**[Homework 17](https://github.com/hendraanggrian/IIT-ITM511/blob/assets/assignments/hw17.docx): Distributed SWE**

**Problem 1**

*What do you understand by “scalability”? Discuss the differences between scaling up and scaling out and explain when these different approaches to scalability may be used.*

Software requirements may evolve as users interact with the system, or naturally as the system progresses. Scalability is how well a system can cope with growing user requirements. To ensure that a system can deliver high-quality services, scalability can be achieved through scaling upward or outward:

1. **Scaling up:** Enhance the system's capabilities by replacing an existing component. To illustrate this point, a server with an aging CPU socket is upgraded with a newer motherboard and processor.
2. **Scaling out:** Extending the capabilities by throwing out more resources to the problem. Using the server analogy as an example, we could deploy extra servers in a cluster to balance the workload.

**Problem 2**

*Explain why distributed software systems are more complex than centralized software systems, where all of the system functionality is implemented on a single computer.*

Distributed software systems are collections of independent computers in a network that function as a unified whole. The collaboration serves a scalability purpose but comes at the cost of complexity in managing roles and communication among participants. Since an individual system can break, a distributed system must also consider an error-handling strategy to maximize the system's availability.

**Problem 3**

*Using an example of a remote procedure call, explain how middleware coordinates the interaction of computers in a distributed system.*

Remote Procedural Call (RPC) is a parameterized client-server connection by passing and triggering code executions (What's the difference between RPC and REST?, n.d.). In an RPC environment, the middleware intercepts the client's request to perform sub-tasks before directing it to the server. The server then converts the data to a client-readable format.

For example, a customer is shopping in an online marketplace that utilizes RPC. The customer does not need to know the server's specific details to browse or buy a product. As the customer triggers a purchase call, the middleware translates the network packet for local services to find availability, verify the transaction and route a shipment.

**Problem 4**

*What are the different logical layers in an application with a distributed client–server architecture?*

1. **Presentation layer:** Manage how information is presented and the client's relationship with each other (Sommerville, 2016, p. 500).
2. **Data-handling layer:** Receives the input data, processes it, and returns it to the client.
3. **Application processing layer:** Execute business logic that provides a service to users.
4. **Database layer:** Layer that holds information for retrieval and manipulation.

**Problem 5**

*You have been asked to design a secure system that requires strong authentication and authorization. The system must be designed so that communications between parts of the system cannot be intercepted and read by an attacker. Suggest the most appropriate client-server architecture for this system and, giving the reasons for your answer, propose how functionality should be distributed between the client and the server systems.*

Given the high-security requirement of the system, a three-tier architecture is ideal to protect against unauthorized access to sensitive information. A three-tier architecture refers to the communication layers between the client, as the initial party, and the server, as the final party. The communication is intermediated by a middleware which acts as an interpreter and enforces additional rules.

In the context of the software system with robust authentication demand, the database is the third party storing account credentials. Meanwhile, a web server functions as the middleware that controls login sessions, cryptographic standards and other security checks. This layered approach verifies user access before the application reaches business logic.

**Problem 6**

*Your customer wants to develop a system for stock information where dealers can access information about companies and evaluate various investment scenarios using a simulation system. Each dealer uses this simulation in a different way, according to his or her experience and the type of stocks in question. Suggest a client–server architecture for this system that shows where functionality is located. Justify the client–server system model that you have chosen.*

A two-tier client-server architecture is suitable for the aforementioned stock simulation application. Unlike a multi-tiered middleware that handles most work, two-tier architecture shares some responsibilities with client devices. This is particularly true in the fat-client model where the client is also responsible for processing as well as presenting data.

While a multi-tier architecture can apply to the software, I believe the personalization in processing and visualizing data makes two-tier a better fit in this case. Two-tier requires fewer layers in the communication protocol, thus simplifying the initiation. Unfortunately, multi-tier architecture is still a popular option for enhanced security and a unified experience.

**Problem 7**

*Using a distributed component approach, propose an architecture for a national theater booking system. Users can check seat availability and book seats at a group of theaters. The system should support ticket returns so that people may return their tickets for last-minute resale to other customers.*

The proposed distributed component architecture is appropriate for a system that enables users to book theater seats online. A distributed component architecture is described as a group of separate services linked by a communication middleware. The middleware also serves as a data provider to client requests.

A distributed component system allows a booking vendor to integrate with local theaters. To allow ticket returns, the system would also need to integrate with additional services like payment verification and resale to other vendors. Finally, users get alerts on incoming shows and browse available seats using a client application.

**Problem 8**

*What is the fundamental problem with a two-tier client–server approach? Define how a multitier client–server approach overcomes this.*

In a two-tier architecture, the client is in charge of receiving and computing information from the server. This exchange exposes sensitive information to clients, increasing the risk of a data breach when the client is compromised. Fundamentally, the computing capability is limited to the processing power of the client device. With a multi-tier architecture, a centralized server is deployed to treat information internally, leaving clients with processed messages and a unified user experience.

**Problem 9**

*List the benefits that a distributed component model has when used for implementing distributed systems.*

1. **Deferred implementation:** Non-critical services can be delayed until the middleware recognizes them (Sommerville, 2016, p. 507).
2. **Highly extensible:** Modularity in designed system to accommodate future change.
3. **Flexible and scalable:** The services are reliable, that is, one disturbance should not affect others.

**Problem 10**

*Your company wishes to move from using desktop applications to accessing the same functionality remotely as services. Identify three risks that might arise and suggest how these risks may be reduced.*

1. **Service availability:** Desktop applications run on the client device with no dependency. Whereas in remote protocol, the client is reliant on server activity and a stable internet connection that bridges them.

* **Risk mitigation:** Subscribe to online services with a respectable uptime rating, and handle connection errors in the client interface.

1. **Server overload:** Applications natively written in a desktop platform can make use of local system resources. Delegating computing-heavy tasks to the server inflicts sustained stress and potentially fails the system.

* **Risk mitigation:** Devise a response queuing mechanism in the server, and apply distributed component architecture to balance workload between servers.

1. **Distinctive experience:** The company uses Software Development Kit (SDK) to create a platform-specific desktop application that interacts with the operating system, creating a unique user experience. Some functionality may not be easily replicable in the client interface using the remote protocol.

* **Risk mitigation:** Develop and integrate RPC protocol into the existing desktop application, or migrate to a modern web framework that facilitates rich styling.

# References

Sommerville, I. (2016). Software Engineering. In *Software engineering diversity* (10 ed., pp. 500, 507). Pearson Education.

*What's the difference between RPC and REST?* (n.d.). Retrieved from Amazon Web Services: https://aws.amazon.com/compare/the-difference-between-rpc-and-rest/