Enterprise Architect

Current Release
Version 14.0, Build 1421
25-May-2018

Enterprise Architect Trial

Resources
MDG Technologies
White paper Repository
Demonstrations
UML Database modeling
Mapping Use Cases
RTF Documentation
Image Library
MDA Resources
XML Schema Generation
Extra UML Resources
Enterprise Architect User Guide

Webinars
Webinar Registration
Model-Driven Use Case Analysis
Preparing Visual Models
Simulation and Prototyping
Tools for Traceability

Team Modeling Resources
DBMS Repository Scripts
Cloud Services
User Security Key

Developers
Automation Interface
Patterns
UML Profiles

MDA Style Transforms
Built-in Transformations
Writing Transformations

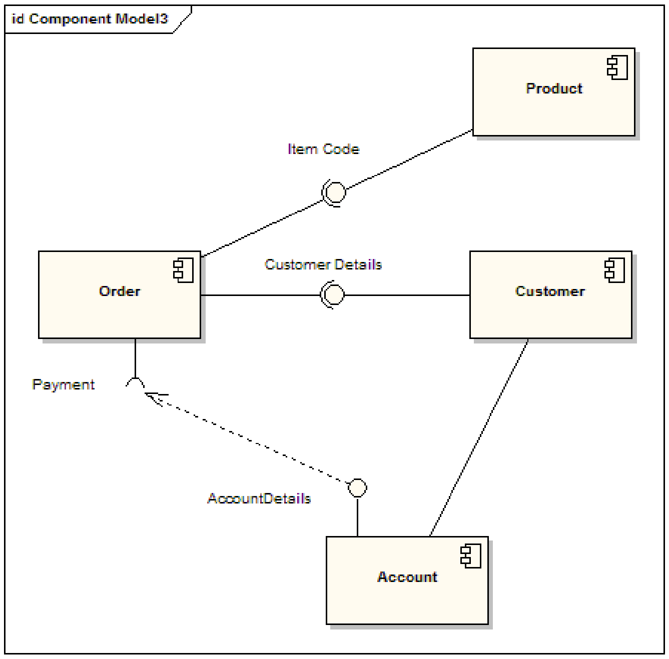
UML Tutorials
UML Tutorial
UML Tutorial - Part 2
UML 2.0 Tutorial
EA Demonstrations
UML Models
Business Process Model
Custom Model
Dynamic Model
Logical Model
Physical Models
Use Case Model

Enterprise Architect Tutorials
BABOK Guide v3
NIEM 3 Guide
Strategic Models
Diagram Filters
BPEL Guide
Resource Management
Testing Support
Traceability
Use Case Metrics

UML 2 Component Diagram

Component Diagrams

Component diagrams illustrate the pieces of software, embedded controllers, etc., that will make up a system. A component diagram has a higher level of abstraction than a Class Diagram - usually a component is implemented by one or more classes (or objects) at runtime. They are building blocks so a component can eventually encompass a large portion of a system.

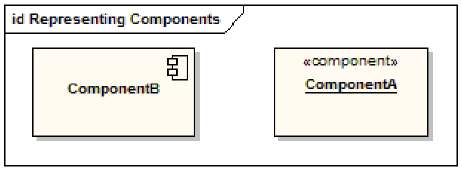


The diagram above demonstrates some components and their inter-relationships. Assembly connectors "link" the provided interfaces supplied by "Product" and "Customer" to the required interfaces specified by "Order". A dependency relationship maps a customer's associated account details to the required interface; "Payment", indicated by "Order".

Components are similar in practice to package diagrams, as they define boundaries and are used to group elements into logical structures. The difference between package diagrams and component diagrams is that Component Diagrams offer a more semantically rich grouping mechanism. With component diagrams all of the model elements are private, whereas package diagrams only display public items.

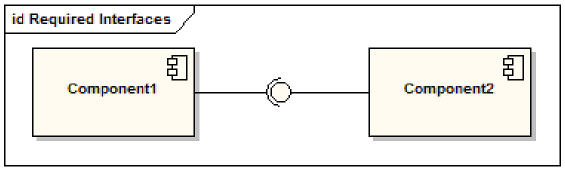
Representing Components

Components are represented as a rectangular classifier with the keyword «component»; optionally the component may be displayed as a rectangle with a component icon in the right-hand upper corner.



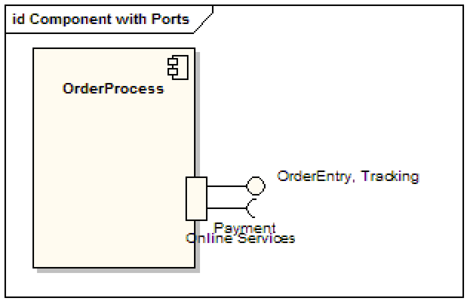
Assembly Connector

The assembly connector bridges a component’s required interface (Component1) with the provided interface of another component (Component2); this allows one component to provide the services that another component requires.



Components with Ports

Using Ports with component diagrams allows for a service or behavior to be specified to its environment as well as a service or behavior that a component requires. Ports may specify inputs and outputs as they can operate bi-directionally. The following diagram details a component with a port for online services along with two provided interfaces order entry and tracking as well as a required interface payment.



Products
Enterprise Architect
Eclipse Integration
Visual Studio Integration
UPDM Technology
SysML Technology
DDS Technology
DOORS Link
Additional Extensions

UML at a Glance
UML
UML Tools
PHP UML Modeling
Business Process Modeling
Model Driven Architecture
Requirements Management
Software Development

Solutions
Corporate
Government
Small/Medium Enterprise
IT Professionals
Trainers
Academic

Resources
UML 2.0 Tutorial
MDG Technologies
Corporate Resources
Developer Resources
Media Resources

Support
Online Manual
User Forum
Report a Bug
Feature Request
Compare Editions
System Requirements

Global Partners
Trainers
Value Added Resellers
Resellers
Sister Companies
Technical Partners
Standards Organizations