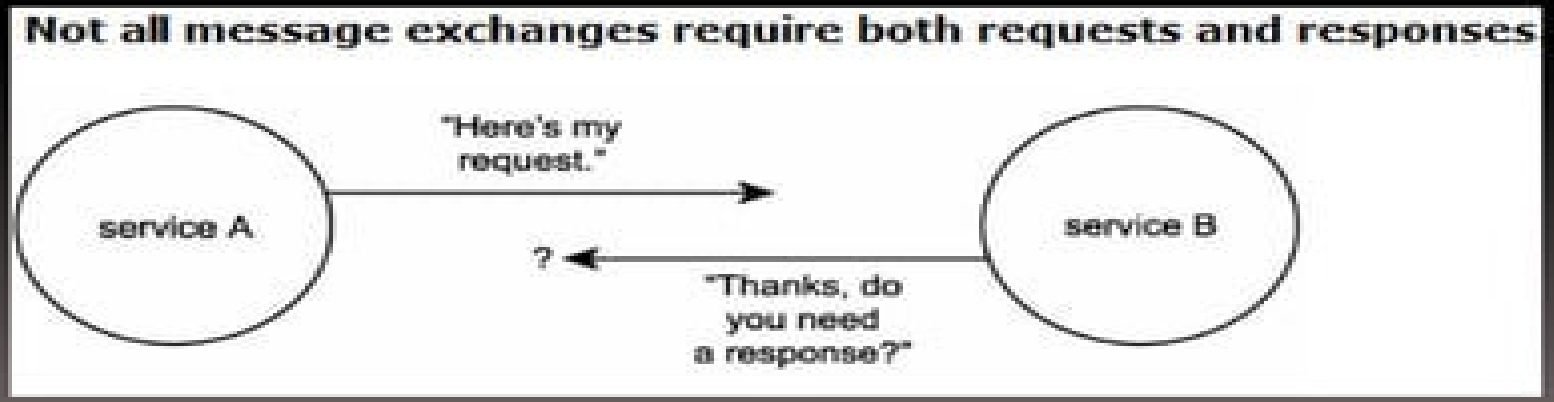
WEB SERVICES AND CONTEMPORARY SOA (PART I: ACTIVITY MANAGEMENT AND COMPOSITION)

Message exchange patterns (MEP)

- Every task automated by a Web service can differ in both the nature of the application logic being executed and the role played by the service in the overall execution of the business task.
- Regardless of how complex a task is, almost all require the transmission of multiple messages.
- The challenge lies in coordinating these messages in a particular sequence so that the individual actions performed by the message are executed properly and in alignment with the overall business task



WEB SERVICES AND CONTEMPORARY SOA (PART I: ACTIVITY MANAGEMENT AND COMPOSITION)

Message exchange patterns (MEP)

- Message exchange patterns (MEPs) represent a set of templates that provide a group of already mapped out sequences for the exchange of messages.
- The most common example is a request and response pattern.
- Here the MEP states that upon successful delivery of a message from one service to another, the receiving service responds with a message back to the initial requestor.
- Many MEPs have been developed, each addressing a common message exchange requirement.
- It is useful to have a basic understanding of some of the more important MEPs, as no doubt be finding while applying MEPs to specific communication requirements when designing service-oriented solutions.
- An MEP is like a type of conversation. It's not a long conversation; it actually only covers one exchange between two parties

WEB SERVICES AND CONTEMPORARY SOA (PART I: ACTIVITY MANAGEMENT AND COMPOSITION)

Primitive MEPs

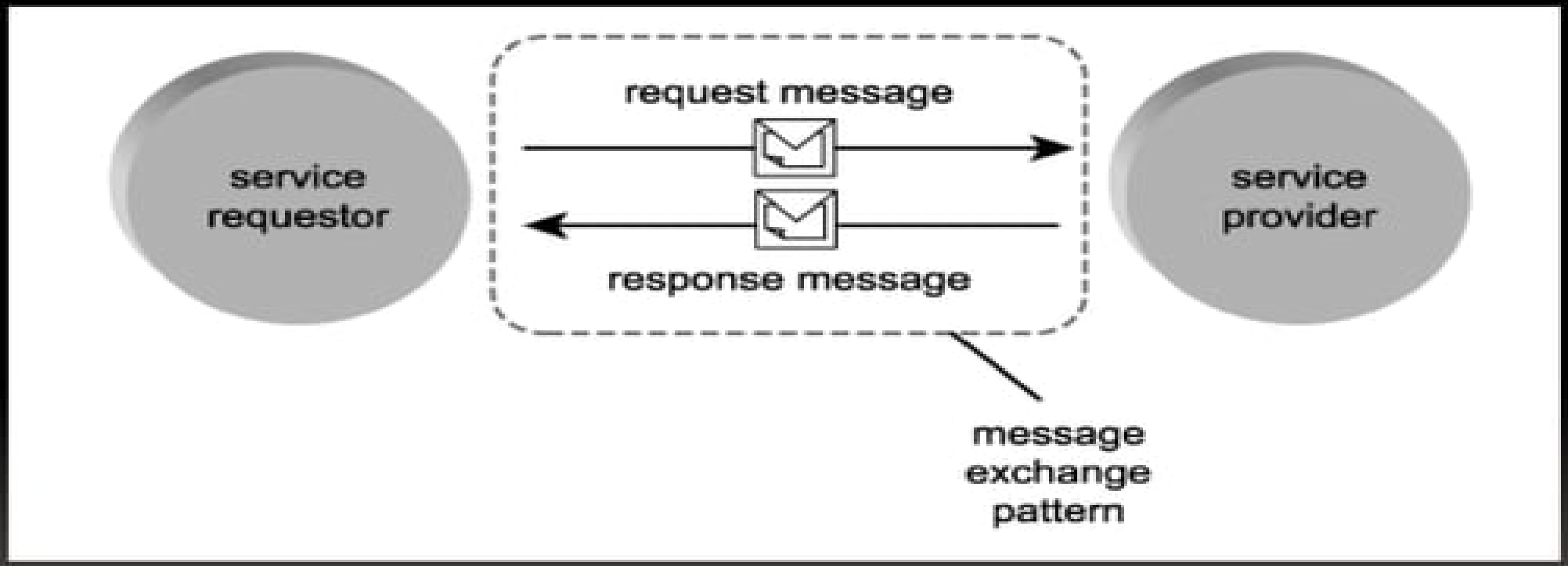
- Before the arrival of contemporary SOA, messaging frameworks were already well used by various messaging-oriented middleware products. As a result, a common set of primitive MEPs has been in existence for some time.

Request-response MEP

- This is the most popular MEP in use among distributed application environments and the one pattern that defines synchronous communication (although this pattern also can be applied asynchronously).
- The request-response MEP establishes a simple exchange in which a message is first transmitted from a source (service requestor) to a destination (service provider).
- Upon receiving the message, the destination (service provider) then responds with a message back to the source (service requestor).

WEB SERVICES AND CONTEMPORARY SOA (PART I: ACTIVITY MANAGEMENT AND COMPOSITION)

The request-response MEP



WEB SERVICES AND CONTEMPORARY SOA (PART I: ACTIVITY MANAGEMENT AND COMPOSITION)

Fire-and-forget MEP

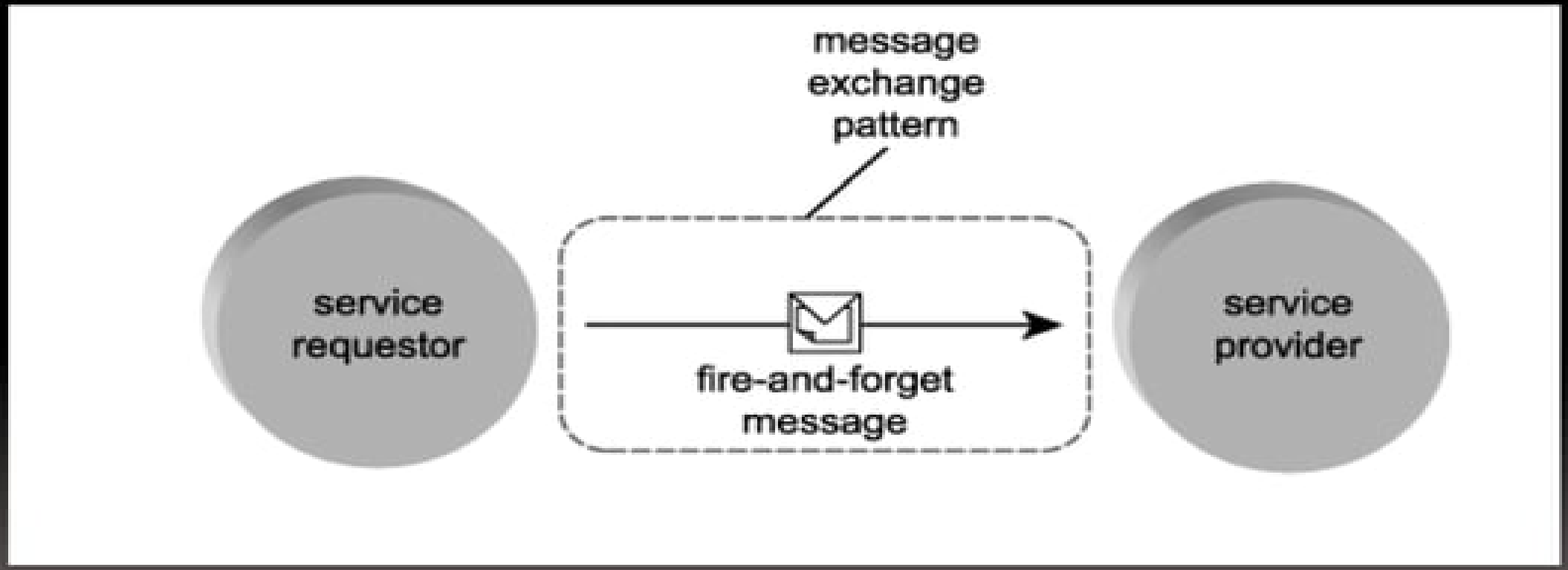
- simple asynchronous pattern is based on the unidirectional transmission of messages from a source to one or more destinations

Variations of the fire-and-forget :

- The single-destination pattern, where a source sends a message to one destination only.
- The multi-cast pattern, where a source sends messages to a predefined set of destinations.
- The broadcast pattern, which is similar to the multi-cast pattern, except that the message is sent out to a broader range of recipient destinations.
- The fundamental characteristic of the fire-and-forget pattern is that a response to a transmitted message is not expected

WEB SERVICES AND CONTEMPORARY SOA (PART I: ACTIVITY MANAGEMENT AND COMPOSITION)

Fire-and-forget MEP



WEB SERVICES AND CONTEMPORARY SOA (PART I: ACTIVITY MANAGEMENT AND COMPOSITION)

Complex MEPs

- Even though a message exchange pattern can facilitate the execution of a simple task, it is really more of a building block intended for composition into larger patterns.
- Primitive MEPs can be assembled in various configurations to create different types of messaging models, sometimes called complex MEPs.
- A classic example is the publish-and-subscribe model.
- The publish-and-subscribe pattern introduces new roles for the services involved with the message exchange.
- They now become publishers and subscribers, and each may be involved in the transmission and receipt of messages.
- This asynchronous MEP accommodates a requirement for a publisher to make its messages available to a number of subscribers interested in receiving them.

Complex MEPs

The steps involved :

Step 1:

- The subscriber sends a message to notify the publisher that it wants to receive messages on a particular topic.

Step 2:

- Upon the availability of the requested information, the publisher broadcasts messages on the particular topic to all of that topic's subscribers.

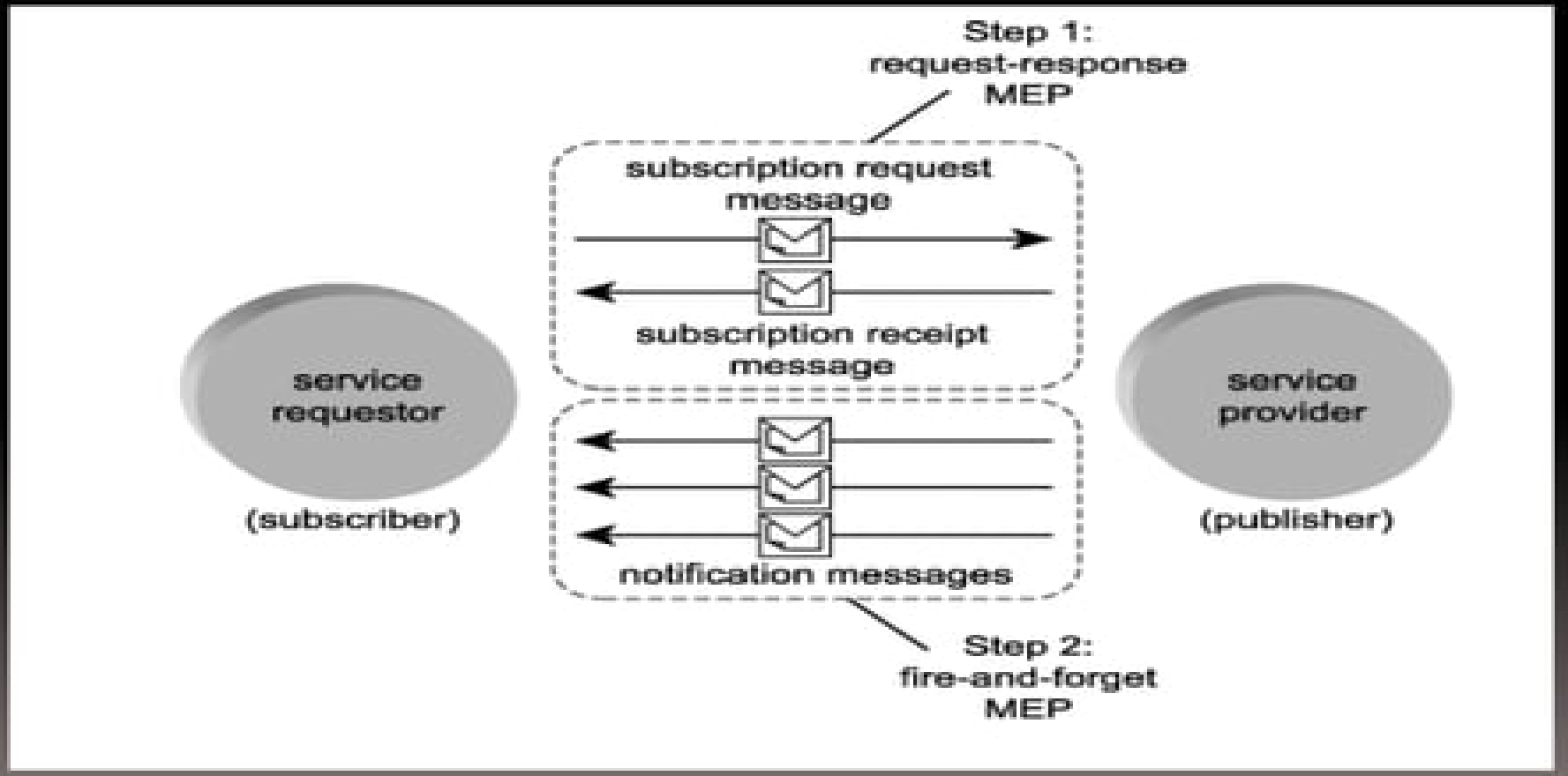
Complex MEPs

- This pattern is a great example of how to aggregate primitive MEPs
- Step 1 in the publish-and-subscribe MEP could be implemented by a request-response MEP, where the subscriber's request message, indicating that it wants to subscribe to a topic, is responded to by a message from the publisher, confirming that the subscription succeeded or failed.
- Step 2 then could be supported by one of the fire-and-forget patterns, allowing the publisher to broadcast a series of unidirectional messages to subscribers.

WEB SERVICES AND CONTEMPORARY SOA (PART I: ACTIVITY MANAGEMENT AND COMPOSITION)

Complex MEPs

The publish-and-subscribe messaging model is a composite of two primitive MEPs



WEB SERVICES AND CONTEMPORARY SOA (PART I: ACTIVITY MANAGEMENT AND COMPOSITION)

Complex MEPs

WS-* specifications that incorporate this messaging model include:

- WS-BaseNotification
- WS-BrokeredNotification
- WS-Topics
- WS-Eventing

WEB SERVICES AND CONTEMPORARY SOA (PART I: ACTIVITY MANAGEMENT AND COMPOSITION)

MEPs and SOAP

- On its own, the SOAP standard provides a messaging framework designed to support single-direction message transfer.
- The extensible nature of SOAP allows countless messaging characteristics and behaviors (MEP-related and otherwise) to be implemented via SOAP header blocks.
- The SOAP language also provides an optional parameter that can be set to identify the MEP associated with a message.

WEB SERVICES AND CONTEMPORARY SOA (PART I: ACTIVITY MANAGEMENT AND COMPOSITION)

MEPs and WSDL

- Operations defined within service descriptions are comprised, in part, of message definitions.
- The exchange of these messages constitutes the execution of a task represented by an operation.
- MEPs play a larger role in WSDL service descriptions as they can coordinate the input and output messages associated with an operation.
- The association of MEPs to WSDL operations thereby embeds expected conversational behavior into the interface definition.
- WSDL operations support different configurations of incoming, outgoing, and fault messages.
- These configurations are equivalent to message exchange patterns, but within the WSDL specification, they often are referred to simply as patterns.

WEB SERVICES AND CONTEMPORARY SOA (PART I: ACTIVITY MANAGEMENT AND COMPOSITION)

MEPs and WSDL

- Release 1.1 of the WSDL specification provides support for four message exchange patterns that roughly correspond to the MEPs. These patterns are applied to service operations from the perspective of a service provider or endpoint.

In WSDL 1.1 terms, they are represented as follows:

Request-response operation:

- Upon receiving a message, the service must respond with a standard message or a fault message.

Solicit-response operation:

- Upon submitting a message to a service requestor, the service expects a standard response message or a fault message.

One-way operation :

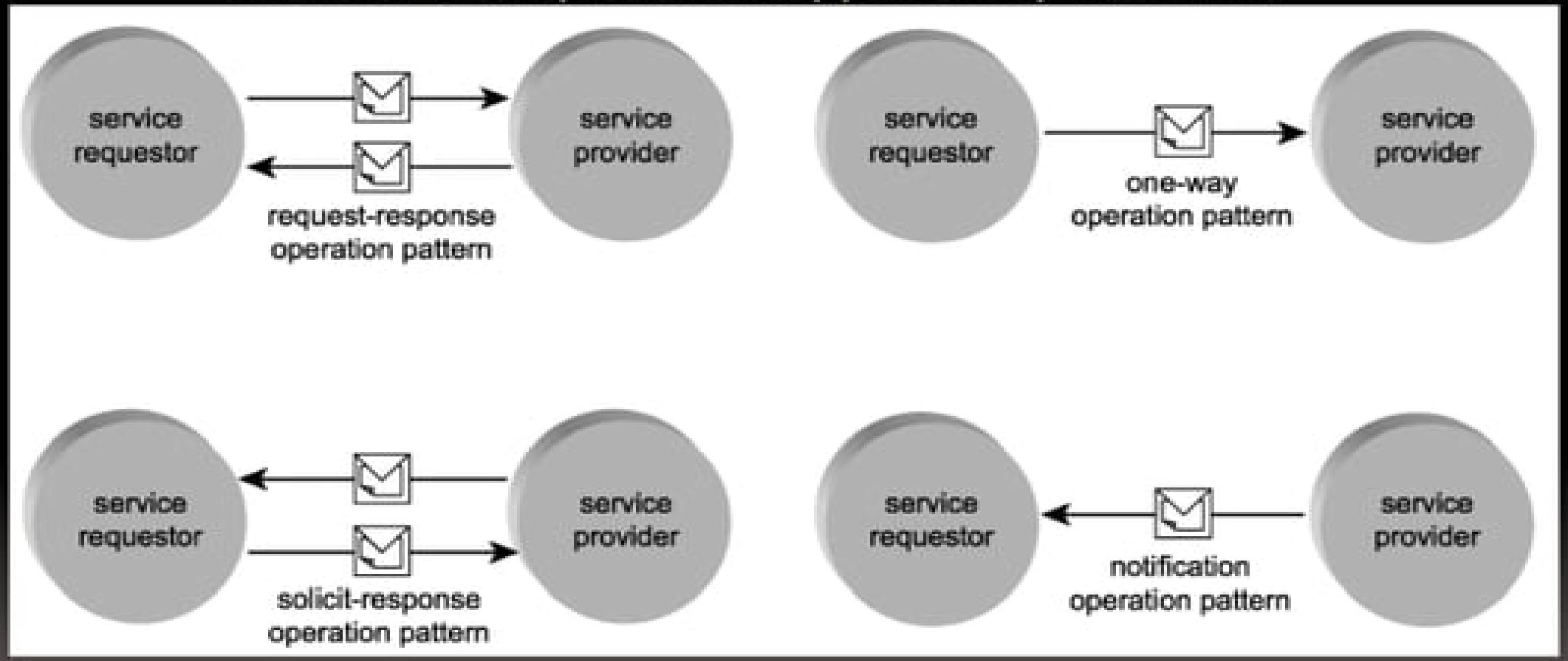
- The service expects a single message and is not obligated to respond.

Notification operation:

- The service sends a message and expects no response.

WEB SERVICES AND CONTEMPORARY SOA (PART I: ACTIVITY MANAGEMENT AND COMPOSITION)

The four basic patterns supported by WSDL 1.1



WEB SERVICES AND CONTEMPORARY SOA (PART I: ACTIVITY MANAGEMENT AND COMPOSITION)

release 2.0 of the WSDL specification extends MEP support to eight patterns (and also changes the terminology) as follows.

The in-out pattern

comparable to the request-response MEP (and equivalent to the WSDL 1.1 request-response operation).

The out-in pattern

which is the reverse of the previous patternwhere the service provider initiates the exchange by transmitting the request. (Equivalent to the WSDL 1.1 solicit-response operation.)

The in-only pattern

which essentially supports the standard fire-and-forget MEP. (Equivalent to the WSDL 1.1 one-way operation.)

The out-only pattern

which is the reverse of the in-only pattern. It is used primarily in support of event notification. (Equivalent to the WSDL 1.1 notification operation.)

WEB SERVICES AND CONTEMPORARY SOA (PART I: ACTIVITY MANAGEMENT AND COMPOSITION)

The robust in-only pattern

a variation of the in-only pattern that provides the option of launching a fault response message as a result of a transmission or processing error.

The robust out-only pattern

which, like the out-only pattern, has an outbound message initiating the transmission. The difference here is that a fault message can be issued in response to the receipt of this message.

The in-optional-out pattern

which is similar to the in-out pattern with one exception. This variation introduces a rule stating that the delivery of a response message is optional and should therefore not be expected by the service requestor that originated the communication. This pattern also supports the generation of a fault message.

The out-optional-in pattern

which is the reverse of the in-optional-out pattern, where the incoming message is optional. Fault message generation is again supported.

WEB SERVICES AND CONTEMPORARY SOA (PART I: ACTIVITY MANAGEMENT AND COMPOSITION)

Service activity

- The completion of business tasks is an obvious function of any automated solution.
- Tasks are comprised of processing logic that executes to fulfill a number of business requirements.
- In service-oriented solutions, each task can involve any number of services.
- The interaction of a group of services working together to complete a task can be referred to as a service activity.

In an activity, multiple Web services collaborate to do a specific piece of work.

WEB SERVICES AND CONTEMPORARY SOA (PART I: ACTIVITY MANAGEMENT AND COMPOSITION)

In Plain English

Any type of task will consist of one or more steps. The task of turning on the TV is relatively simple, consisting of perhaps the following two steps:

1. Pick up the remote control.
2. Press the "Power" button.

The task of washing a car, on the other hand, can be a bit more omplicated. It could exist of a series of steps, including:

1. Locate bucket.
2. Locate sponge.
3. Locate hose.
4. Fill bucket with warm water.
5. Add soap to water.
6. Soak sponge in water.
7. Rub sponge on car....and so on.

WEB SERVICES AND CONTEMPORARY SOA (PART I: ACTIVITY MANAGEMENT AND COMPOSITION)

The steps that comprise this more complex task could be summarized into a series of simple (or primitive) tasks, as follows:

1. Gather required equipment.
2. Prepare water.
3. Wash car.

- Each simple task consists of a smaller number of steps.
- Collectively these simple tasks represent a larger, logical unit of work.
- Individually, simple tasks do not accomplish anything of relevance, primarily because each subsequent task is dependent on the completion of the former. It is only when they are assembled into a complex task that they represent a useful unit of work.

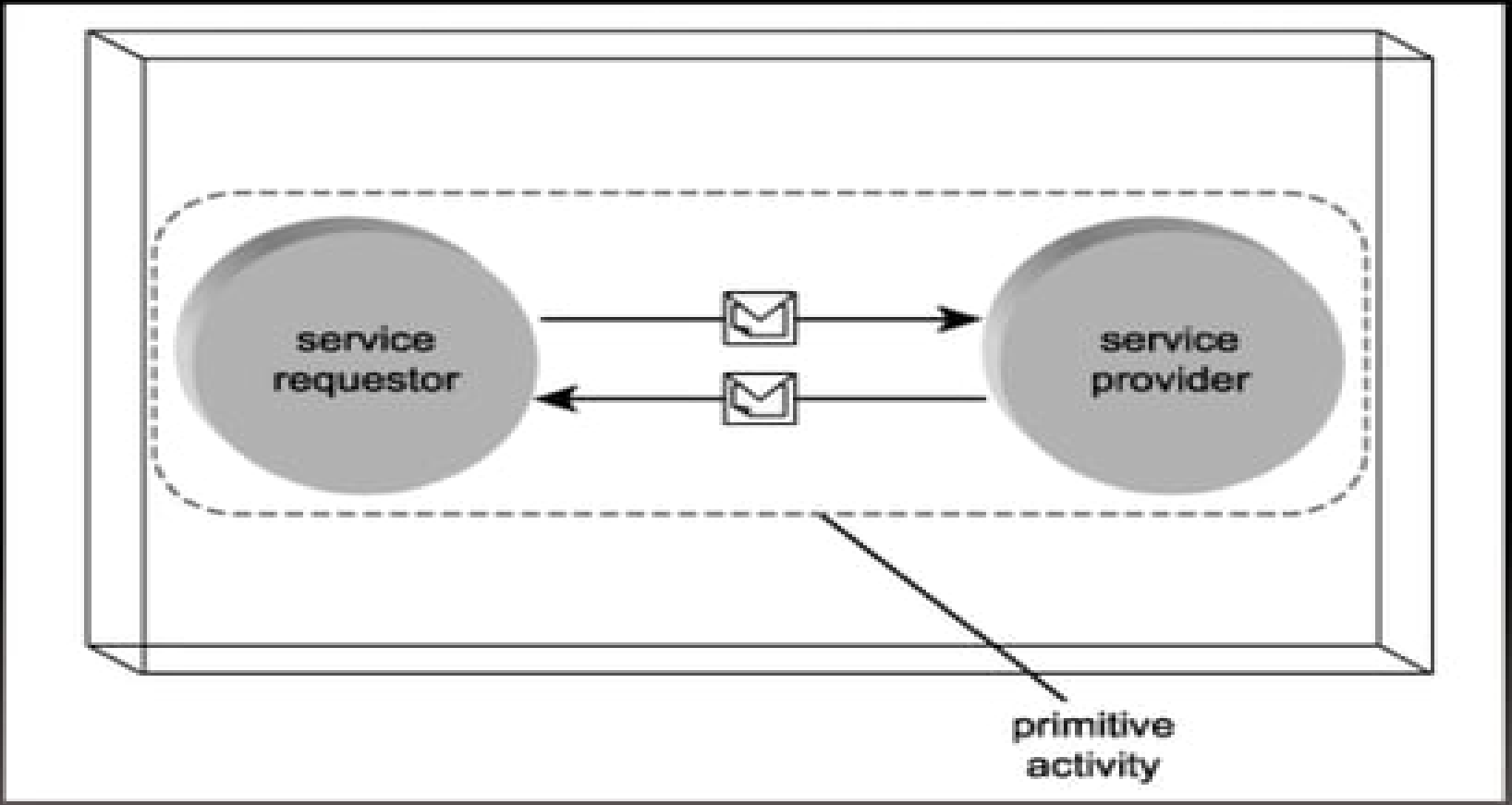
WEB SERVICES AND CONTEMPORARY SOA (PART I: ACTIVITY MANAGEMENT AND COMPOSITION)

Primitive and complex service activities

- The scope of an activity can drastically vary.
- A simple or primitive activity is typified by synchronous communication and consists of two services exchanging information using a standard request-response MEP .
- Primitive activities are almost always short-lived;
- the execution of a single MEP generally constitutes the lifespan of a primitive activity.

WEB SERVICES AND CONTEMPORARY SOA (PART I: ACTIVITY MANAGEMENT AND COMPOSITION)

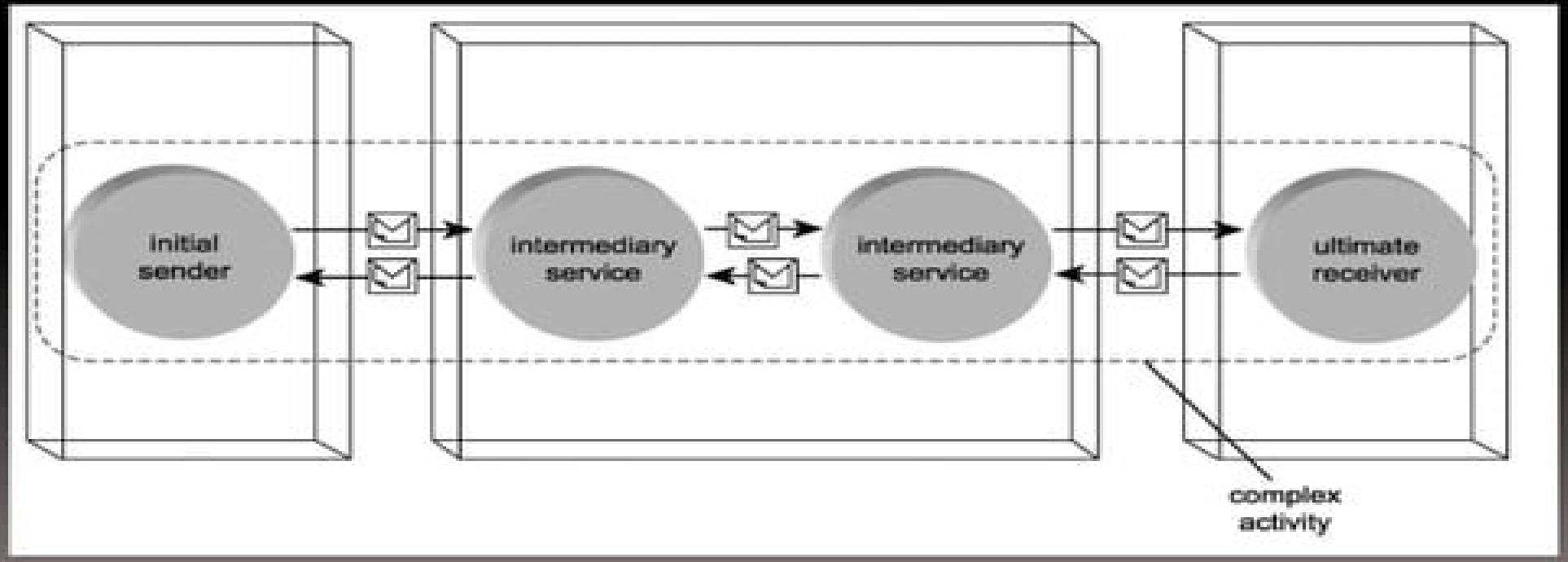
A primitive service activity consisting of a simple MEP



WEB SERVICES AND CONTEMPORARY SOA (PART I: ACTIVITY MANAGEMENT AND COMPOSITION)

- Complex activities can involve many services (and MEPs) that collaborate to complete multiple processing steps over a long period of time.
- These more elaborate types of activities are generally structured around extension-driven and composition-oriented concepts, such as choreography and orchestration.

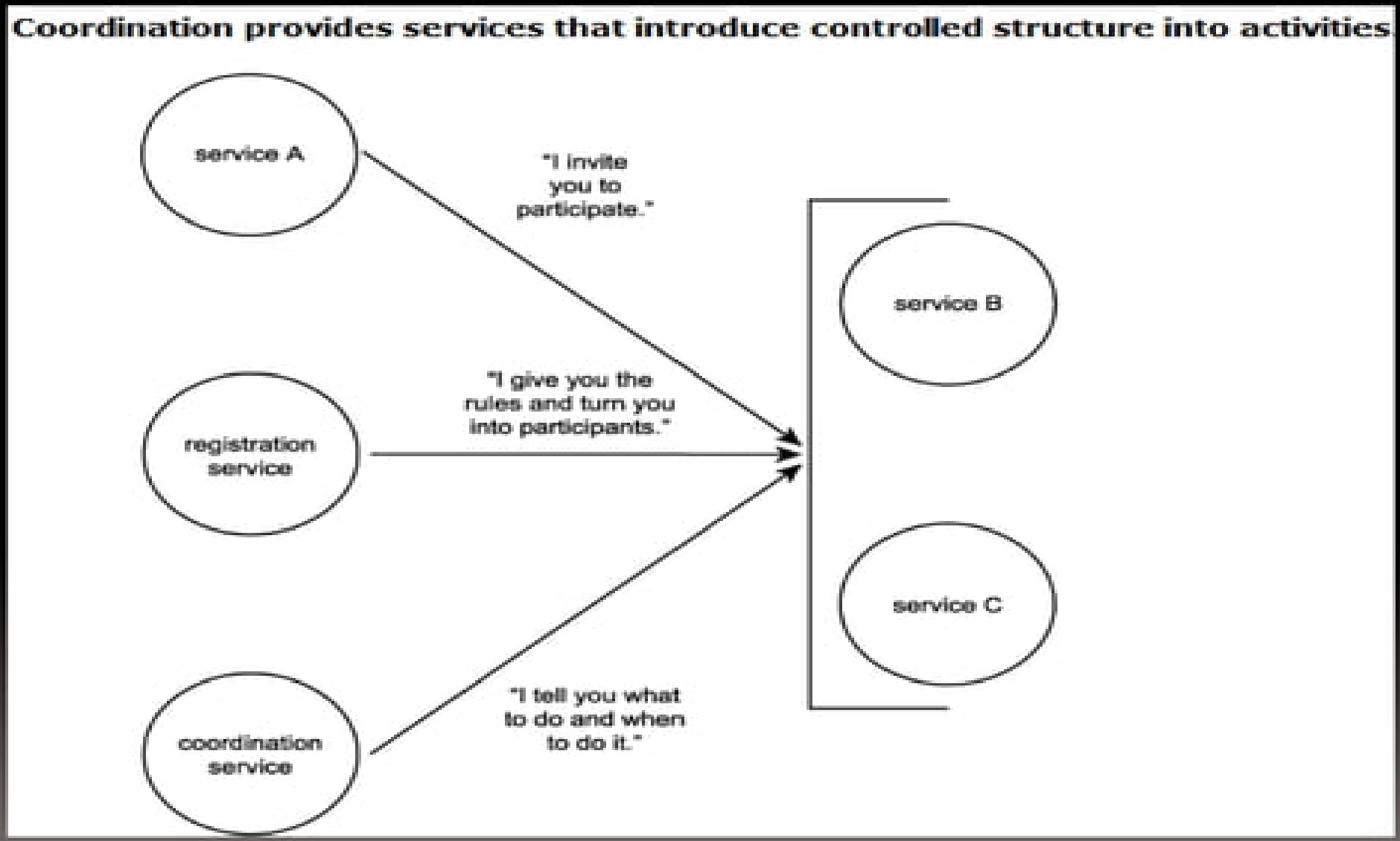
A complex activity involving four services



Coordination

- Every activity introduces a level of context into an application runtime environment.
- Something that is happening or executing has meaning during its lifetime, and the description of its meaning (and other characteristics that relate to its existence) can be classified as context information.
- The more complex an activity, the more context information it tends to bring with it.
- The complexity of an activity can relate to a number of factors, including:
  - the amount of services that participate in the activity
  - the duration of the activity
  - the frequency with which the nature of the activity changes
  - whether or not multiple instances of the activity can concurrently exist
- A framework is required to provide a means for context information in complex activities to be managed, preserved and/or updated, and distributed to activity participants. Coordination establishes such a framework

WEB SERVICES AND CONTEMPORARY SOA (PART I: ACTIVITY MANAGEMENT AND COMPOSITION)



In Plain English

I decide to get Chuck, Bob, and Jim to wash my car. When we're all ready to go, I assign each of them a simple task, as follows:

ChuckGather equipment.

BobPrepare the water.

JimWash the car.

You might recall each of these simple tasks consists of a series of steps. Chuck's task, for instance, is comprised of the following steps:

1. Locate bucket.
2. Locate sponge.
3. Locate hose.

Completion of the first step is required before Bob can begin his task of preparing the water. Completion of Chuck's and Bob's tasks is required for Jim to start washing the car. However, allowing Bob to start on his task after Chuck completes only the first step of his task will allow both Chuck and Bob to complete their respective tasks at around the same time. The benefit here is that Jim can start his task sooner than if Bob had to wait for Chuck to fully complete all steps in his task.

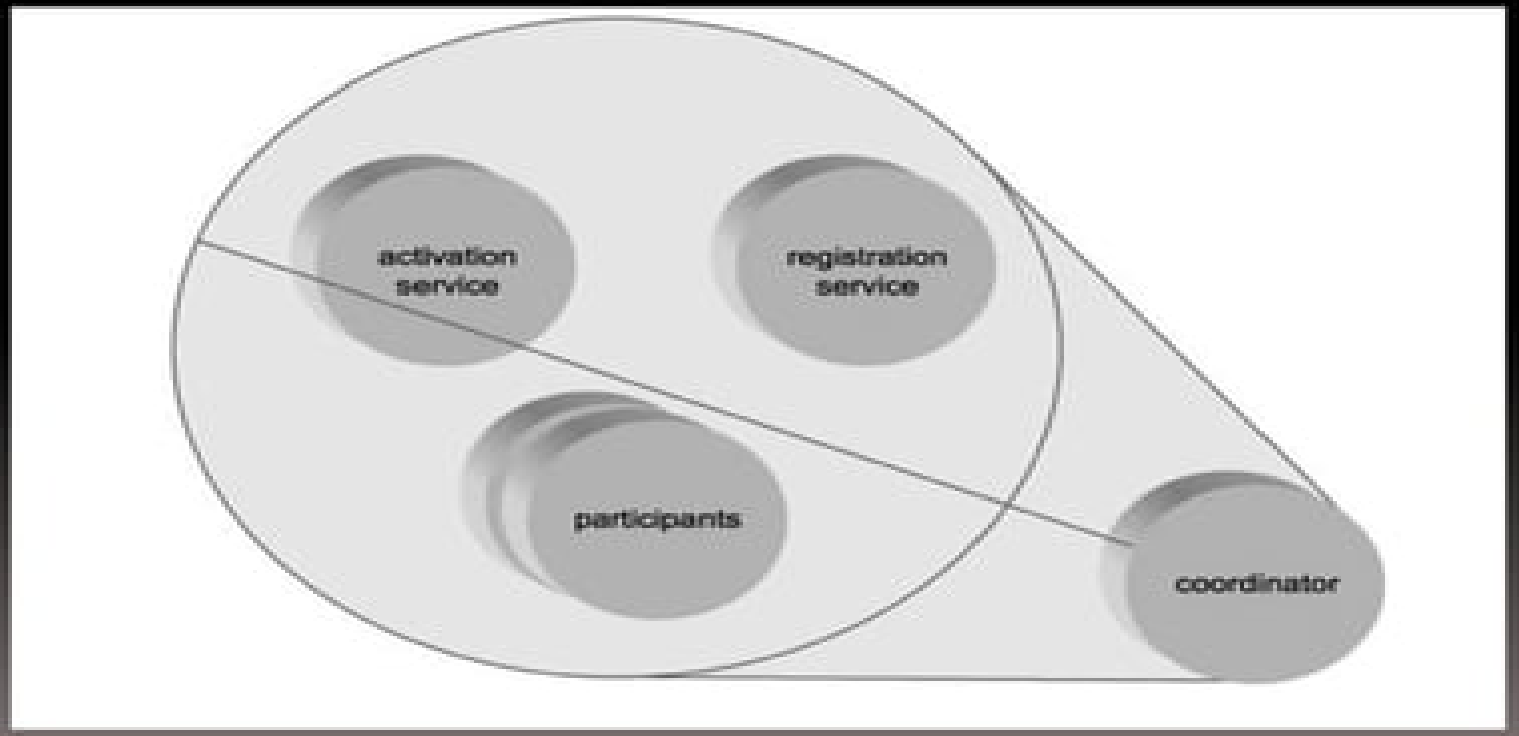
To coordinate the complex activity of car washing in an efficient manner, Chuck needs to communicate to Bob when he has located the bucket. Because Bob may not be immediately available and because Chuck doesn't want to go looking for Bob to hand over the bucket, Chuck tells me that the bucket is ready. Chuck then continues with his other work, and when I see Bob next, I can relay the status of the bucket availability to him.

In this scenario, the bucket availability status is considered context information that I managed. Because a separate context manager (me) was in place, Chuck was alleviated of the responsibility of remembering and communicating the context information to Bob. This freed Chuck to continue with his other work. It also spared Bob from having to directly locate and communicate with Chuck to get the context information. Finally, my knowledge of who was doing what in this car washing process also would be classified as context information.

Coordinator composition

- WS-Coordination establishes a framework that introduces a generic service based on the coordinator service model.
- This service controls a composition of three other services that each play a specific part in the management of context data.

The coordinator service composition



Coordinator composition

The coordinator composition consists of the following services:

1. Activation service:
   - Responsible for the creation of a new context and for associating this context to a particular activity.
2. Registration service:
   - Allows participating services to use context information received from the activation service to register for a supported context protocol.
3. Protocol-specific services:
   - These services represent the protocols supported by the coordinator's coordination type.
4. Coordinator :
   - The controller service of this composition, also known as the coordination service.

Coordinator composition

Coordination types and coordination protocols

- Each coordinator is based on a coordination type, which specifies the nature and underlying logic of an activity for which context information is being managed.
- Coordination types are specified in separate specifications.
- The WS-Coordination framework is extensible and can be utilized by different coordination types, including custom variations.
- The two coordination types most commonly associated with WS-Coordination are WS-AtomicTransaction and WS-BusinessActivity.
- Coordination type extensions provide a set of coordination protocols, which represent unique variations of coordination types and consist of a collection of specific behaviors and rules.
- A protocol is best viewed as a set of rules that are imposed on activities and which all registered participants must follow.

Coordinator composition

Coordination contexts and coordination participants

- A context created by the activation service is referred to as a coordination context.
- It contains a collection of information that represents the activity and various supplementary data.
- Examples of the type of data held within a coordination context include:
  - a unique identifier that represents the activity
  - an expiration value
  - coordination type information
- A service that wants to take part in an activity managed by WS-Coordination must request the coordination context from the activation service.
- It can use this context information to register for one or more coordination protocols.
- A service that has received a context and has completed registration is considered a participant in the coordinated activity.

Coordinator composition

The activation and registration process

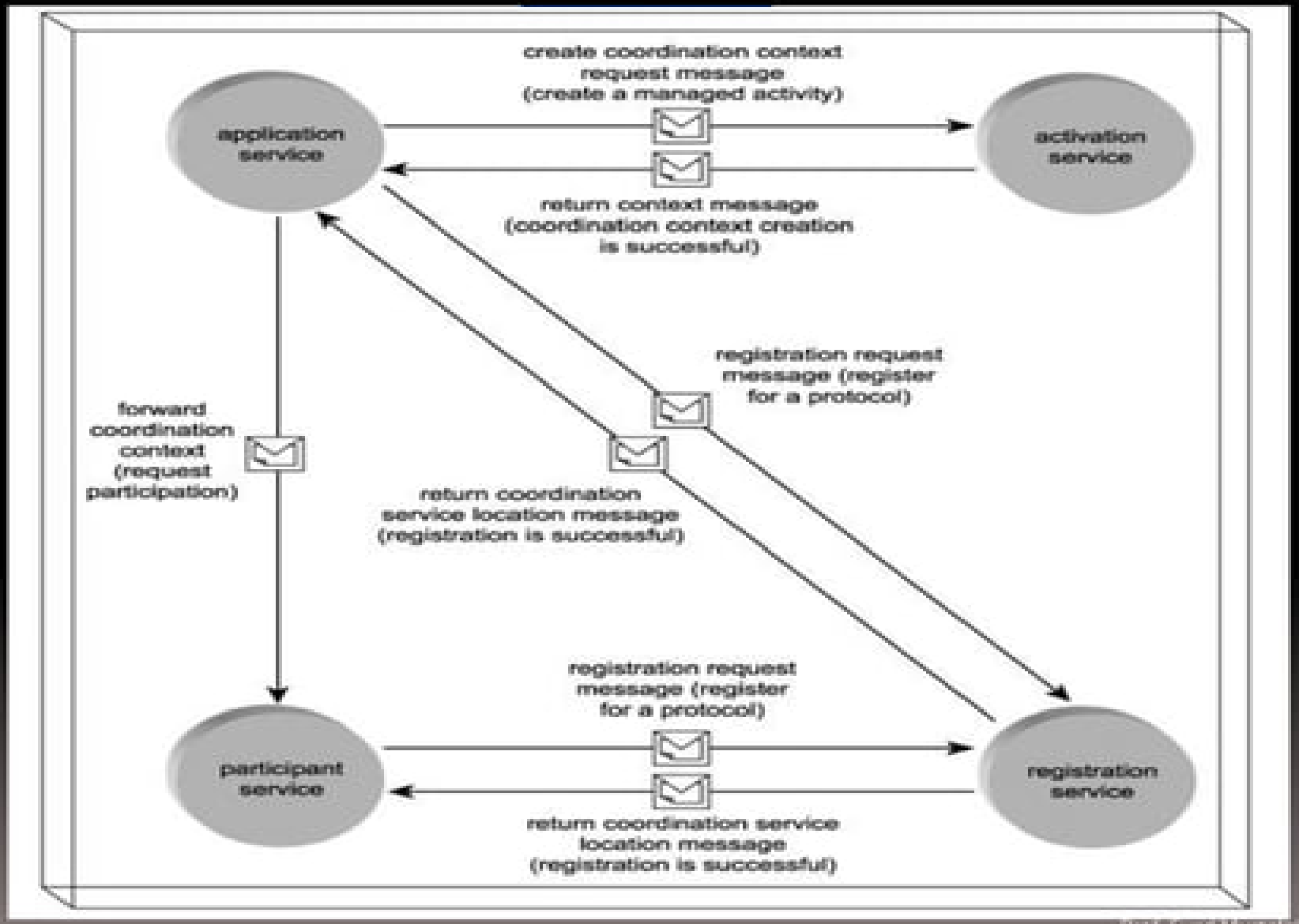
- The coordination service composition is instantiated when an application service contacts the activation service.
- Via a CreateCoordinationContext request message, it asks the activation service to generate a set of new context data.
- Once passed back with the ReturnContext message, the application service now can invite other services to participate in the coordination.
- This invitation consists of the context information the application service originally received from the activation service.

Coordinator composition

The activation and registration process

- Any Web service in possession of this context information may issue a registration request to the registration service.
- This allows the service to enlist in a coordination based on a specific protocol.
- Upon a successful registration, a service is officially a participant. The registration service passes the service the location of the coordinator service, with which all participants are required to interact.
- At this time, the coordination service is also sent the address of the new participant.

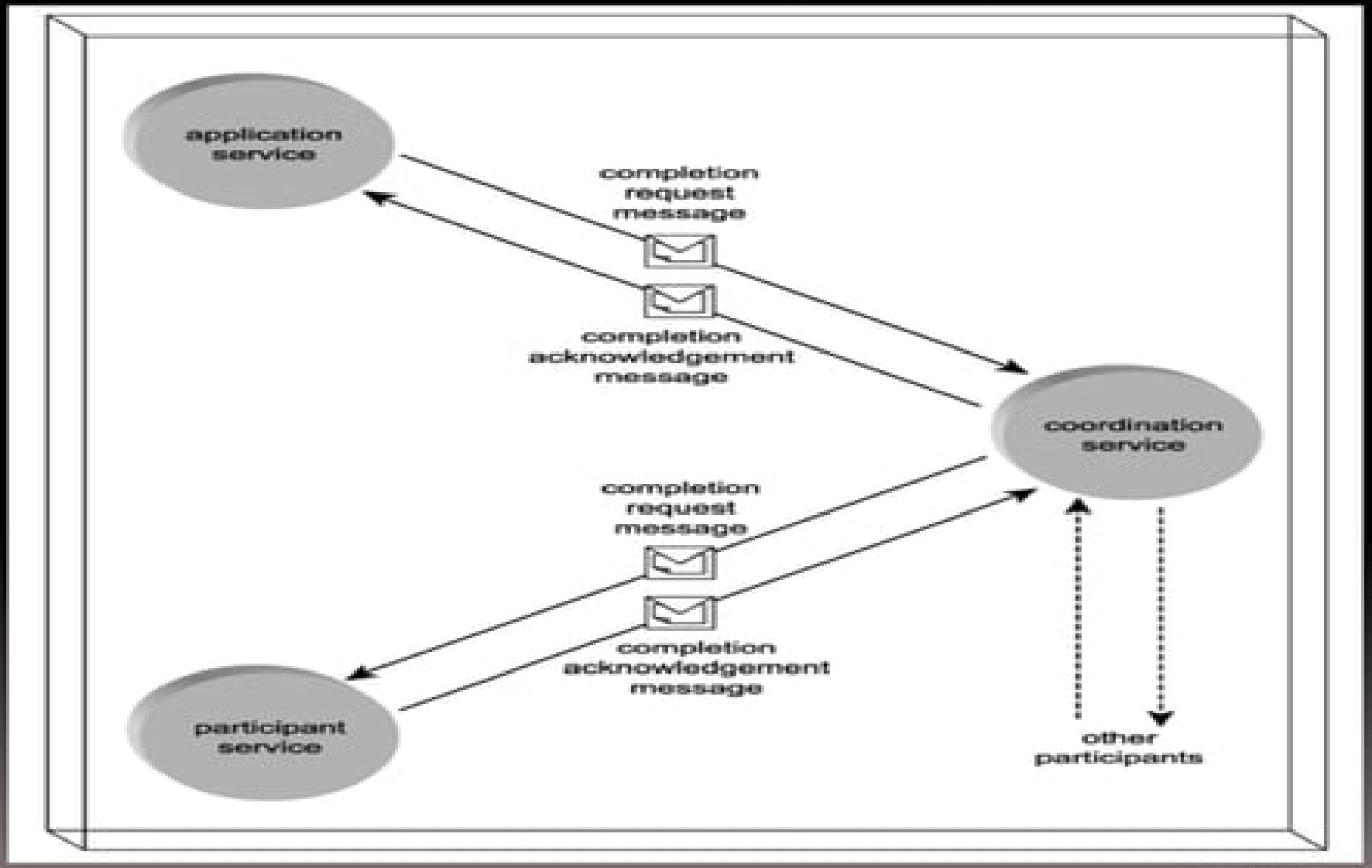
The WS-Coordination registration process



The completion process

- The application service can request that a coordination be completed by issuing a completion request message to the coordination service.
- The coordinator, in turn, then issues its own completion request messages to all coordination participants.
- Each participant service responds with a completion acknowledgement message

The WS-Coordination completion process



Coordination and SOA

- A coordinator-based context management framework, as provided by WS-Coordination and its supporting coordination types, introduces a layer of composition control to SOAs .
- It standardizes the management and interchange of context information within a variety of key business protocols.
- Coordination also alleviates the need for services to retain state. Statelessness is a key service-orientation principle applied to services for use within SOAs.
- Coordination reinforces this quality by assuming responsibility for the management of context information.

Coordination and SOA

Coordination as it relates to other parts of SOA.