UML 2.0 pocket reference

introduction

Object Management Group (OMG).

Model Driven Architecture (MDA)

static modeling with uml

* Class diagrams
* Package diagrams
* Component diagrams
* Composite Structure diagrams
* Deployment diagrams

UML Classifiers

class

component

datatype

interface

node

signal

subsystem

use case

stereotypes

actor

access, Applies to a dependency.

association, Applies to the end of a link.

bind

call

copy

create

derive

destroy

document

enumeration

exception

façade

file

framework

friend

global

import

implementation

implementationClass

include

instanceOf

Instantiate

interface

invariant

library

local

metaclass

model

parameter

powertype

precondition

process

refine, applies to a dependency

requirement

responsibility

self

send

signal

stereotype

stub, apply to a package

subsystem, apply to a package

system

table

thread

trace, origining element is an earlier version of the ending element

type

use

utility

Tag values to specify user defined properties

Constraints:

Complete

Destroyed

Disjoint

Implicit

Incomplete

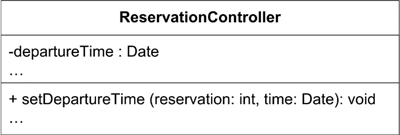
New

Overlapping

Transient, the instance or link is created during the execution of an operation but destroyed before the operation completes

Xor, applies to multiple associations

Class diagram



Class attributes

readOnly

union

substes <attribute-name>

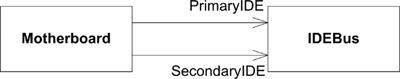
redefines <attribute-name>

composite

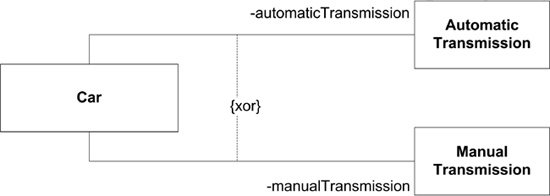
Class relationship

association

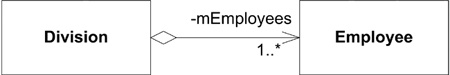




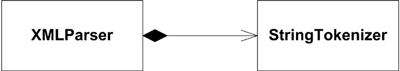
constraints between attributes



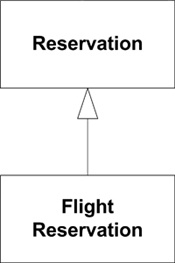
Aggregation is a special type of association used to represent a stronger relationship between two classes than regular associations. It is usually read as “owns a,” as in “ClassA owns a ClassB.”



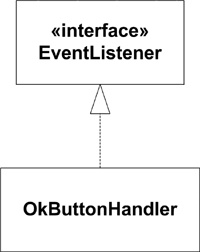
Composition is the strongest relationship between two classes. It is usually read as “is made up of,” as in “Class A is made up of ClassB”



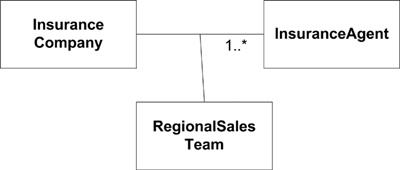
Generalization is used to show inheritance; subclass B has an “is a” relationship with superclass A, or superclass A is a generalization of subclass B.



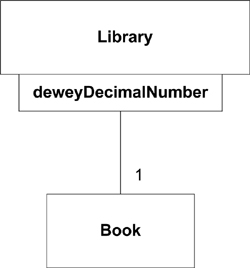
Realization indicates that a class implements, or realizes, an interface at the other end (see the upcoming section, Interfaces). Similar to inheritance, realization indicates that the class realizing an interface is an implementation of the referenced interface.



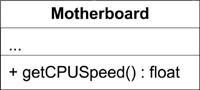
Association classes There are times when the relationship between two classes has information that you want to capture.



Association qualifiers relationships between larger entities and smaller entities are keyed by some unique value.



Operations



will defines

visibility

parameter list

return type

constraints

Polymorphism

abstract operation



properties and constraints on operations

isQuery, indicates that an operation will return a value without modifying the class in any way

sequential, indicates that an operation must be called in only one thread of execution at a time

sequential

guarded, similar to sequence but the operation itself enforces the rule that only a single thread can call

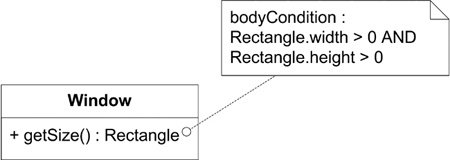
concurrent

three constraints that you may use on your operations

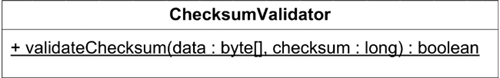
precondition

postcondition

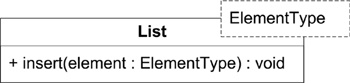
bodyconditioin constraints on the return value. It may be overridden by subclasses



operation scope



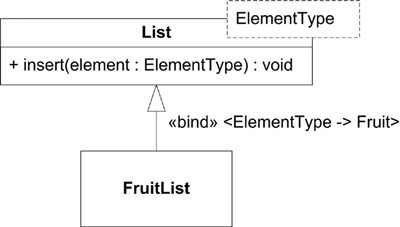
template class



implicit binding



explicit binding



interfaces



icon version interface



Package diagrams

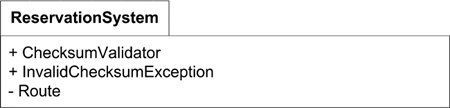


visibility of package elements

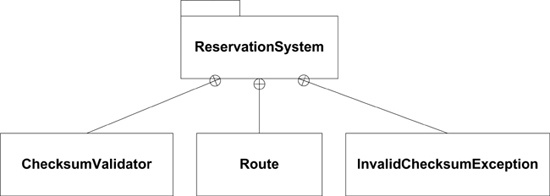
public

protected

private



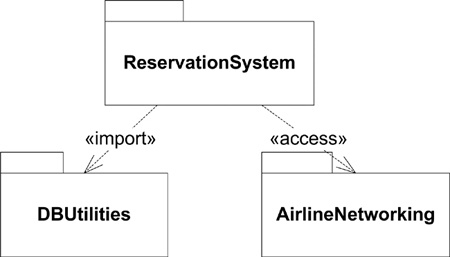
package composition



dependencies between packages

access a package uses the public elements of another package

import a package actually incorporates another package’s public elements into its own namespace



UML stereotypes that are specific to packages

façade

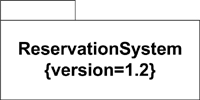
framework

stub

subsystem

system, indicate a package contains an entire system

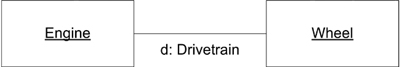
package tagged values



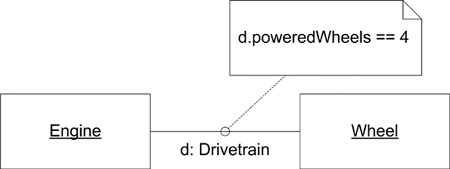
merging packages

UML 2.0 introduces the concept of merging packages. A fairly complex set of rules governs package merging

composite structures



constraints to a connector using a note



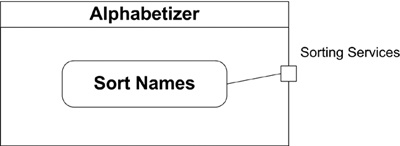
ports A port indicates that a structure offers some functionality without detailing how that functionality is realized internally

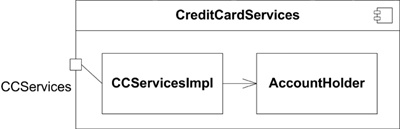


provide and required interfaces

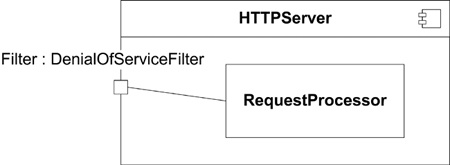


realizing ports, a piece of functionality that must be implemented at some point

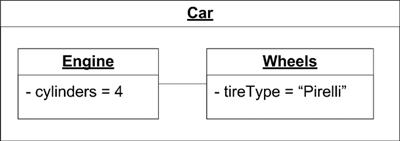




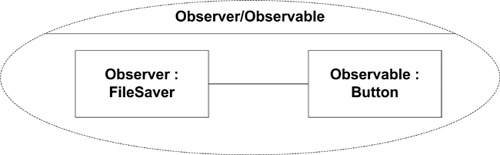
typing ports explicitly type a port as a different classifier



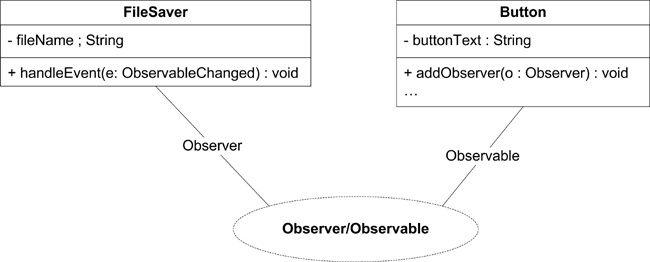
structured classes and properties



collaborations used to model patterns with a dash oval with the name of the collaboration inside



collaborations with details



component diagrams

simple component



types of components

document

entity

executable

file

library

process

realization

service

specification

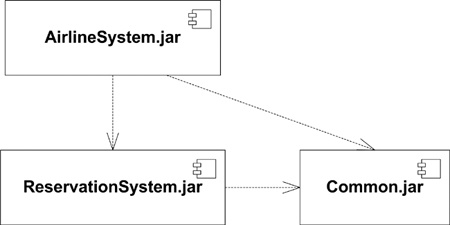
subsystem

table

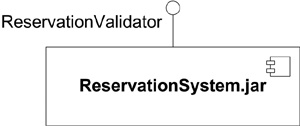
a component showing contained classes within a compartment



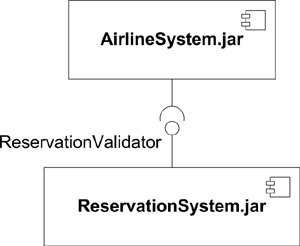
component modeling



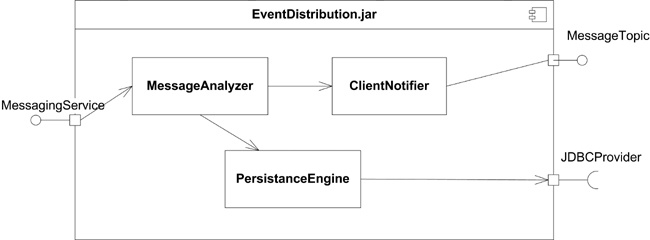
interface realization using stick and lollipop notation



interface realization



component views, white box and black box

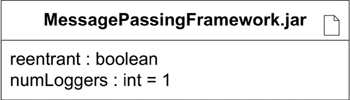




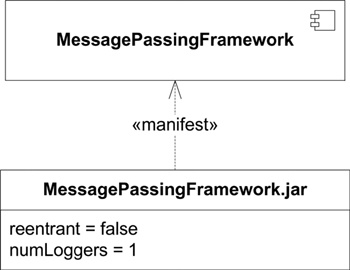
deployment diagrams



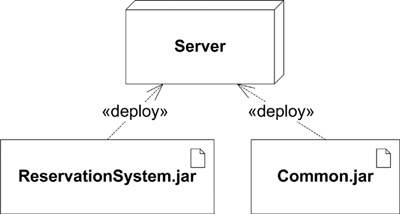
artifact with attributes



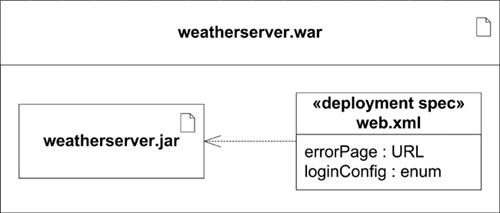
manifestation of a component as an artifact



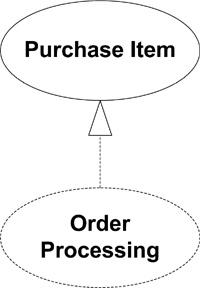
nodes, a physical entity that executes one or more components, subsystems, or executables



UML allows you to capture this information with a stereotype called «deployment spec».



use case realization



use case documents

brief description

preconditions

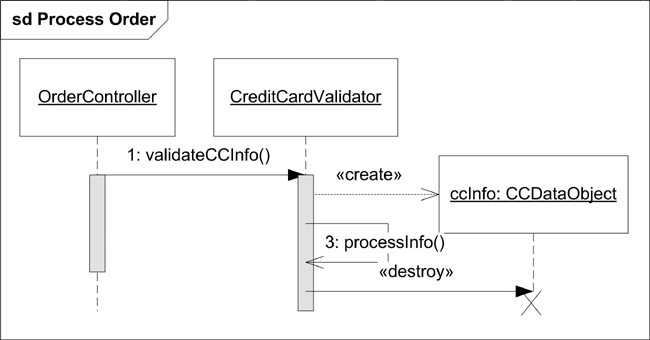
basic flow

alternate flows

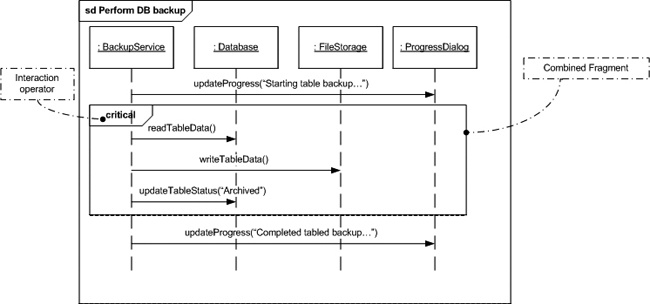
postconditions

Interaction diagrams

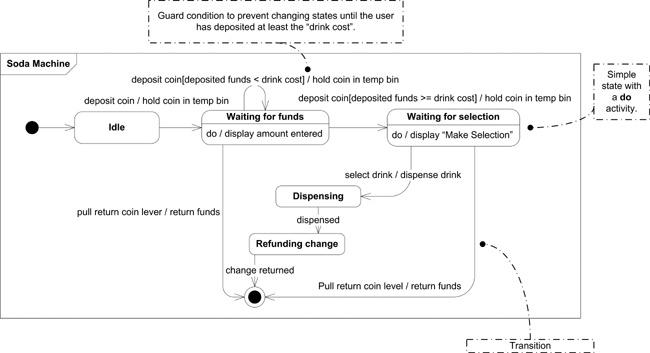
sequence diagrams



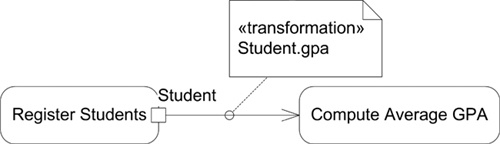
combined fragment



state diagrams



active diagram



partitions

