Sequence Diagrams

Topic # 10

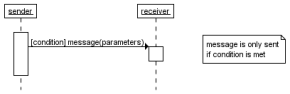
Chapter 15 – UML Interaction Diagrams

Chapter 10 – System Sequence Diagram

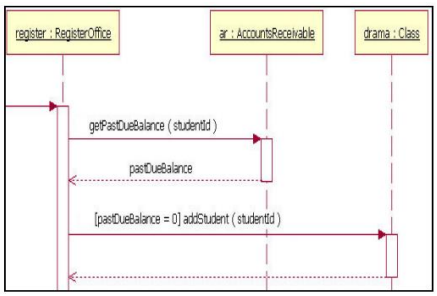
Craig Larman

Guards

- There will be times when a condition must be met for a message to be sent to the object. i.e Conditional interaction
- There will be certain prerequisite for communication or a message to be sent to the sender.
- These conditions are attributed as "Guard" in sequence diagram, a guard behaves likes "if statements" in the sequence diagram.
- They are used to control the flow of the messages between objects.

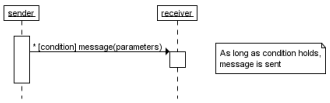


Guards



Repeated Interaction

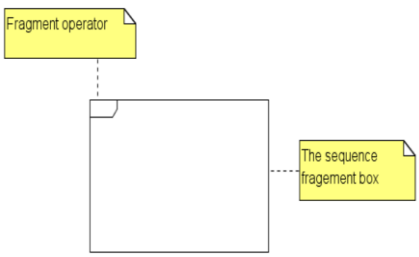
- When a message is prefixed with an asterisk (the '*'-symbol), it means that the message is sent repeatedly. A guard indicates the condition that determines whether or not the message should be sent (again). As long as the condition holds, the message is repeated.



Fragments

- Sequence diagrams can be broken up into chunks called fragments or combined fragments.
- **Manage complex interactions with sequence fragments**
- It is used to show complex interactions such as alternative flows and loops in a more structured way. On the top left corner of the fragment sits an operator. This – the fragment operator – specifies what sort of a fragment it is.
  - Fragment types: ref, loop, break, alt, opt, parallel
- Sequence fragments make it easier to create and maintain accurate sequence diagrams

Combined Fragment



Frame

- `If ->` (opt) [condition]
- `if/else` -> (alt) [condition], separated by horizontal dashed line
- loop -> (loop) [condition or items to loop over]

Frame Operator	Description
alt	Alternative fragment for mutual exclusive logic expressed in the guards.
loop	Loop fragment while guard is true. Can also write loop(n) to indicate looping n times.
Opt	Optional fragment that execute if the guard is true.
par	Parallel fragments that execute in parallel.
region	Critical region within which only one thread can run.

Conditional Behavior-Option

- The option combination fragment is used to model a sequence that, given a certain condition, will occur; otherwise, the sequence does not occur.
- An option is used to model a simple "if then" statement
  - Example, if there are fewer than five donuts on the shelf, then make two dozen more donuts.

Conditional Behavior-Alt

- When showing conditional behavior, the interaction operator keyword **alt** is put in the pentagram, the fragment is partitioned horizontally with a dashed line separator, and constraints are shown in square brackets .
- At most one of the alternatives occurs;

Loops

- A repetition or loop within a sequence diagram is depicted as a frame.
- In the frame's name box the text "loop" is placed.
- Loops are designated by placing the interaction operator keyword loop in the pentagram. Textual syntax of the loop operand is “loop [‘(‘ <minint> [,<maxint>] ‘)’]” .
- Inside the frame's content area the loop's guard is placed towards the top left corner, on top of a lifeline.

A simple example of two operations being repeated five times.

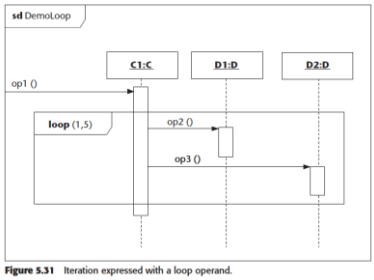
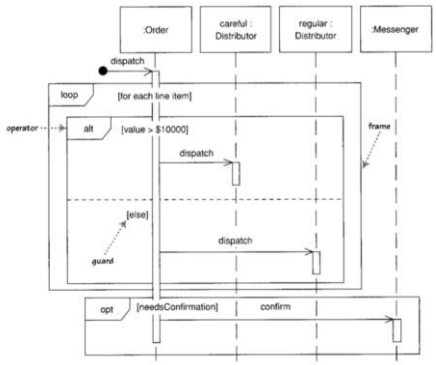
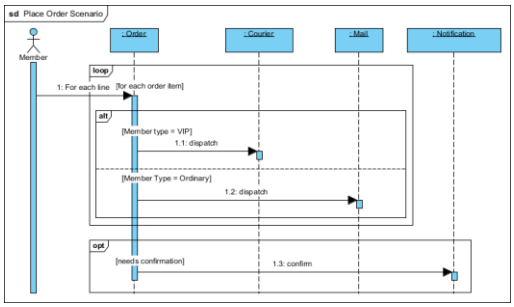


Figure 5.31 Iteration expressed with a loop operand.



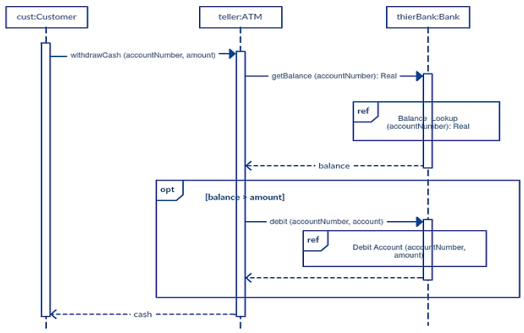
Indicating selection and loops



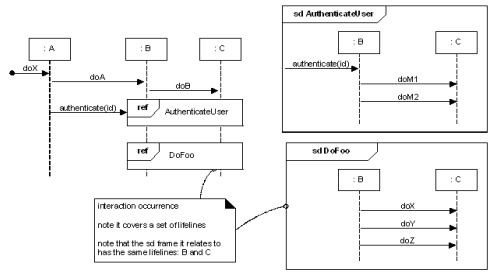
Reference Fragment

- You can use the **ref** fragment to manage the size of large sequence diagrams. It allows you to reuse part of one sequence diagram in another, or in other words, you can reference part of a diagram in another diagram using the ref fragment.
- To specify the reference fragment, you have to mention 'ref' in the name box of the frame and the name of the sequence diagram that is being referred to inside the frame.

Reference Fragment- Example



Reference Fragment- Example



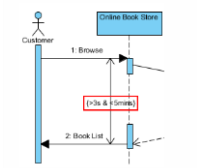
Common Operators for Interaction Frames

Operator	Meaning
alt	**Alternative multiple fragments:** only the one whose condition is true will execute.
opt	**Optional:** the fragment executes only if the supplied condition is true. Equivalent to an alt only with one trace.
par	**Parallel:** each fragment is run in parallel.
loop	**Loop:** the fragment may execute multiple times, and the guard indicates the basis of iteration.
region	**Critical region:** the fragment can have only one thread executing it at once.
neg	**Negative:** the fragment shows an invalid interaction.
ref	**Reference:** refers to an interaction defined on another diagram. The frame is drawn to cover the lifelines involved in the interaction. You can define parameters and a return value.
sd	**Sequence diagram:** used to surround an entire sequence diagram.

Specifying Timing Requirements

- When modeling a real-time system, or even a time-bound business process, it can be important to consider the length of time it takes to perform actions.
- Assume you need to specify the time limit between Browse message and Book List message, you therefore, have to add duration constraint between them.
- For example, it should take more than 3 seconds but less than 5 minutes.

You can enter > 3s & < 5mins.



How to Produce Sequence Diagrams

1. Decide on Context:
   a) Identify behavior (or use case) to be specified
2. Identify structural elements:
   a) Model objects (classes)
   b) Model lifelines
   c) Model activations
   d) Model messages
   e) Model Timing constraints
3. Elaborate as required

Why not just code it?

- Sequence diagrams can be somewhat close to the code level. So why not just code up that algorithm rather than drawing it as a sequence diagram?
  - A good sequence diagram is still a bit above the level of the real code (not all code is drawn on diagram)
  - Non-coders can do sequence diagrams
  - Easier to do sequence diagrams as a team
  - Can see many objects/classes at a time on same page (visual bandwidth)

Sequence Diagrams and Use Cases System Sequence Diagram

- System sequence diagrams are actually a sub-type of sequence diagrams.
- Sequence diagrams show the progression of events over a certain amount of time, while system sequence diagrams go a step further and present sequences for specific use cases.

Sequence Diagrams and Use Cases System Sequence Diagram

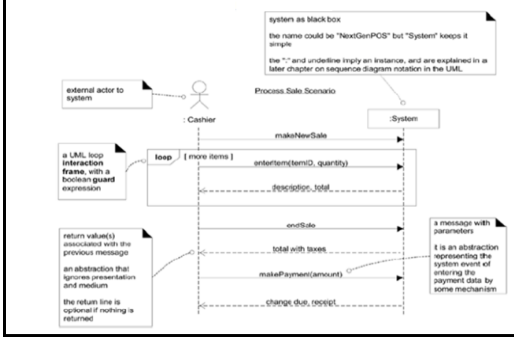
- SSD: The elements participating (exchanging messages) in a system sequence diagram are Actors and Systems. The messages exchanged by these elements could be any type depending on the systems (from web service calls to data input from a human).
- SD: The elements participating in a sequence diagram are objects (instances of various classes). The messages exchanged by these elements are method invocations.

System Sequence Diagram (SSD)

- A system sequence diagram (SSD) is a picture that shows, for a particular scenario of a use case, the events that external actors generate, their order, and inter-system events.
- All systems are treated as a black box; the emphasis of the diagram is events that cross the system boundary from actors to systems.
- An SSD should be done for the main success scenario of the use case, and frequent or complex alternative scenarios.

System Sequence Diagram

SSD for a *Process Sale* scenario.



Sequence to code demo

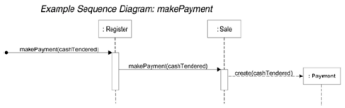

```
public class A
{
private B myB = new B();

public void doOne()
{
   myB.doTwo();
   myB.doThree();
}
// …
}
```

What might this represent in code?[1] Probably, that class *A* has a method named *doOne* and an attribute of type *B*. Also, that class *B* has methods named *doTwo* and *doThree*. Perhaps the partial definition of class *A* is:

** Class A has a method named done and an attribute of type B. Also class B has methods named doTwo and doThree.

**Code mapping or generation rules will vary depending on the OO language

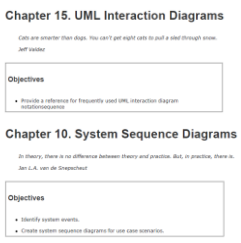
Code demo




```
public class Sale
{
private Payment payment;

public void makePayment( Money cashTendered )
{
   payment = new Payment( cashTendered );
   //…
}
// …
}
```

READING

Chapter 10,15 – Applying UML and Pattern by Craig Larman 3rd Edition


Chapter 15. UML Interaction Diagrams

Cats are smarter than dogs. You can't get eight cats to pull a sled through snow.

Jeff Valdez

Objectives

- Provide a reference for frequently used UML interaction diagram notationsequence

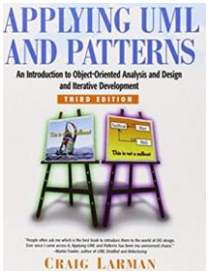
Chapter 10. System Sequence Diagrams

In theory, there is no difference between theory and practice. But, in practice, there is.

Jan L.A. van de Snepscheut

Objectives

- Identify system events.
- Create system sequence diagrams for use case scenarios.



END OF TOPIC 10

-COMING UP!!!!!!

-Collaboration Diagrams
-Timing Diagrams
- 4+ 1 Model View
- Midterm II