Ask me anything

5 questions
0 upvotes
<table>
<thead>
<tr>
<th>Task Description</th>
<th>Answer</th>
<th>Relevant Knowledge</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wordpress theme in PHP</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Front end for an app that allows users to buy crypto data in a non-programmatic way</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Can you explain the hierarchy of apple concerning conway's law?</td>
<td>No</td>
<td>Develop a contract generator.</td>
</tr>
</tbody>
</table>
How did you agree on what was to be implemented?

- The client provided a full specification: 3
- We iteratively refined the tasks every N weeks: 3
- We made it up as we went along: 0
- Other: 0
What was the most difficult part of the project?

- Changes in plan
- Client satisfaction with the project.
- Always changing requirements and expectations

The most frustrating part was that every time that the client saw a new “version” of the product, they had new ideas that they thought would lead to an even more interesting frontend. In the end, it was just a complicated non-user friendly mess.
What percentage of the effort in a typical software project is spent actually programming?
Why are object-oriented methods so successful in building long-lived software systems?

- Because of its distributed nature
- The possibility for adaptability, saves costs in the long-run
- Have a self-describing for someone new on a project
- They are very maintainable as the structure (if well-built) can easily be extended.
- It's easy to update
- Intuitive way of structuring a project
- Good maintenance
- Easy to add additional functionality
- It's easier to maintain these methods because they have their own responsibility
- Requirements are clearly defined in the objects and not in the customer's "technical words".
Last chance for questions