Table of Contents **Page**

8 Patterns

8.1 Business Patterns

At the time of this writing, UN/CEFACT has not specified any business patterns that could be used in this workflow. There are efforts underway to discover reference material applicable for e-Business.

8.2 Requirements Patterns

Patterns to describe business collaborations will be developed by UN/CEFACT and added as they become available

8.3 Analysis Patterns

Business models may find it convenient to develop business transaction design patterns to facilitate the development of their specifications. The following six property-value conventions for business transactions have proven useful in the application of the metamodel to existing business requirements.

1. Business Transaction
2. Request / Confirm
3. Query / Response
4. Request / Response
5. Notification
6. Information Distribution

These conventions are applied by stereotyping the requesting business activity with the following syntax.

Transaction	Stereotype
Business Transaction	«BusinessTransactionActivity»
Request / Confirm	«RequestConfirmActivity»
Query / Response	«QueryResponseActivity»
Request / Response	«RequestResponseActivity»
Notification	«NotificationActivity»
Information Distribution	«InformationDistributionActivity»

The following table specifies the property-values for requesting business activities for each of the business transaction stereotypes.

	Time to Acknowledge Receipt	Time to Acknowledge Acceptance	Time to Perform	Authorization Required	Non-repudiation of Origin and Content	Non-repudiation of Receipt	Recurrence
Business Transaction	2hrs	6hr	24hr	true	true	true	3
Request / Confirm	null	Null	24hrs	false	false	true	3
Request / Response	null	Null	4hrs	false	false	null	3
Query / Response	null	Null	4hrs	false	false	null	3
Notification	24hrs	Null	24hrs	false	true	true	3
Information Distribution	24hrs	null	24hrs	false	false	false	3

The following table specifies the property-values for responding business activities for each of the business transaction stereotypes.

	Time to Acknowledge Receipt	Time to Acknowledge Acceptance	Time to Perform	Authorization Required	Non-repudiation of Origin and Content
Business Transaction	2hrs	6hr	24hr	true	true
Request / Confirm	2hrs	null	24hrs	true	false
Request / Response	null	null	4hrs	false	false
Query / Response	null	null	4hrs	false	false
Notification	24hrs	null	24hrs	false	false
Information Distribution	24hrs	null	24hrs	false	false

It is recommended that the following stereotype be used on the responding business activity when authorization is required for a responding activity to respond to a business document request.

Business Activity	Stereotype
Authorized Activity	«AuthorizedActivity»

Another convention that makes the application of these stereotypes easier is to only stereotype the requesting business activity when a symmetrical business relationship is designed. With this convention the time to perform, time to acknowledge receipt, time to acknowledge acceptance, non-repudiation and authorization requirements are assumed symmetrical and thus applicable equally to both the requesting and responding business activities.

8.3.1 Timeout Exceptions

A time-out parameter shall be specified whenever a requesting partner expects one or more responses to a business document request. A requesting partner shall not remain in an infinite wait state. There shall be a time-out parameter specified for each expected response. There are four possible responses and hence four potential time-out specifications:

Acknowledge Receipt. The time a responding role has to acknowledge receipt of a business document.

Non-Substantive Acknowledge Business Acceptance. The time a responding role has to non-substantively acknowledge business acceptance of a business document.

Substantive Acknowledge Business Acceptance. The time a responding role has to substantively acknowledge business acceptance of a business document.

Perform Transaction. The time a business transaction has to complete.

The time-out value for each of the time-out parameters is absolute i.e. not relative to each other. All timers start when the requesting business document is sent. The timer values shall comply with the well-formedness rules in the previous section.

If the retry count is not zero and a time-out condition is signaled for any of the expected responses then the original business document shall be resent from the initiating partner role. The original business document shall be sent even if responding acknowledgements have already been received.

If an initiating partner receives a response after a time-out condition is signaled and the original business document has already been resent then this shall be ignored. A responding partner that receives a business document from a retry shall terminate their responding transaction for the previous business document and the retry request shall be serviced.

Upon sending a business document retry, it SHALL be guaranteed that the sending party resends an identical business document, save for a timestamp. Otherwise, a receiving partner shall be capable of rolling back an incoming business document at any point in time through the acknowledgment interval, acceptance interval, and back-end processing interval.

When the time to perform an activity equals the time to acknowledge receipt or the time to acknowledge business acceptance then the highest priority time out exception shall be used when the originator provides a reason for revoking their original business document offer. The time to perform exception is lower priority than both the time to acknowledge receipt and the time to acknowledge business acceptance.

8.3.2 Business Protocol Exceptions

A business protocol exception terminates the business transaction. The following are business protocol exceptions.

1. Negative acknowledgement of receipt. The structure/schema of a message is invalid.
2. Negative acknowledgement of acceptance. The business rules are violated.
3. Performance exceptions. The requested business action cannot be performed.
4. Sequence exceptions. The order or type of a business document or business signal is incorrect.
5. Syntax exceptions. There is invalid punctuation, vocabulary or grammar in the business document or business signal.
6. Authorization exceptions. Roles are not authorized to participate in the business transaction.
7. Business process control exceptions. Business documents are not signed for non-repudiation.

A responding role that throws a business protocol exception signals the exception back to the requesting role and then terminates the business transaction. A requesting role that throws a business protocol exception terminates the transaction and then sends a notification revoking the offending business document request. The requesting role cannot send a business signal to the responding role.

8.3.3 Pattern Property Modification Rules

The following rules apply when modifying design pattern properties.

1. If the convention for a time property value is >0 then it cannot be changed to NA.
2. If the convention for a time property value is NA then it cannot be changed. This is because the change would add or remove a role interaction that is not allowed, as it will change the convention.
3. The non-repudiation values specified in the convention cannot be changed except that both Query/Response and Request/Confirm can be changed to non-repudiation required. Changing any of the other non-repudiation values would change the semantic meaning of the business transaction.
4. The Authorization Required property can only be changed to N. It cannot be changed to Y if it is already set to N according to the convention.

8.3.4 Requesting Business Activity

Preconditions and post-conditions should be specified when there are structure or content constraints that apply to the document when it is used in a particular business transaction. Preconditions are specified in the guard of a transition from the initial pseudo state to a requesting business activity. Post-conditions are specified in the guard of a transition from a requesting business activity to state vertex that is the state of the machine when the business activity is successfully performed.

8.3.5 Object Flow

Business documents flow states specify the business document flow between roles as they perform business activities. Each business document has a source and target business activity.

An object flow has a type that is a document envelope. An envelope contains one or more structured and unstructured business documents. A business document is signed if non-repudiation of origin and content is required. A detached signature is used to provide non-repudiation of origin and content of a business document as it pertains to the entire document. The signature shall be part of the business document for authorization as the content of the document is authorized.

Structured business documents contain information entities. Information entities contain other information entities. Containment is modeled using UML associations.

8.3.6 Business Transaction Modeling Patterns

The e-business Business Transaction Pattern metamodel provides a framework for constructing e-business collaboration model specifications. This section describes the modeling patterns that apply the metamodel to represent specific business

transactions.

8.3.6.1 Introduction

The business transaction patterns metamodel provides a language and grammar for constructing business collaboration models. Modeling patterns are applications of the metamodel to common business transaction representations. Representations capture common structure and semantics applicable to specific business transactions. The UML activity diagram notation is used to specify business transaction patterns.

Modeling patterns are reusable, generalized business process abstractions that can be applied to many business areas. A metamodel provides the syntax and grammar for expressing designs. Modeling patterns are subjective constructions that meet the requirements of specific business transactions.

8.3.6.2 Business Transaction State Semantics

A business transaction specifies the contract formation process between two business partners. A contract is used to legally bind parties to a clearly stated intention (promise, obligation) and responsibilities of each party. A contract usually outlines what each party can do in the event the intended actions are not carried out (promised services not rendered, services rendered but payment not issued). Prudent parties execute (sign) contracts prior to carrying out the intended actions, to limit their liability and to protect their interests.

There are many types of business contract formation processes. For example, an "OFFER-AND-ACCEPTANCE" contract is formed when a product order is "accepted" by a vendor. An "accepted" (signed, mutually agreed upon) purchase order forms a contract between buyer and seller to provide a quantity of product at an agreed-upon price. After the contract is formed, the seller provides the product and the buyer pays for the product. In the event something goes wrong, the buyer and seller both have recourse as described in the contract.

Another example of contract formation occurs when a claim has been accepted for payment; this is a "contract" to perform the issuance of monetary payment (or another form of credit) some time after the "acceptance" (contract formation) of the claim.
All business transactions are treated as contract forming processes in that there is always an obligation (perhaps not residual) between each of the parties participating in the transaction.

The UML activity diagram notation is used to graphically specify these business transactions as design patterns. The pattern for specifying and interpreting these diagrams and the textual notation used to specify element names as well as conditional expressions is provided in this section.

Figure 29 illustrates a business transaction specification that does not include a responding business document and Figure 30 illustrates a business transaction specification that includes a responding business document.

Error! Reference source not found. illustrates a business transaction specification that does not include a responding business document and Figure 8-2 illustrates a business transaction specification that includes a responding business document.

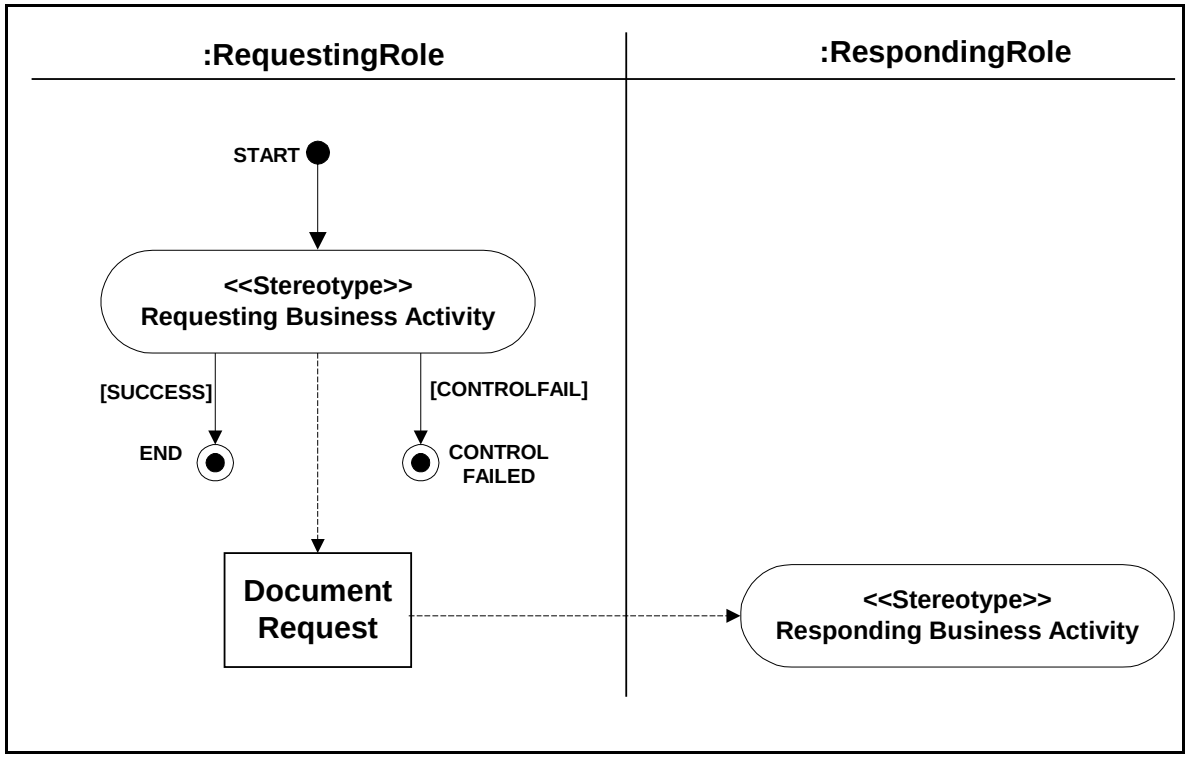

Figure 8-1 Business Transaction without Responding Business Document

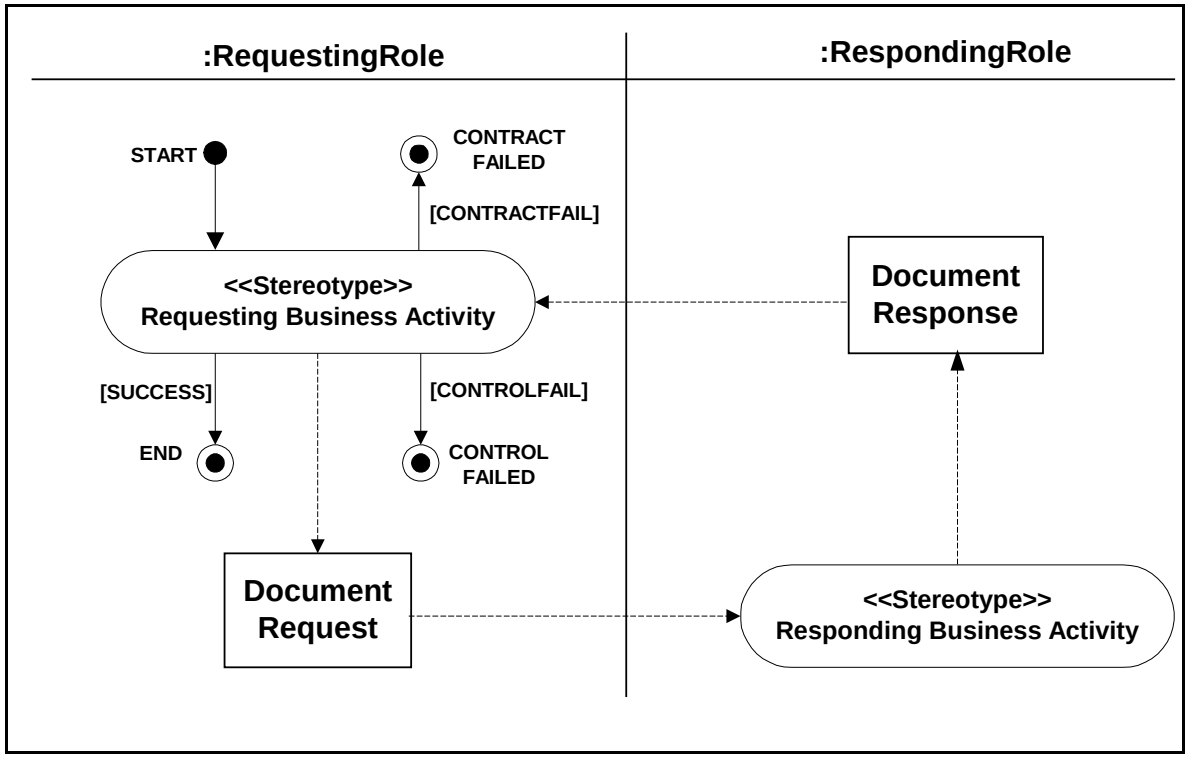

Figure 8-2 Business Transaction with Responding Business Document

It is recommended that all business transaction specifications use the layout illustrated in these figures. This will provide a consistent method of communicating business transactions.

The initial (START) state and the final (END, CONTROLFAILED, CONTRACTFAILED) state represent the state of a business transaction and not the state of any role that participates in the transaction. It is "by convention" that the initial and final states are placed into the requestor's swim lane. This has no semantic meaning with respect to any participating role. These states could be anywhere in the activity graph as they still pertain to the entire transaction and not to any particular role. The start state and final state conditions should therefore specify conditions that shall hold before the business transaction can transition into the "default" state (a UML definition).

START State Semantics

The condition that shall hold before transitioning into the initiating transaction activity should test the following [note that a Trading Partner Agreement (TPA)contains the transaction specifications agreed to by participating partners]:

1. The ability of each employee/organization to fulfill their obligations with respect to a TPA e.g.
    a. Are the roles approved trading partners i.e. does a TPA exist that governs the terms and conditions of the transaction?
    b. Do each of the participating roles meet the criterion required for performing the activity e.g. is the employee/organization performing the role authorized to perform the role if authorization is required?
    c. Is a business document non-repudiated if required in a TPA?
    d. Are all data entities tamper-proof, confidential and authenticated as

required in a TPA?

2. If a business record exists and it is also syntactically and structurally formatted with respect to the agreed message guideline in a TPA.

Start State Notation

Note that the START conditions are actually guard conditions on the transition from the initial state to the initiating activity in the activity graph. There is no pseudo state "condition" in the UML metamodel. These conditions are not, however, specified as guards in the transaction diagram to improve readability.

It is preferred that these conditions are captured using the following syntax. This improves consistency and will facilitate the translation of these conditions to OCL at a later stage.

States conditions are named in the form <Noun><Property>(<Verb>or<Code>)

- The <Noun> can be a Business Data Entity and the property is named "Status" in the form BDE Status <Code >. Purchase Order Status Open
- The <Noun> can be a Business Document with no named property in the form <Noun> <verb>. Purchase Order Exists
- The <Property> can be the name of a business process support system with no <Noun> in the form <Property><Verb>. Buyer Authorized.

Use the following notation to specify the START conditions:

- TPA Exists
- Requesting Partner Approved
- Responding Partner Approved
- <Business Document> Status <Code> etc. The values for this can be found in the business dictionary (just search for *StatusCode in the Entity Instances table). Use only valid status from the dictionary or add another valid status to the dictionary e.g. Purchase Order Status Revoked
- <Requesting Role> Authorized e.g. Buyer Authorized
- <Business Document> Exists e.g. Purchase Order Exists
- <Business Document> Non-Repudiated
- <Business Document> Valid
- <Business Document> <Property> Tamper-Proof
- <Business Document> <Property> Confidential
- <Business Document> <Property> Authenticated

END, CONTROLFAILED and CONTRACTFAILED State Semantics

The state of the business transaction transitions into the END state if both parties in a business transaction meet the conditions agreed to in their TPA. There are two final states specified for business transactions:

1. *Contract Failure*. The state machine shall transition into the CONTRACTFAILED state if the intended business contract is not formed but none of the control conditions are violated. For example, a responding role may return a negative business acceptance document that contains a status BDE whose value is “Reject”. In these cases a test on the BDE status for reject shall transition the state machine into the CONTRACTFAILED state. The contract failure end state shall only be used for business transactions that permit negative acknowledgements. In these instances the business transaction activity graph is shown in Figure 8-2. If there is no contract failure condition then the transaction activity graph is shown in Figure 8-3.

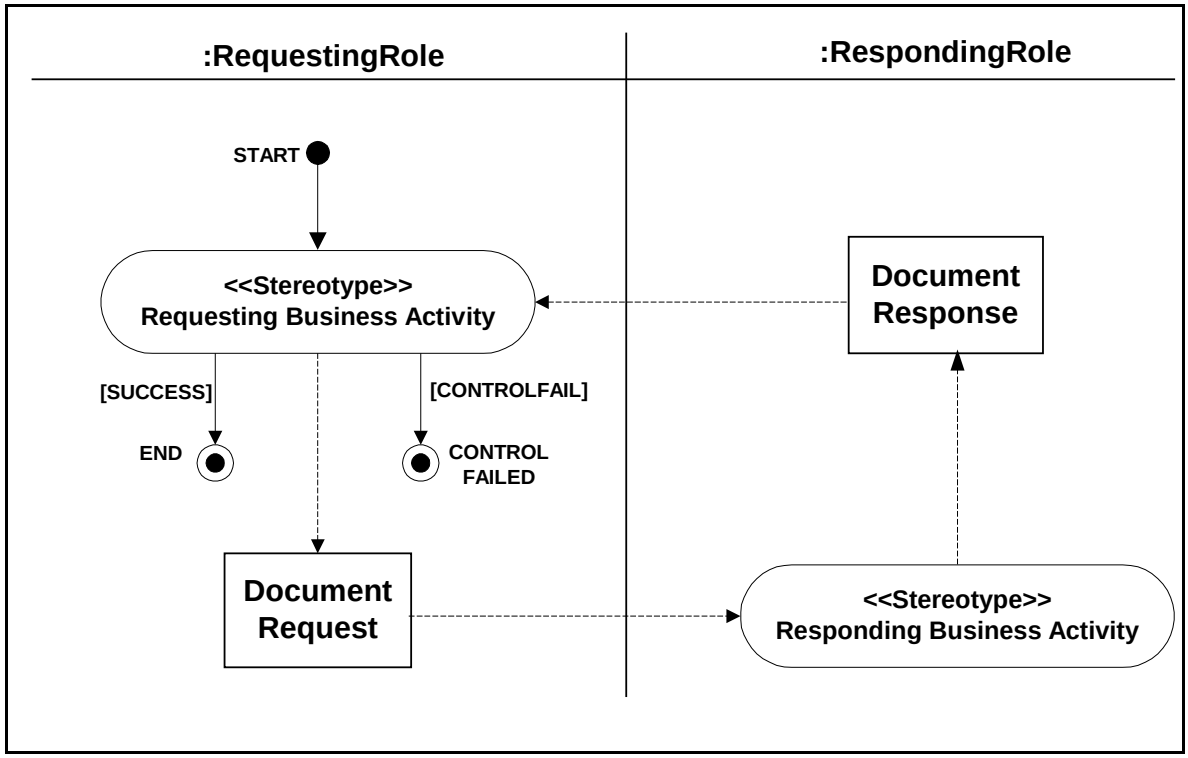

Figure 8-3 Business Transaction with no Contract Failure State

2. *Control Failure*. The activity shall transition into the CONTROLFAILED state if any business collaboration control parameter is violated. For example, timeouts, processing exceptions, non-repudiation and authorization exceptions. In these cases both the transaction fails and the contract is not formed.

The conditions that shall hold before transitioning into the SUCCESS state should test the following [note that a TPA contains the transaction specifications agreed to by participating partners]:

1. Each employee/organization has fulfilled their obligations with respect to a trading partner agreement (TPA) e.g.
   a. Have each of the participating roles met the criterion required for performing the activity e.g. were the employee/organization performing the roles authorized to perform the role if authorization is required?
   b. Is a business document non-repudiated if required in a TPA?
   c. Are all data entities in the responding document tamper-proof, confidential and authenticated as required in a TPA?
   d. Were all documents and business signals received by both parties as agreed to in the TPA.
2. If a business record exists and it is also syntactically and structurally formatted with respect to the agreed message guideline specified in a TPA.
3. The retry count has not exceeded the maximum specified.
4. The state machine transitions to the CONTRACTFAILED state if the conditions to transition to the END state are not met and/or a condition on a negative response is satisfied. It is redundant to re-specify the negation of all of the SUCCESS conditions in the FAILED state conditions. Therefore, the following are the only conditions necessary for the CONTRACTFAILED conditions.

- SUCCESS and (<Business Data Element> Status <Code > and/or ….)

5. The state machine transitions to the CONTROLFAILED state if the conditions to transition to the END state and CONTRACTFAILED states are not met. It is redundant to re-specify the negation of all of the SUCCESS and CONTRACTFAILED conditions in the CONTROLFAILED state conditions. Therefore, the following are the only conditions necessary for the CONTROLFAILED conditions.
    - Not SUCCESS or Not CONTRACTFAIL

EDITORS NOTE: 4 AND 5 REQUIRE FURTHER EXPLANATION

END State Notation

Note that the END conditions are actually guard conditions on the transition from the end status in the activity graph. There is no pseudo state "condition" in the UML metamodel. These conditions are not, however, enumerated as guards in the transaction diagram to improve readability. It is preferred that these conditions are captured using the following syntax. This improves consistency and will facilitate the translation of these conditions to OCL at a later stage.

States conditions are named in the form <Noun><Property>(<Verb>|<Code>)

- The <Noun> can be a Business Data Entity and the property is named "Status" in the form BDE Status <Code>. Purchase Order Status Open
- The <Noun> can be a Business Document with no named property in the form <Noun> <verb>. Purchase Order Acceptance Exists
- The <Property> can be the name of a business process support system with no <Noun> in the form <Property><Verb>. Seller Authorized, Receipt Non-Repudiated.

Use the following notation to specify the END conditions:

- <Business Document> Status <Code> etc. The values for this can be found in the business dictionary (just search for *StatusCode in the Entity Instances table). Make sure you only use valid status from the dictionary or add another valid status to the dictionary e.g. Purchase Order Acceptance Status Approved
- <Responding Role> Authorized e.g. Seller Authorized
- <Business Document> Exists e.g. Purchase Order Acceptance Exists.
- <Business Signal> Exists e.g. Verification of Receipt Exists
- <Business Document> Non-Repudiated
- Verification of Receipt Non-Repudiated
- <Business Document> Valid
- <Business Signal> Valid
- <Business Document> <Property> Tamper-Proof
- <Business Document> <Property> Confidential
- <Business Document> <Property> Authenticated

8.3.6.3 Business Transaction Pattern Rationale

This section provides the design rationale for the time-out specification in each business transaction pattern. This pattern rational is presented within a document-processing framework that comprises the following steps.

1. Grammar validation. The task of verifying that the grammar of a message is valid (usually only the header of the message at this step).

2. Sequence validation. The task of verifying that the collaboration control information is valid with respect to the business transaction specification.
3. Schema validation. The task of verifying that the message schema is valid with respect to a message guideline agreed to by both partners. It is recommended that message receipt be acknowledged after this validation step to ensure that documents are “readable” as well as “accessible”.
4. Content validation. The task of verifying that the content of a message is valid with respect to any business rules that govern the formation of a contract. It is recommended that business acceptance be acknowledged after this validation step.
5. Activity processing. The task of processing the request in the initiating business document.

Figure 8-4 illustrates the processing of an initiating message when the contract-closing (contract acceptance document) message is an acknowledgement of receipt. The acknowledgement of receipt is a business signal i.e. it does not map onto a business document.

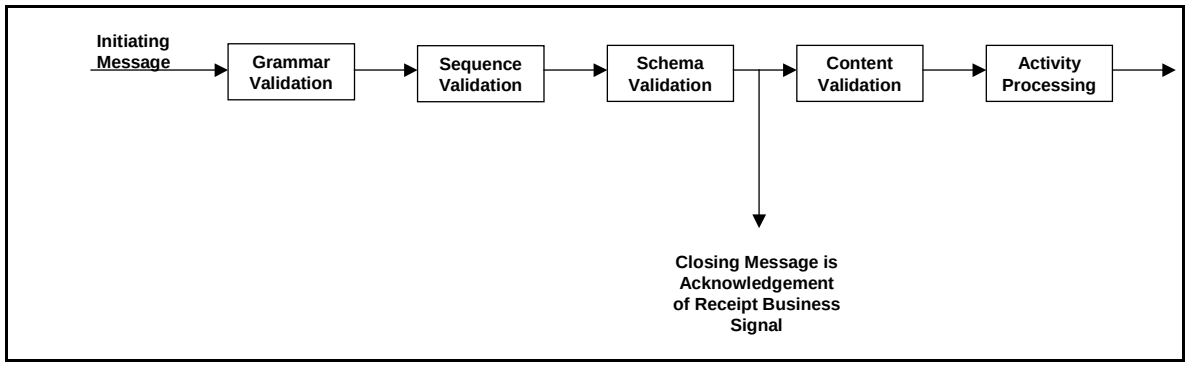

Figure 8-4 Acknowledgement of Receipt Closing Message

The following table shows example timeout parameters for this business transaction. The Information Distribution and Notification business activity specification (see Modeling Metamodel) use this design pattern.

Role Name	Activity Name	Time to Acknowledge Receipt	Time to Acknowledge Acceptance	Time to Perform
Role	Activity	24hr	N/A	24hr

Figure 8-5 illustrates the processing of an initiating message when the closing message is an acknowledgement of acceptance. The acceptance message can be either substantive or non-substantive. A substantive business acceptance message includes business data from the initiating message e.g. product, price and quantity in a substantive purchase order acceptance document. A substantive business acceptance

message contains a business document. A positive non-substantive business acceptance message contains the initiating business document identification data. A negative non-substantive business acceptance message contains the initiating business document identification data, the reason for rejection and syntactic error messages indicating the business data elements in which the error was found. A positive non-substantive acceptance message is a business signal i.e. it does not map onto a business document. Note the following:

1. If a substantive business acceptance is required then a responding **business document** is specified in a business transaction.
2. If a non-substantive business acceptance is required then a responding business document is **not** specified in a business transaction.

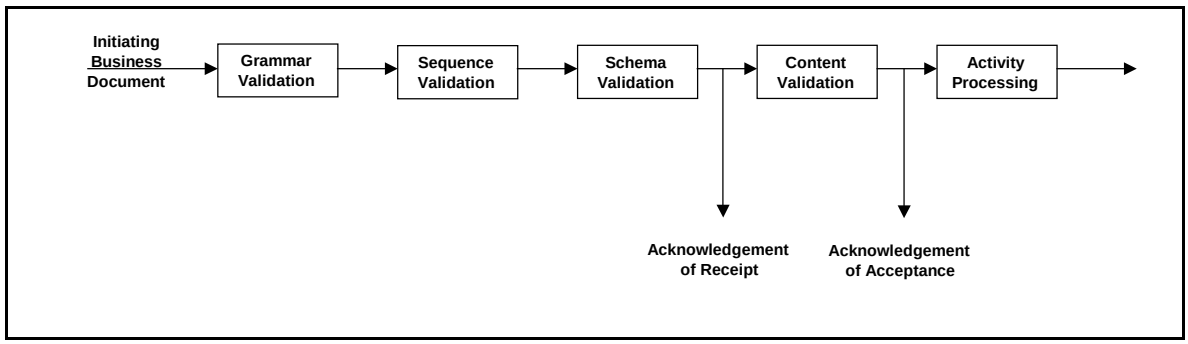

Figure 8-5 Acknowledgement of Business Acceptance Closing Message

The following table shows example timeout parameters for this business agreement. The Business Transaction Activities use this design pattern when a substantive business acknowledgement of acceptance is required.

Role Name	Activity Name	Time to Acknowledge Receipt	Time to Acknowledge Acceptance	Time to Perform
Role	Activity	2hr	6hr	6hr

Figure 8-6 illustrates the processing of an initiating message when the closing message is a responding business document. The Query/Response business activity specification uses this design pattern.

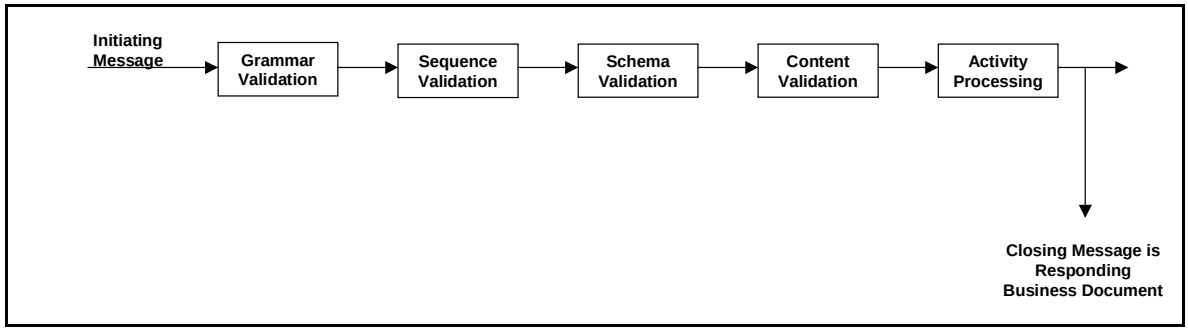

Figure 8-6 Responding Business Document is Closing Message

The following table shows example timeout parameters for this business transaction.

Role Name	Activity Name	Time to Acknowledge Receipt	Time to Acknowledge Acceptance	Time to Perform
Role	Activity	N/A	N/A	24hr

It is possible to specify acknowledgements and a responding business document as part of the business agreement. Figure 8-7 illustrates the processing of an initiating message when there is a requirement for an acknowledgement of receipt, a non-substantive acknowledgment of acceptance and a responding document. Note that the acceptance message cannot be specified as substantive i.e. a business document. It can only be a non-substantive i.e. a business signal. If the acceptance shall be substantive then two business transactions are required.

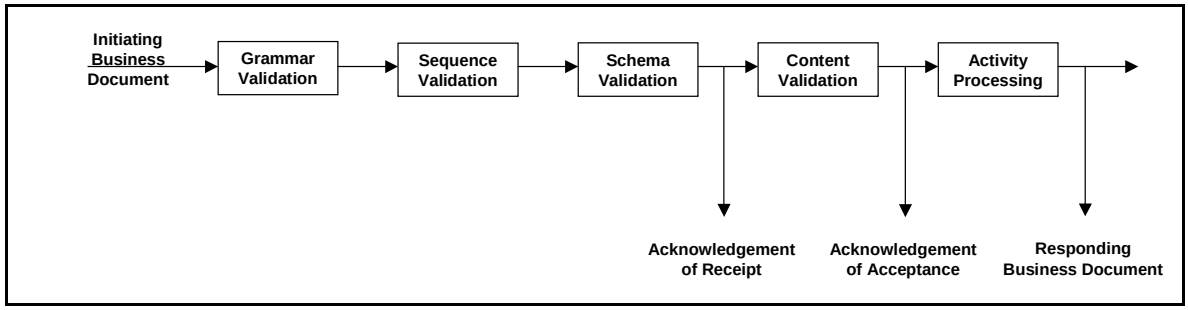

Figure 8-7 Receipt, Business Acceptance and Business Document Response

The following table shows example timeout parameters for this business transaction. The Business Transaction business activity specification that mandates the return of a non-substantive business acceptance acknowledgement uses this design pattern.

Business Activity Performance Controls				
Role Name	**Activity Name**	**Time to Acknowledge Receipt**	**Time to Acknowledge Acceptance**	**Time to Perform**
Role	Activity	2hr	6hr	24hr

Interpreting how the contact is closed using a substantive or non-substantive acknowledgement of acceptance is a based on three cues.

1. There is a value for “Time to Acknowledge Acceptance”.
2. The value for “Time to Perform” is either equal or not equal to the “Time to Acknowledge Acceptance”.
3. There either is or is not a business document response.

Case 1:

If

1. There is a value for “Time to Acknowledge Acceptance”.
2. The value for “Time to Perform” equals the “Time to Acknowledge Acceptance”.
3. There is no business document response.

Then the acknowledgement of acceptance is non-substantive.

Case 2:

If

1. There is a value for “Time to Acknowledge Acceptance”.
2. The value for “Time to Perform” equals the “Time to Acknowledge Acceptance”.
3. There is a business document response with the verb acceptance appended to a noun e.g. Purchase Order Acceptance.

Then the acknowledgement of acceptance is substantive.

Case 3:

If

1. There is a value for “Time to Acknowledge Acceptance”.
2. The value for “Time to Perform” does not equal the “Time to Acknowledge Acceptance”.
3. There is a business document response.

Then the acknowledgement of acceptance is non-substantive.

8.3.6.4 Business Transaction Pattern [Contract formation, e.g., place order]

The business transaction design pattern is illustrated in Figure 8-8. This design pattern is best used to model the “*offer and acceptance*” business transaction process that results in a residual obligation between both parties to fulfill the terms of the contract. The following principals and definitions of offer and acceptance are taken from the following URL:
http://www.anu.edu.au/law/pub/edinst/anu/contract/lectures/moles/semest1/MContractFormationOfferAnd.html#MContrac-Whatisanoffer.

Offer and acceptance are a means of analyzing the process of

negotiation to decide whether and when a contract has been made and what therefore constitute its terms.
There is no satisfactory definition of an offer beyond identifying it by reference to the fact that it can be converted into a contract by an act of acceptance. Whether it can be accepted depends upon the objective intention of the party making the statement which is alleged to be an offer.

Making an offer exposes one to the imposition of legal liability by another. In deciding whether statements amount to an offer, the courts are said to use an objective test. Therefore under the objective test an apparent intention to be bound will suffice if 2 conditions are satisfied:

- *conduct of the alleged offeror shall be such as to induce a "reasonable person" to believe that he/she is making the alleged offer.*

- *the alleged offeree shall actually hold that belief - ie believe that the offeror is making a genuine offer, as opposed, for example, to playing a game.*

The pattern specifies an originating business activity sending a business document to a responding business activity that may return a business signal or business document as the last responding message. The pattern mandates the acknowledgement of the requesting business document when it passes a “Business Acceptance” test, i.e. passes the content validation step as illustrated in Figure 36. This acknowledgement can be substantive i.e. contains the terms of acceptance of a contract or it may be non-substantive i.e. a general auditable business signal. The intent of this business transaction pattern is to model the formation of an offer and acceptance business contract[1]. If the requesting role transitions from their business activity into the control failure state then the role shall initiate a notification of failure (see notification design pattern) business transaction to revoke their original offer.
Note that the “CONTRACTFAILED” final state can be omitted from the business transaction specification if there are no negative business acceptance documents specified.

[1] Refer to the following documents to understand on-line business contract formation.

- PART 2 UNIFORM RULES OF CONDUCT FOR INTERCHANGE OF TRADE DATA BY TELETRANSMISSION (UNCID), CHAPTER 2 - Text of the Uniform Rules of Conduct, http://www.unece.org/trade/untdid/texts/d220_d.htm
- UN/ECE RECOMMENDATION No.26, THE COMMERCIAL USE OF INTERCHANGE AGREEMENTS FOR ELECTRONIC DATA INTERCHANGE, http://www.unece.org/trade/untdid/texts/d240_d.htm
- The Commercial use of Electronic Data Interchange, Section of Business Law American Bar Association, A report and model trading partner agreement, http://www.abanet.org/buslaw/catalog/5070258.html

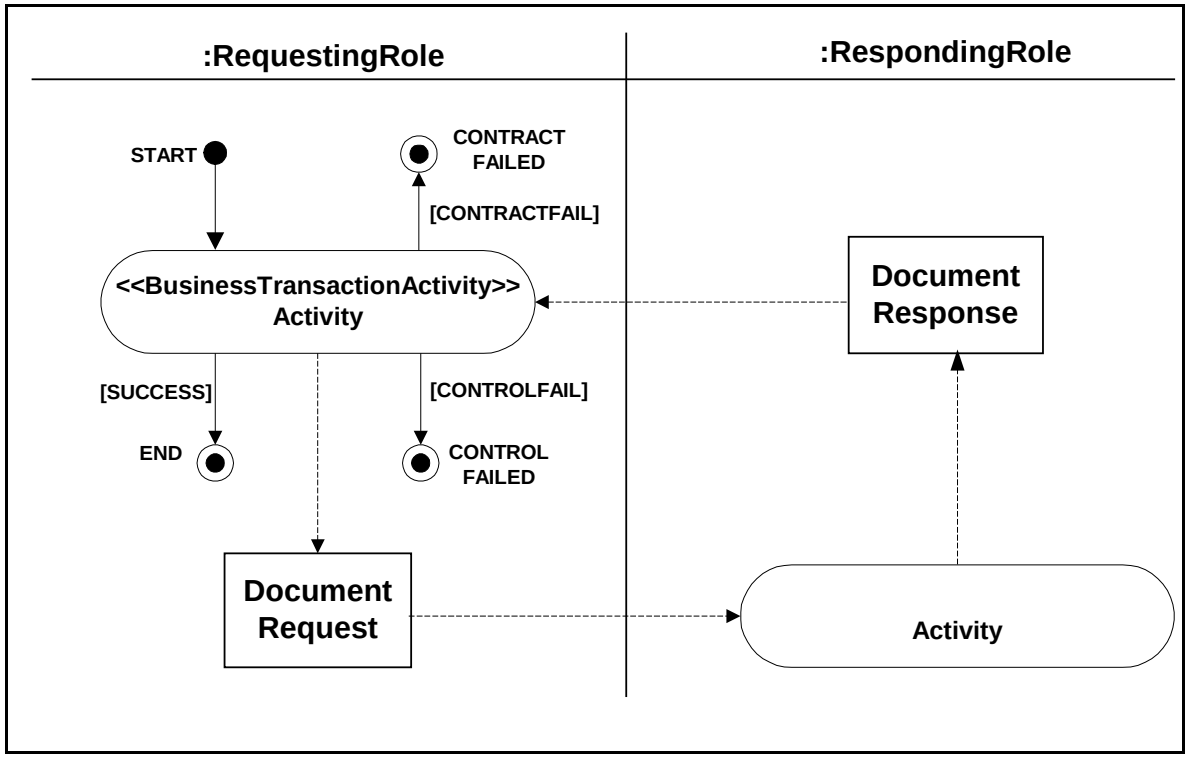

Figure 8-8 Business Transaction Activity Design Pattern

8.3.6.5 Query/Response Pattern [Static information, e.g., obtain catalog]

Figure 8-9 illustrates the query/response design pattern. The query/response design pattern specifies one business document as output and one business document as input. These documents adhere to the query/response business document design pattern specified in the previous section. Query/Response does not permit the return of auditable business signals i.e. receipt acknowledgement or business acceptance acknowledgement.
The responding activity is most likely to be serviced by an organizational role i.e. not by an employee role. There is no non-repudiation requirement for these activities.

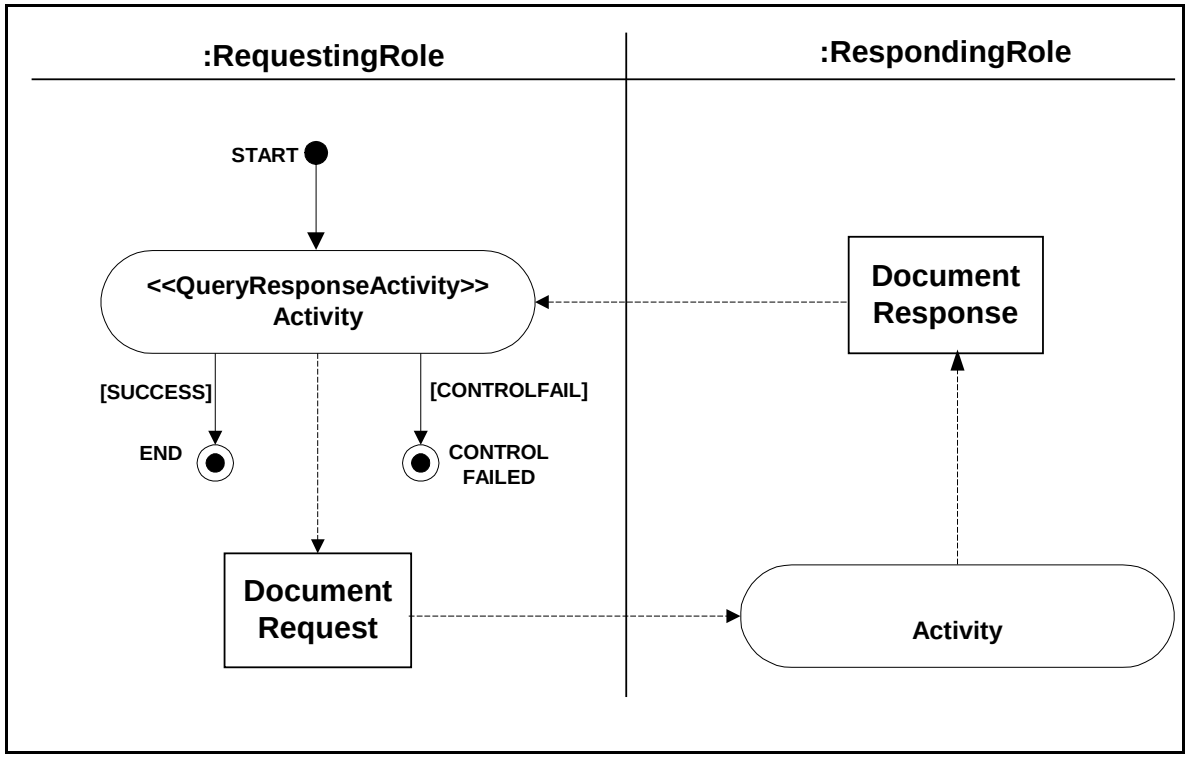

Figure 8-9 Query/Response Activity Design Pattern

The query/response design pattern specifies a query for information that a responding partner already has e.g. against a fixed data set that resides in a database. The response comprises zero or more results each of which meet the constraining criterion in the query. For example, a query for the products under $500 will yield any number of product results in the same response all of which have a price under $500. This pattern should be used when the response comprises a collection of results each of which meet the constraining criterion specified in the query. The Request/Response design pattern should be used instead.

8.3.6.6 Request/Response Pattern [Dynamic information, e.g., obtain Buyer ID, obtain quote]

Figure 8-10 illustrates the request/response design pattern. Note that there is usually no residual obligation between both parties to fulfil the terms of a contract as in the Business Transaction Activity pattern. For example, a request for price and availability does not result in the responding party allocating product for future purchase and it does not result in the requesting party being obligated to purchase the products. This pattern specifies the exchange of a requesting and responding business document. Acknowledgement of business acceptance is not permitted – use the “Business Transaction Activity” stereotype if this is required.
The responding activity is most likely serviced by organizational or employee roles. Non-repudiation is an optional requirement for these activities.

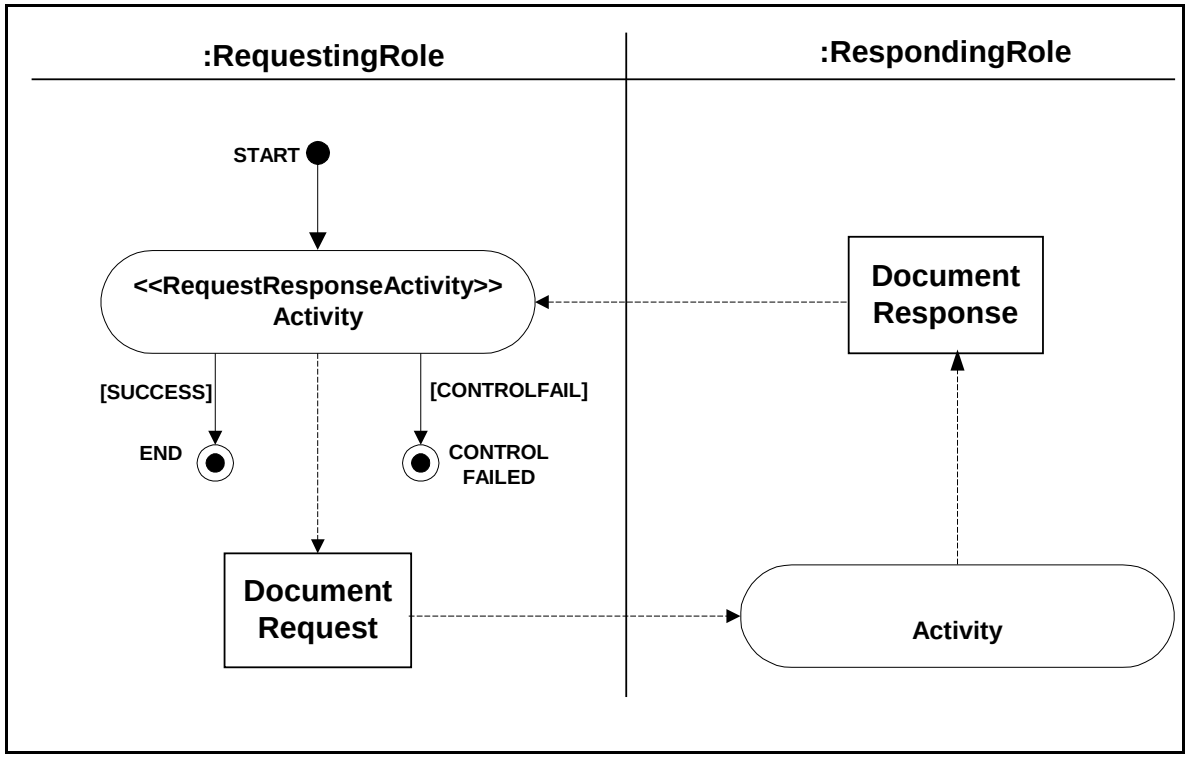

Figure 8-10 Request/Response Activity Design Pattern

The request/response activity pattern shall be used for business contracts when an initiating partner requests information that a responding partner already has and when the request for business information requires a complex interdependent set of results. For example, a price and availability request may constrain the response such that the sum of all products returned in each of the results (one response may comprise zero or more results) shall be less than 100. This response requires some business processing on a query before a response is returned to the requestor. This flow pattern is used in conjunction with the Request/Response business document design pattern that includes syntax for expressing Business Constraints that apply to the collection of results in the response. If there is no “aggregate” or “interdependent” constraints that shall be applied to a set of results then the query/response pattern shall be used.

8.3.6.7 Request/Confirm Pattern [Status information, e.g., Obtain order status]

Figure 8-11 illustrates the request/confirm design pattern. Note that there is usually no residual obligation between both parties to fulfill the terms of a contract as in the Business Transaction Activity pattern. For example, a request for authorization to sell certain products expects a confirmation response to the request that confirms if the requestor is authorized or not authorized to sell the products. This pattern specifies the exchange of a requesting and responding business document. If acknowledgement of receipt is expected it is the initiator’s obligation to follow up on the request until an acknowledgement of receipt is received. Acknowledgement of business acceptance is not permitted – use the “Business Transaction Activity” stereotype if this is required.
The responding activity is most likely serviced by organizational or employee roles. Non-repudiation is an optional requirement for these activities.

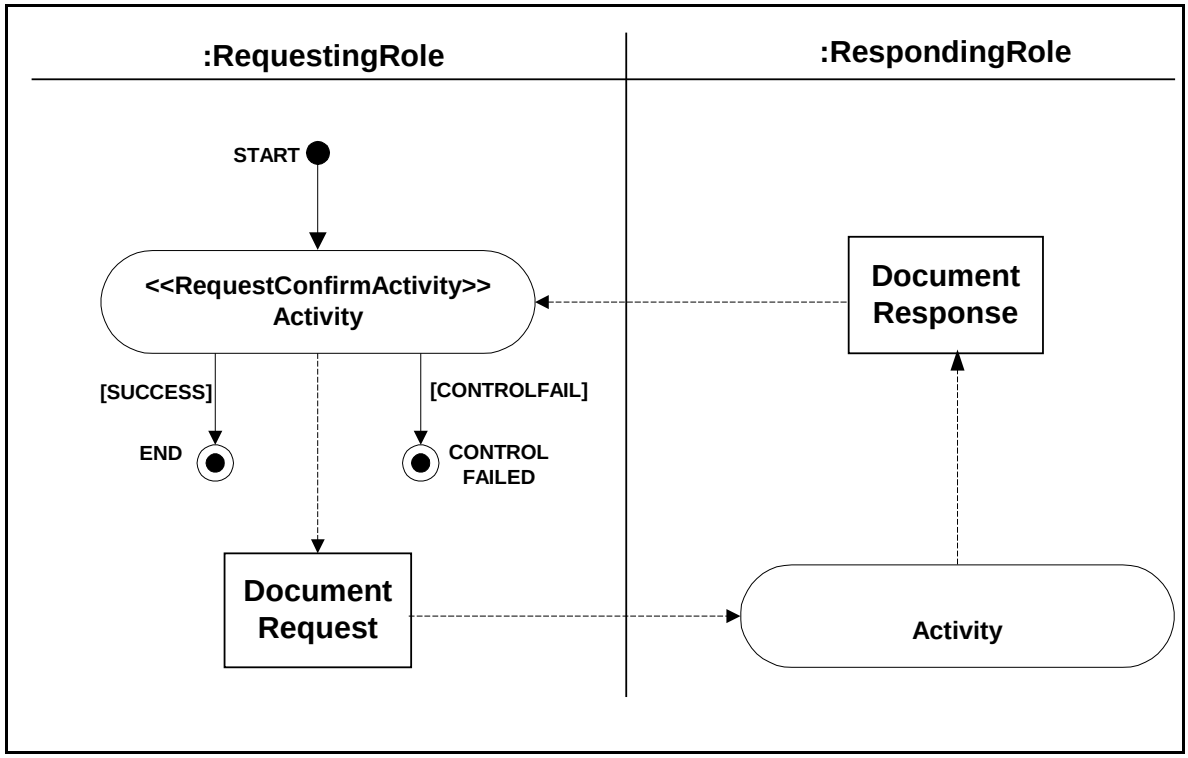

Figure 8-11 Request/Confirm Activity Design Pattern

The request/confirm activity pattern shall be used for business contracts where an initiating partner requests confirmation about their status with respect to previously established contracts or with respect to a responding partner's business rules.

8.3.6.8 Information Distribution Pattern

Figure 8-12 illustrates the information distribution design pattern. This pattern specifies the exchange of a requesting business document and the return of an acknowledgement of receipt business signal. This pattern is used to model an *informal* information exchange business transaction that therefore has no non-repudiation requirements.

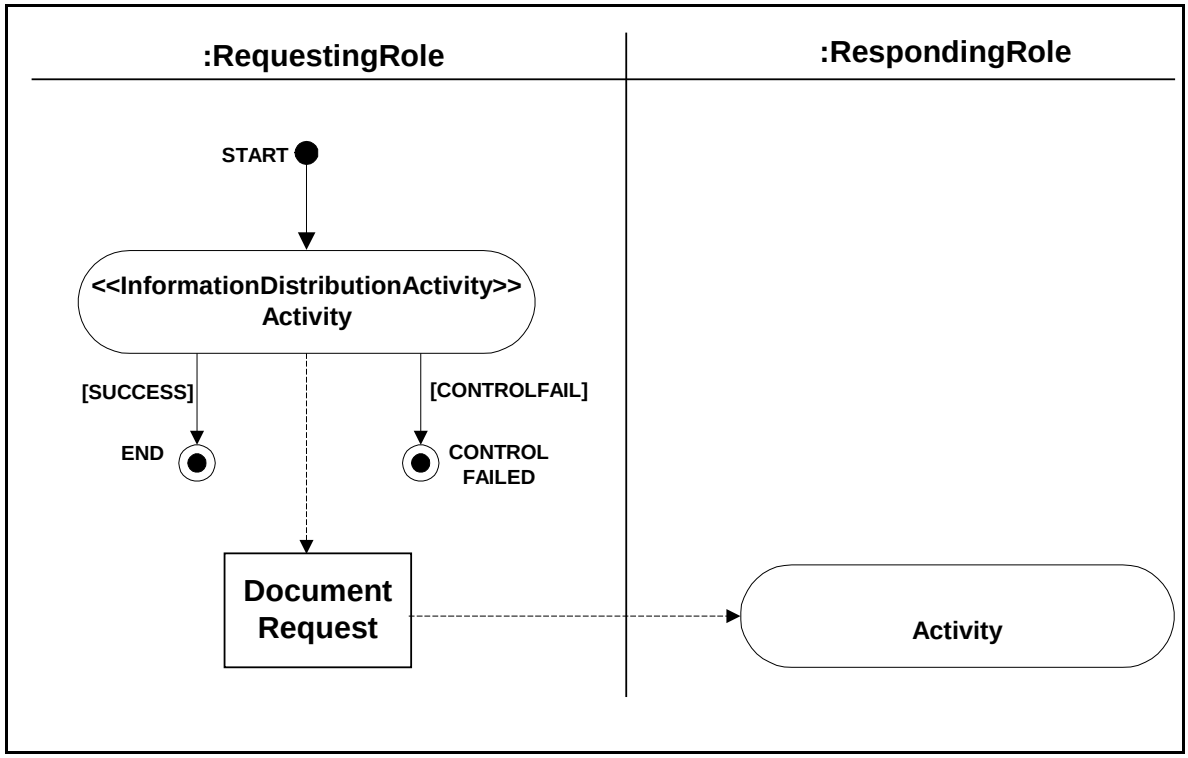

Figure 8-12 Information Distribution Design Pattern

8.3.6.9 Notification Pattern

Figure 8-13 illustrates the notification design pattern. This pattern specifies the exchange of a notifying business document and the return of an acknowledgement of receipt business signal. This pattern is used to model a *formal* information exchange business transaction that therefore has non-repudiation requirements.

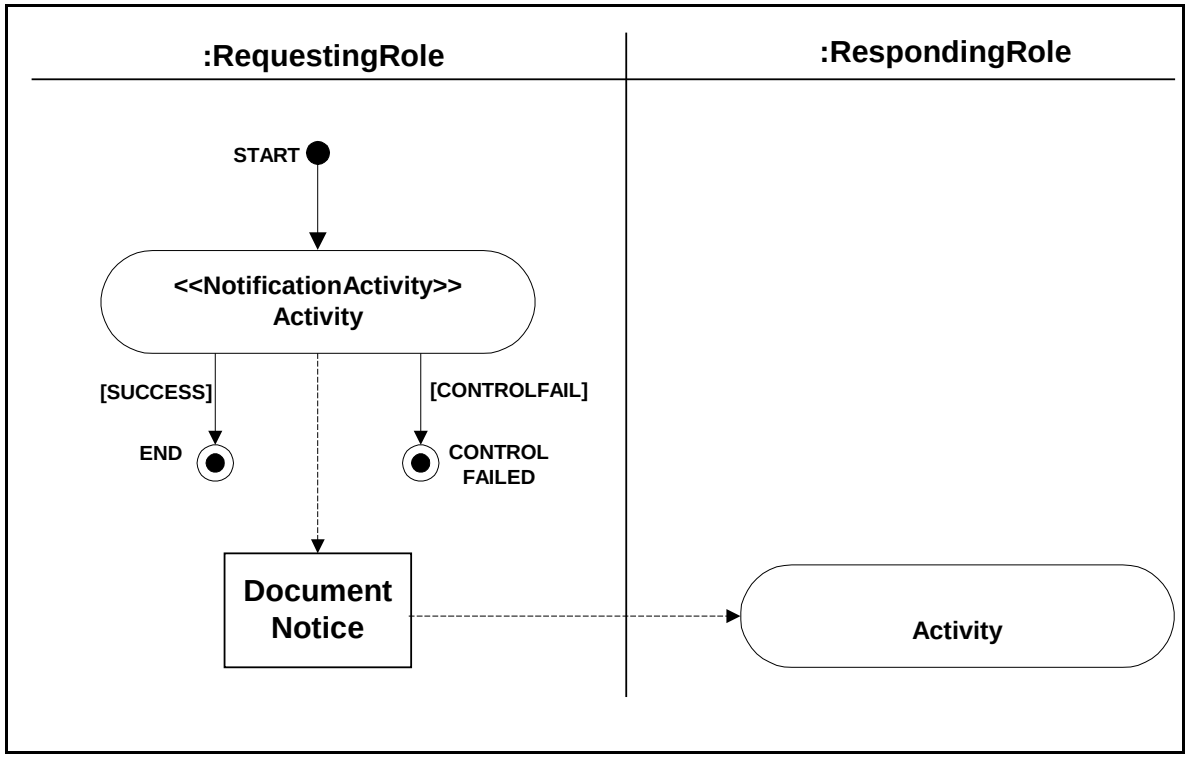

Figure 8-13 Notification Design Pattern

8.3.6.9.1 Notification of Failure Semantics

The intent of the notification of failure business transaction is to revoke an initial business contract offer if the contract formation process fails. The requesting partner can only initiate this business transaction. A responding partner is required to return an exception document or a negative acknowledgement document when an error is generated.

Notification of failure shall only be initiated when a terminating transaction does not leave both parties with a mutual agreement as to the state of a business transaction. This condition exists when:

1. The initiating partner's business activity times-out when waiting for a specified response to its requesting business document.
2. The responding business document is erroneous, not authorized or not digitally signed as agreed to in a Trading Partner Agreement.

The UN/EDIFACT model trading partner agreement (http://www.unece.org/trade/untdid/texts/d240_d.htm) recommends the following procedure be agreed to by both partners in their Trading Partner Agreement so as to leave each partner with a mutual understanding of when a contract is not formed:

> "3.2.3. In the event that the originating party has not received, for a properly transmitted Message, a required acknowledgement and no further instructions have been provided, the originating party may declare the Message null and void by so notifying the receiving party."

The contract requestor initiates this business transaction when the originating partner times-out when waiting for a specified response. Where notifications are sent is defined in a trading partner agreement and may be different for each business

transaction.
It is recommended that the Notification of Failure business transaction be executed over an alternate communication channel to prevent the inability to report failures potentially caused by communication failures. It is recommended that the organizational entity responding to the notification of failure is different from the organization that failed to respond to the original business document request (“offer”).
In an e-business network environment, this “alternate communications channel” should at least be interpreted to mean communicating with an application server that is different from the application server that has not serviced the original business document request. Trading partners should, however, agree on this “alternate communications channel.”
This business transaction is not exercised when a responding business partner encounters a business process or control exception when responding to a business document request.

8.4 Design Patterns

Networked business services and business agents are configured to execute business transactions and business collaboration agreements. The UML sequence diagram notation is used to specify Business Service interactions. The following Business Service interactions are possible.

1. Service-Service.
2. Agent-Service-Service.
3. Service-Service-Agent.
4. Service-Agent-Service.
5. Agent-Service-Agent

8.4.1 Service-Service

Figure 8-14 illustrates the Service-Service business service interaction pattern used in the business transaction patterns of Section 8.3.

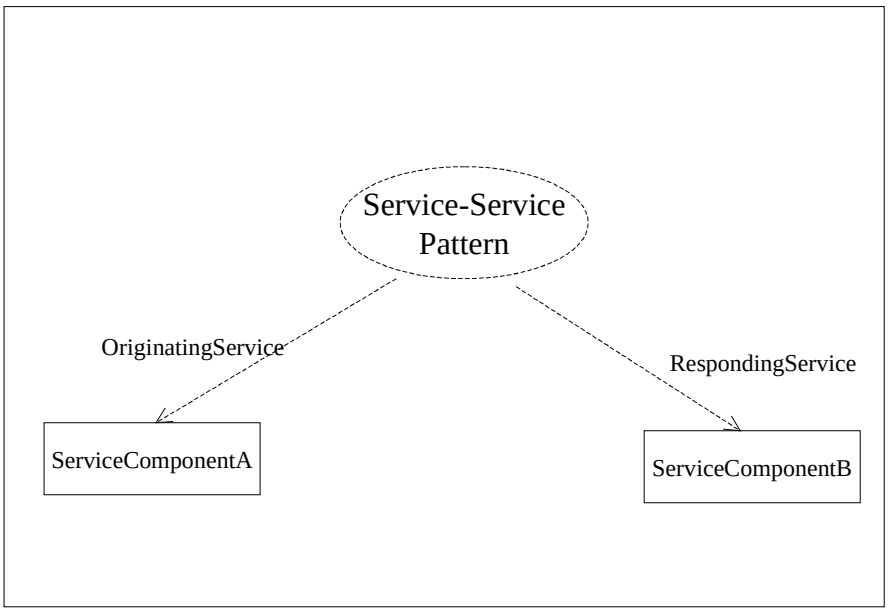

Figure 8-14 Service-Service pattern

Figure 8-15 shows the class diagram for the Service-Service pattern, which is the base pattern.

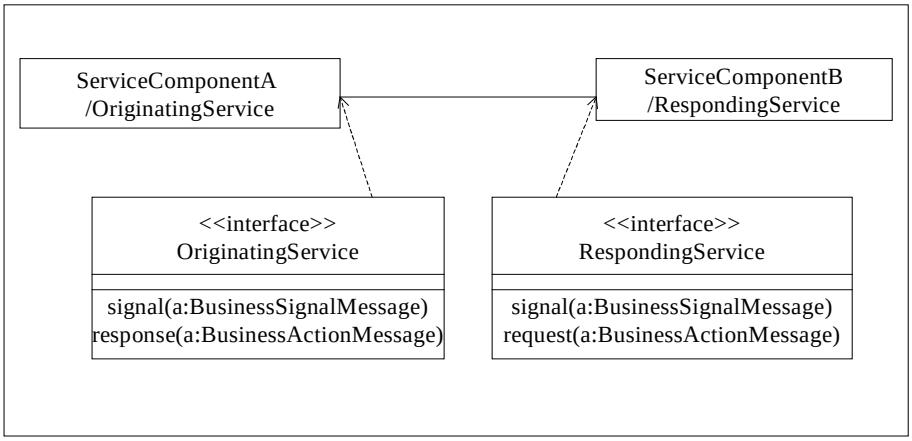

Figure 8-15 Service-Service pattern class diagram

There are five variations within the Service-Service pattern:

1. Interaction Pattern A applies to the Business Transaction Pattern where time to perform equals time to acknowledge acceptance and there is no responding business document.
2. Interaction Pattern B also applies to the Business Transaction Pattern where time to perform equals time to acknowledge acceptance and a responding business document.
3. Interaction Pattern C also applies to the Business Transaction Pattern where time to perform is greater than time to acknowledge acceptance.
4. Interaction Pattern D applies to the Query/Response, Request/Response, and Request/Confirm Patterns.
5. Interaction Pattern E applies to the Information Distribution and Notification Patterns.

Business Transaction Activity

Figure 8-16, Figure 8-17, and Figure 8-18 illustrate Interaction Patterns A, B, and C respectively.

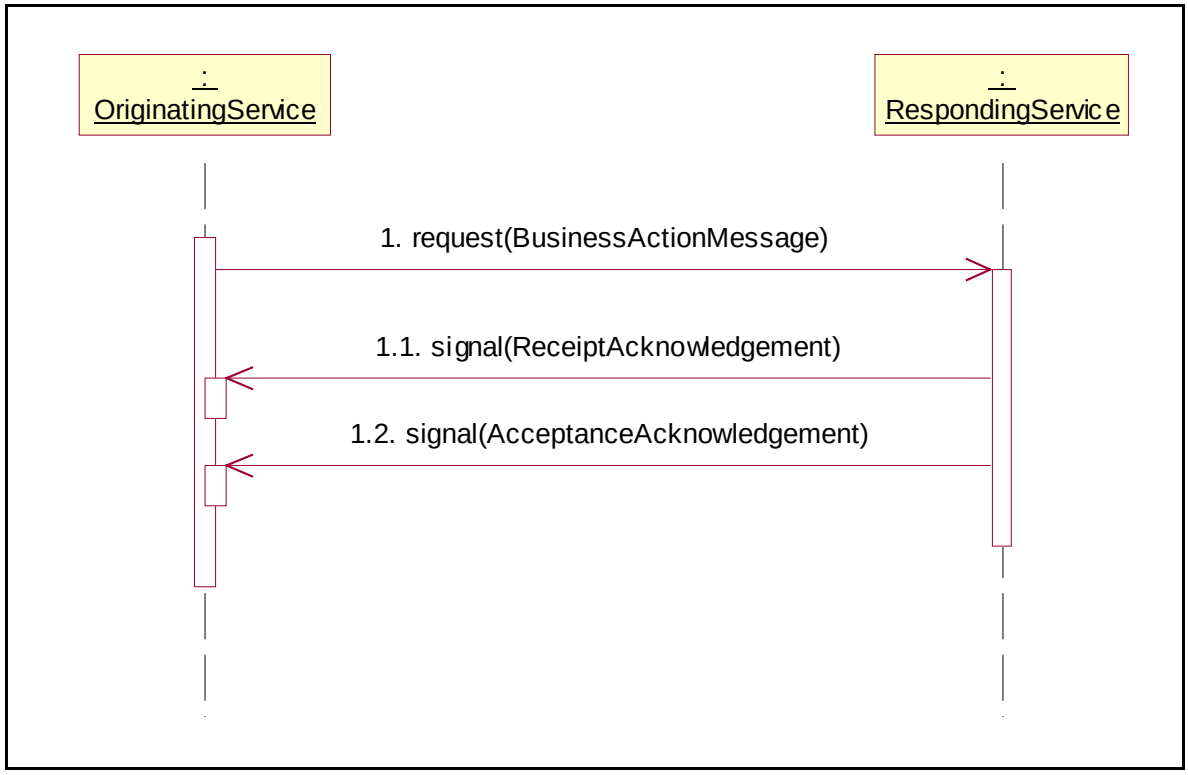

Figure 8-16 Service-Service Interaction Pattern A

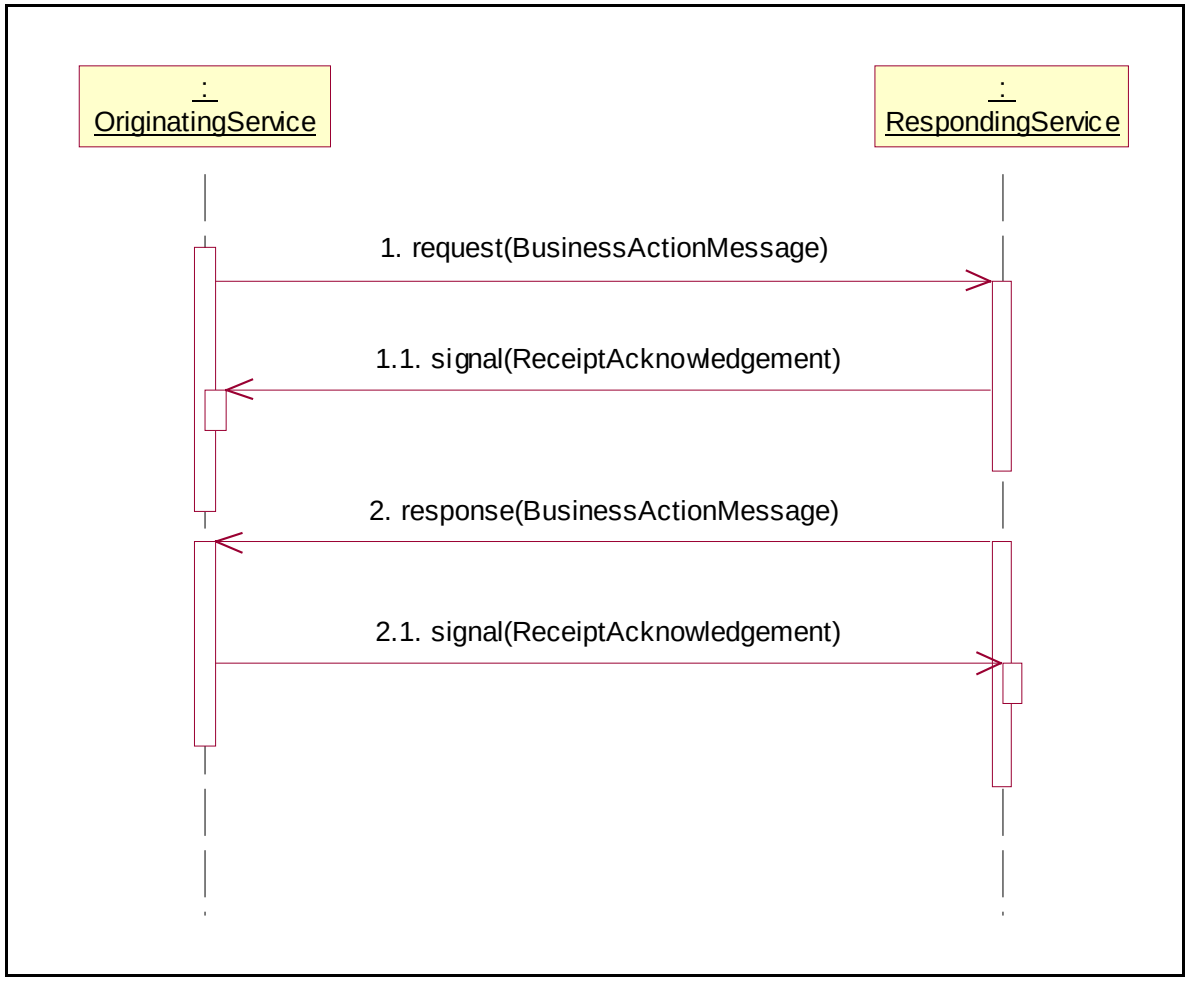

Figure 8-17 Service-Service Interaction Pattern B

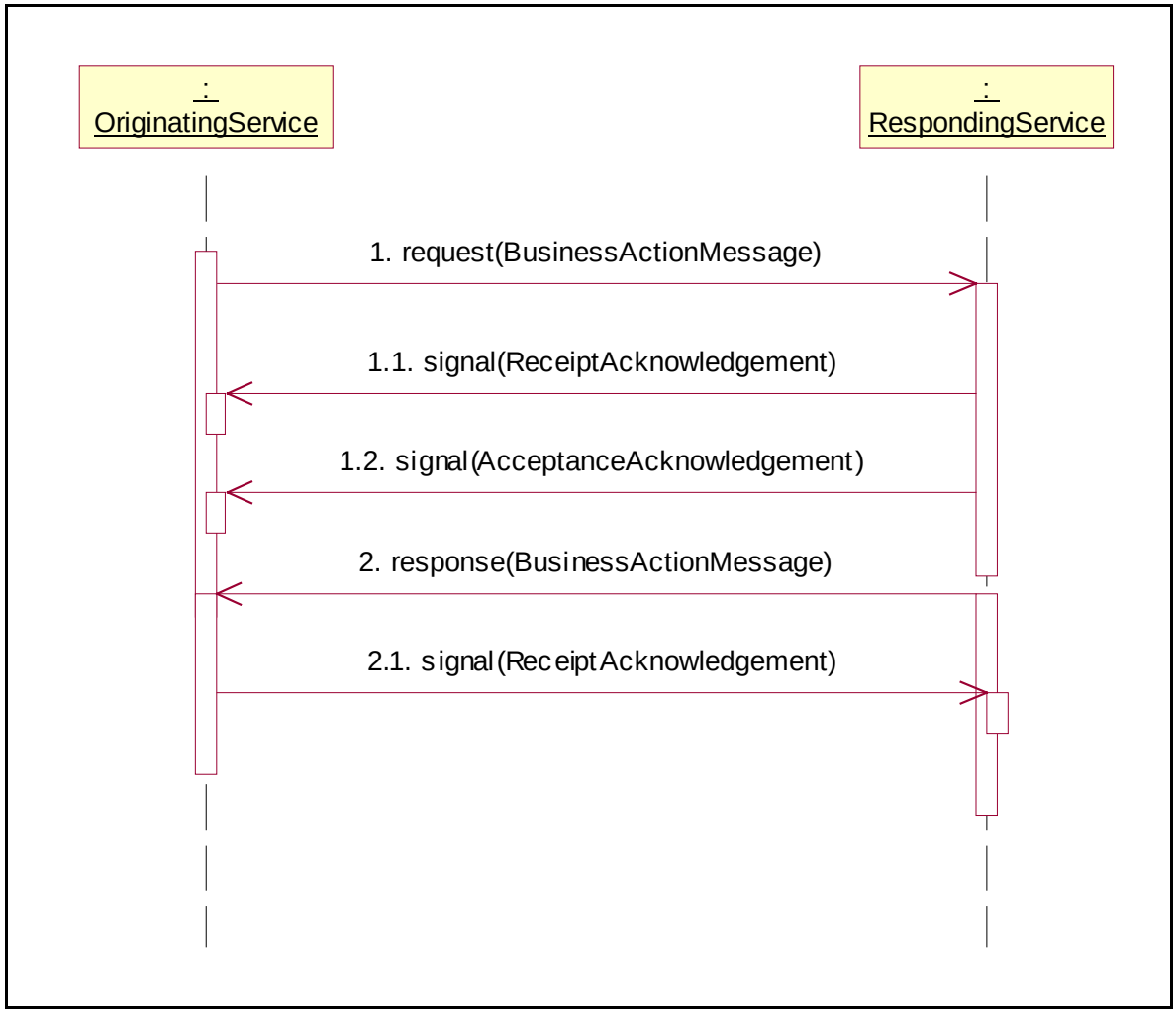

Figure 8-18 Service-Service Interaction Pattern C

Query/Response, Request/Response, and Request/Confirm Activities

Figure 8-19 illustrates Interaction Pattern D.

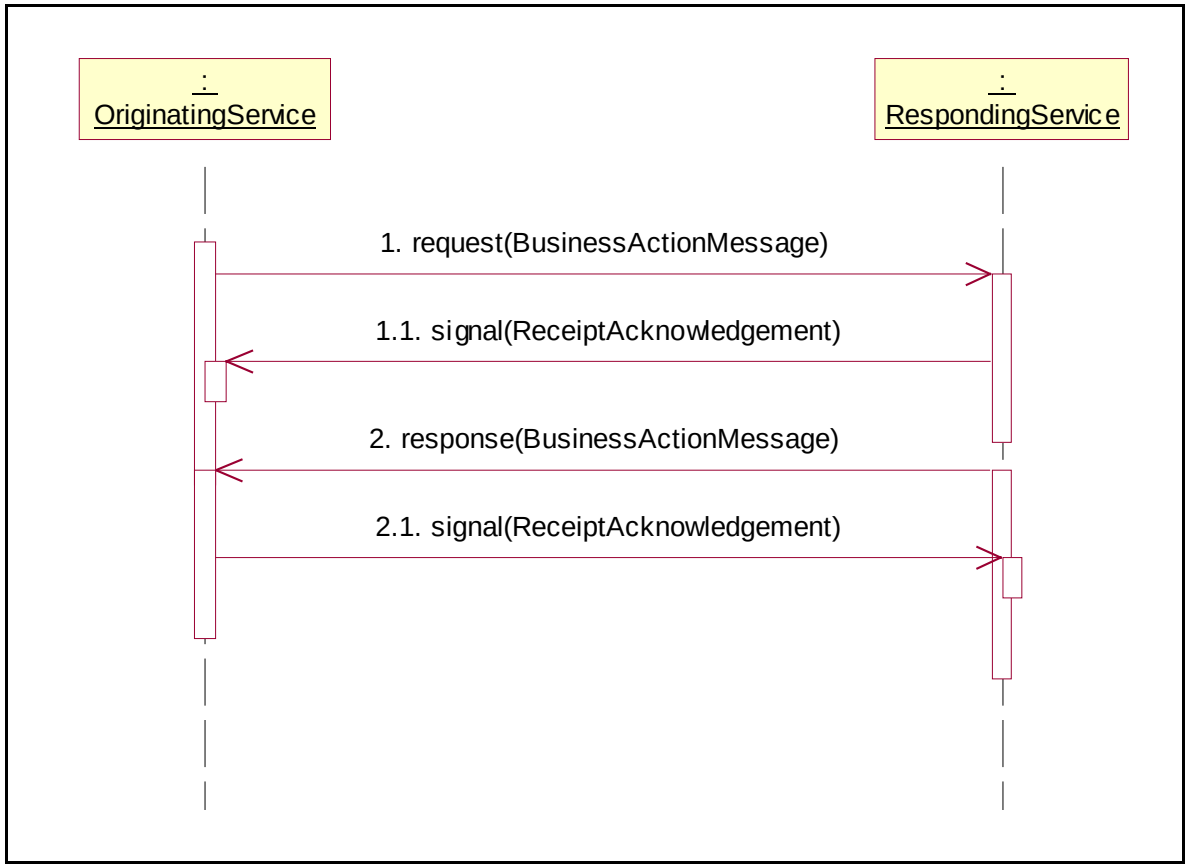

Figure 8-19 Service-Service Interaction Pattern D

Information Distribution and Notification Activities

Figure 8-20 illustrates Interaction Pattern E.

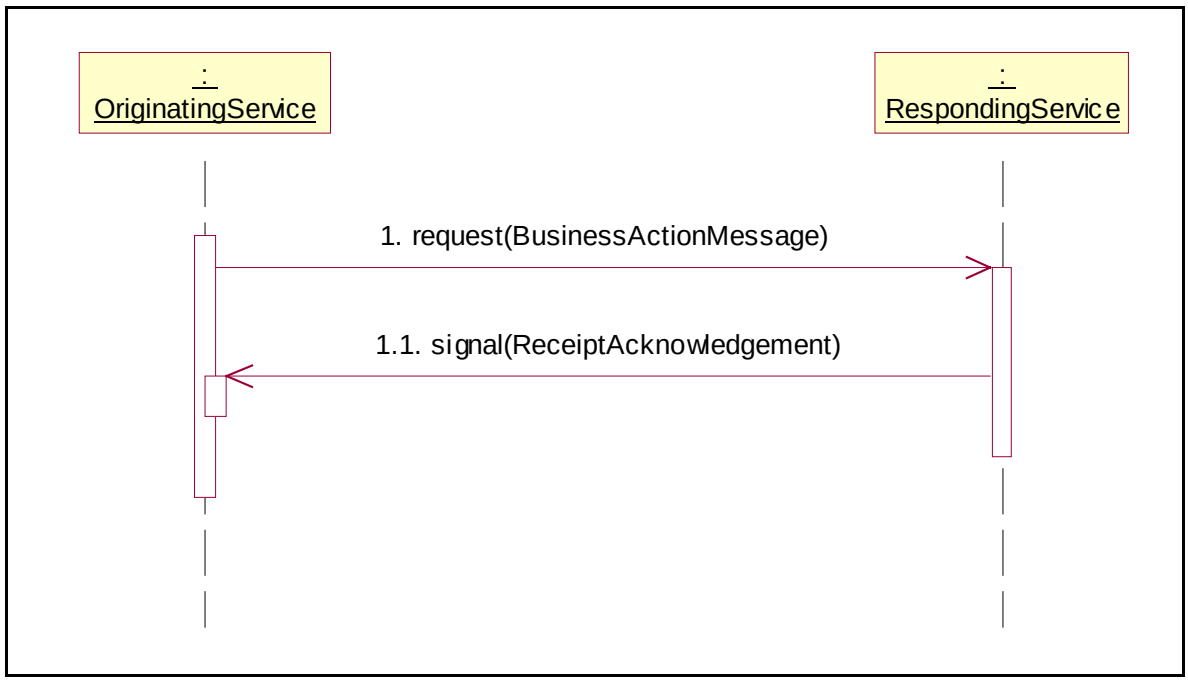

Figure 8-20 Service-Service Interaction Pattern E

8.4.2 Agent-Service-Service

Figure 8-21 illustrates the Agent-Service-Service business service interaction pattern used in the business transaction patterns of Section 8.3.

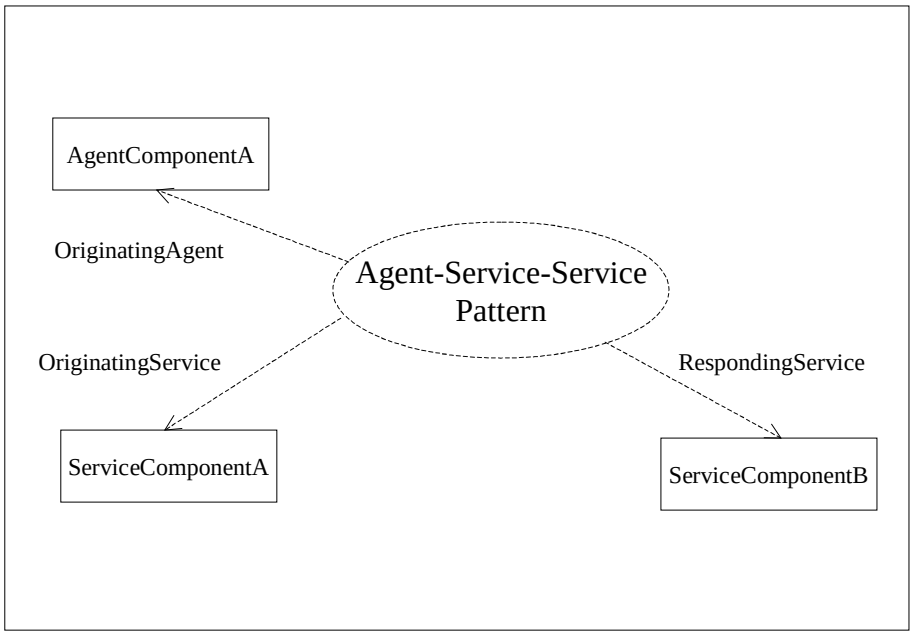

Figure 8-21 Agent-Service-Service Pattern

Figure 8-22 shows the class diagram for the Agent-Service-Service pattern, and Figure 8-23 shows the class diagram for the Agent-Service-Service unfolded from the base pattern.

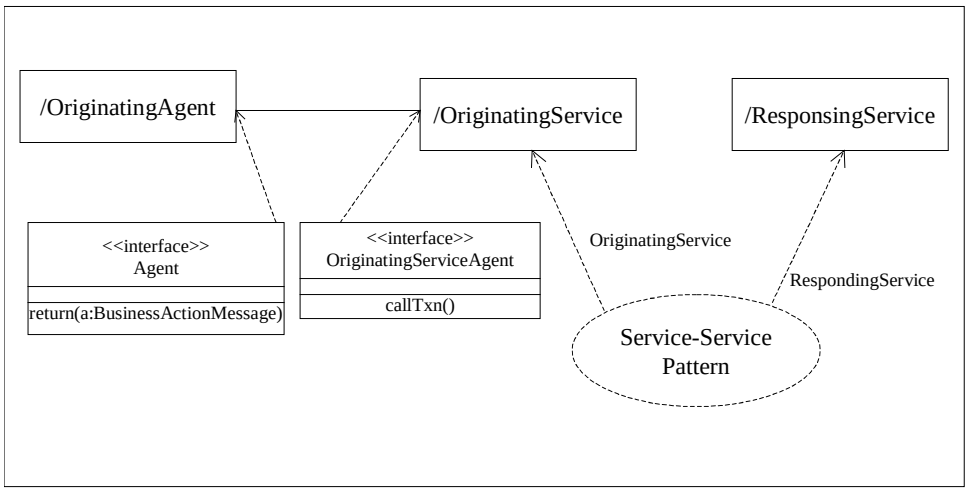

Figure 8-22 Agent-Service-Service pattern class diagram

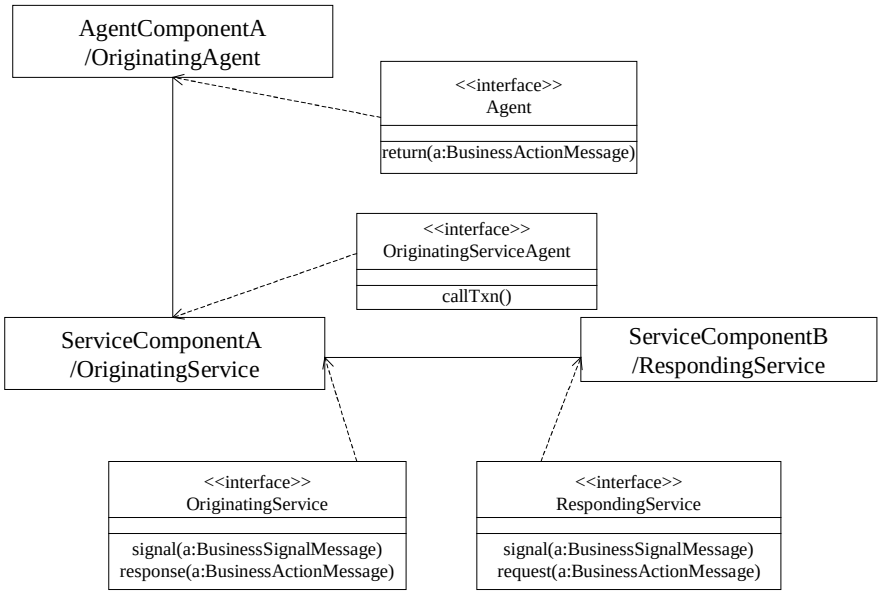

Figure 8-23 Agent-Service-Service pattern class diagram unfolded

There are five variations within the Agent-Service-Service pattern:

1. Interaction Pattern A applies to the Business Transaction Pattern where time to perform equals time to acknowledge acceptance and there is no responding business document.
2. Interaction Pattern B also applies to the Business Transaction Pattern where time to perform equals time to acknowledge acceptance and a responding business document.
3. Interaction Pattern C also applies to the Business Transaction Pattern where time to perform is greater than time to acknowledge acceptance.
4. Interaction Pattern D applies to the Query/Response, Request/Response, and Request/Confirm Patterns.
5. Interaction Pattern E applies to the Information Distribution and Notification Patterns.

Business Transaction Activity

Figure 8-24, Figure 8-25, and Figure 8-26 illustrate Interaction Patterns A, B, and C respectively.

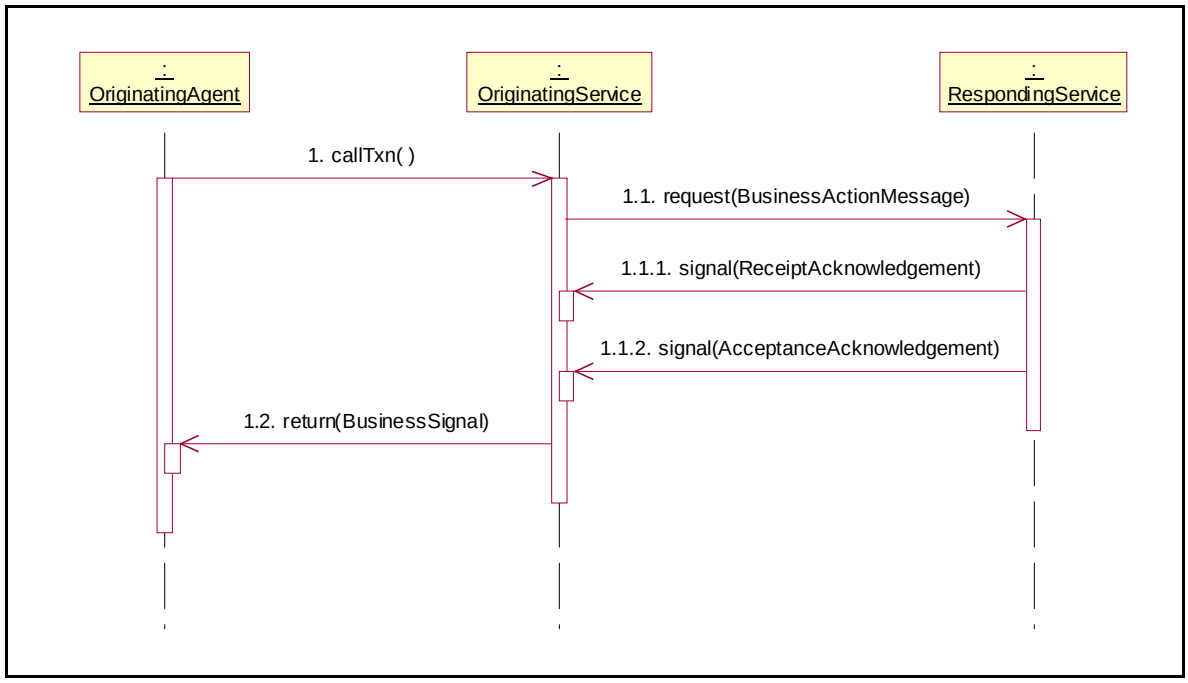

Figure 8-24 Agent-Service-Service Interaction Pattern A

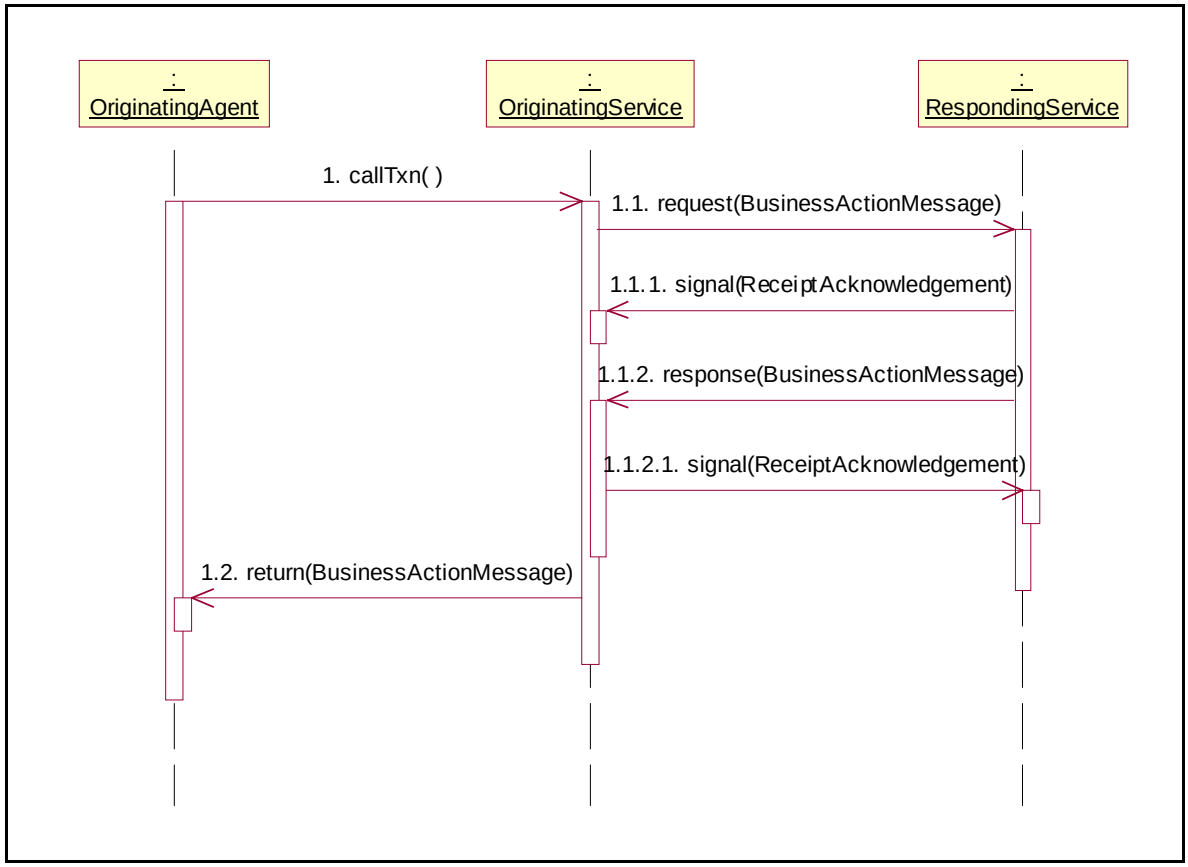

Figure 8-25 Agent-Service-Service Interaction Pattern B

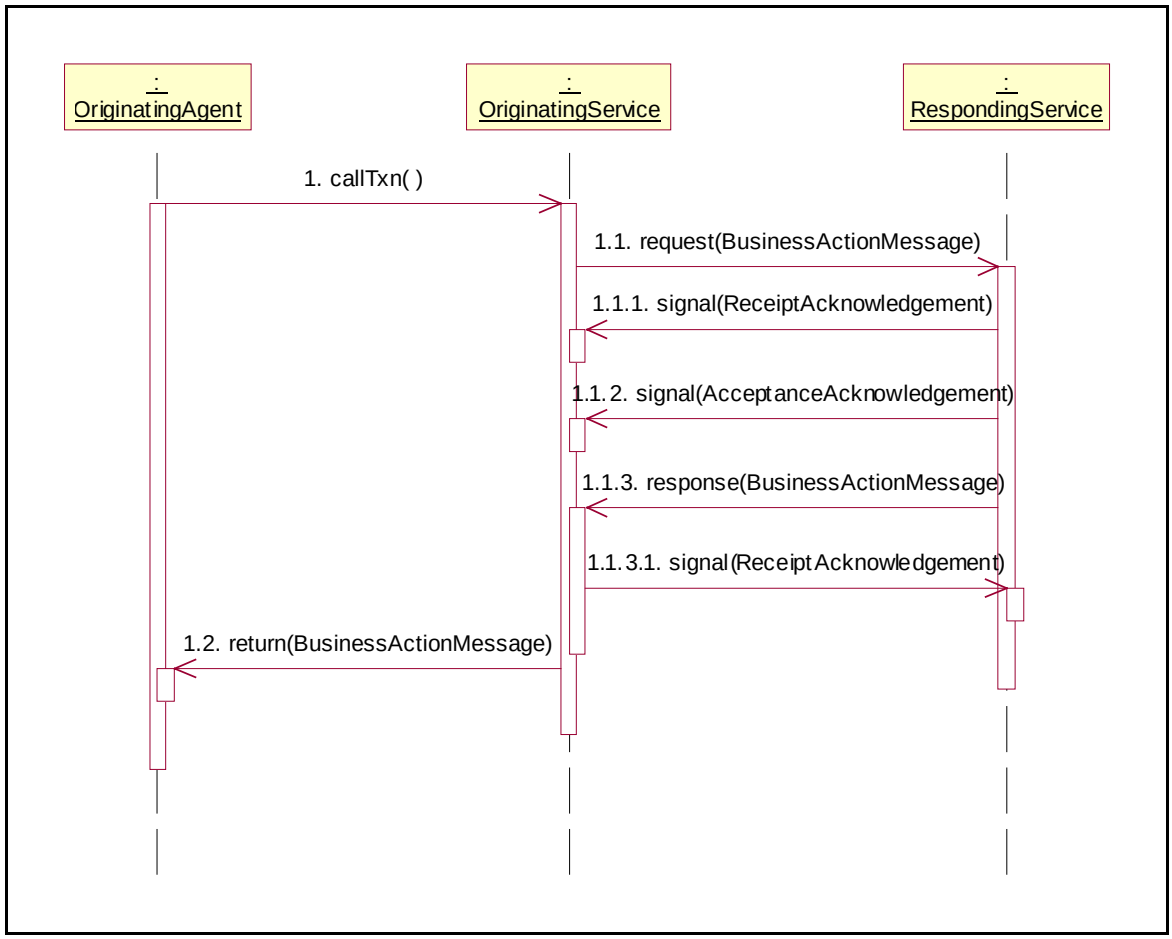

Figure 8-26 Agent-Service-Service Interaction Pattern C

Query/Response, Request/Response, and Request/Confirm Activities

Figure 8-27 illustrates Interaction Pattern D.

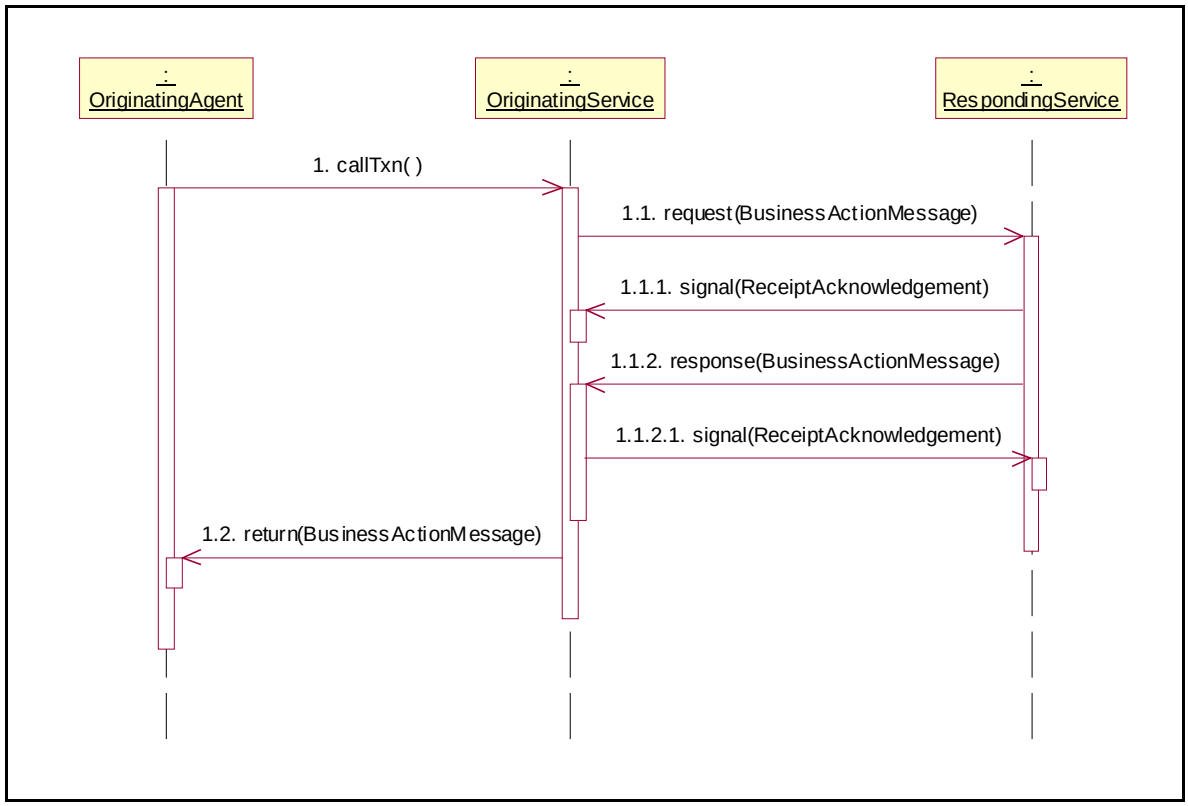

Figure 8-27 Agent-Service-Service Interaction Pattern D

Information Distribution and Notification Activities

Figure 8-28 illustrates Interaction Pattern E.

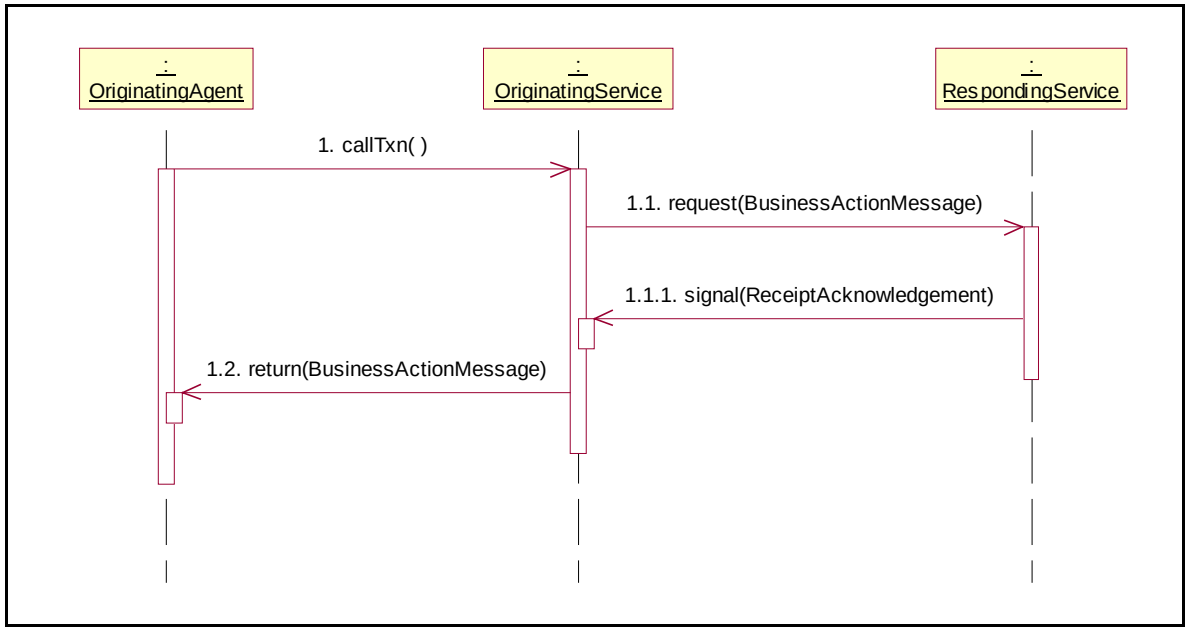

Figure 8-28 Agent-Service-Service Interaction Pattern E

8.4.3 Service-Service-Agent

Figure 8-29 illustrates the Service-Service-Agent business service interaction pattern used in the business transaction patterns of Section 8.3.

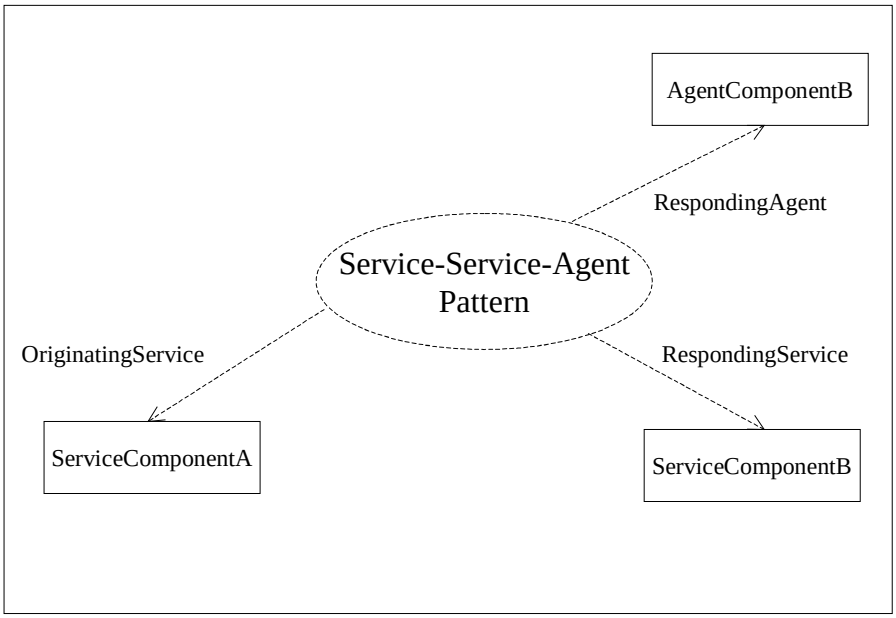

Figure 8-29 Service-Service-Agent pattern

Figure 8-30 shows the class diagram for the Service-Service-Agent pattern, and Figure 8-31 shows the class diagram for the Service-Service-Agent unfolded from the base pattern.

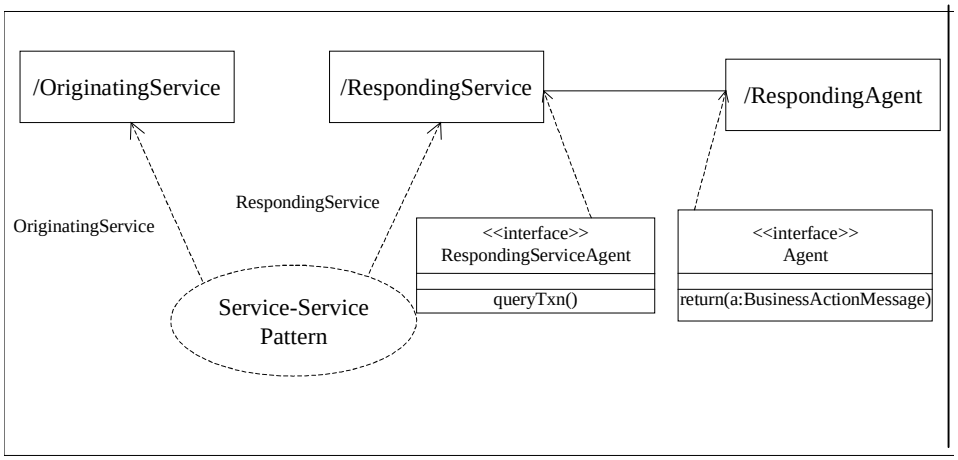

Figure 8-30 Service-Service-Agent pattern class diagram

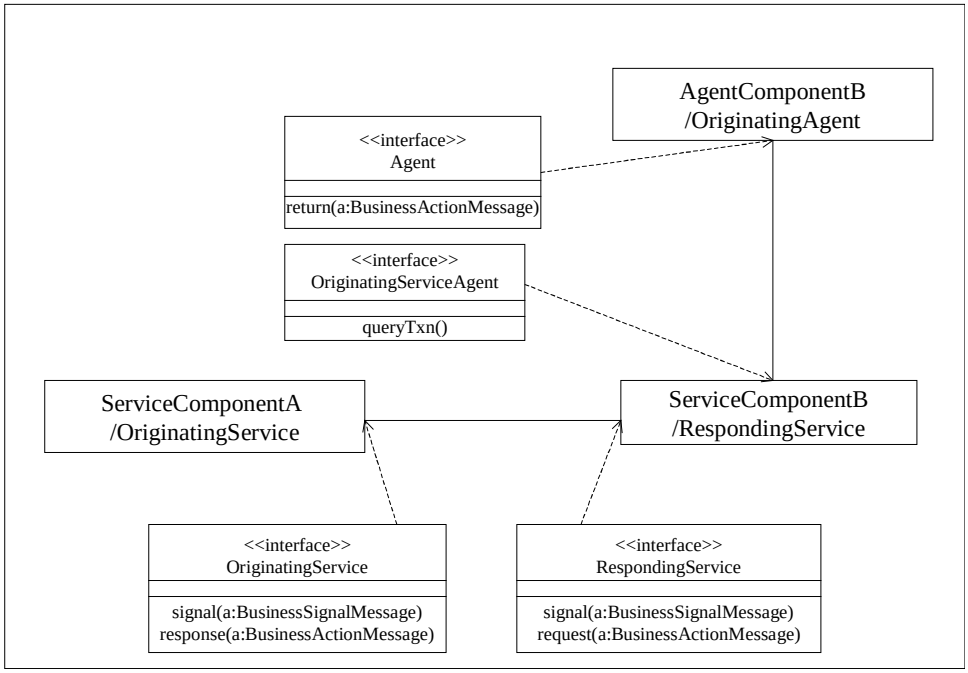

Figure 8-31 Service-Service-Agent pattern class diagram unfolded

There are five variations within the Service-Service-Agent pattern:

1. Interaction Pattern A applies to the Business Transaction Pattern where time to perform equals time to acknowledge acceptance and there is no responding business document.
2. Interaction Pattern B also applies to the Business Transaction Pattern where time to perform equals time to acknowledge acceptance and a responding business document.
3. Interaction Pattern C also applies to the Business Transaction Pattern where time to perform is greater than time to acknowledge acceptance.
4. Interaction Pattern D applies to the Query/Response, Request/Response, and Request/Confirm Patterns.
5. Interaction Pattern E applies to the Information Distribution and Notification Patterns.

Business Transaction Activity

Figure 8-32, Figure 8-33, and Figure 8-34 illustrate Interaction Patterns A, B, and C respectively.

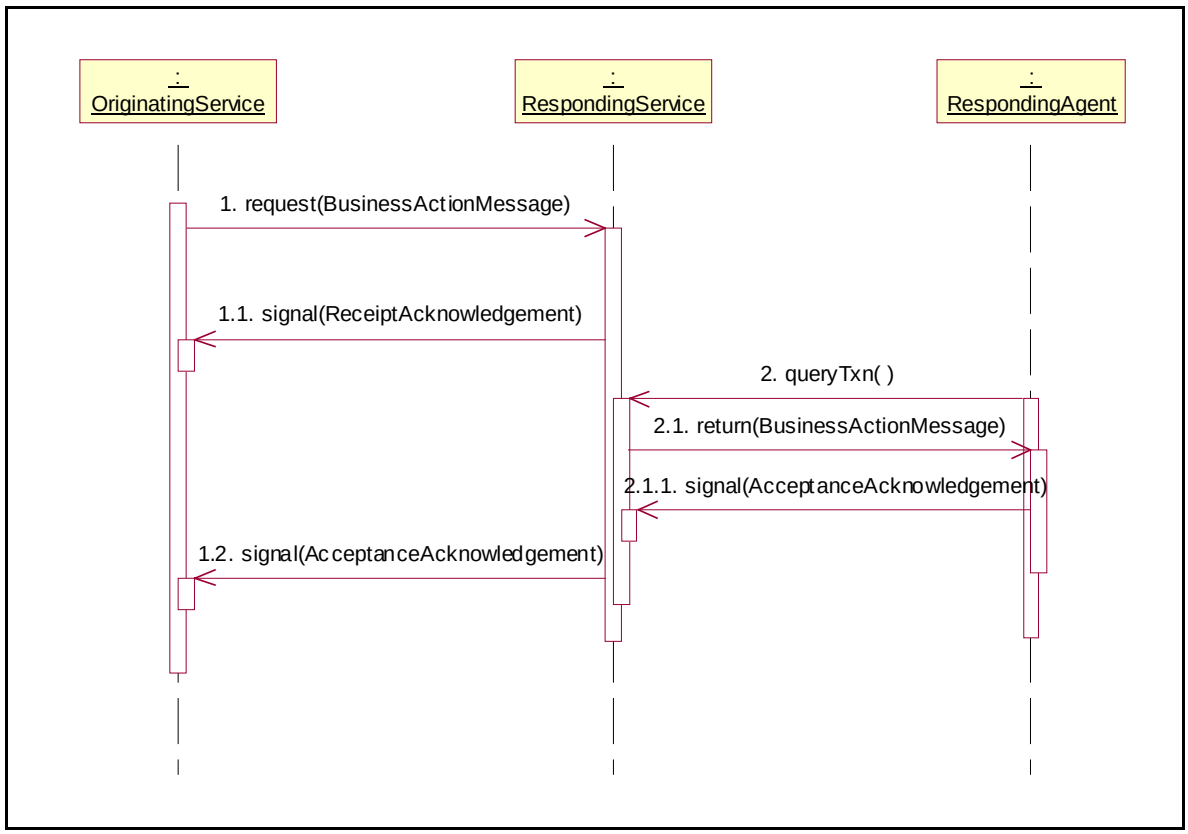

Figure 8-32 Service-Service-Agent Interaction Pattern A

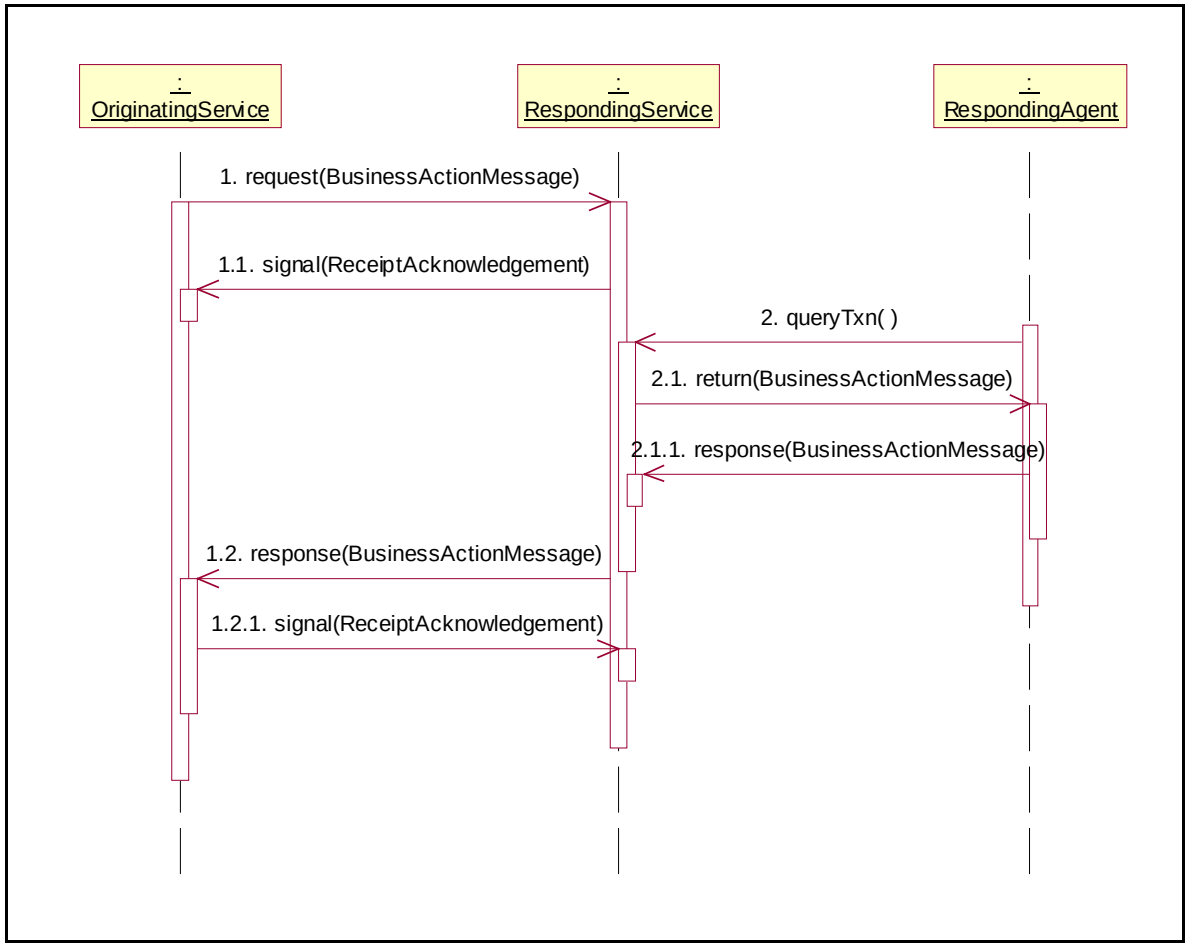

Figure 8-33 Service-Service-Agent Interaction Pattern B

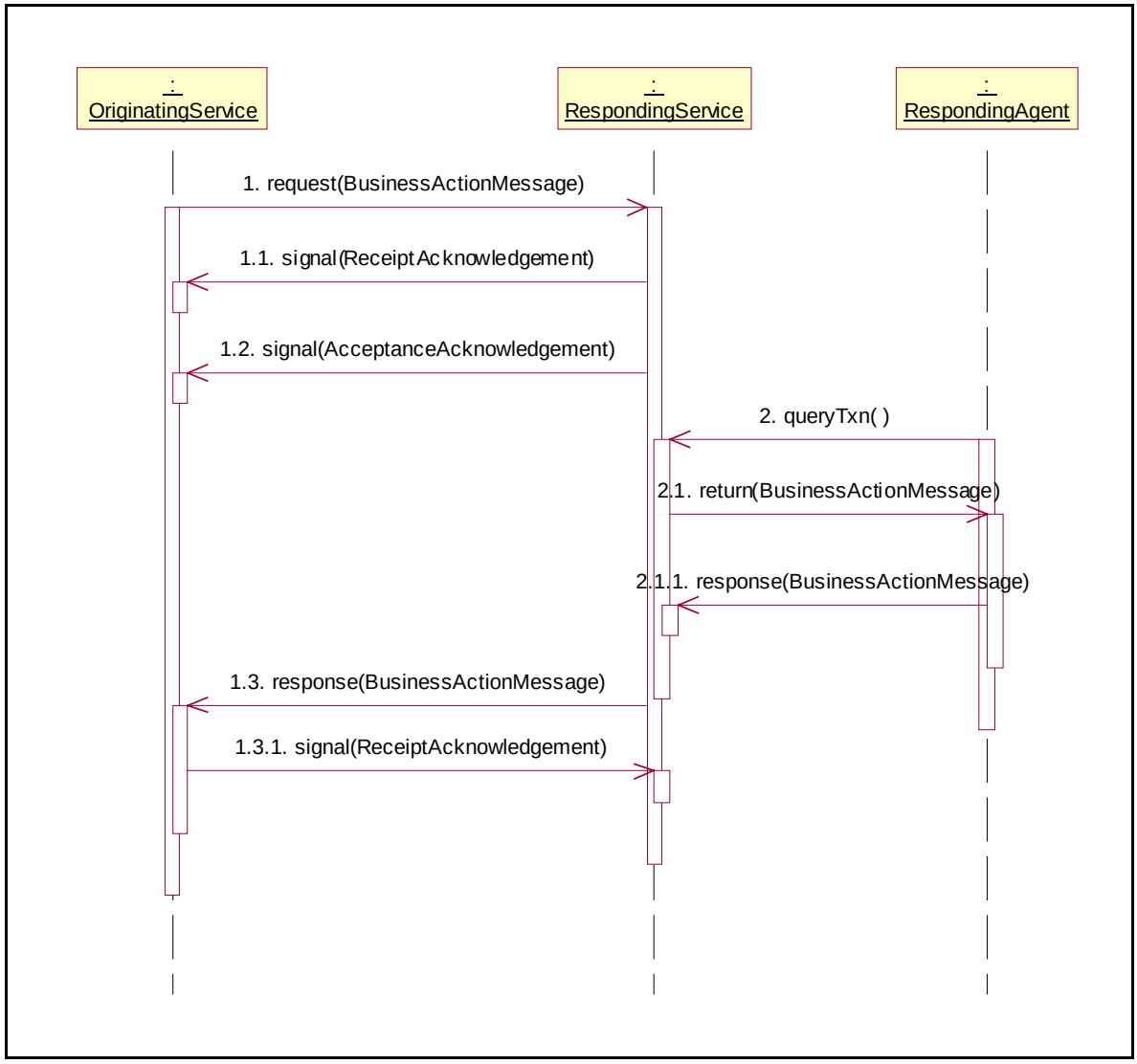

Figure 8-34 Service-Service-Agent Interaction Pattern C

Query/Response, Request/Response, and Request/Confirm Activities

Figure 8-35 illustrates Interaction Pattern D.

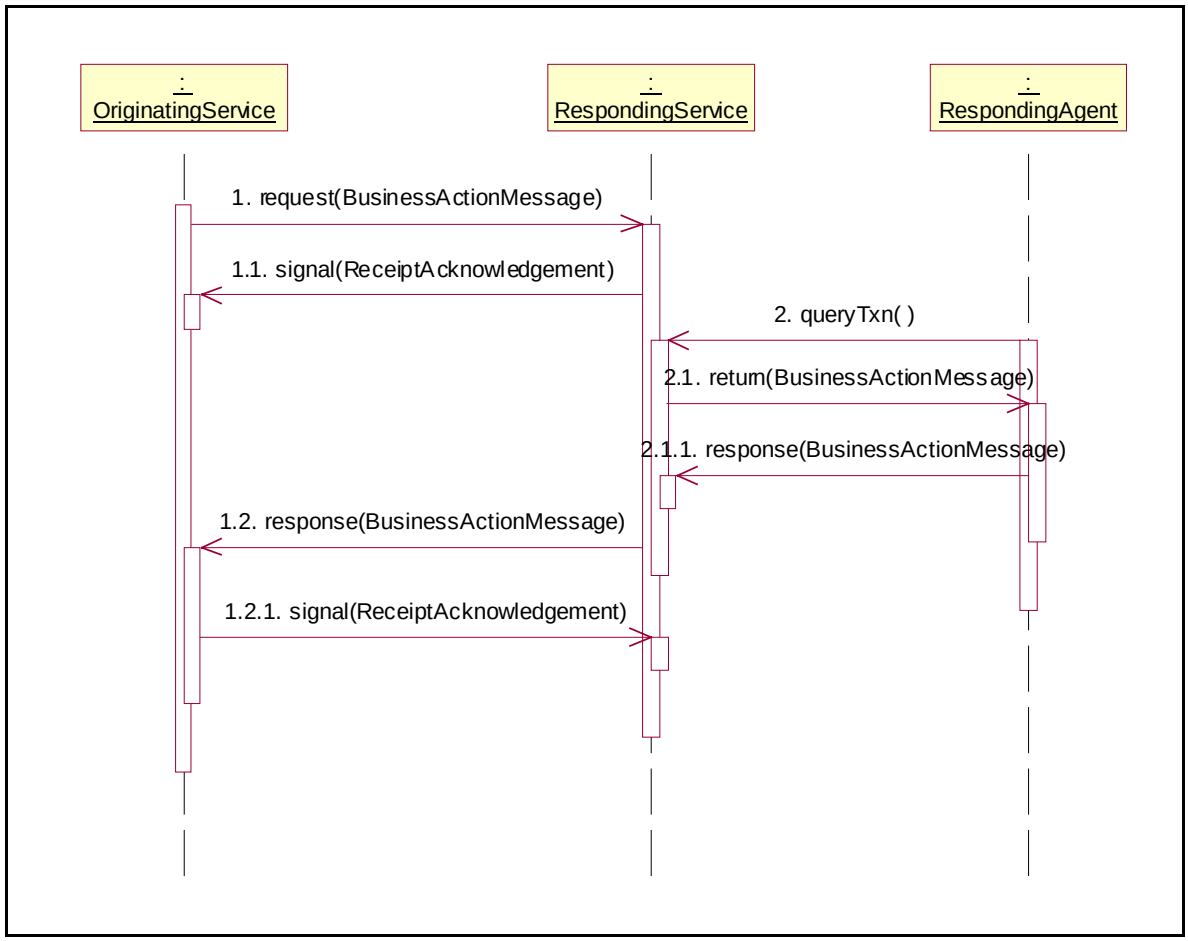

Figure 8-35 Service-Service-Agent Interaction Pattern D

Information Distribution and Notification Activities

Figure 8-36 illustrates Interaction Pattern E.

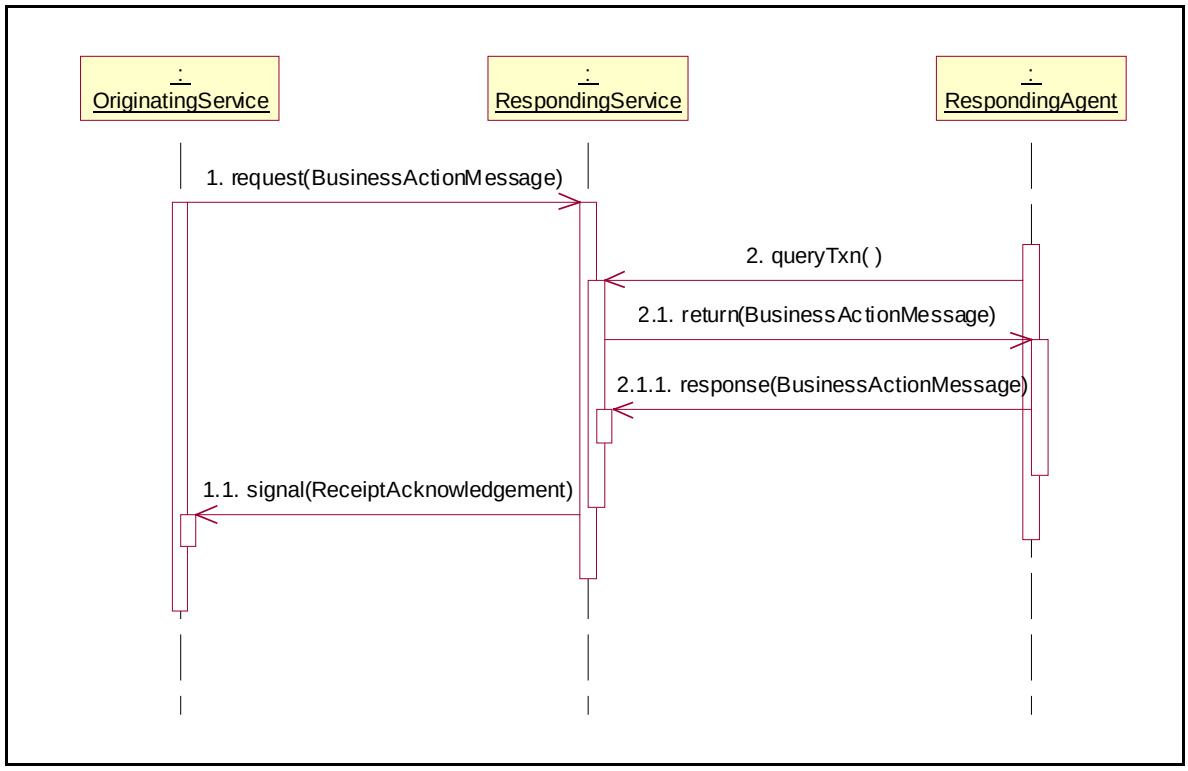

Figure 8-36 Service-Service-Agent Interaction Pattern E

8.4.4 Service-Agent-Service

Figure 8-37 illustrates the Service-Agent-Service business service interaction pattern used in the business transaction patterns of Section 8.3.

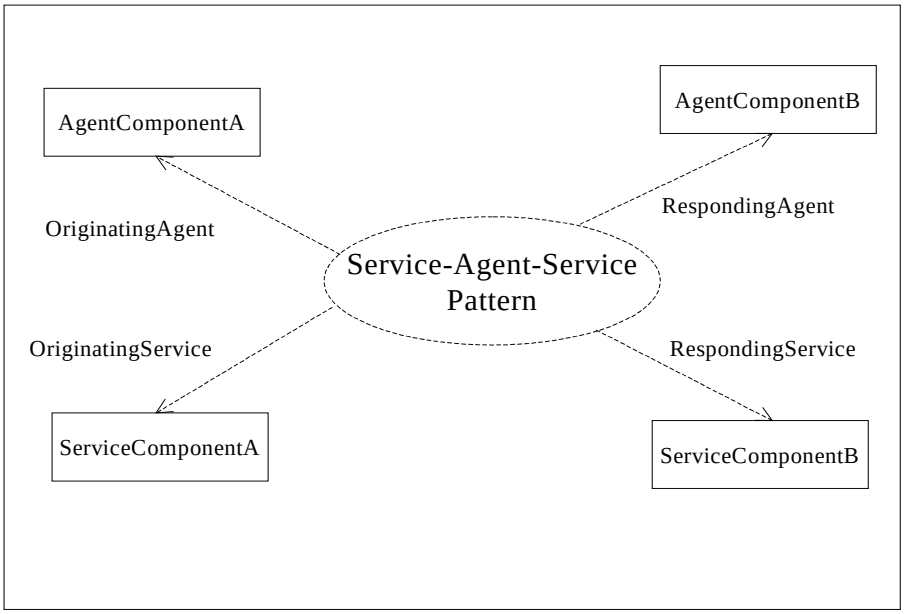

Figure 8-37 Service-Agent-Service pattern

Figure 8-38 shows the class diagram for the Service-Agent-Service pattern, and Figure 8-39 shows the class diagram for the Service-Agent-Service unfolded from the base pattern.

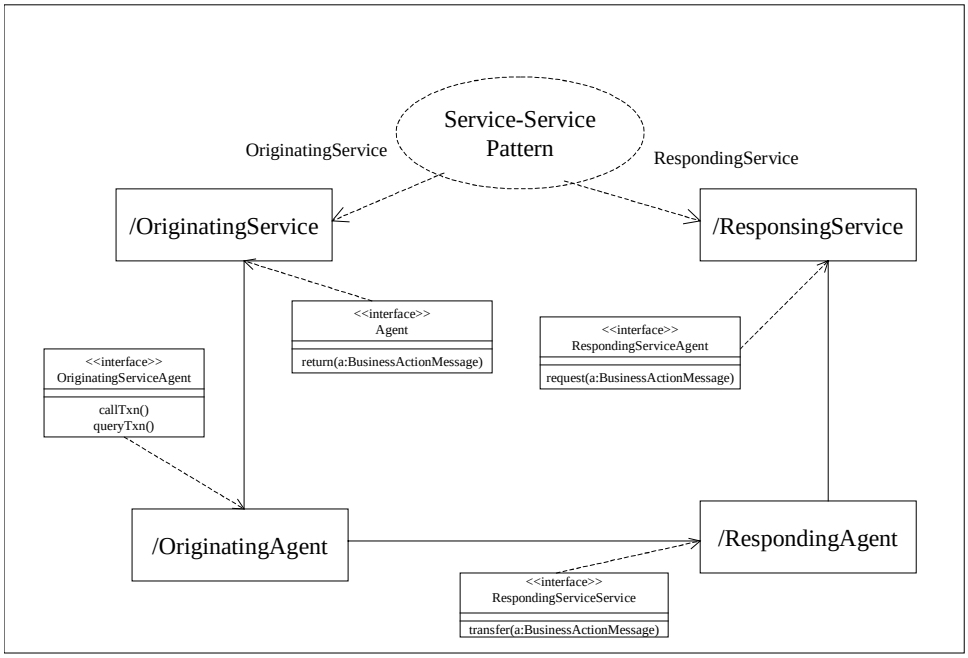

Figure 8-38 Service-Agent-Service pattern class diagram

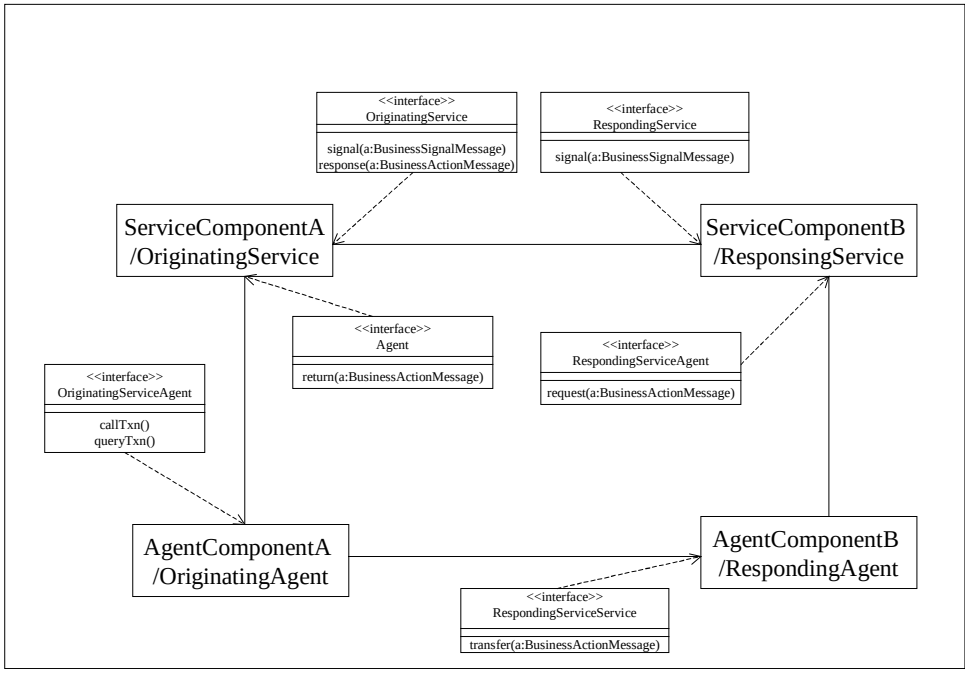

Figure 8-39 Service-Agent-Service pattern class diagram unfolded

There are six variations within the Service-Agent-Service pattern:

1. Interaction Pattern A applies to the Business Transaction Pattern where time to perform equals time to acknowledge acceptance and there is no responding business document.
2. Interaction Pattern B also applies to the Business Transaction Pattern where time to perform equals time to acknowledge acceptance and a responding business document.
3. Interaction Pattern C also applies to the Business Transaction Pattern where time to perform is greater than time to acknowledge acceptance.
4. Interaction Pattern D applies to the Query/Response and Request/Response Patterns.
5. Interaction Pattern E applies to the Request/Confirm Pattern.
6. Interaction Pattern F applies to the Information Distribution and Notification Patterns.

Business Transaction Activity

Figure 8-40, Figure 8-41, and Figure 8-42 illustrate Interaction Patterns A, B, and C respectively.

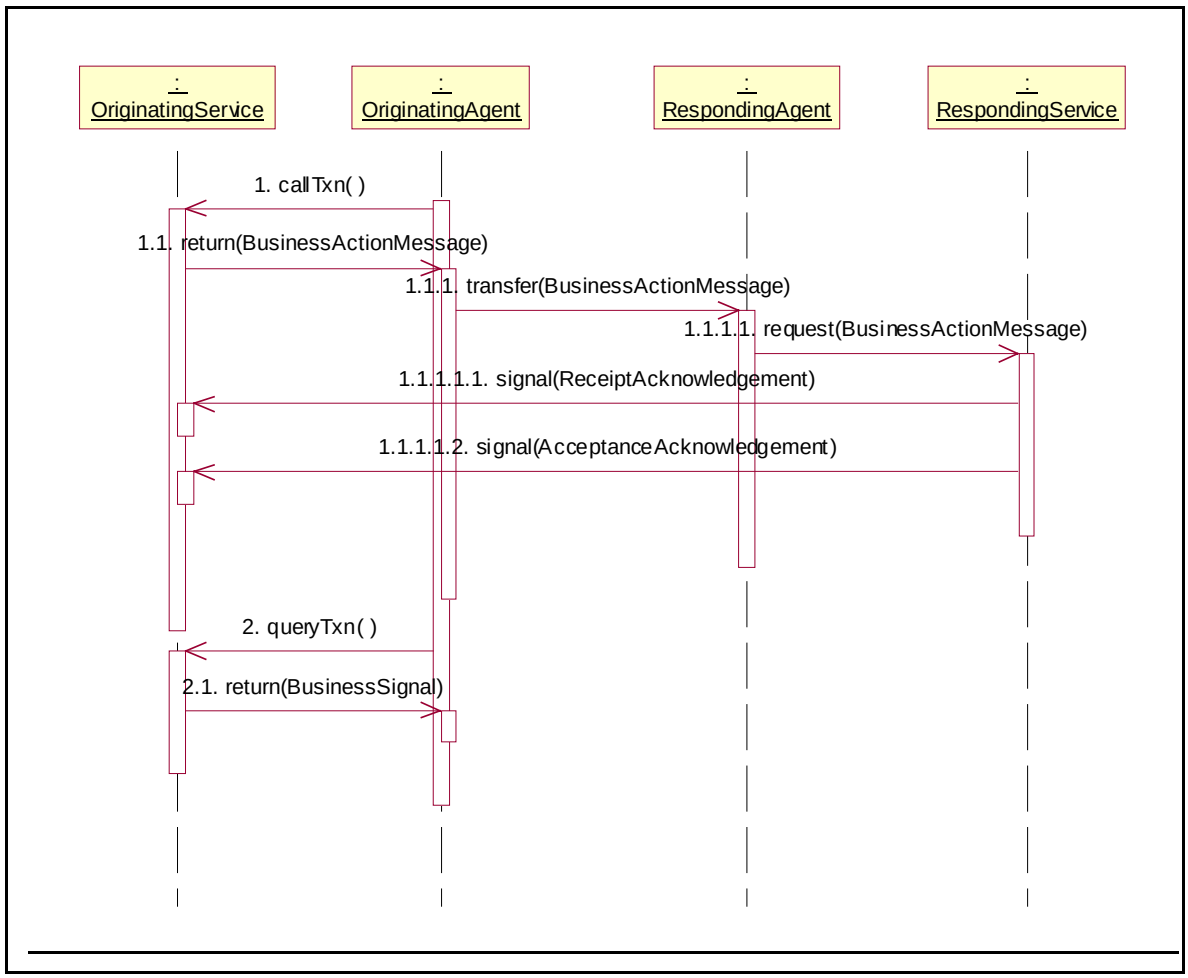

Figure 8-40 Service-Agent-Service Interaction Pattern A

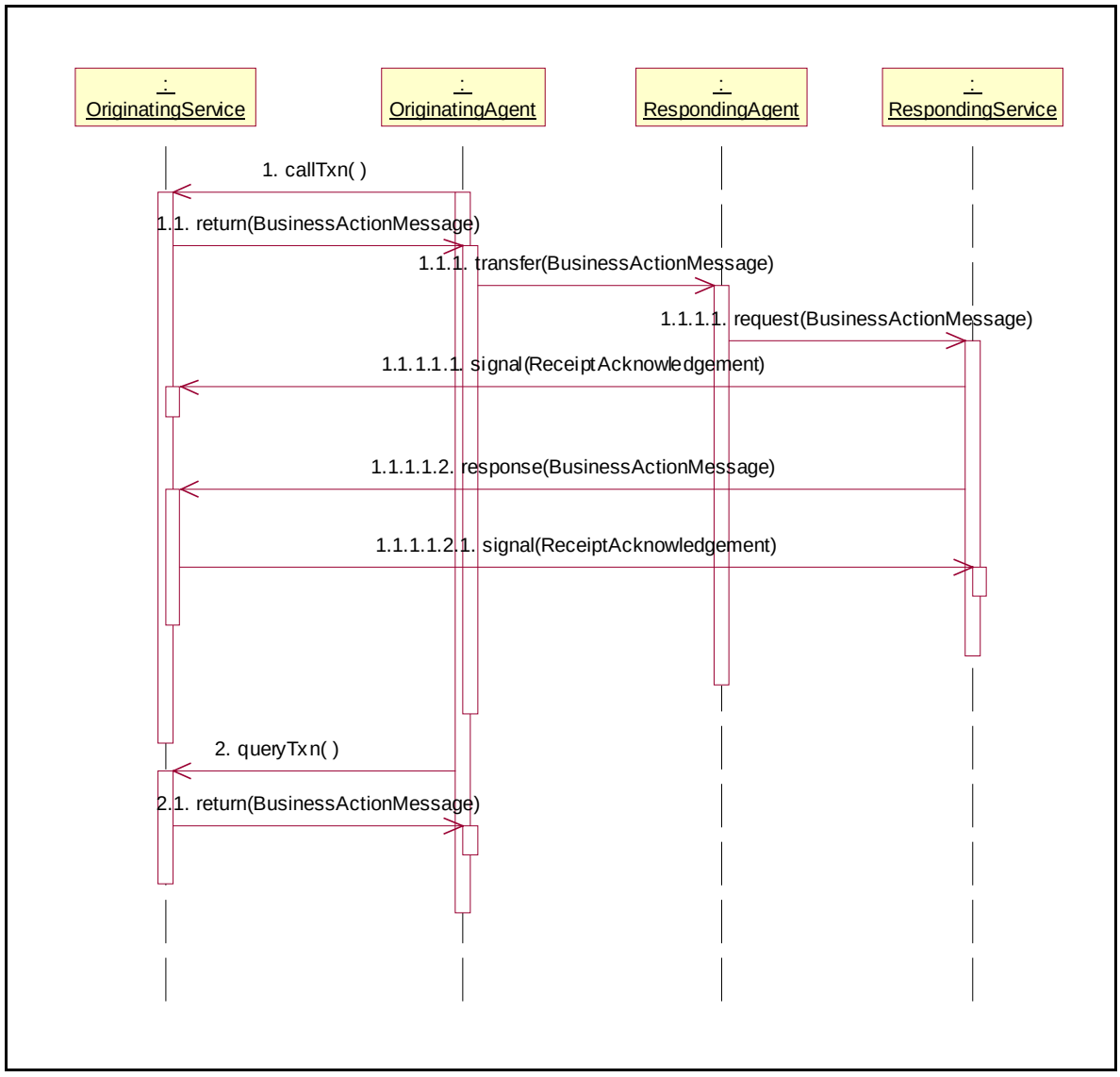

Figure 8-41 Service-Agent-Service Interaction Pattern B

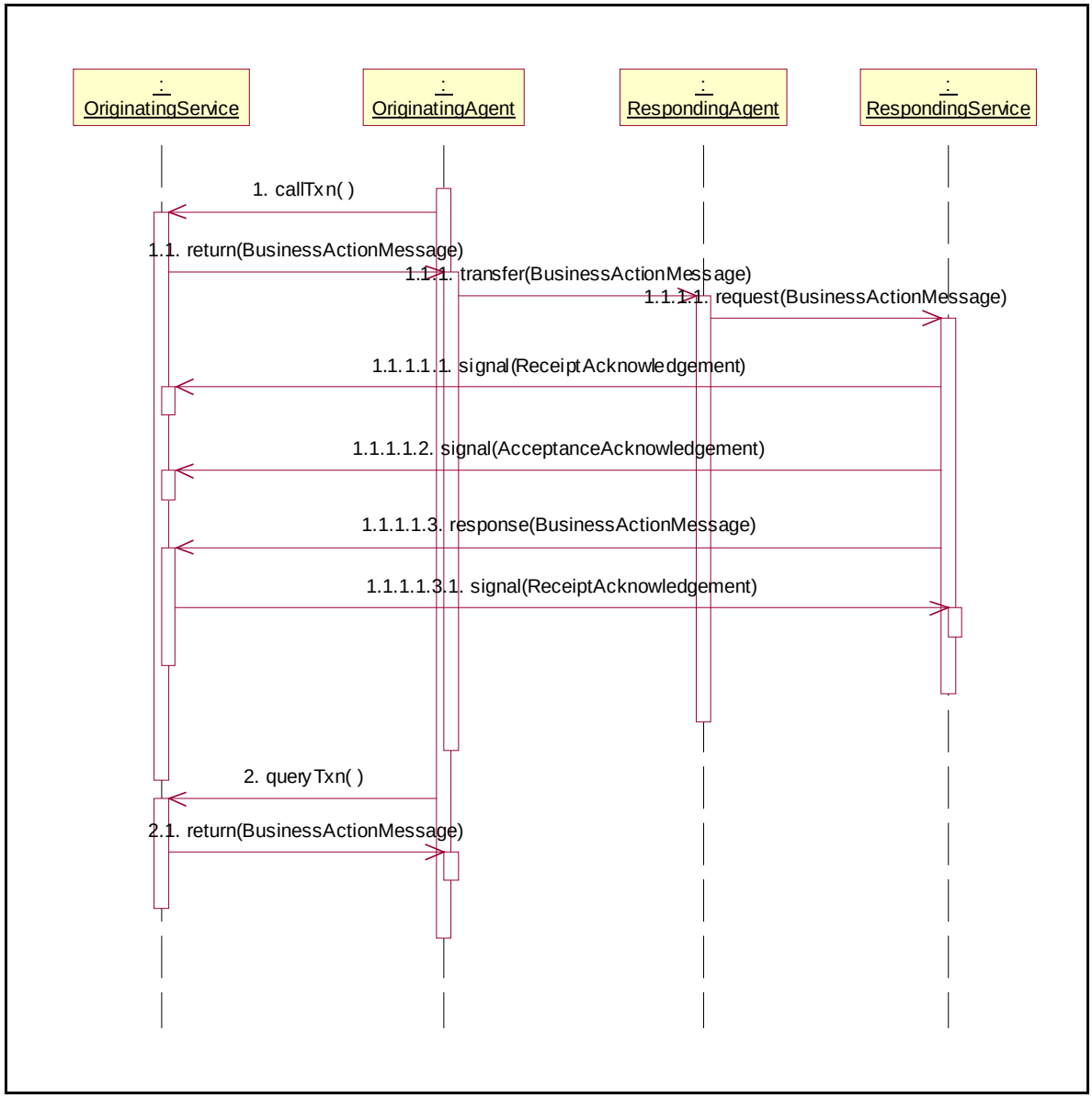

Figure 8-42 Service-Agent-Service Interaction Pattern C

Query/Response and Request/Response Activities

Figure 8-43 illustrates Interaction Pattern D.

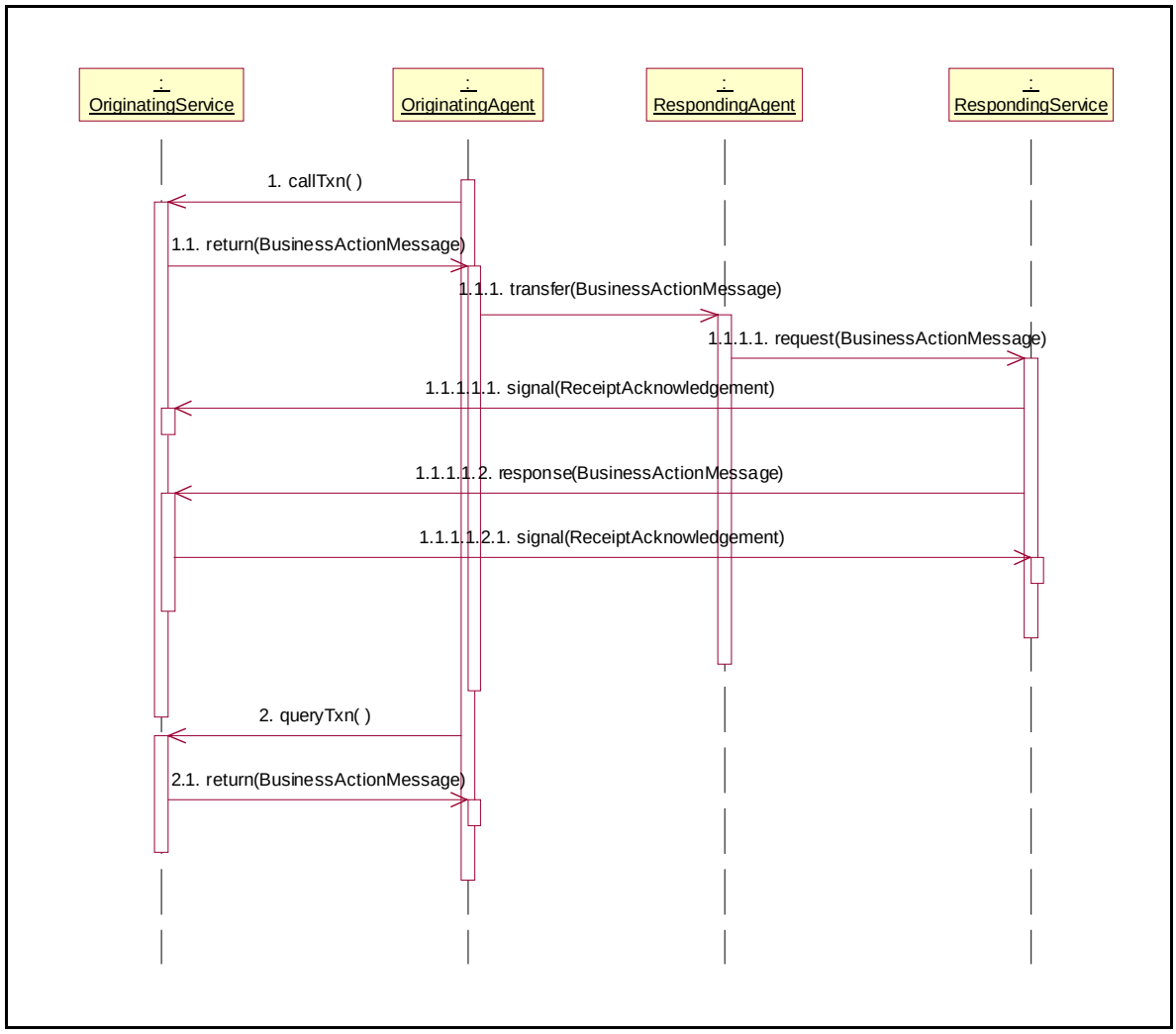

Figure 8-43 Service-Agent-Service Interaction Pattern D

Request/Confirm Activity

Figure 8-44 illustrates Interaction Pattern E.

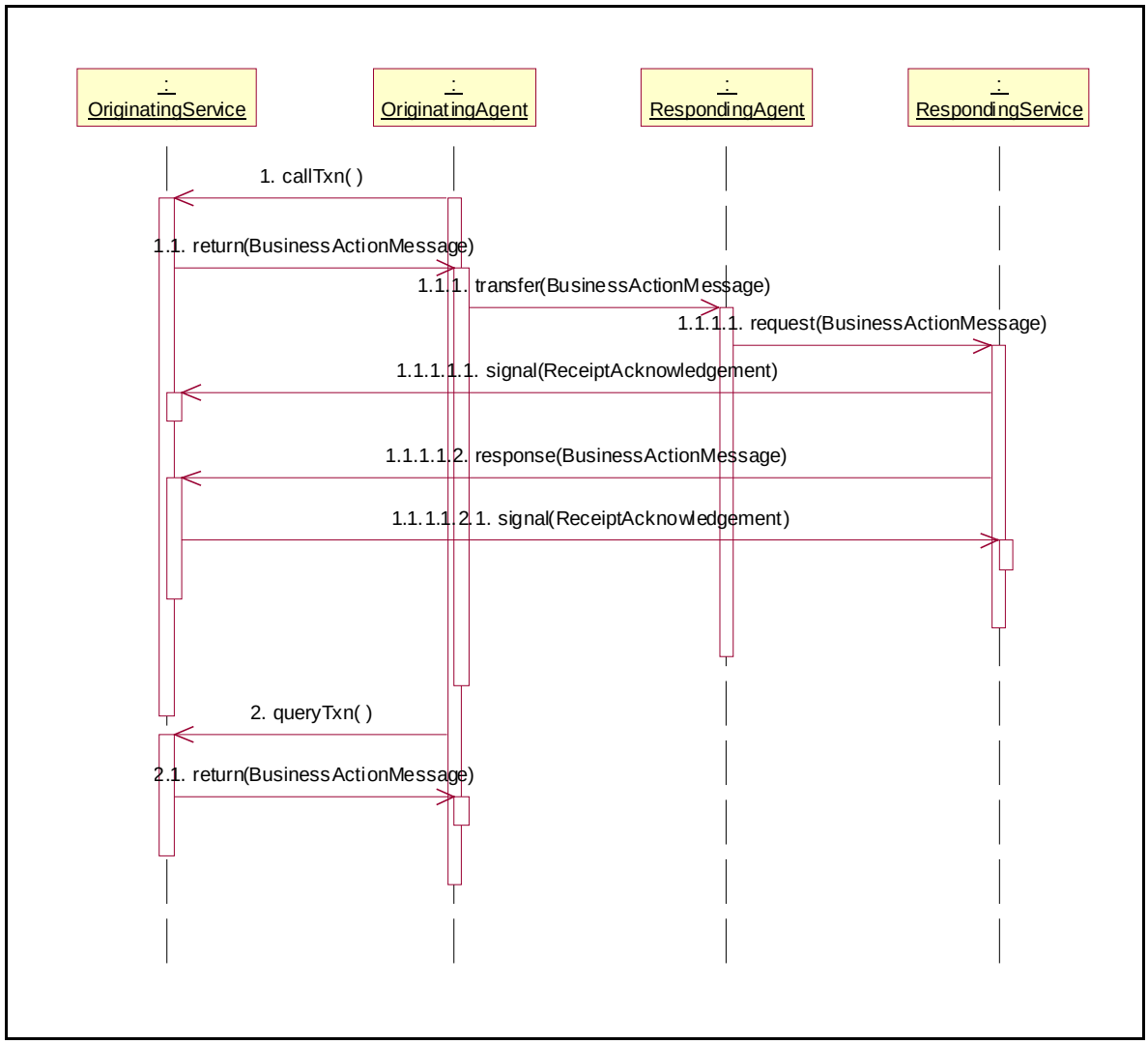

Figure 8-44 Service-Agent-Service interaction Pattern E

Information Distribution and Notification Activities

Figure 8-45 illustrates Interaction Pattern F.

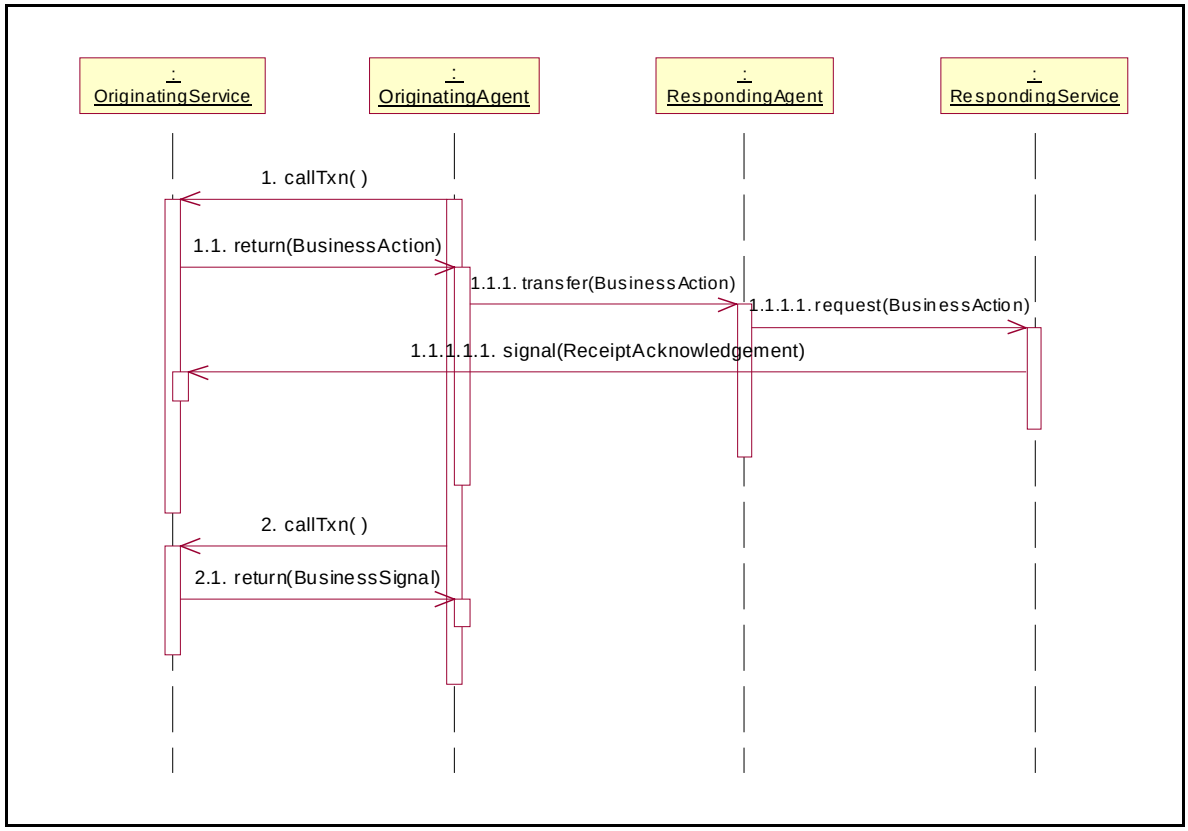

Figure 8-45 Service-Agent-Service Interaction Pattern F

8.4.5 Agent-Service-Agent

Figure 8-46 illustrates the Agent-Service-Agent business service interaction pattern used in the business transaction patterns of Section 8.3.

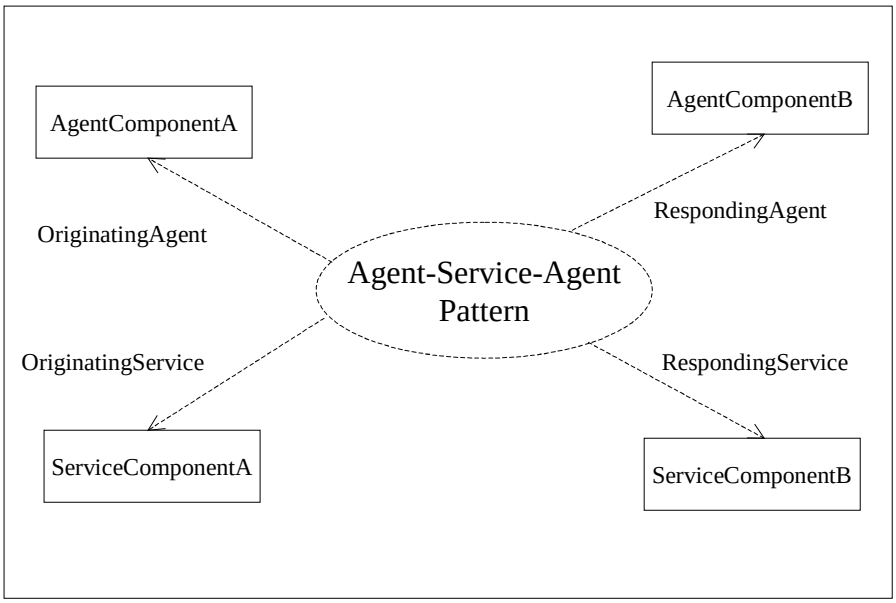

Figure 8-46 Agent-Service-Agent pattern

Figure 8-47 shows the class diagram for the Agent-Service-Agent pattern, and Figure 8-48 shows the class diagram for the Agent-Service-Agent unfolded from the base pattern.

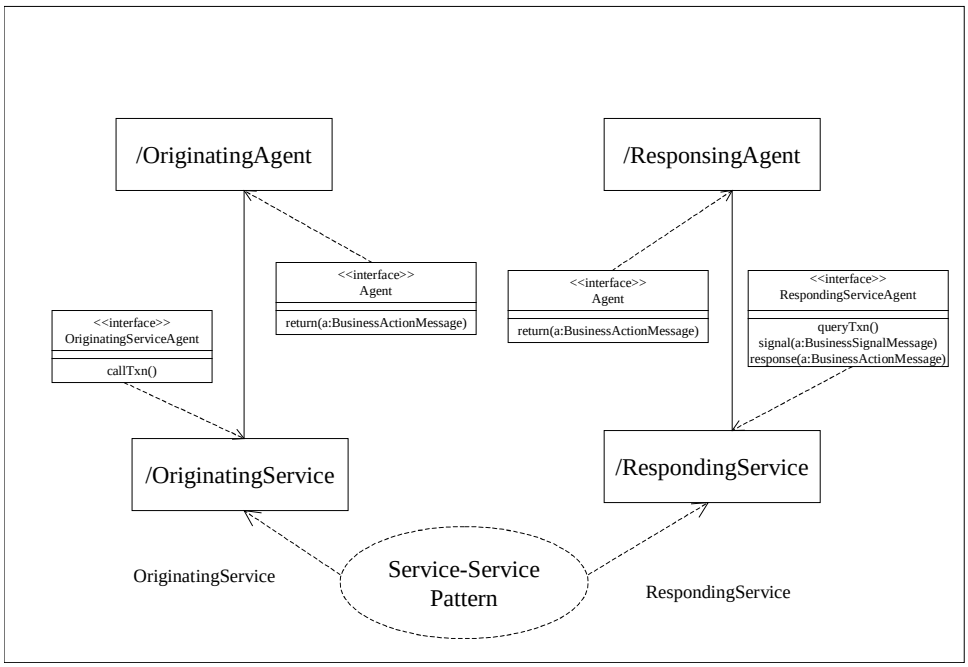

Figure 8-47 Agent-Service-Agent class diagram

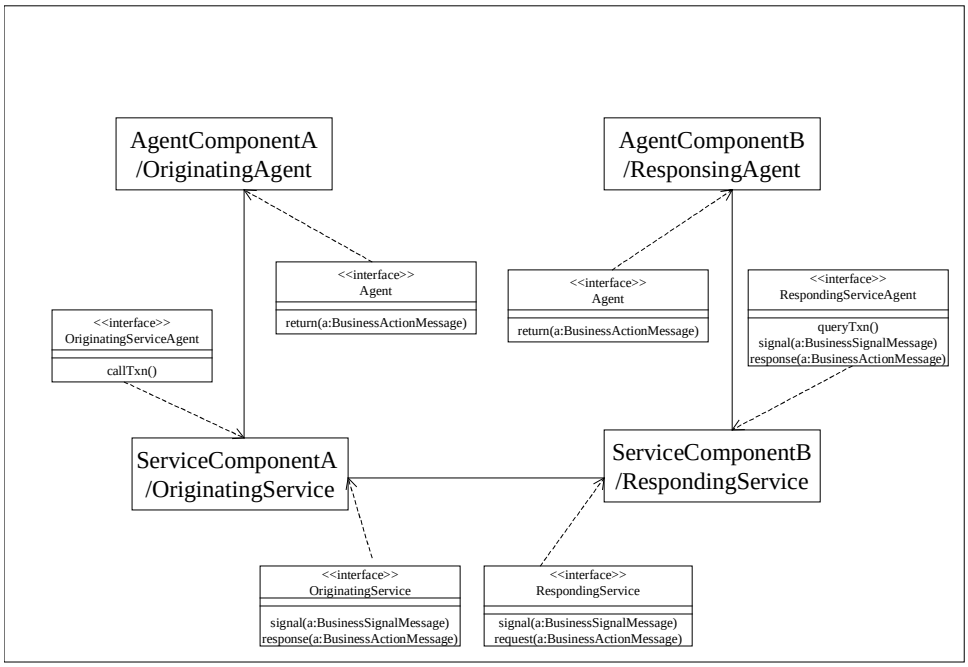

Figure 8-48 Agent-Service-Agent class diagram unfolded

There are five variations within the Agent-Service-Agent pattern:

1. Interaction Pattern A applies to the Business Transaction Pattern where time to perform equals time to acknowledge acceptance and there is no responding business document.
2. Interaction Pattern B also applies to the Business Transaction Pattern where time to perform equals time to acknowledge acceptance and a responding business document.
3. Interaction Pattern C also applies to the Business Transaction Pattern where time to perform is greater than time to acknowledge acceptance.
4. Interaction Pattern D applies to the Query/Response, Request/Response, and Request/Confirm Patterns.
5. Interaction Pattern E applies to the Information Distribution and Notification Patterns.

Business Transaction Activity

Figure 8-49, Figure 8-50, and Figure 8-51 illustrate Interaction Patterns A, B, and C respectively.

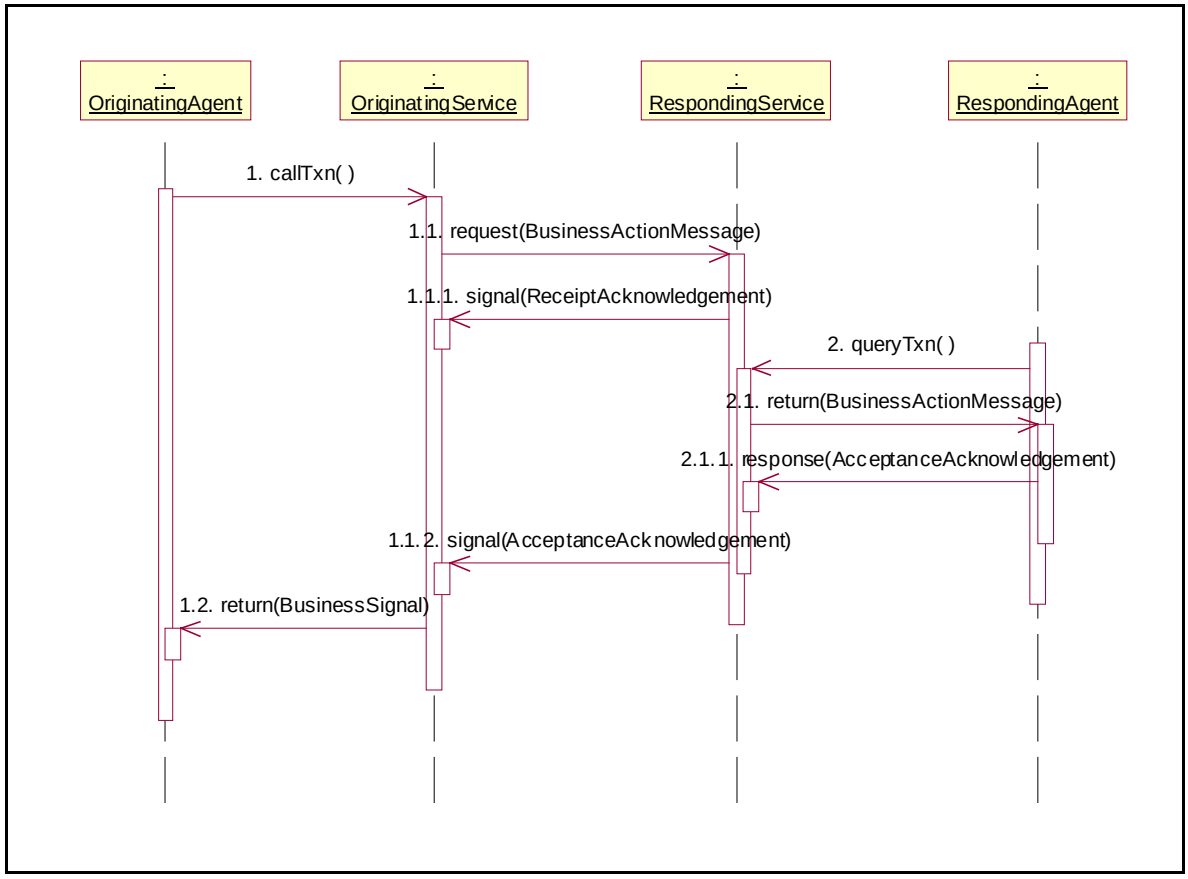

Figure 8-49 Agent-Service-Agent Interaction Pattern A

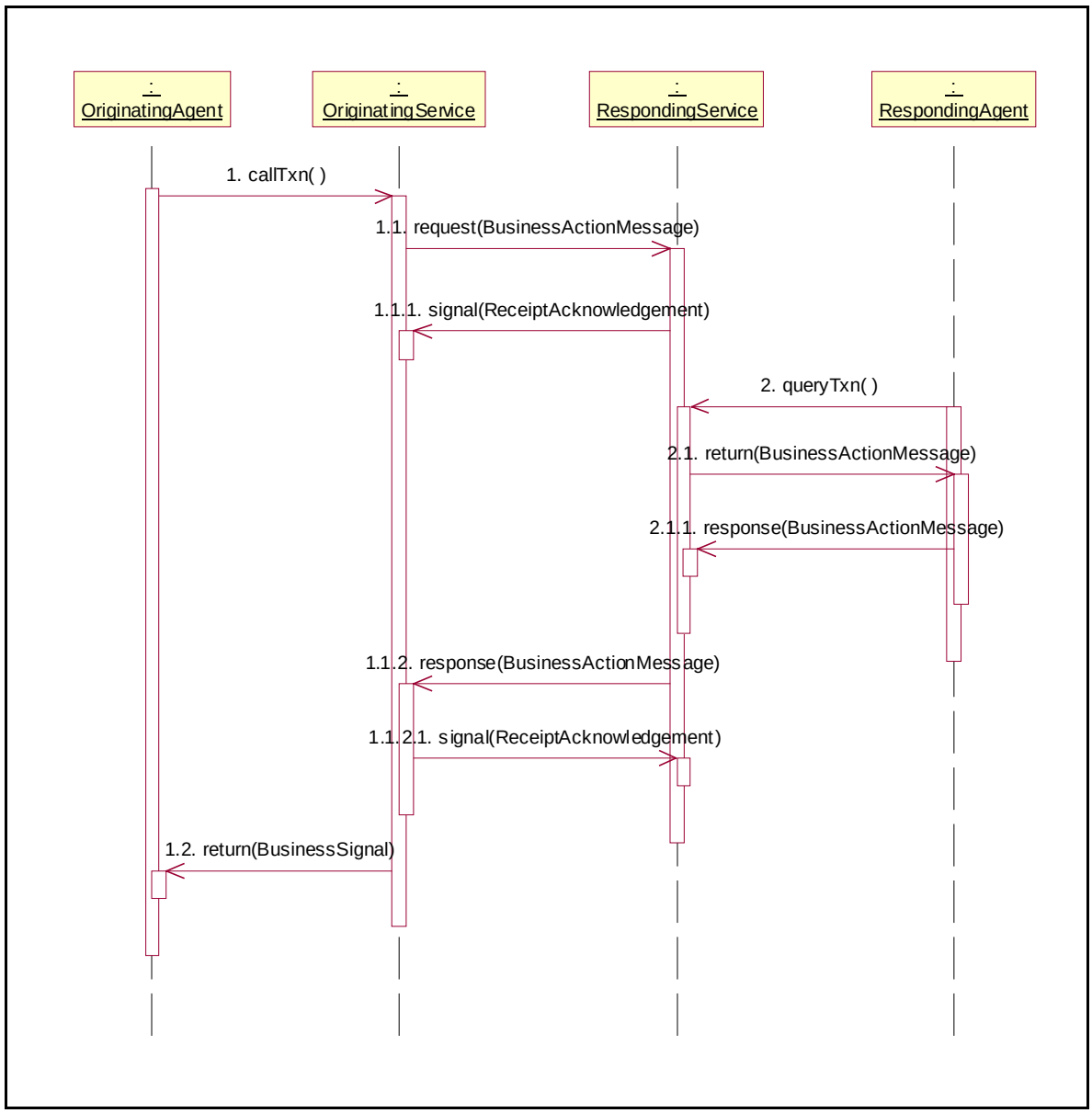

Figure 8-50 Agent-Service-Agent Interaction Pattern B

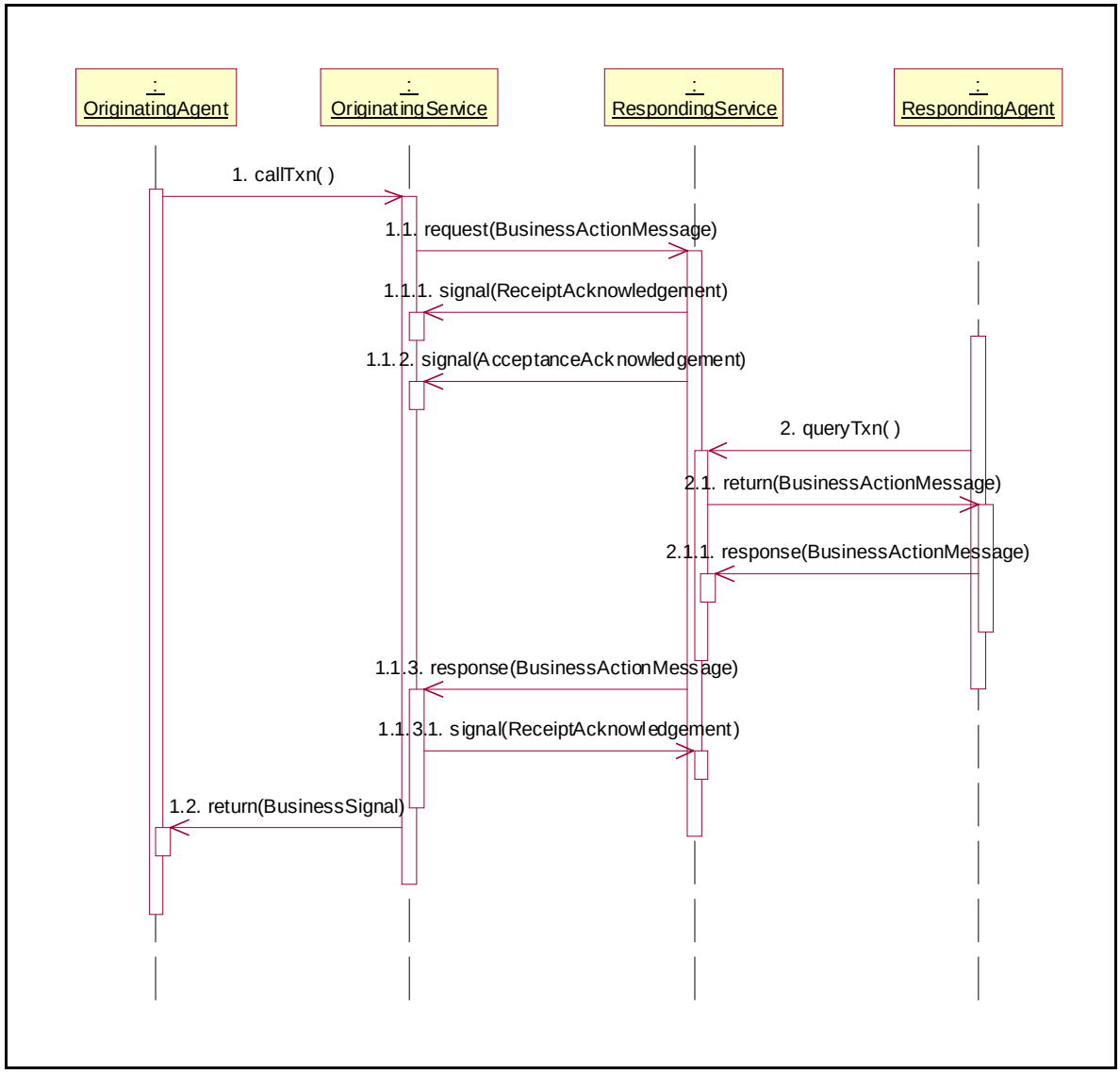

Figure 8-51 Agent-Service-Agent Interaction Pattern C

Query/Response, Request/Response, and Request/Confirm Activities

Figure 8-52 illustrates Interaction Pattern D.

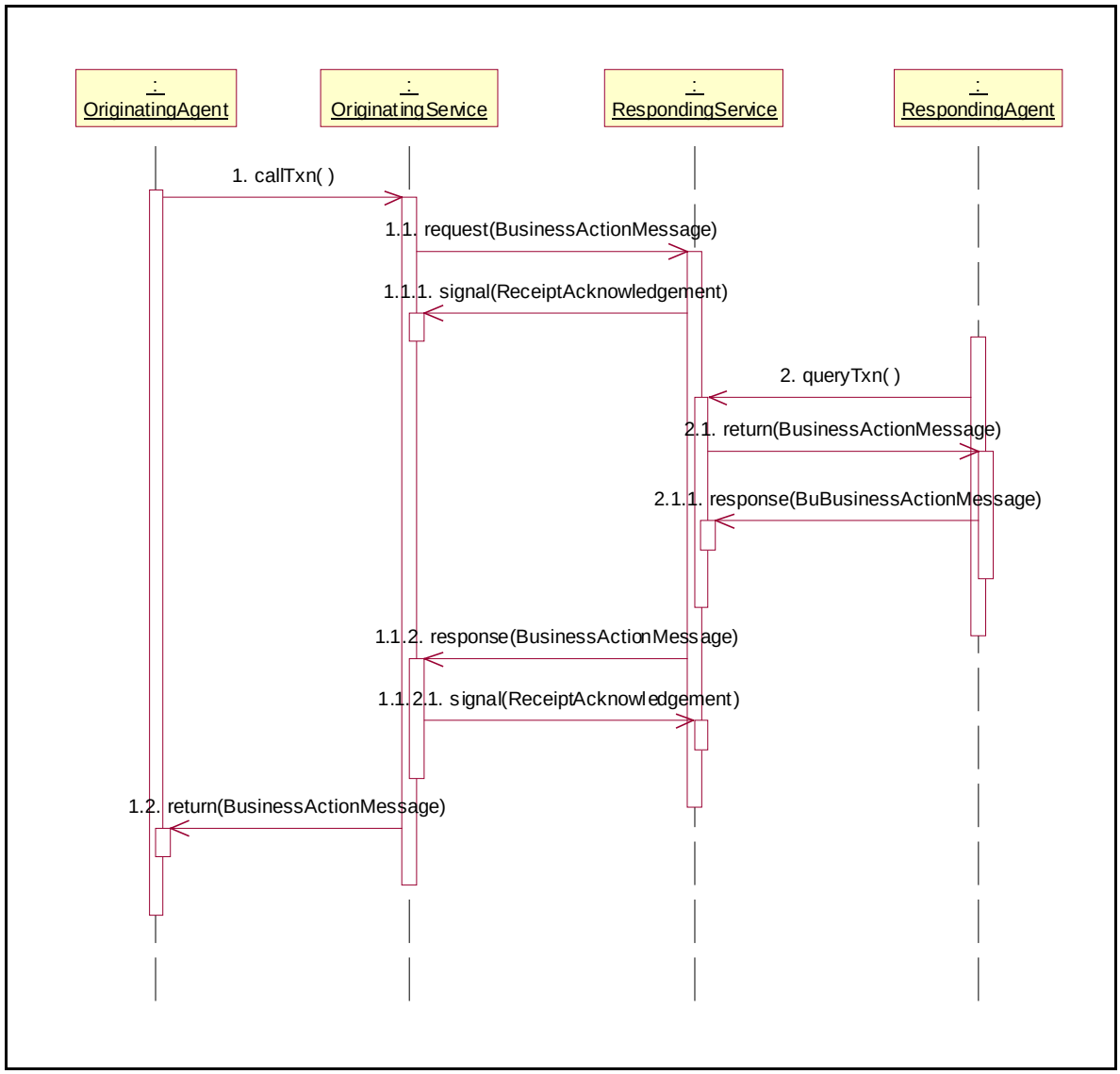

Figure 8-52 Agent-Service-Agent Interaction Pattern D

Information Distribution and Notification Activities

Figure 8-53 illustrates Interaction Pattern E.

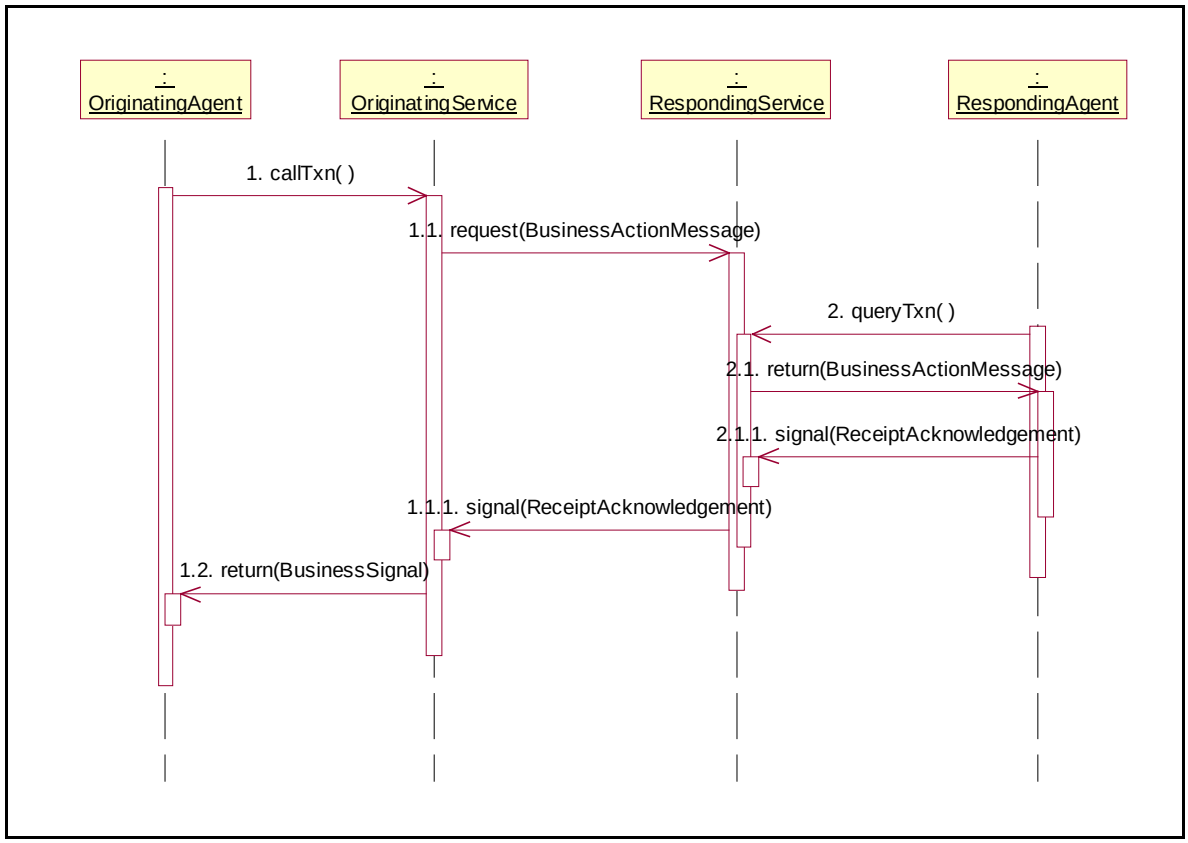

Figure 8-53 Agent-Service-Agent Interaction Pattern E

8.5 Business Information Structure Design Patterns

8.5.1 The Reference Design Pattern

Business entity containers can reference themselves and other entities by explicitly modeling the reference association as an entity with association properties. As shown in Figure 8-54, the reference association (*SubComponent*) should minimally contain cardinality properties and a name that has a semantic definition specifying the relationship between the related entities. This design pattern is useful for reusing common sub-entity representations between multiple entity containers.

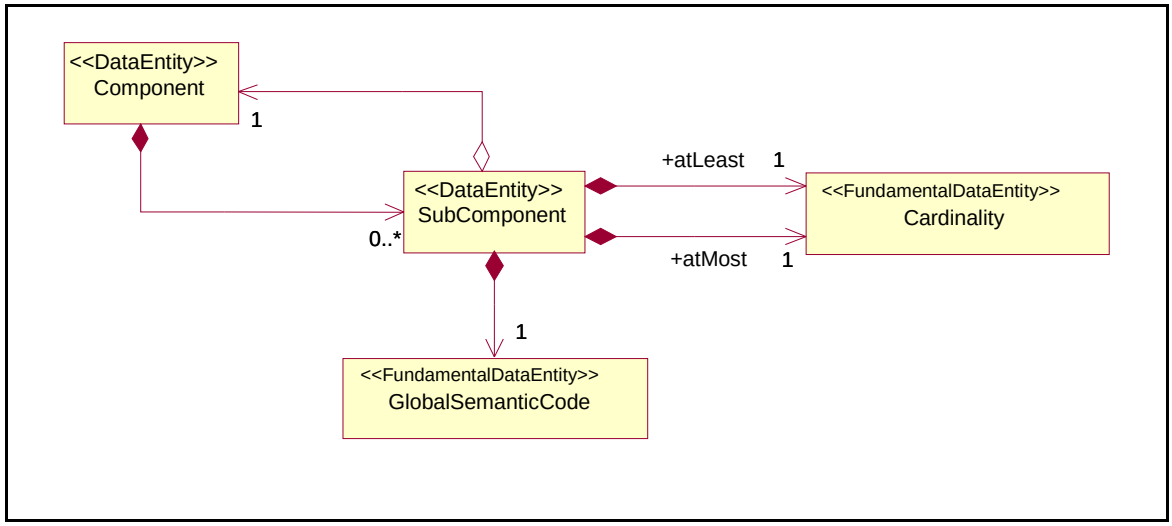

Figure 8-54 A Reference Relationship between Entities

Figure 8-54shows a *Component* entity containing zero or more *SubComponent* entities that contain a reference to the same *Component* entity. Entities cannot be self-referencing via a UML association directly i.e. the client and supplier of a UML association cannot be the same. The UML association between the *SubComponent* and *Composite* entities must be unidirectional.

Figure 8-55 illustrates the use of parenthesis in a message guideline document to specify a reference from one entity to another. The supplier of the UML association is enclosed in parenthesis.

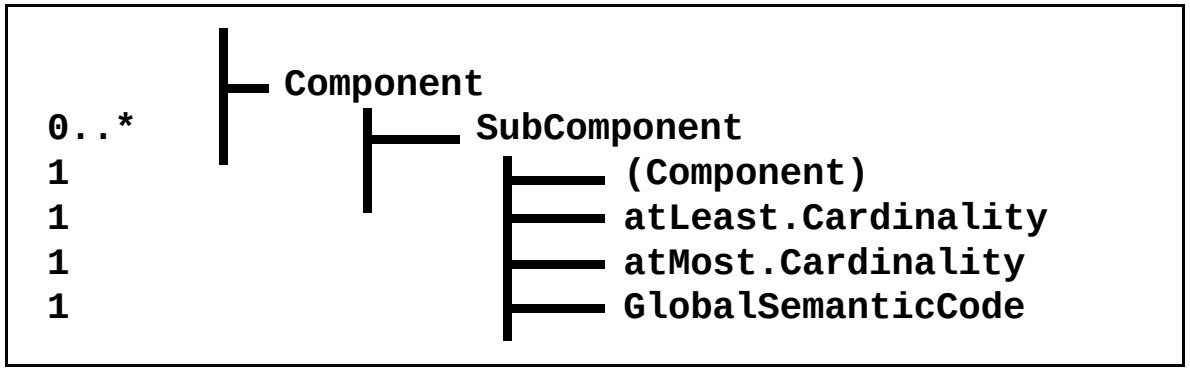

Figure 8-55 Illustration Showing Referenced Entity in Parenthesis

The XML document schema for this design pattern is shown inFigure 8-56. The *Component* element either comprises *SubComponent* sub-elements or it comprises the *Association* sub-element. The *Component* element also has an implied *ID* attribute that is only necessary when it is the target of a *reference* attribute value.

```
<!ELEMENT Component ( ( SubComponent* ) |
                       Association ) >
<!ATTLIST Component
          id ID #IMPLIED >
<!ELEMENT SubComponent ( Component,
                         atMost,
                         atLeast,
                         GlobalSemanticCode ) >
<!ELEMENT atMost ( Cardinality ) >
<!ELEMENT atLeast ( Cardinality ) >
<!ELEMENT GlobalSemanticCode ( PCDATA ) >
<!ELEMENT Cardinality ( PCDATA ) >
<!ELEMENT Association EMPTY >
<!ATTLIST Association
          reference IDREF #REQUIRED>
```

Figure 8-56 Document Schema for Reference Design Pattern

The *SubComponent* element contains a *Component* sub-element as its content along with the cardinality and semantic properties. The design does not permit a reference attribute to be specified for the *SubComponent* element, as the “type” of the reference is then lost. Specifying the *Component* as a sub-element of *SubComponent* and then allowing *Association* to be a sub-element of *Component* is one method of retaining the “type” of the association allowing better type-checking and a better method for specifying the meaning of the *SubComponent* entity.

Figure 8-57 illustrates the use of the design pattern for creating XML document

instances that comply with the DTD fragment in Figure 8-56. ***You will notice*** that the DTD permits other valid document instance construction, for example, the *Component* element with id 'PartA' could contain the *Association* sub-element and the *Component* sub-element of *SubComponent* could have an 'id' association. Both of these document instance fragments would, however, have no meaning with respect to the entity model in Figure 8-54 and the guideline in Figure 8-55.

This design specification holds when there is no requirement of a DTD to completely validate a document instance as in the Business Collaboration Framework. Documents must be valid with respect to a guideline that may contain business rules that constrain the structure and content of a document in a specific business process context as shown in Figure 8-57.

```
<Component id='partA'>
   <!– properties go here -->
</Component>

<Component>
   <SubComponent>
      <Component>
         <Association reference='partA' />
      </Component>
      <atLeast>
         <Cardinality>1</Cardinality>
      </atLeast>
      <atMost>
         <Cardinality>5</Cardinality>
      </atMost>
      <GlobalSemanticCode>Requires</GlobalSemanticCode>
   </SubComponent>
<Component>
```

Figure 8-57 Valid Reference Design Pattern Document Instance

Applications must ensure that the graph described by the ID-IDREF pairs do not recurse infinitely. A *reference* attribute value should therefore not equal the *id* attribute value of a containing *Component* element.

8.5.2 Query/Response Business Document Design Pattern

The query/response design pattern is useful for both querying business information and for specifying the structure of the response to the query. There are a number of approaches to designing query/response business documents.

1. The query and response are modeled as individual documents with fixed, independent structure.
2. The query is modeled as a constraint on a fixed structure that is used to return the response.
3. The query can be modeled as a constrained ‘template’ that must be ‘completed’ by a responding business partner.

The first approach is typical of Electronic Data Interchange (EDI) query/response message specifications. The second approach is typical of Structured Query Language (SQL) message specifications and the third approach is typical of symbolic programming languages such as Lisp or Prolog that implement unification. The BCF provides a design pattern for the third approach to query/response messages, as it is the most flexible approach to query/response message design where the query and response messages permit unlimited canonical data structures. The first two do not require a design pattern, as they are no different from standard business document specifications and are thus do not need a pattern.

Figure 8-58 illustrates a query/response data entity model. A product information query comprises zero or more query constraints and one product description. A product information response comprises zero (no results in query) or more product descriptions that match the query. A query constraint is an Object Constraint Language (OCL) expression that constraints the results returned in the query.

Specifying a template for the query results and placing constraints on the template by either filling in some of the template content or by constraining the content of the template using query constraints produces a product information query. Filling in the template in accordance with the already specified content and the constraints produces a product information response.

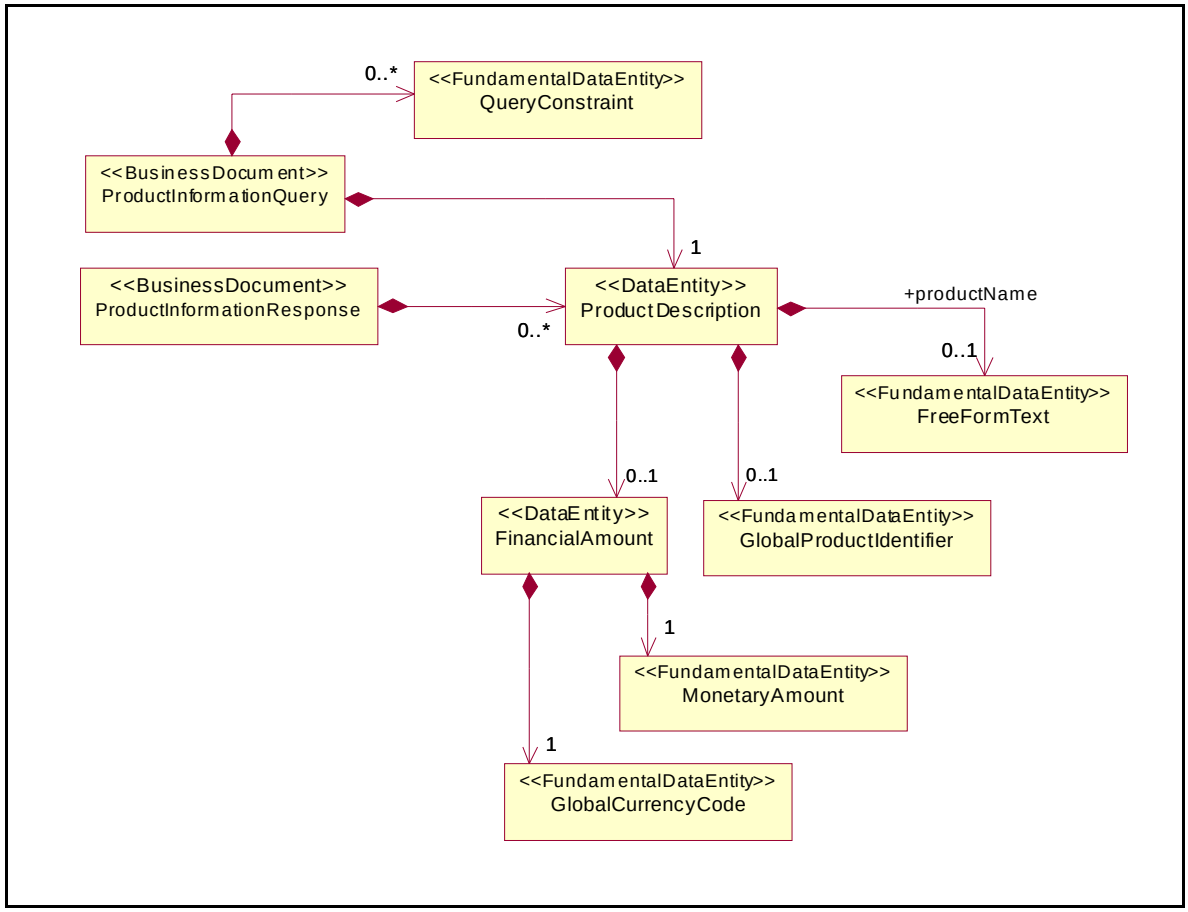

Figure 8-58 Query/Response Data Entity Model

The XML document schema for this design pattern is shown in Figure 8-59. The product description structure is used for both the query and response business documents. The template for the query is created from the product description schema.

```
<!ELEMENT ProductInformationQuery ( QueryConstraint*,
                                    ProductDescription ) >
<!ELEMENT ProductInformationResponse ( ProductDescription * ) >
<!ELEMENT QueryConstraint ( PCDATA ) >
<!ELEMENT ProductDescription ( productName?,
                               GlobalProductIdentifier?,
                               FinancialAmount? ) >
<!ELEMENT productName ( FreeFormText ) >
<!ELEMENT FreeFormText ( PCDATA ) >
<!ELEMENT GlobalProductIdentifier ( PCDATA ) >
<!ELEMENT FinancialAmount ( MonetaryAmount,
                            GlobalCurrencyCode ) >
<!ELEMENT MonetaryAmount ( PCDATA ) >
<!ELEMENT GlobalCurrencyCode ( PCDATA ) >
```

Figure 8-59 Query/Response Document Schema

An example product information query is shown in Figure 8-60. Information on a product with the name ‘aName’ is requested if the price of the product is less than 500 monetary units of any currency. The template requests the global product identifier, monetary amount and global currency code to be returned in the response.

```
<ProductInformationQuery>
   <QueryConstraint>
      ProductDescription.FinancialAmount.MonetaryAmount %lt; 500
   </QueryConstraint>
   <ProductDescription>
      <ProductName>aName</ProductName>
      <GlobalProductIdentifier></GlobalProductIdentifier>
      <FinancialAmount>
         <MonetaryAmount></MonetaryAmount>
         <GlobalCurrencyCode></GlobalCurrencyCode>
      </FinancialAmount>
   </ProductDescription>
</ProductInformationQuery>
```

Figure 8-60 An Example Product Information Query

An example product information query response is shown in Figure 8-61. The result of the query returns two product descriptions, their product identifiers and their cost.

```
<ProductInformationResponse>
   <ProductDescription>
      <ProductName>aName</ProductName>
      <GlobalProductIdentifier>3456789093</GlobalProductIdentifier>
      <FinancialAmount>
         <MonetaryAmount>100</MonetaryAmount>
         <GlobalCurrencyCode>USD</GlobalCurrencyCode>
      </FinancialAmount>
   </ProductDescription>

   <ProductDescription>
      <ProductName>aName</ProductName>
      <GlobalProductIdentifier>123456890</GlobalProductIdentifier>
      <FinancialAmount>
         <MonetaryAmount>50</MonetaryAmount>
         <GlobalCurrencyCode>SF</GlobalCurrencyCode>
      </FinancialAmount>
   </ProductDescription>
</ProductInformationResponse>
```

Figure 8-61 An Example Product Information Query Response

8.5.3 Disjunction Design Pattern

The disjunction design pattern is useful for representing business information entities that contain one or more of a number of disjunctive entities (the pattern is also useful to inherit common data properties). This pattern is not necessary for representations of zero or more of a number of disjunctive entities. Figure 8-62 illustrates a model that employs a disjunctive design pattern.

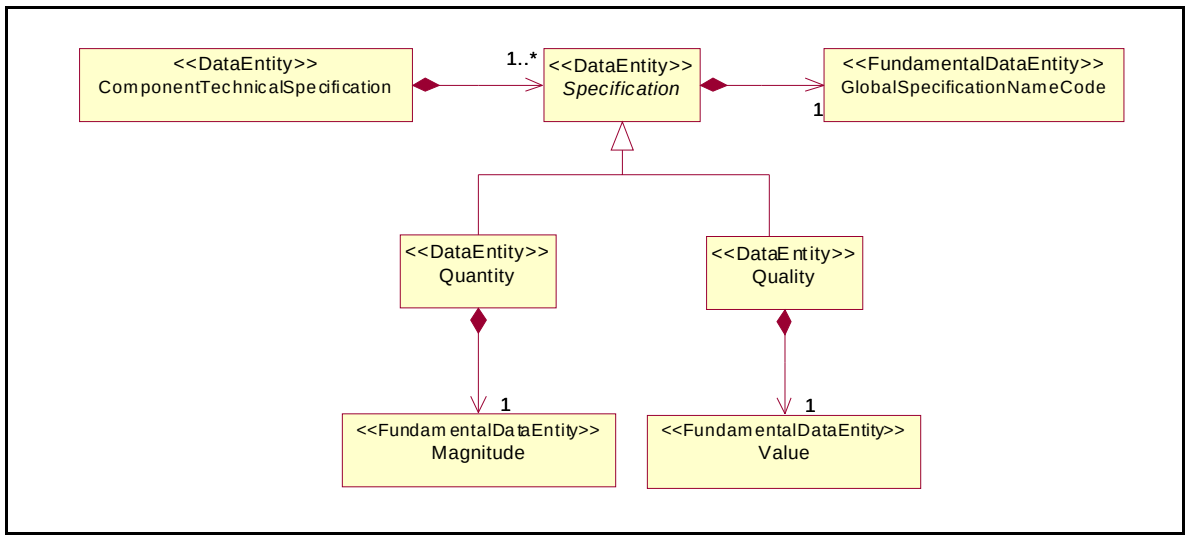

Figure 8-62 Disjunctive Data Entity Model

A component technical specification contains one or more specifications that are either quantities or qualities. Other representations of this specification allow either zero or more or two or more specification properties; now of which are meet the requirements of one or more specifications. Note that the specification data entity in Figure 8-62 is abstract (italicized class name). This prevents the data entity from being used as an object.

Figure 8-63 illustrates how the representation is shown in a message guideline document. The *Choice* node in the hierarchy shows the cardinality of one or more and the choice (disjunctive) nodes do not show any cardinality. The inherited *GlobalSpecificationNameCode* is repeated for each concrete class in the data entity model.

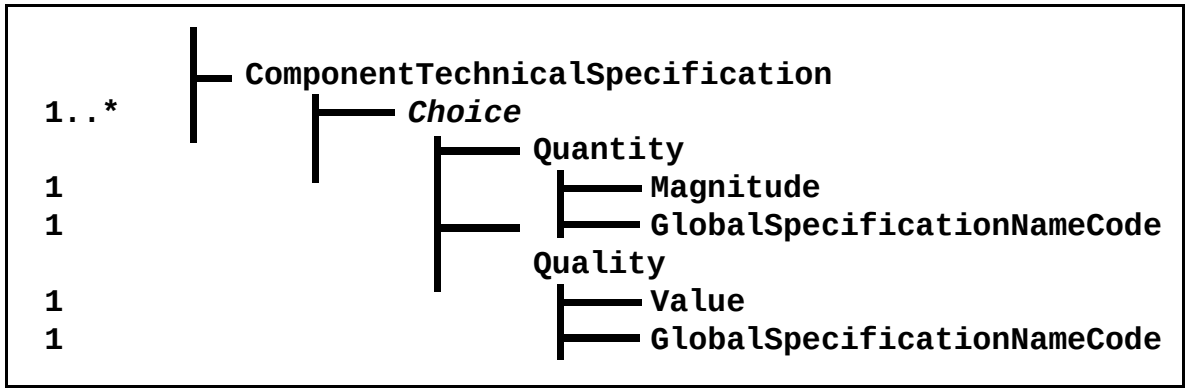

Figure 8-63 Disjunction Illustrated in a Message Guideline

The XML document schema for this design pattern is shown in Figure 8-64.

```
<!ELEMENT ComponentTechicalSpecification ( Quantity |
                                           Quality )+ >
<!ELEMENT Quantity ( Magnitude,
                     GlobalSpecificationNameCode ) >
<!ELEMENT Quality ( Value,
                    GlobalSpecificationNameCode ) >
<!ELEMENT Magnitude ( PCDATA ) >
<!ELEMENT Value ( PCDATA ) >
<!ELEMENT GlobalSpecificationNameCode ( PCDATA ) >
```

Figure 8-64 Disjunction Design Pattern Document Schema

A compliant XML document can provide one ore more occurrences of the quantity or quality specification properties.

8.5.4 Reification Design Pattern

The reification design pattern is useful for representing common business information entities that share a common design pattern but are verbose in their representation. Figure 8-65 illustrates an entity model for representing a manufacturer name and a product name.

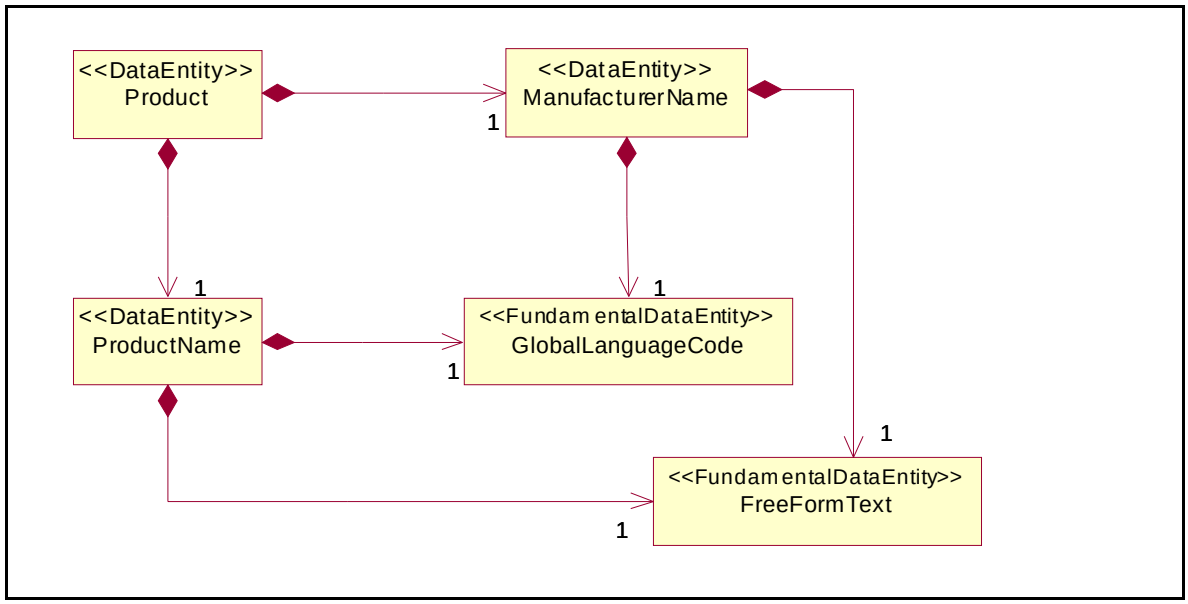

Figure 8-65 Illustration of a Free Form Text Entity

Each 'name' entity contains a free form text entity and a global language code. It is very verbose to specify these entities and relationships for each 'name' entity in a large entity model. Figure 8-66 illustrates how the *ManufacturerName* and the *ProductName* entities can be reified to property names if a design pattern always emits a global language code requirement for each free form text requirement.

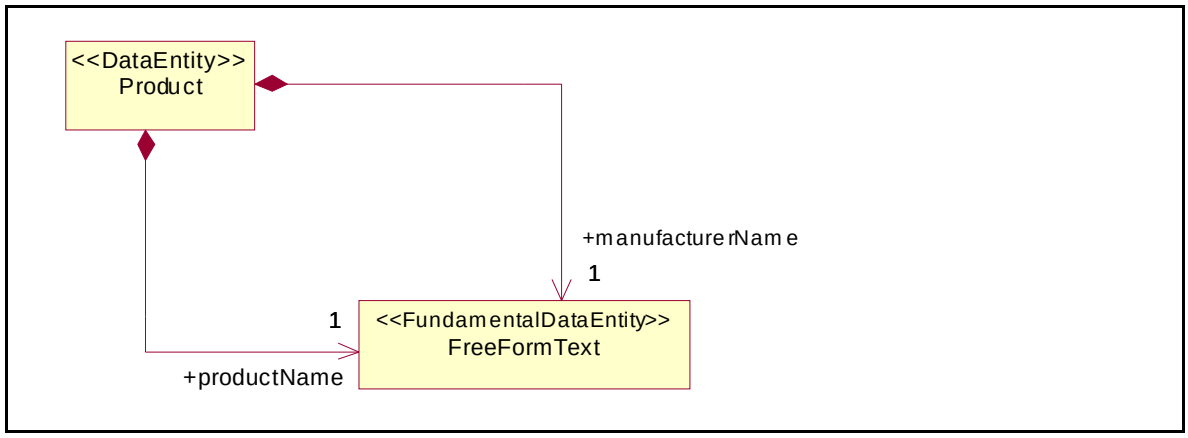

Figure 8-66 Illustration of Reified Data Entities

The XML document schema for this design pattern is shown in Figure 8-67.

```
<!ELEMENT Product ( manufacturerName,
                    productName ) >
<!ELEMENT manufacturerName ( FreeFormText ) >
<!ELEMENT productName ( FreeFormText ) >
<!ELEMENT FreeFormText ( PCDATA ) >
<!ATTLIST FreeFormText
          xml:lang CDATA #REQUIRED >
```

Figure 8-67 Reification Document Schema

The *xml:lang* attribute is added to each free form text element. Figure 8-67 illustrates the *xml:lang* attribute as CDATA and not as an enumerated option list as this could lead to very large files.

The BCF uses this design pattern to reify the language code for free form text and the physical unit of measure code for each quantitative data entity.

8.5.5 UML/XML Translation Design Pattern

The UML/XML DTD design pattern is useful for translating UML business document models into XML DTD document schema. It can be confusing, however, when the cardinality of data entities in a message guideline do not concur with the cardinality of XML DTD elements in a document schema. The reason for this discrepancy is that all the elements in a DTD are globally scoped. XML technology does provide tag syntax for namespace declaration but this can become verbose with deep element nesting. The design pattern thus chosen for UML to XML DTD conversion renders a DTD inadequate for validating a message with respect to a message guideline. Applications are therefore required to validate messages with respect to a guideline and not only with respect to a DTD.

Figure 8-68 illustrates an example data entity model where a *Document* entity comprises a *fromBusiness* and *toBusiness* declaration and a *Business* comprises zero or one *Address* entity.

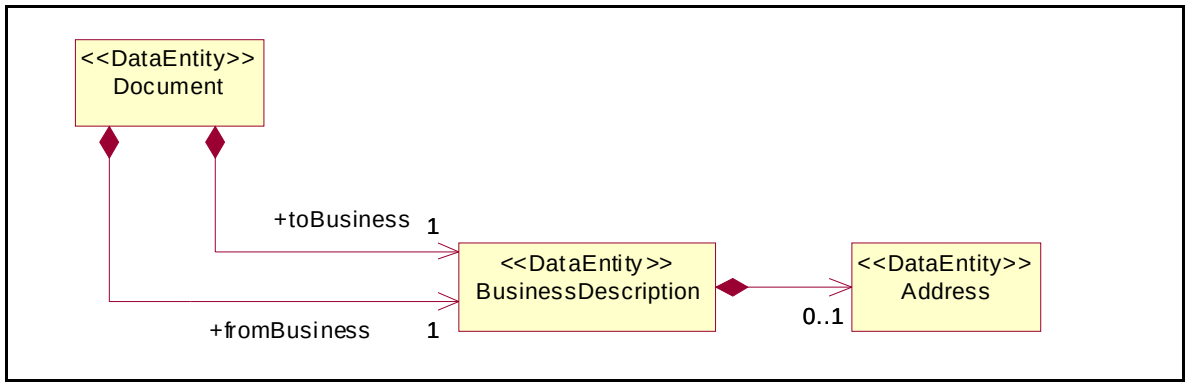

Figure 8-68 Illustration of a Data Entity Model

The UML model in Figure 8-68 is a 'network' model in that nodes in the network are interrelated in a network of associations. A message guideline, however, is a canonical hierarchy where each node in unique even though it is prototyped on a node in the UML network model. The algorithm to covert the network to a canonical hierarchy produces a graph shown in Figure 100 where each node in the graph is dependant on a prototypical node in the network.

The graph is a guideline that is modified to accurately represent the business data requirements. For example, Figure 8-69 illustrates that the *toBusiness* declaration of a *BusinessDescription* is not required to contain an *Address* (it needs to contain at least one Fundamental Data Entity but for the purposes of this illustration it is not necessary to show this). The *fromBusiness* declaration of a *BusinessDescription* is, however, required to contain an *Address*.

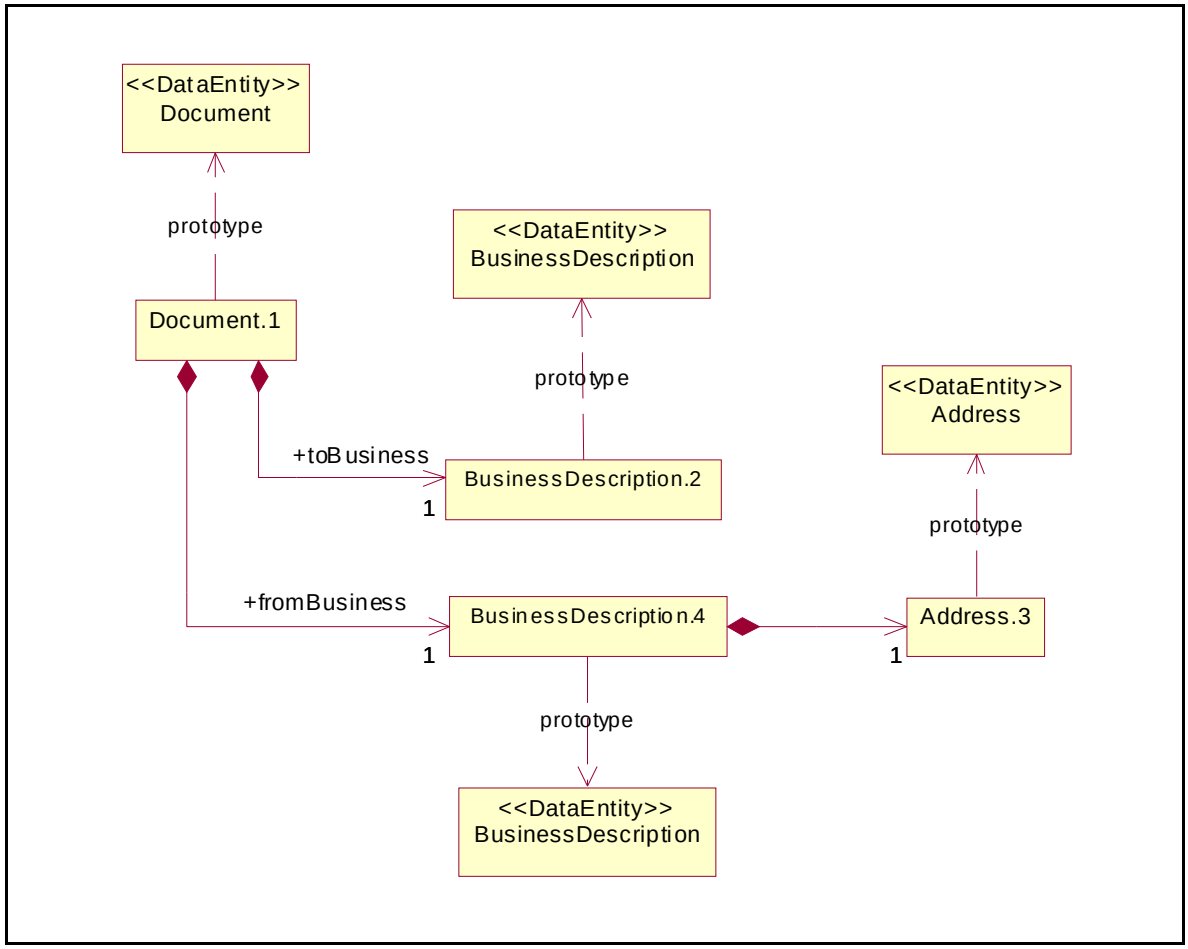

Figure 8-69 Illustration of a Canonical Hierarchy

The design of an algorithm that creates an XML DTD from the graph in Figure 8-69 needs to account for this conditional composition of the *BusinessDescription* node. It is possible to create an extremely large DTD where each node of the DTD is labeled with the name of the prototypical UML class and the unique identifier of the instance necessary to provide unique identity with respect to the nodes in a canonical hierarchy. The BCF design, however, does not take this route, as there is no requirement for complete message validation with respect to a DTD. Instead, a DTD as shown in Figure 8-70, is produced by the UML to XML DTD algorithm.

```
<!ELEMENT Document ( fromBusiness,
                     toBusiness ) >
<!ELEMENT fromBusiness ( BusinessDescription ) >
<!ELEMENT toBusiness ( BusinessDescription ) >
<!ELEMENT BusinessDescription ( Address? ) >
<!ELEMENT Address … >
```

Figure 8-70 Document Schema Example

The *BusinessDescription* element in Figure 8-70 specifies the *Address* sub-element as optional that seems in disagreement with the specification in Figure 8-69. What is more, the DTD permits zero sub-elements for *BusinessDescription* when provided as a sub-element to *toBusiness* and it permits one sub-element for *BusinessDescription* when provided as a sub-element to *fromBusiness*, both of which will be in disagreement with the graph specification in Figure 8-69.

8.5.6 Business Document Design Pattern

The following information is required in all business documents.

- Each business document must contain information that identifies the role, partner and business that is sending the business document. Each business document must also contain information that identifies the role, partner and business description that the document is going to. This information is similar to the information contained in the letterhead of a business document. Only the business identifier needs to be in the document as the identifier is the electronic equivalent of an address. Figure 8-71 illustrates the role descriptions in a business document.

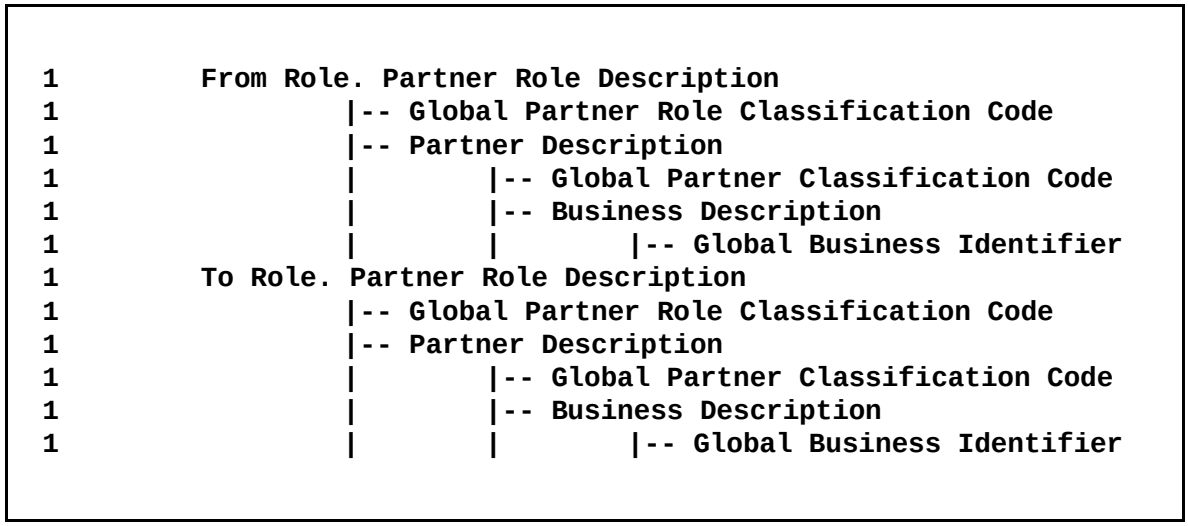

Figure 8-71 Role Specification in a Business Document

- The contact information of the initiating role must be included into the business document. The responding partner will be obligated to contact the initiating partner if there are errors in the received business document and a response (business signal or business document) cannot be delivered to the initiating partner, or there is no response specified. Figure 8-72 illustrates the contact information in a business document.

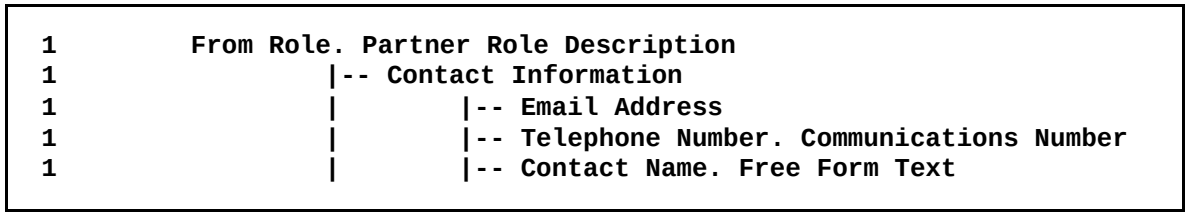

Figure 8-72 Contact Information in a Business Document

- The partner type, role type and supply chain code must be included as most conditional composition constraints are predicated on this information. Figure 8-73 illustrates supply chain specification in a business document.

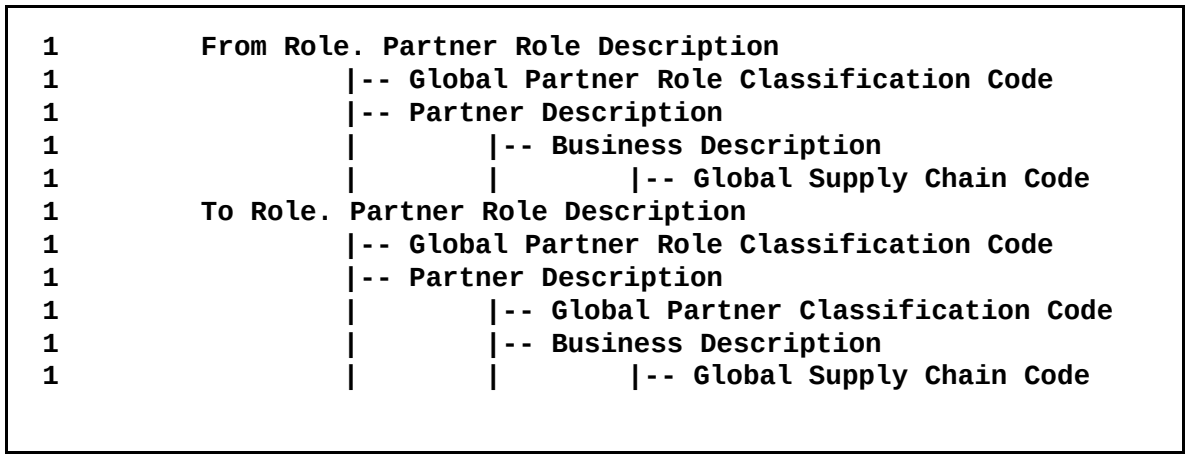

Figure 8-73 Supply Chain Specification in a Supply Chain

- Each document has an identifier. Each responding document must include the identifier of a requesting document. This allows documents to be tracked and reconciled. Figure 8-74 illustrates the specification of a document identifier in a business document.

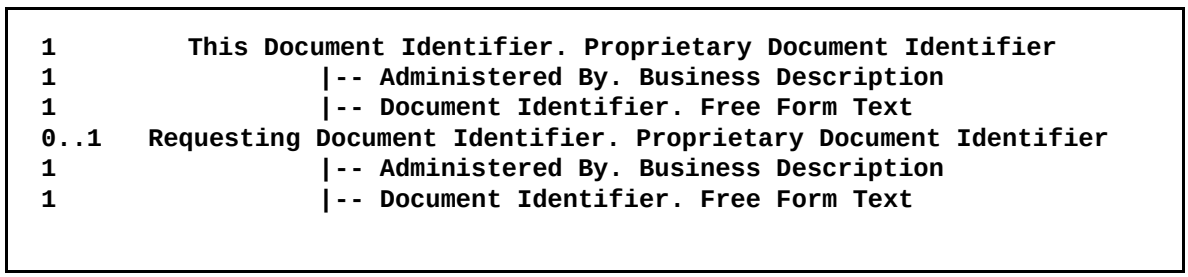

Figure 8-74 Document Identifier in a Business Document

- Each document must have a time and date stamp for auditing control. The date and time stamp is also used for legal purposes. Figure 8-75 illustrates the specification of a data and time stamp in a business document.

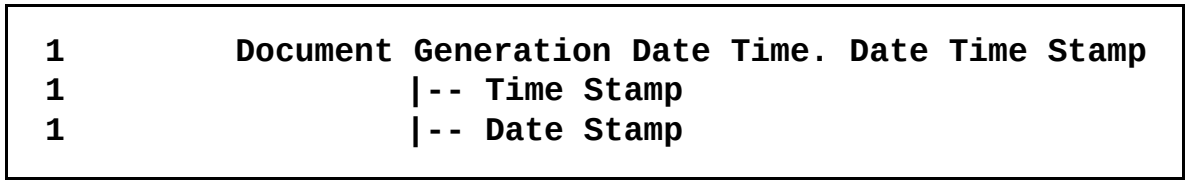

Figure 8-75 Data and Time Stamp in a Business Document

8.5.7 Request/Response Business Document Design Pattern

The request/response design pattern is useful for requesting a business partner to perform a business action and to return a response that meets given constraints. This design pattern differs from the query/response design pattern in two respects:

1. Semantically, a query/response transaction specifies an initiator's request for information that the responder has. A request/response transaction, however, asks the responder to perform an action and return a result of the action. This is an algorithmic response base on a prescriptive request.
2. Syntactically, a "Request" business document design pattern can comprise business rules that apply to the aggregation of the results in a response. Business applications responding to a request need to perform an additional processing step to apply these business rules to ***all*** the results of a query and not to ***each*** result of a query.

Figure 8-76 illustrates a request/response data entity model. A product availability request comprises zero or more query constraints, one or more business constraints and zero ore more product descriptions. The query constraints are constraints that must be met by each result returned in the response. The business constraints are the constraints that must be met by the entire response. Consider, for example, an initiator's product availability request for a maximum of 100 products of a particular type. The request for 100 products is a business constraint as the sum of all the product availability results must not be greater than 100. The type of product is a query constraint as each result must be the availability for the particular product type. A responding business partner may have less than 100 products and the partner may have more than 100 products in each of a number of locations. They therefore are required to perform a business action that reasons about how they will respond to such a request for availability. This may require some planning or optimisation algorithm to provide the response.

A product availability response comprises zero or more product availability results that match both the query constraint and the business constraint. A query constraint is an Object Constraint Language (OCL) expression that constraints each result returned in the response. A business constraint is an Object Constraint Language (OCL) expression that constraints the response.

Specifying a template for the response results and placing constraints on the template by either filling in some of the template content or by constraining the content of the template using query constraints produces a product availability query. Filling in the template in accordance with the already specified content, the query constraints and the business constraints produces a product availability response.

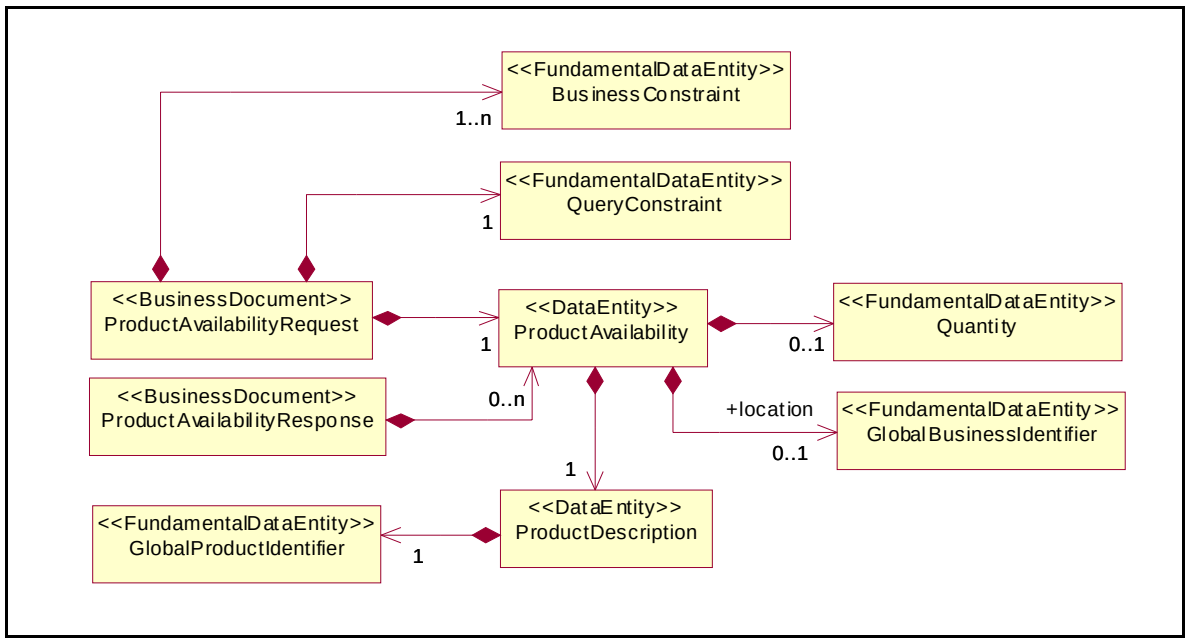

Figure 8-76 Request/Response Data Entity Model

The XML document schema for this design pattern is shown in Figure 8-77. The product availability structure is used for both the request and response business documents. The template for the request is created from the product availability schema.

```
<!ELEMENT ProductAvailabilityRequest ( BusinessConstraint+,
                                       QueryConstraint*,
                                        ProductAvailability ) >
<!ELEMENT ProductAvailabilityResponse ( ProductAvailability * ) >
<!ELEMENT QueryConstraint ( PCDATA ) >
<!Element BusinessContraint ( PCDATA ) >
<!Element ProductAvailability ( ProductDescription,
                                Quantity?,
                                GlobalBusinessIdentifier? ) >
<!ELEMENT ProductDescription ( GlobalProductIdentifier ) >
<!ELEMENT GlobalProductIdentifier ( PCDATA ) >
<!ELEMENT Quantity ( PCDATA ) >
```

Figure 8-77 Request/Response Document Schema

An example product availability request is shown in Figure 8-78. Availability on a product with the global product identifier is requested. The template requests the global product identifier, availability and locations to be returned in the response.

```
<ProductAvailabilityRequest>
   <QueryConstraint>
      ProductAvailability.ProductDescription.GlobalProductIdentifier = 123456789
   </QueryConstraint>
   <BusinessConstraint>
        (this->collect(ProductAvailability.Quantity))->sum <= 100
   </BusinessConstraint>
   <ProductAvailability>
      <ProductDescription>
         <GlobalProductIdentifier></GlobalProductIdentifier>
      </ProductDescription>
      <Quantity></Quantity>
      <Location></Location>
   </ProductAvailability>
</ProductAvailabilityRequest>
```

Figure 8-78 An Example Product Availability Request

An example product availability request response is shown in Figure 8-79. The result of the request returns two product availability results, their product identifiers and the location at which they are available. Note that the total number of available products is 100 and that the number of available products at each location is less than 100. [A query/response design pattern cannot express this requirement].

```
<ProductAvailabilityResponse>
   <ProductAvailability>
      <ProductDescription>
         <GlobalProductIdentifier>123456789</GlobalProductIdentifier>
      </ProductDescription>
      <Quantity>40</Quantity>
      <location>
         <GlobalBusinessIdentifier>987654321</GlobalBusinessIdentifier>
      </location>
   </ProductAvailability>

   <ProductAvailability>
      <ProductDescription>
         <GlobalProductIdentifier>123456789</GlobalProductIdentifier>
      </ProductDescription>
      <Quantity>60</Quantity>
      <location>
         <GlobalBusinessIdentifier>654987321</GlobalBusinessIdentifier>
      </location>
   </ProductAvailability>
</ProductAvailabilityResponse>
```

Figure 8-79 An Example Product Availability Response