S90.09$^{Q\&As}$

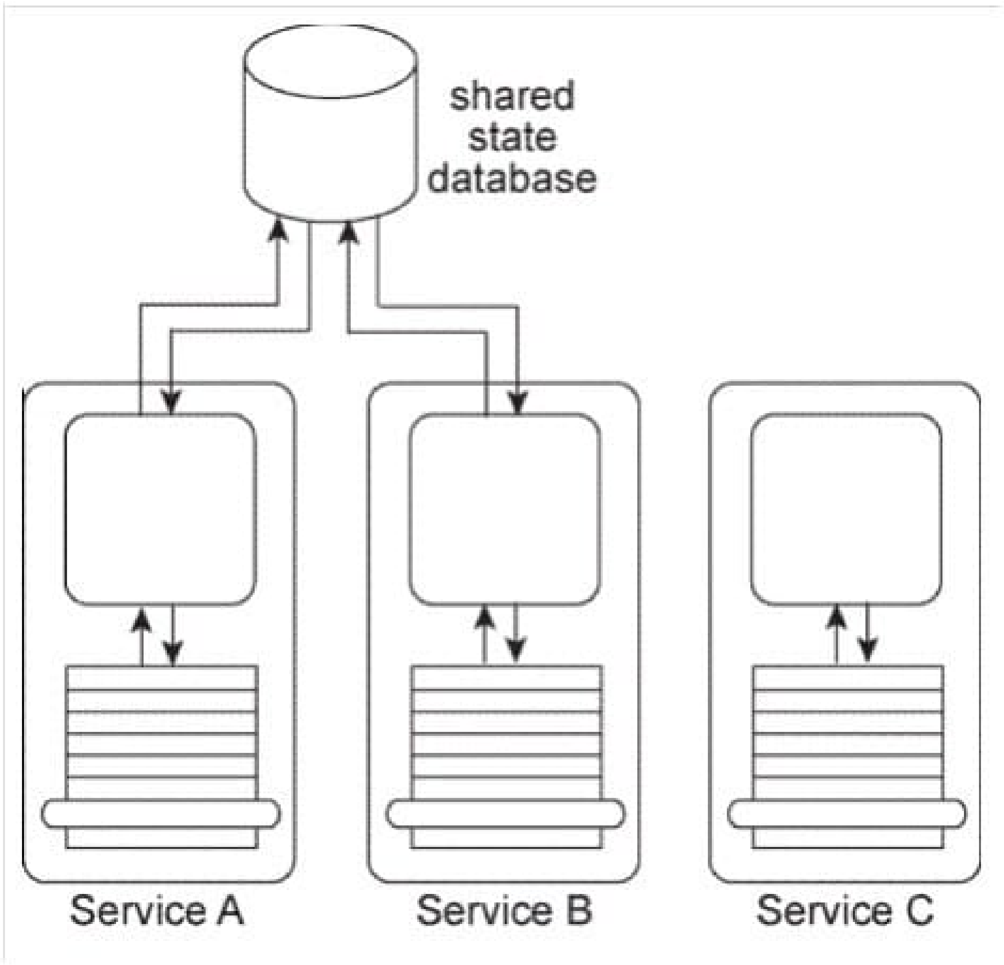
SOA Design & Architecture Lab

QUESTION 1

Services A, B, and C are non-agnostic task services. Service A and Service B use the same shared state database to defer their state data at runtime.

An assessment of these three services reveals that each contains some agnostic logic, but because it is bundled together with the non-agnostic logic, the agnostic logic cannot be made available for reuse.

The assessment also determines that because Service A and Service B and the shared state database are each located in physically separate environments, the remote communication required for Service A and Service B to interact with the shared state database is causing an unreasonable decrease in runtime performance.



How can the application of the Orchestration pattern improve this architecture?

A. The application of the Orchestration pattern will result in an environment whereby the State Repository and Service Data Replication patterns are naturally applied, allowing the shared state database to be replicated for Services A and B so that each task service can have its own dedicated state database. The Process Centralization pattern can also be applied to Services A and B, so that their logic is physically centralized, turning them into orchestrated task services.

B. The application of the Orchestration pattern will result in an environment whereby the Process Abstraction and Process Centralization patterns are naturally applied to Services A, B, and C, resulting in a clean separation of non-agnostic task services from newly designed agnostic services with reuse potential. Also, the State Repository pattern can be applied by the availability of a central state database that can be shared by Services A and

C. This database can be made available as a local part of the environment so that Services A and B can avoid remote communication.

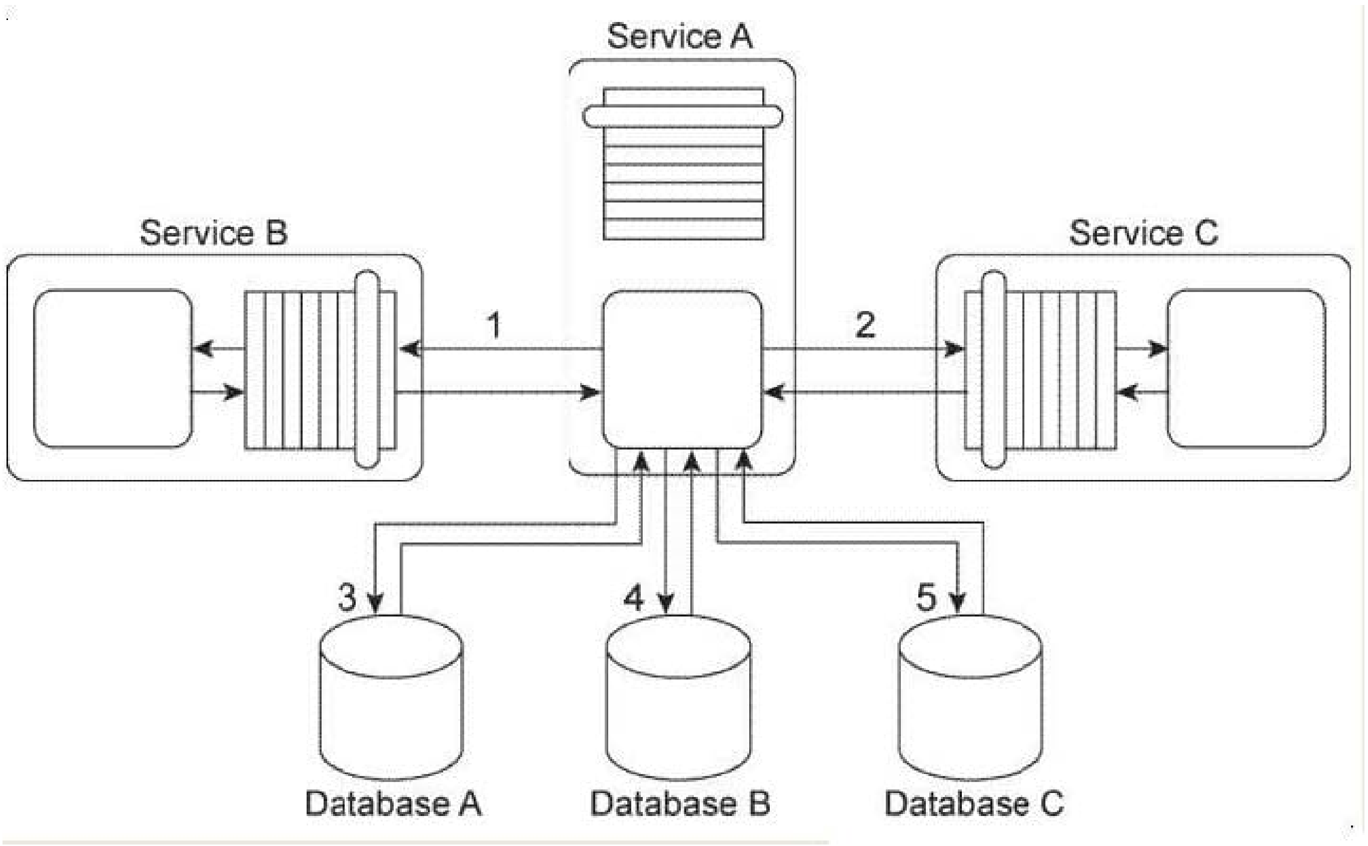
D. The application of the Orchestration pattern will result in an environment whereby the Compensating Service Transaction is naturally applied, resulting in the opportunity to create sophisticated exception logic that can be used to compensate for the performance problems caused by Services A and B having to remotely access the state database. The Process Abstraction and Service Broker patterns are also naturally applied, enabling the separation of non-agnostic logic and agnostic logic while providing common transformation functions required to overcome any disparity in the service contracts that will need to be created for the new agnostic services.

E. None of the above.

Correct Answer: B

QUESTION 2

You are told that in this service composition architecture, all four services are exchanging invoice-related data in an XML format. The services in Service Inventory A are standardized to use a specific XML schema for invoice data. Design standards were not applied to the service contracts used in Service Inventory B, which means that each service uses a different XML schema for the same kind of data. Database A and Database B can only accept data in the Comma Separated Value (CSV) format and therefore cannot accept XML formatted data. What steps can be taken to enable the planned data exchange between these four services?



A. The Data Model Transformation pattern can be applied so that data model transformation logic is positioned between Service A and Service B, between Service A and Service C, and between Service C and Service D . The Data Format Transformation pattern can be applied so that data format transformation logic is positioned between the Service B logic and Database A and between the Service D logic and Database B.

B. The Data Model Transformation pattern can be applied so that data model transformation logic is positioned between Service A and Service C and between Service C and Service D . The Data Format Transformation pattern can be applied so that data format transformation logic is positioned between

the Service B logic and Database A and between the Service D logic and Database B.

C. The Data Model Transformation pattern can be applied so that data model transformation logic is positioned between Service A and Service C . The Protocol Bridging pattern can be applied so that protocol bridging logic is positioned between Service A and Service B and between the Service C and Service D . The Data Format Transformation pattern can be applied so that data format transformation logic is positioned between the Service B logic and Database A and between the Service D logic and Database B.

D. None of the above.

Correct Answer: A

QUESTION 3

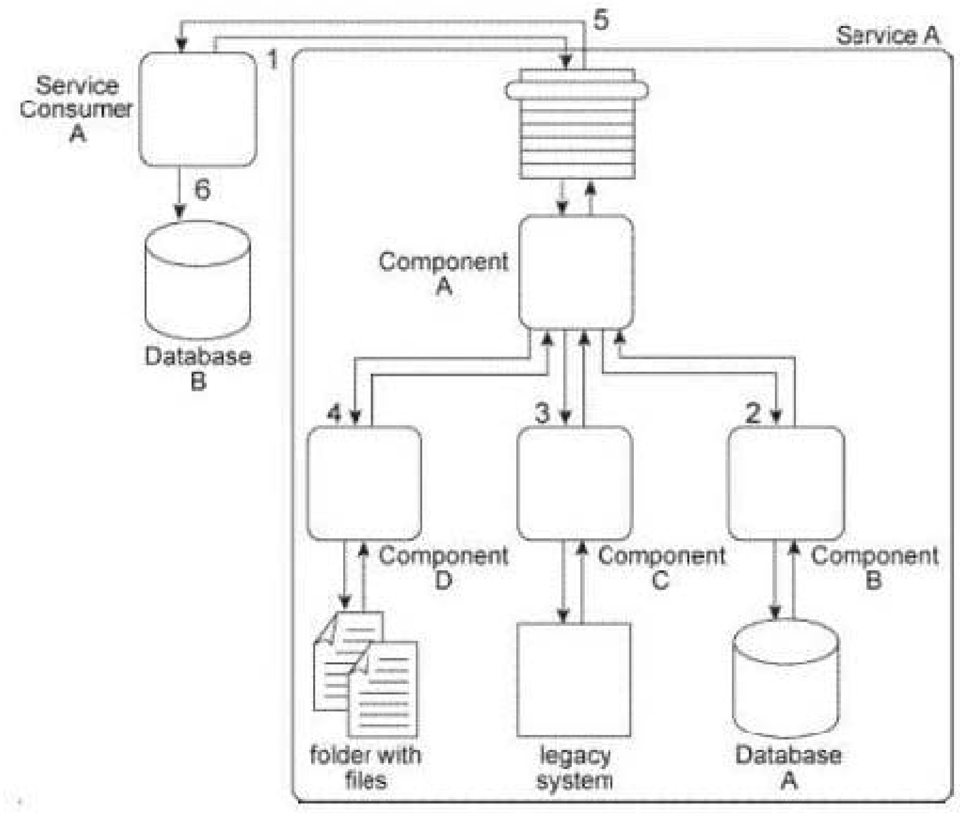
When Service A receives a message from Service Consumer A(1),the message is processed by Component A. This component first invokes Component B (2), which uses values from the message to query Database A in order to retrieve additional data. Component B then returns the additional data to Component A.

Component A then invokes Component C (3), which interacts with the API of a legacy system to retrieve a new data value. Component C then returns the data value back to Component A.

Next, Component A sends some of the data it has accumulated to Component D (4), which writes the data to a text file that is placed in a specific folder. Component D then waits until this file is imported into a different system via a regularly scheduled batch import. Upon completion of the import, Component D returns a success or failure code back to Component A.

Component A finally sends a response to Service Consumer A (5) containing all of the data collected so far and Service Consumer A writes all of the data to Database B (6).

Components A, B, C. and D belong to the Service A service architecture. Database A, the legacy system, and the file folders are shared resources within the IT enterprise.



Service A is a task service that completes an entire business task on its own without having to compose other services. However, you have received many complaints about the reliability of Service A . Specifically, it has three problems. First, when Component B accesses Database A, it may not receive a response for several minutes when the database is being accessed by other applications in the IT enterprise. Secondly, the legacy system accessed by Component C frequently crashes and therefore becomes unavailable for extended periods of time. Third, for Component D to respond to Component A, it must first wait for the batch import of the files to occur. This can take several minutes during which Service Consumer A remains stateful and consumes excessive memory. What steps can be taken to address these three problems?

A. The Legacy Wrapper pattern can be applied so that Component B is separated to wrap the shared database, thereby allowing Component A to interact with this new service instead of directly interacting with the database. The Legacy Wrapper pattern can be applied again so that Component C is separated into a separate service that acts as a wrapper of the legacy system API. Component D can then be separated into a separate service and the Event-Driven Messaging pattern can be applied to establish a publisher- subscriber relationship between this new service and Component A and between Service A and Service Consumer A. The interaction between Service Consumer A and Component A is then redesigned so that Component A issues a message back to Service Consumer A when the event related to the batch import is triggered.

B. The Service Data Replication pattern can be applied so that Component B can access a replicated database instead of having to access the shared Database A directly. The Legacy Wrapper pattern can be applied so that Component C is separated into a separate service that acts as a wrapper of the legacy system API. Next, the Reliable Messaging pattern can be applied so that acknowledgements are issued from the new wrapper service to Component A, thereby enabling notifying Component A during times when the legacy system is unavailable. Finally, Component D is separated into a separate service and the Event-Driven Messaging pattern is applied to establish a publisher-subscriber relationship between this new service and Component A. The interaction between Service Consumer A and Component A is then redesigned so that Component A first interacts with Component B and the new wrapper service. Service A then issues a final message back to Service Consumer A.

C. The Service Data Replication pattern can be applied so that Component B can access a replicated database instead of having to access the shared Database A directly. The Legacy Wrapper pattern can be applied so that Component C is separated into a separate service that acts as a wrapper of the legacy system API. Next, the Asynchronous Queuing pattern can be applied so that a messaging queue is positioned between Component A and the new wrapper service, thereby enabling communication during times when the legacy system is unavailable. Finally, Component D is separated into a new service and the Event-Driven Messaging pattern is applied to establish a publisher-subscriber relationship between this service and Component A and between Service A and Service Consumer A. The interaction logic is redesigned as follows: Component A interacts with Component B, the new wrapper service, and then issues a request to the new event-driven service. Upon receiving a response triggered by the event related to the batch import, Service A responds to Service Consumer A.

D. None of the above.

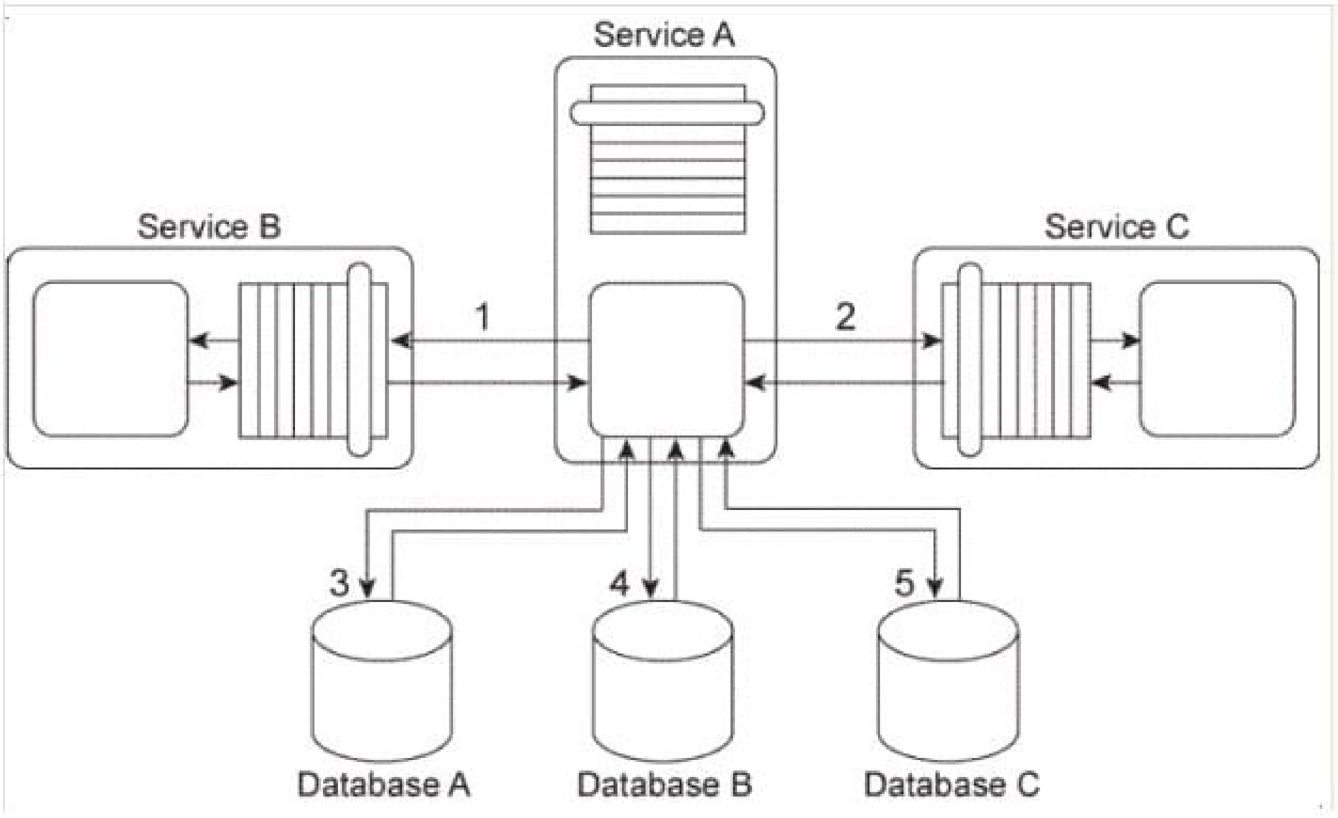
Correct Answer: C

QUESTION 4

Service A sends a message to Service B (1). After Service B writes the message contents to Database A

(2) it issues a response message back to Service A (3). Service A then sends a message to Service C (4). Upon receiving this message, Service C sends a message to Service D (5), which then writes the message contents to Database B (6) and issues a response message back to Service C (7).

Service A and Service D are in Service Inventory A. Service B and Service C are in Service Inventory B.



You are told that in this service composition architecture, all four services are exchanging invoice-related data in an XML format. However, the services in Service Inventory A are standardized to use a different XML schema for invoice data than the services in Service Inventory B. Also, Database A can only accept data in the Comma Separated Value (CSV) format and therefore cannot accept XML formatted data. Database B only accepts XML formatted data. However, it is a legacy database that uses a proprietary XML schema to represent invoice data that is different from the XML schema used by services in Service Inventory A or Service Inventory B. What steps can be taken to enable the planned data exchange between these four services?

A. The Data Model Transformation pattern can be applied so that data model transformation logic is positioned between Service A and Service B, between Service C and Service D, and between the Service D logic and Database B. The Data Format Transformation pattern can be applied so that data format transformation logic is positioned between Service A and Service C, and between the Service B

logic and Database A.

B. The Data Model Transformation pattern can be applied so that data model transformation logic is positioned between the Service B logic and Database A. The Data Format Transformation pattern can be applied so that data format transformation logic is positioned between Service A and Service B, between Service A and Service C, between Service C and Service D, and between the Service D logic and Database B.

C. The Data Model Transformation pattern can be applied so that data model transformation logic is positioned between Service A and Service B, between Service A and Service C, between Service C and Service D, and between the Service D logic and Database B. The Data Format Transformation pattern can be applied so that data format transformation logic is positioned between the Service B logic and Database A.

D. None of the above.

Correct Answer: C

QUESTION 5

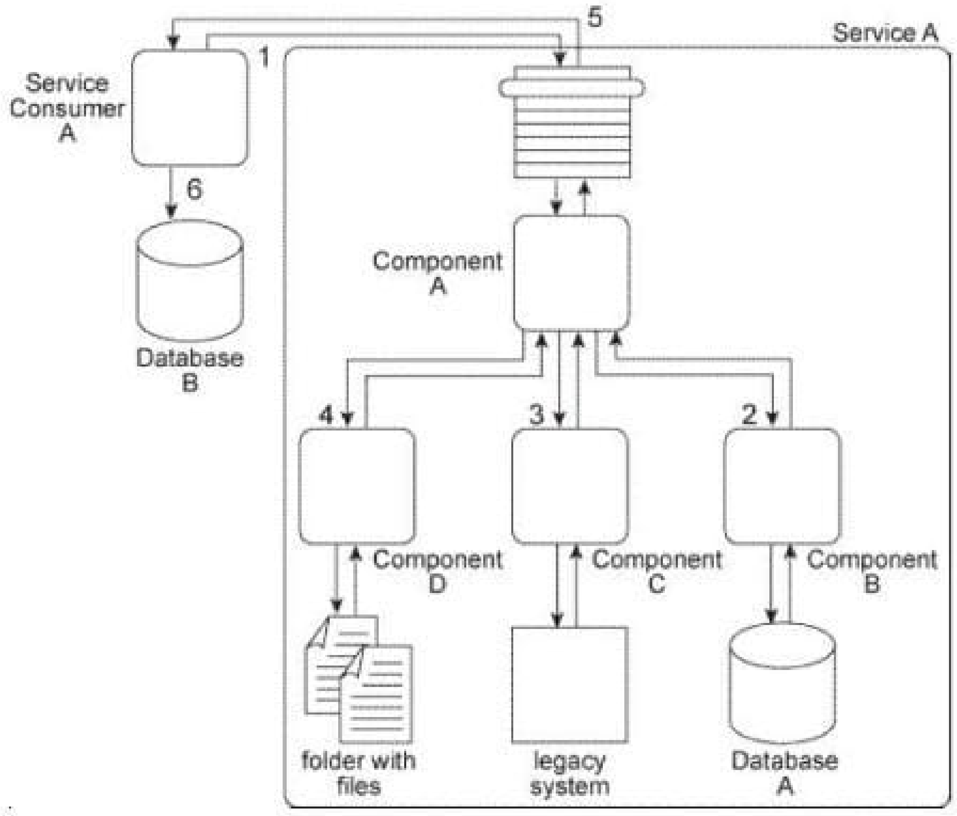
When Service A receives a message from Service Consumer A(1),the message is processed by Component A. This component first invokes Component B (2), which uses values from the message to query Database A in order to retrieve additional data. Component B then returns the additional data to Component A.

Component A then invokes Component C (3), which interacts with the API of a legacy system to retrieve a new data value. Component C then returns the data value back to Component A.

Next, Component A sends some of the data it has accumulated to Component D (4), which writes the data to a te>X file that is placed in a specific folder. Component D then waits until this file is imported into a different system via a regularly scheduled batch import. Upon completion of the import, Component D returns a success or failure code back to Component A.

Component A finally sends a response to Service Consumer A (5) containing all of the data collected so far and Service Consumer A writes all of the data to Database B (6).

Components A, B, C. and D belong to the Service A service architecture. Database A, the legacy system, and the file folders are shared resources within the IT enterprise.



Service A is an entity service with a service architecture that has grown over the past few years. As a result of a service inventory-wide redesign project, you are asked to revisit the Service A service architecture in order to separate the logic provided by Components B, C, and D into three different utility services without disrupting the behavior of Service A as it relates to Service Consumer A . What steps can be taken to fulfill these requirements?

A. The Legacy Wrapper pattern can be applied so that Component B is separated into a separate wrapper utility service that wraps the shared database. The Asynchronous Queuing pattern can be applied so that a messaging queue is positioned between Component A and Component C, thereby enabling communication during times when the legacy system may be unavailable or heavily accessed by other parts of the IT enterprise. The Service Facade pattern can be applied so that a Facade component is added between Component A and Component D so that any change in behavior can be compensated. The Service Autonomy principle can be further applied to Service A to help make up for any performance loss that may result from splitting the component into a separate wrapper utility service.

B. The Legacy Wrapper pattern can be applied so that Component B is separated into a separate utility service that wraps the shared database. The Legacy Wrapper pattern can be applied again so that Component C is separated into a separate utility service that acts as a wrapper for the legacy system API. The Legacy Wrapper pattern can be applied once more to Component D so that it is separated into another utility service that provides standardized access to the file folder. The Service Facade pattern can be applied so that three Facade components are added: one between Component A and each of the new wrapper utility services. This way, the Facade components can compensate for any change in behavior that may occur as a result of the separation. The Service Composability principle can be further applied to Service A and the three new wrapper utility services so that all four services are optimized for participation in the new service composition. This will help make up for any performance loss that may result from splitting the three components into separate services.

C. The Legacy Wrapper pattern can be applied so that Component B is separated into a separate utility service that wraps the shared database. The Legacy Wrapper pattern can be applied again so that Component C is separated into a separate utility service that acts as a wrapper for the legacy system API. Component D is separated into a separate service and the Event-Driven Messaging pattern is applied to establish a publisher-subscriber relationship between this new service and Component A. The interaction between Service Consumer A and Component A is then redesigned so that Component A first interacts with Component B and the new wrapper service. Service A then issues a final message back to Service Consumer A. The Service Composability principle can be further applied to Service A and the three new wrapper utility services so that all four services are optimized for participation in the new service composition. This will help make up for any performance loss that may result from splitting

the three components into separate services.

D. None of the above.

Correct Answer: B