Coordinator-Worker-Context Process Pattern

John Liebenau
Capital Group Companies Inc.
USA
jfl@capgroup.com

Abstract

This paper describes the *Coordinator-Worker-Context* process pattern—a pattern for designing processes that contain embedded elements responsible for ensuring proper execution and coordination. *Coordinator-Worker-Context* addresses some of the problems that occur in complex processes involving both human workers and automated systems.

Categories and Subject Descriptors

D.2.11 [Software Engineering]: Software Architectures—

Patterns

General Terms

Design

Keywords

business automation, business modeling, business process, patterns, socio-technical systems

1. Intent

Ensure a process executes consistently by organizing process participants into a coordinator role that directs process execution and a set of worker roles that perform process tasks. Artifacts and information produced and consumed during process execution are managed in a common context that enables communication and information transfer between process participants.

2. Motivation

A process can be defined by specifying: the workers that execute the process, the step-by-step activities that guide or direct the workers, and the artifacts that are inputs and outputs of the activities. A process worker embodies a set of skills, behaviors, and responsibilities which can be instantiated by individuals, groups, or automated systems.

Suppose that you are designing a process for your organization that will involve both human workers and automated systems. How will you ensure that the process is successfully implemented from your design and will be executed properly, both now and in the future? The focus of many process designs is the sequence of activities that make up the process and mapping those activities to the appropriate human workers and automated systems. This kind of process design is a good starting point but it exposes the process to several problems. Processes that involve human workers can be difficult to execute consistently because:

- Human workers can misinterpret the activities assigned to them. Detailed activities must be performed accurately in order for a process to execute properly but this becomes harder as the complexity of process activities increases:
  - Human workers may misunderstand the instructions of an activity and perform that activity incorrectly.
  - Over time human workers may forget the exact instructions of an activity by incrementally altering small elements until the alterations accumulate to a degree that affects the process.
  - Human workers are replaced by newer workers that may have less knowledge about the process causing them to perform activities incorrectly.
- Human workers may have difficulties coordinating who performs what activities. One way of managing the complexity of a large process is to break it down into smaller activities handled by specialized process workers but this shifts the complexity to coordinating process workers. Common issues include coordinating:
  - The correct execution sequence of activities between many process workers,
  - The parallel execution of activities between multiple process workers, and
  - The notification of critical events or state changes during process execution.

A better approach is to design elements into the process itself that will ensure proper execution and coordination. For example, you could specify a role that is responsible for coordinating workers that execute the process. This design separates the responsibilities of coordination from execution allowing coordinators to concentrate on monitoring the overall process while enabling workers to concentrate on their tasks. You could also specify a central work repository where artifacts and information could be stored, enabling easier collaboration between the coordinator and the workers. Together the coordinator role, worker roles, and work repository make up the Coordinator-Worker-Context process pattern and provide a reusable design for implementing many kinds of business processes.

Let us examine how an online bookstore could implement an order fulfillment process using the Coordinator-Worker-Context process pattern. An online bookstore sells books to customers through its web site. A customer accesses the web site and selects books to be purchased. The purchase order is created when the customer provides credit card and shipping information to the order page on the web site. The order is saved in the bookstore's order database and made available to various applications that will process the order. A billing application charges the customer's credit card and verifies that processing may continue. An inventory application checks warehouse inventory and notifies a packaging clerk of the order. The clerk packages the order for shipment, registers package contents with the inventory application, and sends the package to shipping. The shipping clerk sends all outgoing packages to the delivery company partnered with the bookstore. The delivery company delivers the package to the customer. The major components of the bookstore's order fulfillment process are shown in figure 1 (see section 13 for notation guide).

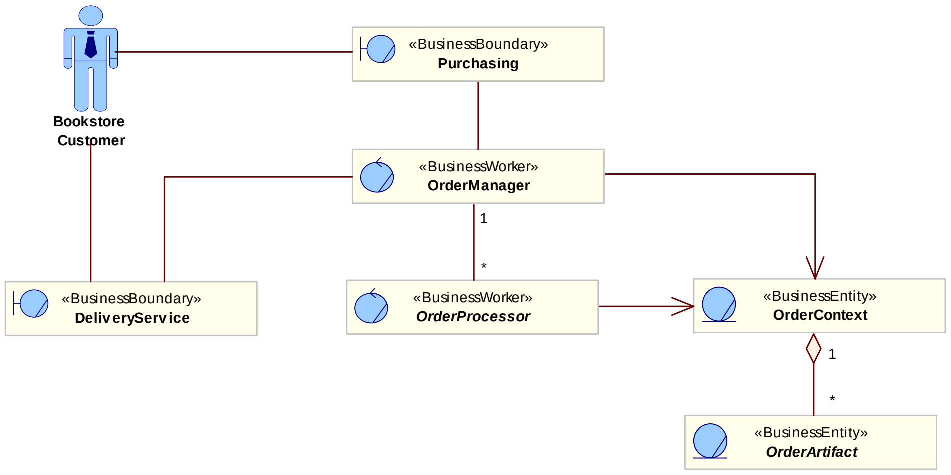
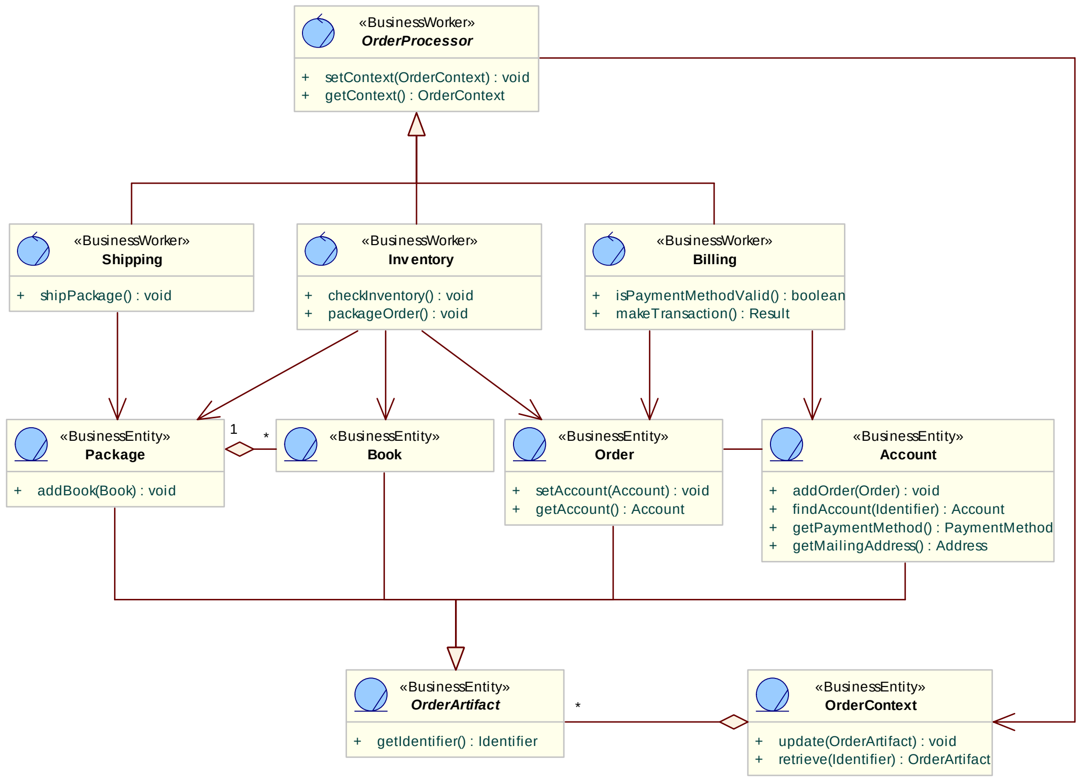

Bookstore Customer
«BusinessBoundary» Purchasing
«BusinessWorker» OrderManager
1
*
«BusinessBoundary» DeliveryService
«BusinessWorker» OrderProcessor
«BusinessEntity» OrderContext
1
*
«BusinessEntity» OrderArtifact

Figure 1

The major components of the order fulfillment process include:

- Purchasing – A boundary of the bookstore and provides features that allow customers to make book orders and inquire about the status of existing orders.
- OrderManager – A worker that coordinates and monitors the work required to complete the book order. The OrderManager delegates activities to other workers.
- DeliveryService – A boundary of the delivery company partnered with the bookstore.
- OrderProcessors – Workers representing the bookstore's billing, inventory, and shipping capabilities all play a part in processing customer orders. These workers are notified by the OrderManager to start working on activities.
- OrderContext - A work context that tracks the state of the customer's order and other information needed to complete the order. The online bookstore has an order and inventory databases that keep track of customer orders and the inventory of books in the bookstore's warehouse.
- OrderArtifacts – Entities representing the informational and physical artifacts that are produced, consumed, and manipulated during the order fulfillment process.

Figure 2 expands the OrderProcessor and OrderArtifact components into hierarchies that show the specialized kinds of processors and the various kinds of artifacts that they manipulate in the order fulfillment process. Each OrderProcessor has an association with the OrderContext and uses that to update and retrieve OrderArtifacts .



«BusinessWorker» OrderProcessor
+ setContext(OrderContext) : void
+ getContext() : OrderContext
«BusinessWorker» Shipping
+ shipPackage() : void
«BusinessWorker» Inventory
+ checkInventory() : void
+ packageOrder() : void
«BusinessWorker» Billing
+ isPaymentMethodValid() : boolean
+ makeTransaction() : Result
«BusinessEntity» Package
+ addBook(Book) : void
1
*
«BusinessEntity» Book
«BusinessEntity» Order
+ setAccount(Account) : void
+ getAccount() : Account
«BusinessEntity» Account
+ addOrder(Order) : void
+ findAccount(Identifier) : Account
+ getPaymentMethod() : PaymentMethod
+ getMailingAddress() : Address
«BusinessEntity» OrderArtifact
+ getIdentifier() : Identifier
*
«BusinessEntity» OrderContext
+ update(OrderArtifact) : void
+ retrieve(Identifier) : OrderArtifact

Figure 2

The Coordinator-Worker-Context process pattern separates the elements responsible for process coordination from the elements responsible for executing process activities. Referring to the bookstore example, the OrderManager ensures that the order fulfillment process executes properly by coordinating the activities of automated and human workers that make up the Billing, Inventory, and Shipping capabilities of the bookstore. Customer Account and Order information are managed in an OrderContext, a datastore that facilitates the exchange of information needed to coordinate order fulfillment activities between the bookstore's automated systems and human clerks.

Consistency and efficiency are important to the bookstore. Bookstore customers expect the same experience each time they order books and they expect this service to be performed in a timely fashion. The Coordinator-Worker-Context process pattern enables the bookstore to implement and execute its order fulfillment process in a consistent and efficient manner satisfying customer requirements and providing clear responsibilities for bookstore workers.

3. Applicability

Use the Coordinator-Worker-Context process pattern when:

- You are implementing a process that involves multiple workers of different types. Each type of worker is dedicated to specific activities in the process and is performed by specially trained human workers or special purpose automation.
- Human workers and automated systems must collaborate to execute the process. Human workers play important roles in the process and must interact with automated systems to accomplish the activities of the process.
- A high degree a consistency in process execution is required.
- The work being done by the process is complex and requires coordination between workers to ensure consistency and efficiency. Examples include processes that have detailed instructions and well defined service level agreements.
- The output of the process is produced incrementally and must be communicated or transferred between workers at each step of the process.
- Workers may be selected from a pool rather than being dedicated to a specific customer, process, or activity.

Do not use the Coordinator-Worker-Context process pattern when:

- The process is implemented entirely through automated systems.
- Workers have well defined input and output relationships between themselves that specify the transfer of process artifacts.
- The work being done by the process allows workers to work independently with little need to coordinate their activities.

4. Structure

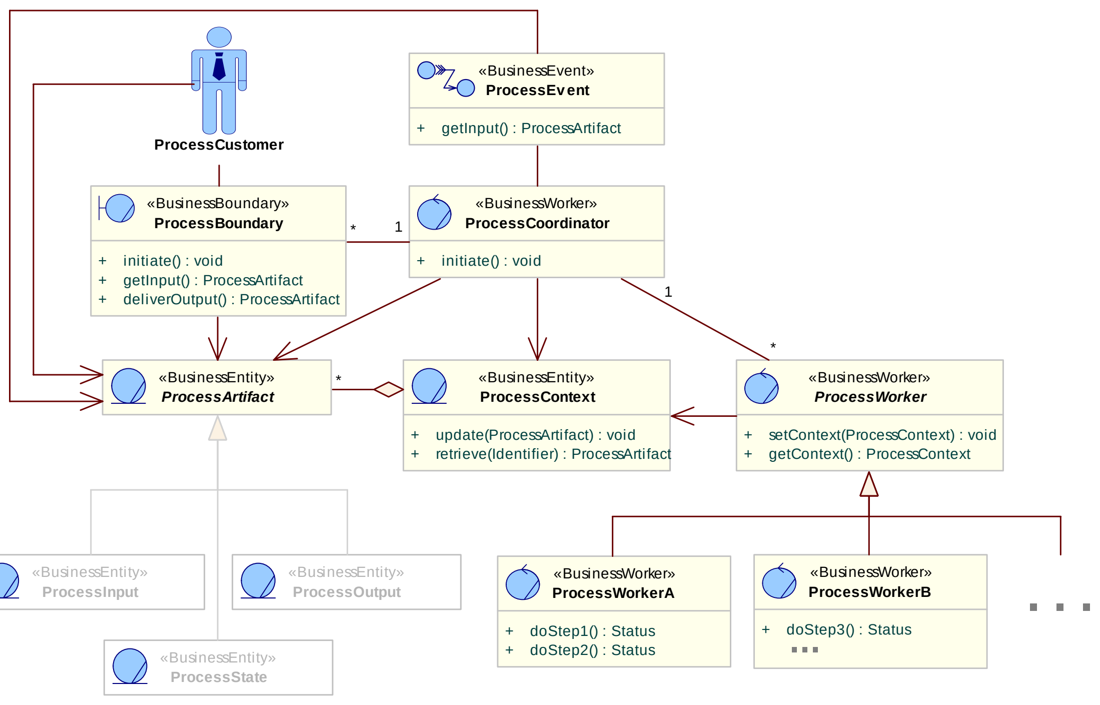


Figure 3

5. Participants

- PROCESSCUSTOMER (Bookstore Customer)
  - — Initiates a process by interacting with the PROCESSBOUNDARY
  - — Provides PROCESSARTIFACT(s) as input to PROCESSBOUNDARY at the beginning of process execution
  - — Obtains PROCESSARTIFACT(s) as output from PROCESSBOUNDARY at the end of process execution
- PROCESSEVENT
  - — Initiates a process execution by notifying the PROCESSCOORDINATOR
- PROCESSBOUNDARY (Purchasing, DeliveryService)
  - — Interacts with PROCESSCUSTOMER to accept PROCESSARTIFACTS as input and may return PROCESSARTIFACTS as output
  - — Initiates a process execution by notifying the PROCESSCOORDINATOR
- PROCESSCOORDINATOR (OrderManager)
  - — Interacts with PROCESSBOUNDARY to accept PROCESSARTIFACTS as input and may return PROCESSARTIFACTS as output
  - — Coordinates PROCESSWORKERS during process execution
- PROCESSWORKER (Billing, Inventory, Shipping)
  - — Performs the main processing steps of the process as directed by PROCESSCOORDINATOR
  - — Updates PROCESSCONTEXT with intermediate state during process execution
- PROCESSARTIFACT (OrderArtifact, Account, Order, Book, Package)
  - — Information or other resources that are inputs to the process
  - — State or other information produced and consumed during process execution
  - — Products, services, information, or other things that are resulting outputs from the process
- PROCESSCONTEXT (OrderContext)
  - — Organizes or manages the intermediate state of the process
  - — Provides a common point of access to PROCESSARTIFACTS

6. Collaborations

The following diagram (see figure 4) describes the overview of the Coordinator-Worker-Context collaborations. A process can be initiated by a customer or by an event. Once the process has been initiated, the main work of the process is performed. During this stage the output of the process is being produced. After the process's main work is complete, the resulting output is delivered to the customer.

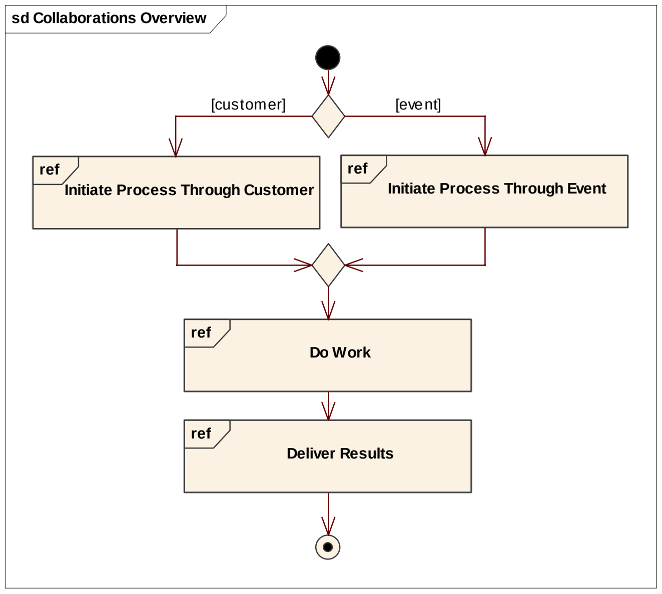

sd Collaborations Overview
[customer]
[event]
ref
Initiate Process Through Customer
ref
Initiate Process Through Event
ref
Do Work
ref
Deliver Results

Figure 4

A customer initiates a process by sending a request to the process's boundary. The boundary relays this request to the coordinator which starts the process by directing the boundary to obtain any necessary input from the customer. This input is stored in the context for use in subsequent work steps. (see figure 5)

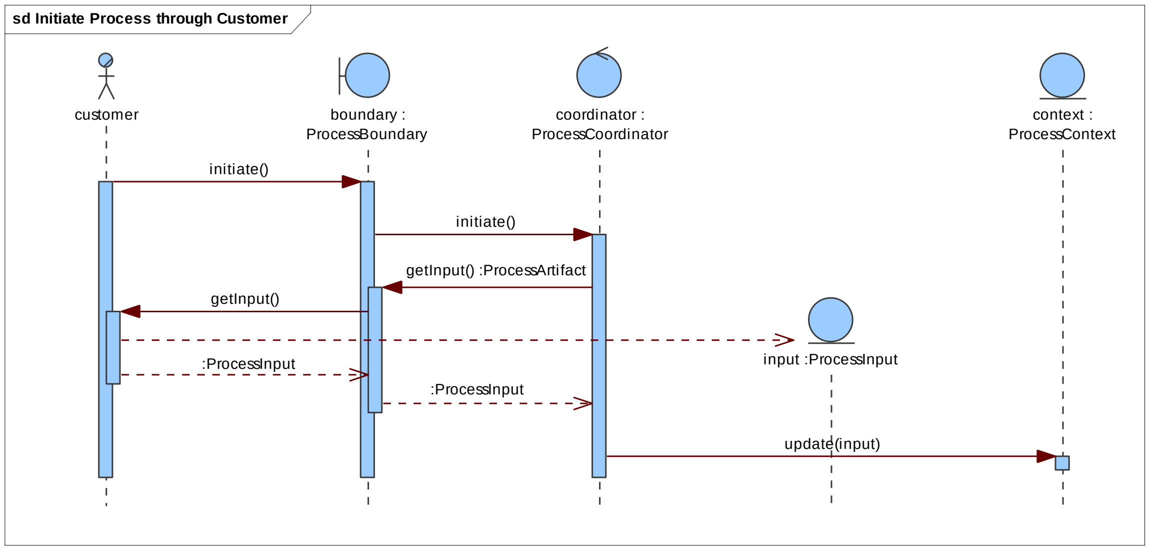

sd Initiate Process through Customer
customer
boundary : ProcessBoundary
coordinator : ProcessCoordinator
context : ProcessContext
initiate()
initiate()
getInput() :ProcessArtifact
getInput()
input :ProcessInput
:ProcessInput
:ProcessInput
update(input)

Figure 5

An event initiates a process by signaling to the coordinator to begin the process. The coordinator obtains any input information from the event and stores this input in the context for use in subsequent work steps. (see figure 6)

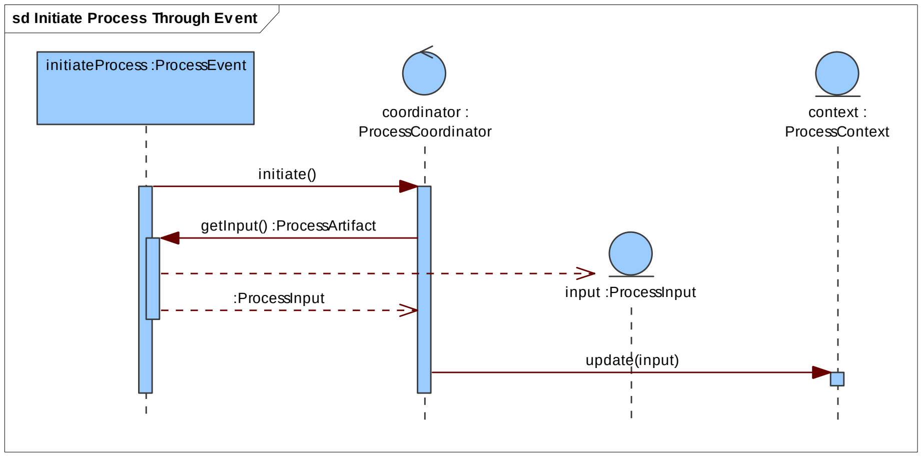

sd Initiate Process Through Event
initiateProcess :ProcessEvent
coordinator : ProcessCoordinator
context : ProcessContext
initiate()
getInput() :ProcessArtifact
input :ProcessInput
:ProcessInput
update(input)

Figure 6

The main work of the process is coordinated by the coordinator and executed by the workers. The coordinator directs the workers to begin working on their tasks in the order specified by the process. Work may proceed sequentially with each worker notifying the coordinator that their task is complete or tasks can be executed in parallel depending on the nature of the work. (see figure 7)

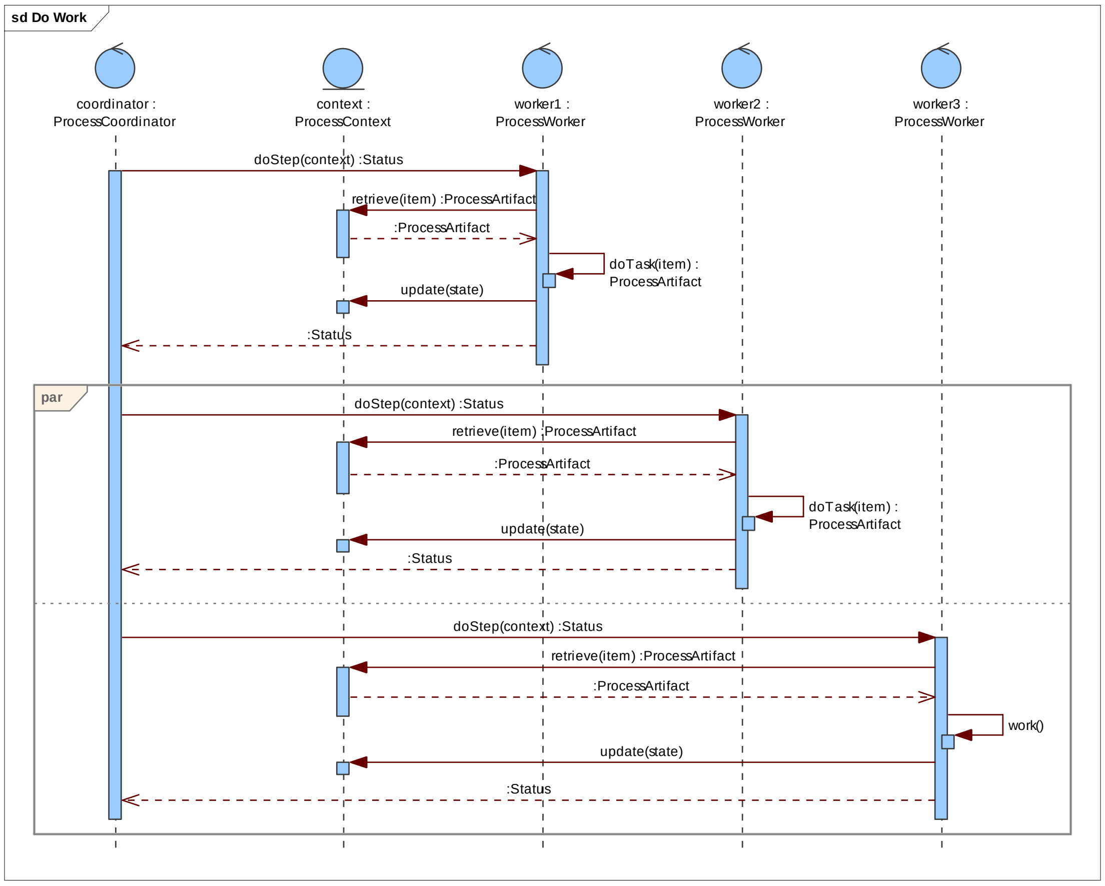


Figure 7

The coordinator—or a worker designated for the task—produces or packages the process output from data or materials in the process context. The coordinator directs the boundary to deliver the output to the customer and the process is complete. (see figure 8)

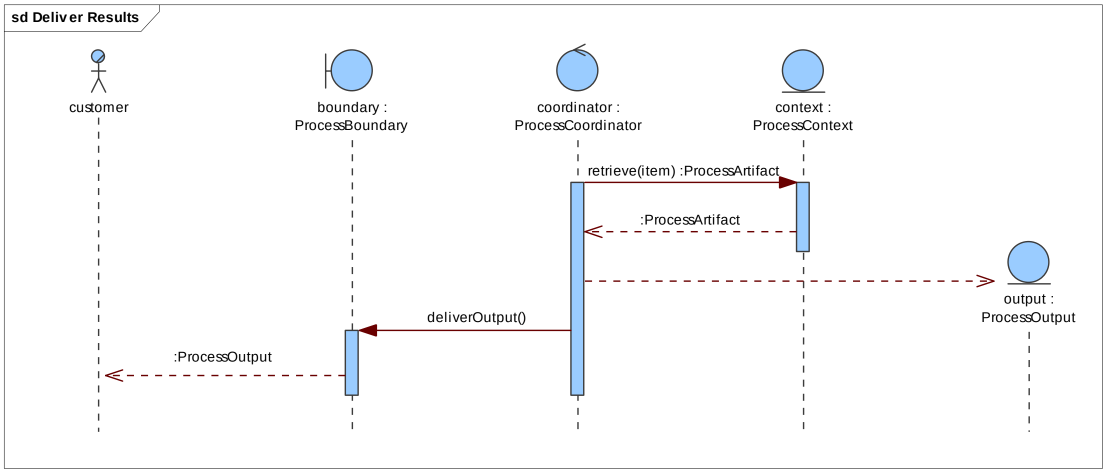


Figure 8

7. Consequences

The Coordinator-Worker-Context process pattern has the following benefits:

- Enables clear separation of responsibilities. The Coordinator-Worker-Context pattern separates process participants according to their responsibilities. For example, the PROCESSCOORDINATOR role is responsible for ensuring that the process executes correctly by coordinating the work of PROCESSWORKERS and monitoring the progress of work being done in the process. PROCESSWORKERS are responsible for specific activities that must be completed during process execution. This separation of responsibilities enable each participant to concentrate on their activities which has the effect of making the process easier understand form a participant's point of view and therefore easier to execute.

- Ensures that work is efficiently distributed to workers. The PROCESSCOORDINATOR assigns work to specific PROCESSWORKERS and tracks their work load. If a PROCESSWORKER is working at maximum load the PROCESSCOORDINATOR selects another PROCESSWORKER to

- Provides common access point for information and material needed to complete work. A PROCESSCONTEXT that is equally accessible to the PROCESSCOORDINATOR and all PROCESSWORKERS improves communication between participants and facilitates the transfer of information and materials needed for efficient process execution. The PROCESSCONTEXT may also provide mechanisms that assist in synchronizing parallel work.

Coordinator-Worker-Context has the following liability:

- PROCESSCOORDINATOR could become a bottleneck if not properly implemented. The PROCESSCOORDINATOR is the primary decision maker during process execution and controls the progress from task to task until the process has completed executing. This responsibility can become a bottleneck if the PROCESSCOORDINATOR becomes overwhelmed by too many items that need monitoring including: process executions occurring at the same time, concurrent tasks within a process, and complex decisions. Some of these issues are addressed in the Implementation section.

8. Implementation

The following implementation issues should be considered when implementing processes with the Coordinator-Worker-Context process pattern:

- Process automation. In many cases it is possible to improve the efficiency and consistency of a process by automating some of its elements. The PROCESSCOORDINATOR can be automated when the plan defining the process steps is relatively simple and the decision criteria for advancing from step to step can be represented in an automated system. Business Rule Management Systems [Halle 2002][Morgan 2002] and Business Process Execution Language engines [OASIS 2007] are frequently used technologies for implementing automated PROCESSCOORDINATORS. PROCESSWORKERS can also be automated if the tasks they perform are conducive to automation in either mechanical or software-intensive systems. Some tasks require a combination of both human and automated workers. These types of tasks can be implemented with a business ensemble that combines human worker(s) and automation into a cohesive unit (see figure 9).

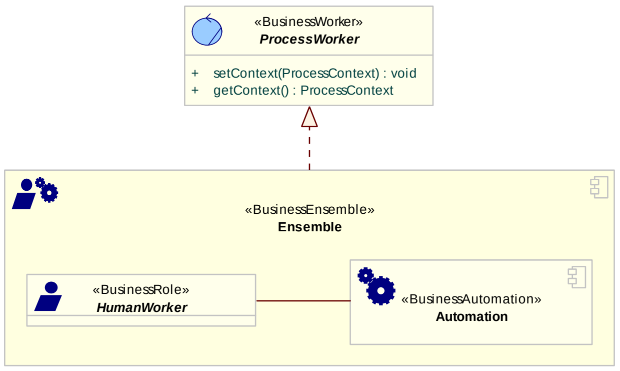


Figure 9

- Designing the ProcessContext. The PROCESSCONTEXT holds the information and materials needed by all PROCESSWORKERS to complete the process. The following are important design issues that must be addressed when implementing the PROCESSCONTEXT:
  - Integrating physical and digital elements into a single context. It is common to manipulate physical as well as digital elements in a process. For example, the book order process described in the Motivation section manipulates the state of the book order which is a digital element and the physical books that make up the order. One way of managing physical elements that interact with digital elements is to create a digital proxy [Gamma+ 1995] for each type of physical element. The digital proxy is stored in the PROCESSCONTEXT and provides the needed interaction with the digital elements. Using our example, a physical book can have a book record that describes the book, the number of copies in inventory, and other data. Book records represent the physical books.
  - Notifying process participants of changes to the PROCESSCONTEXT. The state of the PROCESSCONTEXT changes during process execution. Often it is important for process participants to be notified of these changes so they can continue the next step or monitor progress. If the PROCESSCONTEXT is completely implemented in software then mechanisms can be designed into the PROCESSCONTEXT to trigger notifications on changes. This design conceptually follows the Observer design pattern [Gamma+ 1995] where the PROCESSCONTEXT acts as the Subject and the process participants act as the Observers. The notification medium depends on the process and the systems that implement it. Example notification mechanisms include email, pop-up windows, and automated phone messages.
  - Separating the PROCESSCONTEXT from PROCESSWORKER sandboxes. The PROCESSCONTEXT contains the shared state of a process execution but there may be tasks that use state private to a specific PROCESSWORKER. This state can be stored in a sandbox that is dedicated to a specific PROCESSWORKER rather than mixing this state into the PROCESSCONTEXT. In the bookstore example the Order Management System may provide functionality for an Inventory Clerk to assume control of a Book Order and assemble the Book Order Package without updating the shared state of the Book Order until the clerk's work is completed. This insulates other PROCESSWORKERS from being exposed to incremental or incomplete work that could be confusing or difficult to follow.
- Handling process parallelism. Many processes will have activities that can be (or need to be) executed in parallel. Parallel execution of activities requires synchronized access and update of the PROCESSCONTEXT and the PROCESSCONTEXT or the PROCESSCOORDINATOR can be responsible for synchronizing the activities of multiple workers either through synchronization state in the PROCESSCONTEXT or explicit signaling from the PROCESSCOORDINATOR. For example the PROCESSCONTEXT or PROCESSCOORDINATOR may provide functionality for reserving or checking out information or materials contained in the PROCESSCONTEXT preventing other PROCESSWORKERS from accessing these item simultaneously.
- Process Initiation. A process can be initiated by PROCESSCUSTOMER making a request and providing a PROCESSARTIFACT as input to the PROCESSCOORDINATOR through the PROCESSBOUNDARY. A process can also be initiated by the PROCESSCOORDINATOR receiving a PROCESSBOUNDARY. If your process can be initiated by both a customer request and an event, the PROCESSCOORDINATOR must be designed to accept both modes of initiation.
- Multiple PROCESSBOUNDARIES. Processes may receive input in the form of ProcessArtifacts from multiple types of PROCESSBOUNDARIES and send PROCESSARTIFACTS as output to multiple PROCESSBOUNDARIES. For example, in an order fulfillment process a customer could make the same kinds of orders over a web site, a phone, or face-to-face with a sales representative. Delivery of the order could be through a parcel delivery company, purely electronic (depending on the product), or in a brick and mortar store. If these PROCESSBOUNDARIES have common behavior they can be grouped into a hierarchy (see figure 10).

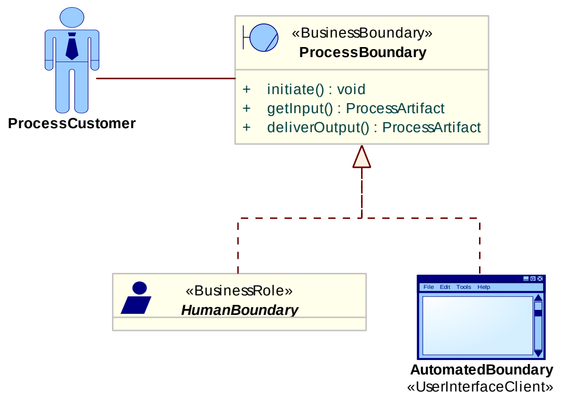


Figure 10

- Hierarchical PROCESSCOORDINATORS. Large or complex process can be decomposed into a hierarchy of sub-processes, each with their own PROCESSCOORDINATOR. This can be accomplished by making PROCESSCOORDINATOR a specialization of PROCESSWORKER (see figure 11) The resulting design follows the Composite design pattern [Gamma+ 1995].

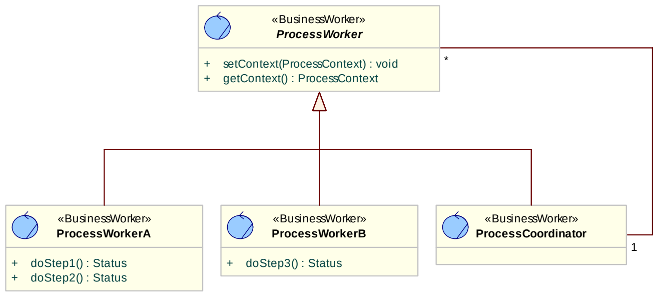


Figure 11

PROCESSCOORDINATOR instances form a tree with the main coordinator as the root (see figure 12).

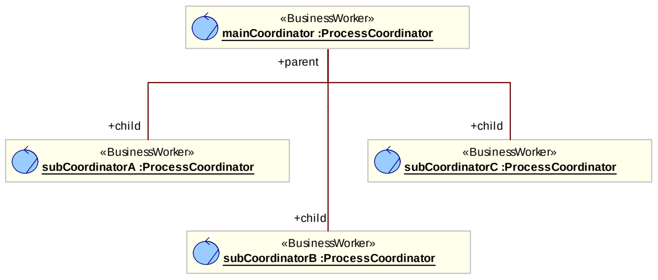

«BusinessWorker»
mainCoordinator :ProcessCoordinator
+parent
+child
«BusinessWorker»
subCoordinatorA :ProcessCoordinator
+child
«BusinessWorker»
subCoordinatorC :ProcessCoordinator
+child
«BusinessWorker»
subCoordinatorB :ProcessCoordinator

Figure 12

- Multiple instances of PROCESSCOORDINATOR or a single instance of PROCESSCOORDINATOR. The capabilities of the organization and systems implementing the process will determine if the process has multiple PROCESSCOORDINATORS responsible for groups of process executions or a single PROCESSCOORDINATOR responsible for all process executions. Using the order fulfillment process as an example, a small organization may only have the resources to process few concurrent orders and would have a single instance of PROCESSCOORDINATOR for all order fulfillment executions. A larger organization would have more resources available to process many concurrent orders and may choose to have multiple instances of PROCESSCOORDINATOR to manage a greater number of order fulfillment executions.

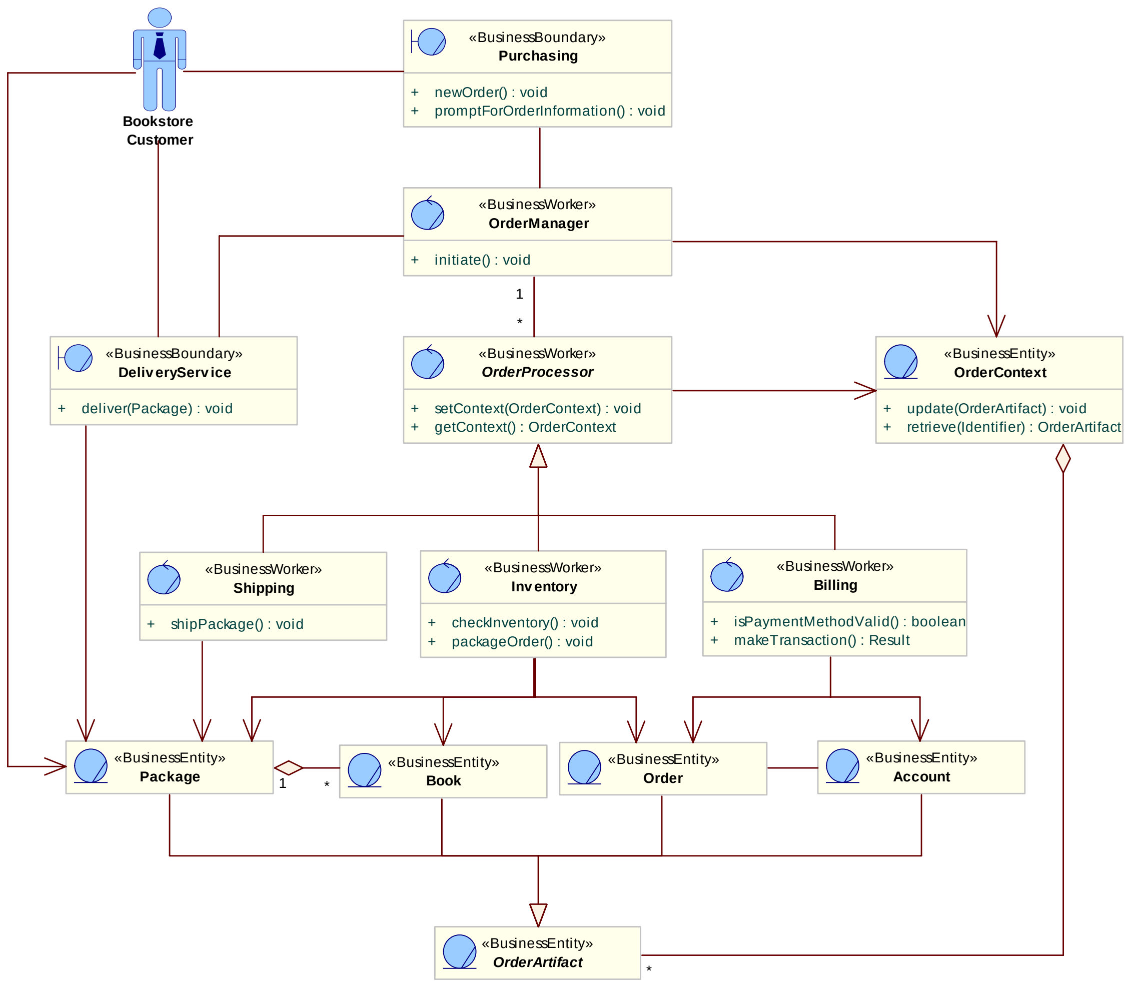
- Reporting status to Customer. PROCESSCUSTOMERS of long running process executions may require status during process execution. This can be accomplished by a mechanism in the PROCESSBOUNDARY for the PROCESSCUSTOMER to request status and a mechanism in the PROCESSCOORDINATOR for obtaining execution status from the PROCESSCONTEXT. The PROCESSCOORDINATOR transforms the execution status into a format consumable by the PROCESSCUSTOMER and instructs the PROCESSBOUNDARY to deliver this status to the PROCESSCUSTOMER.

- Process Cancellation. PROCESSCUSTOMERS may require the ability to cancel a process execution. This functionality requires a mechanism in the PROCESSBOUNDARY for the PROCESSCUSTOMER to send a cancel request to the PROCESSCOORDINATOR The PROCESSCOORDINATOR cancels the ongoing work of PROCESSWORKERS and clears the PROCESSCONTEXT. The PROCESSCUSTOMER instructs the PROCESSBOUNDARY to notify the PROCESSCUSTOMER that the process execution has been cancelled. Some types of processes may require a transaction mechanism to ensure the consistency of important information (such as account balances) after process cancellation.

9. Example

In the Motivation section we presented a high level view of an order fulfillment process for an online book store. We developed business class diagrams (see figures 1 and 2) that described the major business boundaries, business workers, and business entities of the bookstore's order fulfillment process. In this section we further illustrate the Coordinator-Worker-Context process pattern by providing a more detailed view of the bookstores components and their collaborations.

The business classes shown in the following diagram (see figure 13) represent the logical components of the bookstore's order fulfillment process. These logical components are realized by human workers performing well defined business roles or automated systems designed to provide the necessary functionality that executes the process.



Bookstore Customer
«BusinessBoundary» Purchasing
+ newOrder() : void
+ promptForOrderInformation() : void
«BusinessWorker» OrderManager
+ initiate() : void
1
*
«BusinessBoundary» DeliveryService
+ deliver(Package) : void
«BusinessWorker» OrderProcessor
+ setContext(OrderContext) : void
+ getContext() : OrderContext
«BusinessEntity» OrderContext
+ update(OrderArtifact) : void
+ retrieve(Identifier) : OrderArtifact
«BusinessWorker» Shipping
+ shipPackage() : void
«BusinessWorker» Inventory
+ checkInventory() : void
+ packageOrder() : void
«BusinessWorker» Billing
+ isPaymentMethodValid() : boolean
+ makeTransaction() : Result
«BusinessEntity» Package
1
*
«BusinessEntity» Book
«BusinessEntity» Order
«BusinessEntity» Account
«BusinessEntity» OrderArtifact
*

Figure 13

Figure 14 provides an example of mapping business roles and automation to the logical components of the business process. The Shipping business worker is realized by a ShippingClerk business role. ShippingClerk specifies the necessary skill required to perform the role and the responsibilities and tasks assigned to the role. The Billing business worker is realized by a BillingSystem software application that provides all of the necessary functionality to implement the Billing component of the process. The Inventory business worker is realized by an InventoryEnsemble made up of an InventoryClerk business role interacting with an InventorySystem software application. This is an example of a business ensemble that combines business roles and automation into a unified component that realizes a business process element.

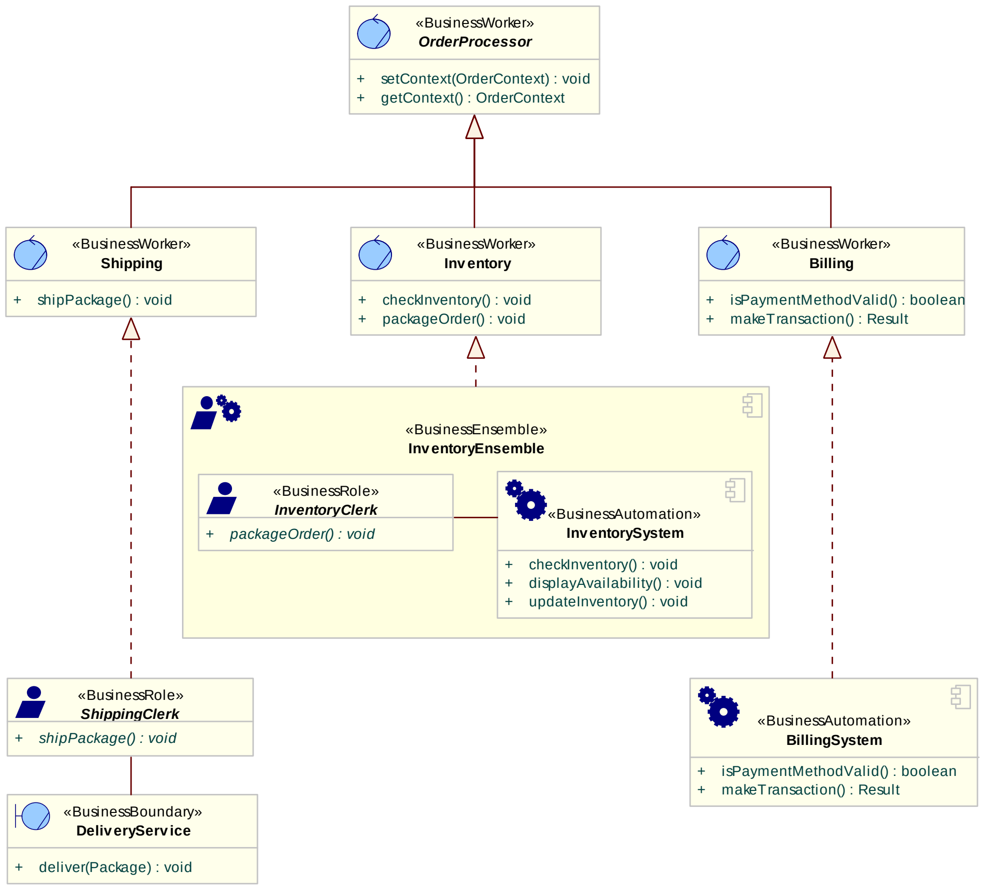


Figure 14

The remainder of this section focuses on describing the collaborations that implement the order fulfillment process. The following diagram (see figure 15) describes the high level interactions of the online bookstore's order process. This diagram is the framework that organizes the process's interactions. Subsequent diagrams drill down into each step.

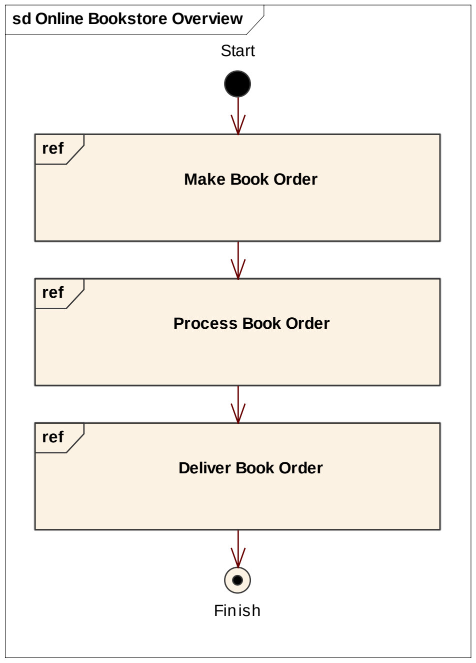


Figure 15

The following sequence diagram (see figure 16) contains the interactions for making a book order. The customer initiates the process by starting a new book order. The customer selects the books to be purchased and submits the order to the bookstore's order management system. The order management system saves the order and initializes it for processing.

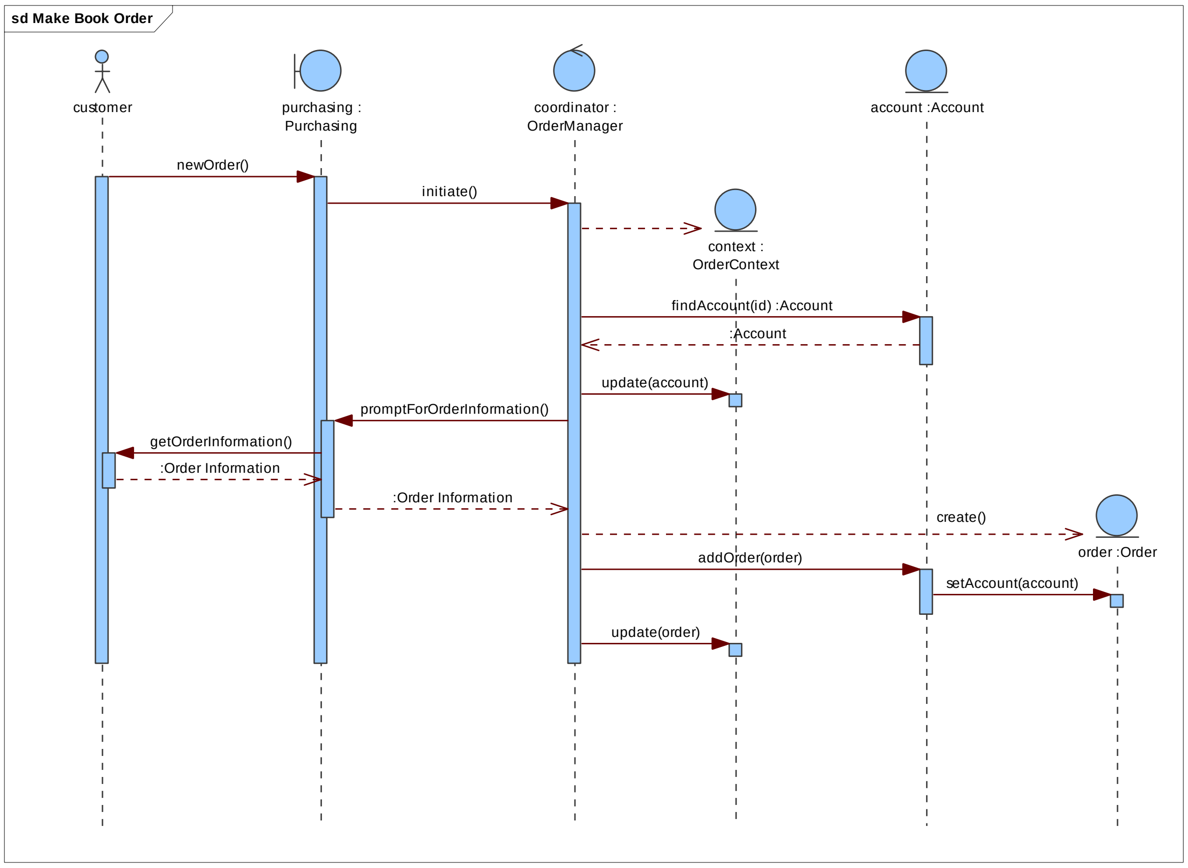


Figure 16

The next diagram (see figure 17) contains the interactions for processing the book order. The order management system retrieves the billing information from the customer's account and sends that information along with the order amount to the billing system.

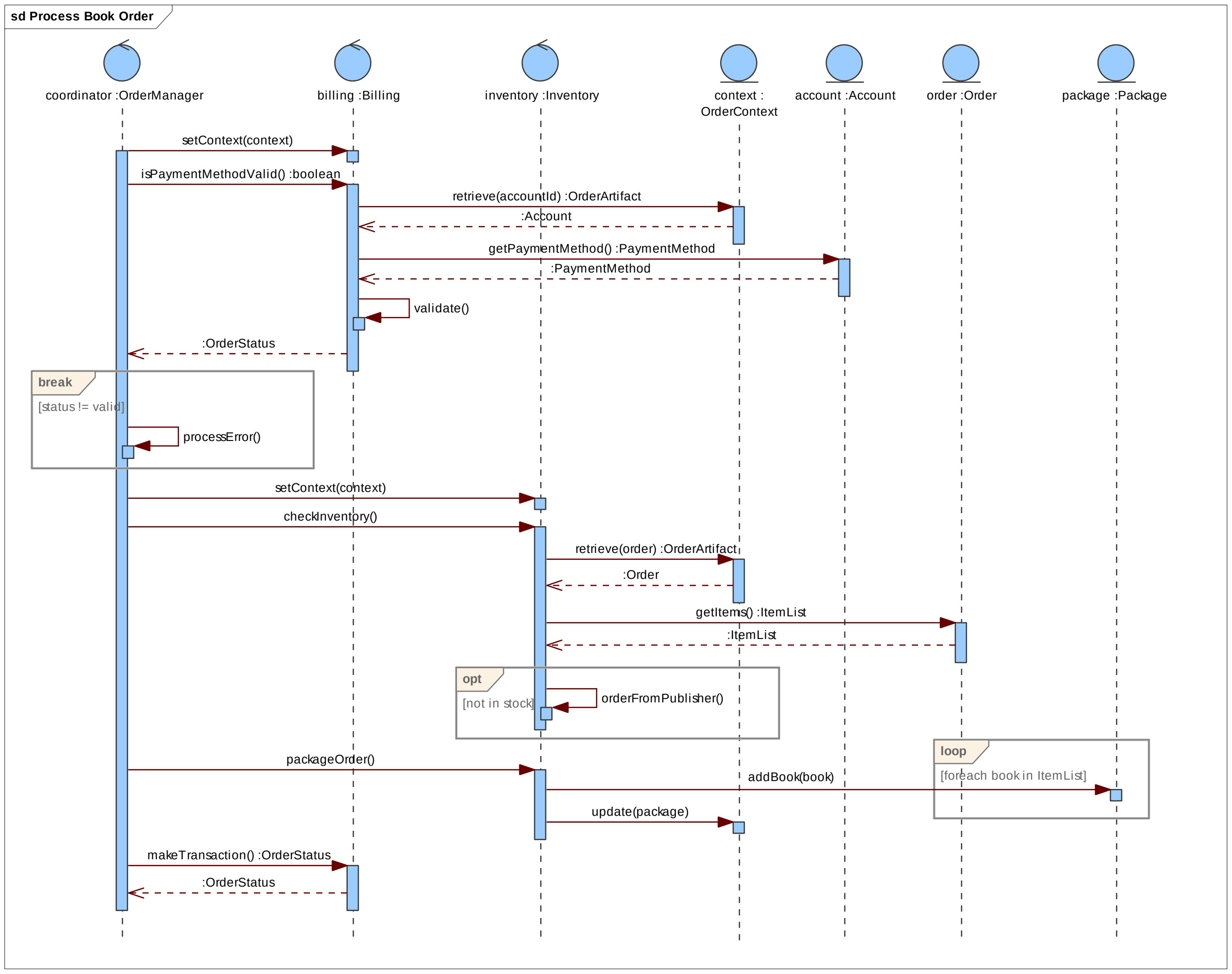


Figure 17

The final diagram (see figure 18) shows the delivery of the package containing the books that were ordered and the completion of the book order. The order management system directs the shipping clerk to send the package to the shipping company that partners with the bookstore. Once the package has been sent, the clerk notifies the order management system which completes the book order. The process concludes when the shipping company delivers the package to the customer.

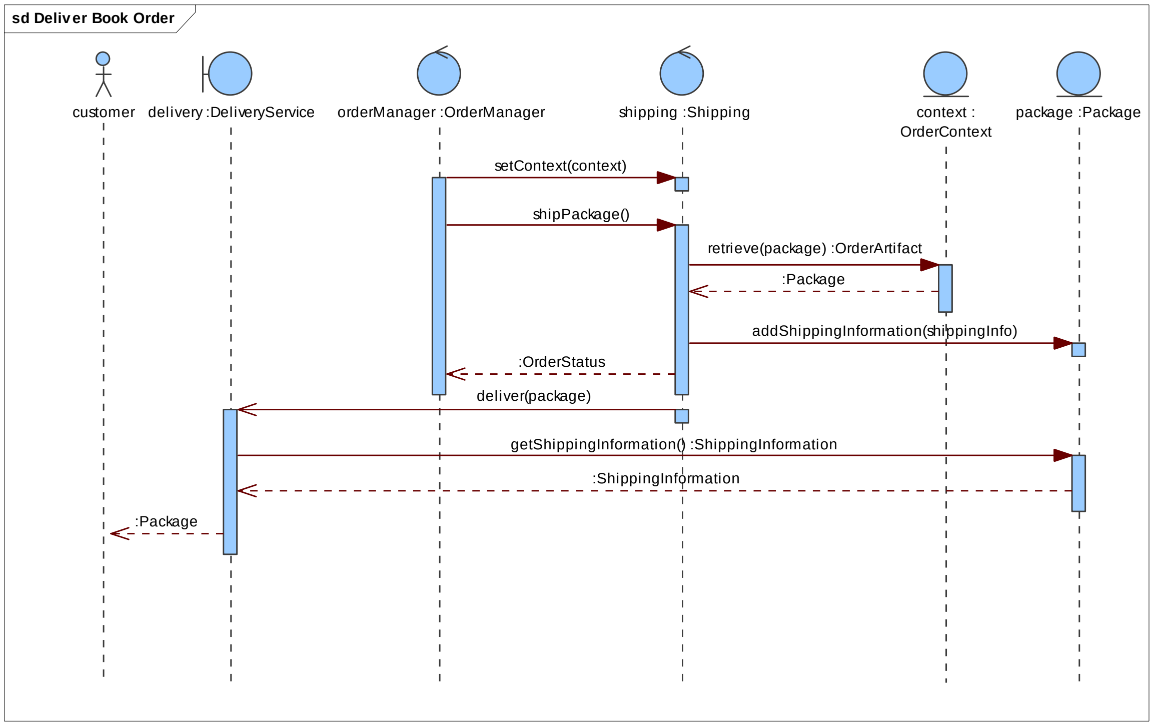


Figure 18

10. Known Uses

There are numerous instances of Coordinator-Worker-Context in a variety of businesses and organizations. Many implementations of the Order Fulfillment Process like the ones described in the Motivation and Example sections conform to Coordinator-Worker-Context.

Software Development Processes typically use Coordinator-Worker-Context to organize their work. Stakeholders provide input to the process in the form of Requirements. Project Managers act as PROCESSCOORDINATORS to ensure that work is being performed properly and in the correct order. Analysts, Architects, Developers, and Testers are all process workers contributing to the development of the end product. The source code repository plays the part of the process context that holds the state of the development effort. Another example from software development is Continuous Integration Servers such as CruiseControl or IBM Rational's BuildForge. A Continuous Integration Server acts as an automated PROCESSCOORDINATOR that triggers builds and tests when a developer (PROCESSWORKER) delivers a new version of source code (PROCESSARTIFACT) to the source code repository (PROCESSCONTEXT). If the build and tests are successful the Continuous Integration Server notifies the developers on the development team that a new source code baseline is available to use in their next set of development tasks.

The Object Advantage [Jacobson+ 1995] presents a vision of process-centric business organizations that includes Process Leader and Process Operator roles. These roles are similar to PROCESSCOORDINATOR and PROCESSWORKER respectively but in Jacobson's description they appear to describe human workers only.

11. Related Patterns

Coordinator-Worker-Context can optionally use the following patterns in its application (see figure 19):

- The Proxy design pattern [Gamma+ 1995] can be used in a conceptual manner to provide a digital proxy for a real world entity combining both digital and physical entities into a common PROCESSCONTEXT.
- The Observer design pattern [Gamma+ 1995] can be used in a conceptual manner to notify PROCESSWORKERS of changes to the PROCESSCONTEXT.
- The Composite design pattern [Gamma+1995] can be used in a conceptual manner to create a hierarchy of PROCESSCOORDINATORS that help to decompose a highly complex process into more manageable sub-processes.

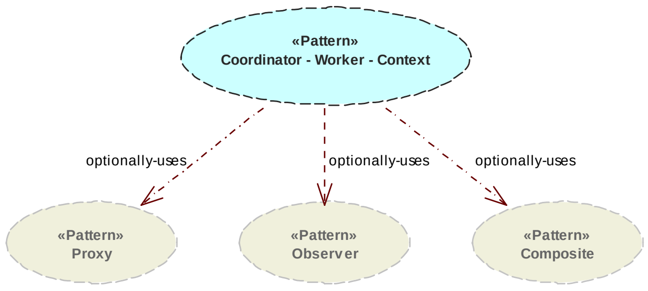
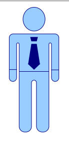
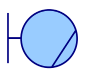
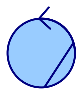
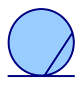
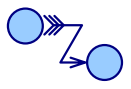
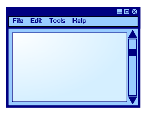
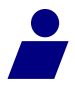
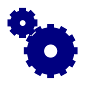
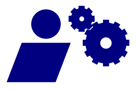


Figure 19

12. Acknowledgements
Many thanks to Hironori Washizaki for being my PLoP ’08 shepherd and providing many useful suggestions for improving this paper. Thanks and gratitude to the PLoP ’08 Process & Services workshop (Bobby Woolf, Lisa Hvatum, Paul Austrem, Philipp Bachman, Rob Daigneau, Geert Monsieur, Mark Mahoney, and Miyuko Naruse) for excellent feedback and discussions.

13. Notation Guide

«Stereotype»	Description	Icon
BusinessActor	A person or system outside of a business that interacts with one or more business processes.	
BusinessBoundary	An interface between a BusinessActor and a business.	
BusinessWorker	A component that embodies a set of behaviors, skills, and responsibilities that are applied to executing the activities of a business process.	
BusinessEntity	Informational or physical item that is input to or output from a business process, or is used in the internal activities of a business process.	
BusinessEvent	An event that has significance to a business and may trigger one or more business process activities.	
UserInterfaceClient	A kind of Business Boundary that is a software system that serves as the interface between a Business Actor and a business.	
BusinessRole	A role defined in the business and performed by a person or group that executes the activities of a business process.	
BusinessAutomation	A realization of a Business Worker that is a software system or mechanical device that executes the activities of a business process.	
BusinessEnsemble	A realization of a Business Worker that is a combination of a human worker performing a Business Role in conjunction with some kind of Business Automation such as a software application or mechanical device.	

14. References
[Gamma+1995] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. Design Patterns-Elements of Reusable Object-Oriented Software. Reading, MA: Addison-Wesley, 1995.

[Halle 2002] Barbara von Halle. Business Rules Applied. John Wiley and Sons, 2002.

[Jacobson+ 1995] I. Jacobson, M. Ericsson, A. Jacobson. The Object Advantage: Business Process Reengineering with Object Technology . ACM Press, 1995.

[Morgan 2002] Tony Morgan. Business Rules and Information Systems. Addison-Wesley, 2002.

[OASIS 2007] OASIS WS-BPEL Technical Committee. Web Services Business Process Execution Language Version 2.0. OASIS, 2007. Link:

http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf