Feature” modal is open and you press the browser back, the modal view is still visible and pages behind it navigate through history.

3. With this feature - https://prod.cometa.rocks/#/Default/Test/329 test run was stuck at 00:01 seconds with spinner spinning and “Initializing Feature” message. No error message was shown and Feature Log output had no entries. Maybe this was because the first step of the feature is NOT to open a browser.

4. Incorrect syntax highlighting should be available e.g. By mistake I added an extra double quote in one of the steps. Highlight of such syntactical errors should be shown when the user is writing new tests.

Conclusion:

As per our assessment, co.meta is definitely a tool testing industry can greatly benefit from. Not only that but we also see it as a great tool to involve even non-technical members of the team to support automation. What a tester can achieve with this tool presently, possibilities of that are many. And this can be a double edged sword for the first time if not presented properly. Our key recommendations would be:

- To make the design of the tool (as a product) more intuitive and easy to consume.
- Investing in creating detailed documentation of all the offerings, features, work around, supporting documentations etc would make a big difference (e.g. Cypress).
- Also, if the information from dashboards or pass/fail reports could be tied back to business-impact/risk then this tool will make history in the time to come.