

# Model Driven Engineering

William MADIE  
Chadi GROLLEAU-RAOUX  
Yann JEANMAIRE dit CARTIER

# Table of contents

- |                                    |                               |
|------------------------------------|-------------------------------|
| 1. Project Overview                | 4. Platform Independent Model |
| 2. MDE Pipeline & Tools            | 5. Platform Specific Model    |
| 3. Computational Independent Model | 6. Demonstration              |

# Project Overview

## Implement the Order Process

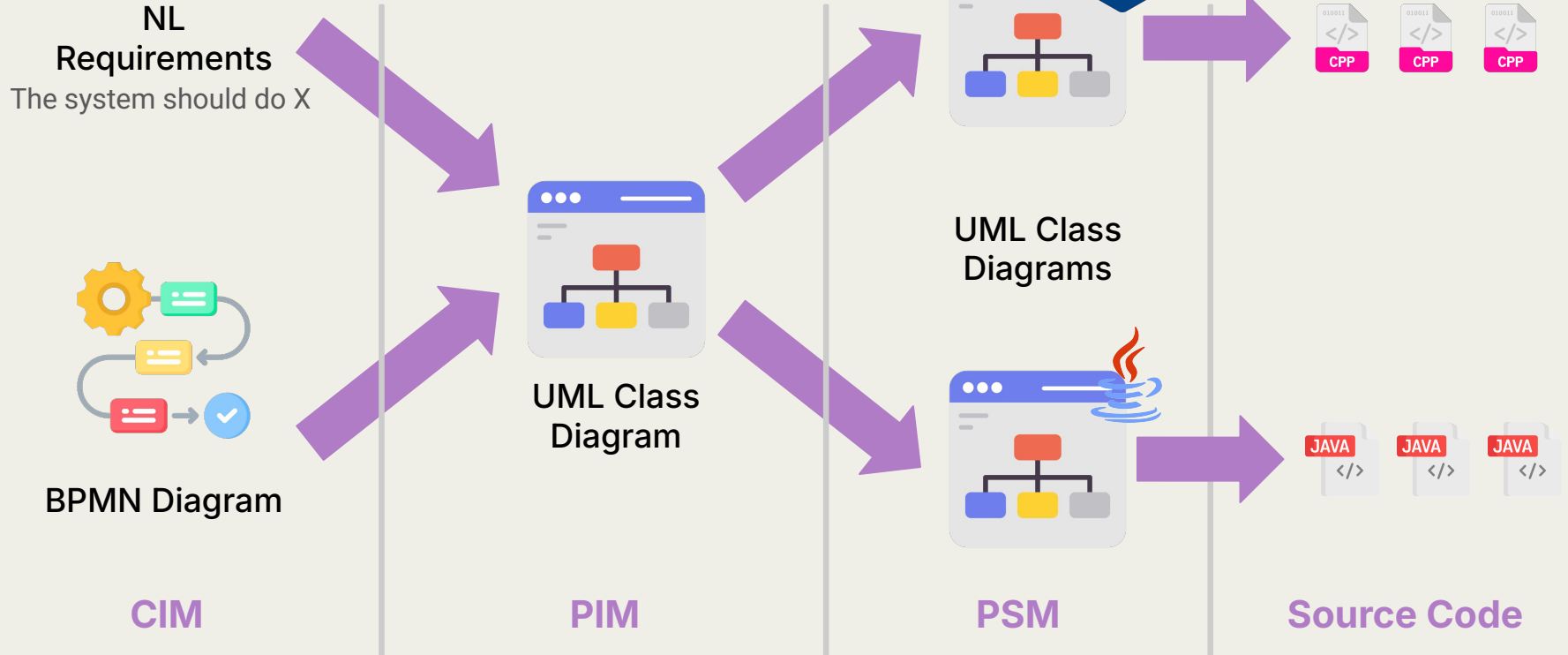
- **Client Management** (personal information)
- **Product Management** (stock, auto-reordering on low stock)
- **Payment Management** (credit card details and bank communication)
- **Order Receipt Management**

## Use of MDE

- Maintain alignment between codebase and requirements.
- Reduce complexity by focusing on high-level models.
- Enhance maintainability, consistency and evolvability.
- Provide a support for decision-making (future design and architecture choices)



# MDE Pipeline



# MDE Tools

CIM >> PIM



ChatGPT



Draw.io



Google  
Docs



Enterprise  
Architect

PIM >> PSM

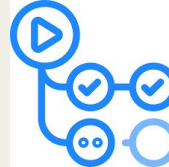


Enterprise  
Architect

PSM >> Source Code



GitHub



GitHub  
Actions



C++



Java



VSCode



Enterprise  
Architect

# Generative AI in our project

We leveraged generative AI as a supportive tool throughout the duration of our project in several key areas:



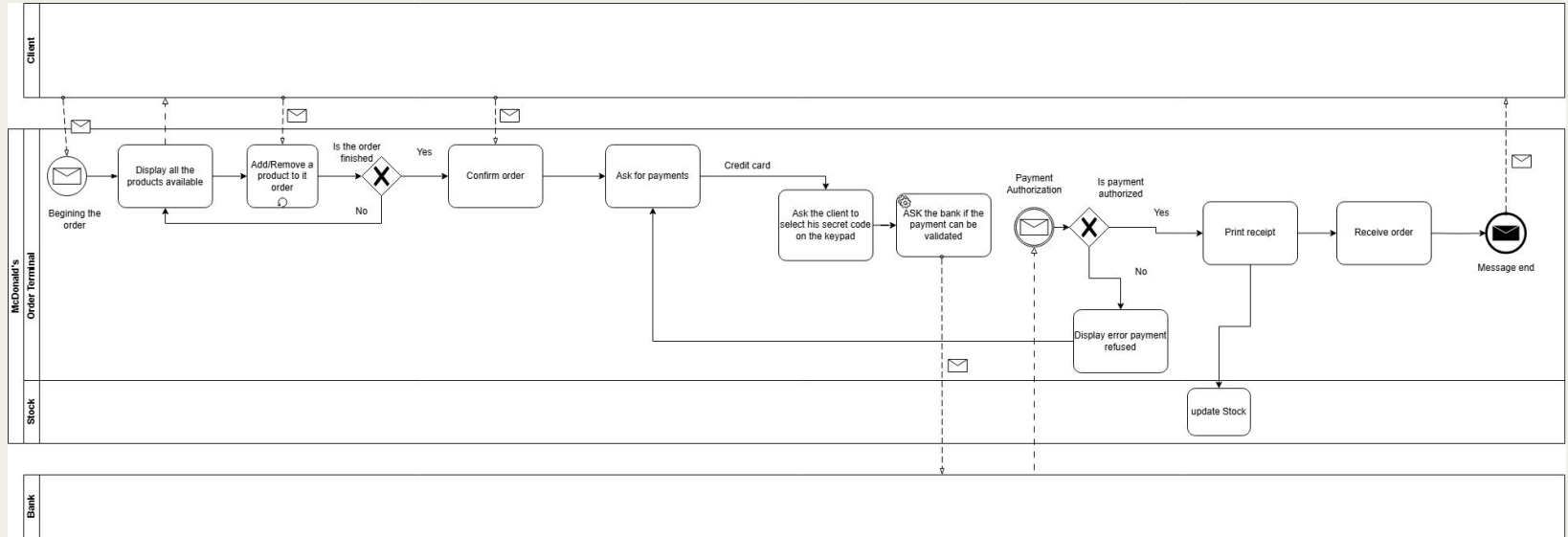
ChatGPT

- **Software Requirements Definition:** Generative AI assisted in drafting and refining our software requirements. It helped with sentence construction and served as a reviewer against various quality criteria, including clarity, ambiguity, redundancy, and overall understandability.
- **BPMN Diagram Development:** AI supported the creation of our BPMN process diagram by determining which elements were essential to represent and which could be omitted.
- **Code Implementation in Java and C++:** We utilized generative AI to assist with writing and troubleshooting code in both Java and C++, streamlining development across both languages.
- **Navigation of the C++ Tooling Ecosystem:** The AI proved especially valuable in helping us understand and effectively use C++ tools that were previously unfamiliar to us.
- **Project Structure and Tool Integration:** Generative AI also helped us integrating tools such as Enterprise Architect with Java and C++ command-line interfaces, as well as version control systems like Git and GitHub.

# Computational Independent Model - Natural Language Requirements

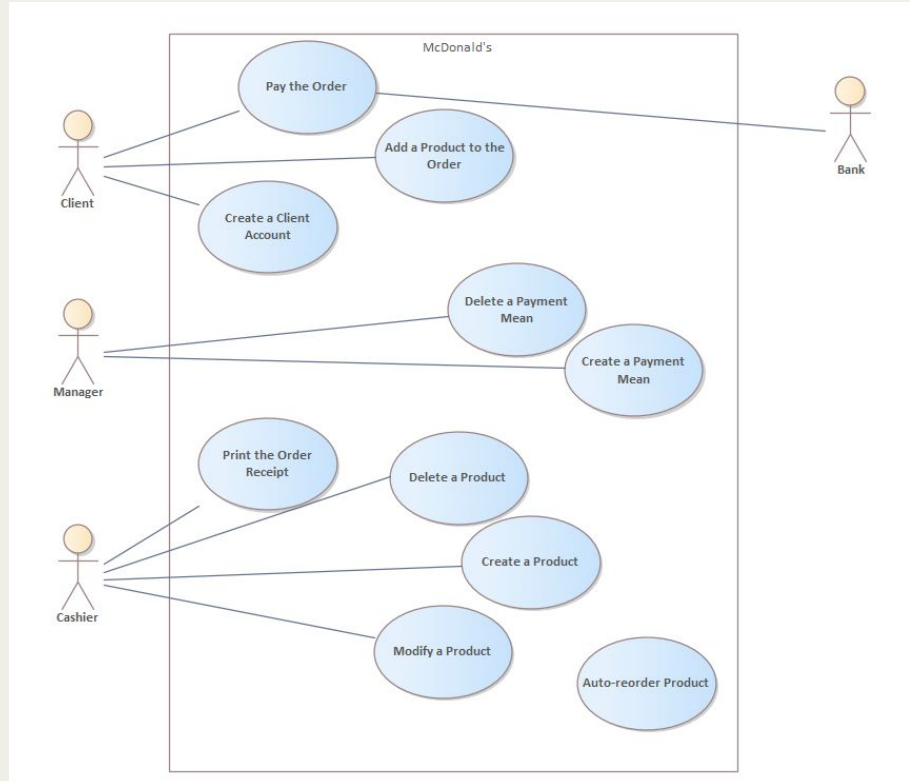
ID	Description
1	The system shall let a client order from an order terminal.
2	The order shall contain at least one product and at most 4 products to its order.
3	The order is identified by a unique number
4	The system shall let a client add a product in its order.
5	The system shall verify if a payment is authorized
6	The system shall produce the receipt of the order
7	The system shall allow the user to remove a product from his order
8	The system shall track the stock of each product
9	The system shall display if the product is out of stock
10	The system shall send valid orders to the kitchen.

# Computational Independent Model | BPMN

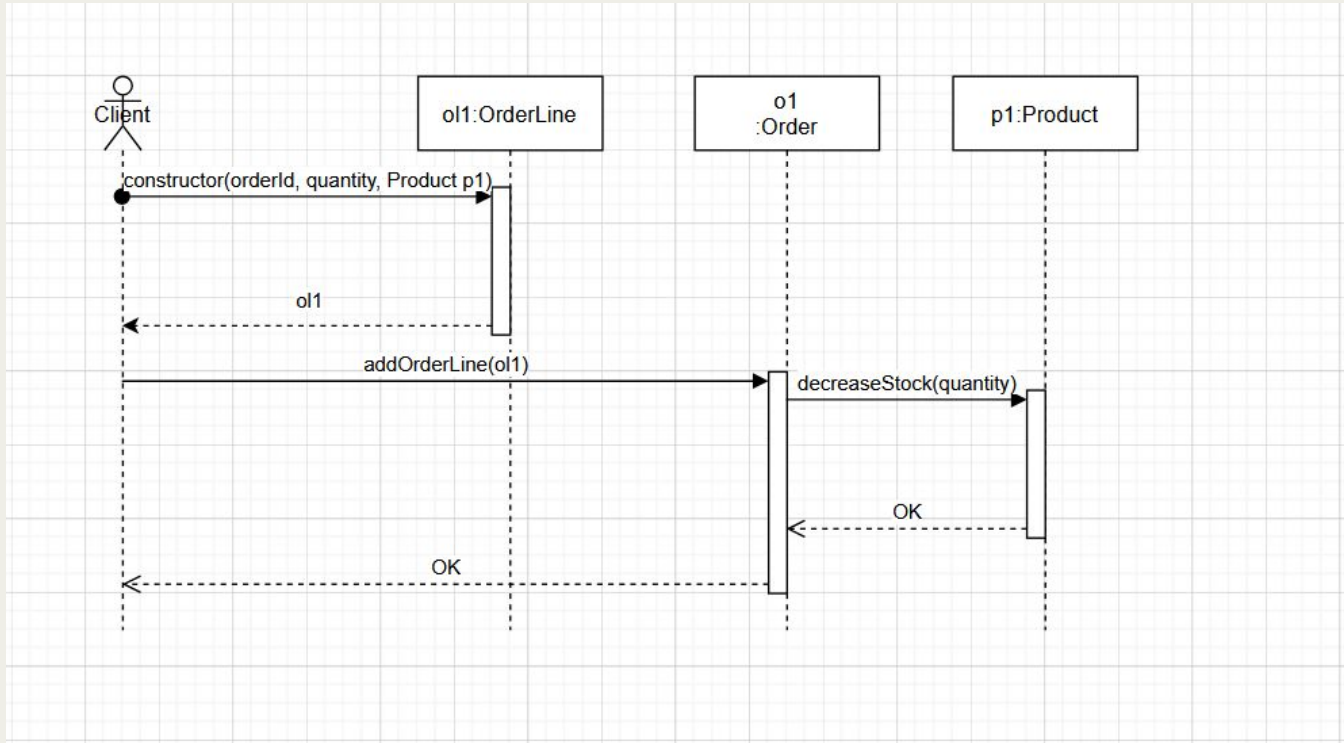




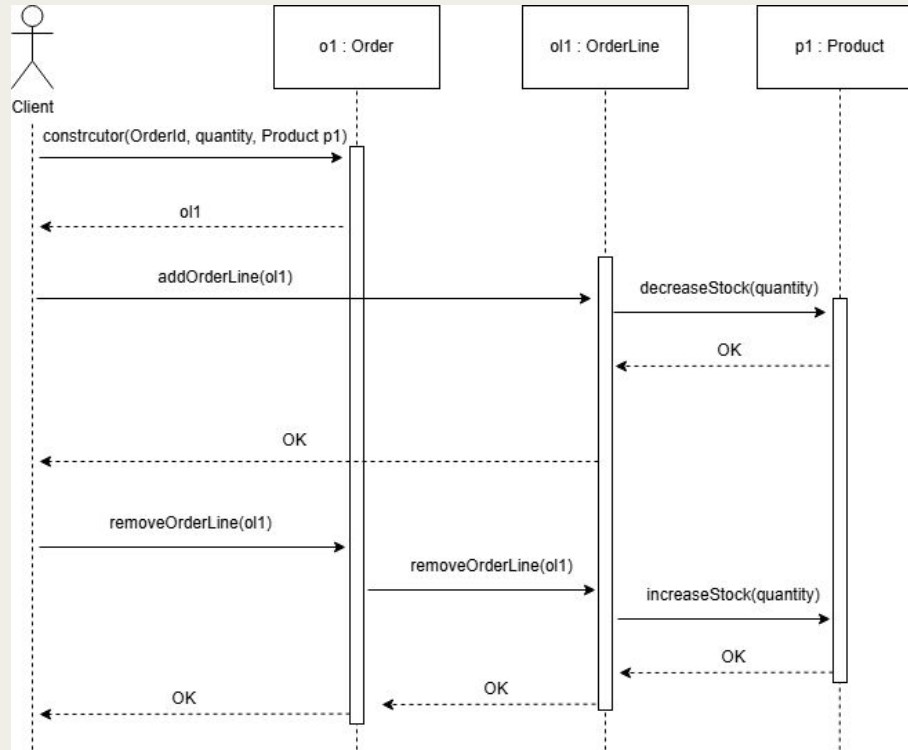
# Platform Independent Model | Use Case Diagram



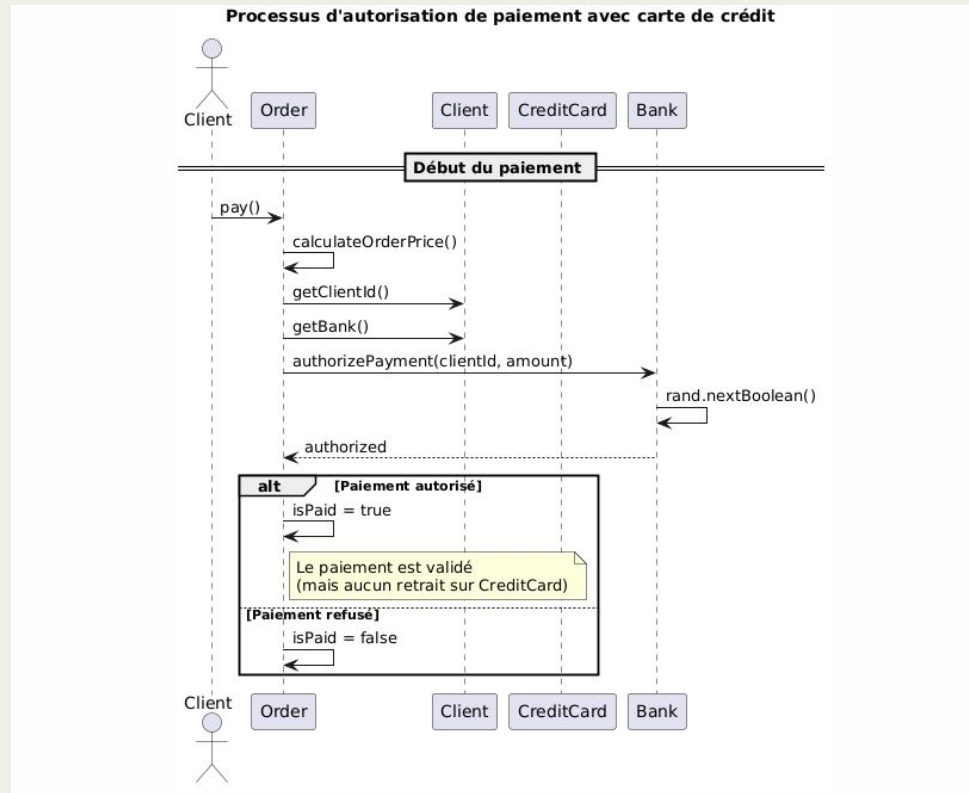
# PIM | Sequence Diagram | Add an Order Line



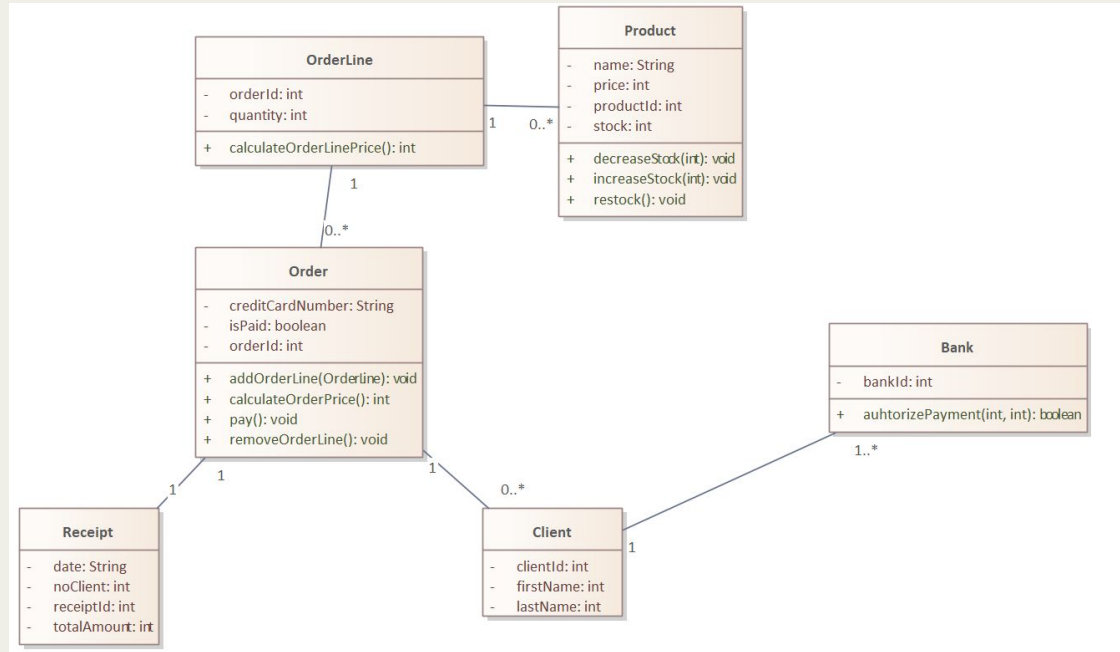
# PIM | Sequence Diagram | Remove an Order Line



# PIM | Sequence Diagram | Pay the Order

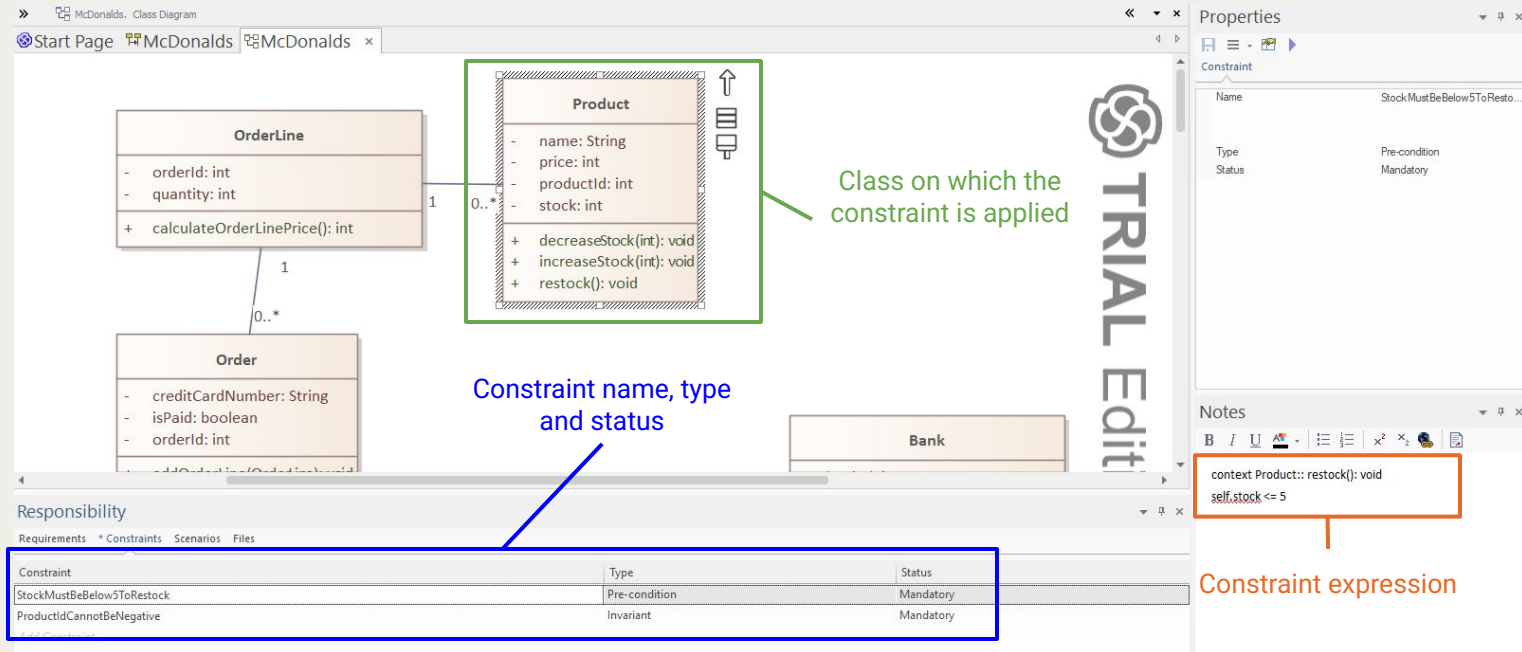


# Platform Independent Model | Class Diagram



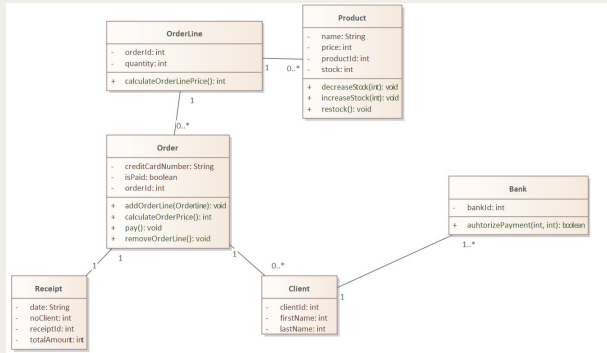
Class Diagram

# Platform Independent Model | OCL Constraints



Class Diagram

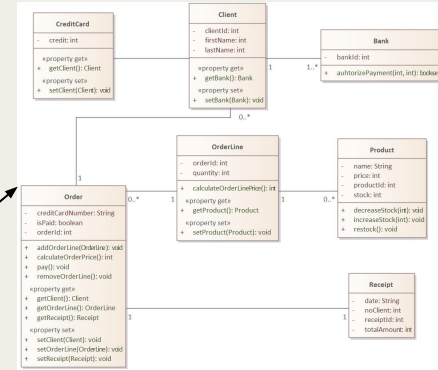
# Platform Specific Model



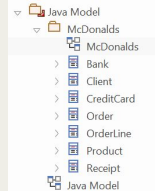
Platform Independent Model



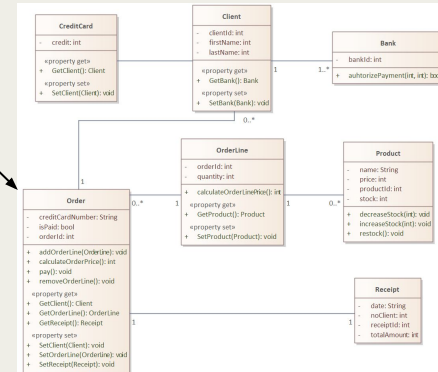
Java



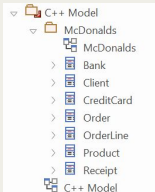
PSM  
+  
Java class



C++



PSM  
+  
.cpp and .h files



# Code & Project Structure

```
1 package McDonalds;
2
3 /**
4  * Test rigging
5  * Author: null
6  * Version 1.0
7  * Generated: 22-Jul-2023 18:13:04
8  */
9 public class Order {
10
11     private String creditCardNumber;
12     private boolean isPaid;
13     private int orderId;
14     private Client m_Client;
15     private OrderLine m_OrderLine;
16     private Receipt m_Receipt;
17
18     public Order(){}
19
20 }
21
22 public void finalize() throws Throwable {
23
24 }
25
26 public void addOrderLine(){
27
28 }
29
30 public int calculateOrderPrice(){
31     return 0;
32 }
33
34 public Client getClient(){
35     return m_Client;
36 }
37
38 public OrderLine getOrderLine(){
39     return m_OrderLine;
40 }
41
42 public Receipt getReceipt(){
43     return m_Receipt;
44 }
45
46 public void pay(){
47
48 }
49
50 public void removeOrderLine(){
51
52 }
53
54 /**
55  * Remove newval
56  */
57 public void setClient(Client newval){
58     m_Client = newval;
59 }
60
61 /**
62  *
63  */
64 /**
65  * Remove newval
66  */
67 public void setOrderLine(OrderLine newval){
68     m_OrderLine = newval;
69 }
70 }
```

Enterprise Architect  
generated file

```
1 package McDonalds;
2 import java.util.ArrayList;
3 import java.util.Iterator;
4 import java.util.List;
5
6 /**
7  * Order class
8  * Author: null
9  * Version 1.0
10  * Generated: 22-Jul-2023 18:13:04
11  */
12 public class Order {
13
14     private String creditCardNumber;
15     private boolean isPaid;
16     private int orderId;
17     private Client m_Client;
18     private Receipt m_Receipt;
19     private List<OrderLine> orderLines;
20
21     public Order() {
22         this.orderLines = new ArrayList<>();
23         this.isPaid = false;
24     }
25
26     public void addOrderLine(OrderLine orderLine) {
27         if (orderLines.size() < 4) {
28             orderLines.add(orderLine);
29         } else {
30             System.out.println("Cannot add more than 4 products to the order.");
31         }
32     }
33
34     public int calculateOrderPrice() {
35         int total = 0;
36         for (OrderLine line : orderLines) {
37             total += line.calculateOrderPrice();
38         }
39         return total;
40     }
41
42     public void pay() {
43         int amount = calculateOrderPrice();
44         boolean authorized = m_Client.authorizePayment(m_Client.getClientId(), amount);
45         this.isPaid = authorized;
46
47         if (authorized) {
48             System.out.println("Order Class : Payment authorized.");
49             // Automatically create receipt here
50         } else {
51             System.out.println("Order Class : Payment declined.");
52         }
53     }
54
55     public void removeOrderLine(Product product) {
56         Iterator<OrderLine> iterator = orderLines.iterator();
57         while (iterator.hasNext()) {
58             OrderLine line = iterator.next();
59             if (line.getProduct().equals(product)) {
60                 iterator.remove();
61                 System.out.println("Order Class : Product removed from order.");
62             }
63         }
64         return;
65     }
66
67     public void print() {
68         System.out.println("Order Class : Product not found in order.");
69     }
70 }
```

Implemented file



# 8

# Demonstration

# Our Experience

William	Yann	Chadi
Many open source tools are not updated anymore	Most of the open-source tools are outdated or deprecated, and the generated code often fails to compile due to broken or obsolete dependencies.	Almost all the tools are deprecated or not usable
Generated code contains deprecated code parts	While the code sometimes reflects the UML structure, it's rarely clean or production-ready, leading to significant manual corrections and increased project costs.	The file generated are almost empty and we have to download one by one every files
Different levels of support depending on target language		
Round-trip mechanism broke several times		

# Thanks!

Do you have any questions?

8

# Demonstration