TECHNICAL WHITE PAPER

EDIFECS

THE DNA OF B2B

E-business Collaboration Design Patterns

Version 2

27 October 2000

Contents

Examples

Figures

Tables

Executive Summary

The e-business collaboration modeling metamodel (see BCF#7 document “E-business Collaboration Modeling Metamodel”) provides a framework for constructing e-business collaboration model specifications. This document describes the design patterns that apply the metamodel to represent specific business process scenarios.

Preface

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

Purpose of the Document

The purpose of this document is to describe business collaboration design patterns that are applications of the e-business collaboration modeling metamodel.

Intended Audience

This document is written for business process modelers who reuse design patterns during model construction and for system implementers that need to understand how these design patterns are implemented.

Prerequisites

It is assumed that the audience is familiar with or has knowledge of the following technologies, techniques and documents:

- Business process modeling techniques and principles
- The Unified Modeling Language (UML) syntax and semantics
- The "E-business Collaboration Modeling Metamodel" (BCF#7)

Scope of the Document

This document includes design patterns for both business information and business information flow representations.

Style Conventions

This document uses typographical and language conventions to convey specific meanings.

Typographical Conventions

The use of a ***bold/italic font*** indicates a UML or business collaboration metamodel entity name.

Language Conventions

This specification adopts the conventions expressed in the IETF's[1] RFC 2119 "Key Words for Use in RFCs to Indicate Requirement Levels." The key words "MUST," "MUST NOT," "REQUIRED," "SHALL," "SHALL NOT," "SHOULD," "SHOULD NOT," "RECOMMENDED," "MAY," and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

Authors

Arthur Greef, Gary Ham, Jim Clark, John Yunker, Mark Smith, and Tony Weida.

Acknowledgments

Edifecs[2]: Edifecs is administering the creation of the Business Collaboration Framework (BCF). The BCF is a collection of documents that prescribe the policy, architecture and specifications for executing business collaborations for e-business.

Contacts

Jim Clark—Edifecs Business Collaboration Services (BCS) Director of Industry Solutions

JamesC@edifecs.com

Tony Weida—Edifecs BCS Enterprise Architect

weida@edifecs.com

Copyright

[1] http://www.ietf.org/
[2] http://www.edifecs.com/

1 Introduction

The e-business collaboration modeling metamodel provides a language and grammar for constructing business collaboration models. Design patterns are applications of the metamodel to common business process representations. Representations capture common structure and semantics applicable to specific business process domains.

This document describes the following design patterns:

1. Business information structure:
    a. **Reference** design pattern—used to reference business information descriptions to describe aggregate business information containers.
    b. **Query/Response** business document design pattern—used for both querying business information and for specifying the structure of the response.
    c. **Disjunction** design pattern—used for representing business information entities that contain one or more of a disjunctive entity.
    d. **Reification** design pattern—used for representing common business information entities.
    e. **UML/XML DTD translation** design pattern—used for translating UML business document models into XML DTD document schema.
    f. **Business document** design pattern—used for exchanging messages that can be interpreted as “legal writings” with respect to commercial law.
    g. **Request/Response** business document design pattern—used for requesting complex query results and for specifying the structure of the response.
2. Business information flow:
    a. **Commercial transaction** diagram design pattern—used for specifying commercial transactions using the UML activity diagram notation.
    b. **Business collaboration protocol** design pattern—used for specifying business collaborations using the UML activity diagram notation.
    c. **Network component interaction diagram** design pattern—used for specifying network component interactions using the UML sequence diagram notation.

2 Business Information Structure Design Patterns

2.1 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 2.1, 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 2.1 shows 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.

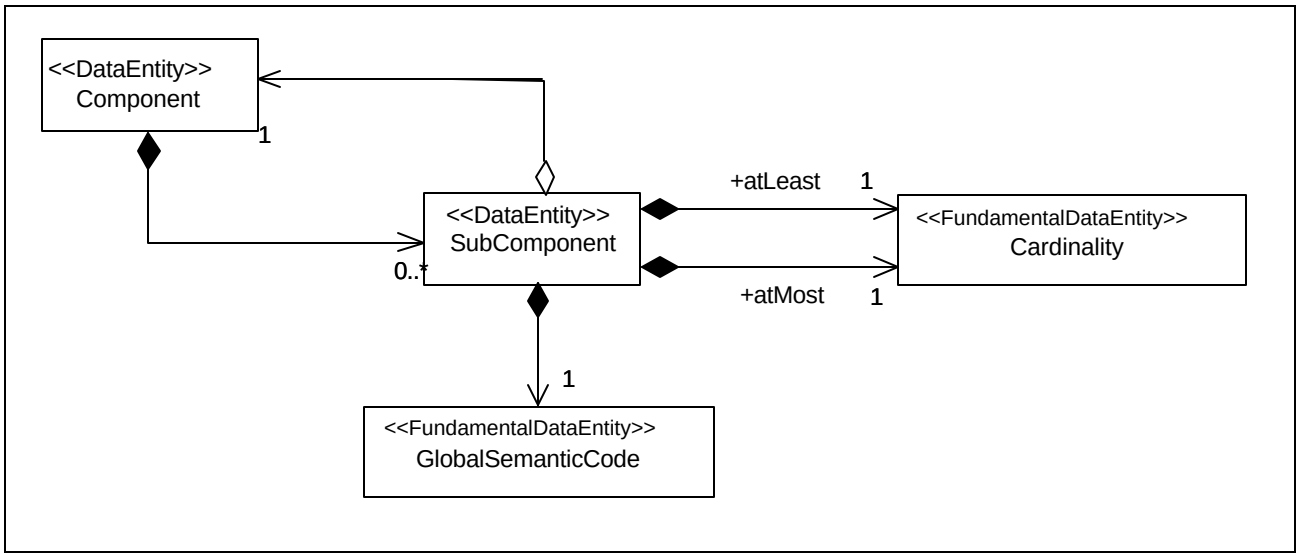

Figure 2.1 A Reference Relationship between Entities

Figure 2.2 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.

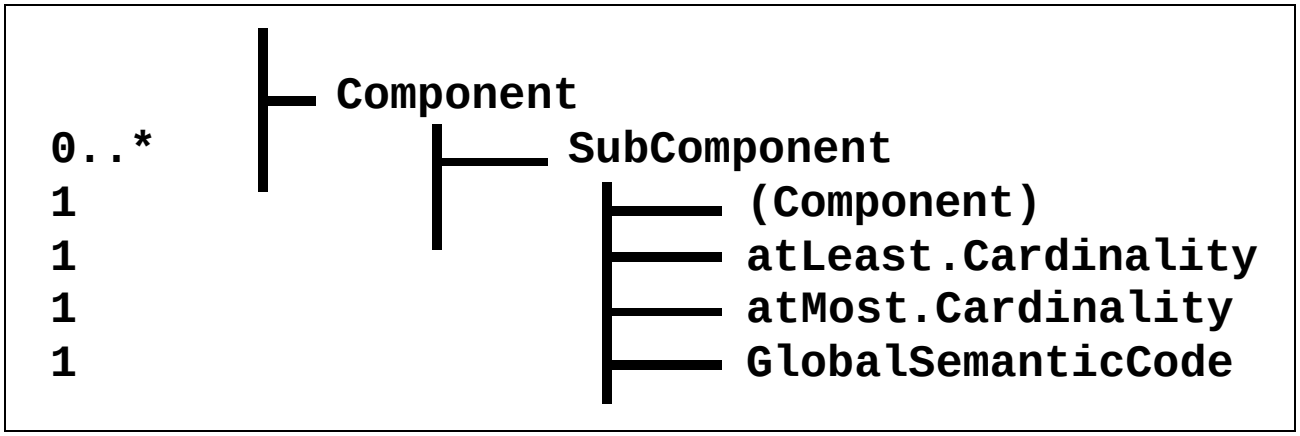

Figure 2.2 Illustration Showing Referenced Entity in Parenthesis

The XML document schema for this design pattern is shown in Example 2.1. 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>
```

Example 2.1 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.

Example 2.2 illustrates the use of the design pattern for creating XML document instances that comply with the DTD fragment in Example 2.1. 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 2.1 and the guideline in Figure 2.2

```
<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>
```

Example 2.2 Valid Reference Design Pattern Document Instance

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.

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.

2.2 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[3] 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.

[3] LISP, Prolog, etc.

Figure 2.3 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[4] (OCL) expression that restrains the results returned in the query.

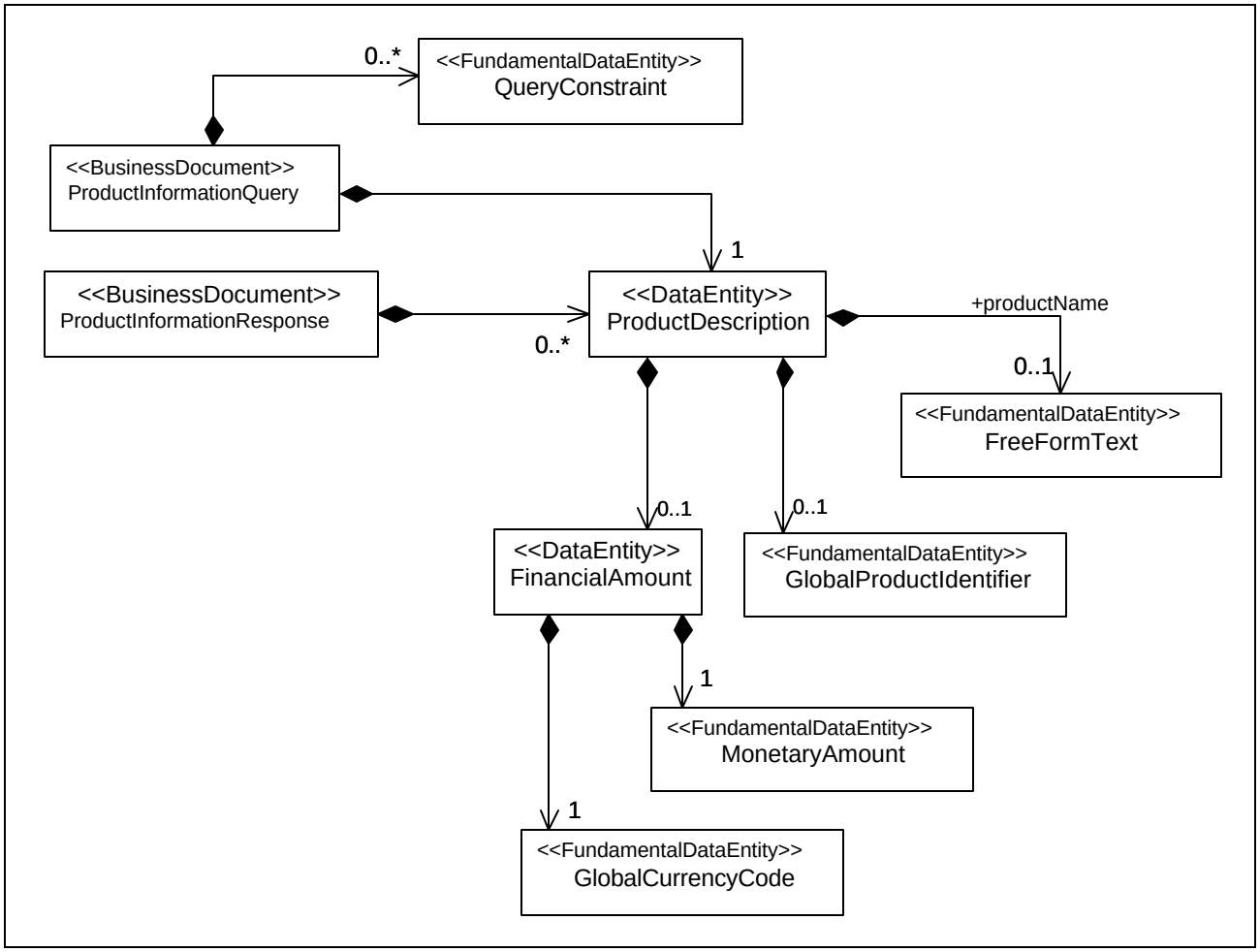

Figure 2.3 Query/Response Data Entity Model

Specifying a template for the query results and placing constraints on the template by either filling in some of the template content or by confining 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.

[4] Refer to http://www.omg.org for standard specification.

The XML document schema for this design pattern is shown in Example 2.3. 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 ) >
```

Example 2.3 Query/Response Document Schema

An example product information query is shown in Example 2.4. Information on a product with the name “aName” is requested if the price of the product is less than (%lt;) 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>
```

Example 2.4 An Example Product Information Query

An example product information query response is shown in Example 2.5. The result of the query returns two product descriptions with their product identifiers and costs.

```
<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>
```

Example 2.5 An Example Product Information Query Response

2.3 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 2.4 illustrates a model that employs a disjunctive design pattern.

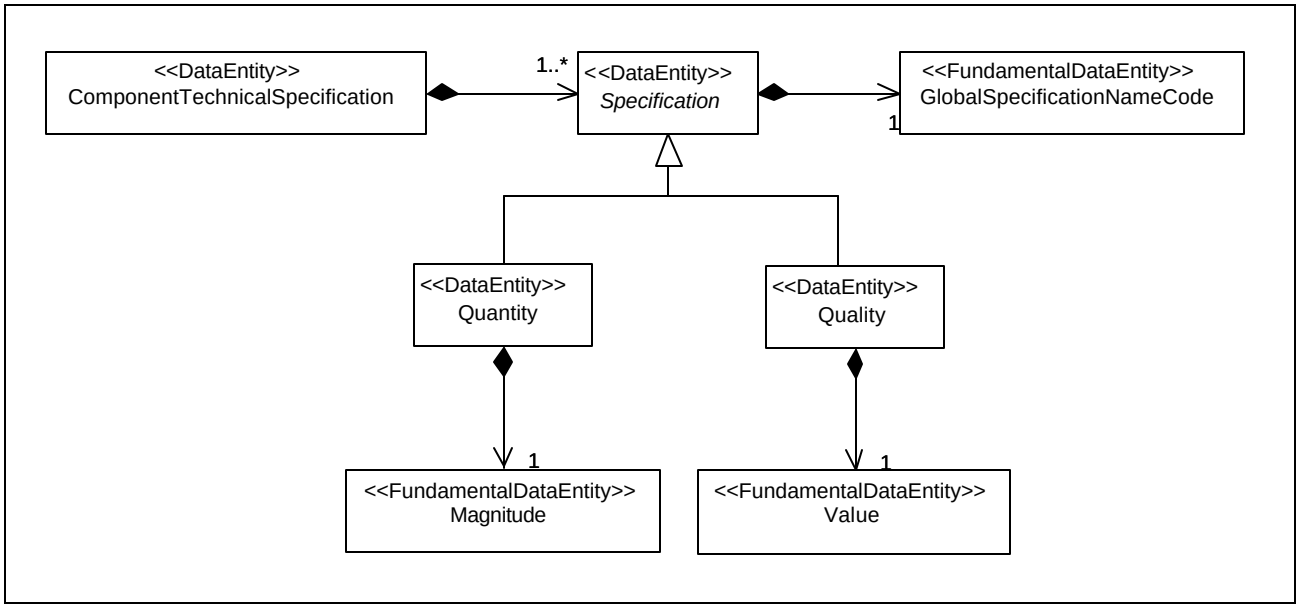

Figure 2.4 Disjunctive Data Entity Model

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; none of which meet the requirements of one or more specifications. Note that the specification data entity in Figure 2.4 is abstract (italicized class name). This prevents the data entity from being used as an object.

Figure 2.5 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.

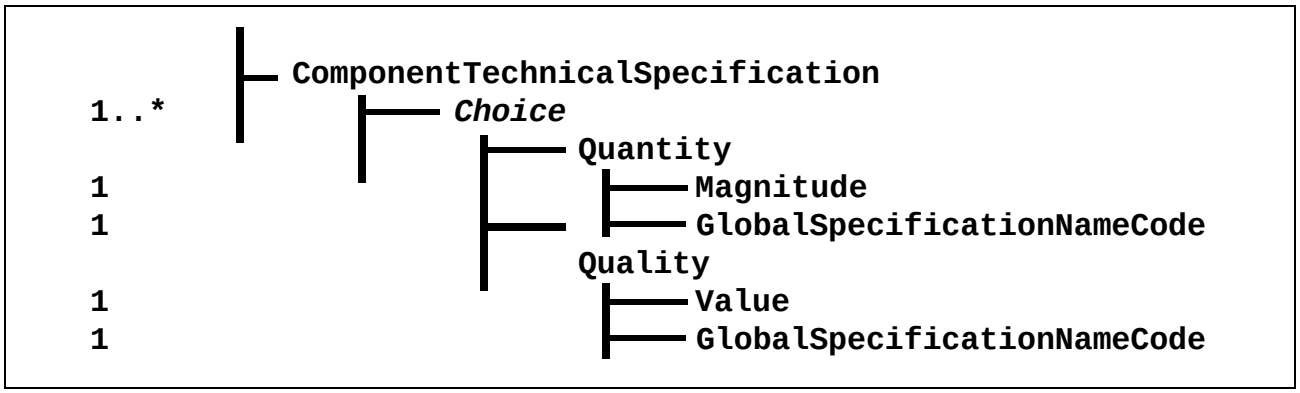

Figure 2.5 Disjunction Illustrated in a Message Guideline

The XML document schema for this design pattern is shown in Example 2.6.

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

Example 2.6 Disjunction Design Pattern Document Schema

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

2.4 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 2.6 illustrates an entity model for representing a manufacturer name and a product name.

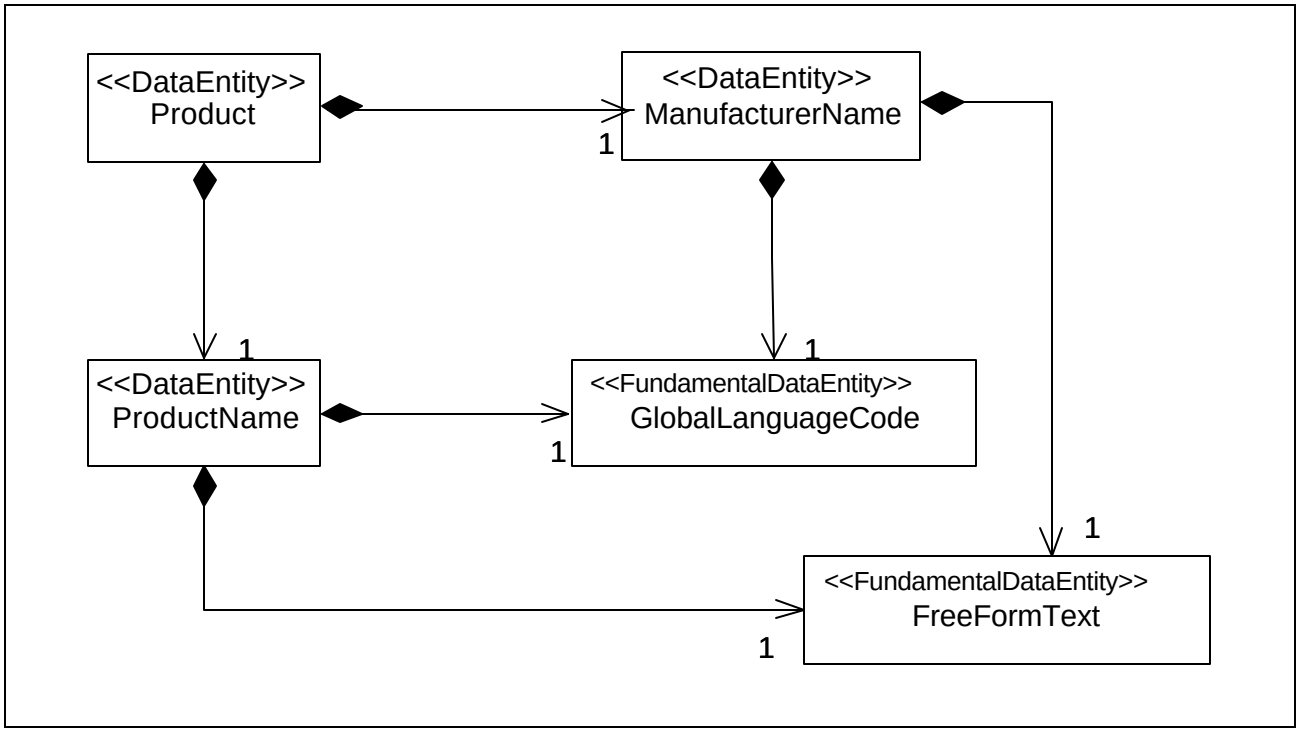

Figure 2.6 Illustration of a Free Form Text Entity

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 2.7 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.

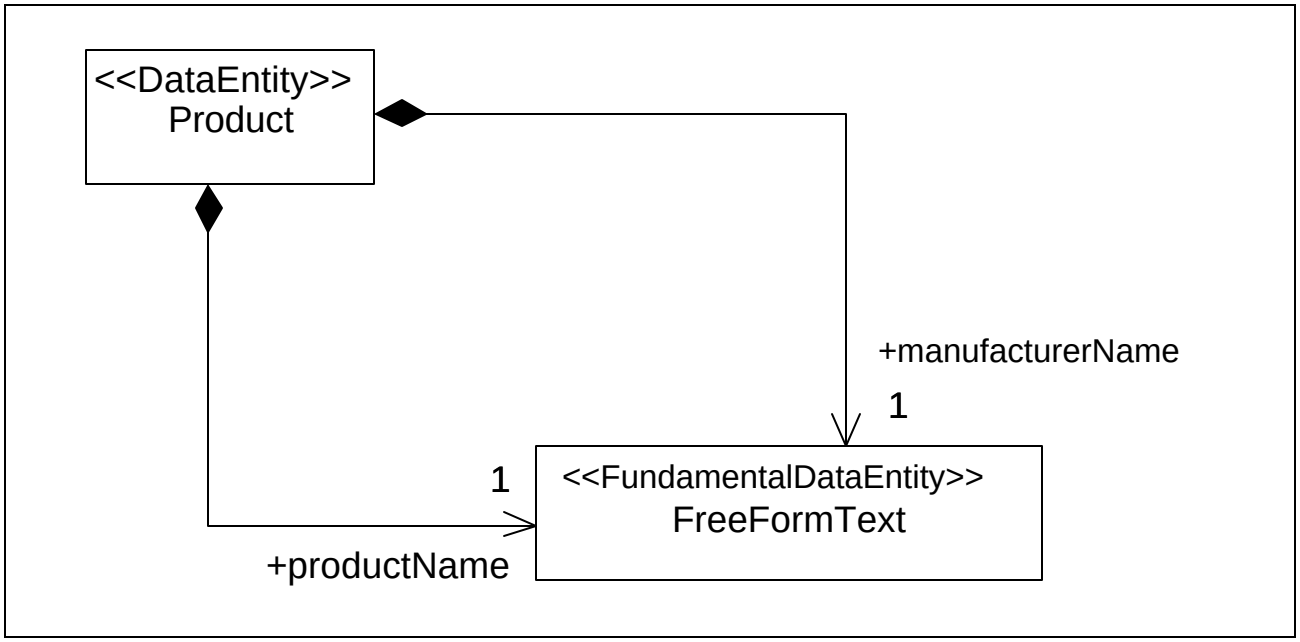

Figure 2.7 Illustration of Reified Data Entities

The XML document schema for this design pattern is shown in Example 2.7.

The *xml:lang* attribute is added to each free form text element. Example 2.7 illustrates the *xml:lang* attribute as CDATA and not as an enumerated option list as this could lead to very large files.

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

Example 2.7 Reification Document Schema

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.

2.5 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 2.8 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.

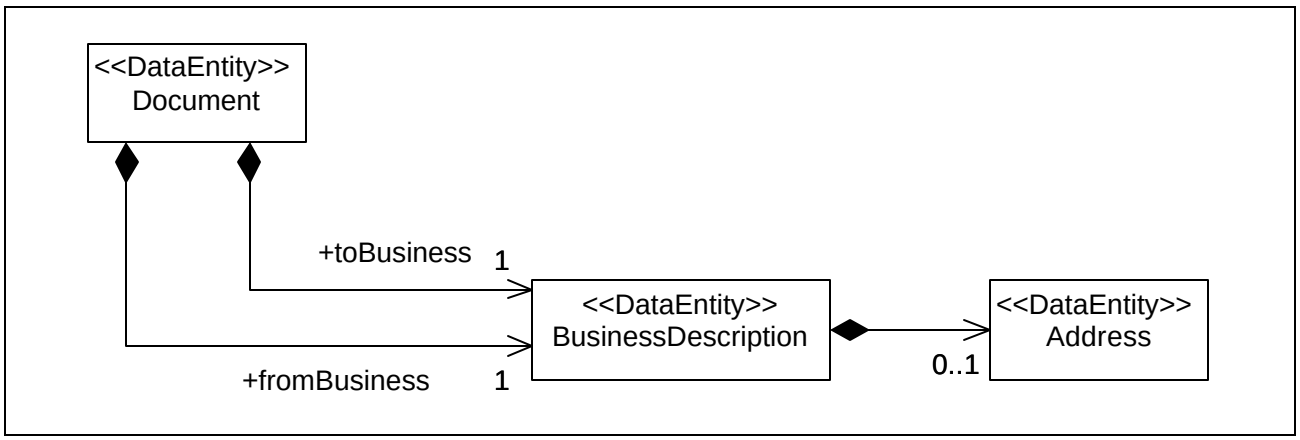

Figure 2.8 Illustration of a Data Entity Model

The UML model in Figure 2.8 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 2.9 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 2.9 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*.

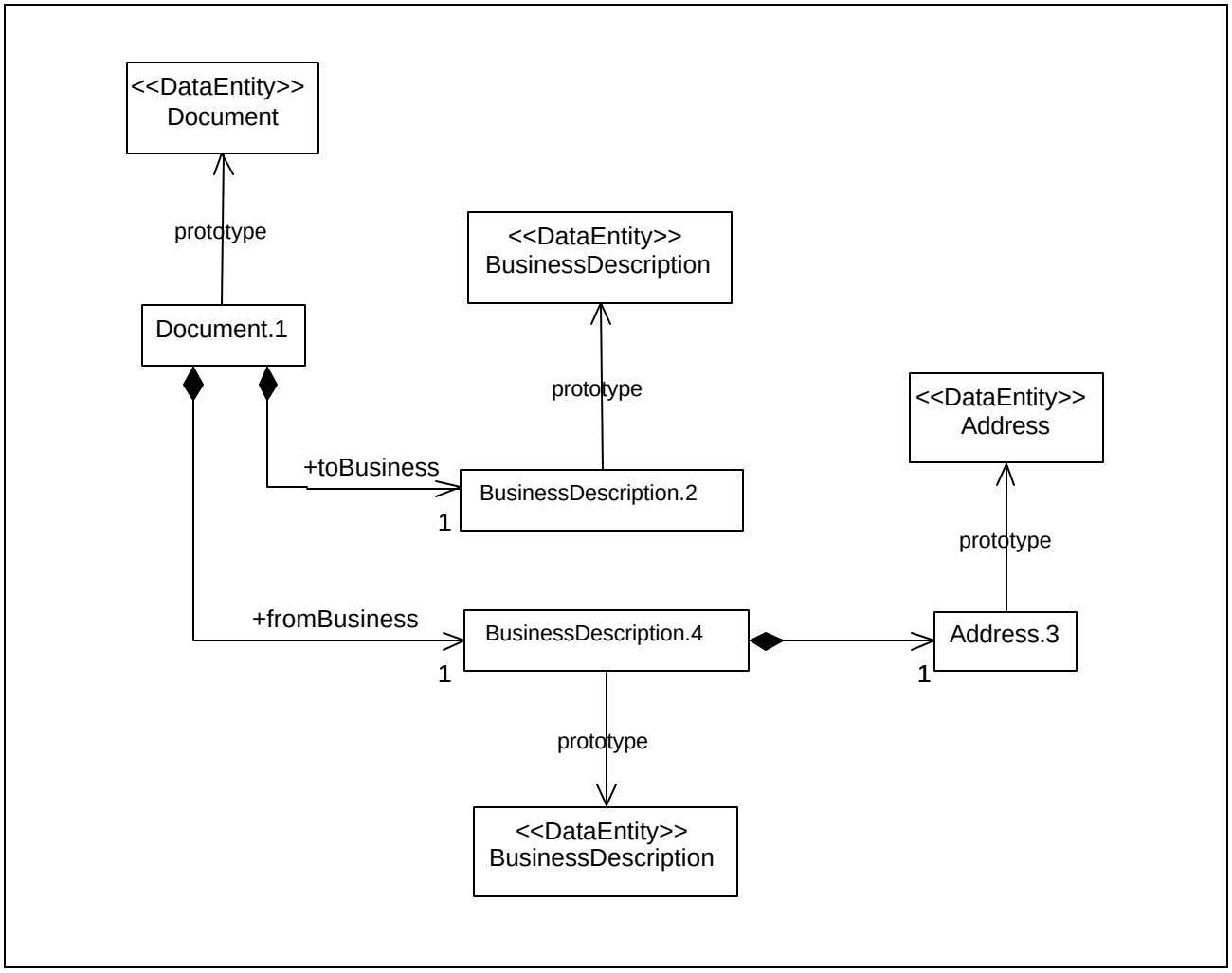

Figure 2.9 Illustration of a Canonical Hierarchy

The design of an algorithm that creates an XML DTD from the graph in Figure 2.9 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 Example 2.8, is produced by the UML to XML DTD algorithm.

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

Example 2.8 Document Schema Example

The *BusinessDescription* element in Example 2.8 specifies the *Address* sub-element as optional that seems in disagreement with the specification in Figure 2.9. 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 2.9.

2.6 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. Example 2.9 illustrates the role descriptions in a business document.

```
1         From Role. Partner Role Description
1                  |-- Global Partner Role Classification Code
1                  |-- Partner Description
1                  |        |-- Global Partner Classification Code
1                  |        |-- Business Description
1                  |        |        |-- Global Business Identifier
1         To Role. Partner Role Description
1                  |-- Global Partner Role Classification Code
1                  |-- Partner Description
1                  |        |-- Global Partner Classification Code
1                  |        |-- Business Description
1                  |        |        |-- Global Business Identifier
```

Example 2.9 Role Specification in a Business Document

- The contact information of the initiating role must be included in 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. Example 2.10 illustrates the contact information in a business document.

```
1         From Role. Partner Role Description
1                  |-- Contact Information
1                  |        |-- Email Address
1                  |        |-- Telephone Number. Communications Number
1                  |        |-- Contact Name. Free Form Text
```

Example 2.10 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. Example 2.11 illustrates supply chain specification in a business document.

```
1         From Role. Partner Role Description
1                  |-- Global Partner Role Classification Code
1                  |-- Partner Description
1                  |        |-- Business Description
1                  |        |        |-- Global Supply Chain Code
1         To Role. Partner Role Description
1                  |-- Global Partner Role Classification Code
1                  |-- Partner Description
1                  |        |-- Global Partner Classification Code
1                  |        |-- Business Description
1                  |        |        |-- Global Supply Chain Code
```

Example 2.11 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. Example 2.12 illustrates the specification of a document identifier in a business document.

```
1          This Document Identifier. Proprietary Document Identifier
1                  |-- Administered By. Business Description
1                  |-- Document Identifier. Free Form Text
0..1    Requesting Document Identifier. Proprietary Document Identifier
1                  |-- Administered By. Business Description
1                  |-- Document Identifier. Free Form Text
```

Example 2.12 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. Example 2.13 illustrates the specification of a data and time stamp in a business document.

```
1          Document Generation Date Time. Date Time Stamp
1                  |-- Time Stamp
1                  |-- Date Stamp
```

Example 2.13 Data and Time Stamp in a Business Document

2.7 Request/Response Business Document Design Pattern

The request/response design pattern is useful for requesting a business partner to perform a business action and 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 possesses. 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 2.10 illustrates a request/response data entity model. A product availability request comprises zero or more query constraints, one or more business constraints and zero or more product descriptions. The query constraints are restrictions that must be met by each result returned in the response. The business constraints are the tests 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. Therefore the responding business partner is required to perform a business action that includes reasoning about how they will respond to such a request for availability. This may require some planning or optimization 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[5] (OCL) expression that constraints each result returned in the response. A business constraint is an 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.

---

[5] Refer to http://www.omg.org for standard specification.

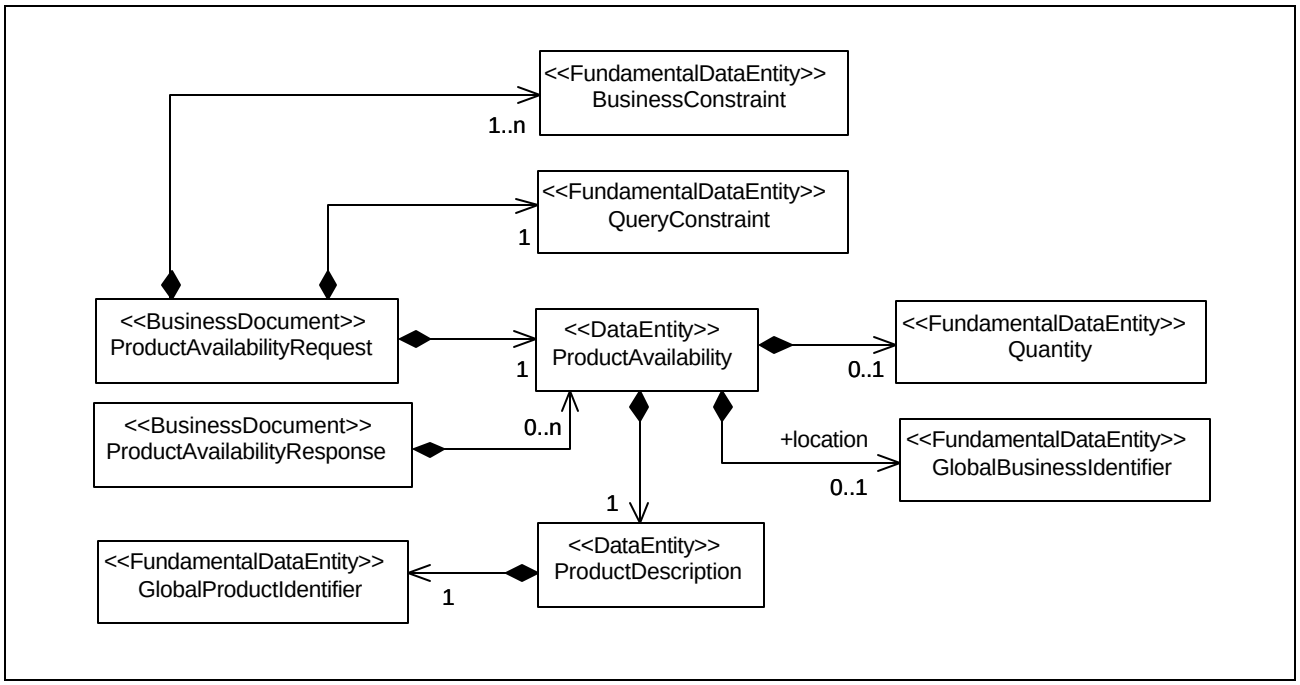

Figure 2.10 Request/Response Data Entity Model

The XML document schema for this design pattern is shown in Example 2.14. 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 ) >
```

Example 2.14 Request/Response Document Schema

An example product availability request is shown in Example 2.15. Availability on a product with the designated 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>
```

Example 2.15 Product Availability Request Example

An example product availability request response is shown in Example 2.16. The response to 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[6].

```
<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>
```

Example 2.16 Product Availability Response Example

[6] A query/response design pattern cannot express this requirement.

3 Business Information Flow Design Patterns

3.1 Commercial Transaction Design Pattern

A commercial 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 defines the 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 commercial 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 buyer provides the product and the seller 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.

The BCF treats all commercial transactions 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 commercial 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 3.1 illustrates a commercial transaction specification that does not include a responding business document and Figure 3.2 illustrates a commercial transaction specification that includes a responding business document.

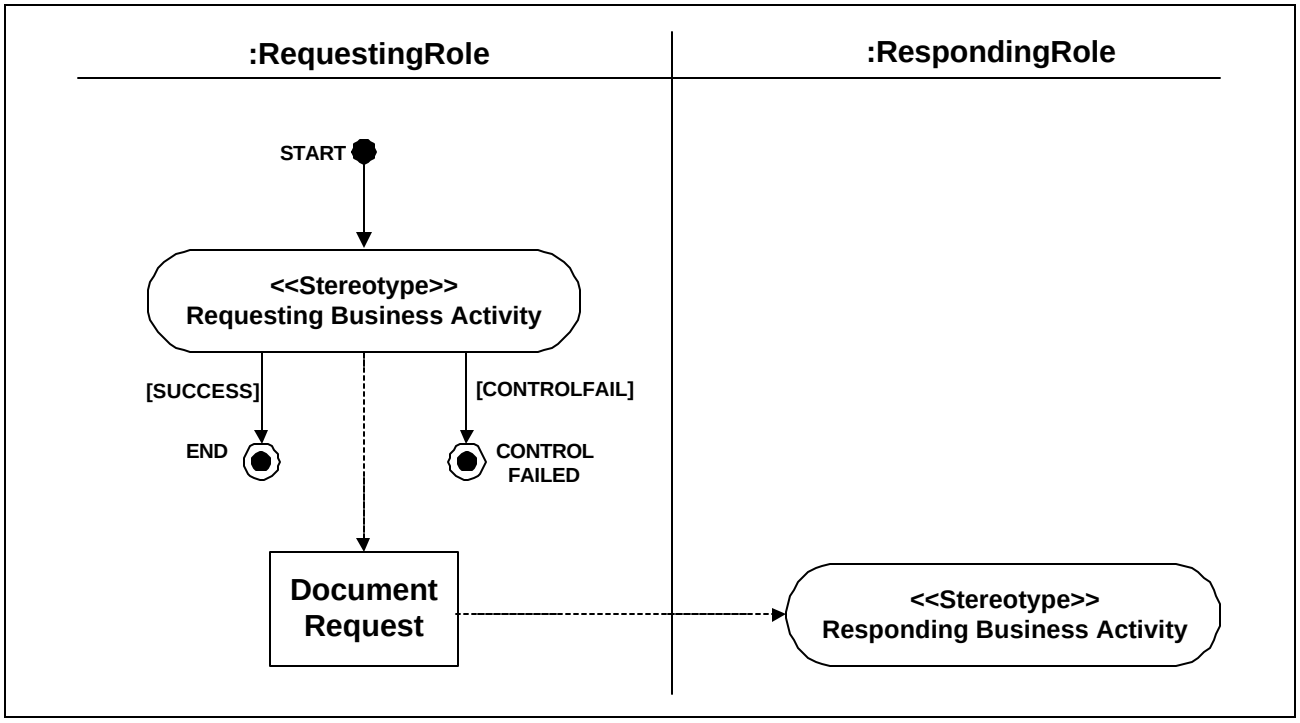

Figure 3.1 Commercial Transaction without Responding Business Document

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

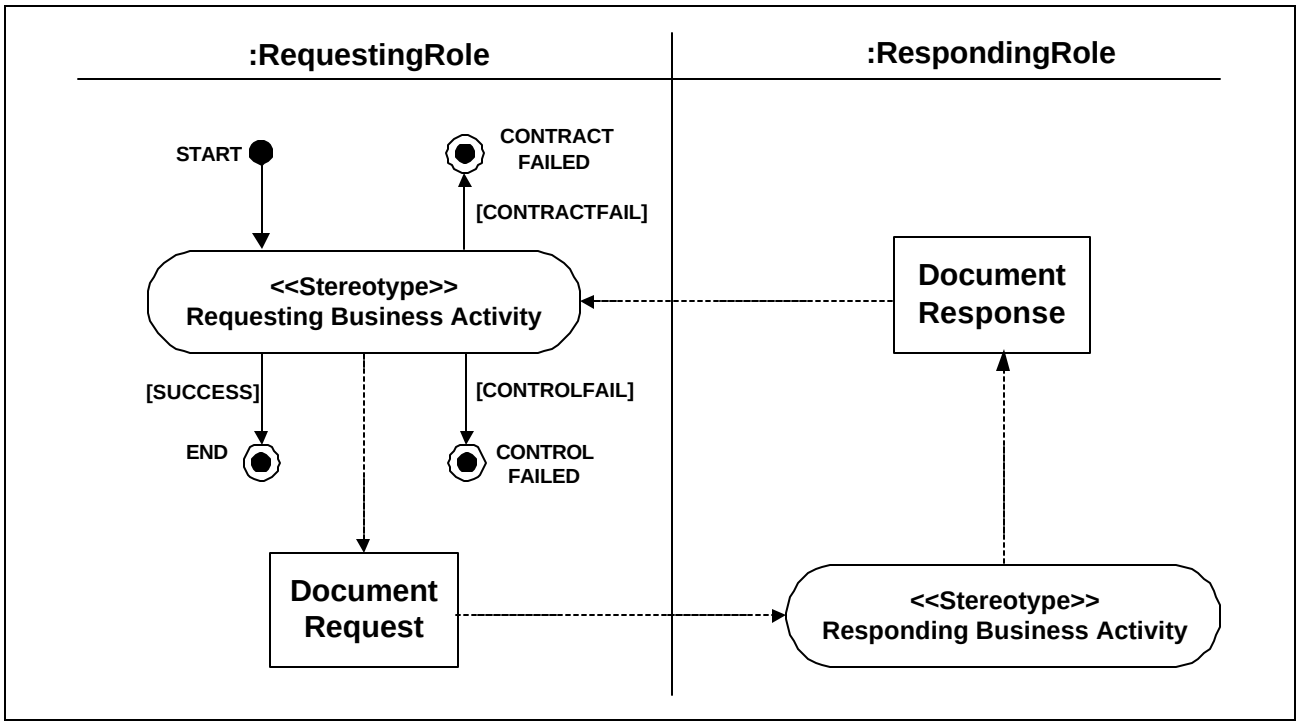

Figure 3.2 Commercial Transaction with Responding Business Document

3.1.1 Commercial Transaction State Semantics

The initial (START) state and the Final (END, CONTROLFAILED, CONTRACTFAILED) state represent the state of a commercial 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 requester'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 must hold before the commercial transaction can transition into the "default" state (a UML definition).

3.1.1.1 START STATE SEMANTICS

The condition that must 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. Does 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.

3.1.1.2 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 >. For example, Purchase Order Status Open.
- The <Noun> can be a Business Document with no named property in the form <Noun> <Verb>. For example, Purchase Order Exists.
- The <Property> can be the name of a business process support system with no <Noun> in the form <Property><Verb>. For example, 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). Make sure only valid status is used from the dictionary or another valid status must be added 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

3.1.1.3 END, CONTROLFAILED AND CONTRACTFAILED STATE SEMANTICS

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

1. *Contract Failure*—The state machine must transition into the CONTRACTFAILED state if the intended commercial 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 must transition the state machine into the CONTRACTFAILED state. The contract failure end state must only be used for commercial transactions that permit negative ACKNOWLEDGMENTs. In these instances the commercial transaction activity graph is shown in Figure 3.2. If there is no contract failure condition then the transaction activity graph is shown in Figure 3.3.

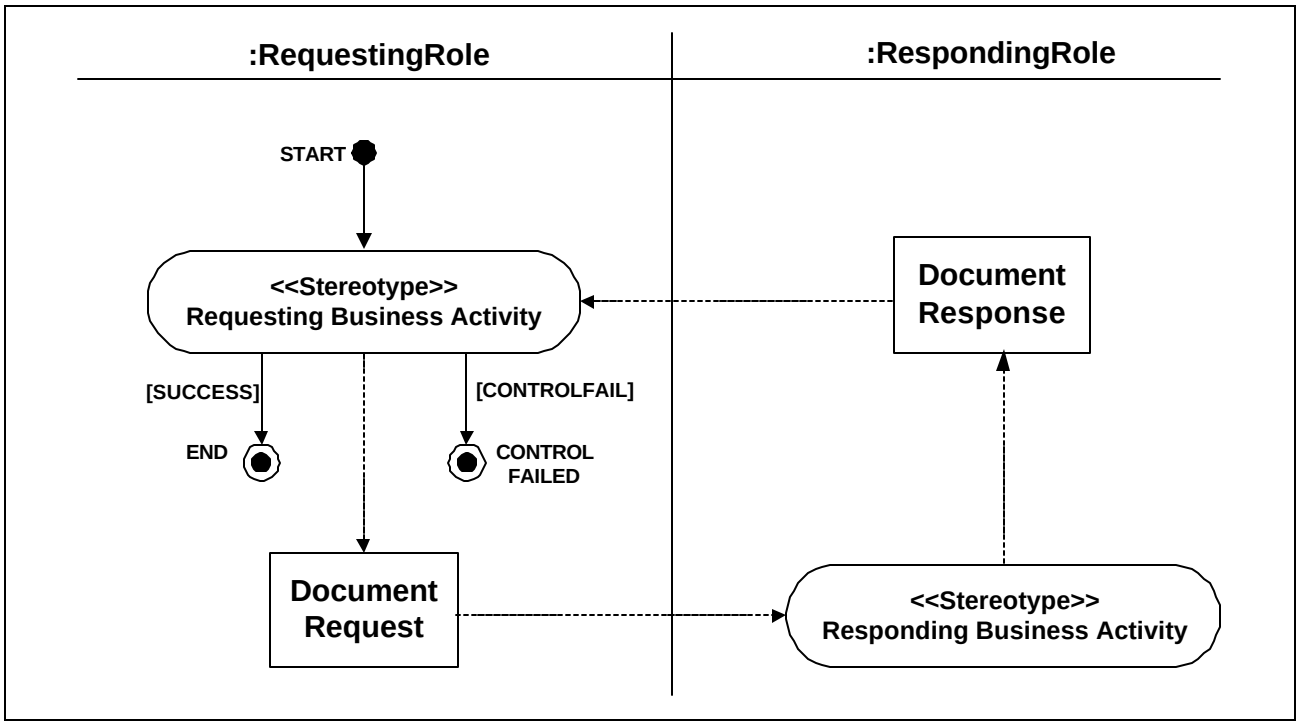

Figure 3.3 Commercial Transaction with No Contract Failure State

2. *Control Failure*—The state machine must transition into the CONTROLFAILED state if any business collaboration control parameter is violated, for example, time outs, processing exceptions, non-repudiation and authorization exceptions. In these cases both the transaction fails and the contract is not formed.

The conditions that must 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 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 (<BDE> 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

3.1.1.4 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>, such as Purchase Order Status Open.
- The <Noun> can be a Business Document with no named property in the form <Noun> <verb>, such as Purchase Order Acceptance Exists.
- The <Property> can be the name of a business process support system with no <Noun> in the form <Property><Verb>, such as 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 only valid status from the dictionary is used or another valid status is added 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

3.1.2 Commercial Transaction Design Rationale

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

1. **Grammar validation**—Task of verifying that the grammar of a message is valid (usually only the header of the message at this step).
2. **Sequence validation**—Task of verifying that the collaboration control information is valid with respect to the commercial transaction specification.
3. **Schema validation**—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**—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**—Task of processing the request in the initiating business document.

Figure 3.4 illustrates the processing of an initiating message when the contract-closing (contract acceptance document) message is an ACKNOWLEDGMENT of receipt. The ACKNOWLEDGMENT of receipt is a business signal, i.e. it does not map onto a business document.

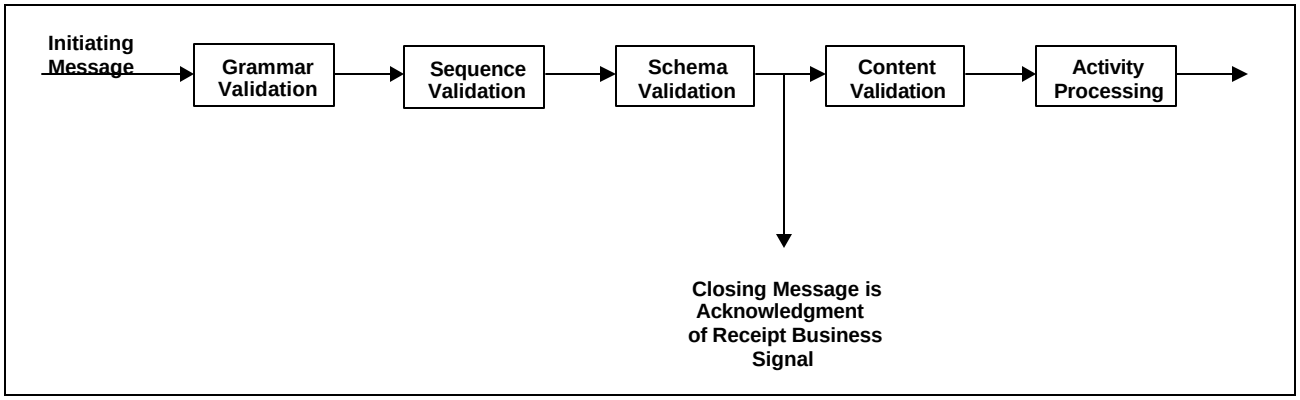

Figure 3.4 ACKNOWLEDGMENT of Receipt Closing Message

Table 3.1 shows an example of time-out parameters for this commercial transaction. The Information Distribution and Notification business activity specification (see BCF#7 document, "E-business Collaboration Modeling Metamodel") 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	24 hrs	N/A	24 hrs

Table 3.1 Time-out Parameters for ACKNOWLEDGMENT of Receipt

Figure 3.5 illustrates the processing of an initiating message when the closing message is an ACKNOWLEDGMENT 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 that initiating business document identification data, the reason for rejection and a 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 commercial transaction.
2. If a non-substantive business acceptance is required then a responding business document is **not** specified in a commercial transaction.

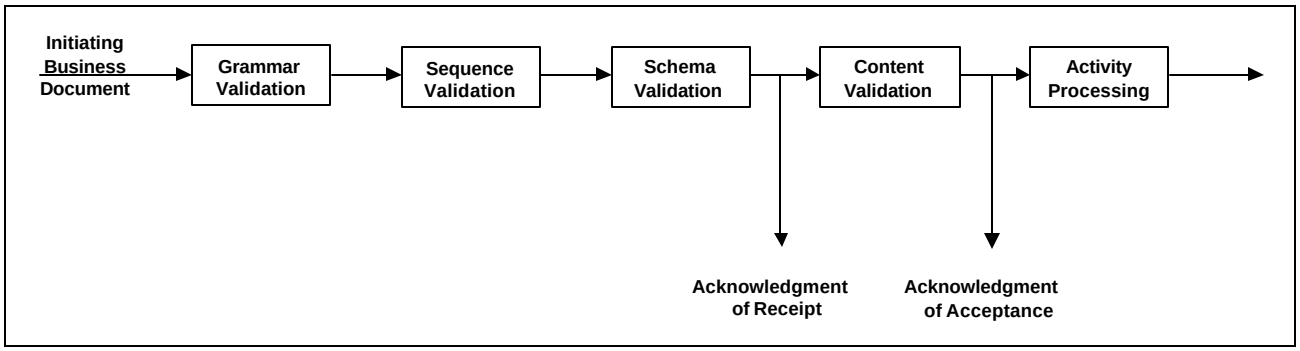

Figure 3.5 ACKNOWLEDGMENT of Business Acceptance Closing Message

Table 3.2 shows example time-out parameters for this commercial agreement. The Business Transaction Activities (see BCF#7, "E-business Collaboration Modeling Metamodel") use this design pattern when a substantive business ACKNOWLEDGMENT of acceptance is required.

Business Activity Performance Controls				
ROLE NAME	ACTIVITY NAME	TIME TO ACKNOWLEDGE RECEIPT	TIME TO ACKNOWLEDGE ACCEPTANCE	TIME TO PERFORM
Role	Activity	2 hrs	6 hrs	6 hrs

Table 3.2 Time-out Parameters for ACKNOWLEDGMENT of Acceptance

Figure 3.6 illustrates the processing of an initiating message when the closing message is a responding business document. The Query/Response business activity specification (see BCF#7 document, "E-business Collaboration Modeling Metamodel") uses this design pattern.

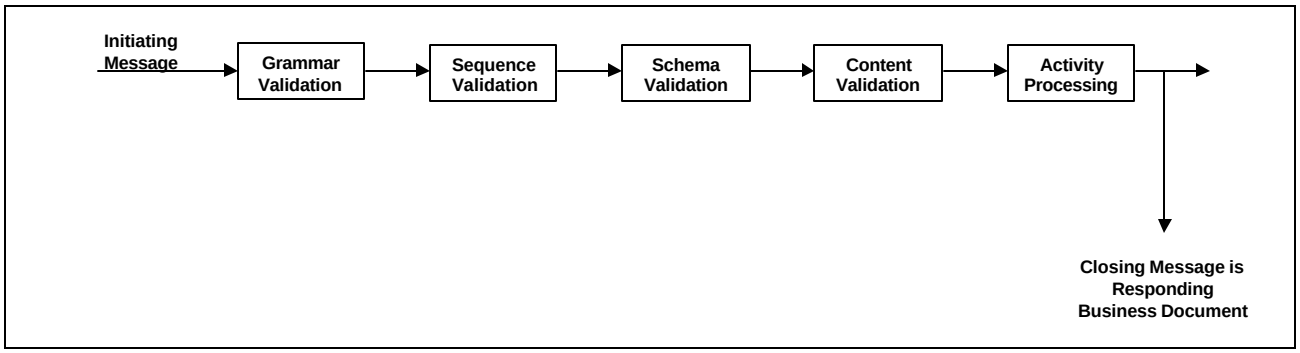

Figure 3.6 Responding Business Document is Closing Message

Table 3.3 shows example of time-out parameters for this commercial transaction.

Business Activity Performance Controls				
ROLE NAME	ACTIVITY NAME	TIME TO ACKNOWLEDGE RECEIPT	TIME TO ACKNOWLEDGE ACCEPTANCE	TIME TO PERFORM
Role	Activity	N/A	N/A	24 hrs

Table 3.3 Time-out Parameters When Closing Message Is a Business Document

It is possible to specify ACKNOWLEDGMENTs and a responding business document as part of the commercial agreement. Figure 3.7 illustrates the processing of an initiating message when there is a requirement for an ACKNOWLEDGMENT 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 must be substantive then two commercial transactions are required.

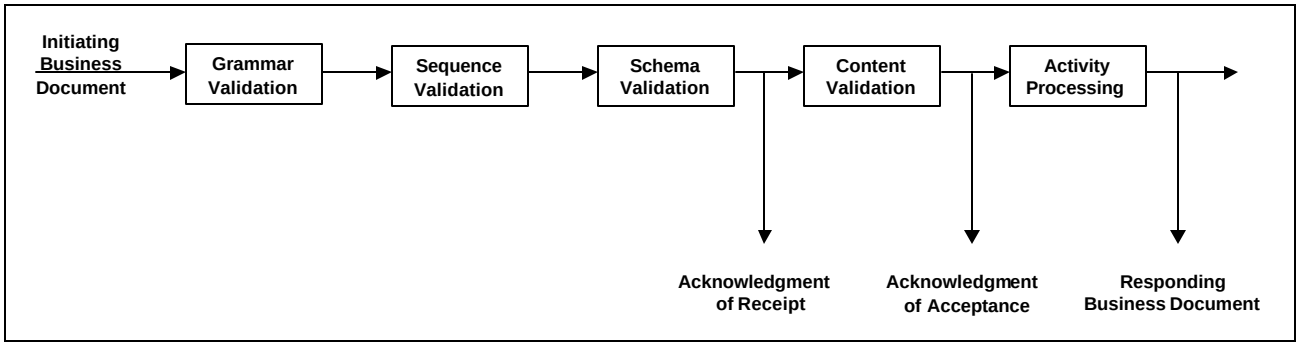

Figure 3.7 Receipt, Business Acceptance and Business Document Response

Table 3.4 shows example time-out parameters for this commercial transaction. The Business Transaction Business Activity specification (see BCF#7, "E-business Collaboration Modeling Metamodel") that mandates the return of a non-substantive business acceptance ACKNOWLEDGMENT 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	2 hrs	6 hrs	24 hrs

Table 3.4 Time-out Parameters for Receipt, Business Acceptance and Business Document Response

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

1. There is a value for "Time to Acknowledge Acceptance."
2. Whether the value for "Time to Perform" is either equal or not equal to the "Time to Acknowledge Acceptance."
3. And whether there 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 ACKNOWLEDGMENT 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 ACKNOWLEDGMENT 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 ACKNOWLEDGMENT of acceptance is non-substantive.

3.1.3 Business Transaction Design Pattern

The business transaction design pattern is illustrated in Figure 3.8.

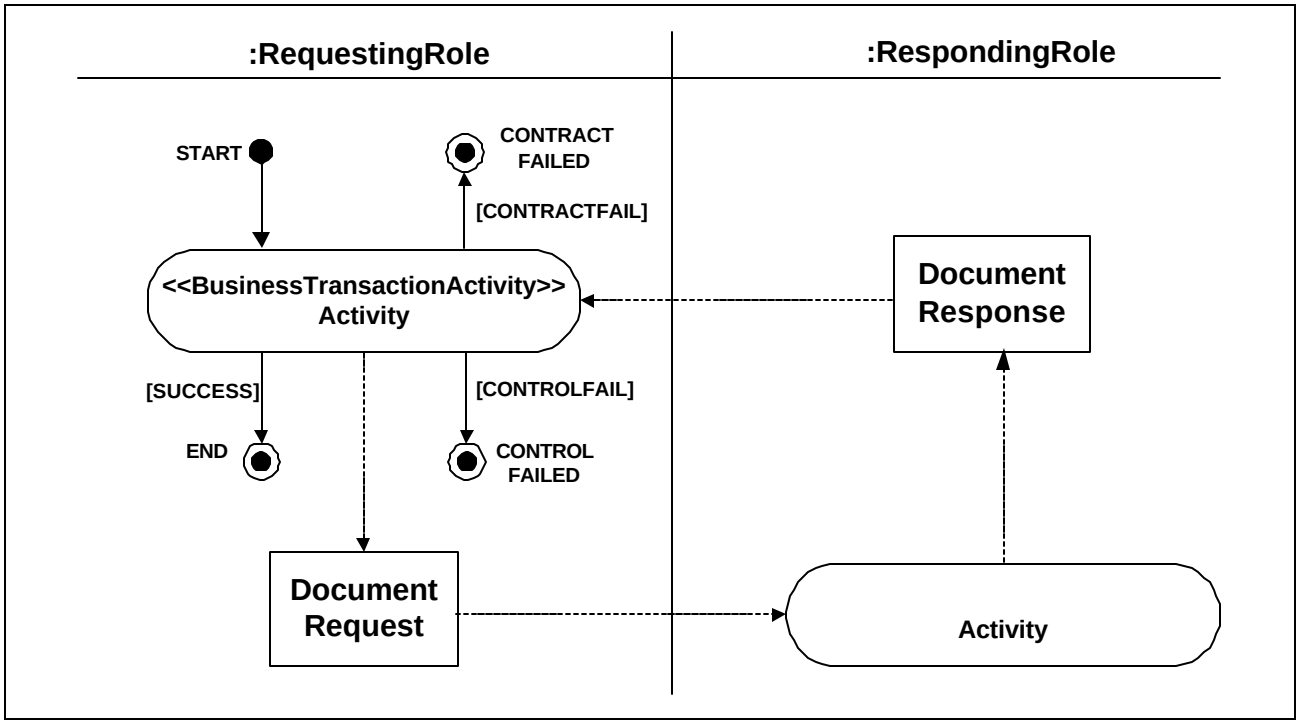

Figure 3.8 Business Transaction Activity Design Pattern

This design pattern is best used to model the "*offer and acceptance*" commercial 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 that 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 offerer must be such as to induce a "reasonable person" to believe that he/she is making the alleged offer.
- The alleged offeree must actually hold that belief—i.e. believe that the offerer 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 ACKNOWLEDGMENT of the requesting business document when it passes a "Business Acceptance" test, i.e. passes the content validation step in illustrated Figure 3.7. This ACKNOWLEDGMENT 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 commercial transaction pattern is to model the formation of an offer and acceptance commercial contract[7] (Refer to the appendix for examples). If the requesting role transitions from their business activity into the control failure state then the role must initiate a notification of failure (see notification of failure design pattern) commercial transaction to revoke their original offer.

Note that the "CONTRACTFAILED" final state can be omitted from the commercial transaction specification if there are no negative business acceptance documents specified.

[7] Refer to the following documents to understand on-line commercial 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

3.1.4 Query/Response Design Pattern

Figure 3.9 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 ACKNOWLEDGMENT or business acceptance ACKNOWLEDGMENT.

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.

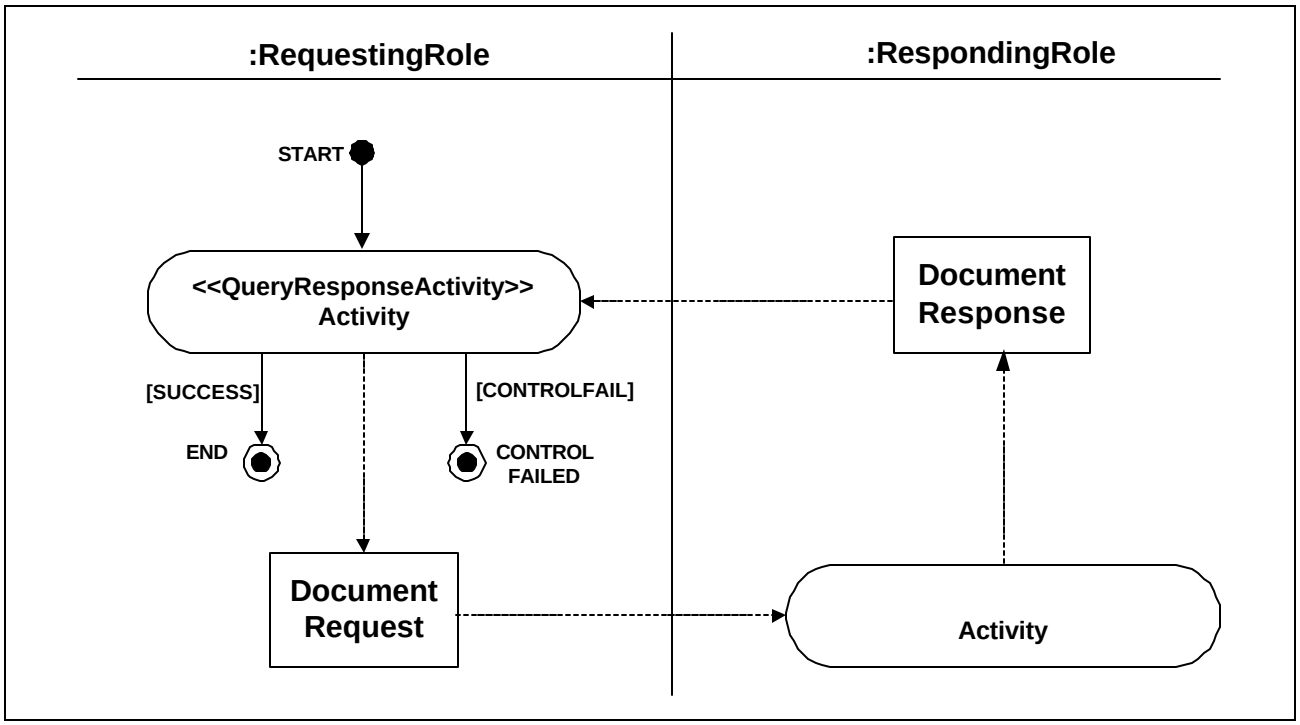

Figure 3.9 Query/Response Activity Design Pattern

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. However, the request/response design pattern should be used instead when there are “aggregate” or “interdependent” constraints that must be applied to a set of results.

3.1.5 Request/Response Design Pattern

Figure 3.10 illustrates the request/response 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 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. ACKNOWLEDGMENT 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.

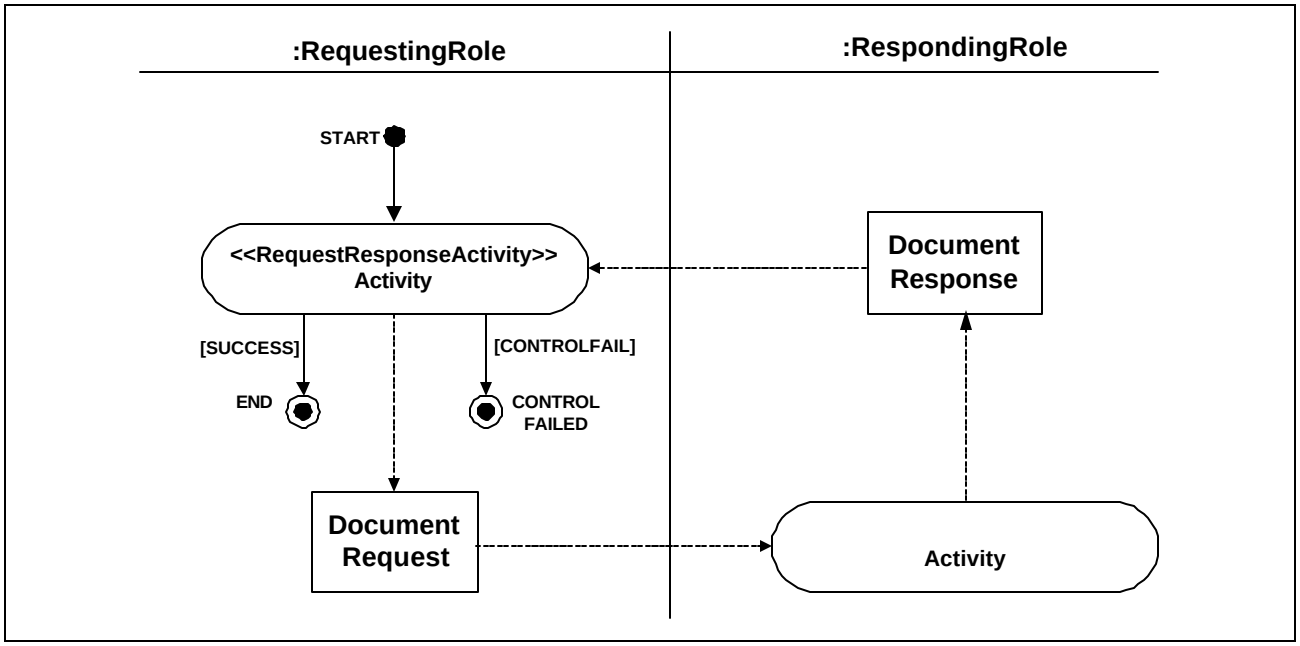

Figure 3.10 Request/Response Activity Design Pattern

The request/response activity pattern must be used for commercial contracts when 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) must be less than 100. This response requires some business processing on a query before a response is returned to the requester. 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 must be applied to a set of results then the query/response pattern must be used.

3.1.6 Request/Confirm Design Pattern

Figure 3.11 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 requester is authorized or not authorized to sell the products. This pattern specifies the exchange of a requesting and responding business document. ACKNOWLEDGMENT of receipt is expected—it is the initiator's obligation to follow up on the request until an ACKNOWLEDGMENT of receipt is received. ACKNOWLEDGMENT 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.

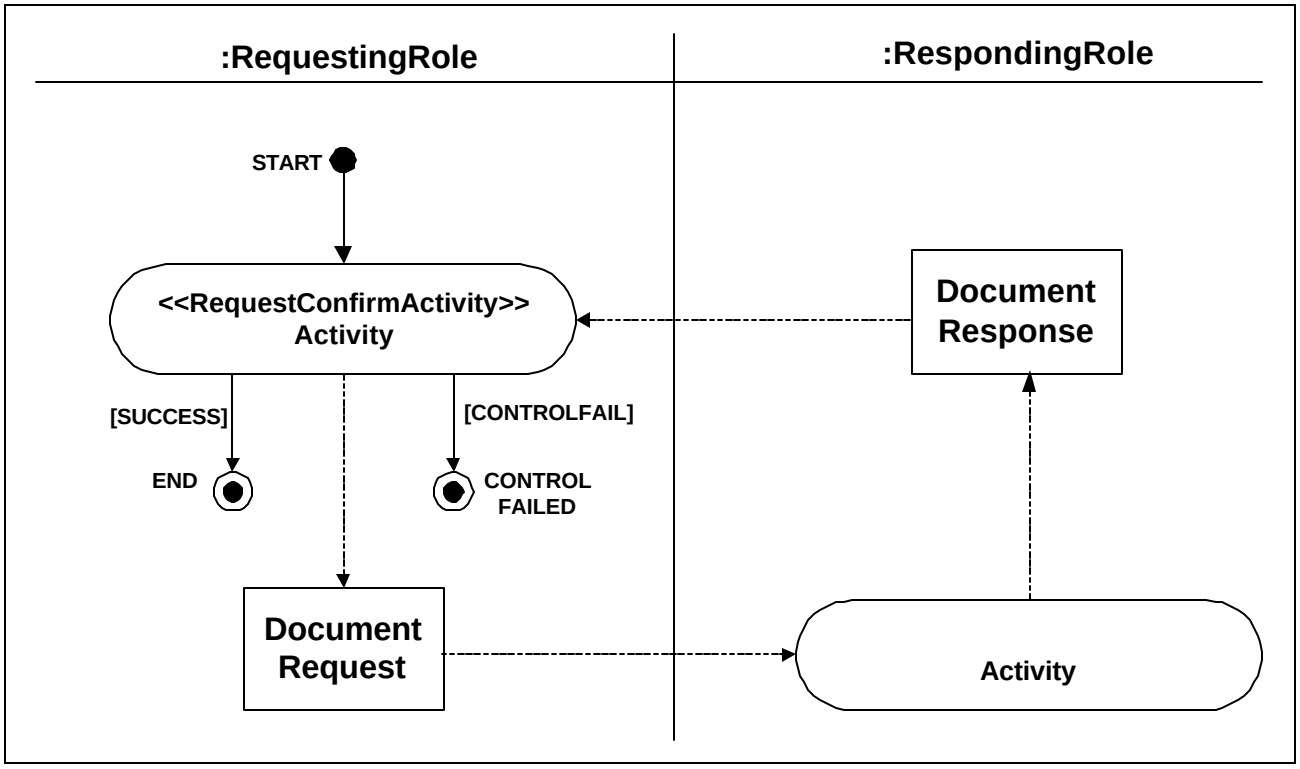

Figure 3.11 Request/Response Activity Design Pattern

The request/confirm activity pattern must be used for commercial 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.

3.1.7 Information Distribution Design Pattern

Figure 3.12 illustrates the information distribution design pattern. This pattern specifies the exchange of a requesting business document and the return of an ACKNOWLEDGMENT of receipt business signal. This pattern is used to model an **informal** information exchange commercial transaction that therefore has no non-repudiation requirements.

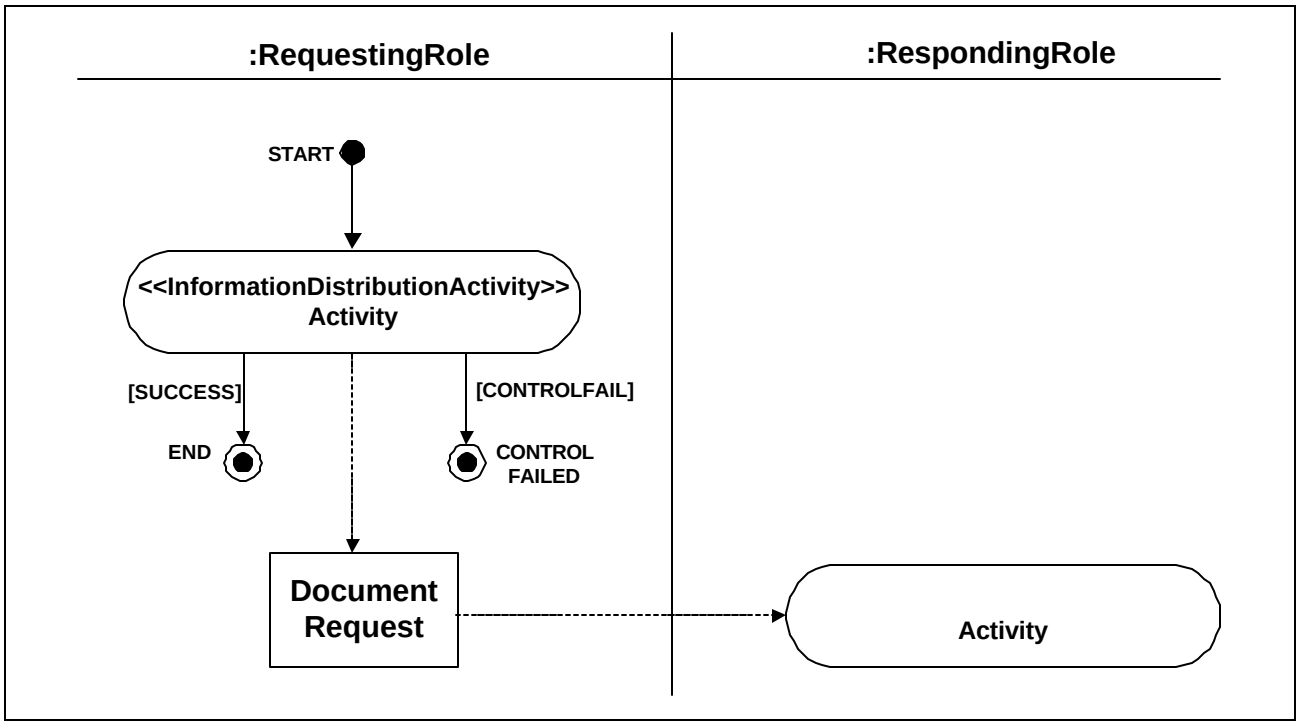

Figure 3.12 Information Distribution Design Pattern

3.1.8 Notification Design Pattern

Figure 3.13 illustrates the notification design pattern. This pattern specifies the exchange of a notifying business document and the return of an ACKNOWLEDGMENT of receipt business signal. This pattern is used to model a **formal** information exchange commercial transaction that therefore has non-repudiation requirements.

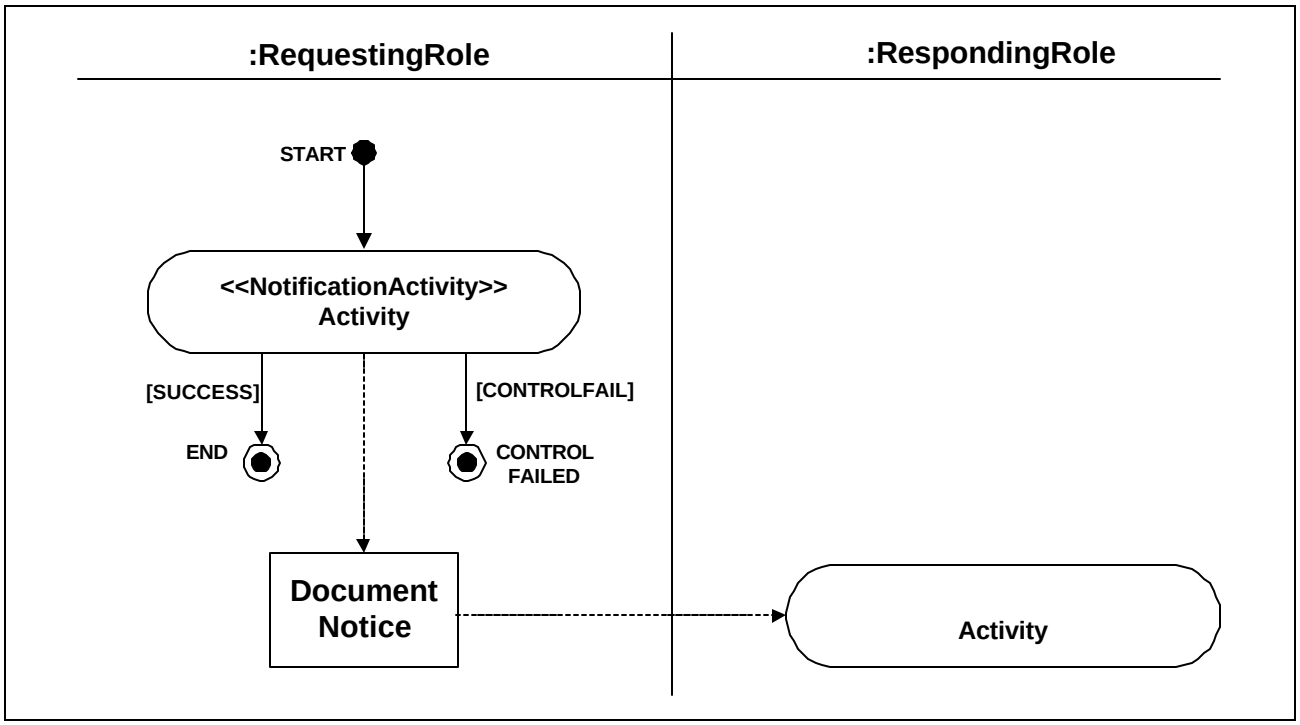

Figure 3.13 Notification Design Pattern

3.1.8.1 NOTIFICATION OF FAILURE SEMANTICS

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

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

1. The originating partner's business activity times-out when waiting for a specified response to their requesting business document.
2. The originating partner's 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 ACKNOWLEDGMENT 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 requester initiates this commercial 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 commercial transaction.

It is recommended that the Notification of Failure commercial 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 commercial transaction is not exercised when a responding business partner encounters a business process or control exception when responding to a business document request.

3.2 Business Collaboration Protocol Design Pattern

A business collaboration protocol choreographs commercial transactions. The UML activity diagram notation is used to specify these business collaborations protocols. The following are named design patterns:

1. Acceptance
2. Others yet to be determined

3.2.1 Acceptance Business Collaboration Design Pattern

A blanket acceptance business collaboration protocol specifies the execution of a single commercial transaction activity that either succeeds or fails. On success the collaboration ends. On failure the commercial transaction activity transitions to the notification of failure commercial transaction activity that is also executed only once. The collaboration is termed a blanked acceptance to model a commercial transaction that expects a contract to be formed upon receipt of a single contract acceptance document. This collaboration is illustrated in Figure 3.14.

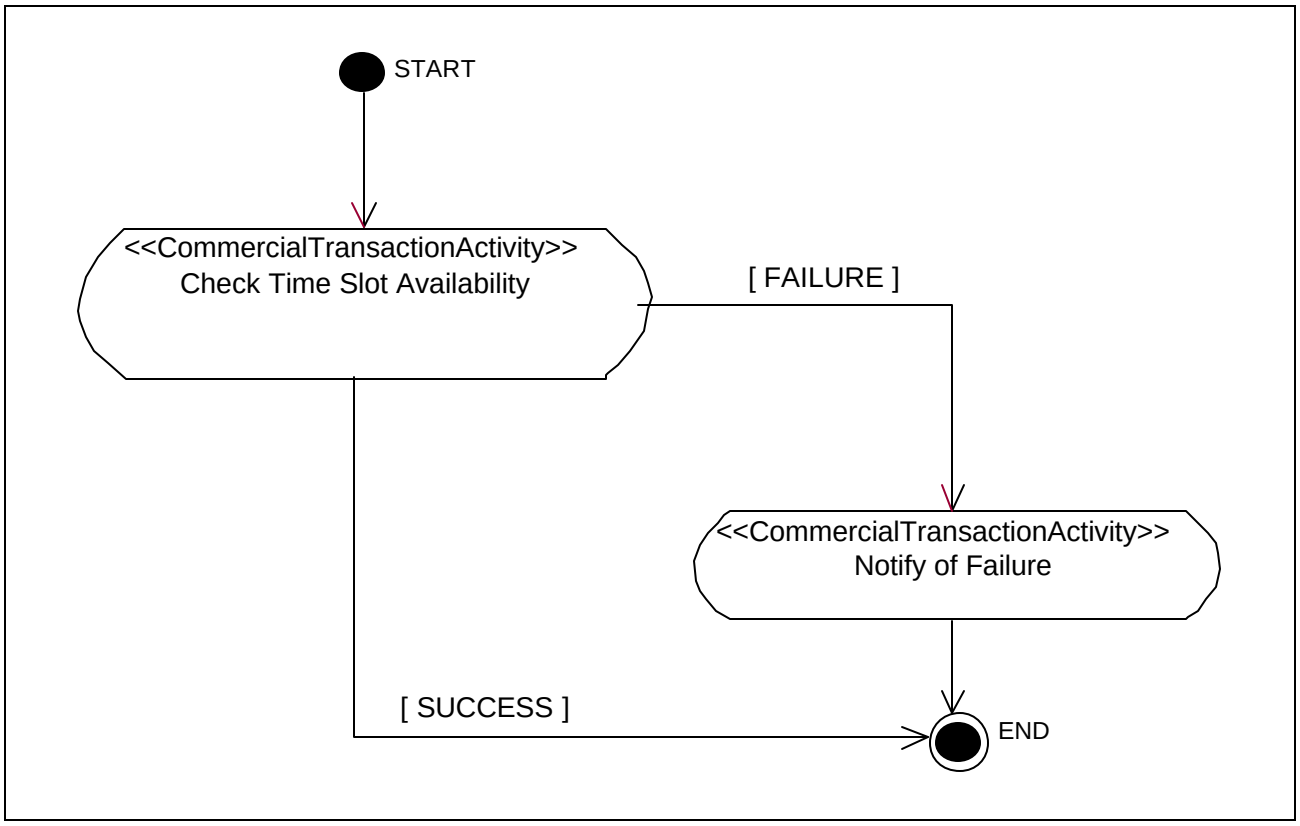

Figure 3.14 Acceptance Business Collaboration

3.3 Network Component Interaction Diagram Pattern

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

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

3.3.1 Service-Service

Business Transaction Activity

There are three variations of the business transaction activity.

First, time to perform equals time to acknowledge acceptance and no responding business document.

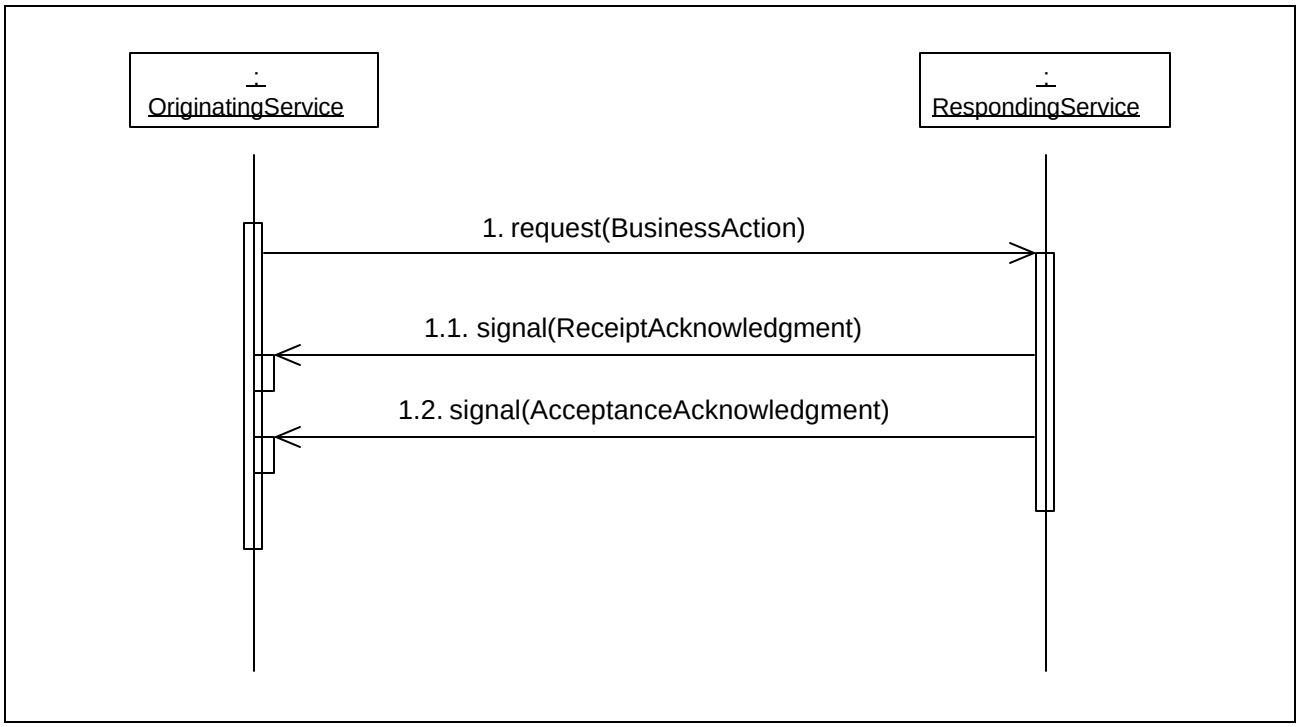

Figure 3.15 Service-Service Interaction Pattern—I

Second, time to perform equals time to acknowledge acceptance and a responding business document.

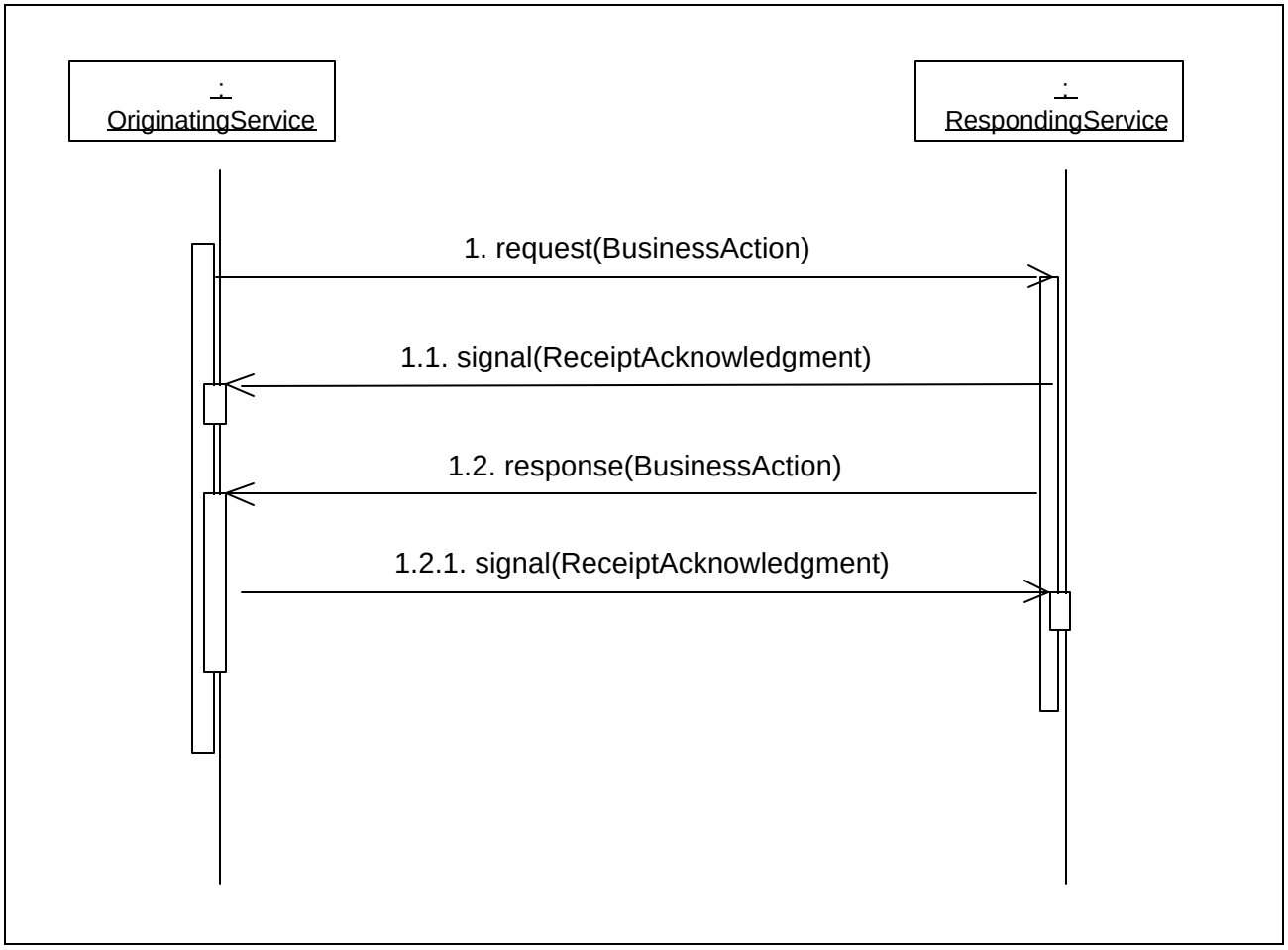

Figure 3.16 Service-Service Interaction Pattern—II

And third, time to perform is greater than time to acknowledge acceptance.

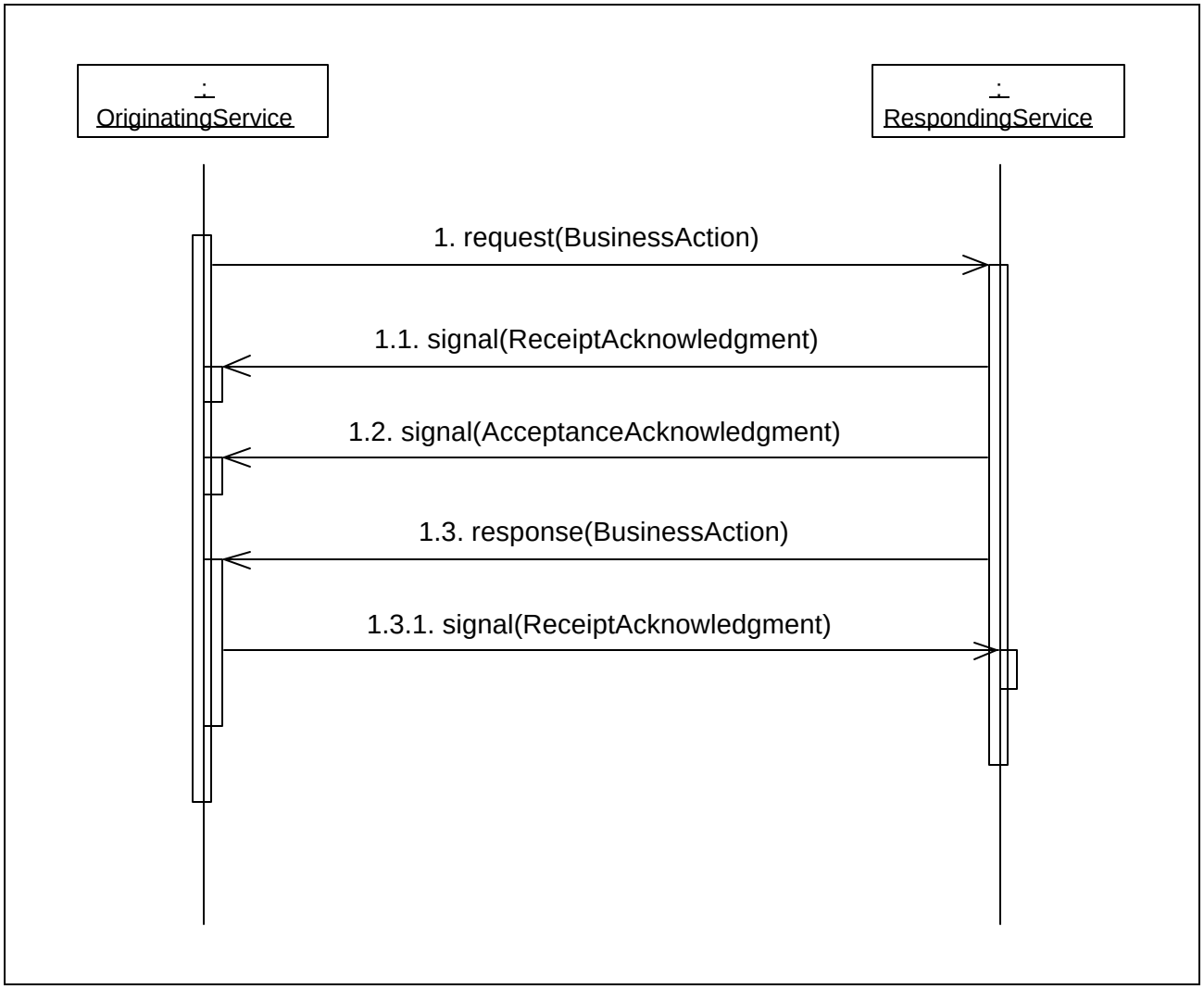

Figure 3.17 Service-Service Interaction Pattern—III

Query/Response Activity, Request/Response Activity and Request/Confirm Activity

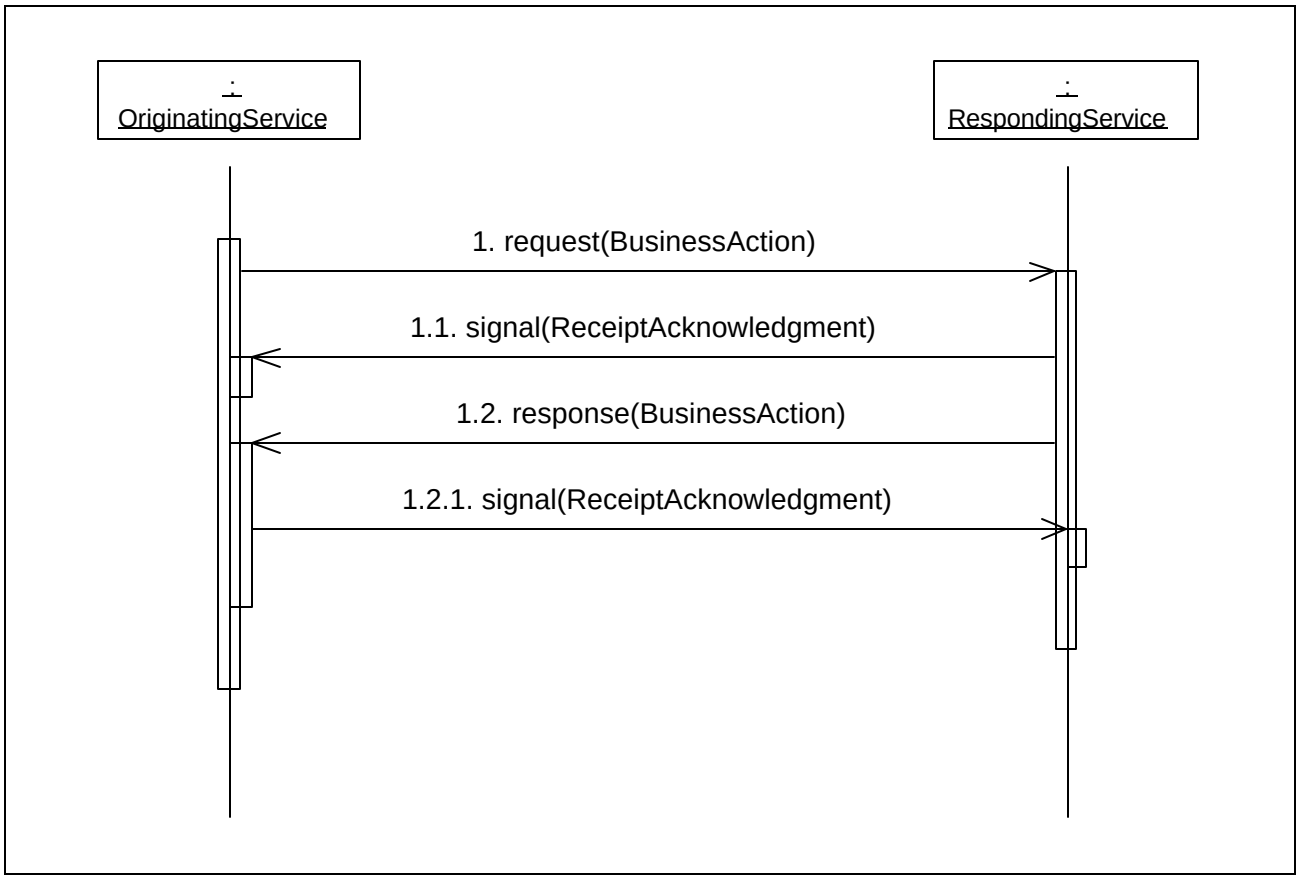

Figure 3.18 Service-Service Interaction Pattern—IV

Information Distribution Activity and Notification Activity

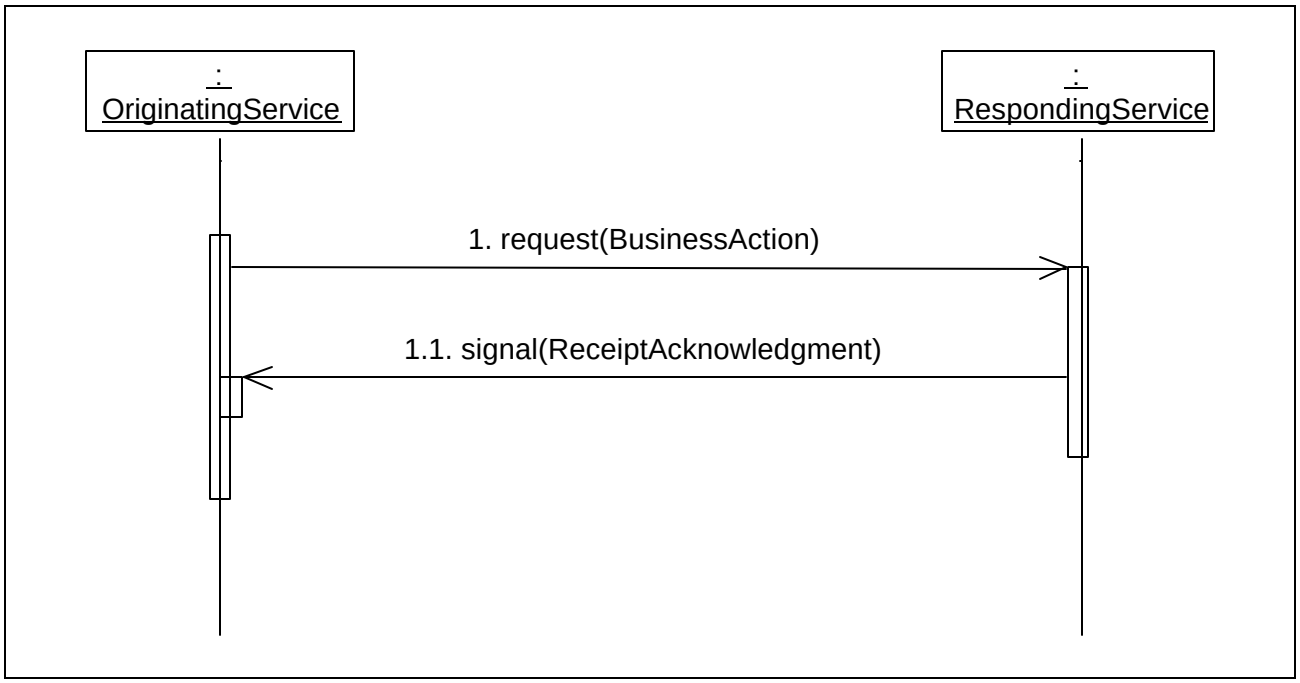

Figure 3.19 Service-Service Interaction Pattern—V

3.3.2 Agent-Service-Service

Business Transaction Activity

Time to perform equals time to acknowledge acceptance and no responding business document.

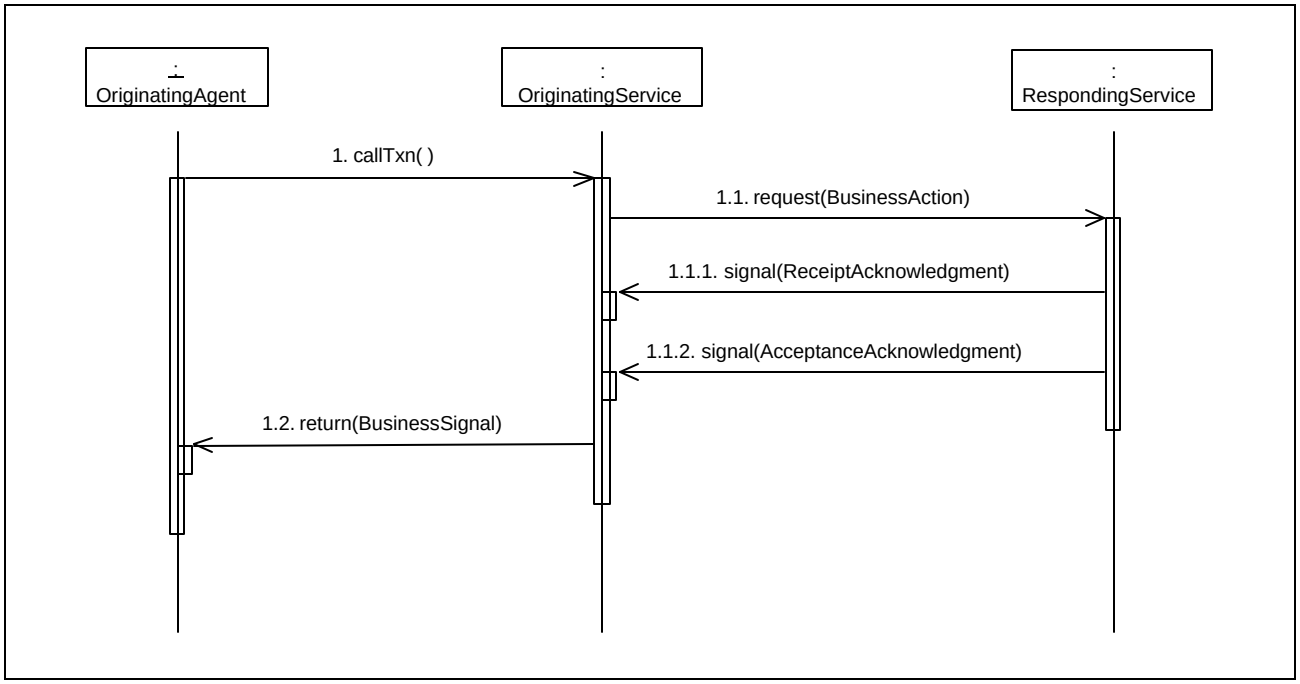

Figure 3.20 Agent-Service-Service Interaction Pattern—I

Time to perform equals time to acknowledge acceptance and a responding business document.

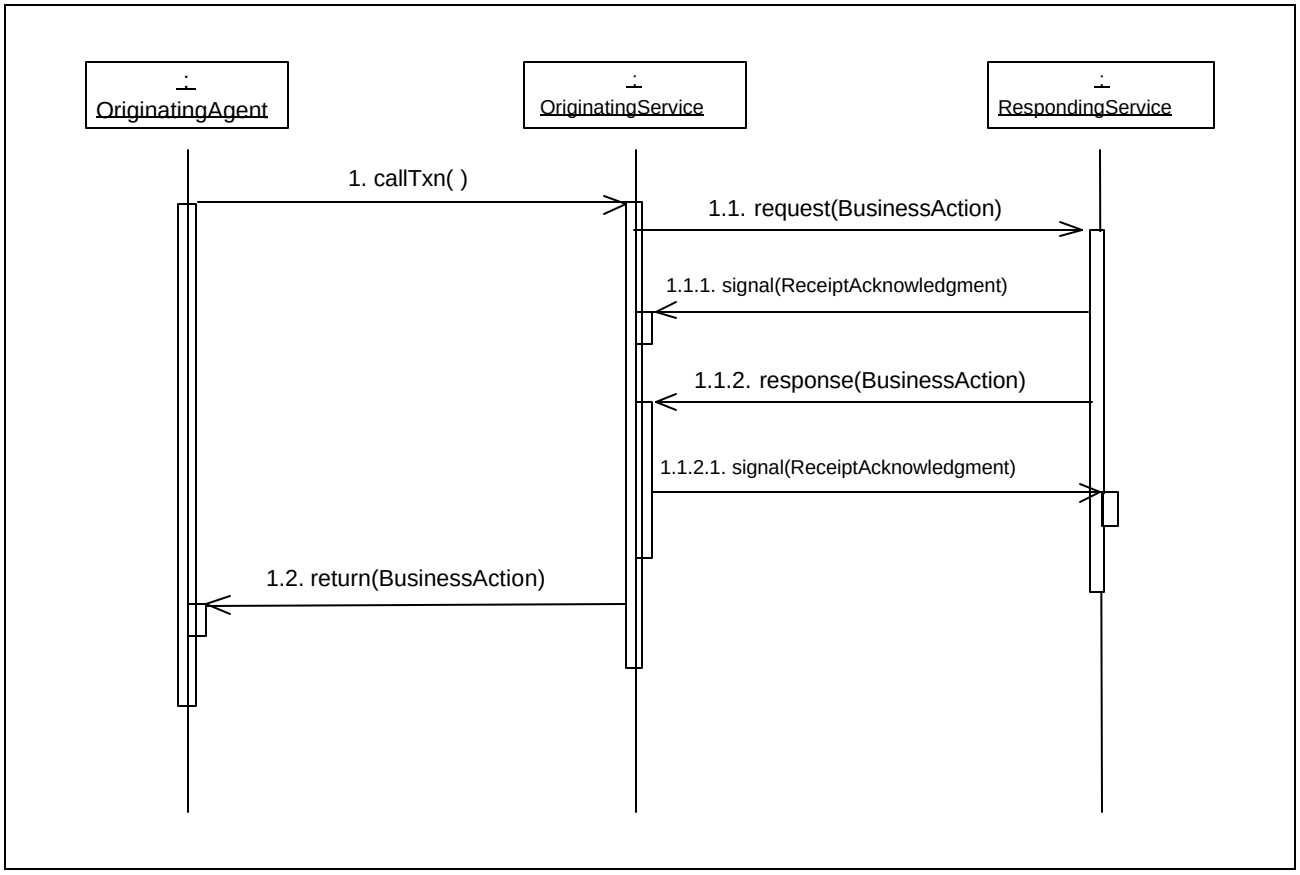

Figure 3.21 Agent-Service-Service Interaction Pattern—II

Time to perform is greater than time to acknowledge acceptance.

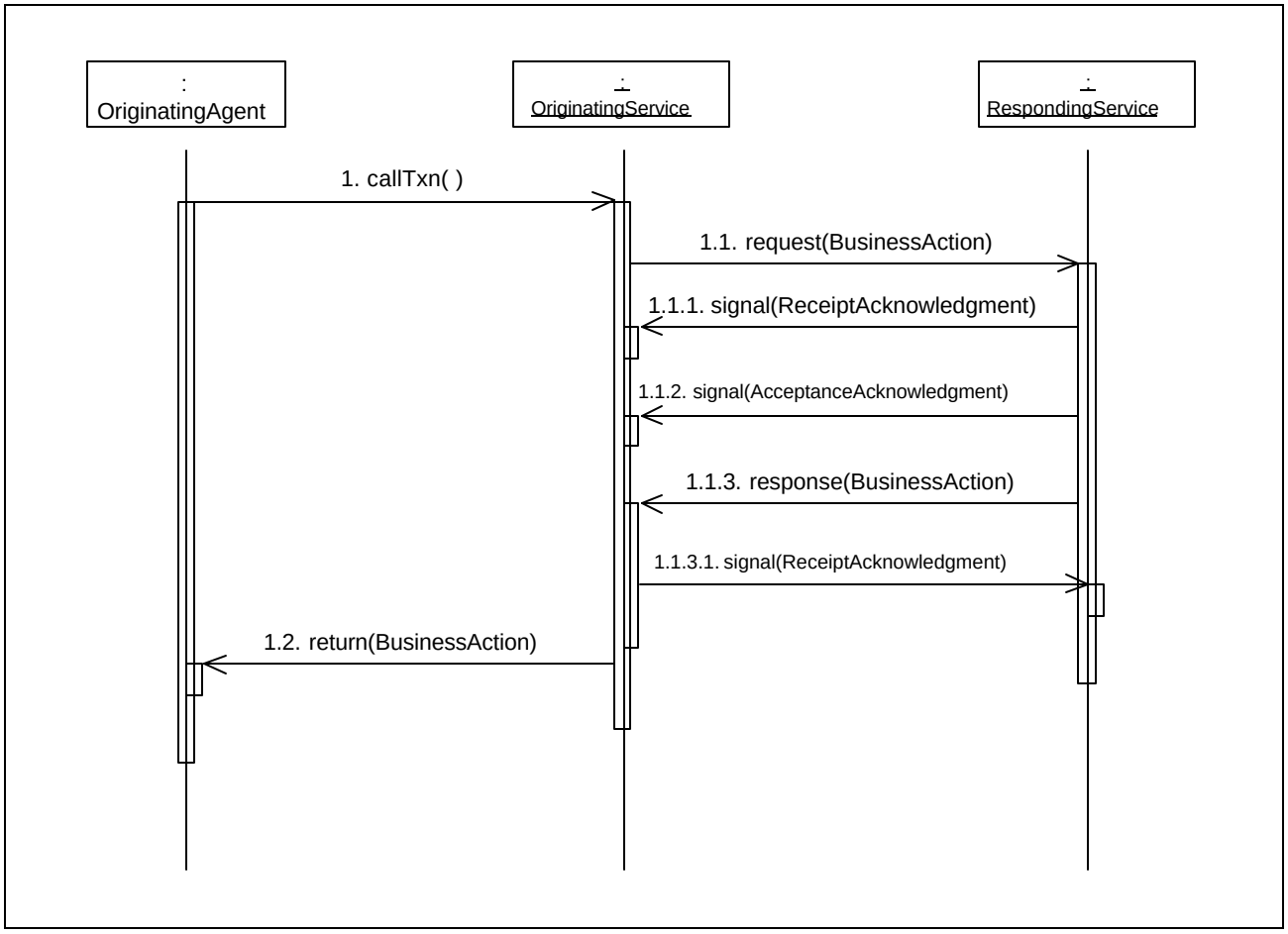

Figure 3.22 Agent-Service-Service Interaction Pattern—III

Query/Response Activity, Request/Response Activity and Request/Confirm Activity

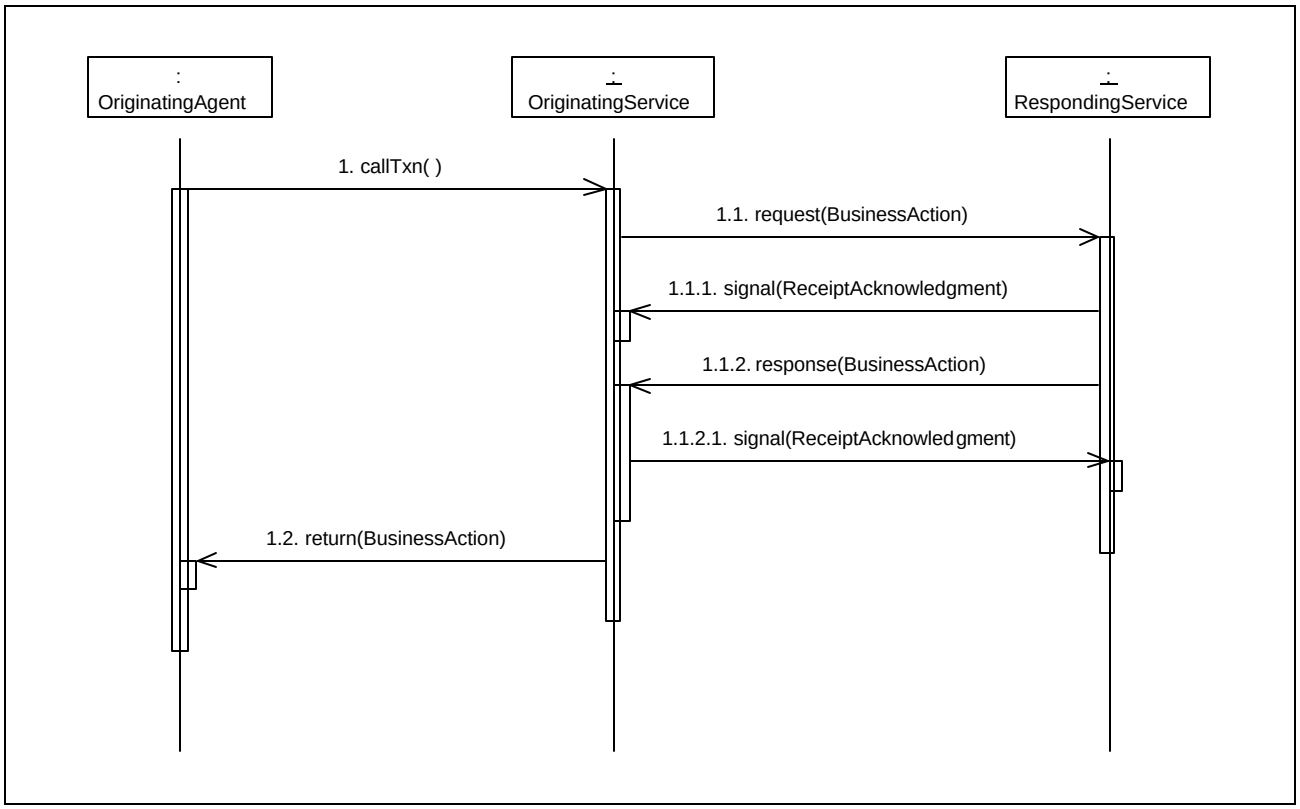

Figure 3.23 Agent-Service-Service Interaction Pattern—IV

Information Distribution Activity and Notification Activity

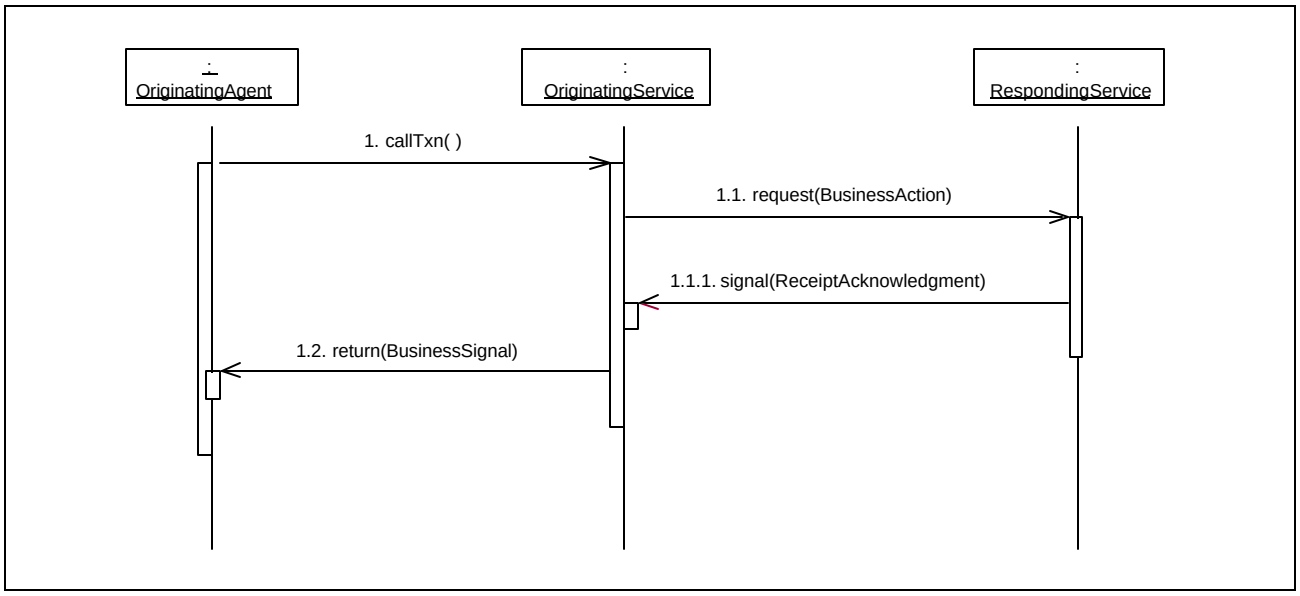

Figure 3.24 Agent-Service-Service Interaction Pattern—V

3.3.3 Service-Service-Agent

Business Transaction Activity

Time to perform equals time to acknowledge acceptance and no responding business document.

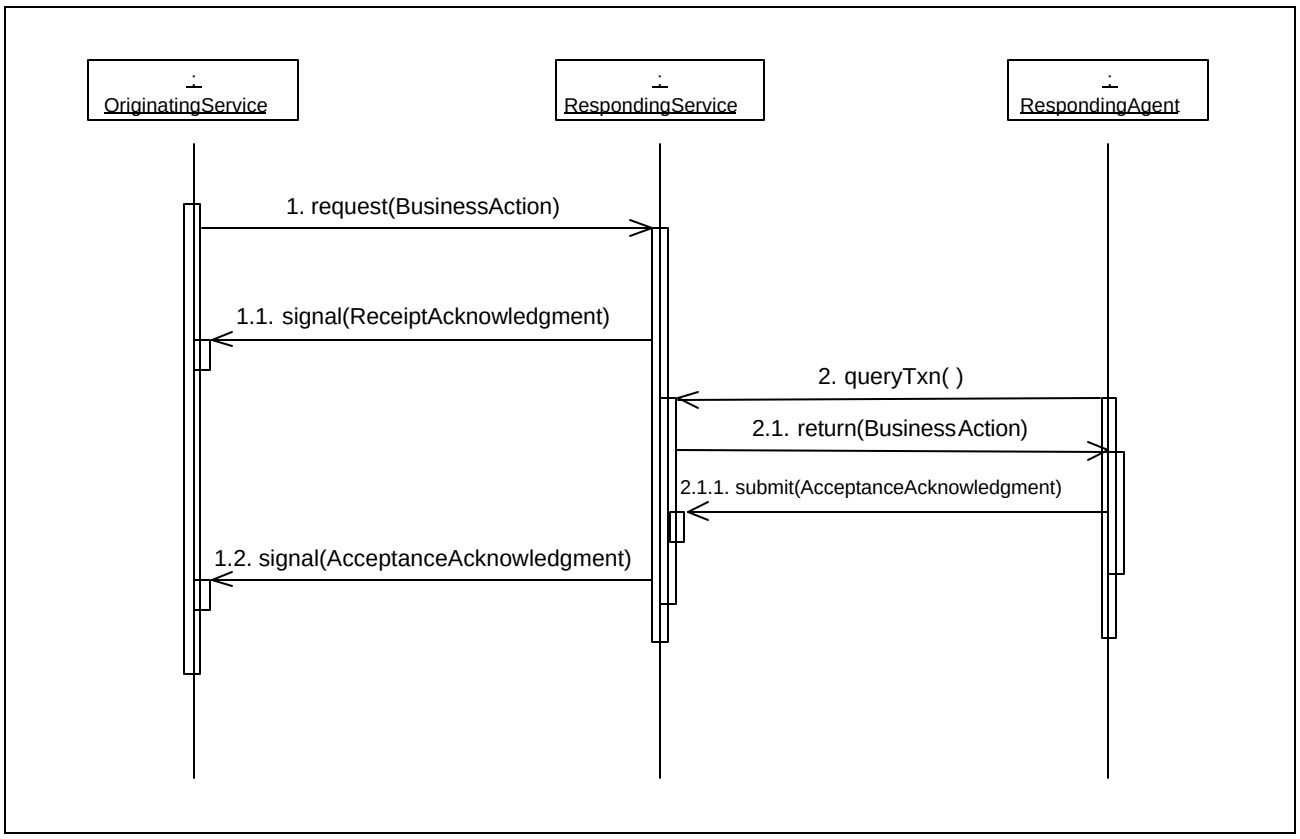

Figure 3.25 Service-Service-Agent Interaction Pattern—I

Time to perform equals time to acknowledge acceptance and a responding business document.

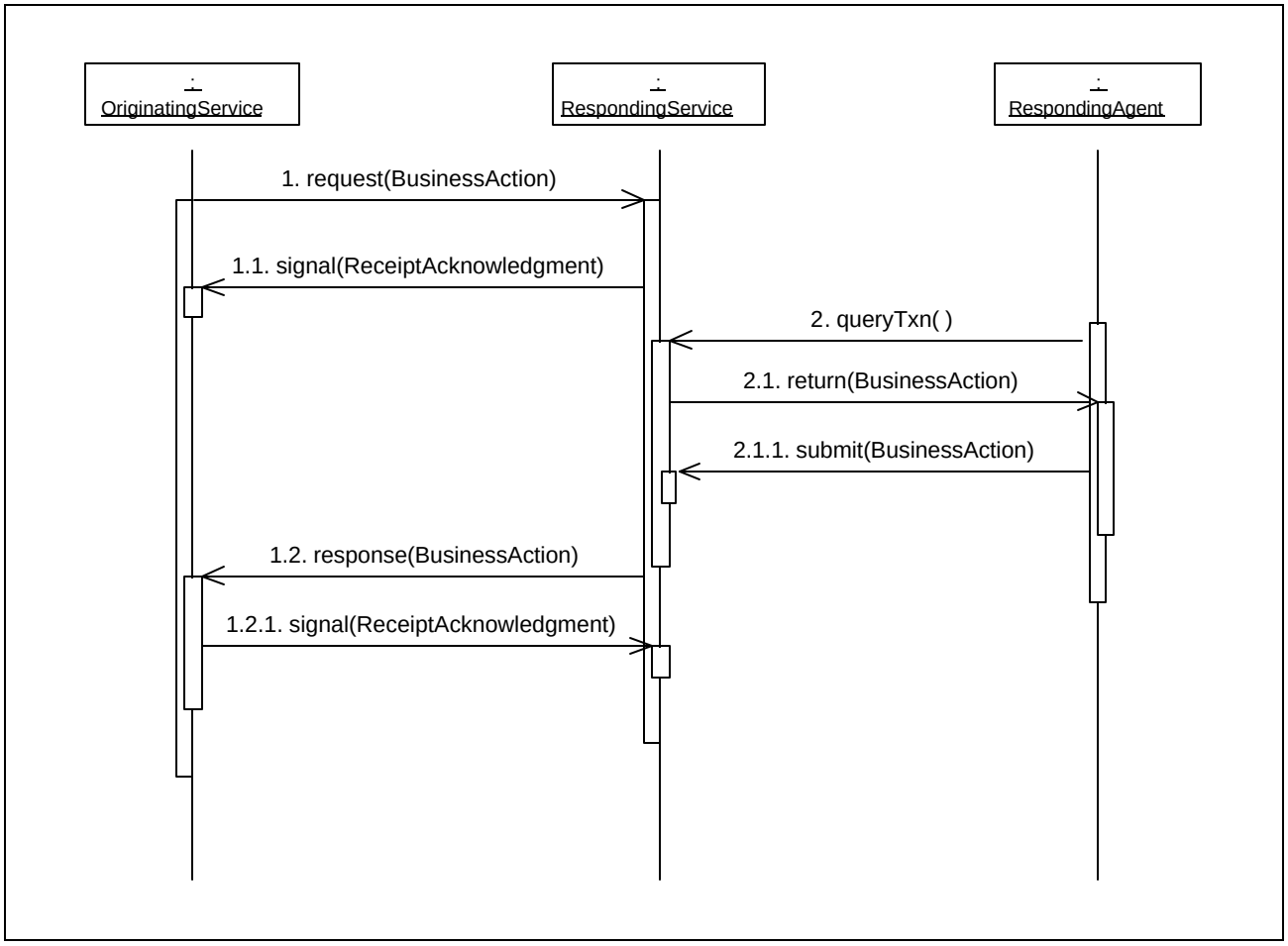

Figure 3.26 Service-Service-Agent Interaction Pattern—II

Time to perform is greater than time to acknowledge acceptance.

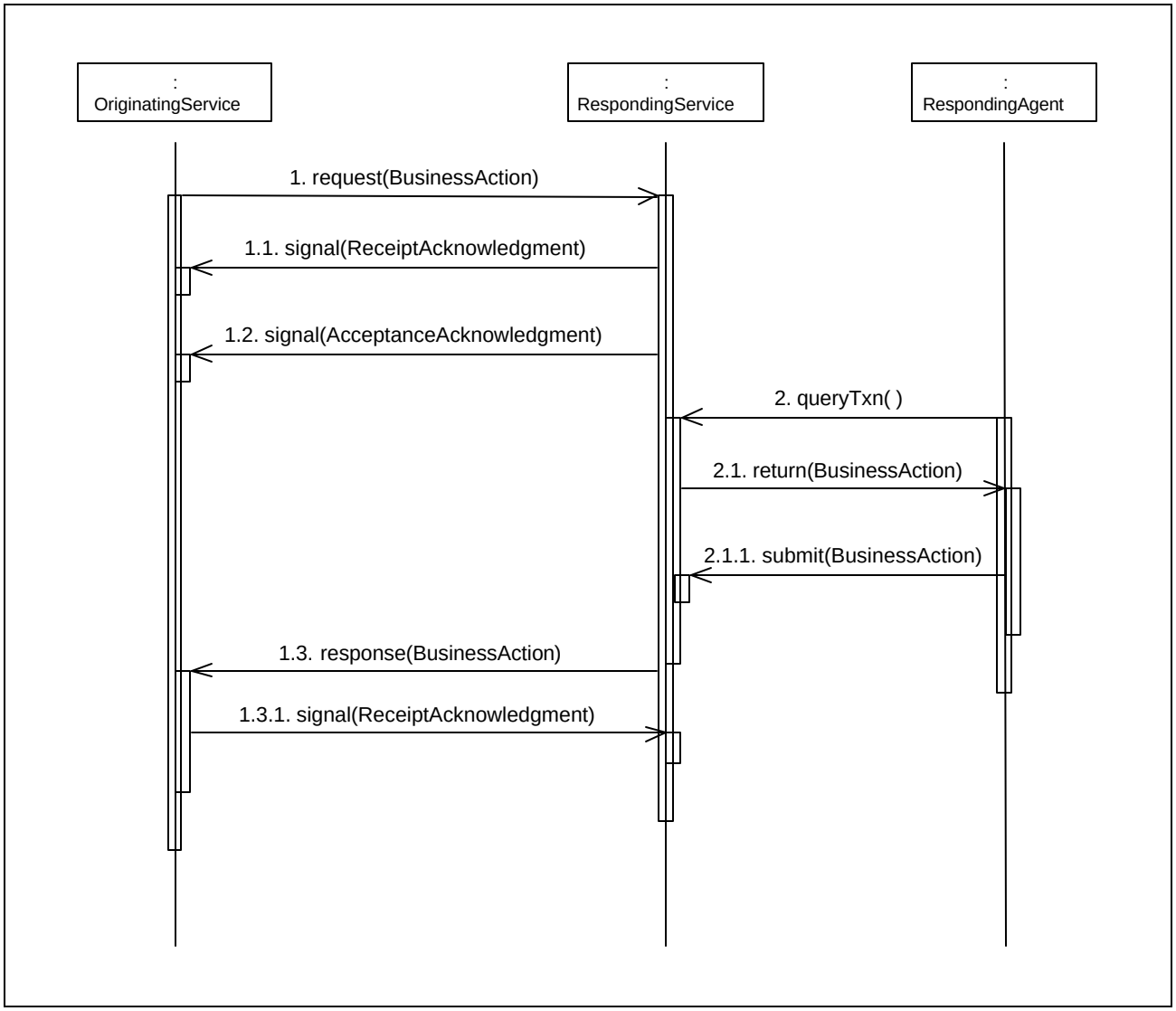

Figure 3.27 Service-Service-Agent Interaction Pattern—III

Query/Response Activity, Request/Response Activity and Request/Confirm Activity

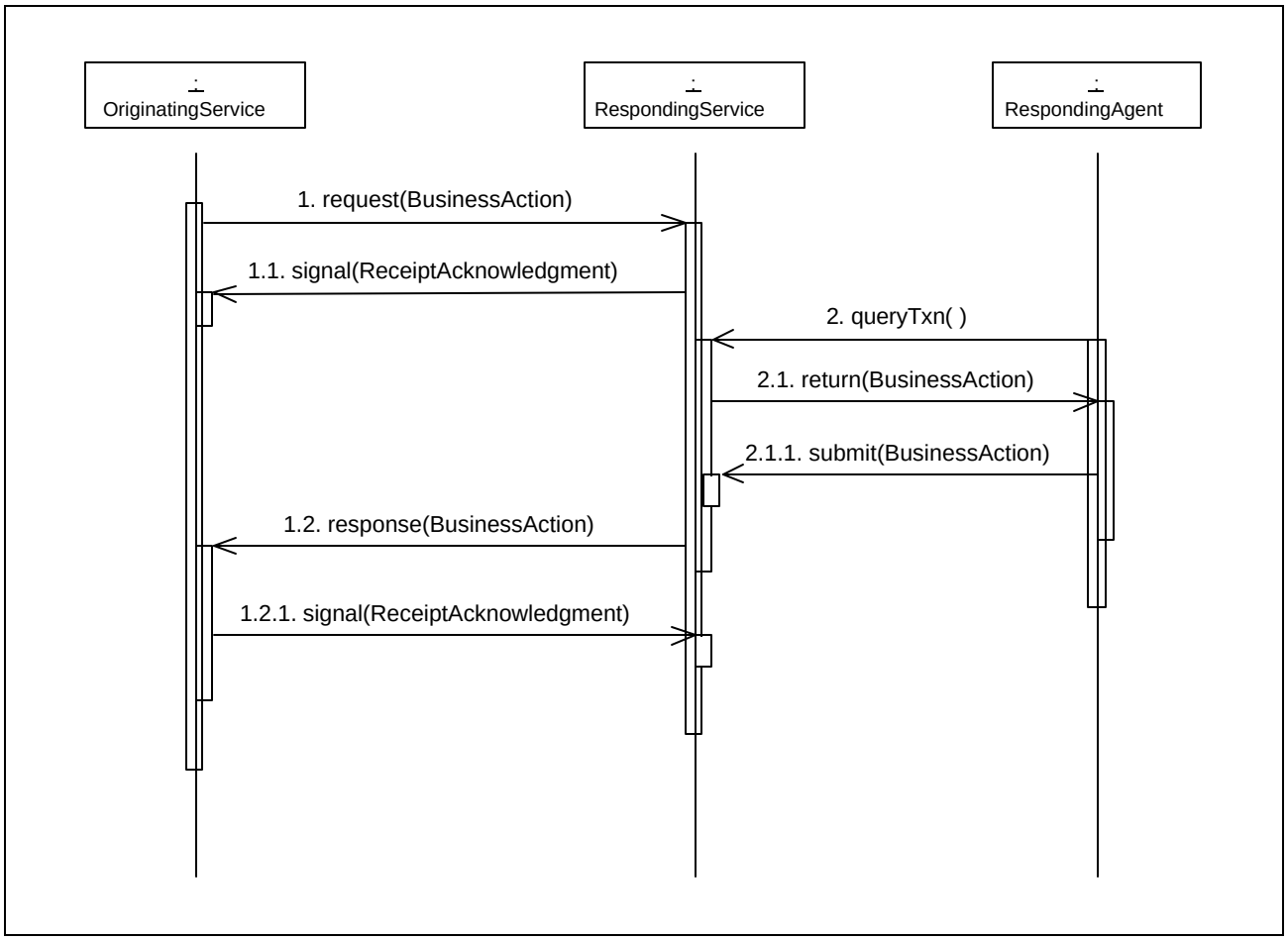

Figure 3.28 Service-Service-Agent Interaction Pattern—IV

Information Distribution Activity and Notification Activity

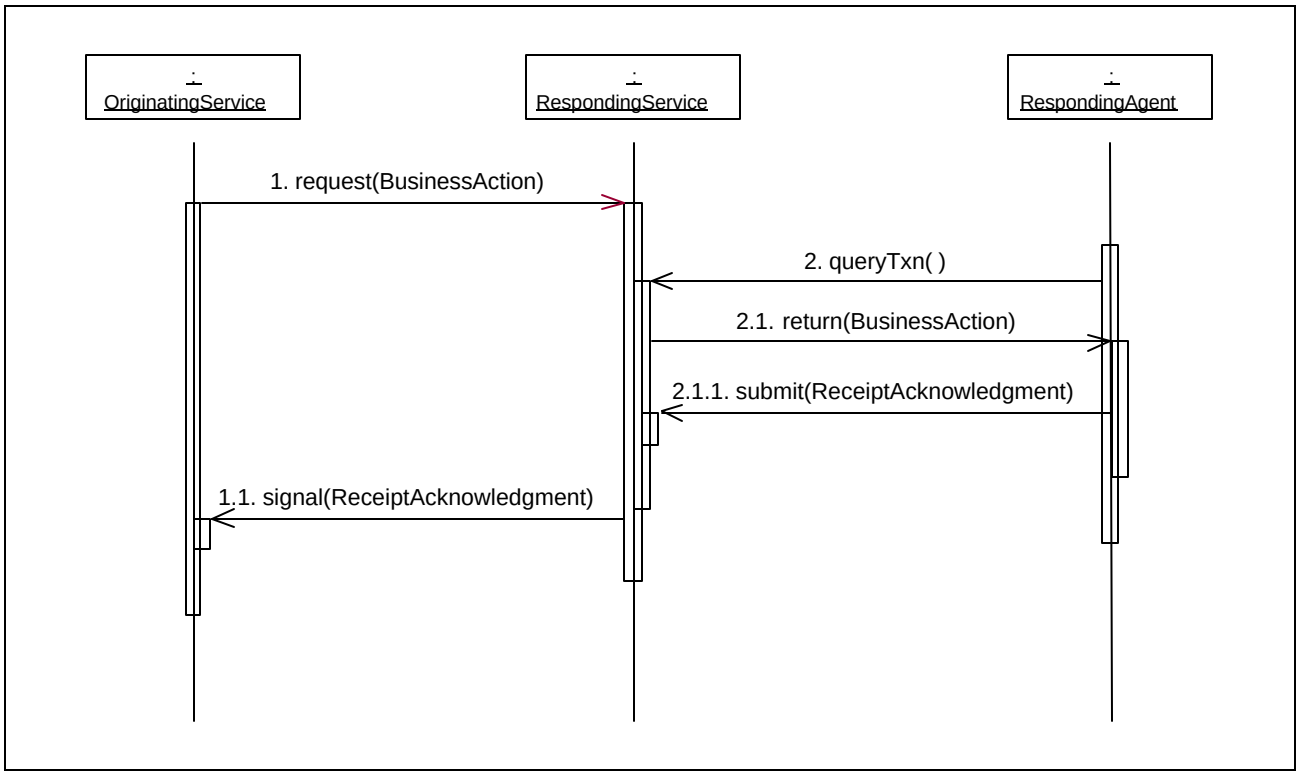

Figure 3.29 Service-Service-Agent Interaction Pattern—V

3.3.4 Service-Agent-Service

Business Transaction Activity

Time to perform equals time to acknowledge acceptance and no responding business document.

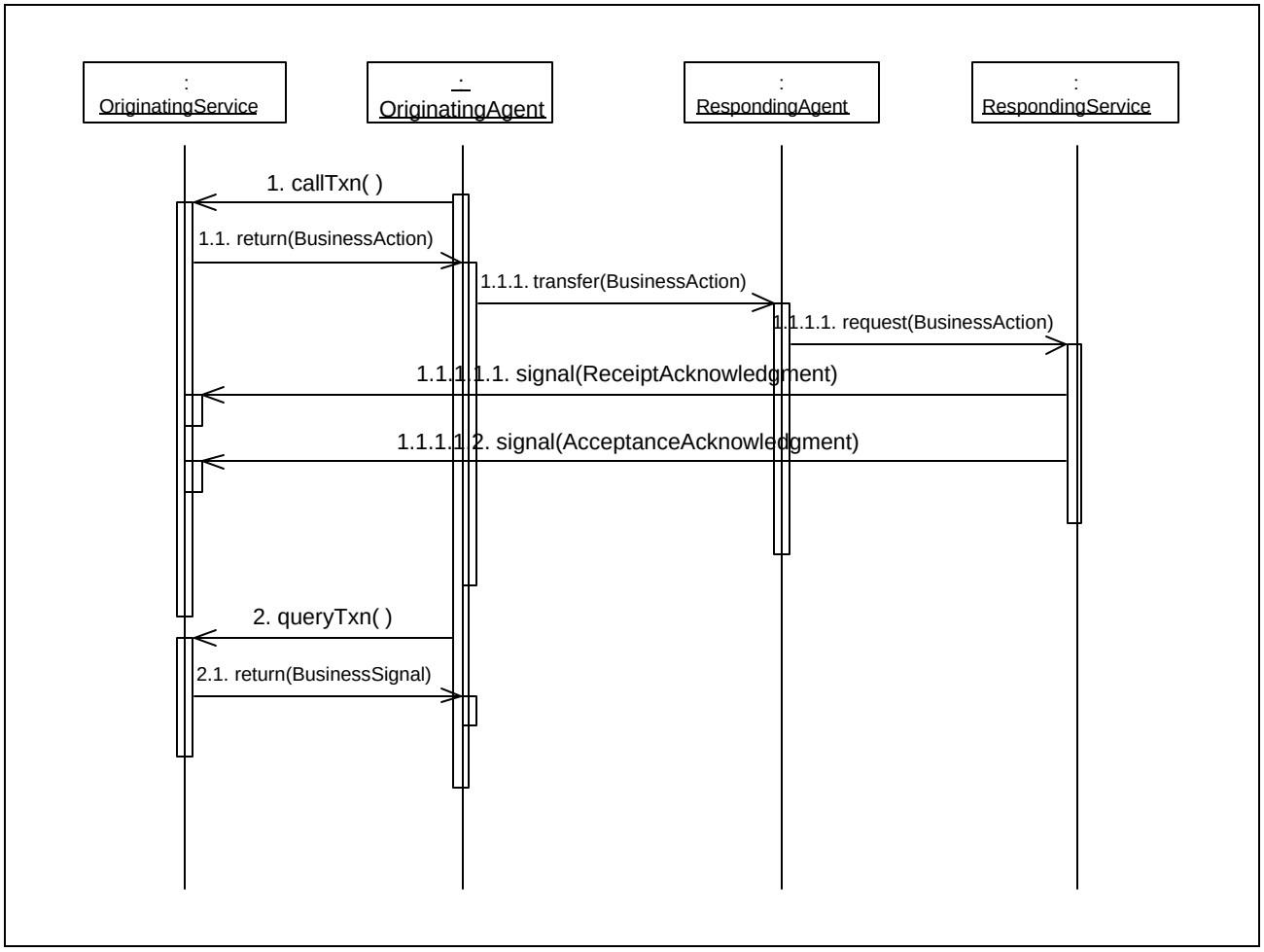

Figure 3.30 Service-Agent-Service Interaction Pattern—I

Time to perform equals time to acknowledge acceptance and a responding business document.

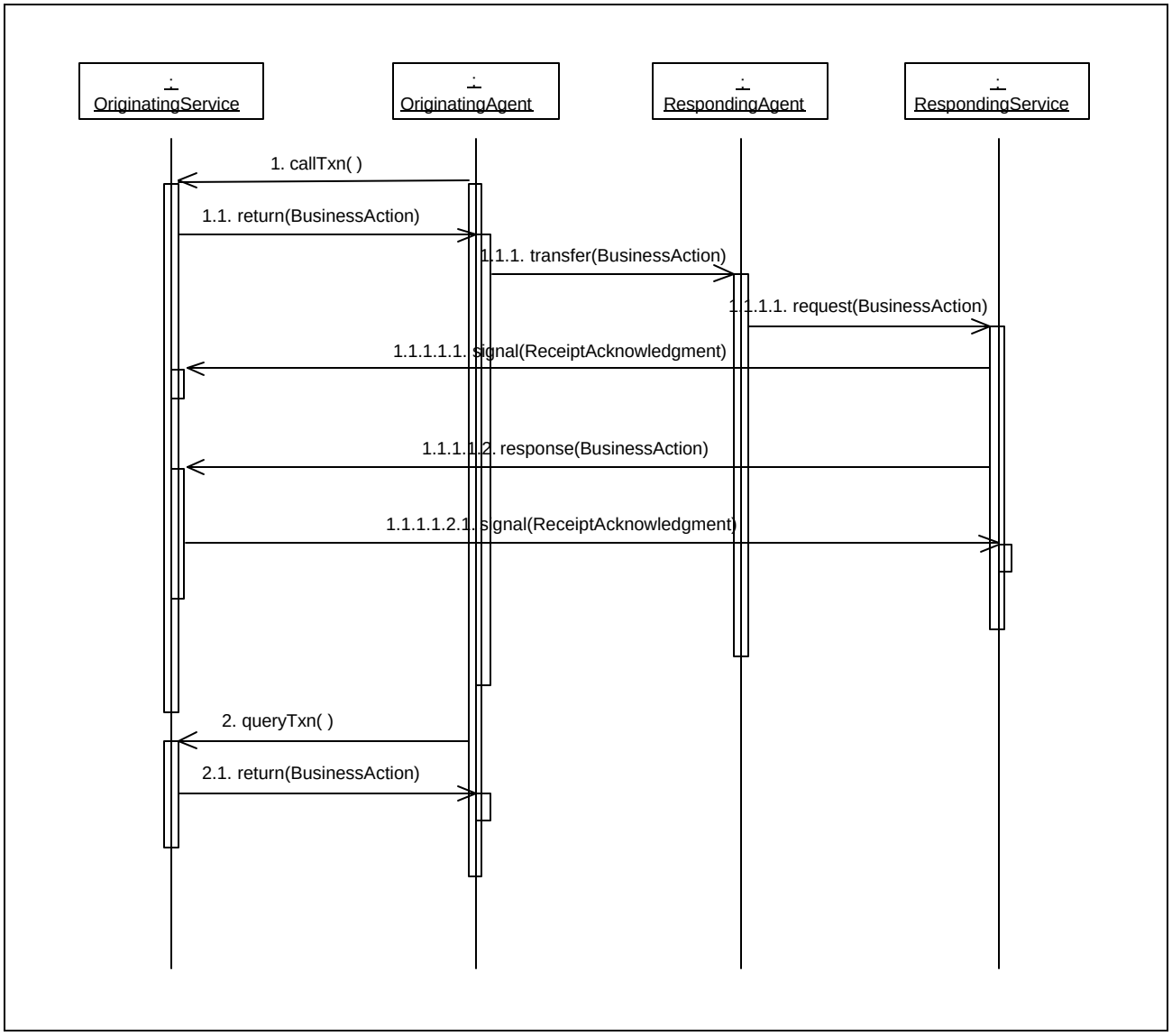

Figure 3.31 Service-Agent-Service Interaction Pattern—II

Time to perform is greater than time to acknowledge acceptance.

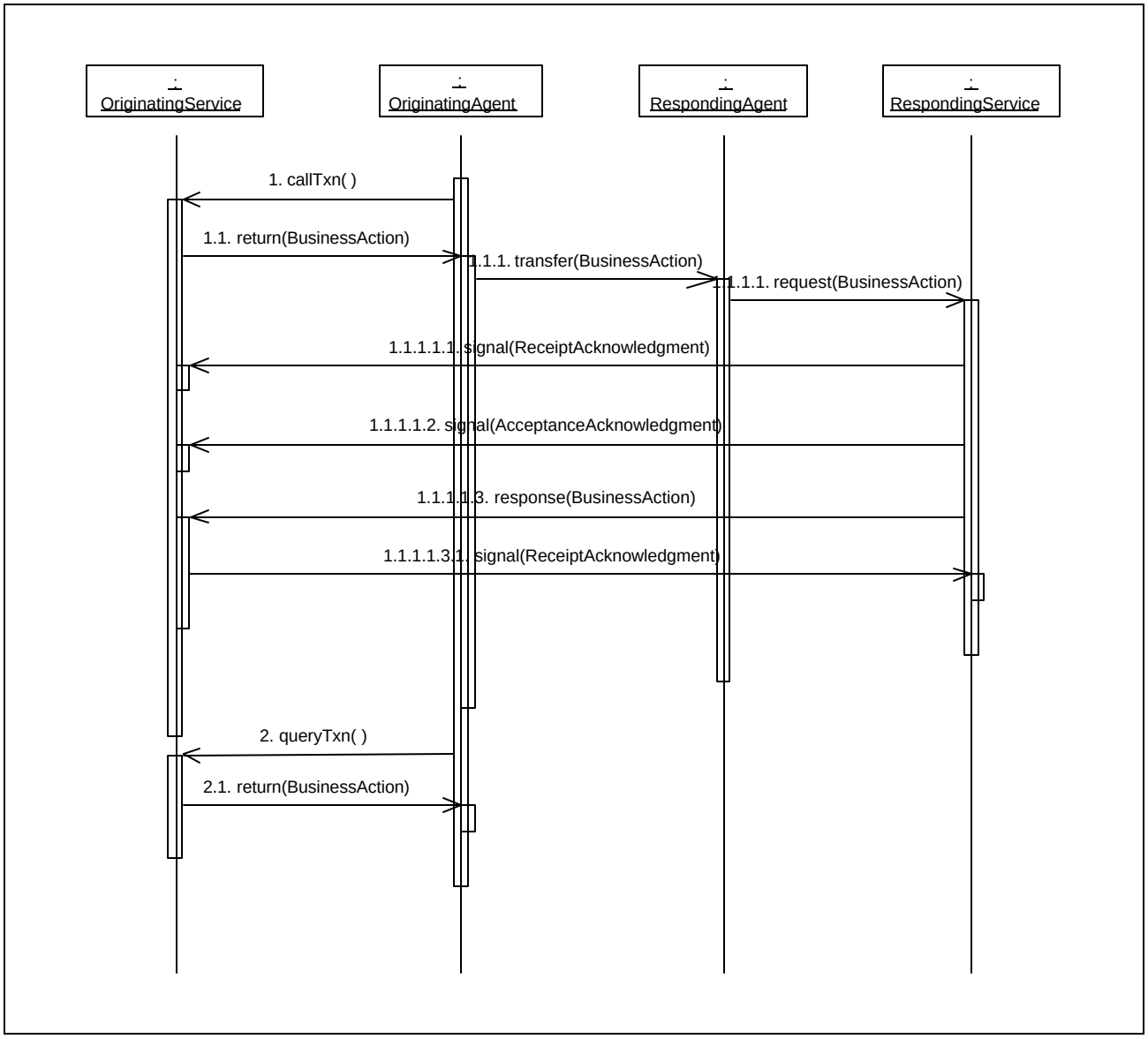

Figure 3.32 Service-Agent-Service Interaction Pattern—III

Query/Response and Request/Response Activity

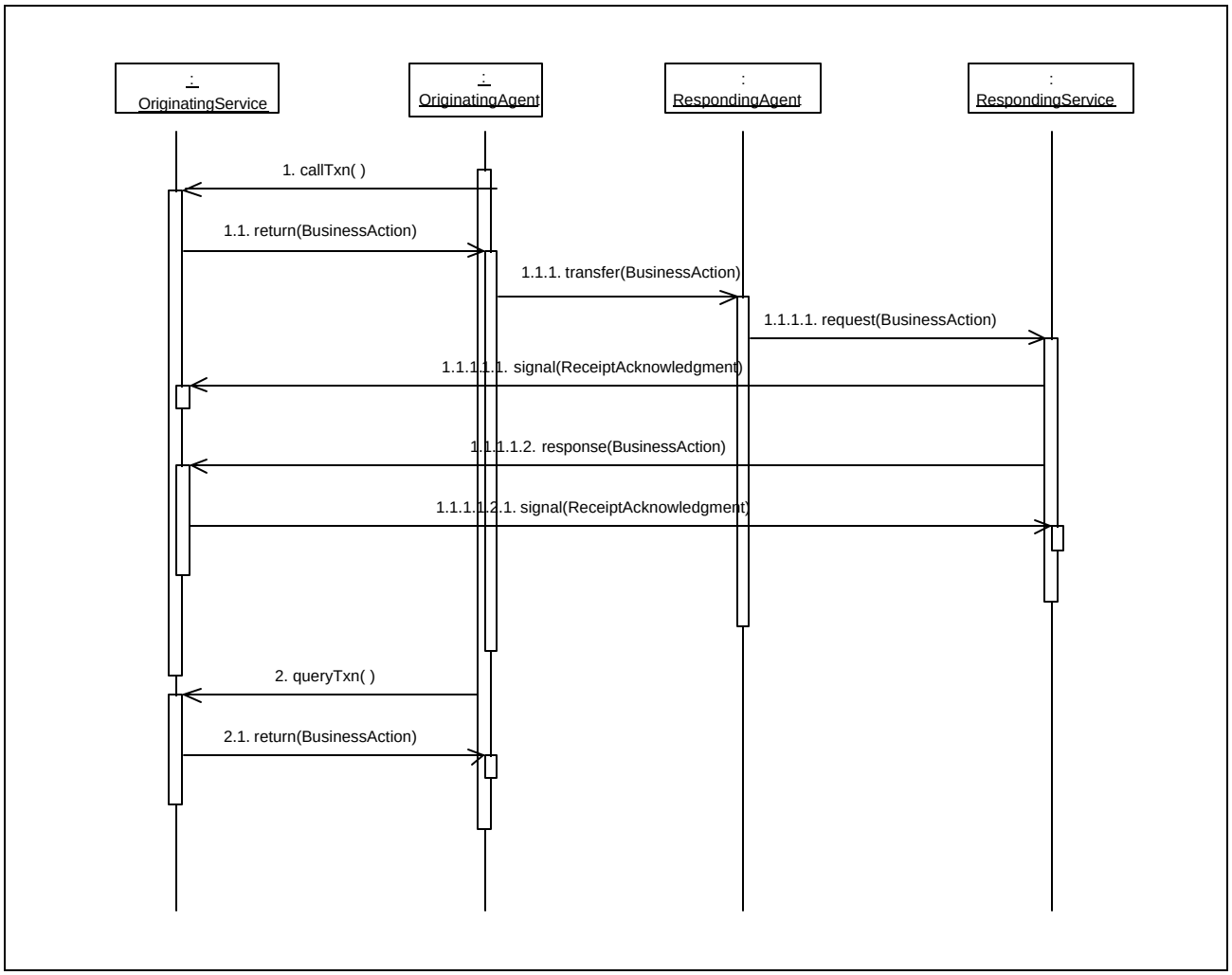

Figure 3.33 Service-Agent-Service Interaction Pattern—IV

Request/Confirm Activity

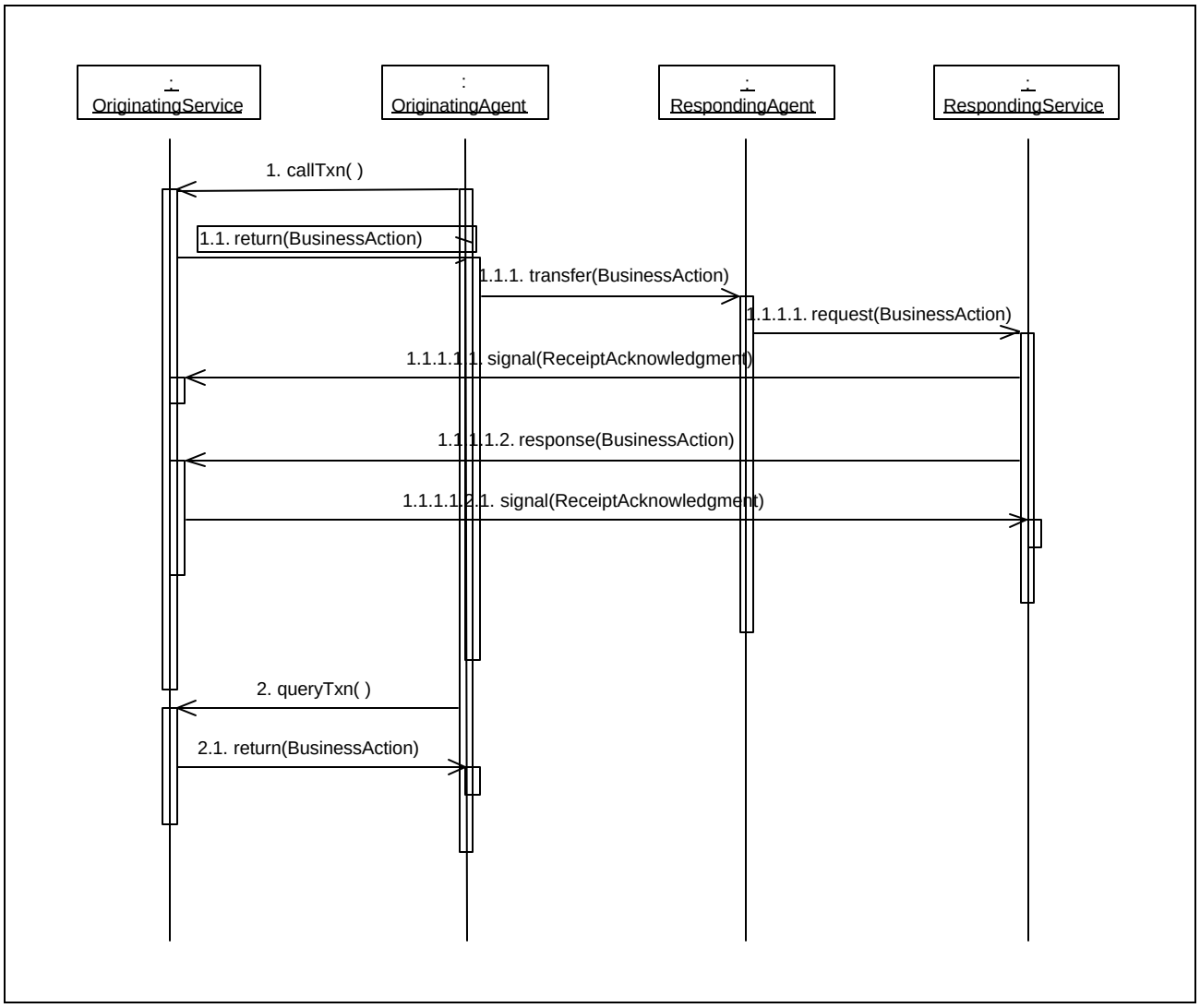

Figure 3.34 Service-Agent-Service Interaction Pattern—V

Information Distribution Activity and Notification Activity

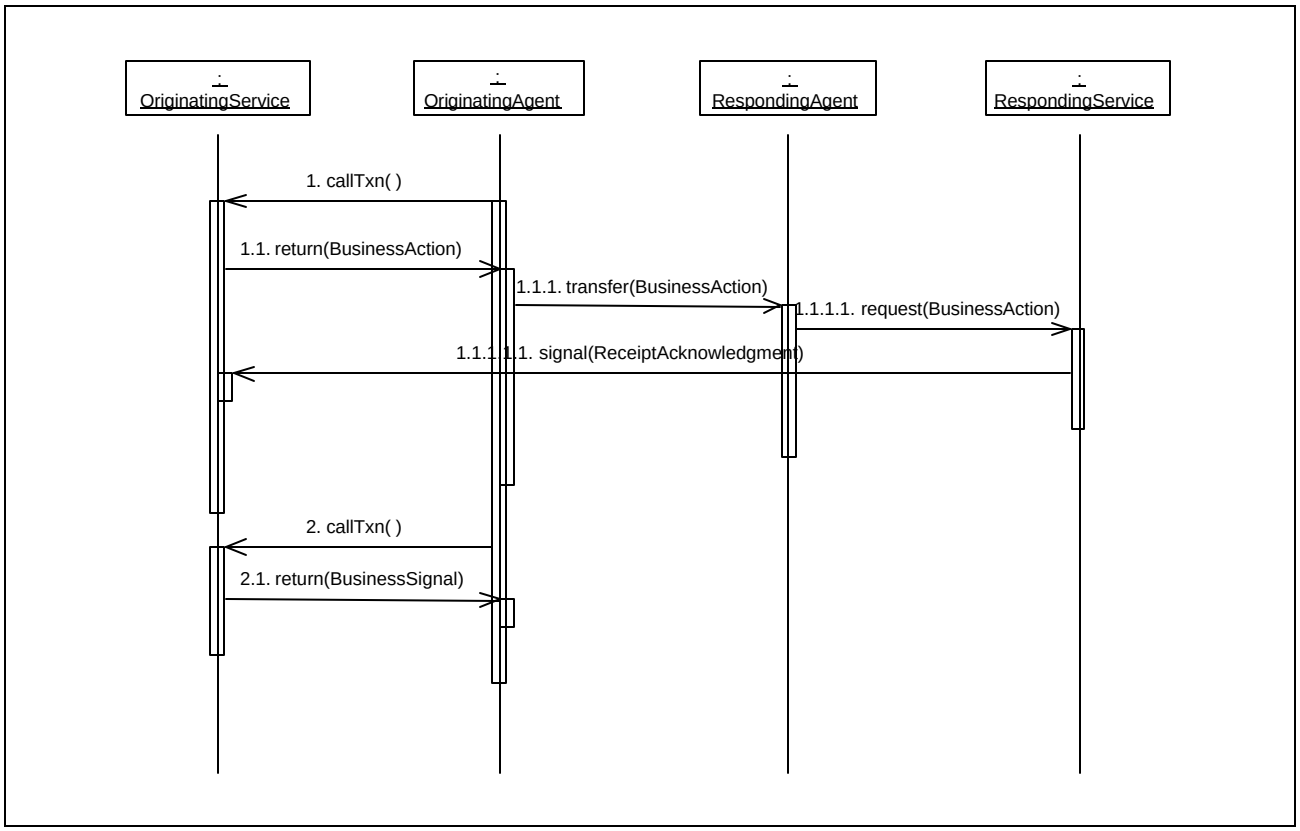

Figure 3.35 Service-Agent-Service Interaction Pattern—VI

3.3.5 Agent-Service-Agent

Business Transaction Activity

Time to perform equals time to acknowledge acceptance and no responding business document.

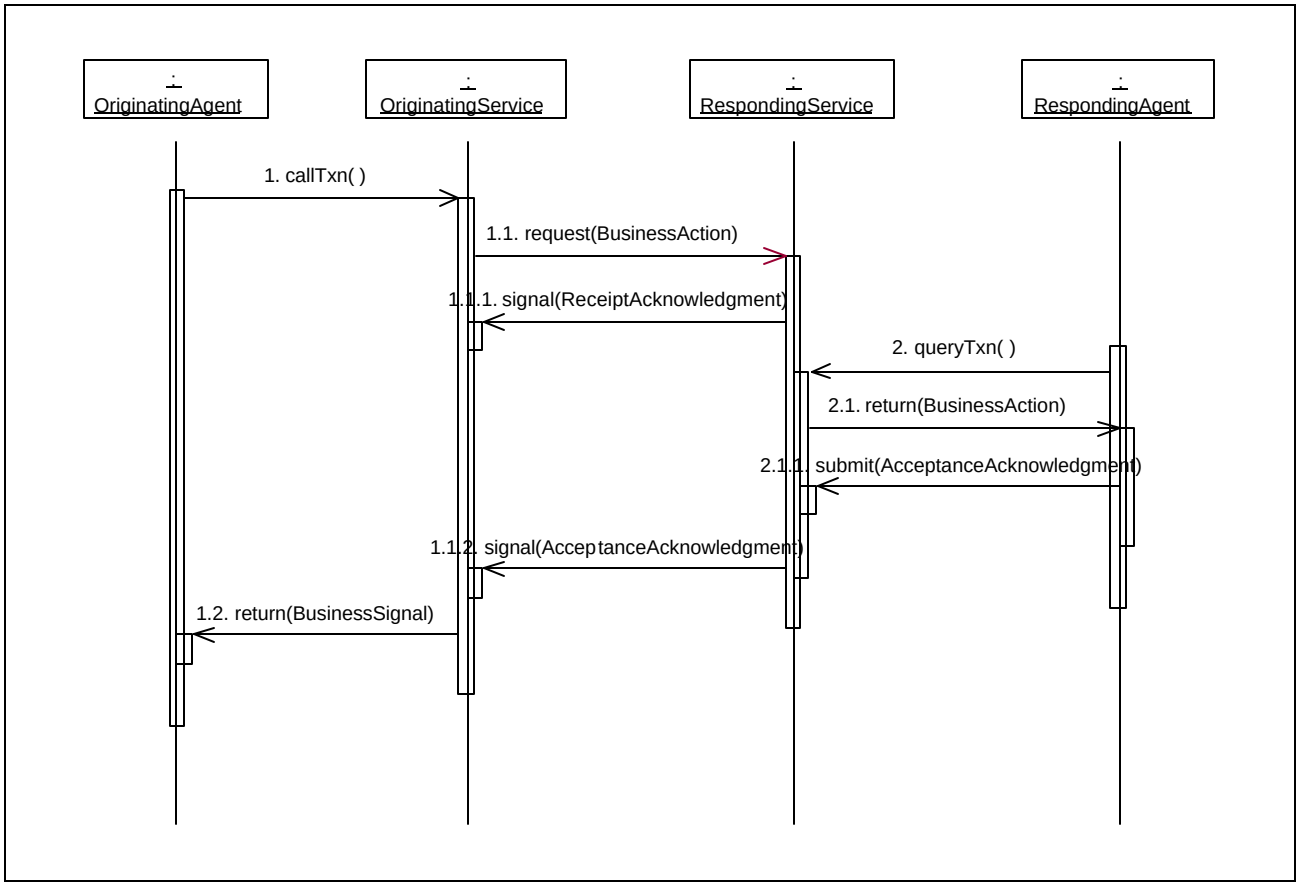

Figure 3.36 Agent-Service-Agent Interaction Pattern—I

Time to perform equals time to acknowledge acceptance and a responding business document.

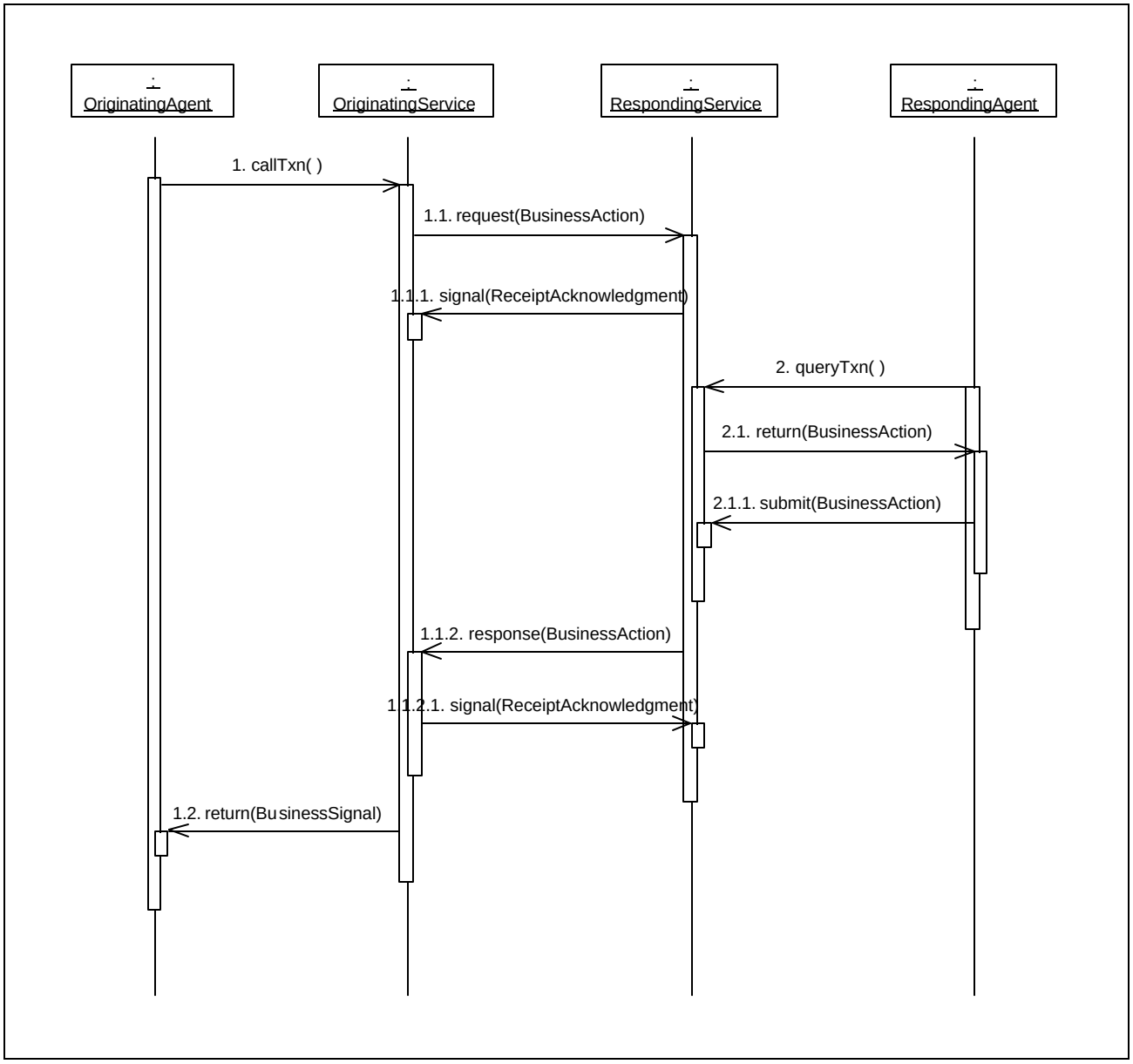

Figure 3.37 Agent-Service-Agent Interaction Pattern—II

Time to perform is greater than time to acknowledge acceptance.

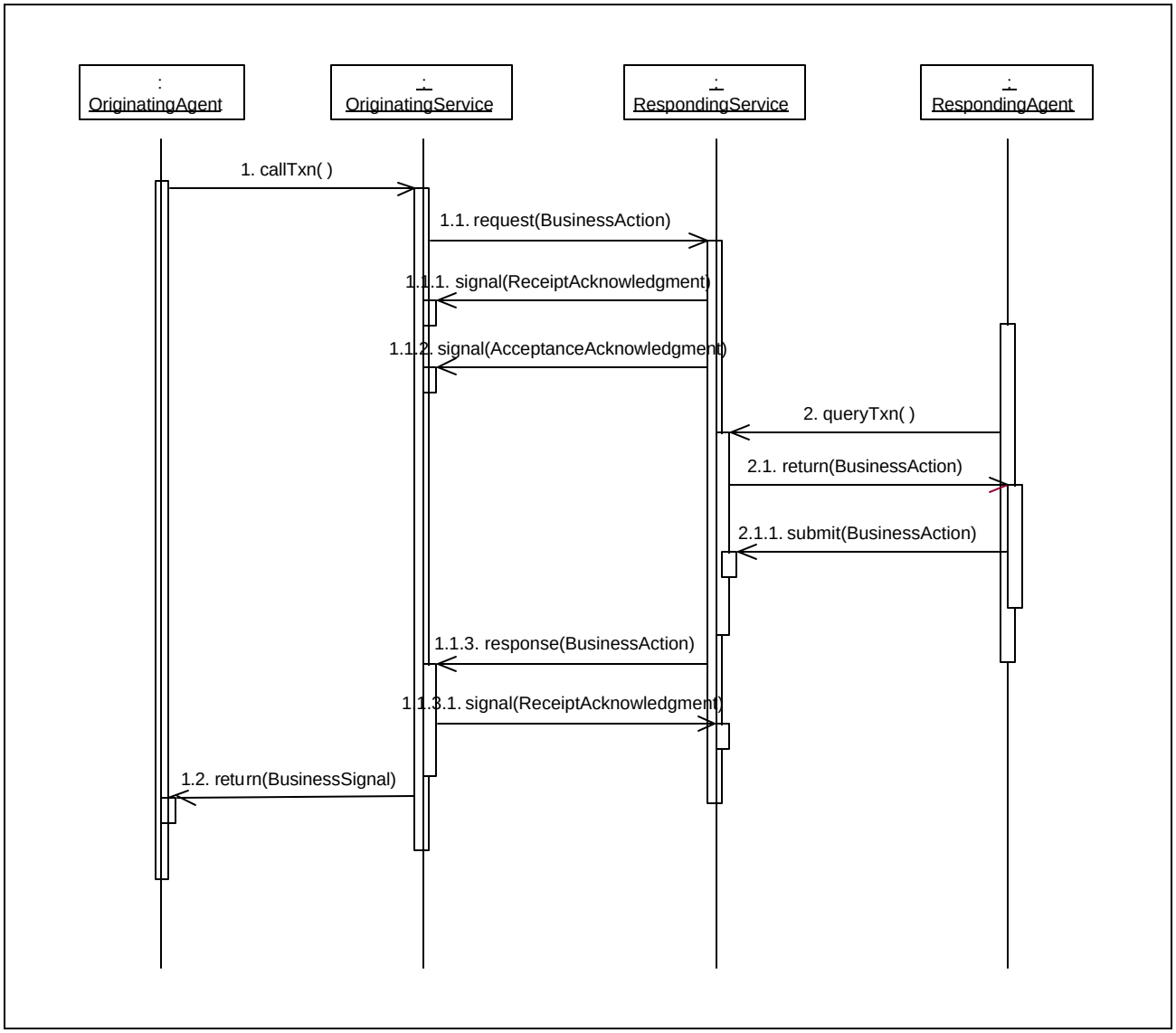

Figure 3.38 Agent-Service-Agent Interaction Pattern—III

Query/Response Activity, Request/Response Activity and Request/Confirm Activity

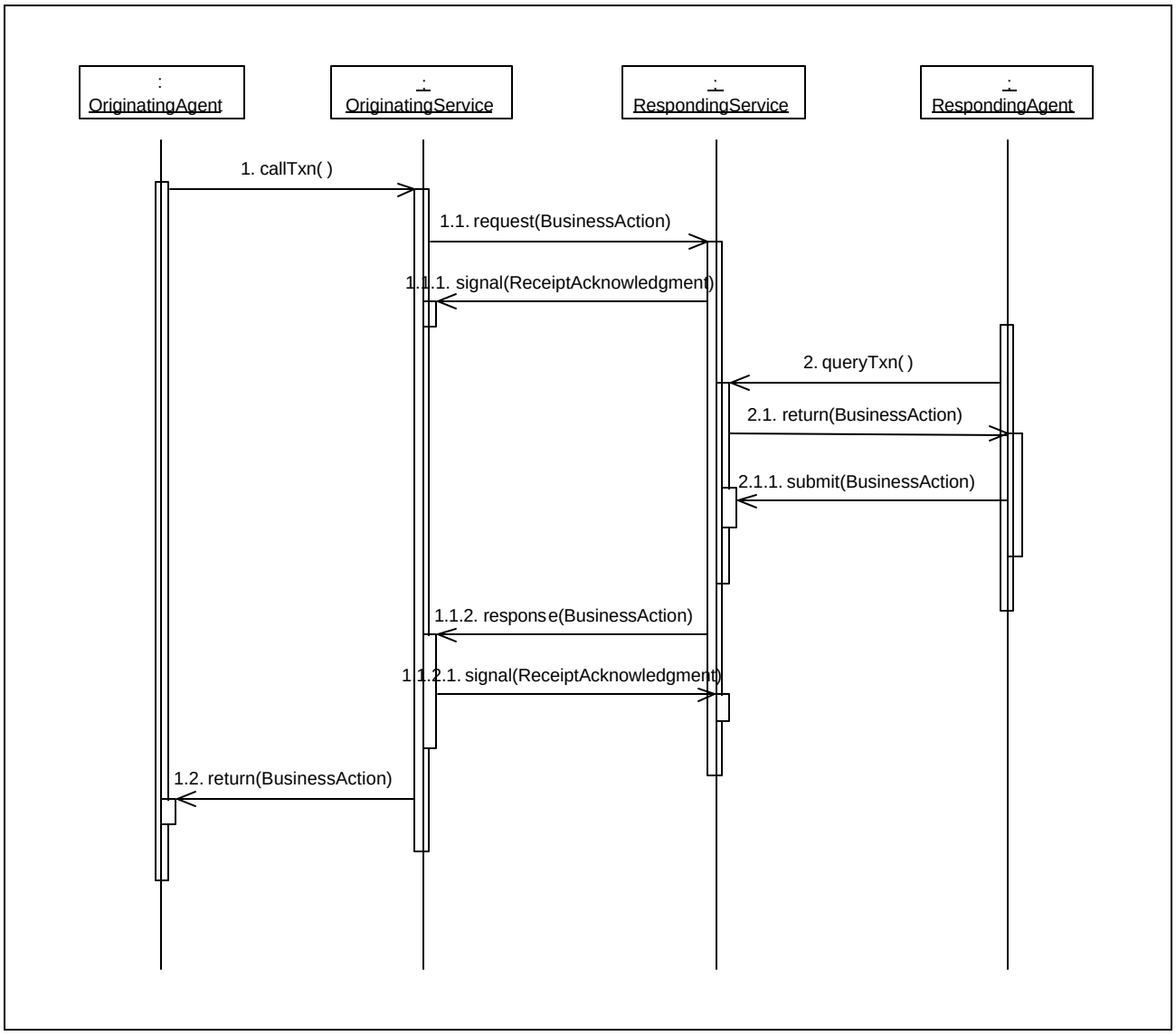

Figure 3.39 Agent-Service-Agent Interaction Pattern—IV

Information Distribution Activity and Notification Activity

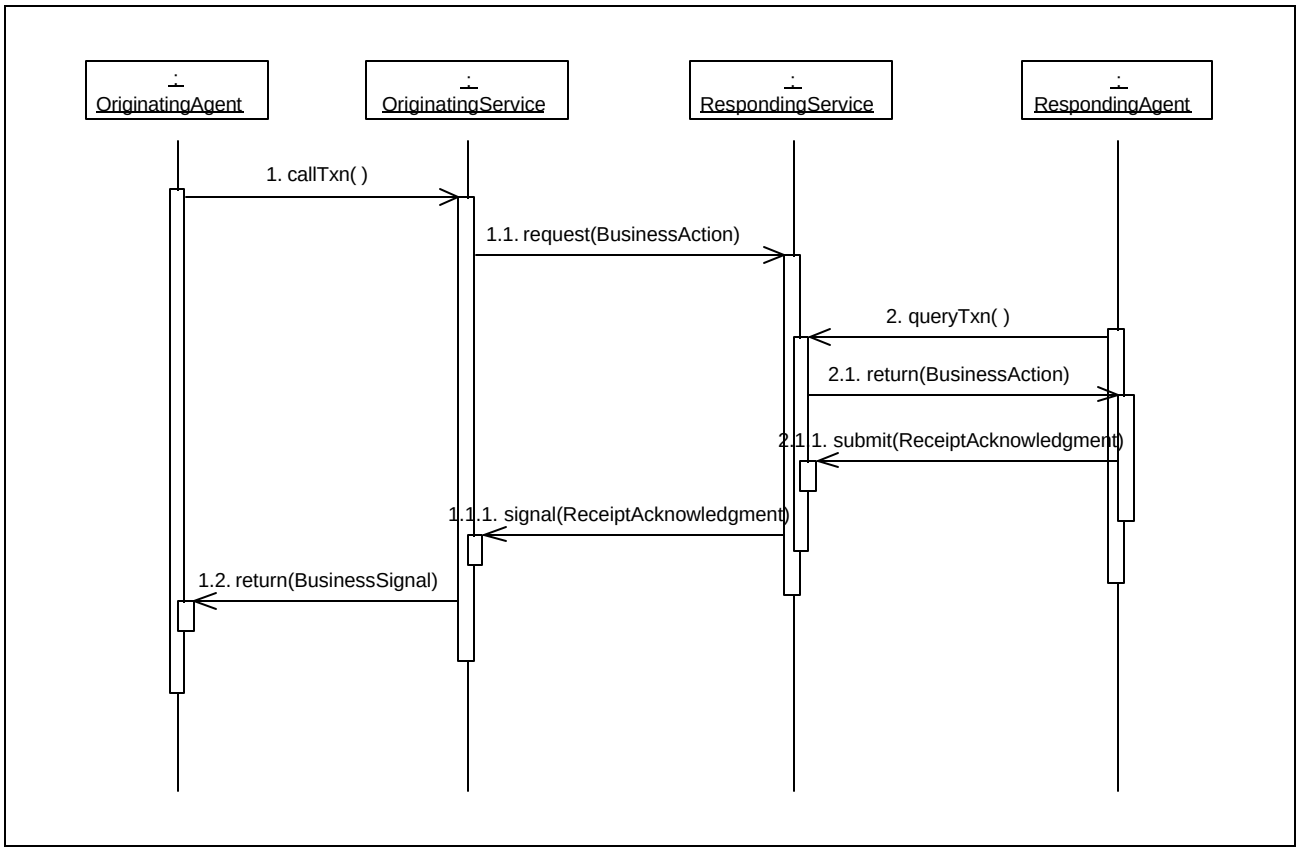

Figure 3.40 Agent-Service-Agent Interaction Pattern—V

ppendix I: Example ommercial ontract ormations

The following commercial contract formations are taken from:

"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."

Example1:

XYZ has specified its mainframe computer as its Receipt Computer. ABC sends a Document to XYZ's Provider, but the Document is never made accessible (fails to pass message schema validation in RNIF[8]) to XYZ's Receipt Computer. ABC's transmission of the legal document has no legal effect.

Example 2:

XYZ properly receives a purchase order from ABC but never transmits (transfer layer requirement, i.e. HTTP request must receipt a corresponding 200—OK response in RNIF) in return either a functional ACKNOWLEDGMENT (ACKNOWLEDGMENT of receipt) or an Acceptance Document (partners agree to the nature of an Acceptance Document[9], figures below specify this agreements as "closing message"). No contract has been formed but XYZ is liable for any damages suffered by ABC, if any, from XYZ's failure to provide verification as required (the HTTP response from posting this document or signal must be 200 otherwise a business user at XYZ should be notified of this failure to transmit).

[8] Partners should be aware of the fact that passing grammar validation with respect to a RN DTD is insufficient to claim "Readability" as there could still be many, many errors in both content and structure. It is recommended that Partners be made aware of this issue when signing their TPAs.

[9] The Acceptance Document is either a non-substantive acknowledgement of acceptance, a substantive acknowledgement of acceptance or a post-processing business document, e.g. shipping order returned to Accept a purchase order.

Example 3:

XYZ properly receives (acknowledges receipt) a purchase order from ABC by which its terms are open for 10 days. XYZ properly transmits an Acceptance Document within the 10 day period, but the Acceptance Document is not "properly received" (passes message schema validation[10]) until the 11th day. No contract is formed. (Note that the contract is properly formed upon proper receipt of an Acceptance Document and not on the verification of receipt of the Acceptance Document. Hence the BOV does not specify the verification of receipt. Note that the requirement to always verify receipt of a Business Document as the FSV allows the accepting partner to know that the contract has or has not been properly formed so that they can take action accordingly.)

Example 4:

The Appendix (to the TPA) requires, as to a purchase order, that a purchase order ACKNOWLEDGMENT (Business Document in RN PIP specification) be sent as an Acceptance Document. ABC, as buyer, send a purchase order, receipt, of which is verified by XYZ, as seller, by sending a functional ACKNOWLEDGMENT. However XYZ never sends an Acceptance Document. No contract for sales has been formed.

Example 5:

XYZ properly transmits an Acceptance Document, which is received by XYZ's Provider (VAN) and stored. Meanwhile, ABC properly transmits a revocation of its offer, which revocation is properly received by XYZ's Receipt Computer before the Acceptance Document is forwarded to ABC's Receipt Computer by XYZ's provider. No contract is formed; the revocation is effective.

Example 6:

The Appendix (of the TPA) requires, as to a purchase order, that a purchase order ACKNOWLEDGMENT (substantive ACKNOWLEDGMENT of acceptance Business Document in RN PIP specification) be sent as an Acceptance Document. XYZ, as seller, properly receives a purchase order from ABC, as buyer, but the price data is missing. XYZ send a functional ACKNOWLEDGMENT, which identifies the omitted data (RN does not guarantee that a DTD will always be able to perform these types of checks due to contextual conditional composition forcing a DTD element's cardinality specification to be different to a message guideline element's cardinality specification, e.g. see contact information in fromRole and toRole). Under Section 2.4 (or reference 3 above), XYZ has met its obligations. If XYZ, without the price data, then sends an Acceptance Document, a contract is formed, with the price to be determined pursuant to applicable law. See UCC (Uniform Commercial Code governing commercial contacts in the USA) 2-305.

---

[10] Same as note 1.

ppendix II: nderstanding ommercial ontracts using 12/EDI

The following examples show how EDI/X12 message guidelines can be used to implement the commercial transactions described in Appendix I. These examples are provided to assist business analysts, familiar with EDI/X12 terminology, translate into the collaboration modeling terminology used in this document.

Example 1:

1. Initiating party sends PO (850).
2. Responding party sends functional ACKNOWLEDGMENT (997).
   - timeToPeform = timeToAcknowlegeRecept = "x"
   - timeToAcknowlegeAcceptance = N/A

Example 2:

1. Initiating party sends PO (850).
2. Responding party sends functional ACKNOWLEDGMENT (997).
3. Responding party sends PO ACKNOWLEDGMENT that substantively acknowledges acceptance (contains product identification and quantity) the content of the PO (855).
   - timeToAcknowlegeRecept="x"
   - timeToPeform = timeToAcknowlegeAcceptance = "y"

Example 3:

1. Initiating party sends PO (850).
2. Responding party sends functional ACKNOWLEDGMENT (997).
3. Responding party sends PO ACKNOWLEDGMENT that non-substantively acknowledges acceptance of the content of the PO (824)
   - timeToAcknowlegeRecept="x"
   - timeToPeform = timeToAcknowlegeAcceptance = "y"

Example 4:

1. Initiating party sends PO (850).
2. Responding party sends functional ACKNOWLEDGMENT (997).
3. Responding party sends shipping notice (856).
   - timeToAcknowlegeRecept="x"
   - timeToAcknowlegeAcceptance = N/A, timeToPeform = "y"

Example 5:

1. Initiating party sends PO (850).
2. Responding party delivers the goods e.g. on-line software.
   - timeToAcknowlegeRecept = N/A,
   - timeToAcknowlegeAcceptance = N/A, timeToPeform = N/A

Example 6:

1. Initiating party sends PO (850).
2. Responding party sends functional ACKNOWLEDGMENT (997).
3. Responding party sends PO acknowledges that non-substantively acknowledges acceptance of the content of the PO (824)
4. Responding party sends shipping notice (856).
   - timeToAcknowlegeRecept="x"
   - timeToAcknowlegeAcceptance = "y," timeToPeform = "z"