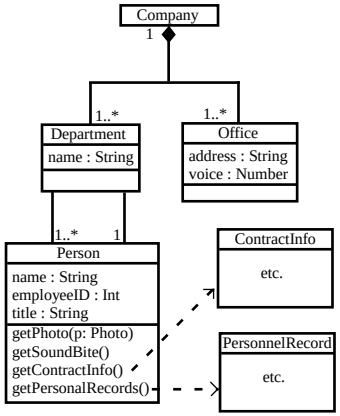
UNIFIED MODELING LANGUAGE

One good picture of your software is worth a 1000 word document.



Unified Modeling Language (UML)

Modeling Language - A system that allows a software designer to graphically layout and model a software application.

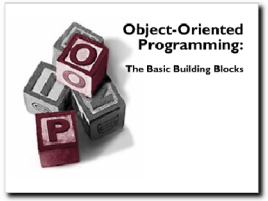
The Object Management Group* (OMG) specification states:

> ***"The Unified Modeling Language (UML) is a graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system. The UML offers a standard way to write a system's blueprints, including conceptual things such as business processes and system functions as well as concrete things such as programming language statements, database schemas, and reusable software components."***

in other words...

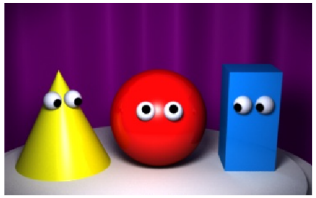
OMG is the original organization responsible for bringing about the creation of UML. OMG is a not-for-profit computer industry specifications consortium whose members define and maintain the UML specification which is published in the series of documents at http://www.uml.org/ You may download any of the documents for free.

What is UML used for?



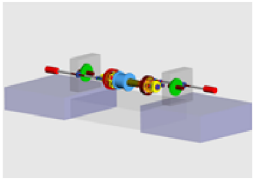
Specification

Visualization



Architecture design

Construction



Simulation and Testing

Documentation

Unified Modeling Language

Primary goals in the design of the UML

What the UML Gurus say about UML.

What it really means.

It provides users with a ready-to-use, expressive visual modeling language so they can develop and exchange meaningful models.

Draw me a picture of your software and I'll understand it.

It is independent of any particular programming languages or development processes.

Show me a UML diagram of an application and I can implement it in any number of other languages.

It encourages the growth of OO thinking, planning, and design.

Creating, studying, and using UML diagrams helps you to learn to think in Object Oriented software design concepts.

Unified Modeling Language

Primary goals in the design of the UML

What the UML Gurus say about UML.

What it really means.

It supports the development of higher-level programming concepts such as collaborations, frameworks, patterns and components.

Even the more complicated parts of Object Oriented programming become simple when you see the picture.

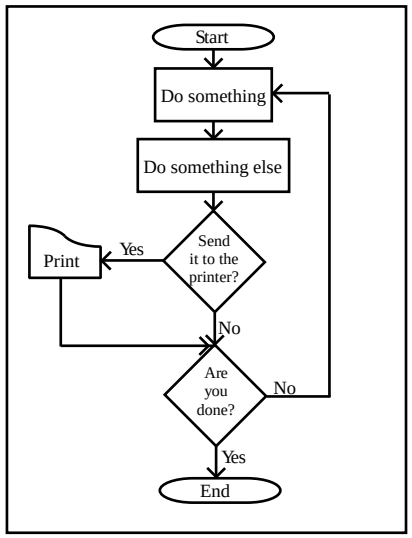
It helps to integrate best programming practices into the software development process.

UML encourages you to follow good Object Oriented Software Engineering processes.

Early Modeling Languages

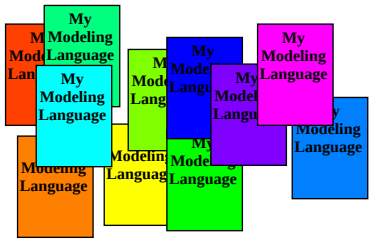
Pre-1990s

1960s-1970s Flow Charts



Works great to diagram FOC, but "What's an object?"

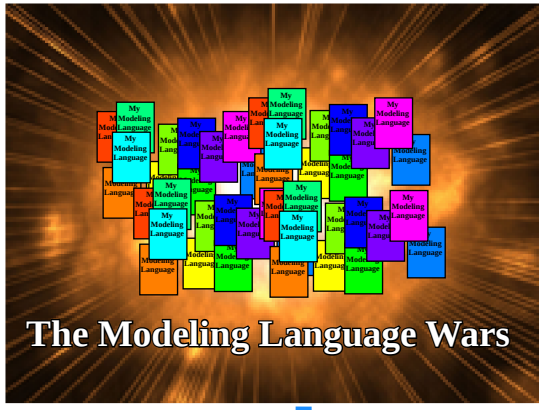
Mid-1970s-late 1980s



OOP begins and the Modeling Languages proliferate

A long time ago, in a galaxy far, far away...

Mid-1990s



To be continued...

Early Modeling Languages

UML Beginnings

1994

Let's get together...

Rational. software Corp.
Grady Booch
The Booch Method

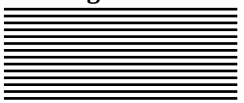
General Electric
James Rumbaugh
Object Modeling Technique

Booch and Rumbaugh begin sharing ideas

1995

Jacobson and Richard Soley (CTO of OMG) invited everyone to...

First Modeling Language Conference Agenda



Hosted by: OMG

Objectory, Inc.
Ivar Jacobson
Object Oriented Software Engineering (OOSE)

And the result was...

1996

Booch, Jacobson and Rumbaugh combine forces to produce...

Lookie here! What do you think?

UML

Versions 0.9 and 0.91

June-October 1996

To be continued...

Rational. software Corp.
Acquired by IBM in 2003
Grady Booch
The Booch Method

Objectory, Inc.
Acquired by Rational in 2003
Ivar Jacobson
Object Oriented Software Engineering (OOSE)

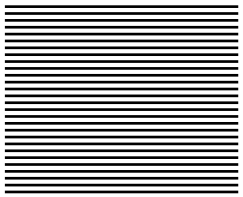
General Electric
James Rumbaugh
Object Modeling Technique

Early Modeling Languages

1996

The same year!

Request For Proposal
Object Management Group



OMG's RFP

Which led to...

Rational establishes the...

UML
Partners Consortium

Digital Equipment Corp.	MCI Systemhouse
HP	Microsoft
i-Logix	Oracle
IntelliCorp	Rational Software
IBM	TI
ICON Computing	Unisys

Let's everybody jump on the band wagon...

UNIFIED MODELING LANGUAGE

Version 1.0

1997

...me too, me too!!!

ObjecTime
Platinum Technology
Ptech
Taskon
Reich Technologies
Softeam

All the late comers finally come on board.

Fall 1997

UNIFIED MODELING LANGUAGE

Version 1.1

New and improved.
Easier to use.

2003

UNIFIED MODELING LANGUAGE

Version 2.0

UML 2.4 was in beta release as of March 2011
The latest version is 2.4.1 as of January 2015

Unified Modeling Language

Why Use It?

To be successful a software company must produce quality software, that meets requirements and is released on time and within budget.

UML helps you do this...

which in turn helps you keep your job!

Sure makes me a believer...

Unified Modeling Language
Three Parts of UML

I Basic Building Blocks

Graphical representations of:

1. **The parts of a system.**
2. **The interconnections between the parts.**
3. **Ways of combining these into a meaningful "picture" of the system.**

This is the one we will focus on. See next slide...

II Rules controlling how the blocks are put together

Prescriptive Rules: State what is and isn't legal in the language. OMG decides on these.

Descriptive Rules: State how the language is used in practice.

You can't break prescriptive rules and still be UML.
You can bend or even break descriptive rules...maybe.

III Common mechanisms applied throughout the language

Standards to apply no matter what type of system UML is used to describe.

Unified Modeling Language

Three Basic Building Blocks

I **Basic Building Blocks**

A **Things**

The "objects" that are parts of a system.

Class Component Module Unit Part Machine

B **Relationships**

How are the objects interconnected.

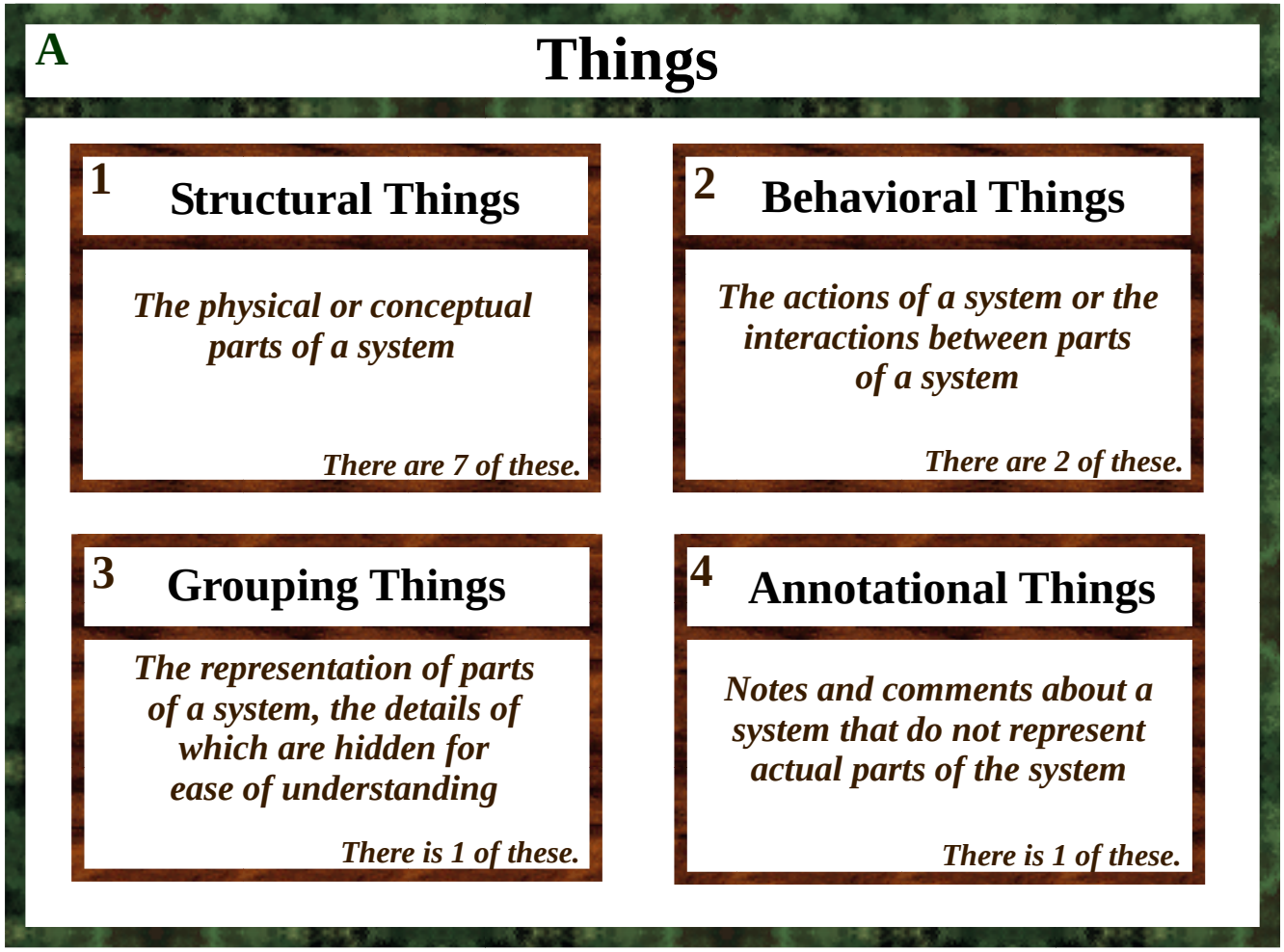
C **Diagrams**

How to combine the things and relationships into meaningful pictures of the system.

We will look at each of these three items...

Unified Modeling Language

Four Types of Things

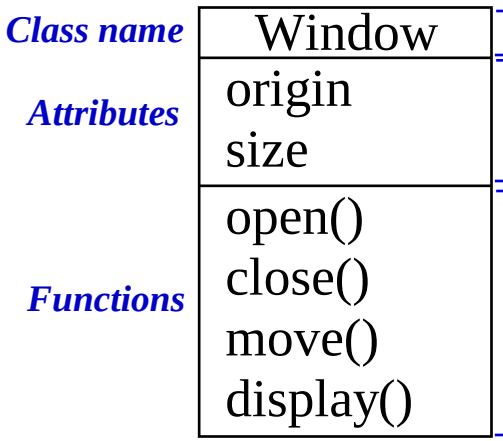


Unified Modeling Language

1

Structural Things

Class



The name of the .h and .cpp files.

Format: *Only this is required.*

visibility | **name** | **:** | **type** | **[multiplicity]** | **= default** | **{property-string}**

visibility: public (+) or private (-)
name: Name of the member variable defined in the .h file.
type: Datatype, i.e. int, double, *MyClass, etc.
[multiplicity]: How many, for arrays the length, for other references [1], [0..*], [1..4], [*], etc.
= default: Default value to be set in the constructor.
{property-string}: Indicates additional properties for the attribute, e.g. "readOnly" or "const" in C++

Example: - m_sName : String [1] = "Untitled" {readOnly}

Defined in .h as... Private:
const string m_sName = "Untitled";

Format: *Only this is required.*

visibility | **name** | **(parameter list)** | **:** | **return-type** | **{property-string}**

visibility: public (+) or private (-)
name: Name of the function prototyped in the .h file.
(parameter-list): Function arguments in the format *direction name:type-default value* where direction is *in, out,* or *inout* with default of *in*.
return-type: Datatype of return value or void

Example: + setID (in ID:integer) : void

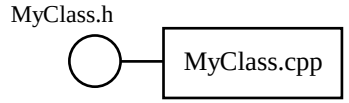
Defined in .h as... Public:
void setID(int ID);

1 of 7

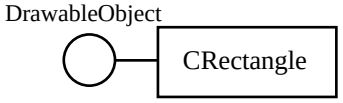
Unified Modeling Language

1 Structural Things

Interface



A header file defines how you interface with a class.

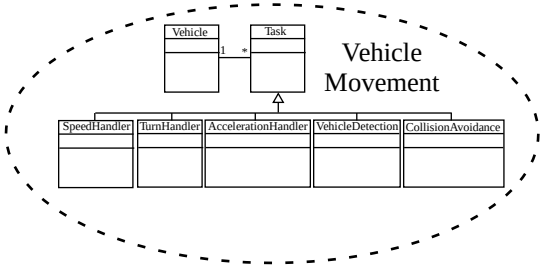


A parent class defines some functions which a sub-class must implement

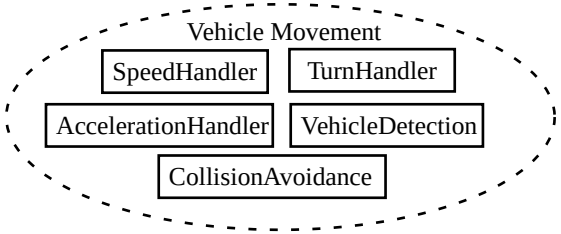
2 of 7

Collaboration

Simple representation of a collaboration.



Collaboration enclosing a class diagram of the classes involved.



Collaboration listing classes involved.

3 of 7

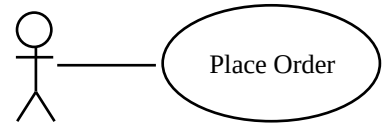
Unified Modeling Language

1

Structural Things

Use Case

Useful when capturing the functional requirements of a system.



Who is interacting.

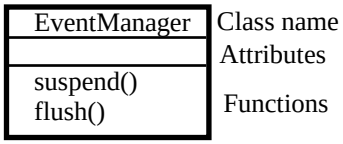
What actions take place.

Shows how a user interacts with the system

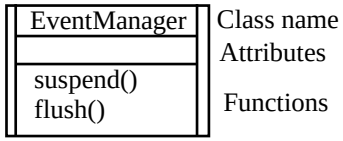
4 of 7

Active Class

Diagrams events occuring in different threads.



UML 1 notation



UML 2 notation

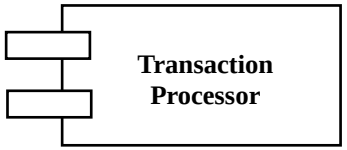
5 of 7

Unified Modeling Language

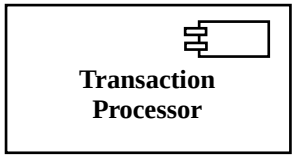
1 Structural Things

6. Component

Represents a part of a system.



UML 1 notation

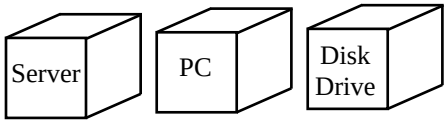


UML 2 notation

6 of 7

7. Node

A part of a system on which software is loaded.



Nodes usually represent hardware components of a system

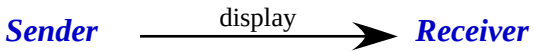
7 of 7

Unified Modeling Language

2 Behavioral Things

1. Interaction

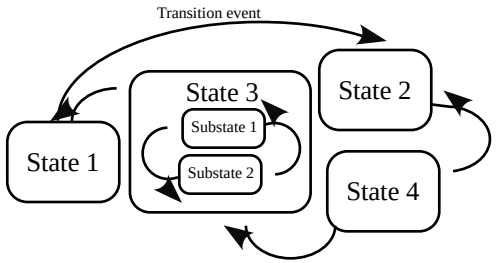
Represents a message sent from one class to another.



1 of 2

2. State Machine

Illustrates how events change an object over its lifespan.



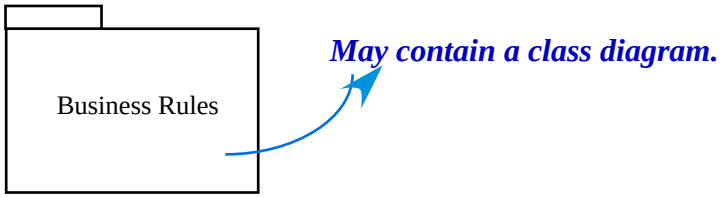
2 of 2

Unified Modeling Language

3 Grouping Things

1. Package

Indicates a higher level or organization.



1 of 1

4 Annotation Things

1. Note

Adds a comment to a diagram.

1 of 1

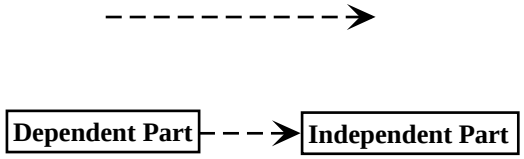
Unified Modeling Language

B

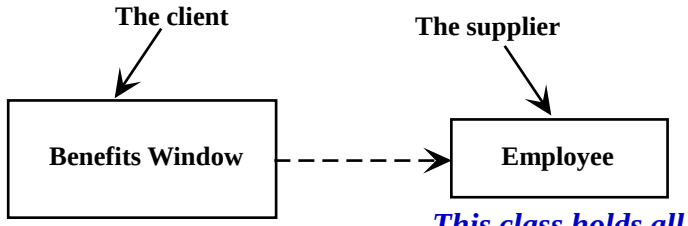
Relationships

Dependency

Represents a relationship in which a change in one class requires a change in another.



Example:



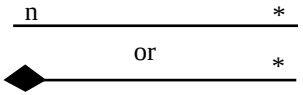
This class displays information on an employee's benefits.

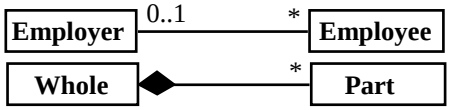
This class holds all information on a single employee.

1 of 4

Association/Aggregation

Represents a relationship in which one class owns one or more instances of another class.





Multiplicity

1 or ◆ = Exactly one instance
0..1 = Zero or one instance
***** = Any number of instances

2 of 4

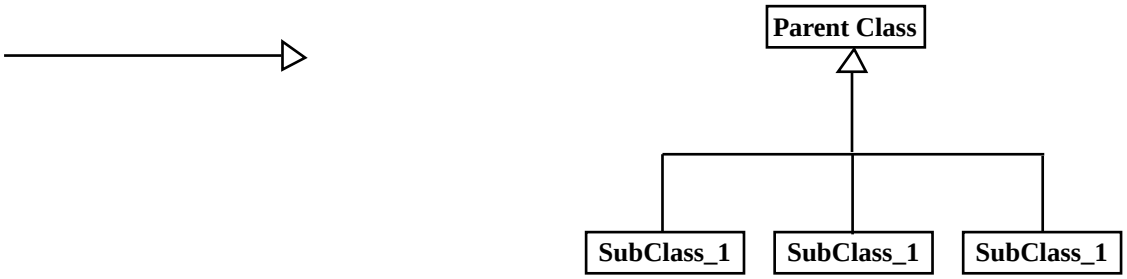
Unified Modeling Language

B

Relationships

Generalization

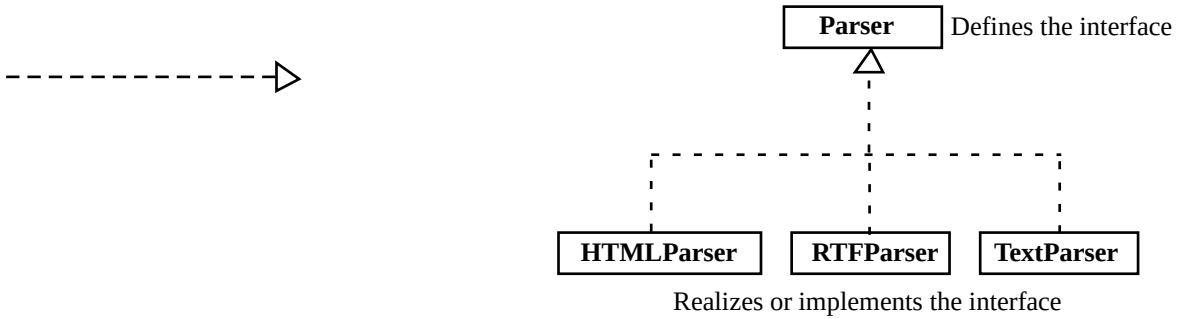
Represents the relationship between a parent class and its sub-classes.



3 of 4

Realization

Represents a relationship in which one class defines functionality that another class must implement.



4 of 4

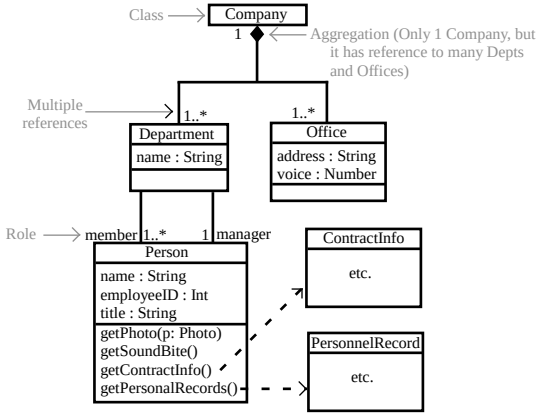
Unified Modeling Language

C

Diagrams

Class Diagram

Shows all classes in the application and their relationships



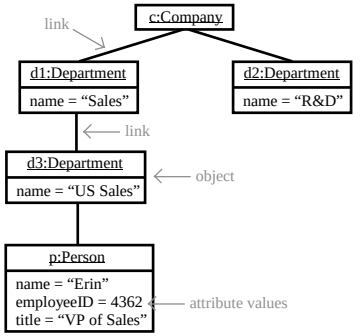
All class relationships at a minimum must show dependency and multiplicity of both ends of the line or a parent and sub-class relationship which ever is appropriate.

1 of 9

Object Diagram

Shows all instances of classes at a particular point in time.

a.k.a. Instance Diagram



This is a sample Object Diagram of the application shown in the Class Diagram.

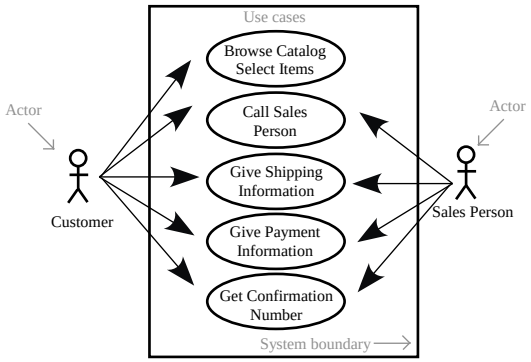
2 of 9

Unified Modeling Language

C

Diagrams

Use Case Diagram *Shows all Use Cases for an application.*

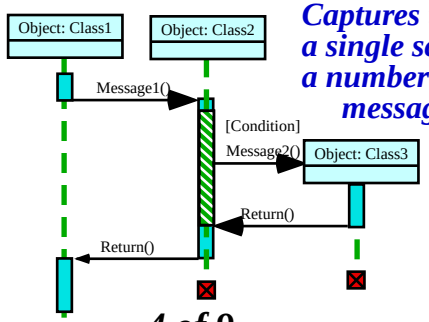


Note: Use Case Diagrams only show the relationship between all the different Use Cases of an application. They do not give any of the detail which is included in the textual description of a Use Case.

3 of 9

Interaction Diagrams *Shows how groups of objects work together to perform some action.*

Sequence Diagram

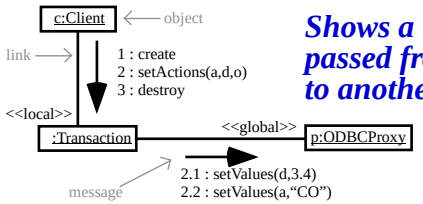


Captures the behavior of a single scenario and shows a number of objects and the messages that are passed between them.

4 of 9

Communication Diagram

Called Collaboration Diagram in UML 1.0



Shows a listing of messages passed from one class instance to another.

5 of 9

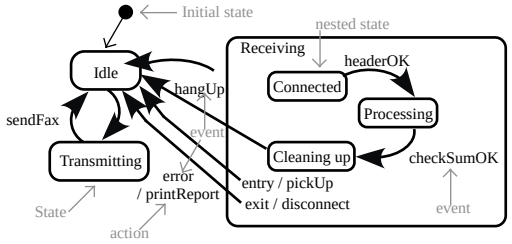
Unified Modeling Language

C

Diagrams

State Machine Diagram

Shows all states of a single class throughout the execution of an application.



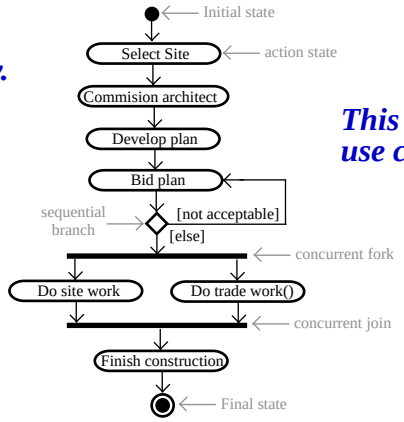
This diagram shows the different states of an object and the transition events that cause it to move from one state to another across a single use case.

6 of 9

Activity Diagram

Shows procedural logic and workflow.

Similar to the old flow charts.



This diagram shows behavior across many use cases or threads.

7 of 9

Unified Modeling Language

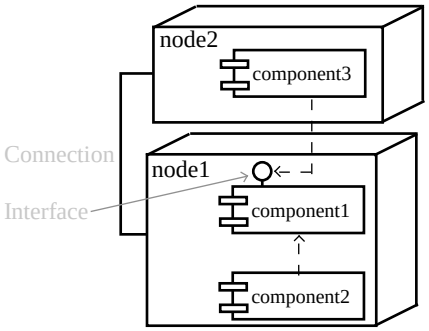
C

Diagrams

Physical Diagrams

Component Diagram

Shows how all the parts of a system fit together.

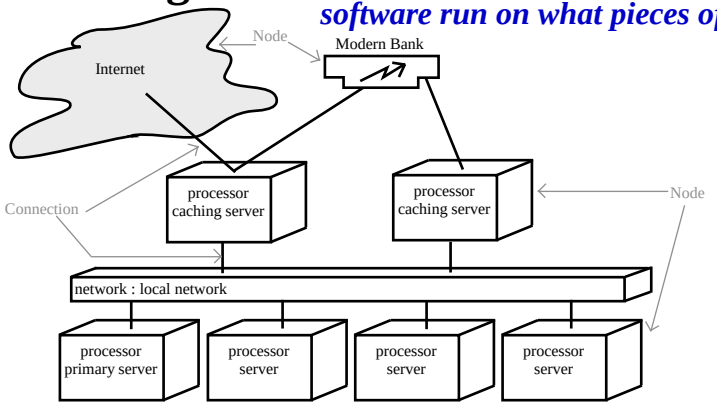


Note: This example uses the UML 1 notation for a component.

8 of 9

Deployment Diagram

Shows a system's physical layout, and which pieces of software run on what pieces of hardware.



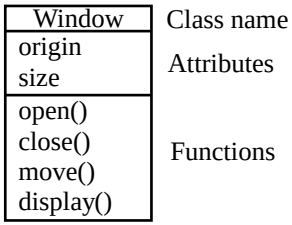
9 of 9

Most Used Diagrams

At least, in this class.

Structural Things

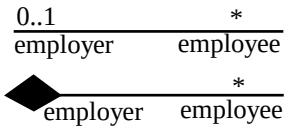
Class



Relationships

Dependency

Association

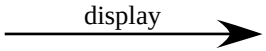


Generalization

Realization

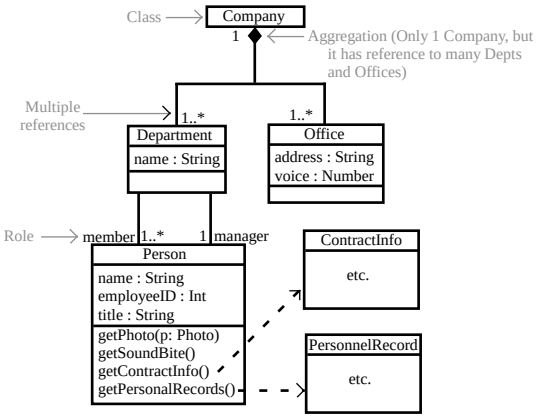
Behavioral Things

Interaction

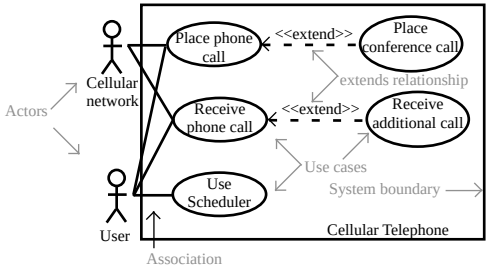


Diagrams

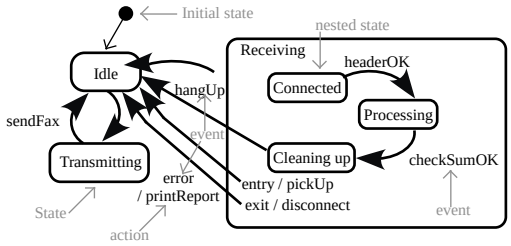
Class Diagram



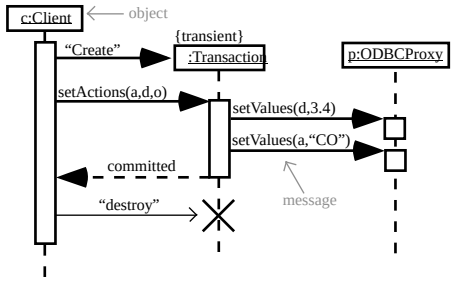
Use Case Diagram



Statechart Diagram



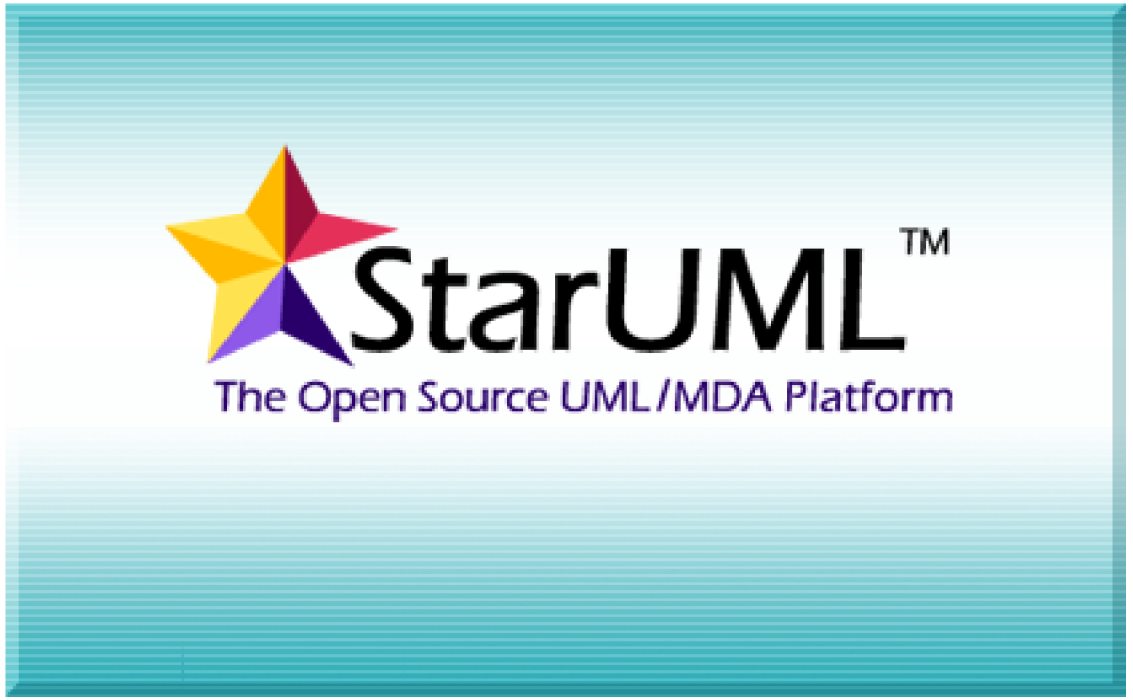
Sequence Diagram



StarUML

A free UML modeling tool.

You should use this to draw the UML diagrams for programming assignments in this course.



Download free from: http://staruml.sourceforge.net/en

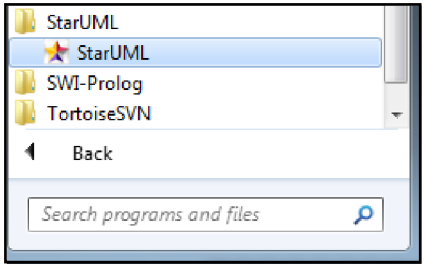
Or, any other of a number of sites. Do a web-search on StarUML

*Or, you can get the installer from Dr. Coleman.**

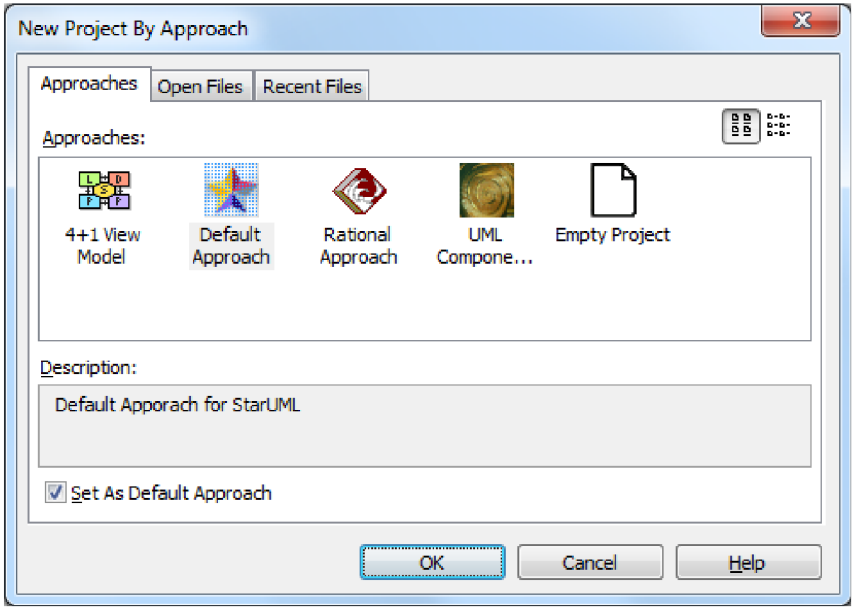
**Found in K:/LABS/cs307/StarUMLInstaller*

StarUML
A Quick Start Guide

1. Start up StarUML



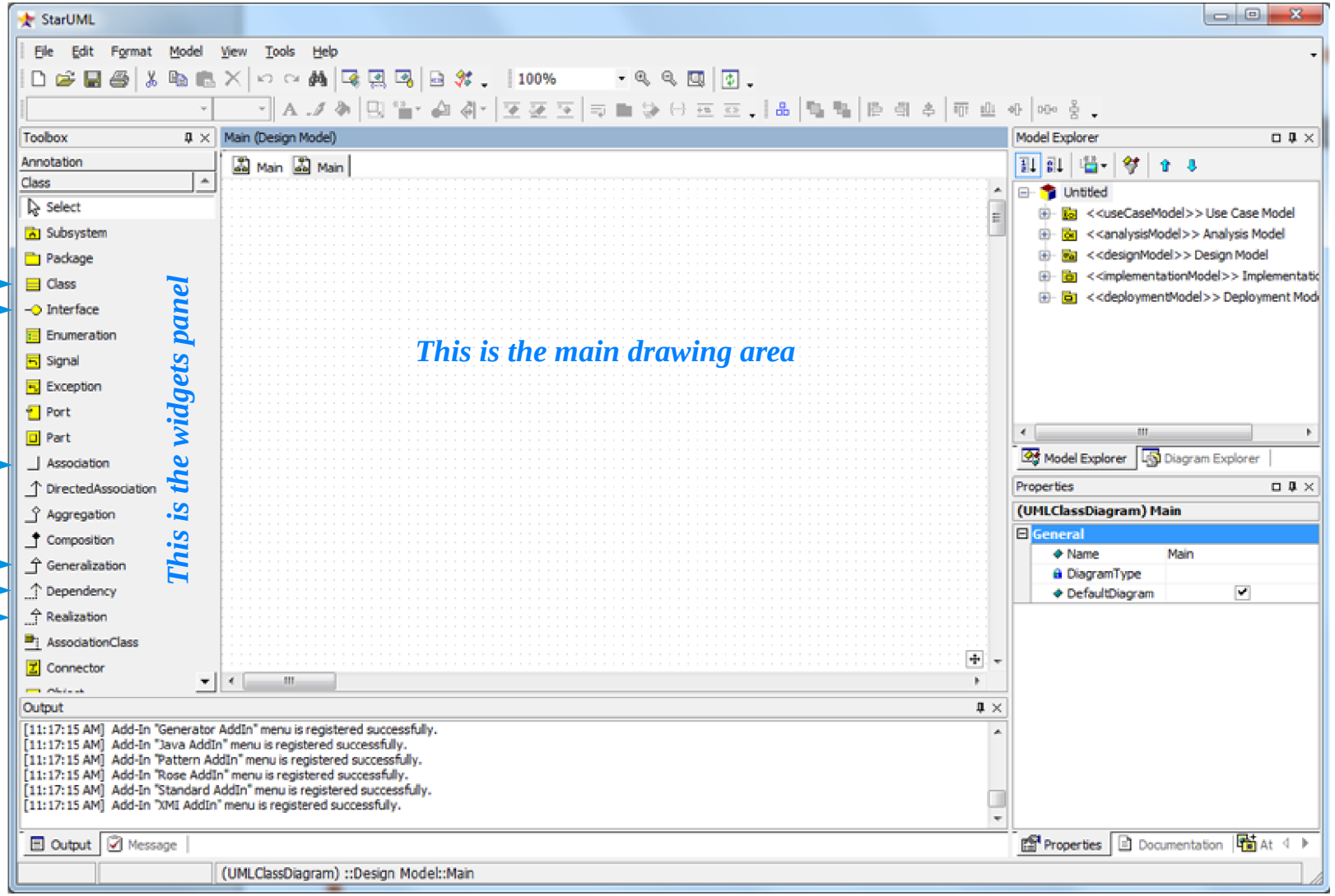
2. In this dialog box select Default Approach then click OK.



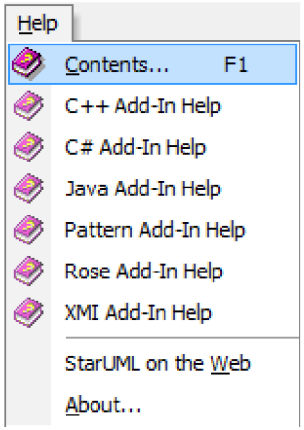
That's it. You take it from here. Hey, I did say this was a Quick Start Guide...OK, just kidding.

StarUML
A Quick Start Guide

There are bunches of widgets, but these are probably all you'll need for the programming assignment UML diagrams.

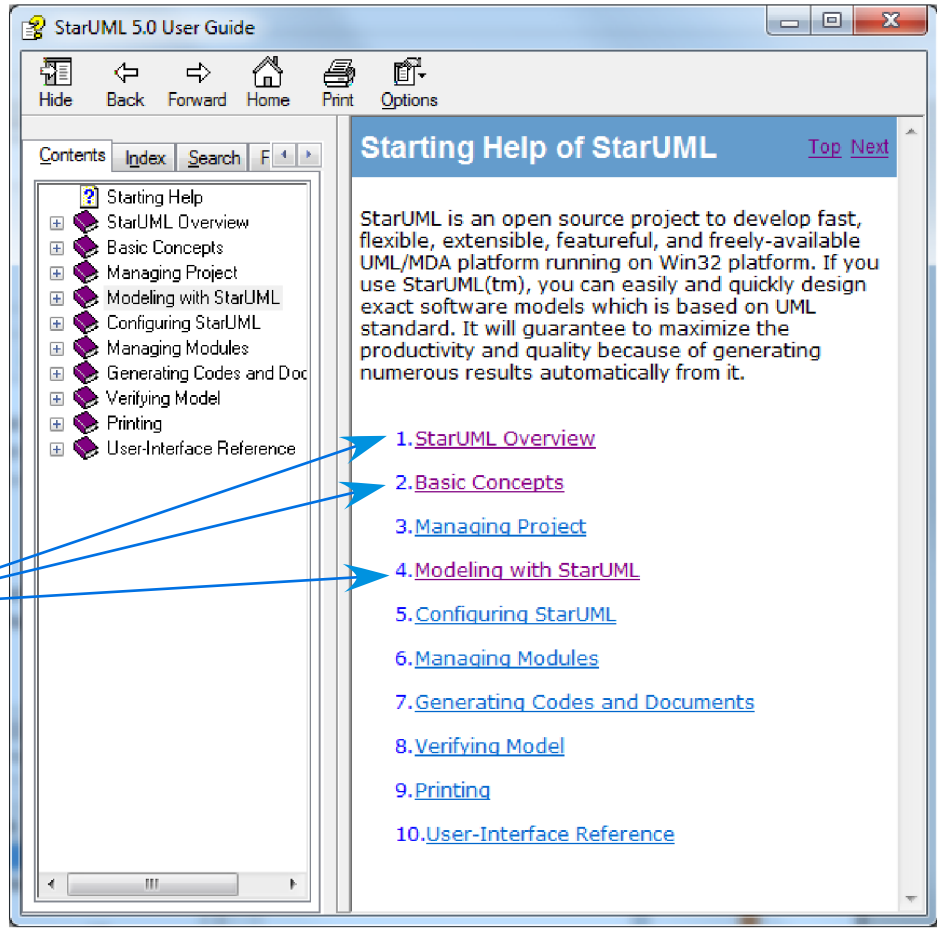


StarUML
A Quick Start Guide



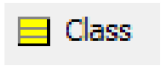
1. Select Contents from Help

2. Read about the basics:

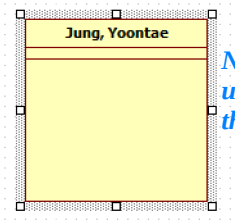


StarUML
The Most Used Stuff

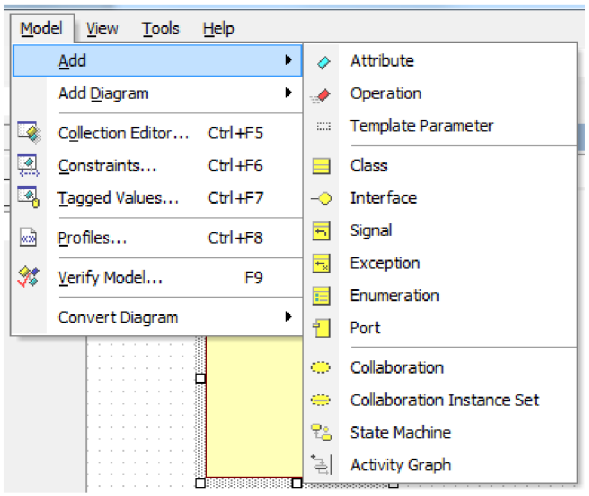
Add a Class to a diagram

1. *Select class from the widget panel* 

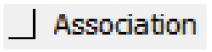
2. *Click and drag a rectangle in the drawing area*
3. *Double click the class name to change it.*
4. *Use the Model->Add submenu to add member variables (Attribute) or member functions (Operation) to the class widget.*



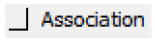
Note that the developers use their own names as the default class names.



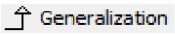
Create a relationship

1. *Select* 

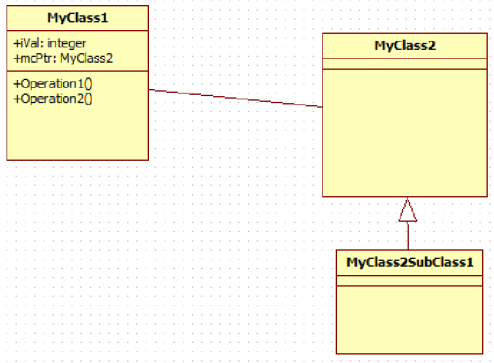
, or other appropriate relationship from the widget panel.
2. *Click and drag from one class to another.*

Example 1:
- *a. Click* 

- *b. Click MyClass1.*
- *c. Drag to MyClass2. (Note an attribute of type MyClass2 has been added to MyClass1.)*

Example 1:
- *a. Click* 

- *b. Click MyClass2SubClass1.*
- *c. Drag to MyClass2. (This makes MyClass2SubClass1 a sub-class of MyClass2.)*

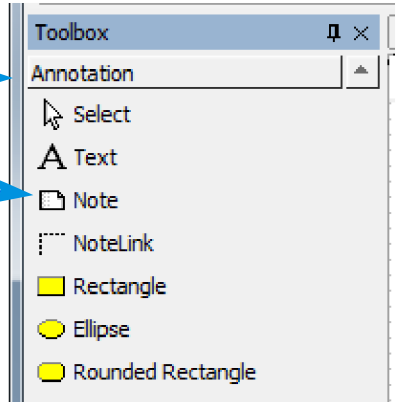


StarUML
Add a Note to a Diagram

Add a Note to a diagram

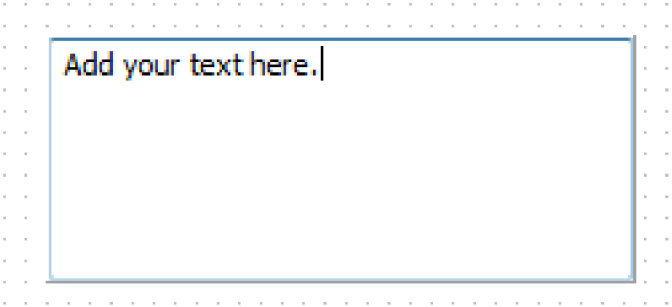
1. Select the Annotation tab in the Toolbox

2. Select the Note widget

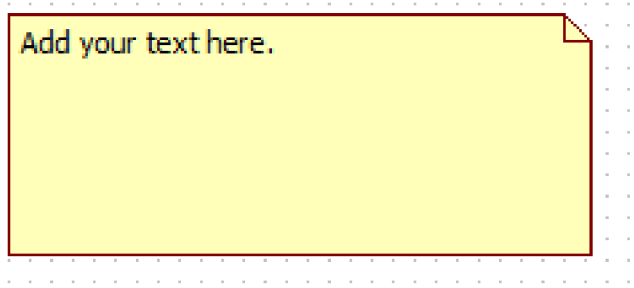


3. Click and drag a rectangle in the drawing area

4. On mouse up this will become an editable text field. Add the desired text. You can edit it again later if desired.



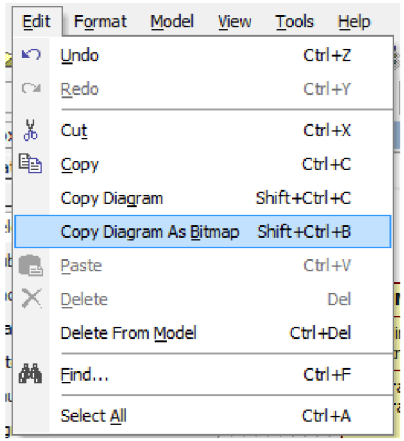
5. When done typing, click anywhere outside the text rectangle and the Note widget appears.



StarUML

Add Your UML Diagram to a Word Document

1. From the Edit menu in StarUML select "Copy Diagram As Bitmap"



2. Open MS Office Word

3. From the Edit menu in MS Word select Paste