


Object-oriented Analysis and Design




## Applying UML and Patterns

An Introduction to  
Object-oriented Analysis  
and Design  
and Iterative Development

Part III Elaboration Iteration I – Basic<sup>2</sup>

Software Engineering


Object-oriented Analysis and Design



## Chapter 15 UML Interaction Diagrams

Software Engineering

Object-oriented Analysis and Design




## Introduction

- The UML includes **interaction diagrams** to illustrate how objects interact via messages.
  - **sequence** and **communication** interaction diagrams.
- This chapter introduces the notation - view it as a reference to skim through - while subsequent chapters focus on a more important question: What are key principles in OO design?

Software Engineering

Object-oriented Analysis and Design




## Sequence and Communication Diagrams

- The term interaction diagram is a generalization of two more specialized UML diagram types:
  - sequence diagrams
  - communication diagrams
- Both can express similar interactions
- A related diagram is the interaction overview diagram;
  - provides a big-picture overview of how a set of interaction diagrams are related in terms of **logic and process-flow**.
  - It's new to UML 2, and so it's too early to tell if it will be practically useful.

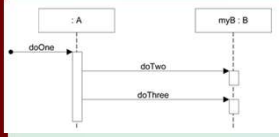
Software Engineering

Object-oriented Analysis and Design



## Sequence Diagram

□ Sequence diagrams illustrate interactions in a kind of fence format, in which each new object is added to the right,




```

public class A {
    private B myB = new B();
    public void doOne() {
        myB.doTwo();
        myB.doThree();
    }
    // ...
}
    
```

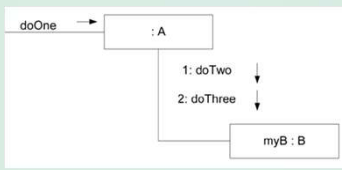
Software Engineering

Object-oriented Analysis and Design



## Communication Diagram

□ Communication diagrams illustrate object interactions in a graph or network format, in which objects can be placed anywhere on the diagram



Software Engineering

Object-oriented Analysis and Design

## Strengths and Weaknesses 1

- Sequence diagrams have some advantages over communication diagrams
  - UML specification is more sequence diagram centric - more thought and effort has been put into the notation and semantics.
    - ◆ Thus, tool support is better and more notation options are available
  - it is easier to see the call-flow sequence with sequence diagrams simply read top to bottom.
    - ◆ With communication diagrams we must read the sequence numbers, such as "1:" and "2:"

Software Engineering

Object-oriented Analysis and Design

## Strengths and Weaknesses 2

- Advantages of communication diagrams
  - communication diagrams have advantages when applying "UML as sketch" to draw on walls (an Agile Modeling practice) because they are much more space-efficient.
  - boxes can be easily placed or erased anywhere horizontal or vertical.
  - In contrast, new objects in a sequence diagrams must always be added to the right edge, which is limiting as it quickly consumes and exhausts right-edge space on a page (or wall)

Software Engineering

Object-oriented Analysis and Design

## Strengths and Weaknesses 3

Type	Strengths	Weaknesses
sequence	clearly shows sequence or time ordering of messages large set of detailed notation options	forced to extend to the right when adding new objects; consumes horizontal space
communication	space economical flexibility to add new objects in two dimensions	more difficult to see sequence of messages fewer notation options

Software Engineering

Object-oriented Analysis and Design

## Example Sequence Diagram

```

sequenceDiagram
    participant Actor
    Actor->>Register: makePayment(cashTendered)
    activate Register
    Register->>Sale: makePayment(cashTendered)
    activate Sale
    Sale-->>Payment: create(cashTendered)
    deactivate Sale
    deactivate Register
    
```

- The message *makePayment* is sent to an instance of a Register. The sender is not identified.
- The Register instance sends the *makePayment* message to a Sale instance.
- The Sale instance creates an instance of a Payment.

What might be some related code for the Sale class and its *makePayment* method?

Software Engineering

Object-oriented Analysis and Design

## Example Communication Diagram

```

classDiagram
    class Register
    class Sale
    class Payment
    Register --> Sale : makePayment(cashTendered)
    Sale --> Payment : create(cashTendered)
    
```

```

public class Sale {
    private Payment payment;
    public void makePayment ( Money cashTendered ) {
        payment = new Payment ( cashTendered );
        //...
    }
    // ...
}

```

Software Engineering

Object-oriented Analysis and Design

## Common UML Interaction Diagram Notation

Software Engineering

Object-oriented Analysis and Design

### Basic Message Expression Syntax

- UML has a standard syntax for these message expressions  
`return = message(parameter : parameterType) : returnType`
- Parentheses are usually excluded if there are no parameters, though still legal.
- Type information may be excluded if obvious or unimportant.
- For example:
  - `initialize(code)`
  - `initialize`
  - `d = getProductDescription(id)`
  - `d = getProductDescription(id:ItemID)`
  - `d = getProductDescription(id:ItemID) : ProductDescription`

Software Engineering

Object-oriented Analysis and Design

### Singleton Objects

- In the world of OO design patterns, there is one that is especially common, called the Singleton pattern
- There is only one instance of a class instantiated - never two

Software Engineering

Object-oriented Analysis and Design

### Basic Sequence Diagram Notation

- Lifeline Boxes and Lifelines
  - In sequence diagrams the lifeline boxes include a vertical line extending below them - these are the actual lifelines.
  - Although virtually all UML examples show the lifeline as dashed (because of UML 1 influence), in fact the UML 2 specification says it may be solid or dashed.
- Messages
  - Each (typical synchronous) message between objects is represented with a message expression on a filled-arrowed solid line between the vertical lifelines
  - The time ordering is organized from top to bottom of lifelines.

Software Engineering

Object-oriented Analysis and Design

### Message and Focus of Control 1

- found message:
  - the sender will not be specified, is not known, or that the message is coming from a random source

Software Engineering

Object-oriented Analysis and Design

### Message and Focus of Control 2

- Sequence diagrams may also show the focus of control using an execution specification bar (previously called an activation bar or simply an activation in UML 1).
  - The bar is optional.
  - Guideline: Drawing the bar is more common (and often automatic) when using a UML CASE tool, and less common when wall sketching.

Software Engineering

Object-oriented Analysis and Design

### Illustrating Reply or Returns

- There are two ways to show the return result from a message:
  - Using the message syntax
    - ◆ `returnVar = message(parameter).`
  - Using a reply (or return) message line at the end of an activation bar.

Software Engineering

Object-oriented Analysis and Design

### Creation of Instances

- The arrow is filled if it's a regular synchronous message (such as implying invoking a Java constructor), or open (stick arrow) if an asynchronous call.
  - The message name *create* is not required - anything is legal - but it's a UML idiom.
- Object Lifelines and Object Destruction

the «destroy» stereotyped message, with the large X and short lifeline indicates explicit object destruction

Software Engineering 19

Object-oriented Analysis and Design

### Diagram Frames in UML Sequence Diagrams 1

- To support conditional and looping constructs (among many other things), the UML uses frames.
  - Frames are regions or fragments of the diagrams;
  - they have an operator or label (such as loop) and a guard (conditional clause).

a UML loop frame, with a boolean guard expression

Software Engineering 20

Object-oriented Analysis and Design

### Diagram Frames in UML Sequence Diagrams 2

Frame Operator	Meaning
alt	Alternative fragment for mutual exclusion conditional logic expressed in the guards.
loop	Loop fragment while guard is true. Can also write <i>loop(n)</i> to indicate looping <i>n</i> times. There is discussion that the specification will be enhanced to define a <i>FOR</i> loop, such as <i>loop(i, 1, 10)</i>
opt	Optional fragment that executes if guard is true.
par	Parallel fragments that execute in parallel.
region	Critical region within which only one thread can run.

Software Engineering 21

Object-oriented Analysis and Design

### Condition message

- Use UML 1 style only for simple single messages when sketching

Software Engineering 22

Object-oriented Analysis and Design

### Mutually Exclusive Conditional Messages

- An ALT frame is placed around the mutually exclusive alternatives

Software Engineering 23

Object-oriented Analysis and Design

### Iteration Over a Collection 1

This lifeline box represents one instance from a collection of many *SalesLineItem* objects.

*lineItems[i]* is the expression to select one element from the collection of many *SalesLineItems*; the "i" value refers to the same "i" in the guard in the LOOP frame

an action box may contain arbitrary language statements (in this case, incrementing "i") it is placed over the lifeline to which it applies

- A common algorithm is to iterate over all members of a collection (such as a list or map), sending the same message to each.
- Often, some kind of iterator object is ultimately used, such as an implementation of *java.util.Iterator* or a C++ standard library iterator, although in the sequence diagram that low-level "mechanism" need not be shown in the interest of brevity or abstraction.

Software Engineering 24

Object-oriented Analysis and Design

### Iteration Over a Collection 2

- The selector expression is used to select one object from a group. Lifeline participants should represent one object, not a collection.

```

public class Sale {
    private List<SalesLineItem> lineItems = new ArrayList<SalesLineItem>();
    public Money getTotal() {
        Money total = new Money();
        Money subtotal = null;
        for ( SalesLineItem lineItem : lineItems ) {
            subtotal = lineItem.getSubtotal();
            total.add( subtotal );
        }
        return total;
    }
    // ...
}
    
```

Software Engineering

Object-oriented Analysis and Design

### Iteration Over a Collection 3

- Another variation is shown below
  - The intent is the same, but details are excluded.
  - A team or tool could agree on this simple style by convention to imply iteration over all the collection elements

Software Engineering

Object-oriented Analysis and Design

### Relating Interaction Diagrams 1

Software Engineering

Object-oriented Analysis and Design

### Relating Interaction Diagrams 2

- An interaction occurrence (also called an interaction use) is a reference to an interaction within another interaction.
  - for example, when you want to simplify a diagram and factor out a portion into another diagram, or there is a reusable interaction occurrence.
  - UML tools take advantage of them, because of their usefulness in relating and linking diagrams.

Software Engineering

Object-oriented Analysis and Design

### Metaclass 1

- You can show class or static method calls by using a lifeline box label that indicates the receiving object is a class, or more precisely, an instance of a metaclass
- in Java and Smalltalk, all classes are conceptually or literally instances of class Class;
- in .NET classes are instances of class Type. The classes Class and Type are metaclasses, which means their instances are themselves classes.
- A specific class, such as class Calendar, is itself an instance of class Class. Thus, class Calendar is an instance of a metaclass! It may help to drink some beer before trying to understand this.

Software Engineering

Object-oriented Analysis and Design

### Metaclass 2

```

public class Foo {
    public void doX() {
        // static method call on class
        Calendar Locale[] locales = Calendar.getAvailableLocales();
        // ...
    }
    // ...
}
    
```

Software Engineering

Object-oriented Analysis and Design

### Polymorphic Messages and Cases

- How to show it in a sequence diagram? That's a common UML question.
  - To use multiple sequence diagrams - one that shows the polymorphic message to the abstract superclass or interface object, and then separate sequence diagrams detailing each polymorphic case, each starting with a found polymorphic message

Software Engineering

Object-oriented Analysis and Design

### Asynchronous and Synchronous Calls

- An asynchronous message call does not wait for a response; it doesn't block.
- They are used in multi-threaded environments such as .NET and Java so that new threads of execution can be created and initiated.
- In Java, for example, you may think of the *Thread.start* or *Runnable.run* (called by *Thread.start*) message as the asynchronous starting point to initiate execution on a new thread

Software Engineering

Object-oriented Analysis and Design

### Conti.

a stick arrow in UML implies an asynchronous call  
 a filled arrow is the more common synchronous call

In Java, for example, an asynchronous call may occur as follows:

```
// Clock implements the Runnable interface
Thread t = new Thread( new Clock() );
t.start();
```

the asynchronous start call always invokes the run method on the Runnable (Clock) object

to simplify the UML diagram, the Thread object and the start message may be avoided (they are standard "overhead"); instead, the essential detail of the Clock creation and the run message imply the asynchronous call

Software Engineering

Object-oriented Analysis and Design

### Conti.

Software Engineering

Object-oriented Analysis and Design

### Basic Communication Diagram Notation

- A **link** is a connection path between two objects;
  - it indicates some form of navigation and visibility between the objects is possible
  - More formally, a link is an instance of an association.

Software Engineering

Object-oriented Analysis and Design

### Message

- Messages**
  - Each message between objects is represented with a message expression and small arrow indicating the direction of the message.
  - Many messages may flow along this link.
  - A sequence number is added to show the sequential order of messages in the current thread of control.

Software Engineering

Object-oriented Analysis and Design

### Creation of Instances 1

- Any message can be used to create an instance, but the convention in the UML is to use a message named create for this purpose (some use new).
- If another (less obvious) message name is used, the message may be annotated with a UML stereotype, like so: «create».
- The create message may include parameters, indicating the passing of initial values. This indicates, for example, a constructor call with parameters in Java.
- Furthermore, the UML tagged value {new} may optionally be added to the lifeline box to highlight the creation.
- Tagged values are a flexible extension mechanism in the UML to add semantically meaningful information to a UML element.

Software Engineering

Object-oriented Analysis and Design

### Creation of Instances 2

Three ways to show creation in a communication diagram

create message with optional initializing parameters. This will normally be interpreted as a constructor call.

if an unobvious creation message name is used the message may be stereotyped for clarity

Software Engineering

Object-oriented Analysis and Design

### Message Number Sequencing 1

- The order of messages is illustrated with sequence numbers. The numbering scheme is:
  - The first message is not numbered. Thus, msg1 is unnumbered
  - Actually, a starting number is legal, but it makes all subsequent numbering more awkward, creating another level of number-nesting deeper than otherwise necessary.
  - The order and nesting of subsequent messages is shown with a legal numbering scheme in which nested messages have a number appended to them.
  - You denote nesting by pre-pending the incoming message number to the outgoing message number

Software Engineering

Object-oriented Analysis and Design

### Message Number Sequencing 2

not numbered

legal numbering

Software Engineering

Object-oriented Analysis and Design

### Message Number Sequencing 3

Software Engineering

Object-oriented Analysis and Design

### Conditional Messages

- A conditional message is shown by following a sequence number with a conditional clause in square brackets, similar to an iteration clause.
- The message is only sent if the clause evaluates to true.

conditional message, with test

Software Engineering

