Scrum- agile roles

The project owner

Is the most customer forward role
In charge of determining the stakeholders
Needs and desires for the project and creating a backlog

The Scrum master

In charge of pruning and organizing the backlog based on the deadline
Organizing it all into a sprint schedule for the team

The tester

Ensures that all the code meshes well and still functions
Makes sure the end product is how the project owner determined is needed

The developer

Writes the code based on what is needed
Integrates it into the existing code if needed

SDLC - Systems Development Life Cycle

Focusing on a specific example of our user stories; displaying the top five destinations at the press of a button.

This was one of the biggest desires from the stakeholders, thus during the planning stage, while converting it to a user story it was listed as the biggest priority

Then we had to define it, for this it was pretty self explanatory, though it tied into another goal, of organizing vacations based on genres. Thus, on top of classifying genre, defining what the top 5 vacations spots were helped as the developer began to plan their code.

Designing and building, I'd argue were completed at the same time, if not simply close to each other, as this particular project was integrated into existing code.

Testing was paramount, as the desire of the costumer changed; the top 5 remained the same but for a more specific genre. It was in testing this was salvaged, all the while making sure that the new integrated code works with the existing code.

Finally, with deployment. Once all the goals were met, and all curve balls were dealt with, the final product were deployed combined with the existing code successfully!

Agile v Waterfall

The project would have gone very differently had we used the waterfall method. As we've discussed before, waterfall is a much more rigid method of planning, essentially SDLC in its most extreme form.

In the beginning, no code would be designed or looked at until every stakeholder desire was translated into a user story. Then code that could meet those goals would have to be designed before looking into the existing code, testing would only be an afterthought until every line of code for every goal was met.

This ridgid, step by step approach is useful, in some cases. As experienced with this project, we are working on an a project that is being actively used, thus an inevitable sudden change would have a much bigger impact.

Agile v Waterfall cont.

Waterfall is still a valuable approach that should be considered, though. It's structure allows each step time to develop thus leading to fewer problems. With how rigid it is, as well, and upset is easily mitigated and quarantined to its own processes.

All in all, in the long term a waterfall approach also lends itself to easier maintenance after deployment.

Agile, while not quite the antithesis, is fundamentally different from waterfall. It plays more fast and loose with its process, while still following a SDLC structure.

I, personally, have come to view Waterfall a good methodology for a new project, one where the team is working from the ground up; while Agile works well with amendments or modifications to existing projects, as it is more flexible to the more drastic changes something in current use might experience.

Reference

The Project Manager’s Guide to Mastering Agile

Charles G. Cobb. (2015). *The Project Manager’s Guide to Mastering Agile : Principles and Practices for an Adaptive Approach*. Wiley.