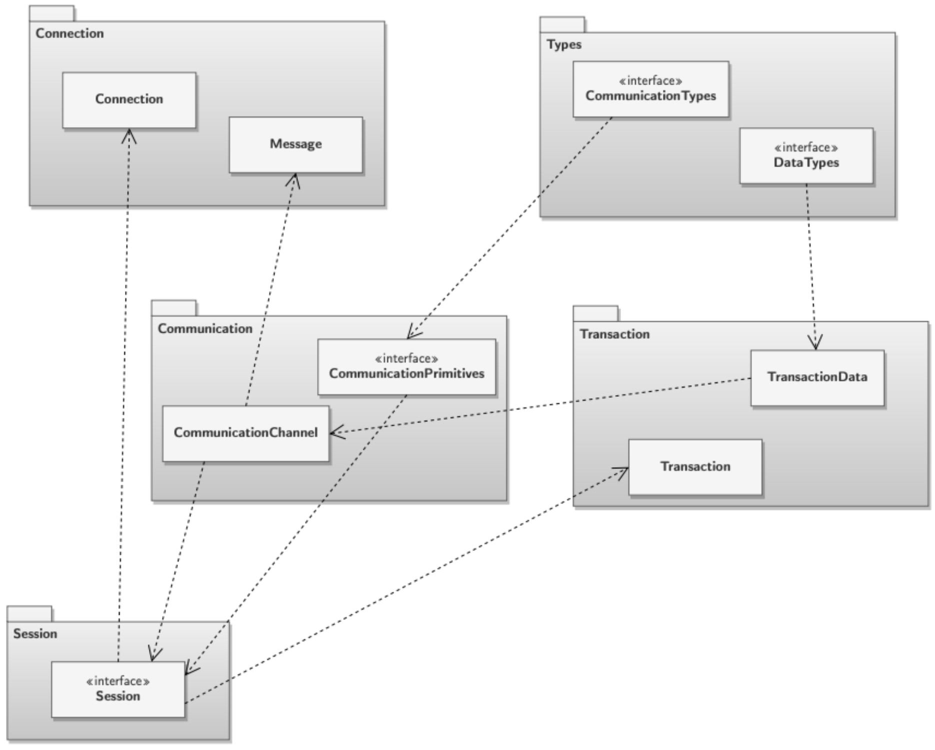
Lab 3: Measuring Architecture Characteristics and Component-Based Thinking

1. Your colleagues are working on a session-oriented communication system for managing transactions. The current iteration of the system design is provided below. To assist your colleagues in better understanding the structural characteristics of the system design, compute the distance from the main sequence for each of the packages in the design. Be sure to show your work and to identify any packages that may fall into the Zone of Uselessness or the Zone of Pain.



$$\mathrm{D} = |\mathrm{A} + \mathrm{I} - 1| \quad \text{where} \quad \mathrm{A} = \frac{\sum m^a}{\sum m^c} \quad \text{and} \quad \mathrm{I} = \frac{C^e}{C^e + C^a} - 1$$

$$\begin{aligned}
\mathrm{D}_{\texttt{Session}} &= |\frac{1}{1} + \frac{2}{2+2} - 1| = 1 + 0.5 - 1 = 0.5 \\
\mathrm{D}_{\texttt{Communication}} &= |\frac{1}{2} + \frac{3}{3+2} - 1| = 0.5 + 0.6 - 1 = 0.1 \\
\mathrm{D}_{\texttt{Transaction}} &= |\frac{0}{3} + \frac{1}{1+2} - 1| = 0 + 0.33 - 1 = 0.67 \\
\mathrm{D}_{\texttt{Types}} &= |\frac{2}{1} + \frac{2}{2+0} - 1| = 2 + 1 - 1 = 2 \text{ (zone of uselessness)} \\
\mathrm{D}_{\texttt{Connection}} &= |\frac{0}{3} + \frac{0}{0+2} - 1| = 0 + 0 - 1 = 1 \text{ (zone of pain)}
\end{aligned}$$

2. One of your colleagues is working a Java project where modularity has been noted as a relevant (and important) architecture characteristic. The project requires the implementation of a basic calculator functionality. You colleague is a new graduate and recalls making similar, simple calculator programs as part of their studies, so they create a simple program where the main method of the program takes two operands separated by an operator. These are used by the calculate method to perform the operation related to the operator provided.

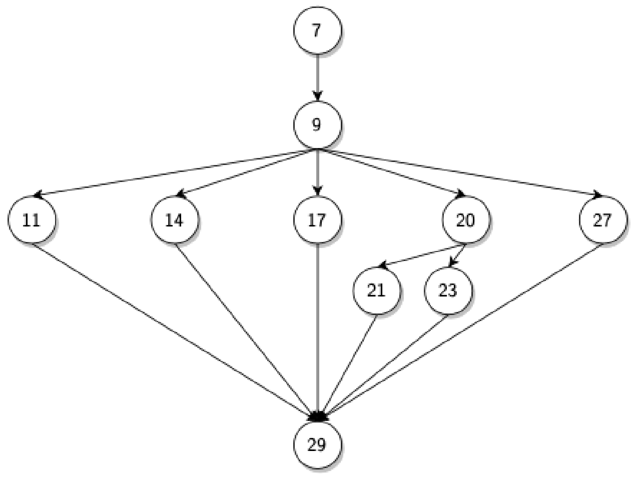
   Calculate the cyclomatic complexity of the calculate method. Be sure to show the control flow graph for the program code. Provide an interpretation of the cyclomatic complexity with respect to the architecture characteristics for this project.

```java
package calc;
import java.lang.IllegalArgumentException;

public class Calculator {

    public double calculate(double operand1, double operand2, char operator){
        double result = 0.0;

        switch(operator){
        case '+':
            result = operand1 + operand2;
            break;
        case '-':
            result = operand1 - operand2;
            break;
        case '*':
            result = operand1 * operand2;
            break;
        case '/':
            if (Math.abs(operand2) > 0.0){
                result = operand1 / operand2;
            } else {
                throw new ArithmeticException("Numerator is zero.");

            }
            break;
        default:
            throw new IllegalArgumentException(operator + " unknown.");
        }
        return result;
    }

    public static void main(String[] args){
        double operand1 = Double.parseDouble(args[0]);
        double operand2 = Double.parseDouble(args[2]);
        char operator = args[1].charAt(0);
        double result = new Calculator().calculate(operand1, operand2, operator);

        System.out.println(operand1 + args[1] + operand2 + "=" + result);
    }
}
```

The cyclomatic complexity is calculated using the following formula:

$$CC = E - N + 2P$$

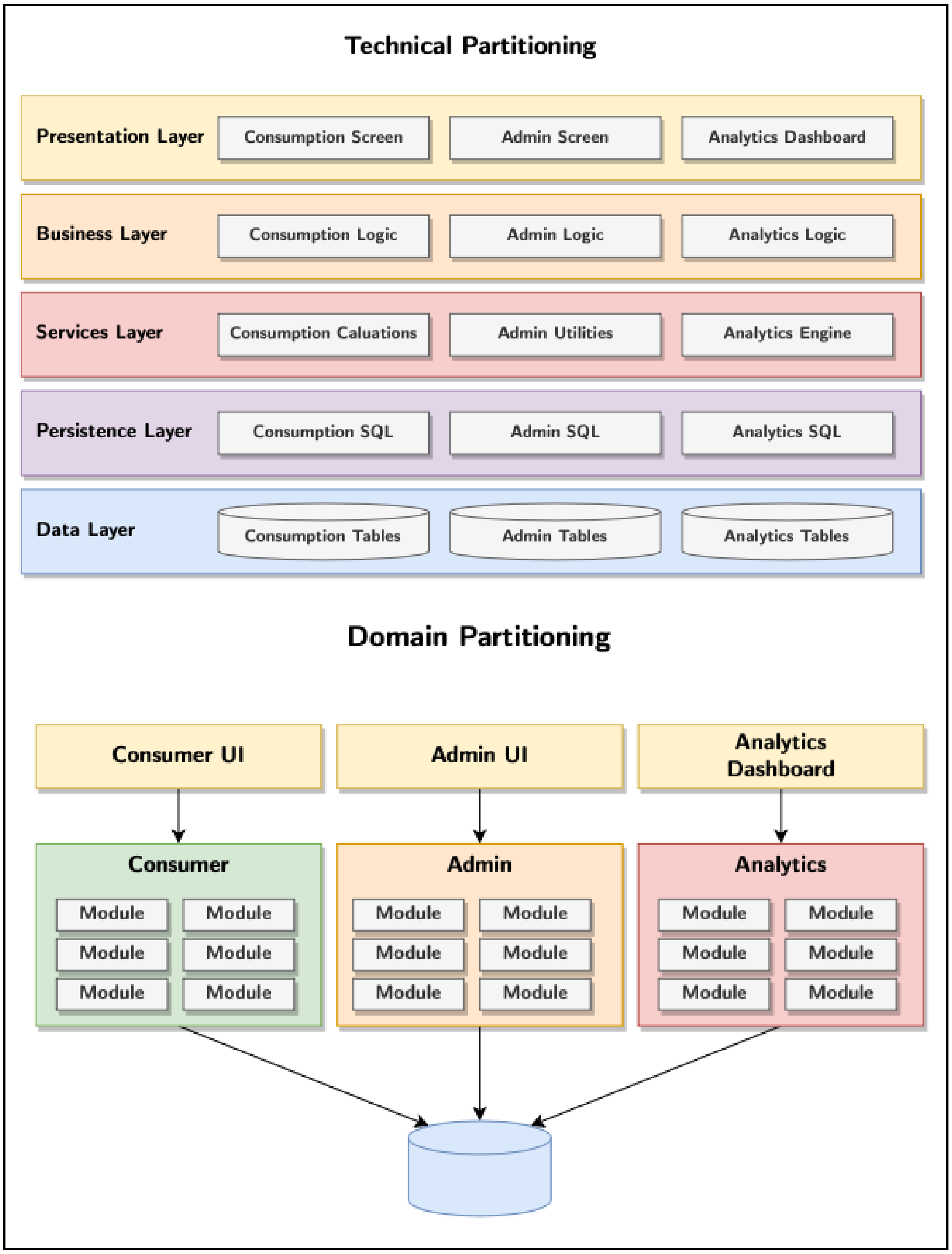
There are many different ways to construct a control flow graph from the provided code. One possible way is given below where the numbers in the nodes correspond to the line numbers in the code.



Given this control flow graph the cyclomatic complexity of the calculate method is:14−10+2= 6. For such a simple functionality, the cyclomatic complexity is rather high. This can indicate that there may be ways to refactor the design to reduce the complexity of this code.

3. You are taking the lead on developing the software architecture for a system that will compute advanced analytics for consumers of internet services within their homes. The system must provide a portal for consumers to view their consumption data, an administrator view to access specific utilities about the provided consumer services to the homes, and an analytics dashboard to visualize the data from the analytics engine. The system must accommodate the user interfaces, business logic, services, data persistence (which is expected to be a wrapper for SQL databases), and the data for the consumers, administrators, and analytics engine.

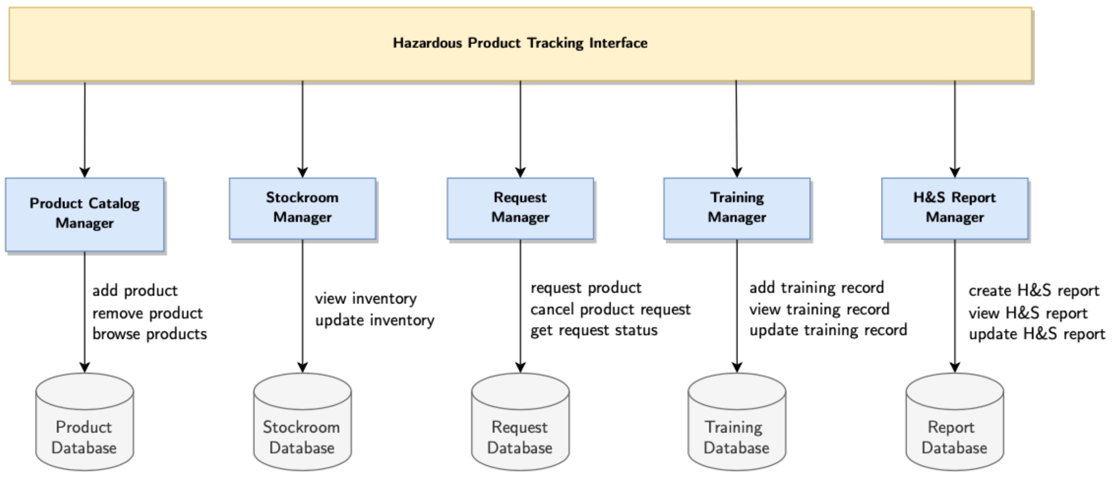
   Sketch two design alternatives for this system, one that is technically partitioned and one that is domain partitioned. Briefly describe the tradeoffs among the two designs you have proposed.



4. Your architecture and design team is responsible for creating a tracking system for hazardous products (e.g., pesticides). The system must manage a catalog of hazardous products as well as a stockroom from which products in the catalog

can be requested by potential users or “product handlers.” In order to make requests for products, product handlers must have completed some training related to the requested product. The system needs to manage training records for each hazardous product. The system must also track any health and safety (H&S) reports when incidents have occurred.

One of your colleagues identified several components and responsibilities and has proposed the following initial architecture. Is there a problem with this architecture?



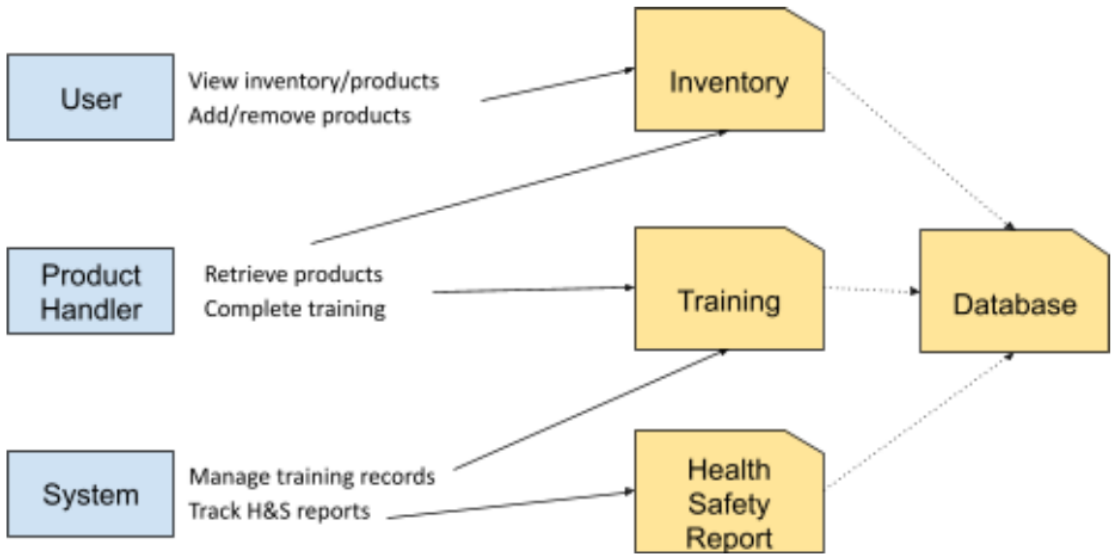
- If no, explain why the identified components and the architecture is appropriate.
- If yes, explain the problem and propose an alternative solution by sketching a new component design for this system and making any reasonable assumptions necessary.

It falls into the entity trap. Each entity identified in the requirements and has been made into a Manager component based on that entity. As a result, this isn't architecture. Instead, it is an object-relational mapping (ORM) of a framework to a database. There are many possible alternative solutions to address this problem. Several are listed below:

- If the system only needs simple database CRUD operations (create, read, update, delete), then the architect can download a framework to create user interfaces directly from the database. Many popular ORM frameworks exist to solve this common CRUD behavior.
- The architect can use the actor/actions approach to map requirements to components. In this solution, components can be designed around the typical users of the system and what kinds of things they might do with the system.
- The architect can use the workflow approach to identify the key roles,

determine the kinds of workflows these roles engage in, and build components around the identified activities.

Proposed actors/actions design:


User
View inventory/products
Add/remove products
Product Handler
Retrieve products
Complete training
System
Manage training records
Track H&S reports
Inventory
Training
Health Safety Report
Database