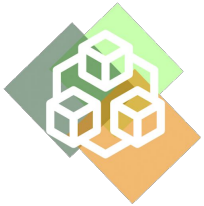


Johnny’s Software Lab

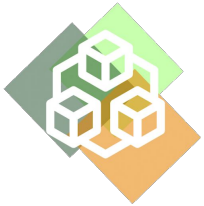
# Code Complexity

## *When is it too complex?*


 https://johnnysswlab.com,

 @johnnysswlab

 ivica@johnnysswlab.com

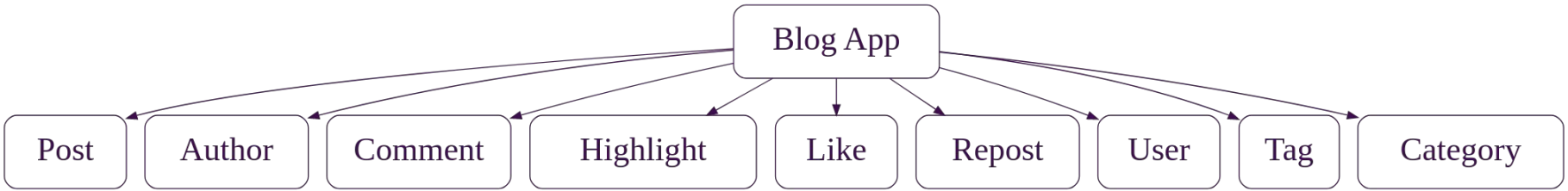


# Introduction

- Code complexity kills productivity
  - Difficult to understand code
  - Difficult to modify
  - A lot of time to get to know the code
- More difficult to deliver features
- More difficult to debug
- How do we measure complexity?
  - Many metrics

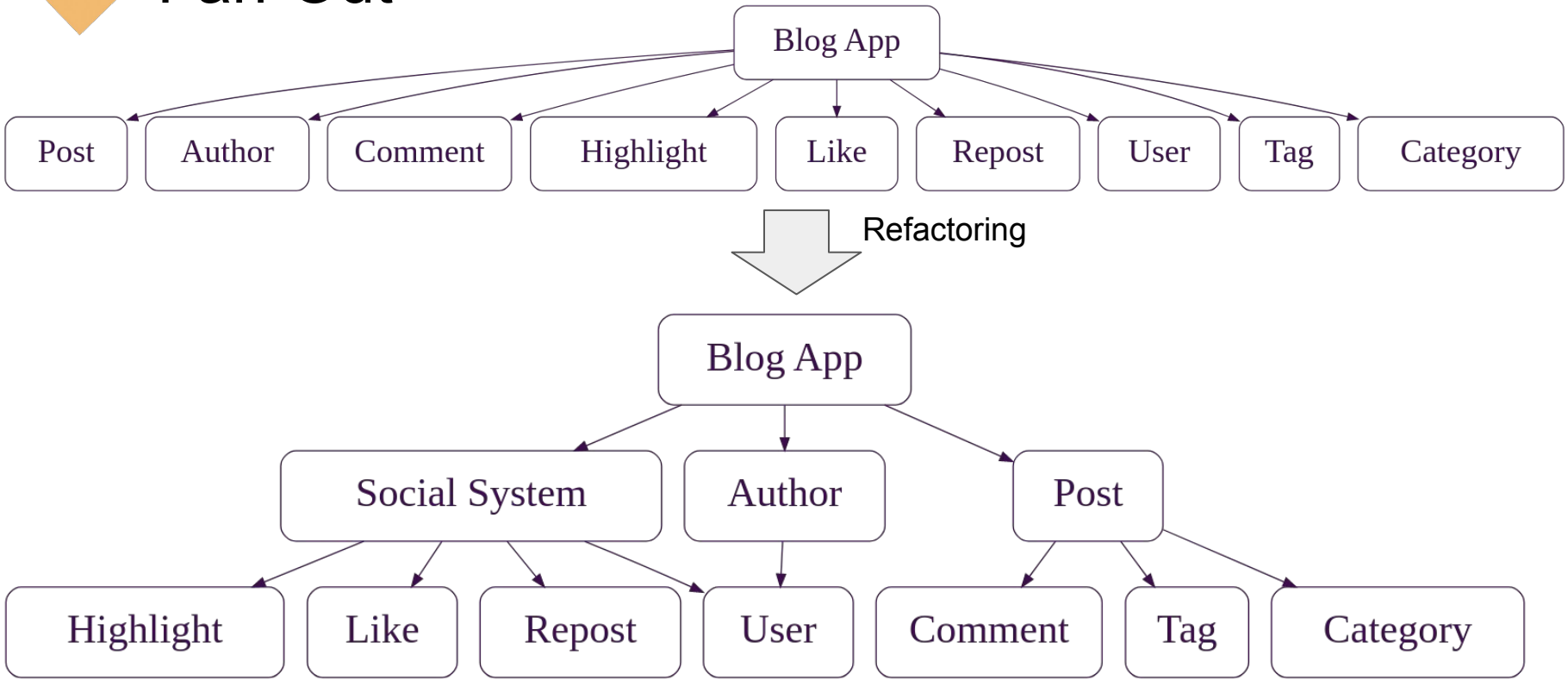
# Fan-Out

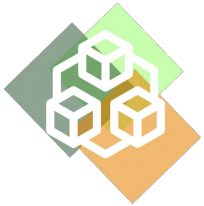
- Fan-Out - number of classes used by your class




- What are the good values for fan-out?

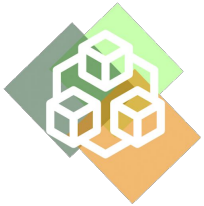
# Fan-Out


Blog App
Post
Author
Comment
Highlight
Like
Repost
User
Tag
Category
Refactoring
Blog App
Social System
Author
Post
Highlight
Like
Repost
User
Comment
Tag
Category



# Fan-Out

- There is not a perfect number, but too large fan-out makes the class to large and difficult to maintain
- Too small fan-out makes the class a wrapper class
- Large fan-outs are definitely places where there is *interleaved complexity* and that would profit from refactoring, i.e. separating responsibility into separate classes
- On experience, a recommended value for fan-out is between 3 and 5, although it can be a bit larger, e.g. 7
    - There are many projects that have classes with fan-out several tens or even hundreds of classes



# Fan-In

- Number of classes that use your class
- Example std::vector
- Useful classes have a large fan-in, and this is in general desirable