Lab 4: Fundamental and Monolithic Architectural Styles

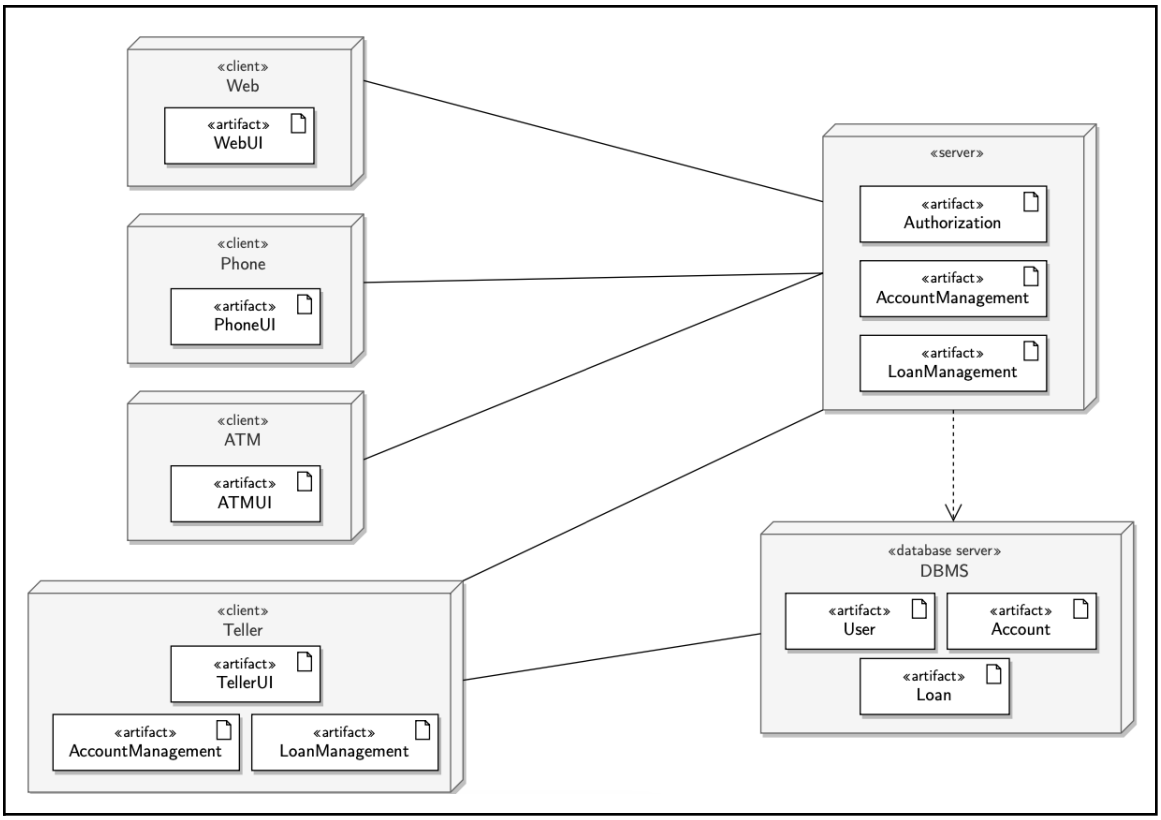
1. A typical bank manages customers who have accounts, loans, and stocks: accounts are used to pay for loans and stocks. Customers can perform transactions on those accounts from either an automated teller machine (ATM), through a phone interface, or through a web interface. Customers can also perform transactions on their loans and stocks, but only through the web interface. Additionally, personnel (tellers) at the branch can look at customers' accounts, loans, and stocks, and perform transactions. The following components have been identified:
    - User: to record and handle general information about customers and tellers
    - Account: to record information on accounts customers may have with the bank
    - AccountManagement: to perform operations on accounts such as: searching for an account, checking the balance, performing payments from an account (e.g., bill, loan), performing a transfer of funds between two accounts, etc.
    - Loans: to record information on loan customers may have with the bank, such as: loan number, customer holding the loan, balance, rate, etc.
    - LoanManagement: to perform operations on loans such as processing monthly payments, creating and removing loans, renewing loans, etc.
    - Authorization: to identify whether the user (customer, teller) is authorized to perform a transaction, accounting for the medium of communication being used
    - TellerUI: the user interface used by a teller at a branch
    - PhoneUI: the user interface used by a customer who calls using their phone
    - WebUI: the user interface used by a customer on the internet
    - ATMUI: the user interface used by a customer at an ATM
    -

    The layered architecture style has been used to create an initial architecture design (presented as a UML component diagram) as shown below.

    Suppose that the following availability requirements have been defined:
    - In case of a network connectivity problem or server problem, the customer shall not be able to successfully complete any transaction using their phone.
    - In case of a network connectivity problem or server problem, the customer shall not be able to successfully complete any transaction from an ATM.

- In case of a network connectivity problem or server problem, the customer shall not be able to perform any transaction using the web interface.
- In case of a network connectivity problem or server problem, the teller shall still be able to perform transactions, though a reduced set of transactions, locally. When the connection is re-established, a procedure to complete transactions so the server side is updated shall be used, which may end in either successful completion or failure to complete the transaction (e.g., there is no longer enough funds in an account).

Which process distribution pattern(s) would you use for the client-server architecture of the bank software? Draw a deployment diagram of the above components and justify your response.

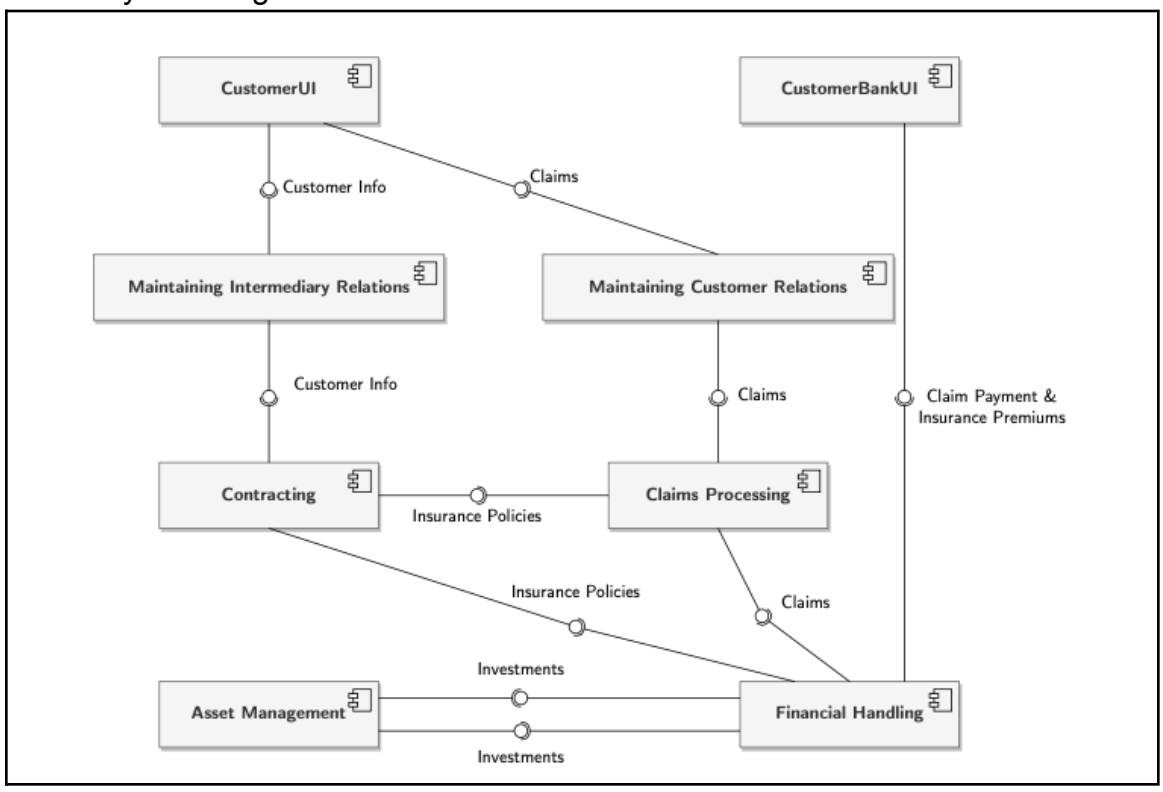


2. All Angles Insurance originally provided home and travel insurance but merged recently with two other insurance companies providing car insurance and legal aid insurance. By streamlining their operations and removing duplication, substantial synergy is expected from this merger. Now, All Angles Insurance is wrestling with the intricacies of integrating these companies and has decided to take an enterprise architecture approach to create more insight into this

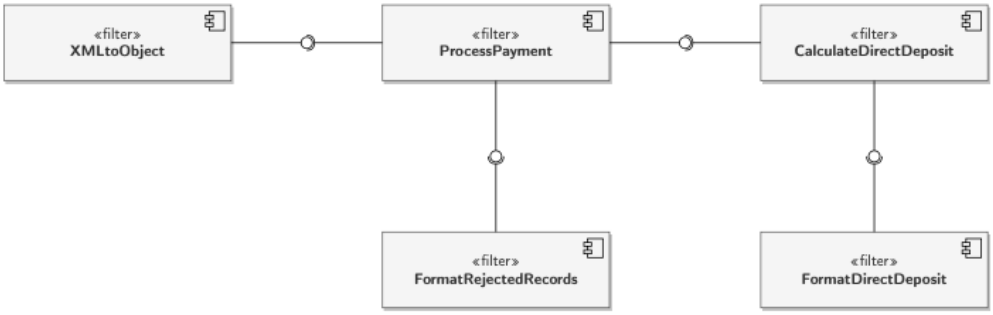
complexity. The primary operations of the All Angles Insurance system are described in terms of the following components:

- Maintaining Customer Relations and Intermediary Relations: These business components are responsible for the contracts of All Angles Insurance with its customers and the intermediaries that sell its products. These components handle customer questions and incoming claims and perform marketing and sales.
- Contracting: This component does the ‘back-office’ processing of contracts. It performs risk analysis and ensures legally and financially correct contracts.
- Claims Processing: This component is responsible for processing insurance claims, judging their validity and valuation, and deciding the further course of action.
- Financial Handling: This component performs the regular premium collection, according to the insurance policies with customers as produced by Contracting and handles the payment of insurance claims.
- Asset Management: This component manages the financial assets of the All Angles Insurance system, e.g., by investing in stocks and bonds.

Using layered architecture style, create a UML component diagram for the All Angles Insurance system. Add two user interfaces: Customer and Customer’s Bank to your design.



3. Consider the initial design of a software system to process payments for direct deposit into a customer's bank account shown below. The initial design uses the pipeline architecture style.



After further analysis of the initial design, it has been determined that error handling and fault tolerance are essential to achieve the availability architecture characteristics. One of the liabilities of using the pipeline architecture style is error handling and fault tolerance. Fortunately, in software architecture design, there are common techniques, known as tactics, an architect can use to achieve the required architecture characteristics. A tactic is a design decision that influences the achievement of an architecture characteristic (quality attribute).

Two such tactics are as follows:

- Redundant Spare: This tactic refers to a configuration in which one or more duplicate components can step in and take over the work if the primary component fails. This tactic is at the heart of the hot spare, warm spare, and cold spare patterns, which differ primarily in how up-to-date the backup component is at the time of its takeover.
    - Active redundancy (hot spare): For stateful components, this refers to a configuration in which all of the nodes (active or redundant spare) in a protection group receive and process identical inputs in parallel, allowing the redundant spare(s) to maintain a synchronous state with the active node(s). A protection group is a group of processing nodes in which one or more nodes are "active," with the remaining nodes serving as redundant spares. Because the redundant spare possesses an identical state to the active processor, it can take over from a failed component in a matter of milliseconds. The simple case of one active node and one redundant spare node is commonly referred to as one-plus-one redundancy. Active redundancy can also be used for facilities protection, where active and standby network links are used to ensure highly available network connectivity.

    - Passive redundancy (warm spare): For stateful components, this refers to a configuration in which only the active members of the protection group process input traffic. One of their duties is to provide the redundant spare(s) with periodic state updates. Because the state maintained by the redundant spares is only loosely coupled with that of the active node(s) in the protection group (with the looseness of the coupling being a function of the period of the state updates), the redundant nodes are referred to as warm spares. Passive redundancy provides a solution that achieves a balance between the more highly available but more compute-intensive (and expensive) active redundancy pattern and the less available but significantly less complex cold spare pattern (which is also significantly cheaper).”
    - Spare (cold spare): Cold sparing refers to a configuration in which redundant spares remain out of service until a failover occurs, at which point a power-on-reset procedure is initiated on the redundant spare prior to its being placed in service. A power-on-reset ensures that a device starts operating in a known state. Due to its poor recovery performance, and hence its high mean time to repair, this pattern is poorly suited to systems having high-availability requirements.
- Voting & Replication: Voting involves comparing computational results from multiple sources that should be producing the same results and, if they are not, deciding which results to use. This tactic depends critically on the voting logic, which is usually realized as a simple, rigorously reviewed, and tested singleton so that the probability of error is low. Voting also depends critically on having multiple sources to evaluate. Replication is the simplest form of voting; here, the components are exact clones of each other. Having multiple copies of identical components can be effective in protecting against random failures of hardware but cannot protect against design or implementation errors, in hardware or software, since there is no form of diversity embedded in this tactic.
    - Triple Modular Redundancy (TMR) is a widely used implementation of the voting tactic. It employs three components that do the same thing. Each component receives identical inputs and forwards its output to a voting component that contains the voting logic which detects any inconsistency among the three output states. Faced with an inconsistency, the voter reports a fault. It must also decide which output to use, and different instantiations of this pattern use

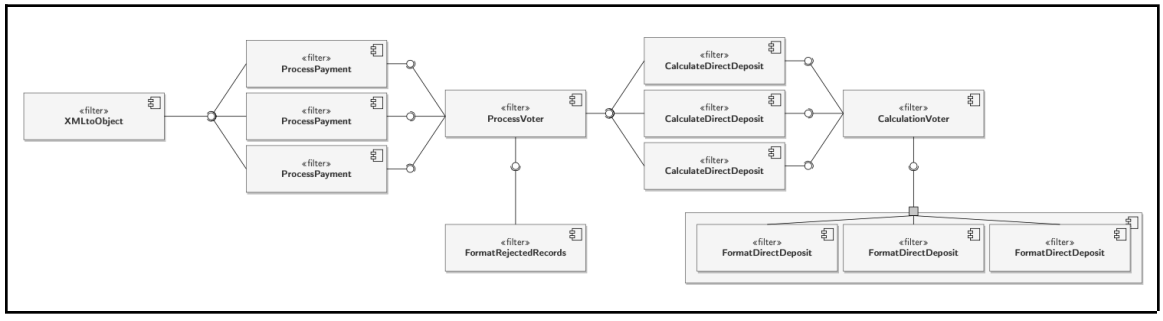
different decision rules. Typical choices are letting the majority rule or choosing some computed average of the disparate outputs.

You and your team have decided to apply these tactics to improve the error handling and fault tolerance in the initial software design. The decisions are listed below.

- The FormatDirectDeposit component is considered critical for the whole application, and it is decided to replicate it. A protection group of three replicas of this component, using active redundancy, is created. Assume that the protection group has a Coordinator mechanism to dispatch requests to all replicas at once, to switch from the primary to a hot spare automatically when the primary is deemed failing, to receive the same request several times and to execute it only once.
- For similar (criticality) reasons, it is decided to use a Triple Modular Redundancy (TMR) tactic for the CalculateDirectDeposit component. Note that the CalculateDirectDeposit component produces data for the FormatDirectDeposit component to consume as this will help correctly identify where the voter of the TMR should be.
- For similar (criticality) reasons, it is decided to use a Triple Modular Redundancy (TMR) tactic for the ProcessPayment component. Again, note that the ProcessPayment component produces data for the CalculateDirectDeposit component to consume as this will help correctly identify where the voter of the TMR should be.

For the use of TMR for the CalculateDirectDeposit and the ProcessPayment component, assume that the voting mechanism is a majority vote, using equality: if a majority of received values equal, then the voter produces this value; otherwise, the voter produces an undefined value.

Produce a new UML component diagram to represent the updated architecture considering the application of the tactics listed above.



4. In Java/Swing, class JList is used to display the contents of a list. A JList is (only) responsible for displaying the contents of a list. The actual values in the list are

supposed to be retrieved from a class that implements interface ListModel. JList is said to be a listener of the ListModel (i.e., it registers to ListModel), and is informed when the contents or the length of the list changes.

Which architectural style is used here? Justify your answer and produce a UML object diagram representing the system using the given object names.

Model-View-Controller (MVC) architecture style. The model is an instance of the ListModel class and the view an instance of the JList class. The view is notified when changes are made to the model. A suitable JController object is needed to handle the view coordination.