System Analysis and Design

Eighth Edition

Alan Dennis, Barbara Wixom, Roberta M. Roth

Chapter 4

Understanding Processes with Use Cases and Process Models

Objectives

- Explain the purpose of use cases in the analysis phase of the SDLC.

- Describe the various parts of a use case and the purpose of each part.

- Describe how use cases contribute to the functional requirements.

- Describe how use cases inform the development of test plans.

- Explain the process used to create a use case.

- Explain the rules and style guidelines for data flow diagrams.

- Describe the process used to create data fl ow diagrams.

- Create data flow diagrams.

Introduction

- A key aspect of determining the requirements for the new system is understanding the *user requirements*
  - The things the users need to accomplish with the new system
- Use cases help us understand and clarify the users' required interactions with the system and can help us more fully understand the functional requirements of the new system
- A use case represents how a system interacts with its environment
- Use cases are especially valuable for business system applications and websites
- Process models have been a part of structured systems analysis and design techniques for many years

What Is a Use Case?

- A use case depicts a set of activities performed to produce some output result

- Each use case describes how an *event triggers* actions performed by the system and the user
  - Everything in the system can be thought of as a response to some trigger event

Basic Information

- Each use case has a *name* and *number*, and brief *description*.
- The *priority* may be assigned to indicate the relative significance.
- The *actor* refers to a person, another system, or a hardware device that interacts with the system to achieve a useful goal.
- The *trigger* for the use case – the event that causes the use case to begin.
- Events triggers can be *external* or *temporal*

Create Preliminary Custom Drone Order Use Case—Casual Format

Use Case Name: Create preliminary custom drone order	ID: UC-6	Priority: High
Actor: Customer		
Description: The customer selects and customizes a commercial drone to purchase		
Trigger: Customer wants to purchase a commercial drone		
Type: ☑ External ☐ Temporal		

Preconditions

- It is important to define clearly what needs to be accomplished before each use case begins

- Preconditions define the state the system must be in before the use case commences

> **Preconditions:**
> 1. The customer is authenticated by logging in to his account
> 2. The Sales System Order Processing application is online

Normal Courses

- The normal course lists the steps that are performed when everything flows smoothly in the system

- Sometimes called the ***happy path***

Normal Course:
1.0 Order a customized drone
   1. The customer selects a base model drone from a list of models
   2. The system provides availability status for that model (in stock, out of stock)
   3. For out of stock status, system displays expected date available
      a. Customer accepts future availability date; proceed to step 4
      b. Customer rejects future availability date; return to step 1
   4. The system displays a list of options and upgrades for the selected model
   5. The customer selects desired model options and upgrades
   6. Preliminary order with cost estimate is created and displayed
   7. Customer may return to step 4, confirm order, save for future consideration, or exit without saving
   8. Unconfirmed orders are stored in Unconfirmed Custom Order datastore
   9. Confirmed orders are saved in Confirmed Custom Order datastore
10. Shop manager is notified of Confirmed Order requiring approval

Postconditions

- In this section of the use case, we define the final products of this use case

- These postconditions also serve to define the preconditions for the next use case in the series
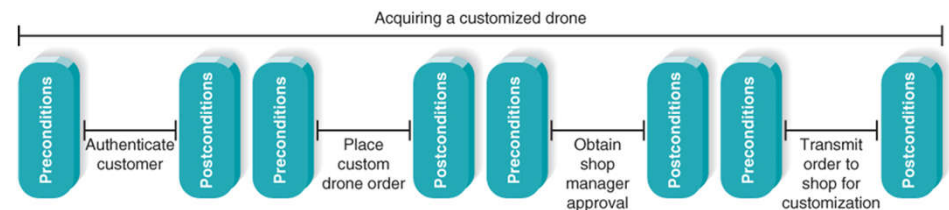
Postconditions:
1. Unconfirmed order is stored in Unconfirmed Custom Order datastore
2. Confirmed order is stored in Confirmed Custom Order datastore
3. Shop manager sent notice of Confirmed Order requiring approval

Exceptions

- To be complete, a use case should describe any error conditions or exceptions that may occur as the use case steps are performed

- These are not normal branches in decision logic but are unusual occurrences or errors that could potentially be encountered and will lead to an unsuccessful result

- We want to be sure that the system does not fail while in use because of an error

Use Cases in Sequence

- Uses cases often performed in sequence

- No single use case should be too large

- Important to define initial and ending states

Acquiring a customized drone

Preconditions | Authenticate customer | Postconditions | Preconditions | Place custom drone order | Postconditions | Preconditions | Obtain shop manager approval | Postconditions | Preconditions | Transmit order to shop for customization | Postconditions

Additional Use Case Issues

- Alternative paths
- Summary of inputs and outputs
- Frequency of use
- Business rules
- Special requirements
- Assumptions
- Notes and issues

More Elaborate Use Cases are Especially Valuable When…

- User representatives are not actively engaged with the development team throughout the project

- The application is complex and has a high risk associated with system failures

- Comprehensive test cases will be based on the user requirements

- Collaborating remote teams need a detailed, shared understanding of the user requirements

Use Case Practical Tips

- Use gradual refinement

- Concentrate on describing the user's objectives with the system completely and accurately

- Keep both audiences in mind – users and developers

- Create use cases only when needed to clarify what the system must do from the user's perspective
  - Not needed for simple events

Use Cases and the Functional Requirements

- Use cases are useful tools to clarify user requirements

- Use cases convey only the user's point of view

- Transforming the user's view into the developer's view through functional requirements is one of the system analyst's key contributions

- The derived functional requirements tell the developers more about what the system must do

Steps for Writing for Use Cases

1. Identify the use cases
2. Identify the major steps within each use case
3. Identify elements within steps
4. Confirm the use case

Creating Use Cases

- Identify events the system must respond to – develop event-response list

- Create use case form for the complex events

- For each use case:
  - Identify the major steps
  - Identify elements with each major step (inputs and outputs)
  - Confirm use case with users through role-playing

- Revise functional requirements as needed

Identify the Major Steps for Each Use Case

- The analyst should ask the users what tasks need to be completed before the use case steps can begin

- Next, the user–system interactions should be outlined as a series of steps in the Normal Course section of the form

- Each step should be about the same size as the others

- Occasionally, a use case is so simple that further refinement is not needed

- Once the steps have been outlined at the proper level of detail, the postconditions can be completed

Identify Elements within Steps

- The last column ("Information for Steps") must be completed, and arrows may be drawn to describe inputs and outputs from the steps
- The goal at this point is to identify the major inputs and outputs for each step
- The users and analysts then return to the steps in the use case and begin tracing the flow of the steps
- It is not unusual at this point for users to discover that they forgot to list the entire steps
- The Summary area for inputs and outputs found at the end of the use case form is completed once the team is satisfied with the steps, inflows, and outflows

Confirm the Use Case

- The final step is for the users to confirm that the use case is correct as written

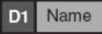- Review the use case with the users to make sure that each step and each input and output are correct
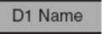
Data Flow Diagrams

- Data flow diagramming is a technique that diagrams the business processes and the data that pass among them

- We use data flow diagrams (DFDs) to describe the to-be system's interactions with its environment, processes, flows of data, and data stores

- Logical process models are models that describe processes, without suggesting how they are conducted

- Physical models provide information that is needed to ultimately build the system
  - Covered in a later chapter

Reading Data Flow Diagrams
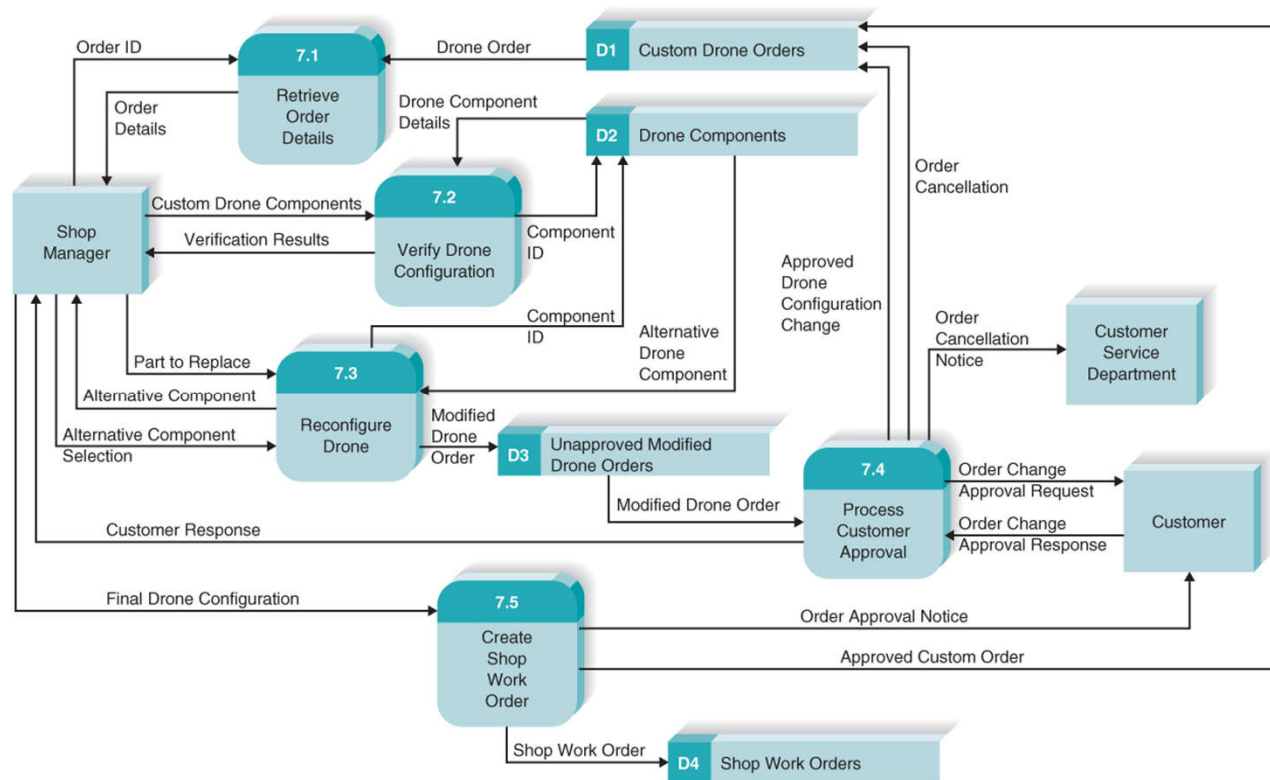
Data Flow Diagram Elements

Data Flow Diagram Element	Typical Computer-Aided Software Engineering Fields	Gane and Sarson Symbol	DeMarco and Yourdon Symbol

Every *process* has
  a number
  a name (verb phase)
  a description
  at least one output
    data flow
  at least one input
    data flow
	Label (name)
	Type (process)
	Description
	(what is it)
	Process number
	Process description
	(structured English)
	Notes
		1
		Name
			Name

Every *data flow* has
  a name (a noun)
  a description
  one or more
    connections to a
    process
	Label (name)
	Type (flow)
	Description
	Alias (another name)
	Composition
	(description of data
	elements)
	Notes
		Name →	Name →

Every *data store* has
  a number
  a name (a noun)
  a description
  one or more input
    data flows
  one or more output
    data flows
	Label (name)
	Type (store)
	Description
	Alias (another name)
	Composition
	(description of data
	elements)
	Notes
		D1 Name	D1 Name

Every *external entity* has
  a name (a noun)
  a description
	Label (name)
	Type (entity)
	Description
	Alias (another name)
	Entity description
	Notes
		Name	Name

Elements of Data Flow Diagrams (Process)

- An activity or function performed for a specific business reason

- Can be manual or computerized

- Includes the following:
  - A number
  - A name (verb phrase)
  - A description
  - At least one output data flow
  - At least one input data flow

Elements of Data Flow Diagrams (Logical Process)

- Logical process models omit any processes that simply move or route data and leave the data unchanged.

- You do include logical processes that:
  - Perform computations (e.g., calculate grade point average)
  - Make decisions (e.g., determine availability of ordered products)
  - Sort, filter or otherwise summarize data (e.g., identify overdue invoices)
  - Organize data into useful information (e.g., generate a report or answer a question)
  - Trigger other processes (e.g., turn on the furnace or instruct a robot)
  - Use stored data (create, read, update or delete a record)

Elements of Data Flow Diagrams (Data Flow)

- A single piece of data or a logical collection of data
- Data Flow names describe the content of the data flow but not how it is implemented
- Always starts or ends at a process
- Includes the following:
  - A name (noun)
  - Description
  - One or more connections to a process

Elements of Data Flow Diagrams (Data in Motion)

- An input of data to a process, or the output of data (or information) from a process

- The creation, deletion, or update of data in a file or database (called a data store on the DFD)

- A data flow is depicted as a solid-line with arrow

- Control flows (non-data flows) trigger processes, such as 'time to run the weekly payroll'

- The control flow is depicted as a dashed-line with arrow

Elements of Data Flow Diagrams (Data Store)

- Most information systems capture data for later use.
- A data store is a collection of data that is stored in some way
- Include the following:
  - A number
  - A name (noun)
  - Description
  - One or more input data flows (somewhere in process model)
  - One or more output data flows (somewhere in process model)

Elements of Data Flow Diagrams (Data Store) Continued

- If data flows are data in motion, think of data stores as data at rest

- Data stores should describe "things" about which the business wants to store data

- Data flows leaving the data store are data retrievals

- Data flows entering the data store are updates or new data added

Elements of Data Flow Diagrams (External Entity)

- A person, organization, or system that is external to the system

- Has interactions with the system (adds data to system or receives data from system)

- Include the following:
  - A name (noun)
  - Description

Using Data Flow Diagrams to Define Business Processes

- Business processes are too complex to be shown on a single D F D

- A deliberate hierarchy is created with multiple "levels" of D F Ds

- To build the hierarchy, use **decomposition**
 - Child diagrams show a portion of the parent diagram in greater detail

DFD Hierarchy (1 of 3)
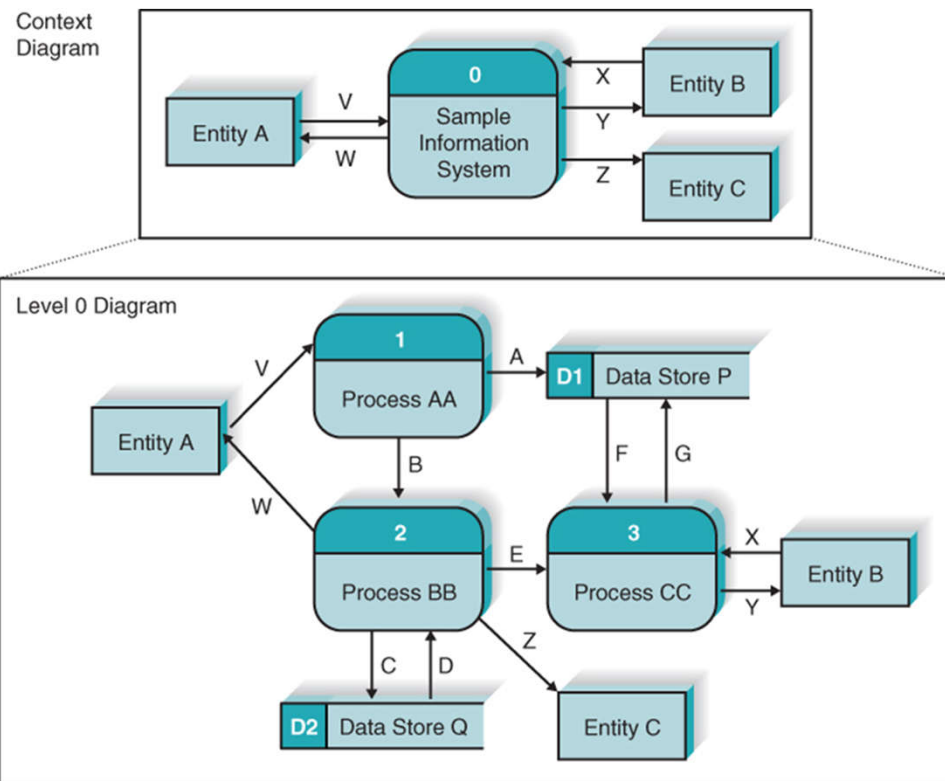
DFD Hierarchy (2 of 3)

- Processes on Level 0 diagram each decompose into separate Level 1 diagrams

- Processes on Level 1 diagrams may or may not be decomposed into separate Level 2 diagrams

- Processes are decomposed until each process is a single-purpose, primitive process

DFD Hierarchy (3 of 3)
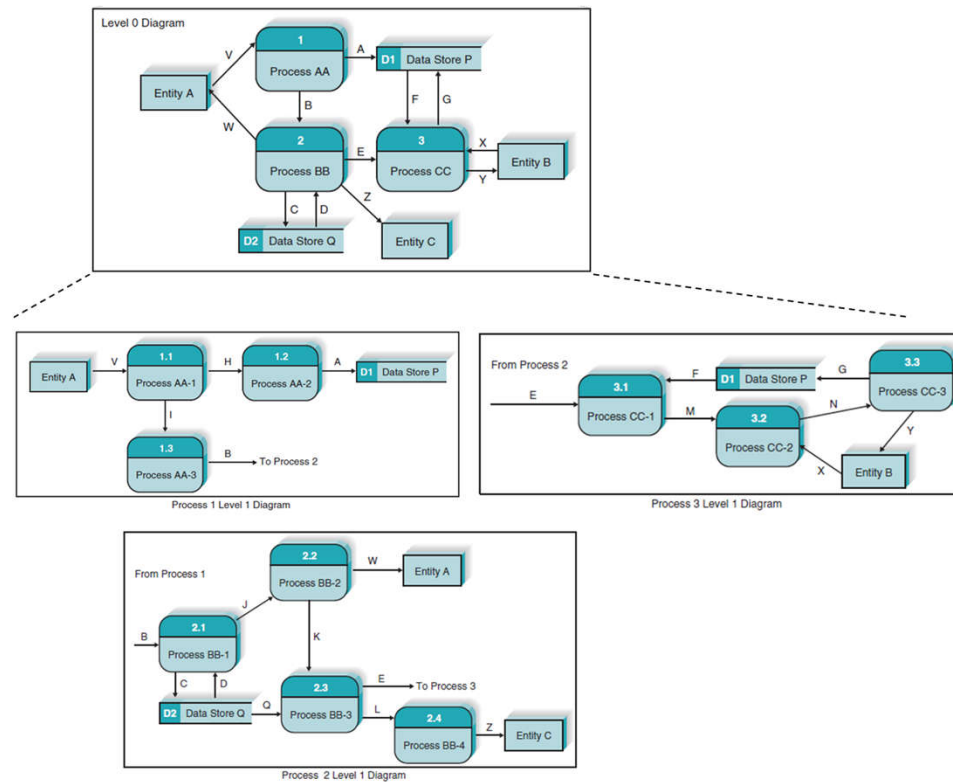
Balancing

- Ensures that information presented at one level of a DFD is accurately represented in the next level DFD

- Data flows on parent diagram are carried down to child diagram

- Child diagram adds new processes and new data flows

Context Diagram

- Top-level DFD in every process model

- Shows the context into which the business process fits

- Shows the overall business process as just one process (process 'zero')

- Shows all the external entities that receive information from or contribute information to the system

Context Diagram

| Entity A | V / W | 0 Sample Information System | X / Y | Entity B |
| | | | Z | Entity C |
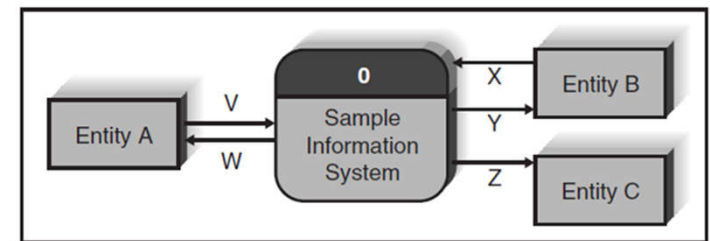
Level 0 Diagram

- Shows all the major processes that comprise the overall system
  - Internal components of process 0

- Shows how the major processes are interrelated by data flows

- Shows external entities and the major processes with which they interact

- Adds stored data via the data stores
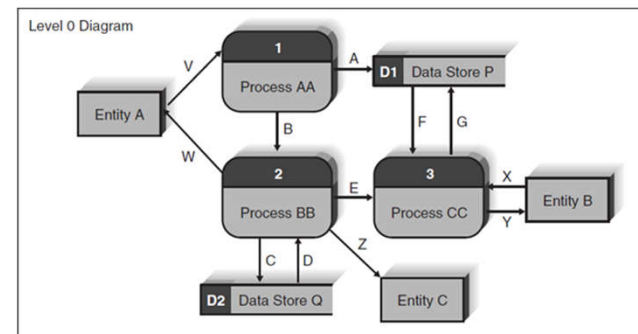
Level 0 Diagram

Level 1 Diagrams

- Create one level 1 diagram for every major process on level 0 diagram

- Shows the internal processes that comprise a single process on the level 0 diagram

- Shows how information moves to and from each of these processes

- If a parent process is decomposed into, say, three child processes, then these three child processes wholly and completely make up the parent process

Process 1 Level 1 Diagram
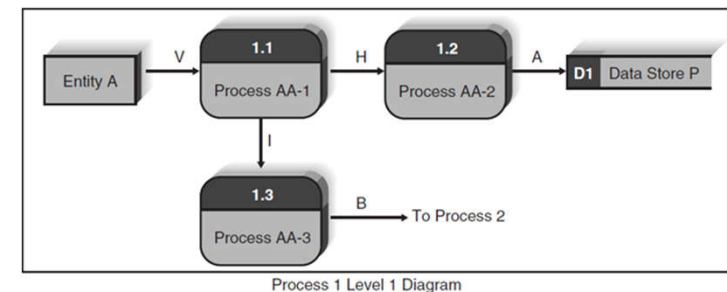
Level 2 Diagrams

- Shows all processes that comprise a single process on the level 1 diagram
- Shows how information moves to and from each of these processes
- Level 2 diagrams may not be needed for all level 1 processes
- Correctly numbering each process helps the user understand where the process fits into the overall system

Diagram Numbering

- Correctly numbering each process helps the user understand where the process fits into the overall hierarchy

- Context Diagram is always "Process 0"

- Level 0 processes are always numbered with integer value (1, 2, 3, and so on)

- Level 1 processes always have one "dot": parent number "dot" unique number (1.1, 1.2, 1.3, and so on)

- Level 2 processes always have two "dots": parent number "dot" unique number (1.1.1, 1.1.2, 1.1.3, and so on)

Alternative Data Flows

- Where a process can produce different data flows given different conditions

- We show both data flows and use the process description to explain why they are alternatives

- Tip -- alternative data flows often accompany processes with IF statements

Process Descriptions

- Text-based process descriptions provide more information about the process than the DFD alone

- CASE tools enable easy creation of descriptions

- If the logic underlying the process is quite complex, more detail may be needed in the form of:
  - Structured English
  - Decision trees
  - Decision tables

Creating Data Flow Diagrams (1 of 2)

- Build the context diagram
  - Identify the external entities and the major inflows they supply and the outflows they receive

- Identify all major processes encompassed by the Context Diagram
  - Each major event / use case is "handled" by a process

- Create DFD "fragments" for each event / use case
  - Each DFD fragment is a mini-diagram showing the process and the external entities and data stores with which it interacts

Creating Data Flow Diagrams (2 of 2)

- Organize DFD fragments into level 0 diagram

- Decompose each level 0 process into a level 1 diagram; decompose level 1 processes into level 2 diagrams as needed; etc.

- Validate DFDs with user to ensure completeness and correctness

Illustration – Developing DFDs

1. Creating the context diagram
2. Create a DFD "fragment" based on a use case
3. Merge DFD "fragment" diagrams into the Level 0 diagram
4. Develop level 1 diagrams for every process on the level 0 diagram

Creating the Context Diagram
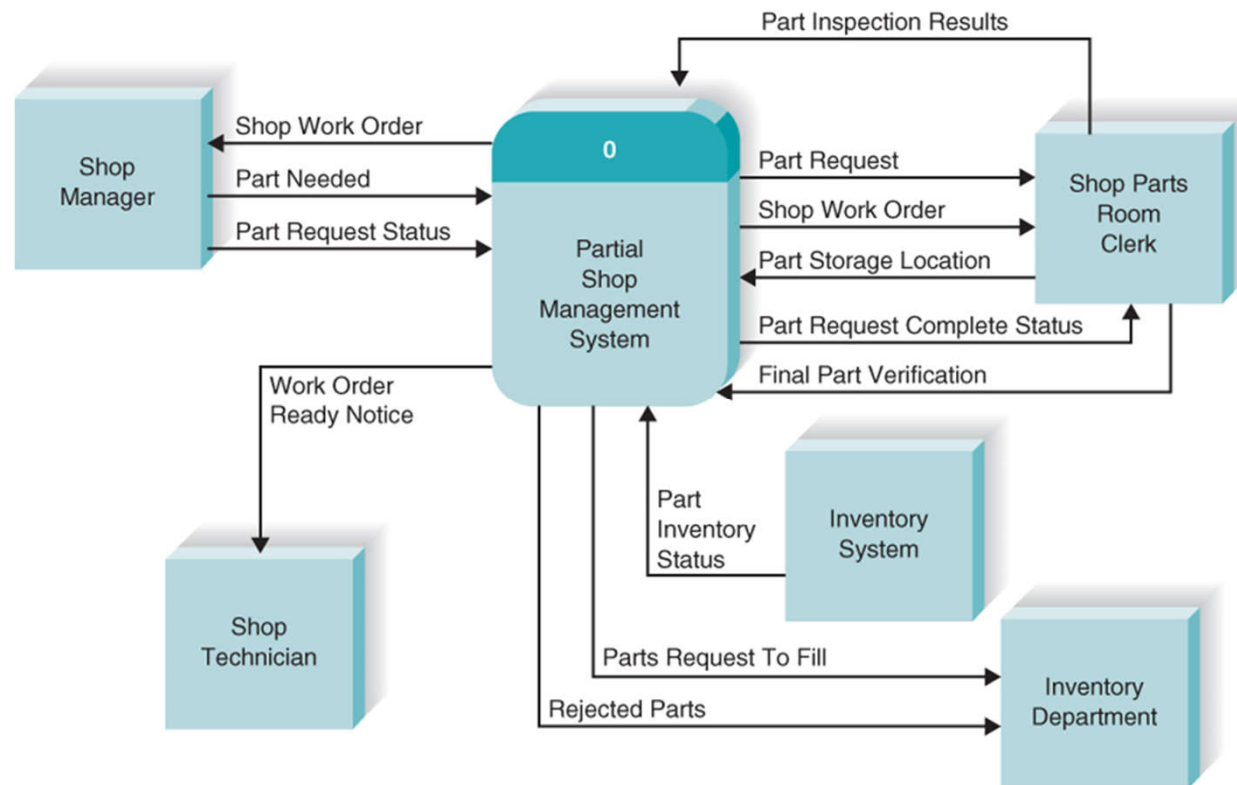
Create a DFD "Fragment" Based on a Use Case

Use Case Name: Create Parts Request		ID: UC-3	Priority: High
Actor: Shop Manager			
Description: This use case describes how the Shop Manager creates a Parts Request			
Trigger: Shop Manager receives notice of new shop work order arrival from Sales System			
Type: ☑ External ☐ Temporal			

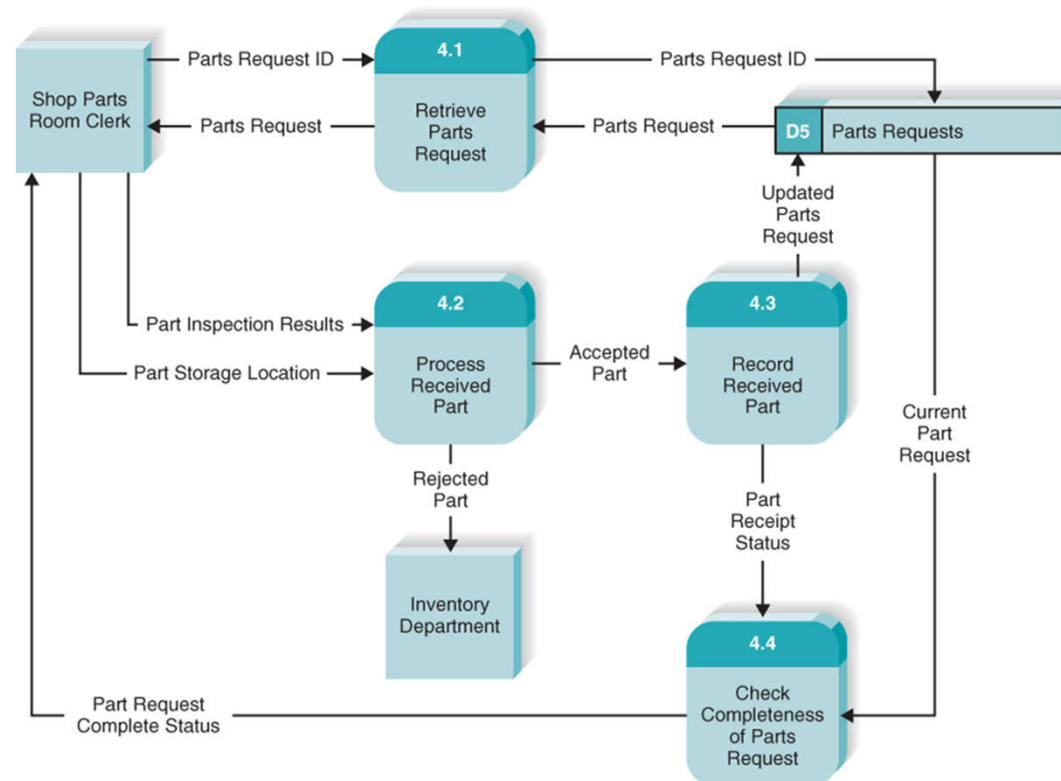| Summary Inputs	Source	Summary Outputs	Destination
Part ID and quantity	Shop manager	Shop work order	Shop manager
Part inventory status	Inventory System	New part request record	Parts Request datastore
Part request status	Shop manager	Parts Request to fill	Inventory department
Shop work order	Shop work order data store		
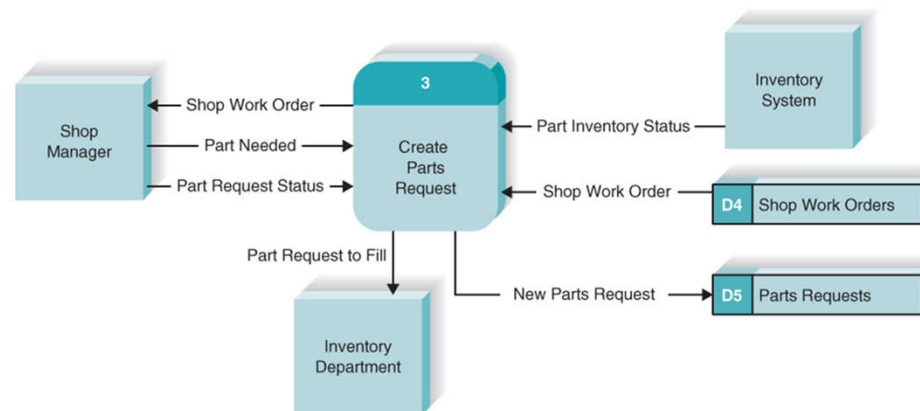
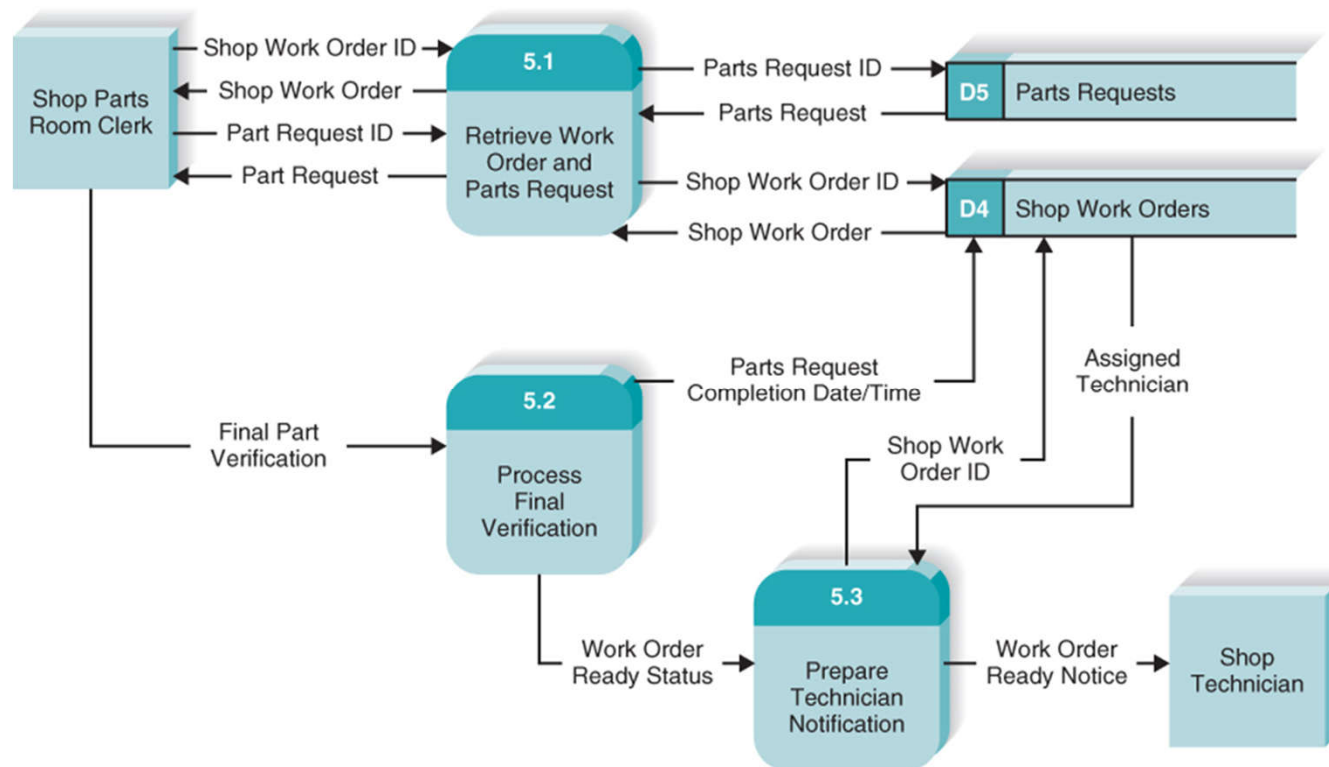# Merge DFD "Fragment" Diagrams into the Level 0 Diagram
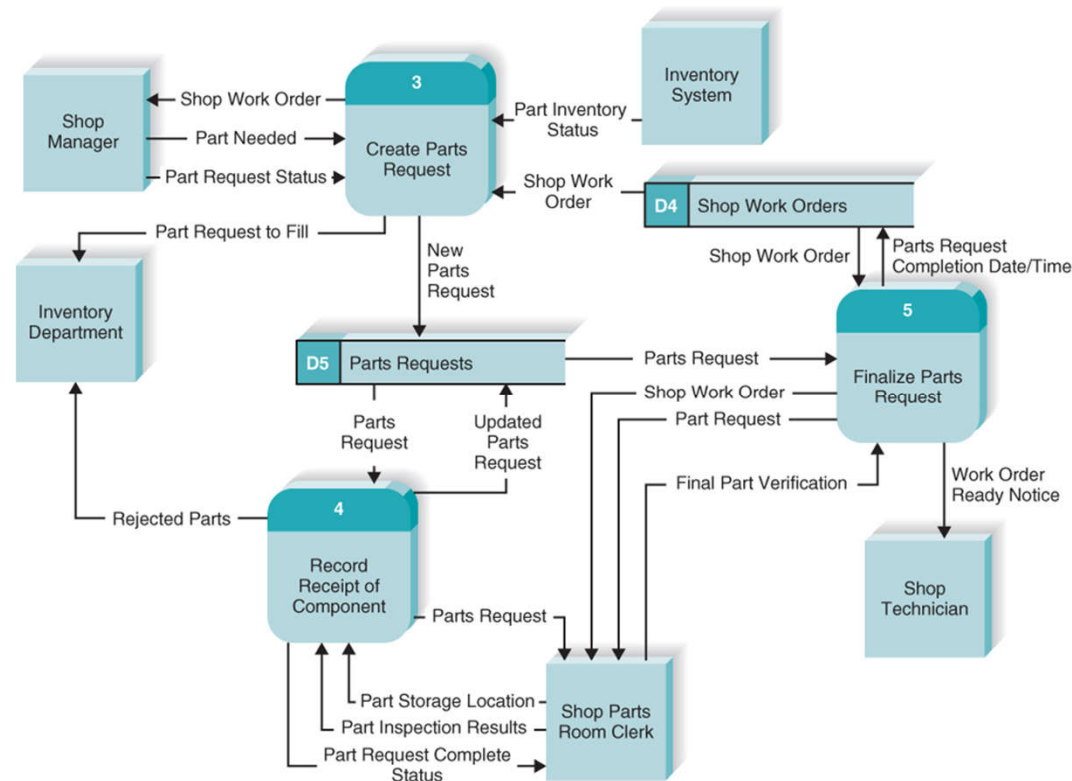
Level 1 – Process 3: Create Parts Request
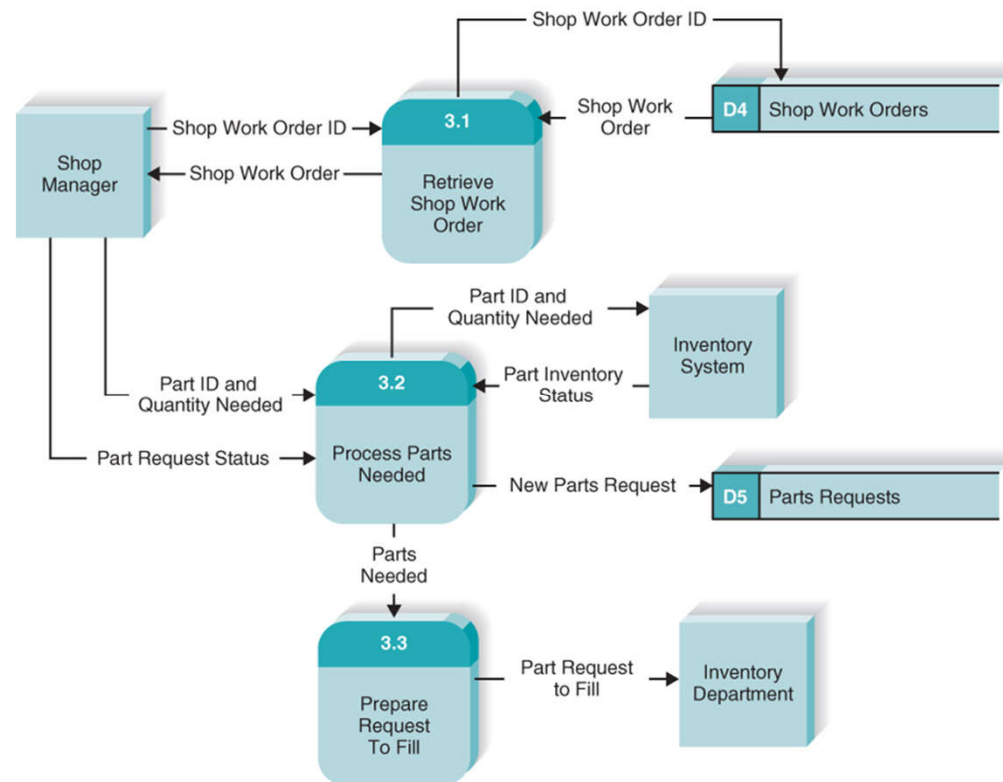
Level 1 – Process 4: Record Receipt of Components

Level 1 – Process 5: Finalize Parts Request

Validating the Data Flow Diagrams (1 of 2)

Within DFD	
Process	• Every process has a unique name that is an action-oriented verb phrase, a number, and a description. • Every process has at least one input data flow. • Every process has at least one output data flow. • Output data flows usually have different names than input data flows because the process changes the input into a different output in some way. • There are between three and seven processes per DFD.
Data Flow	• Every data flow has a unique name that is a noun, and a description. • Every data flow connects to at least one process. • Data flows only in one direction (no two-headed arrows). • A minimum number of data flow lines cross.
Data Store	• Every data store has a unique name that is a noun, and a description. • Every data store has at least one input data flow (which means to add new data or change existing data in the data store) on some page of the process model. • Every data store has at least one output data flow (which means to read data from the data store) on some page of the process model.
External Entity	• Every external entity has a unique name that is a noun, and a description. • Every external entity has at least one input or output data flow.

Validating the Data Flow Diagrams

Across DFDs	
Context diagram	• Every set of DFDs must have one context diagram.
Viewpoint	• There is a consistent viewpoint for the entire set of DFDs.
Decomposition	• Every process is wholly and completely described by the processes on its children DFDs.
Balance	• Every data flow, data store, and external entity on a higher level DFD is shown on the lower-level DFD that decomposes it.
Semantics	
Appropriate Representation	• User validation • Role-play processes
Consistent Decomposition	• Examine lowest-level DFDs
Consistent Terminology	• Examine names carefully

Common D F D Errors

- Syntax errors – violating "drawing" rules
  - Every data flow must connect to a process
  - Every process must have at least one inflow and one outflow
- Semantics errors – errors in the meaning of the diagrams
  - Walk-through diagrams with users
  - Verify that inputs shown are logically sufficient to produce the outputs
  - Check for consistent levels of decomposition
  - Check for consistent use of terminology

Chapter Review

- Explain the purpose of a use case in the analysis phase of the SDLC.
- Explain why use cases are commonly used in the analysis phase.
- Discuss the various sections found in a use case form and the purpose and content of each section.
- Explain how use cases help the systems analyst create a more in-depth understanding of the system's functional requirements.
- Describe how use cases can contribute to the development of test plans for the new system.

Chapter Review

- Discuss the four steps of the process used to create use cases.
- Define the meaning and purpose of the four basic symbols found on a data flow diagram.
- Explain the meaning and purpose of a process model's context diagram.
- Explain the meaning and purpose of a process model's level 0 diagram.
- Explain the meaning and purpose of a process model's level 1 diagrams.

Chapter Review

- Explain the concept of decomposition and why process models are created as a hierarchy of DFDs.

- Describe several common syntax and semantic errors found on DFDs.

- Discuss the process used to create a process model.

- Discuss how the process model contributes to the development of the new information system.

Key Terms

- Actor
- Balancing
- Bundle
- Children
- Context diagram
- Data flow
- Data flow diagram
- (DFD)
- Data store
- Decision tables

- Decision trees
- Decomposition
- DFD fragment
- Essential use cases
- Event-driven modeling
- Event triggers
- External entity
- External trigger
- Happy path
- IF statements

- Inputs
- Iteration
- Layout
- Level 0 diagram or level 0 DFD
- Level 1 diagram, or level 1 DFD
- Level 2 DFD
- Logical process model
- Outputs

Key Terms Continued

- Parent
- Physical model
- Postconditions
- Preconditions
- Primary actor
- Priority
- Process
- Processes

- Process model
- Role-play
- Semantics error
- Step
- Structured English
- Syntax error
- Temporal trigger
- Trigger

- Use case
- Use case packages
- User requirements
- User role
- Viewpoint
- Visualization