UML w Visual Studio

Michał Ciećwierz

UNIFIED MODELING LANGUAGE
(Zunifikowany język modelowania)

- Pozwala tworzyć wiele systemów (np. informatycznych)
- Pozwala obrazować, specyfikować, tworzyć i dokumentować elementy systemu
- Ułatwia wymianę informacji
- Ułatwia wykorzystywanie zalet programowania obiektowego
- Jest językiem modelowania używanym w procesie analizy i projektowania systemów komputerowych

Czym UML nie jest?

- Językiem programowania (chociaż istnieje możliwość generowania kodu z niektórych diagramów)
- Narzędziem- choć zawiera specyfikacje dla narzędzi
- Metodyką- nie definiuje procesu tworzenia oprogramowania
- Nie jest sposobem na analizę i projektowanie systemów komputerowych

Historia UML

- do 1994 wiele konkurencyjnych metod analizy i projektowania
- 1994, 1995 Autorzy trzech dominujących G. Booch (OOAD), I. Jacobson(OOSE) , J. Rumbaugh (OMT) rozpoczynają pracę w Rational Software Corporation
- 1994 rozpoczęto oficjalne prace nad UML
- 1995 ujednolicenie metod OOAD i OMT, powstanie roboczej wersji UML 0.8a
- 1996 rozszerzenie wersji roboczej o OOSE, wersja 0.9
- 1996 powstaje konsorcjum firm (HewlettPackard, IBM, Microsoft, Oracle, Rational .....)
- 1997 styczeń zatwierdzenie języka UML (wersja 1.0) przez OMG

Zakres stosowania UML

- Tworzenie systemów informacyjnych przedsiębiorstw
- Usługi bankowe i finansowe
- Telekomunikacja Transport Przemysł obronny i lotniczy
- Sprzedaż detaliczna
- Elektronika i medycyna
- Rozproszone usługi internetowe
- Nauka

Rodzaje diagramów UML

- **Diagramy struktur**
- Klas (najczęściej spotykane, ang. *class diagram*)
- Obiektów (ang. *object diagram*)
- Komponentów (ang. *component diagram*)
- Wdrożenia (ang. *deployment diagram*)
- --- UML 2.0 ---
- Struktur złożonych (ang. *composite structure diagram*)
- Pakietów (ang. *package diagram*)
- --- UML 2.2 ---
- Profili (ang. *profile diagram*, nowość wprowadzona w UML 2.2)

- **Diagramy czynności**
- Czynności (ang. *activity diagram*)
- Przypadków użycia (ang. *use case diagram*)
- Maszyny stanów (ang. *state machine diagram*) (dla UML 1.x Stanów, ang. *statechart diagram*)
- Interakcji (diagram abstrakcyjny)
  - Komunikacji (ang. *communication diagram*) (dla UML 1.x Współdziałania, ang. *collaboration diagram*)
  - Sekwencji (ang. *sequence diagram*)
  - --- UML 2.0 ---
  - Czasowe (ang. *timing diagram*)
  - Przeglądu interakcji (ang. *interaction overview diagram*)

Modelowanie w Microsoft Visual Studio

Diagram klas

- To statyczny diagram strukturalny, przedstawiający strukturę systemu w modelach obiektowych przez ilustrację struktury klas i zależności między nimi.
- Diagram klas przedstawia klasy (typy) obiektów w programie, w odróżnieniu od diagramu obiektów, który pokazuje jedynie egzemplarze (instancje) obiektów i ich zależności istniejące w konkretnym momencie.

Możliwe zależności pomiędzy obiektami

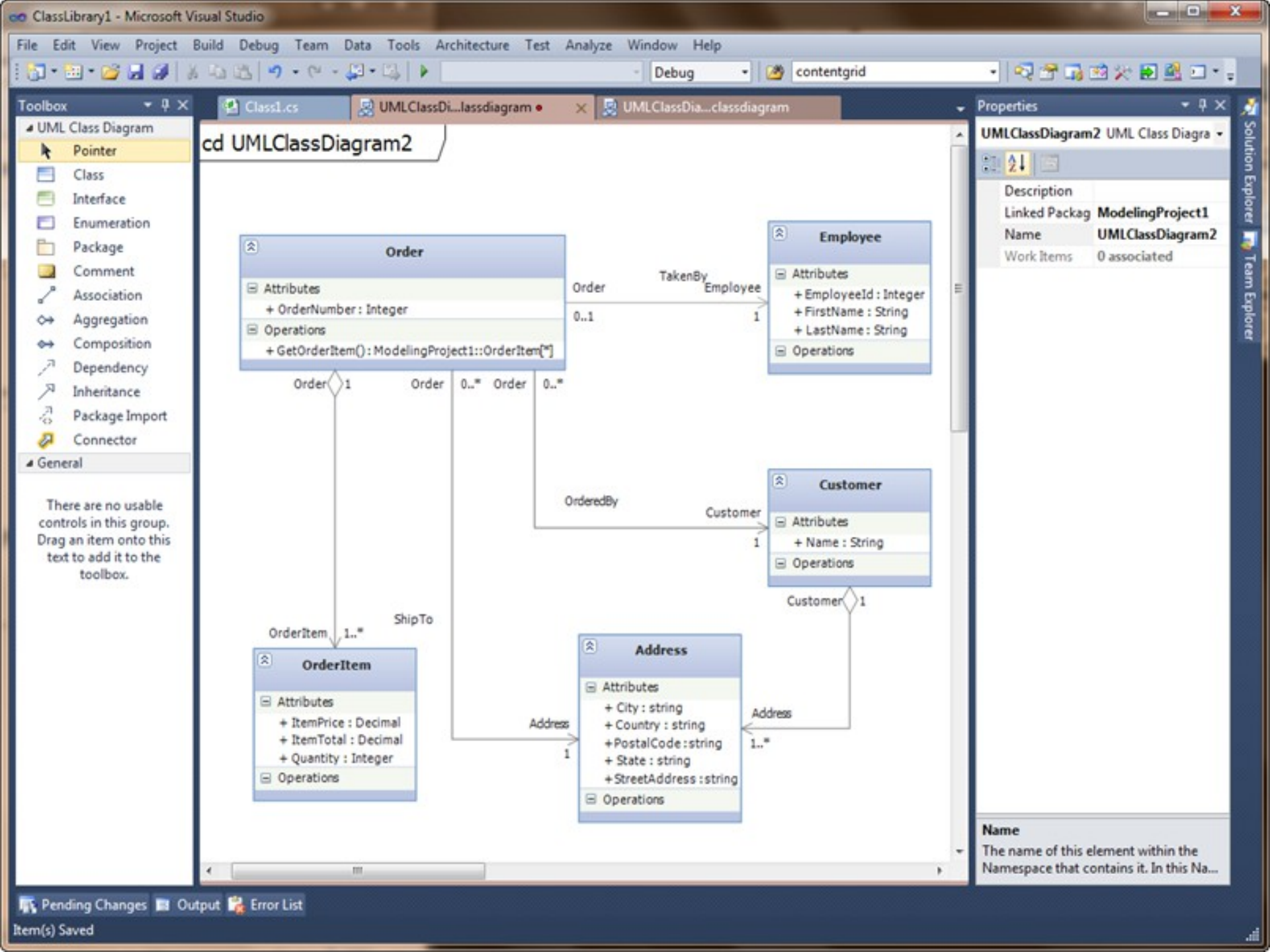
- **Zależność** – najsłabszy związek znaczeniowy między klasami, gdy jedna z nich używa innych klas.
- **Agregacja** reprezentuje związek typu całość-część. Występuje tutaj relacja posiadania — co oznacza, że elementy częściowe mogą należeć do większej całości, jednak również mogą istnieć bez niej. Na diagramie agregację oznacza się za pomocą linii zakończonej pustym rombem.
- **Asocjacja** wskazuje na trwałe powiązanie pomiędzy obiektami danych klas (np. firma zatrudnia pracowników). Na diagramie asocjację oznacza się za pomocą linii zakończonej strzałką (skierowaną od klasy źródłowej do docelowej). Nazwę cechy wraz z krotnością umieszcza się w punkcie docelowym asocjacji.
- **Generalizacja** – związek opisujący dziedziczenie po klasach. Na diagramie generalizację oznacza się za pomocą niewypełnionego trójkąta symbolizującego strzałkę (skierowaną od klasy pochodnej do klasy bazowej).
- **Kompozycja**, zwana również złożeniem, jest związkiem typu całość-część. W relacji kompozycji, części należą tylko do jednej całości, a ich okres życia jest wspólny — razem z całością niszczone są również części. Na diagramie, kompozycję oznacza się za pomocą linii zakończonej wypełnionym rombem.

Ilościowe powiązania pomiędzy obiektami

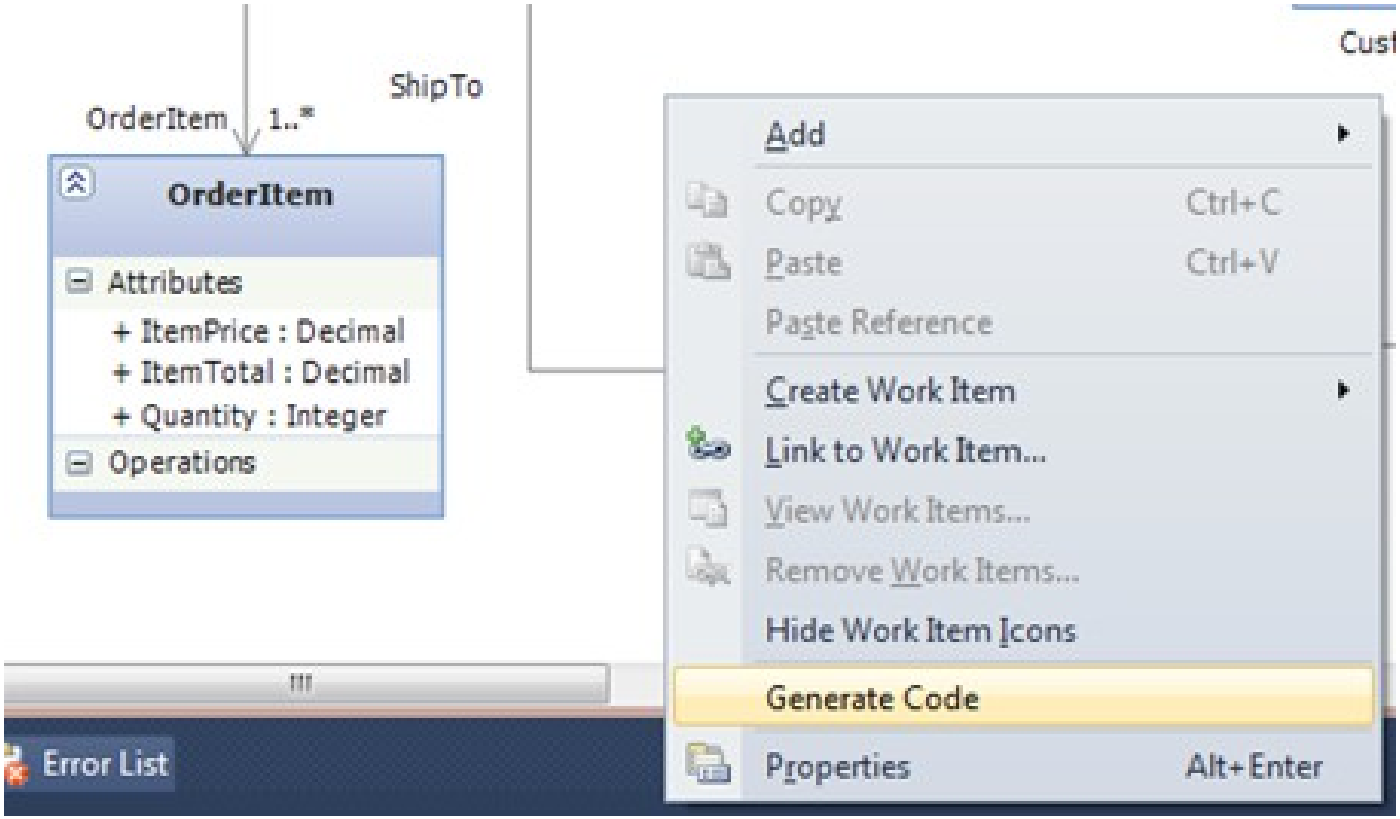
- 1
- 0..1
- *
- 1..*
- n..m

Oznaczenia widoczności atrybutów

- "+" dla *public* – publiczny, dostęp globalny
- "#" dla *protected* – chroniony, dostęp dla pochodnych klasy (wynikających z generalizacji)
- "−" dla *private* – prywatny, dostępny tylko w obrębie klasy (przy atrybucie statycznym) lub obiektu (przy atrybucie zwykłym)
- "~" dla *package* – pakiet, dostępny w obrębie danego pakietu, projektu.


ClassLibrary1 - Microsoft Visual Studio
File Edit View Project Build Debug Team Data Tools Architecture Test Analyze Window Help
Debug
contentgrid
Toolbox
UML Class Diagram
Pointer
Class
Interface
Enumeration
Package
Comment
Association
Aggregation
Composition
Dependency
Inheritance
Package Import
Connector
General
There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.
Class1.cs
UMLClassDi...lassdiagram
UMLClassDia...classdiagram
cd UMLClassDiagram2
Order
Attributes
+ OrderNumber : Integer
Operations
+ GetOrderItem() : ModelingProject1::OrderItem[*]
Employee
Attributes
+ EmployeeId : Integer
+ FirstName : String
+ LastName : String
Operations
Order
TakenBy
Employee
0..1
1
Order
1
Order
0..*
Order
0..*
OrderedBy
Customer
1
Customer
Attributes
+ Name : String
Operations
Customer
1
ShipTo
OrderItem
1..*
OrderItem
Attributes
+ ItemPrice : Decimal
+ ItemTotal : Decimal
+ Quantity : Integer
Operations
Address
Attributes
+ City : string
+ Country : string
+ PostalCode : string
+ State : string
+ StreetAddress : string
Operations
Address
1
Address
1..*
Properties
UMLClassDiagram2 UML Class Diagra
Description
Linked Packag ModelingProject1
Name UMLClassDiagram2
Work Items 0 associated
Name
The name of this element within the Namespace that contains it. In this Na...
Solution Explorer
Team Explorer
Pending Changes
Output
Error List
Item(s) Saved

Generowanie kodu z diagramu klas



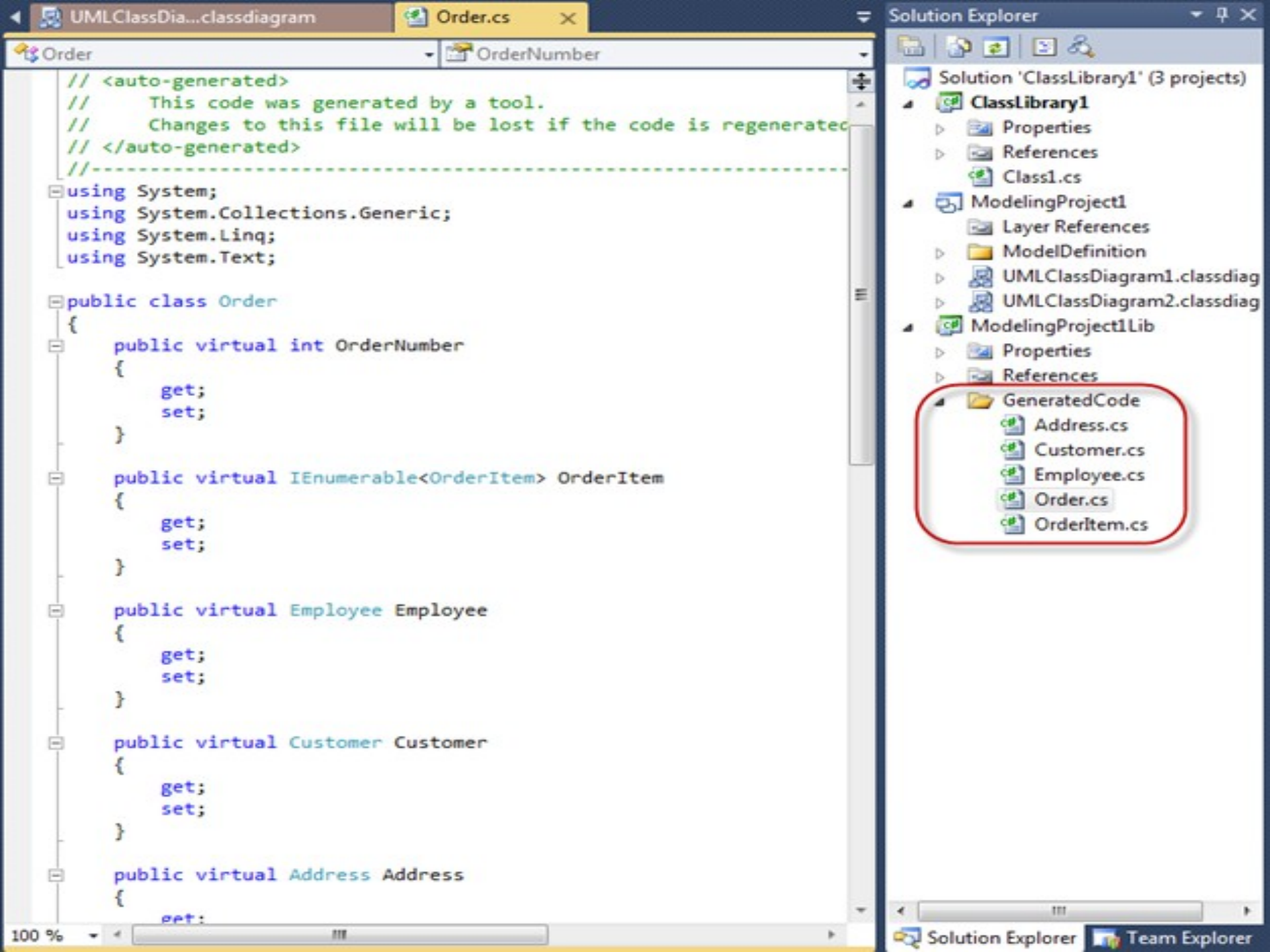

UMLClassDia...classdiagram
Order.cs
Order
OrderNumber
// <auto-generated>
// This code was generated by a tool.
// Changes to this file will be lost if the code is regenerated
// </auto-generated>
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
public class Order
{
public virtual int OrderNumber
{
get;
set;
}
public virtual IEnumerable<OrderItem> OrderItem
{
get;
set;
}
public virtual Employee Employee
{
get;
set;
}
public virtual Customer Customer
{
get;
set;
}
public virtual Address Address
{
100 %
Solution Explorer
Solution 'ClassLibrary1' (3 projects)
ClassLibrary1
Properties
References
Class1.cs
ModelingProject1
Layer References
ModelDefinition
UMLClassDiagram1.classdiag
UMLClassDiagram2.classdiag
ModelingProject1Lib
Properties
References
GeneratedCode
Address.cs
Customer.cs
Employee.cs
Order.cs
OrderItem.cs
Solution Explorer
Team Explorer

Diagram przypadków użycia

Diagram przypadków użycia (ang. *use-case diagram) służy do* modelowania aktorów (użytkowników systemu, odbiorców efektów pracy systemu, systemów zewnętrznych) i ich potrzeb w stosunku do tworzonego systemu. Przypadki użycia prezentowane na tym diagramie to sekwencje czynności, które prowadzą do spełnienia celu przypadku użycia (zaspokojenia pewnej potrzeby użytkownika).

- definiuje granice modelowanego systemu
- określa jego kontekst
- wymienia użytkowników systemu i jednostki zewnętrzne
- przedstawia funkcje dostępne dla użytkowników
- określa powiązania i zależności pomiędzy nimi

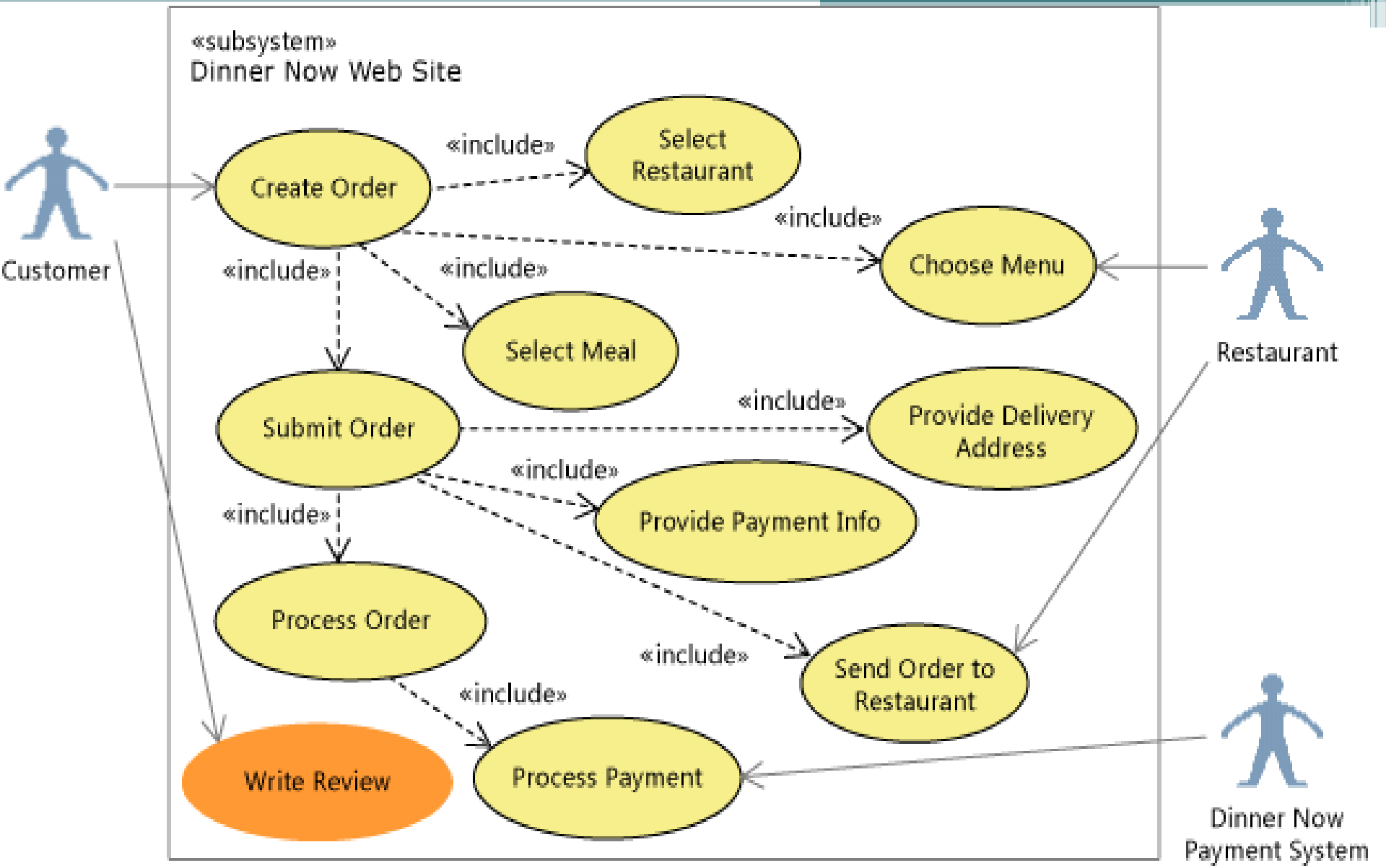

«subsystem»
Dinner Now Web Site
Customer
Create Order
«include»
Select Restaurant
«include»
Choose Menu
Restaurant
«include»
«include»
Select Meal
Submit Order
«include»
Provide Delivery Address
«include»
Provide Payment Info
«include»
Process Order
«include»
Send Order to Restaurant
«include»
Write Review
Process Payment
Dinner Now Payment System

Diagram sekwencji

- opisuje interakcje pomiędzy częściami systemu w postaci sekwencji komunikatów wymienianych między nimi

Elementy diagramu sekwencji

- Uczestnikami diagramów sekwencji są obiekty, opisane nazwą obiektu i jego klasą, które wymieniają między sobą komunikaty.
- Diagram sekwencji składa się z **pionowych linii życia (ang. lifelines)** poszczególnych obiektów uczestniczących w interakcji oraz wymienianych między nimi komunikatów (wywołań operacji). Białe prostokąty umieszczone na linii życia obiektu oznaczają, że obiekt jest zajęty wykonywaniem pewnej czynności (natomiast nie mają bezpośredniego związku z istnieniem obiektu).
- Czas jest reprezentowany w postaci pionowej osi diagramu. Linia życia obiektu to czas, w którym konkretna instancja obiektu jest w stanie przyjmować komunikaty i wysyłać je. Innymi słowy, obejmuje ona czas istnienia obiektu w systemie.
- Obiekt jest tworzony poprzez wysłanie do niego komunikatu-konstruktora, natomiast niekoniecznie jest fizycznie usuwany na końcu linii życia, a raczej przestaje być istotny. Fizycznie usunięcie obiektu można wprost oznaczyć jako znak X na linii życia

Przykładowy diagram sekwencji

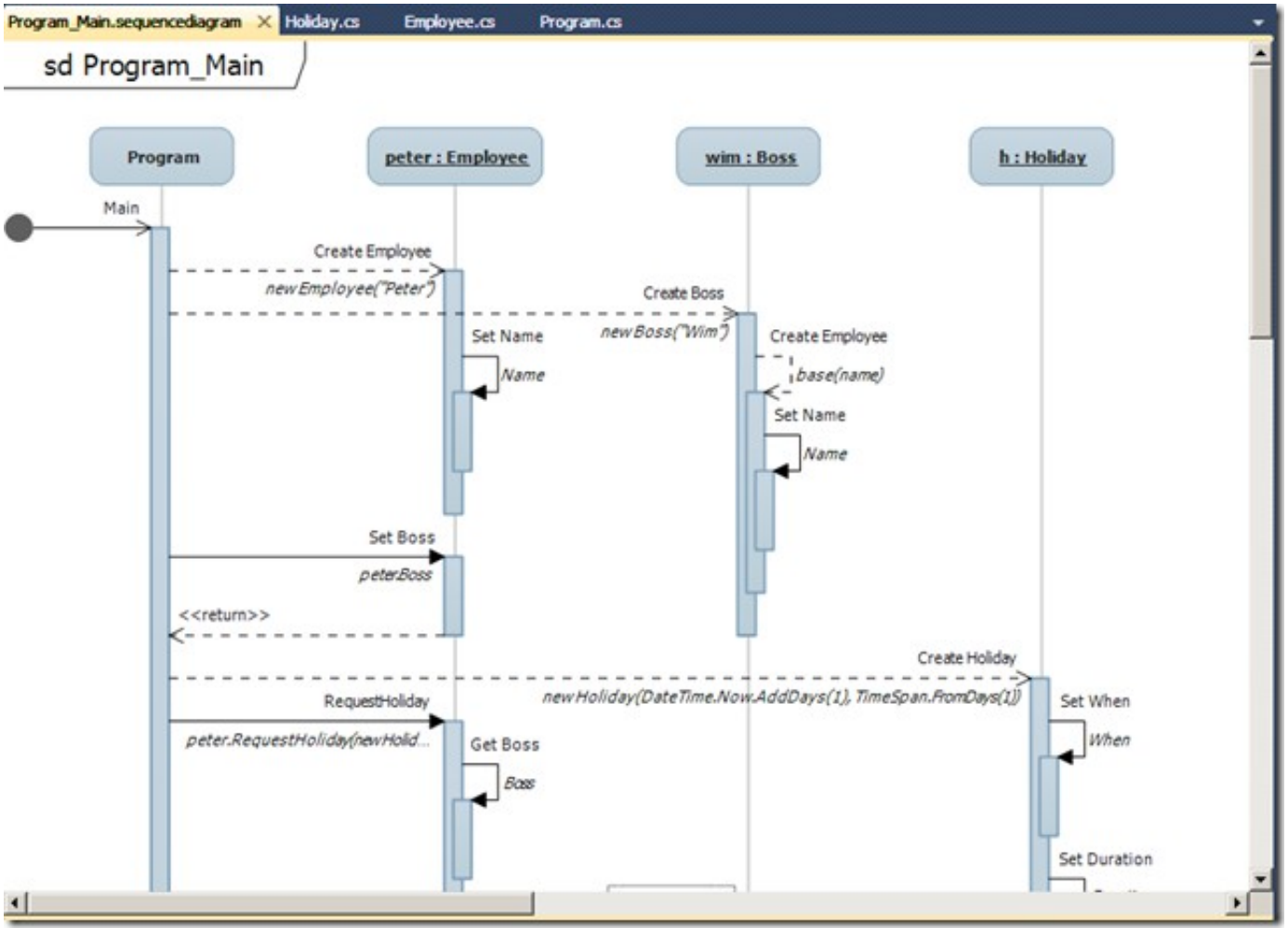


Diagram warstw

Używany do obrazowania wysokopoziomowej, logicznej architektury systemu. Organizuje fizyczne zadania, obiekty w logiczne, abstrakcyjne grupy zwane warstwami. Za ich pomocą obrazowane są zadania jakie pełnią obiekty, lub fizyczne elementy systemu. Każda warstwa może się składać z wielu podwarstw.

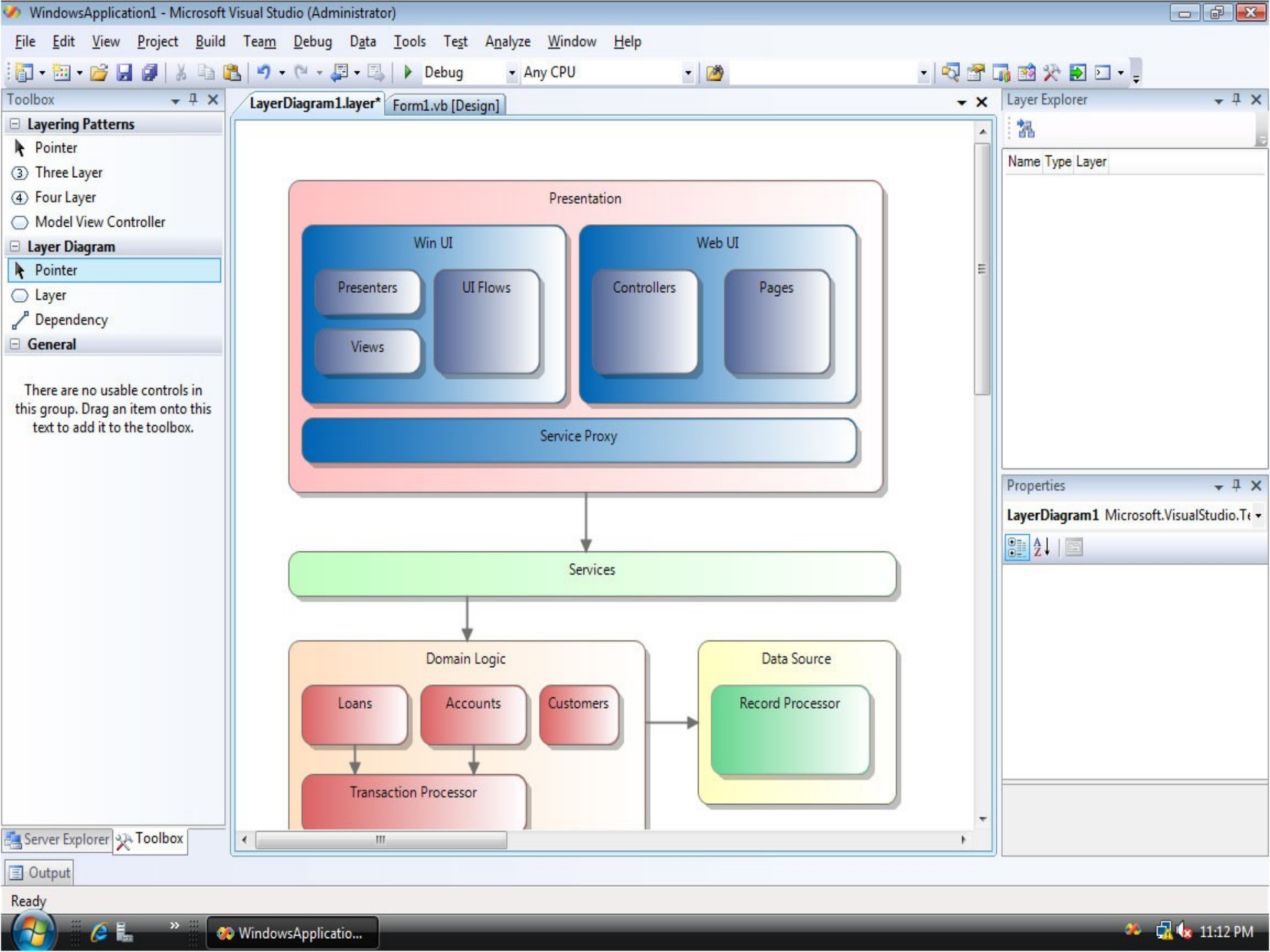
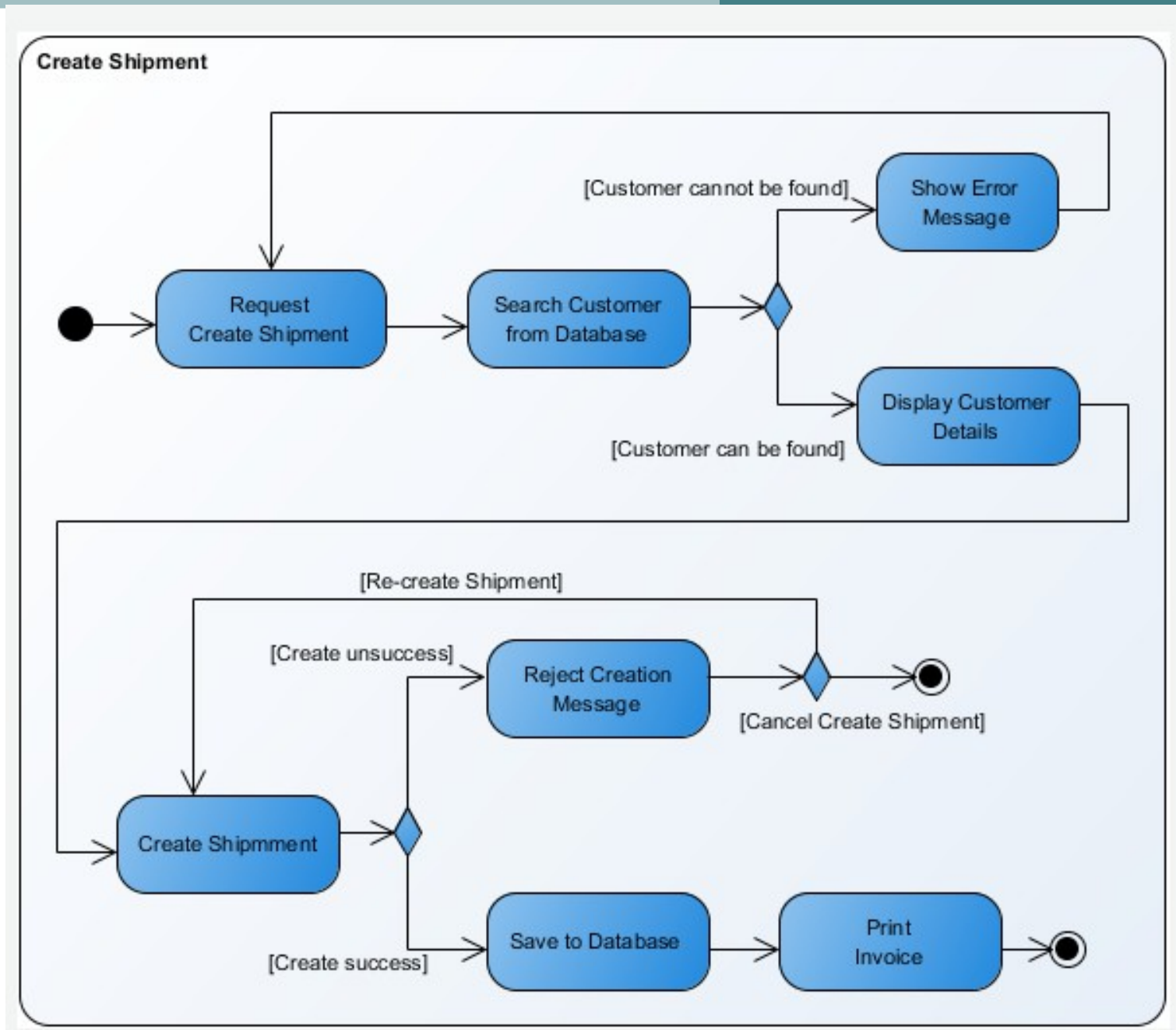

WindowsApplication1 - Microsoft Visual Studio (Administrator)
File Edit View Project Build Team Debug Data Tools Test Analyze Window Help
Debug
Any CPU
Toolbox
Layering Patterns
Pointer
Three Layer
Four Layer
Model View Controller
Layer Diagram
Pointer
Layer
Dependency
General
There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.
LayerDiagram1.layer*
Form1.vb [Design]
Presentation
Win UI
Presenters
UI Flows
Views
Web UI
Controllers
Pages
Service Proxy
Services
Domain Logic
Loans
Accounts
Customers
Transaction Processor
Data Source
Record Processor
Layer Explorer
Name Type Layer
Properties
LayerDiagram1 Microsoft.VisualStudio.T
Server Explorer
Toolbox
Output
Ready
WindowsApplicatio...
11:12 PM

Diagram aktywności

(zwany czasami diagramem czynności) w języku UML służy do modelowania czynności i zakresu odpowiedzialności elementów bądź użytkowników systemu. Jest niejako podobny do diagramu stanu, jednak w odróżnieniu od niego nie opisuje działań związanych z jednym obiektem a wieloma, pomiędzy którymi może występować komunikacja przy wykonywaniu czynności.


Create Shipment
Request Create Shipment
Search Customer from Database
[Customer cannot be found]
Show Error Message
[Customer can be found]
Display Customer Details
[Re-create Shipment]
[Create unsuccess]
Reject Creation Message
[Cancel Create Shipment]
Create Shipmment
[Create success]
Save to Database
Print Invoice