Recitation 4 Metric Discussion
Erdun E
September 26, 2024

CS 5010 Programming Design Paradigm
Recitation 4 Metric Discussion

1. Weighted Methods Per Class(WMC)

Keep each method in a class short and straightforward. For example, in the ProductionSysTem Class, methods like startProduction() and monitorBatch() are simple and easy to understand. My goal is to keep WMC low to make the code easier to read and maintain.

2. Depth of Inheritance Tree (DIT)

Make sure that do not have too many levels of class inheritance, as it can make the code complex and hard to follow. It's best to keep inheritance levels to three or less, to making the class structure easier to understand.

3. Number of Children (NOC)

For example, the classes that ProductionSystem and InventorySystem, keep the number of subclasses low, this prevents changes in the parent class from affecting too many other parts of the system. A low NOC shows that the class hierarchy is manageable.

4. Coupling Between Objects (CBO)

High coupling means that classes are too dependent on each other, making classes hard to test and maintain. For example, the ProductionSystem should mainly interact with classes like Batch and controller. Using interfaces can help reduce dependencies and keep CBO low, making the code more flexible.

5. Response for a Class (RFC)

A high RFC means a class is handling too many tasks, which can lead to more bugs and harder maintenance. For the ProductionSystem class, which manages the production process, the RFC should be moderate. This means it should only use necessary methods and focus on related tasks to keep things simple and organized.

6. Lack of Cohesion in Methods (LCOM)

Classes like Batch and Vat should have high cohesion, meaning their methods should work together on common attributes. If they don't, it suggests that the class's responsibilities aren't clear, leading to scattered functions. Improving cohesion helps the class work better towards a single purpose.

7. Weighted Attributes per Class (WAC)

Classes such as Recipe should only have attributes that are essential, like ingredients and instructions. Having too many or overly complicated attributes can make the class harder to maintain. Keep WAC low by using simple and necessary attributes.

8. Number of Tramps (NOT)

Methods in classes like ProductionController or InventoryController should only have the necessary parameters. Extra or unused parameters can cause confusion and mislead about a method's purpose. Make sure all parameters are needed and contribute to what the method does.

9. Violations of the Law of Demeter (VOD)

Classes should avoid calling methods through a chain of objects. For example, ProductionSystem shouldn't directly change the attributes of Batch or Vat objects without using a proper interface. Minimizing VOD helps reduce dependencies and makes the code clearer.