Building Behavioral UML Models in Visual Studio 2010

Richard Seroter

www.pluralsight.com

pluralsight see what you can learn

Outline

- UML in a Nutshell
- Support for UML in Visual Studio 2010
- Training Course Scenario Review
- Building a Use Case Diagram
- Building an Activity Diagram
- Building a Sequence Diagram
- Summary

UML in a Nutshell

"Unified Modeling Language (UML) is a standardized general-purpose modeling language in the field of object-oriented software engineering."

- Wikipedia (http://en.wikipedia.org/wiki/Unified_Modeling_Language)

- **Behavior Diagrams**
  - Use Case, Activity, State Machine, Sequence, Timing
- **Structural Diagrams**
  - Component, Class, Composite Structure, Package, Deployment

Support for UML in Visual Studio 2010

- Prior to Visual Studio 2010, developers could create UML diagrams in Microsoft tools such as Visio, or leverage 3rd party add-ons to Visual Studio.
- UML modeling is now included as part of Visual Studio 2010 Ultimate edition
  - Use Case Diagrams
  - Activity Diagrams
  - Sequence Diagrams
  - Component Diagrams
  - Class Diagrams
- Visual Studio 2010 Feature Pack 2 includes expanded capabilities to the existing UML modeling tools

Training Course Scenario Review

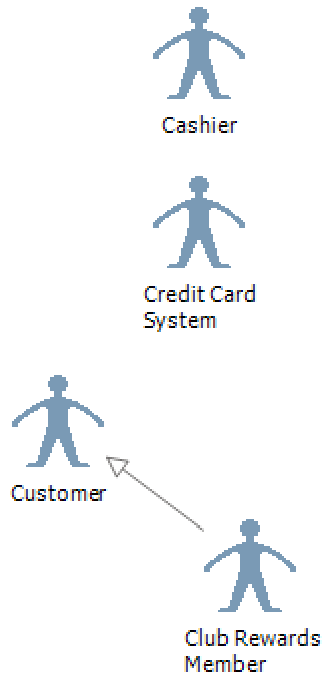
- **Watson's Pet Store**
  - They are a growing local store that provides pet supplies and services.
  - Their customers want to be able to order products and manage their service appoints online.

Building a Use Case Diagram

- **A use case diagram visually shows the relationship between users and their goals when using a system.**
- **The core components of a use case diagram are:**
  - Actors
  - Use Cases
  - System Boundaries

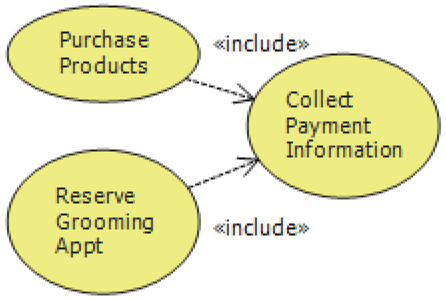
Building Use Case Diagrams - Actors

- **Actors represent users which may be human or non-human.**
  - Tied to roles, not specific users
  - Could be another organization, application or device
  - May represent "time"
- **One actor may be a more specific instance of another.**
  - Uses UML "generalization" notation
  - Read this as a "is-a" relationship
- **There are generally accepted style guidelines.**
  - Put the primary actor on the top left of the diagram
  - Don't have actors interact with each other
  - Put passive actors on the right side of the diagram with the arrow pointing towards the actor



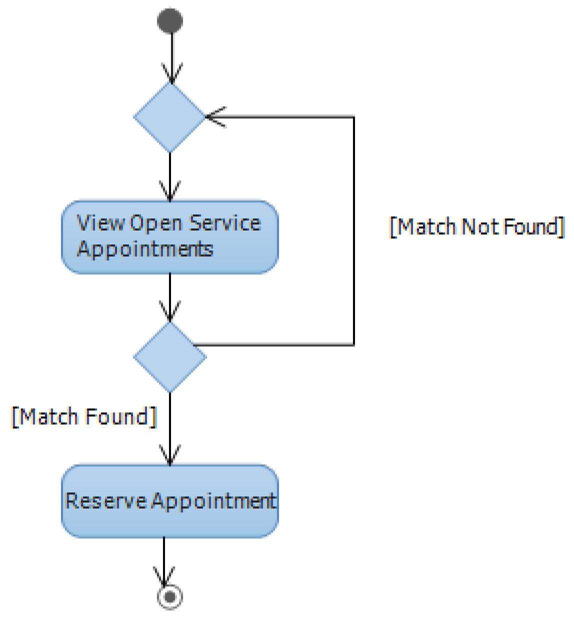
Building Use Case Diagrams – Use Cases

- **Naming is critical in use cases.**
  - Strong verb to start the name
  - Use business language related to the actor's point of view
  - Represent single behavior
- **Use "include" when you want to model common interactions.**
- **Use "extend" to show significant alternate flows.**
- **There are generally accepted style guidelines.**
  - Stack use cases to imply timing
  - Passive actors have arrow towards them
  - Put included use cases to the right of the base use case and have arrow point TO the included use case
  - Put extended use cases below the base use case and have the arrow point TO the base use case



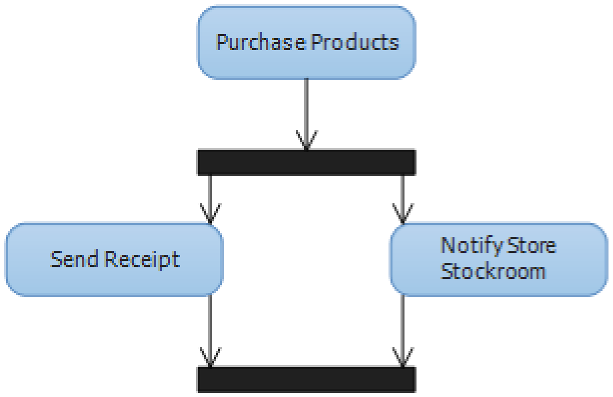
Building an Activity Diagram

- An activity diagram visually shows business processes or system algorithms.
- The core components of an activity diagram are:
  - Actions
  - Control Flow / Connector
  - Object Nodes
  - Control Nodes
  - Send Signal
  - Accept Event
  - Call Behavior
  - Call Operation
  - Input / Output Pins
  - Activity Parameters



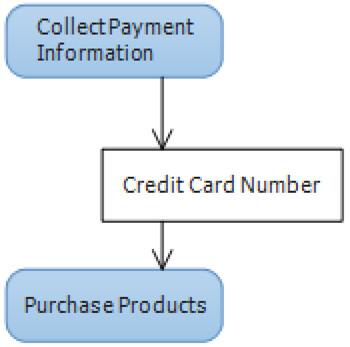
Building an Activity Diagram – Control Nodes

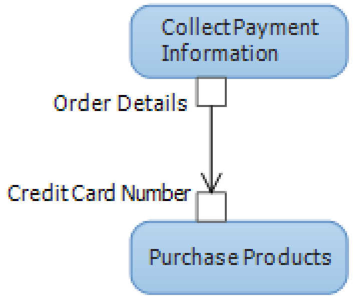
- Initial node starts an activity diagram and the final node completes it.
- Decision nodes splits a single incoming flow into multiple alternative flows.
- Merge node unifies incoming alternate flows into single output flow.
- Parallel activities can be modeled using the “fork” and “join” nodes.



Building an Activity Diagram – Data Flow

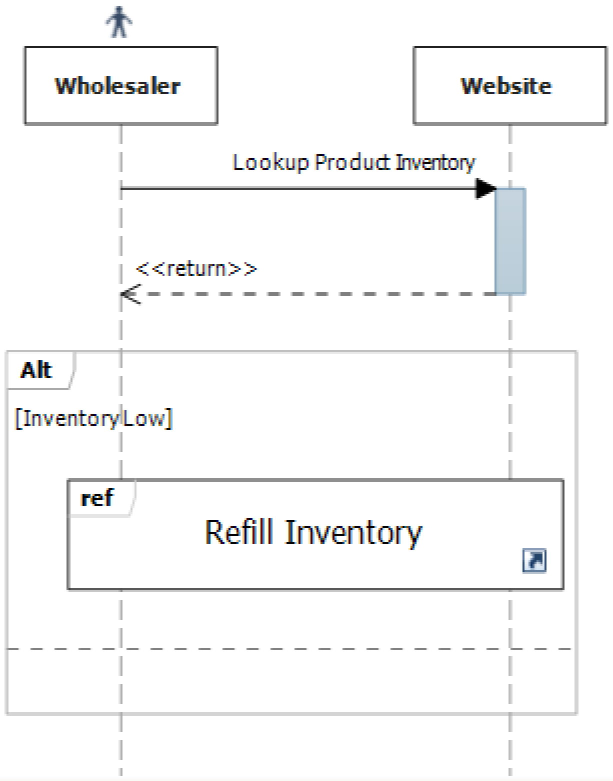
- **Data flow can be modeled using Object nodes.**
  - Can define the Types of the object being passed from Action to Action
- **Input and output pins can also be used to model data flow.**
  - Output pins represent the data that an Action produces
  - Input pins represents the data that an Action consumes





Building a Sequence Diagram

- A sequence diagram shows how components interact over time within a scenario.
- The core components of a sequence diagram are:
  - Lifelines
  - Messages
  - Interaction Occurrence
  - Fragments



Building a Sequence Diagram - Messages

- **Use synchronous messages when the object needs something back.**
  - Return messages may include requested data or other information
  - It is optional to show return messages but Visual Studio requires it
  - Can be a “self message” is from the participant to themselves
- **Use asynchronous messages when the sender does not need a response.**
- **Create messages are used when the sender creates an instance of the receiver.**

Building a Sequence Diagram – References and Fragments

- **An Interaction Occurrence, or "Interaction Use" in Visual Studio 2010, is used to reference an existing diagram.**
- **Fragments encapsulate conditional processing or variations in a sequence diagram. Guards describe when the fragment executes.**
  - Optional (opt)
  - Alternatives (alt)
  - Looping (loop)
  - Break (break)
  - Parallel (par)
  - Sequential (seq)
  - Critical (critical)
  - Strict (strict)

Summary

- UML in a Nutshell
- Support for UML in Visual Studio 2010
- Training Course Scenario Review
- Building a Use Case Diagram
- Building an Activity Diagram
- Building a Sequence Diagram