Refactoring Pairing Session

Date	@October 9, 2025
Attendees	V Vitor Queiroz Borislav Dzodzo B Brad
Created by	V Vitor Queiroz

Summary

Meeting Overview

- Team reviewed code refactoring work done with Large Language Model (LLM) assistance
- Focused on Step 4 (weather data download) implementation and test structure
- Discussed proper organization of code between repositories and step files
- Identified improvements for LLM prompting strategies

Code Structure Analysis

- Examined Weather Data Download Step implementation which uses multiple repositories:
    - Weather API
    - CSV file repository
    - JSON file repository
    - Progress Tracking repository
- Determined that the current Step 4 implementation should be refactored into a repository
    - Step 4 primarily retrieves and formats data for use in Step 5
    - This logic belongs in a repository rather than as a standalone step

Testing Issues

- Current tests only mock dependencies but don't test actual execution
- Tests fail to call the "execute" method which would run the step code
- Need clear standards for test structure similar to existing step structure documentation

LLM Guidance Recommendations

- When asking LLMs to generate tests:
    - Request test code organized in the same sequence as feature files
    - Be extremely specific about execution testing, not just mocking
    - Ask LLM to show its plan before implementing
- Create a dedicated MD file documenting testing standards for LLM reference
- Add a refactoring step to remove code for deleted/unnecessary scenarios

Code Organization Standards

- Create a "utils" folder for non-step components like factories and extractors
- Keep step files only in the steps folder
- Standardize file naming conventions

Action Items

- [ ] Create an MD file documenting testing standards for LLMs
- [ ] Refactor Step 4 logic into a repository for use in Step 5
- [ ] Implement better organization with utils folder for non-step components
- [ ] Develop more specific LLM prompting practices

Notes

Transcript

Okay, hello? Hello.

I don't know if you can hear me. Yes I can, I can hear you. I can't hear you though. You can't hear me? I can see the... That's... No, no, no. Uh, hold on, uh, maybe I was on mute. Okay, try it again. Yeah. Yeah. Yeah. Got it. Cool All right, because that was the office right and

I don't know, did you talk to Brad already? No, no. No, no. Oh, there's a... There's a... There's a return. There's a return. I can hear my own voice. Still? No, not anymore. Okay, cool. All right. Okay, so no, we just we just postponed into three Uh, so everybody's the same

The same conversation. Okay, uh, I just pushed what I did to my own branch for my own ticket. Which branch is that? Wait, before we do anything, let's make sure we're recording. Yes, I'm already transcribing. This is fantastic. Okay, which branch? It's 130 right? Yeah.

Let me see. Did it get created? I cannot see here the branch 130. I see 129 and I see 122.

Of course. Maybe, maybe did you push it to remote? 1.30 I had recent pushes one minute ago, so it could be outdated. For all intents and purposes, the AIS-129 one, I pushed into that one by accident, so I need to roll that back or find a way to get it done.

I haven't rolled that back yet. I was supposed to push it on step 2, but because of a bug, I pushed it in the wrong branch. I need to roll that back. Okay, okay. Can somebody roll it back for me? Just, just... Just wait a bit, because you cut Boris. Boris was talking. Sorry guys, I needed to get headphones because the volume is too low. You know what the problem is.

OK, OK. All right.

All right, let's keep going. Okay, so you push to 100, 130, right? I can see it now. Refactoring step 4. Okay, cool. Yeah, it's a fresh new branch, yeah. Okay, so bread. You're saying that you revert your branch or lost something, right? Okay, 1.29 has been integrated to main so that one should be closed.

I pushed it to the wrong place. Currently, I pushed it to step-2 instead. Step-2. And then, basically, that one, currently, because of some bugs, step-2 has issues, so I need to keep at a steady pace. Okay, guys, every time that you merge something to main, delete the old branch. Otherwise, the branches keep accumulating.

Okay, yeah, so all right, I'll delete the old branches for me. Okay. I'm just paranoid that somehow I messed something up. You know, that's why. Okay. I'll delete it and I'll keep it on the local just in case. Okay. Gotta. No, no, no. The thing is, it's not yet fully fixed. I'm currently double-checking everything. Okay. I'm just like, uh, there was some issues. Okay.

Alright, alright. So, okay. Let's go through what Boris did. I got Boris' email. I read it. Cool. I just want to see on the code. Yeah. Oh, I can't explain anything. I'm sorry, I really... This is the first time that I'm seeing it too. So it's a little bit of a... Yeah, it's... This is like...

Yeah, I'm sorry. I should have probably spent the last hour differently, but this is I was literally giving it just high-level commands Okay to do this

So, uh, well, I can, we can see by the files that got created, uh, let's see, so scripts. And... no, not scripts. Where are the refactored steps? Source...

I mean whether data download step whether data factory because we we are putting this the factory steps on steps

So source, and then steps. Yes. Yep. And this is some of the stuff that got created here. Weather data, let's see here. So the new files obviously are for these last two weather data download step and weather data factory step or just weather data factory which

may not belong here because it's not a step. I don't know how we deal with the utility things. But that's that's the two files that got created. Okay, so you have, okay. Let me just open, let me just share my screen then. It's easier for the way we follow in talk.

Okay, so now we have the weather data download step. So you have three, you use three repositories, Weather API, CSV file repository, JSON file repository, and Progress Tracking repository. Okay, okay, okay. What does the weather API etc. Let me just check the repositories.

Okay, so we have weather API, CSV file ready over there, JSON file repository, and progress tracking repository.

Where does this come from? Oh, where is this? Okay. Where does this come from?

I need to...

Hmm it loose ends. Okay, so this looks terrible for me. No, no, no, no, it's not It's not terrible. We're just trying to understand Where this guy is? Because I know you imported it and you're using it. I just don't know where the definition is. I can ask it.

Where is the progress? Let's see, if I do this, type search, okay? I can look on GitHub.

Okay, I cannot find this file here.

Okay, so we have a repo here that we use.

So we have all this report in this progress tracking repository and I cannot find the reference for this guy so I'm not sure if this guy exists, but let's go. Maybe it exists at one point, or maybe it's inside another file. So that's why it's hard for me when I see something like this, for example.

If I open this ssv repository, you have so many repositories in one file that it is really really hard to navigate and see where the definition is. I personally like more to have one definition per file. So probably this guy here, probably we're going to need to break it down in different files. But I'm not saying that that's your case.

Okay, so let's see here. So you have some data, the period info, step config, okay, configuration step 4, all right, cool, cool, cool. Download stats, weather, data. This is the step. So you have the constants. Okay, cool. Init, blah, blah, blah. Then you have setup.

So, what does setup do? Load coordinates and initialize progress tracking.

Generate years over years period, okay. Okay, you get the progress repo and you load the data from the progress repo. And then you pass the coordinates.

Okay, cool. You get the coordinates repo. Okay, so you get all the data. All right. You pass forward. Start time date now and then in apply you do download weather altitude data for all periods. Okay cool.

Validate. Validate downloaded weather and altitude data.

So, you are basically validating if some files are missing or not in this one. Check altitude coverage. Weather data quality, progress tracking, track integrity. Okay, so if everything is fine. Oh, this is something that I add in my Breaking some validation, it will throw an error, right? And then they will stop and give a log. Hey, OK, I could not proceed because.

This guy here is missing something. So this happens in every validation that we've run so far. That's the strategy. It's not just log the validation, but stop what you're doing. Yes. Okay. Because something is wrong, so it cannot move forward. In the half-persist phase, where you just get everything that you...

Okay, in the persist phase, you save altitude.

You save progress.

OK.

All right, and this is the private methods.

Oh, so here, what you do is like, you get the, um...

I think that the target year amount that was passed, otherwise you have this default value. Any reason to have this default value? No, no, I had, I mean, we're already going into more detail than I have. Yeah, so... It's it's now. Okay.

It should probably fail there. The guy is not being captured. Yeah, the failure is not captured yet. Okay, cool. Be the year of the topo. List of periods tuple and this one. Okay, period info. Okay.

Okay, get last any months period. Okay, collect altitude data. All right.

Okay, so this for example is...

So, when you do collect altitude data, download period with VPN support. When you do download period...

download download weather for store

It should, it sounds like, I mean, download weather sounds like it's downloading it from the API. Fetch weather. So all of this, this logic. Because you have a repo that downloads things, right? This logic here, that you are using to make sure that things are downloaded,

This probably should go inside the repository because this is all logical for retrieving data. It's not to process data. But this is something that you can ask the... You can ask windsurf to refactor input all this method that's related to download data and logic for download data inside the

The repository, and then it will do this for you. Just move, just move forward to there. Okay, save weather file. All right, cool. Because when you do this... It's way easier to test because it will not try to mark all of this because the whole idea is you have a repository and you ask them, give me the data.

How are we going to capture the data? It doesn't matter for this code, as long as the data comes. So, when you were testing, you would just test. The code for apply and validation. All the rest of the code, right, all the retrieving data and saving data, it doesn't concern your test.

You just mock that thing, right? Cool. Yeah, it's just a strange situation that this step is all about retrieval in the first place. So I think that's where the confusion comes from. I think that's why the LLM is a little bit confused. It thinks that that's the process, that's the apply.

This step, it returns weather data for a specific step in the future. or for several steps in the future.

Because if this is just retrieving a step, retrieving this data to one step... In the future, maybe this whole thing here is just a repository. It doesn't have to be a step. And that is called inside the setup of one step. Okay, let me just, okay, hold on a second. So, first of all, the progress tracking repository I checked, it's in the source repository's JSON repository.py. Just like you don't like it. It's exactly like you don't like it. Okay, okay, okay.

And the question, let me just ask the Windsurf thing. Hold on, what's the question again? My question is. When you download this data here, right? This data that's downloaded. Why do we download? Where do we use this data?

The data is actually used downstream in order to create temperature-aware clustering. So it's used in Step 5 basically. In fact, Step 5 uses this data together with some other data to create the real weather-feeling data. So this is just the statistical weather.

But then there is like weather that takes into, or the sense of temperature that involves the feeling of humidity, wind chill, all of this other stuff that can make your sense of weather different. So that's computed. The real feeling of the weather is computed in step 5, and then that's used in step 6 for clustering. Okay. So, for example...

If this step is just to retrieve and format data, and then you save this data somewhere to be used in another step. Maybe this whole thing here is just a repository that is called in step 5. You know what I mean? Oh, so we skipped a step. Yeah, you just transformed the whole step in one Ripple that retrieves and formats data and shares it to you. And then, that Ripple is just called Step 5.

Or any other step that needs retrieved data in the future, right? Retrieving weather data. Yeah, weather data in the future, yes. So maybe this is how it could be, right? Yes. This is why there is this whole confusion. We talked about this earlier and we were thinking that...

The logic of download should be all in the setup, but actually it should just be in the repository. Yeah, yeah. I mean, it eventually be in the setup of step 5, but what will happen is it's just going to be in the repository. Right, okay. Yeah, yeah, okay. So we're not, we're, ultimately we're not, we're factoring step four and five together, so, okay. Yes, yes, exactly, right? So maybe it's in the...

You can do it the way it is right now, when you are refactoring step 5, then you come back and say... You know the step four. Put all that logic in a repo and call it in step five setup. Yeah, got it. Let's establish some rules. So, in the folder steps, you just leave steps. When you have extractor or factory

Let's create a folder that is just utils. It could be inside source, so you migrate things to that.

Of course, just the definitions, okay. Yeah, so with that, things like coordinate extractor and matrix processor and weather data factory or SPU metadata processor. It's just gonna go to the RTOS file. Can you just pronounce it for me? I don't... Yeah, because for the sake of syntax, let's just leave steps file under the steps folder. Right.

I get it, I just want to know what's the name. For the rest, put it in repositories. The rest, put the logic, like, step... step and stepfile, and then the repository is what I usually do as, like, absorbing all the other logic and repositories. It should make it easier, right?

No, I'm not talking about the repository. What we're talking about here, we have other things that are not steps. You have a factory, you've got a factory, you've got a factory. Create a Autos folder.

Oh, utils. Oh, okay, utils. You choose folder, and then put those things there. Yeah, makes sense. Okay. So it makes it, gives it, give it clean. This guy here, apdownloadmerge, is my fault. I should append with an underlined step, because this is a step.

Okay, this is step one. Yeah, yeah I wonder if that step should actually also be in the report. In the repo, yeah. It could be a repo. Okay. Okay, right. What are the tests?

I really relied on this thing heavily, and I'm surprised that it's passing as many of your sniff tests as it is. Um, alright, so the main test file is test underscore step four underscore weather underscore data dot pi. Tests, uh, which way? Press step 4 weather data. Test step 4.

weather data. Wait, we have the test folder and it's in the site inside the test folder? Let me just copy and paste the response, okay? Because it's easier.

Okay.

So that's the... Oh. Type step definitions. Okay. Step definitions. There's a couple of things. There's... There's a couple of things.

Sorry. GitHub.

S-t-e-s-t-o-r-r-y Tests Step definitions, step four.

Test step for weather data. Click OK.

It may not belong there, but that's where it is right now.

You have the fixture, okay, okay. Okay, so it means that you probably have features. Which is step four. This one. Okay.

Now my question is, does this scenario make sense?

So, even in your instructions it said that LLM should make those, but like, I have not reviewed it. Straight up did not review anything. Okay, just gotta be honest with you. It's okay I just wanted to see this was for me an experiment to see if this thing could create anything that made sense Okay, and so I got done with it and then Andrew

He asked me to do some stuff with data science, and then I kind of wasted an hour at the office and came here, and that's it. It's okay. The whole point is we ask the LLM to generate and then we check if it makes sense, right? Because the LLM is just reading a code.

Based on the code there is, it is creating some scenarios for the code that exists, right? The code that exists may be wrong. It does not make sense. So what LLM is this here, it explains to you what the code does. Yes, and I can see why it had that problem with a default value because it's right there.

Yeah, so when you read this, okay, because sometimes if the code is too messy, it's hard for us to understand what the code do. But here it describes, so you go here and say, oh, this makes sense, oh, this doesn't make sense. This doesn't make sense, right? So the step after is you come here and you change the scenarios or remove things.

To things that make sense, I'm going to say now, please refactor the code to make the scenarios The LLM will check here and try to make the change on the code to make the scenario pass, right? Once everything is passing, you just ask to remove code that's not used in the scenarios.

And then it will remove. Right, okay. So, you have here etc, you have the steps. Fine. That's step four. We need to find a standard of how to name those files, because now different ways of naming the file is confusing. But that is something that we can do later, okay? Okay, so let's see what it does. So here, for example, right, he's mocking. He's mocking everything. So okay, it's not calling anything. It's just mocking.

Okay, okay, okay, Jason. Let's see what it does here in the test.

Given, given, given. Usually it acts because everything that's given is preparing the test, right? I'm putting the data in place. And the run actually happens on when.

That's a lot of events. Oh, yeah. Okay, so... Suggestion, right? What does it do here? It looks for this guy here in that file, which is a text file, the feature, and it tries to run. How does it know? Because here you have all the givens in one place.

and all the whens in one place. How does it know when to run this given and this when in sequence? It looks at that file and it knows after I run this given, I run this when. So I come here and run. What I would recommend is asking to LLM to organize this guy here the same way the file is organized. So it will be something like this.

Because the first time that I generate, it creates this as well. So it's organized by the background step scenario. So this is the first scenario. So the code is organized in a sequence that's called. So it's easy for me to read to validate the test So you need to go to ask him or for now one every time that you ask him to generate

The task for that file, you ask, organize the code the same way the feature file is organized. And then it will put things in sequence, so it's easy for us to read. Not to aggregate all given steps in one place and all when steps in another place. I see, okay. Let me just see... We have an issue...

Okay, let me see here.

OK. Not execute. Oh.

Remove the recorder because this guy is...

So, what is it doing now?

It's not testing anything.

Because for him to test, the test needs to call the method of the step the method execute, right? Because if you come here... Let's take a look on steps, on core. So, this is the structure of the When you test, when you create a test, follow this template, right, you implement setup, apply, validate, and persist.

Implement these four methods. When you run Execute, it runs these four methods in sequence. So, when you're going to test the code, you need to run

method that runs that code. So if you go to the implementation of your test and try to look for this method, it is not there. None there. So what it's doing is just mocking a bunch of stuff and checking the mocking. It's not running actual code. Yeah, so that's the issue.

With LLMs, right? It looks, it looks, you ask them to implement the test and mark all the data, it will, right? But you need to be very specific. That happened to me as well. When I was using LLM to test this guy here, I look at that and say, okay, it was there. I start to look at the code and say, but when is it calling?

Execute. Now you're just mocking everything in TestMock. You say, oh yeah, you're right. So, this is another instruction that you put. When you ask them to do things. So imagine LLMs as a five year old kid. If you don't say exactly what you want to do.

It would do whatever you think is right, and then it shows you that it did it. So, yeah, so this is a few things that we can start to... Document and say, every time you're going to refactor, you're going to ask things in this way. You make sure that to add this as well when you are doing your prompting.

So what's disappointing about this is that I actually told the LLM to look at the structure of the files that are analogues to what it did for SAPFOR. And I specifically said, pay attention that you didn't drop a major aspect of what Step 1 was doing. And nonetheless, it actually ignored me.

Because when you ask them, it will take a look on the step one code and try to follow the same principle. And it did. The issue is that when you ask to do the test, you know, it is not it is not the step one code. It's a test code. So maybe we can have, because now what we have here, right, what we have here to help is this one.

Let me just show the docs docs docs docs. What I have is this one. This one here that gives instructions of how to structure your code in steps. I use this one to extract mine. Maybe what we need is a similar one for testing. Yeah. Right. So we need to create an MD file that gives instruction to LLM how you should test a step.

And you just ask them, following the principles in this file, or the standards in this file, please... Test this implementation and share with him the implementation of your step. Mocking and then it's just going to read this. Just going to look to the implementation, and then it's going to create a test for you, right? So maybe we need to do something similar for this one, for a test. This is to create code.

We do want to create tests for the code. Yeah, to be honest with you, the more spelled out everything is, the more likely it is that it's going to do the right thing. Yeah. So I also saw that we have a lot of MD files here, right? Oh that's just, that can be gone. Yeah. That's just to show you

how the thing generates. I just wanted to show you. Yeah, yeah. I imagine it's step by step how they process the thinking. As it was processing everything, it was creating all of this stuff so that it wouldn't miss anything. That's all. But, I mean, now what's missing is to learn how to build the tests, because the implementation it created.

Right? And you split, and you create, and you put it there. It's just the case that this implementation, we don't need a step for this. We just need to put this in our repo. But it was a good exercise because you experiment how to refactor using LLMs.

Right? So I think it was super valid. Yeah.

One thing that I would like to share is, when I was doing this, I also... I wanted the LLM to explain to me what it was doing, because it started to change a bunch of files. Because it's the first time it ever did. So I asked him, do this, but before you do...

Show me your plan. And then it shows me the plan. So it makes sense. Proceed. Yes. Yes, actually, the strange thing is that... If you force it to pre-plan everything before doing anything, write it down what it actually wants, it's likely to catch its own errors in thinking. Yeah.

Because it uses that to validate what it's doing. As soon as it starts to do things, it checks. With the plane, if it's following the plane. Yeah, so pre-planning should actually be put into the procedure, actually. Yeah.

Okay. Okay. All right. Cool. So, this was step four, right? And it's using step five. No, step four is a stand-alone step. Step four does not depend on step five, but it feeds it. Yeah, yeah, exactly. That's what I meant to. Sorry. Step four, feed step five. So what I would advise you is like, do it in the same branch, right?

Take all this implementation, ask Stellar LAMP to create a repository with all this implementation and then you start to refactor in step 5. And use the repository in the setup. And with the step five, then you create the tasks and make sure that it's running properly, et cetera.

I can help to create an MD file for tests that you can reference. Yes, I'm actually quite behind in terms of understanding how the whole structure of everything works for this, to be honest. Like, when it comes to the details, I understand the big picture.

So yes, that would be good. So take a look of what it did with implementation, even if you're going to throw away a greater repo. Just take a look and see if you understand how it breaks down things. The scenarios that are generated. Yeah, you start to get used to it, to validate the scenarios, to validate everything.

Let's add one more step in the refactoring which is, after you remove all the scenarios that doesn't make sense, you ask the LLM to remove the code that Make that scenario pass. Because we're going to remove the scenarios that don't make sense, and LLM is going to remove all the scenarios, all the code that make that scenario pass.

So you've got a cleaning code that is outdated or doesn't make much sense anymore.

OK. OK. Cool.

Awesome. Do you guys have anything else to ask or to share? Not at the moment. I'm just going to debug Step 2 carefully. Very, very carefully. Alright then. See you tomorrow then. I will share the transcription. Cool.