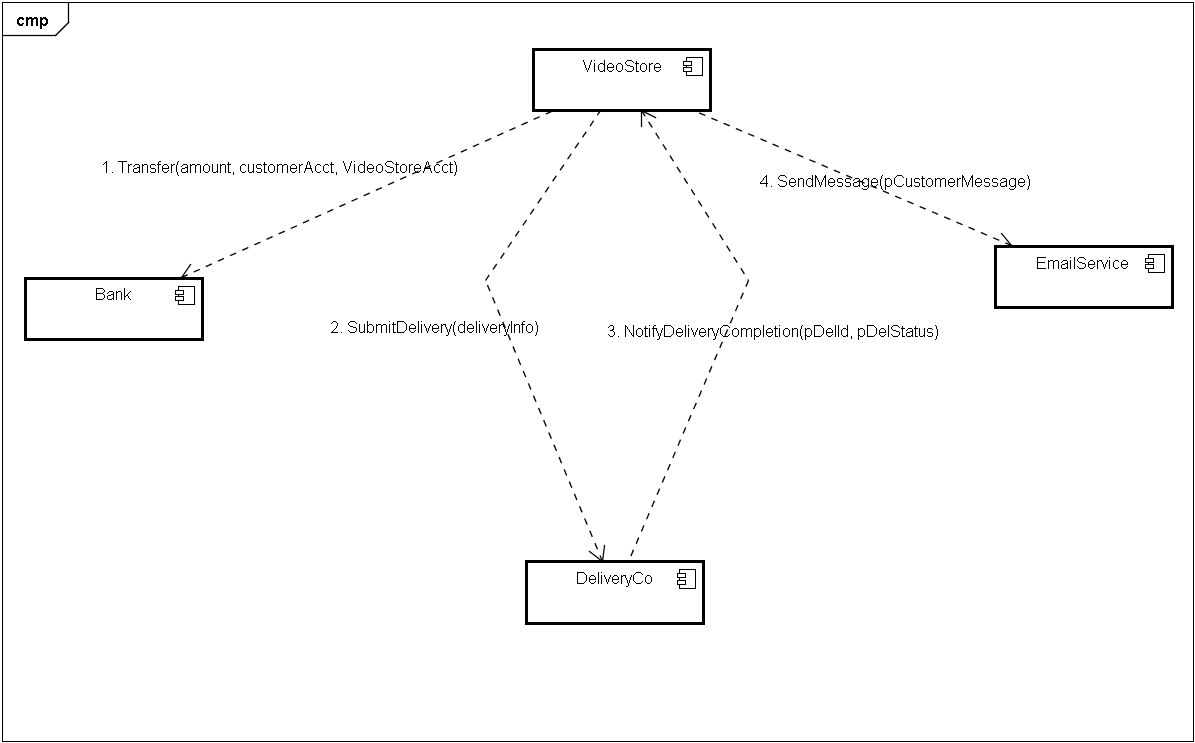
**Enterprise-Scale Software Architecture COMP5348**

Semester 1, 2011

**Group Project - System Integration, Availability and Fault Tolerance**

Enterprise applications usually need to integrate with a number of legacy or third party systems to fulfill business requirements. In this assignment, your task is to provide a suitable integration solution to integrate the VideoStore application with the 3 other applications shown in Figure 1. Integration with these 3 applications is required in order to support the following workflow for submitting orders:

* When an order is placed, the video store application should:  
  + Request a transfer of funds from the customer's account to VideoStore's account using the services provided by the Bank Application.
  + If the transfer is successful, the video store requests delivery for the purchased items from DeliverCo. DeliveryCo notifies the video store when delivery is complete.
    - The customer is notified via email (using the EmailService) when a delivery is submitted to DeliveryCo, AND when DeliveryCo claims that the delivery is complete.
    - The stock levels for an ordered media item should also be updated if an order is successfully submitted.
  + If the transfer fails, the customer is notified via email (using the EmailService) that there is a problem with their credit, and that the purchase order cannot proceed.



**Figure 1- Systems to be Integrated**

The four applications are currently integrated through synchronous point to point calls, and most of the workflow described above can be seen in VideoStore.Business.Components/OrderProvider::SubmitOrder().The solution is setup with a customer account username: “Customer” and password: “Customer”. We suggest you use this account for your testing. The current integration solution is functional, but isn’t ideal in addressing quality attributes like availability and reliability. In tutorials 6 and 7 you looked at how availability and fault tolerance can be improved by making use of asynchronous messaging and message oriented middleware. You will need to modify the integration solution provided and improve it by taking the following into account:

1. How are the availability and reliability quality attributes improved? What other quality attributes does my solution target to improve, and how?
2. Is my integration solution fault tolerant?
   * You will need to consider combinations of failure scenarios like:
     + What happens if the VideoStore is up and the Bank component is down?
     + What happens if the Bank transfer fails because there are insufficient funds => Are stock levels still correct after attempting a SubmitOrder with such a failure?
     + What happens if the Bank component is down but DeliveryCo is up?

You will need to consider how your solution handles failure conditions like those above and any other failure conditions you can identify. You can assume that the delivery company will always be successful in delivering orders, but that their system may go down. In your submission you will need to include a report which describes your integration solution and answers questions 1 and 2, giving specific scenarios that illustrate how your integration solution would improve fault tolerance. Your report should also explain how your integration solution addresses some of the quality attributes mentioned above.

**Marking Criteria**

1. The integration solution supports the business workflow outlined above (3 marks)
2. System design and implementation identifies and considers failure scenarios and can still operate when such failure scenarios occur (4 marks)
3. System design considers data consistency across systems when failure occurs (4 marks)
4. The submitted report shows a good understanding of how fault tolerance and quality attributes are affected by the design decisions made in creating the submitted integration solution (4 marks)

**Deliverables** (due in Week 11 on Friday at 9pm)

1. Source code for your application and a readme file explaining how to run your application.

2. Unit tests and results for all of your components that show you have properly tested your code.

4. A brief (2 pages) description of how your code works focusing on message exchanges between your components and how you will handle the unavailability of various systems. Describe what quality of service attributes you would use if you were building a real system and why.

5. A copy of the Academic Honesty cover sheet (group work version) signed by all members of the group

Submit deliverables 1 to 4 through WebCT, and hand in deliverable 5 in person at the lab.