S90.09 Q&As

SOA Design & Architecture Lab

Pass SOA S90.09 Exam with 100% Guarantee

Free Download Real Questions & Answers **PDF** and **VCE** file from:

https://www.geekcert.com/s90-09.html

100% Passing Guarantee
100% Money Back Assurance

Following Questions and Answers are all new published by SOA Official Exam Center

QUESTION 1

You are an architect with a project team building services for Service Inventory A . You are told that no SLAs for Service B and Service C are available. You cannot determine how available these services will be, but it has been confirmed that both of these services support atomic transactions and the issuance of positive and negative acknowledgements. However, you also find out that the services in Service Inventory B use different data models than the services in Service Inventory A. Furthermore, recent testing results have shown that the performance of Service D is steady and reliable. However, Service D uses a different transport protocol than the services in Service Inventory A. The response time of Service A is not a primary concern, but Service Consumer A does need to be able to issue request messages to Service A 24 hours a day without disruption. What steps can be taken to fulfill these requirements?

A. The Event-Driven Messaging pattern is applied so that a subscriber-publisher relationship is established between Service Consumer A and Service A . This gives Service A the flexibility to provide its response to Service Consumer A whenever it is able to collect the three data values without having to require that Service Consumer A remain stateful. The Asynchronous Queuing pattern is applied so that a central messaging queue is positioned between Service A and Service B and between Service A and Service C . The Data Model Transformation and Protocol Bridging patterns are applied to enable communication between Service A and Service B and between Service A and Service C . The Service Autonomy principle is further applied to Service A in order to improve its overall runtime behavioral predictability.

B. The Reliable Messaging pattern is applied so that a system of acknowledgements is established between Service Consumer A and Service A . This gives Service A the flexibility to provide Service Consumer A with acknowledgements that indicate that the processing steps that are occurring between Service A and Service B, Service C, and Service D are progressing. The Asynchronous Queuing pattern is applied so that a central messaging queue is positioned between Service A and Service B and between Service A and Service C and between Service A and Service D . The Redundant Implementation pattern is applied so that a copy of Service D is brought in-Upon reviewing these requirements it becomes D with a standardized service contract that is in compliance with the design standards used in Service Inventory A.

C. The Asynchronous Queuing pattern is applied so that a central messaging queue is positioned between Service A and Service B and between Service A and Service C and between Service A and Service D and so that a separate messaging queue is positioned between Service A and Service Consumer A. The Data Model Transformation pattern is applied to enable communication between Service A and Service B and between Service A and Service C . The Protocol Bridging pattern is applied to enable communication between Service A and Service D .

D. None of the above.

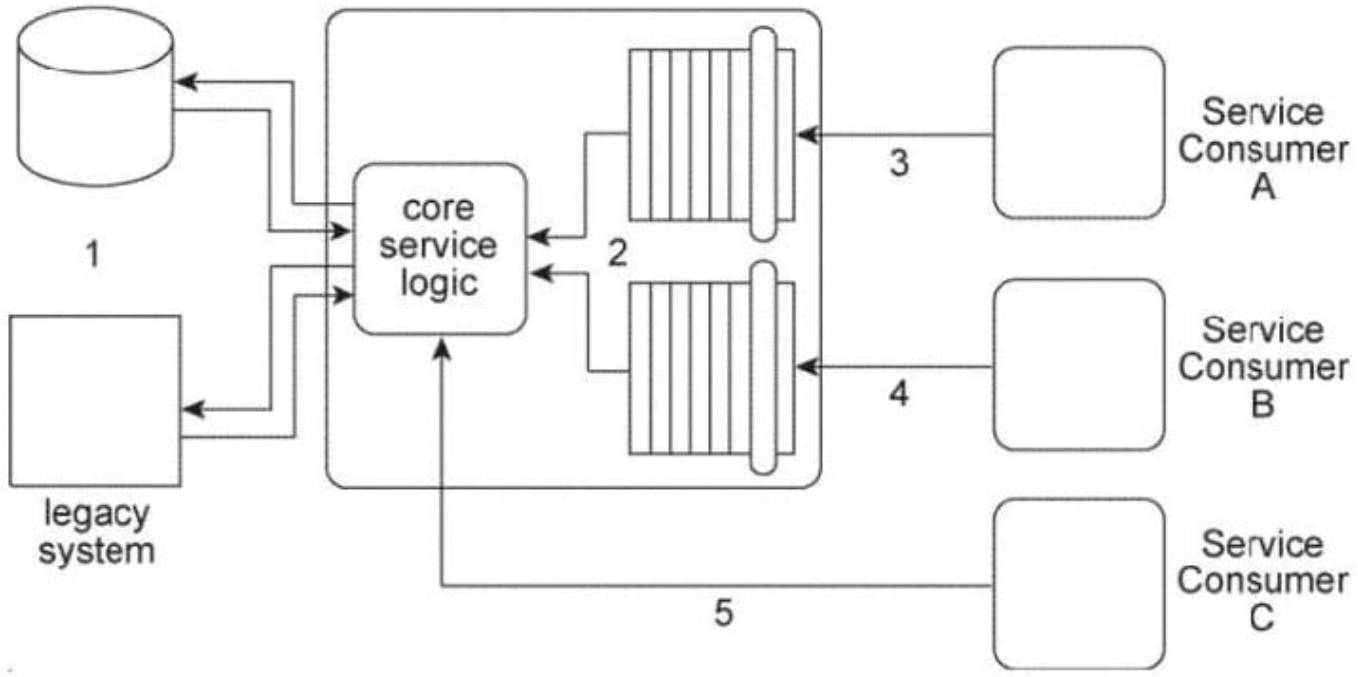
Correct Answer: C

QUESTION 2

The architecture for Service A displayed in the Figure shows how the core logic of Service A has expanded over time to connect to a database and a proprietary legacy system (1) and to support two separate service contracts (2) that are accessed by different service consumers.

The service contracts are fully decoupled from the service logic. The service logic is therefore coupled to the service contracts and to the underlying implementation resources (the database and the legacy system).

Service A currently has three service consumers. Service Consumer A and Service Consumer B access Service A\\'s two service contracts (3, 4). Service Consumer C bypasses the service contracts and accesses the service logic directly (5).



You are told that the database and legacy system that are currently being used by Service A are being replaced with different products. The two service contracts are completely decoupled from the core service logic, but there is still a concern that the introduction of the new products will cause the core service logic to behave differently than before. What steps can be taken to change the Service A architecture in preparation for the introduction of the new products so that the impact on Service Consumers A, B, and C is minimized?

A. The Service Abstraction principle can be applied to hide the implementation details from the core service logic of Service A, thereby shielding this logic from changes to the implementation. In support of this, the Service Facade pattern can be applied to position Facade components between the core service logic and Service Consumers A and B. These Facade components will be designed to regulate the behavior of Service A . The Contract Centralization pattern can be applied to force Service Consumer C to access Service A via one of its existing service contracts.

B. A third service contract can be added together with the application of the Contract Centralization pattern. This will force Service Consumer C to access Service A via the new service contract. The Service Facade pattern can be applied to position a Facade component between the new service contract and Service Consumer C in order to regulate the behavior of Service A . The Service Abstraction principle can be applied to hide the implementation details of Service A so that no future

service consumers are designed to access any of Service A\\'s underlying resources directly.

C. The Service Facade pattern can be applied to position Facade components between the core service logic and the two service contracts. These Facade components will be designed to regulate the behavior of Service A . The Contract Centralization pattern can also be applied to force Service Consumer C to access Service A via one of its existing service contracts.

D. None of the above.

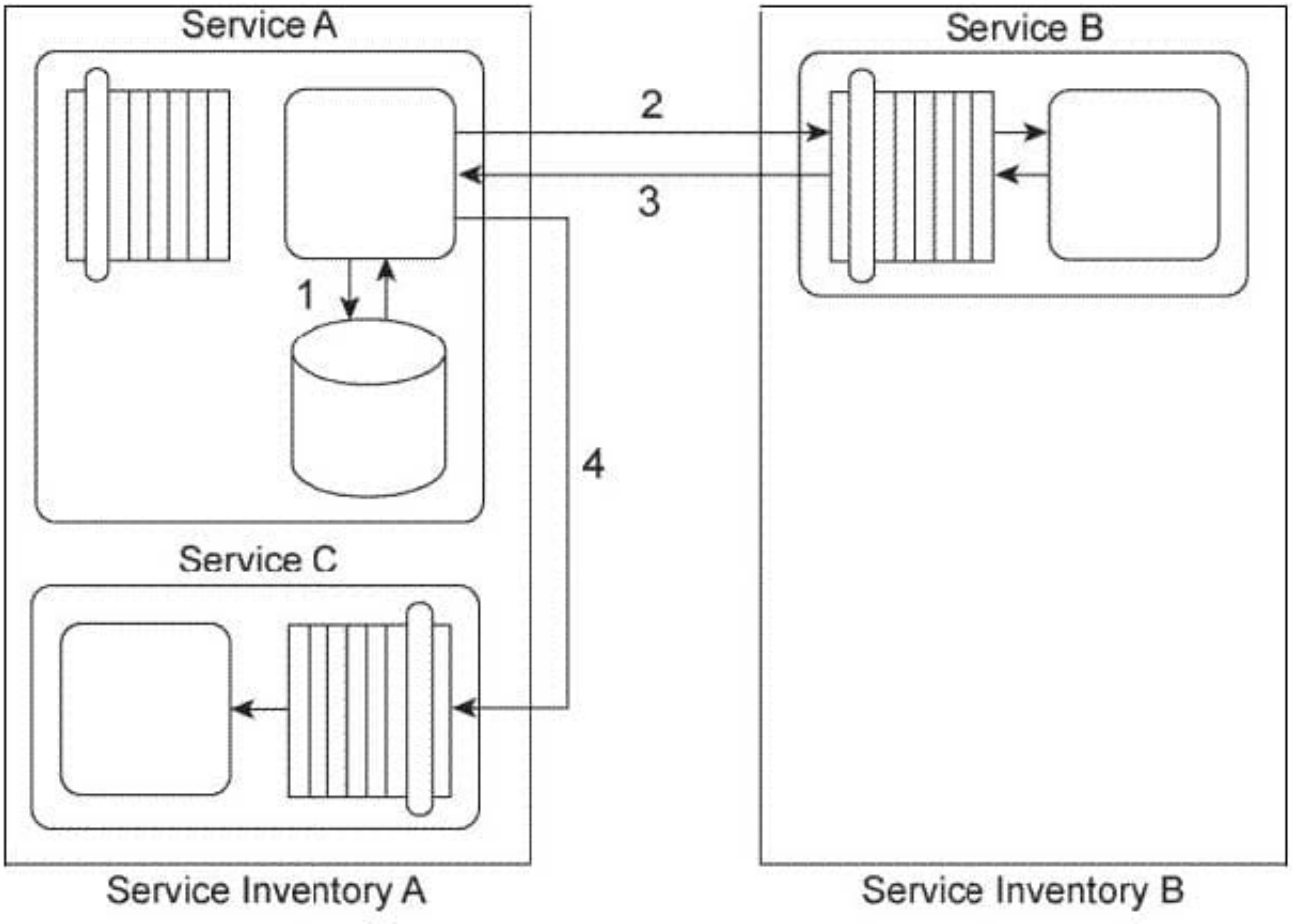
Correct Answer: C

QUESTION 3

Service A is a task service that sends Service B a message (2) requesting that Service B return data back to Service A in a response message (3). Depending on the response received. Service A may be required to send a message to Service C (4) for which it requires no response.

Before it contacts Service B, Service A must first retrieve a list of code values from its own database (1) and then place this data into its own memory. If it turns out that it must send a message to Service C, then Service A must combine the data it receives from Service B with the data from the code value list in order to create the message it sends to Service C. If Service A is not required to invoke Service C, it can complete its task by discarding the code values.

Service A and Service C reside in Service Inventory A. Service B resides in Service Inventory B.



You are told that the services in Service Inventory A were designed with service contracts based on different design standards than the services in Service Inventory B. As a result, Service A and Service B use different data models to represent the data they need to exchange. Therefore, Service A and Service B cannot currently communicate. Furthermore, Service C is an agnostic service that is heavily accessed by many concurrent service consumers. Service C frequently reaches its usage thresholds during which it is not available and messages sent to it are not received. How can this service composition architecture be changed to avoid these problems?

A. The Data Model Transformation pattern can be applied by establishing an intermediate processing layer between

Service A and Service B that can transform a message from one data model to another at runtime. The Intermediate Routing and Service Agent patterns can be applied so that when Service B sends a response message, a service agent can intercept the message and, based on its contents, either forward the message to Service A or route the message to Service C . The Service Autonomy principle can be further applied to Service C together with the Redundant Implementation pattern to help establish a more reliable and scalable service architecture.

B. The Data Model Transformation pattern can be applied by establishing an intermediate processing layer between Service A and Service B that can transform a message from one data model to another at runtime. The Asynchronous Queuing pattern can be applied to establish an intermediate queue between Service A and Service C so that when Service A needs to send a message to Service C, the queue will store the message and retransmit it to Service C until it is successfully delivered. The Service Autonomy principle can be further applied to Service C together with the Redundant Implementation pattern to help establish a more reliable and scalable service architecture.

C. The Data Model Transformation pattern can be applied by establishing an intermediate processing layer between Service A and Service B that can transform a message from one data model to another at runtime. The Intermediate Routing and Service Agent patterns can be applied so that when Service B sends a response message, a service agent can intercept the message and, based on its contents, either forward the message to Service A or route the message to Service C . The Service Statelessness principle can be applied with the help of the State Repository pattern so that Service A can write the code value data to a state database while it is waiting for Service B to respond.

D. None of the above.

Correct Answer: B

QUESTION 4

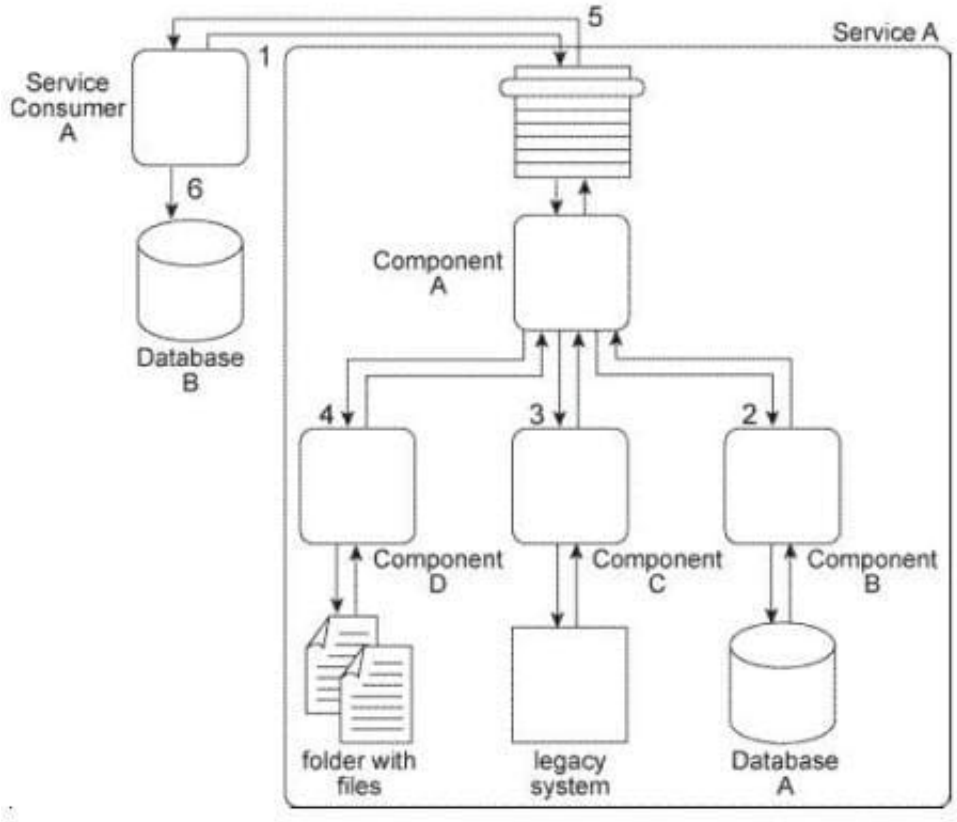
When Service A receives a message from Service Consumer A(1),the message is processed by Component A. This component first invokes Component B (2), which uses values from the message to query Database A in order to retrieve additional data. Component B then returns the additional data to Component A.

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.

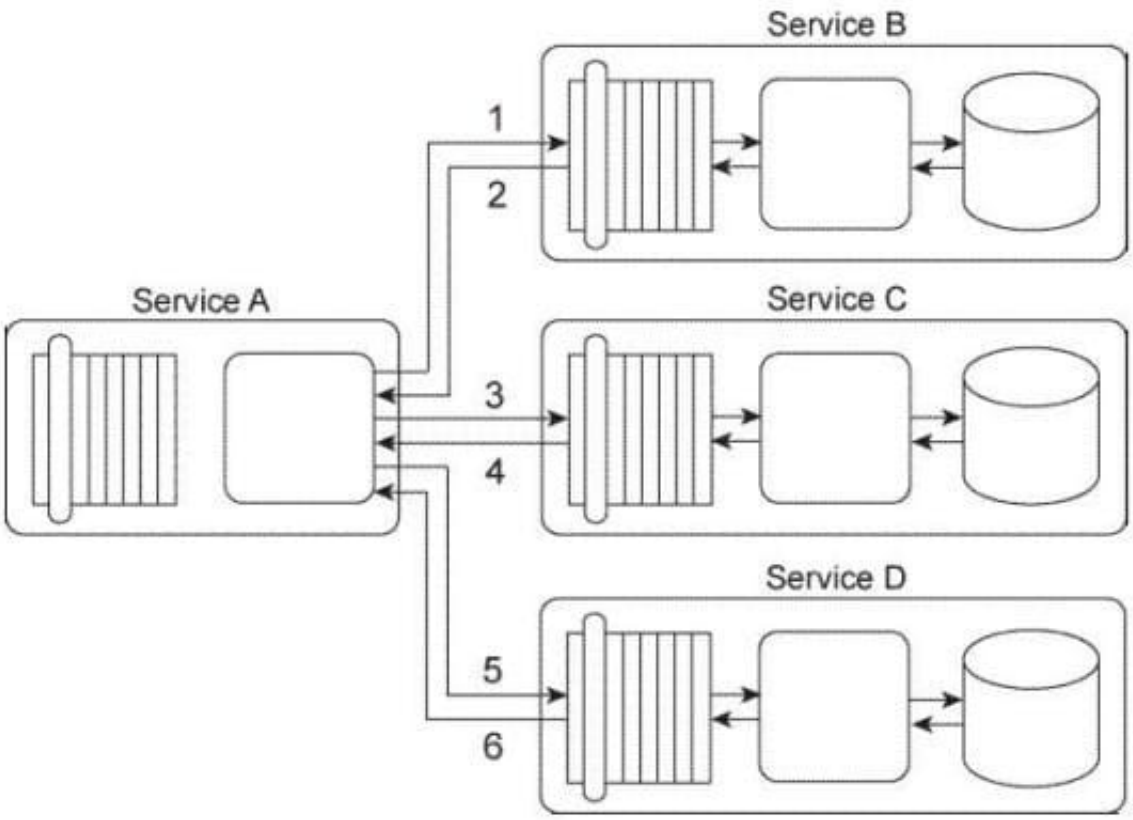
D. None of the above.

Correct Answer: B

QUESTION 5

Service A is a task service that is required to carry out a series of updates to a set of databases in order to complete a task. To perform the database updates Service A must interact with three other services, each of which provides standardized data access capabilities.

Service A sends its first update request message to Service B (1), which then responds with a message containing a success or failure code (2). Service A then sends its second update request message to Service C (3), which also responds with a message containing a success or failure code (4). Finally, Service A sends a request message to Service D (5), which responds with its own message containing a success or failure code (6).



You\\'ve been given a requirement that all database updates must either be completed successfully or not at all. This means that if any of the three response messages received by Service A contain a failure code, all of the updates carried out until that point must be reversed. Note that if Service A does not receive a response message back from Services B, C, or D, it must assume that a failure has occurred. How can this service composition architecture be changed to fulfill these requirements?

A. The Reliable Messaging pattern can be applied to guarantee the delivery of positive or negative acknowledgements. This way, Service A will always be informed of whether a failure condition has occurred with any of the database updates performed by Services B, C, and D. Furthermore, the Service Loose Coupling principle can be applied to ensure that the request and response messages exchanged by the services do not contain any implementation details that would indirectly couple Service A to any of the databases.

B. The Atomic Service Transaction pattern can be applied individually to Services B, C, and D so that each of these services performs its own database update within the scope of an atomic transaction. If anyone update fails, that change can be rolled back on that database. Furthermore, the Service Loose Coupling principle can be applied to ensure that Service A is kept out of the scope of the atomic transaction so that it is not negatively coupled to the proprietary database technologies that are required to enable the atomic transaction functionality.

C. The Compensating Service Transaction can be applied to Service A so that when any one response message containing a failure code is received by Service A, it can invoke exception handling logic that will log the failed database updates. The Service Loose Coupling principle can be further applied to ensure that Services B, C, or D are not indirectly coupled to the exception handling logic, especially if Service A requires additional access to Services B, C, or D in order to collect more information for logging purposes.

D. None of the above.

Correct Answer: D