Making changes to big codebases

Jumping around files and getting confused? Try the exploratory planning process.

1 Define & Refine

Determine what needs to be changed by reading the task, looking at the current behaviour in the application, and making a list if necessary.

Example
We're going to update the password requirements to include one symbol character.

A symbol can be any of these '!@£$%^&*'.

If one isn't given, the user should see a notice to choose another password.

2 Exploratory Planning

Investigate the structure and behaviour of the application, gradually building and refining a plan for the change.

Example
A diagram of everywhere in the application that password validation is done, including classes and flows.

"We'll add a new class called `PasswordValidation` with a method `validate` which we'll call in `UsersController` & `ResetPasswordController`. We'll need to update tests for all of those classes too."

3 Execute & Test

Test and execute the change, updating your plan as necessary.

Example
The actual code being written.

- ❏ Write tests for `PasswordValidation`
- ❏ Implement `PasswordValidation`
- ❏ Update `UsersController` tests.
- ❏ Update `UsersController` to call `PasswordValidation`.
- ❏ ...

4 Verify & Refactor

Verify your change does what it is supposed to, doesn't have side-effects, and delivers quality.

Example
Running the whole test suite.

Running the system and verifying the surrounding functionality works correctly, attempting to find defects.

Running the linter. Comparing the code to good code in the codebase and ensuring the change is well tested, easy to understand, and maintains good structure.