Cognitive-Driven Development Helps Software Teams to Keep Code Units Under the Limit!

Gustavo Pinto
Assistant professor at UFPA.br
Head of research at ZUP.com.br

@gustavopinto
How to reason about this?

How to test this?

How to spot the bug?

How to refactor this?
If we want to keep making changes ...
CDD is a **simple** approach for chunking and slicing
Cognitive-Driven Development (CDD)

- CDD aims to reduce the developers’ cognitive load during coding activities.
- CDD does so by posing a limit on the number of items developers could use at once (at a class or file).
The Magical Number 7

- We are only able to process 7 (+/- 2) units of information in short-term memory.
- As we receive more information simultaneously, we lose the ability to process it (and we tend to make mistakes).
Cognitive-Driven Development (CDD)

The Magical Number 7

- We are only able to process 7 (+/- 2) units of information in short-term memory.
- As we receive more information simultaneously, we lose the ability to process it (and we tend to make mistakes).
CDD in a nutshell

- CDD provides a **clear limit** indicating how much a code unit could grow
- Every class over the limit **must be** refactored
Online training platform by Zup
Composed by 5 services:
- Core service (Java)
- Search service (Java)
- Menu service (Java)
- Frontend Service (TypeScript)
- ML service (Python)
Online training platform by Zup

Composed by 5 services:

- Core service (Java)
- Search service (Java)
- Menu service (Java)
- Frontend Service (TypeScript)
- ML service (Python)
CDD impacts the **size of the classes**

- CDD seems to help keep the classes small, even as the product evolves (almost linearly)
- Developers concur that the size of the classes are probably due to CDD
CDD impacts the size of the methods

- Maintenance effort is positively correlated with method length.
- “Developers should strive to keep their methods within 24 SLOC”
CDD impacts the size of the methods

- 92% of the handora's methods are under 24 SLOC (6.8, on average)
- “Every unit of code is impacted, because we know what the limit is and what goes into that limit.”
CDD impacts testing code

- On average, a testing method has ~8 SLOC
- “I think there is a relationship because the complexity of the test can be seen as a proxy of the complexity of the code under test”.
- There are ~1.3 assertions per method (no method without assertion)

<table>
<thead>
<tr>
<th></th>
<th>SLOC</th>
<th>Methods</th>
<th>Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>7.6k</td>
<td>215</td>
<td>71%</td>
</tr>
<tr>
<td>S2</td>
<td>1.3k</td>
<td>41</td>
<td>61%</td>
</tr>
<tr>
<td>S3</td>
<td>5.2k</td>
<td>128</td>
<td>64%</td>
</tr>
</tbody>
</table>
## Cognitive Driven Development

### Findings

- CDD seems to help designing **small classes**
- CDD seems to help designing **small methods**
- CDD seems to help designing **small testing methods**
Cognitive Driven Development

Findings
- CDD seems to help designing small classes
- CDD seems to help designing small methods
- CDD seems to help designing small testing methods

Challenges
- Still requires manual effort
- We need better tools
- How to ease CDD adoption?