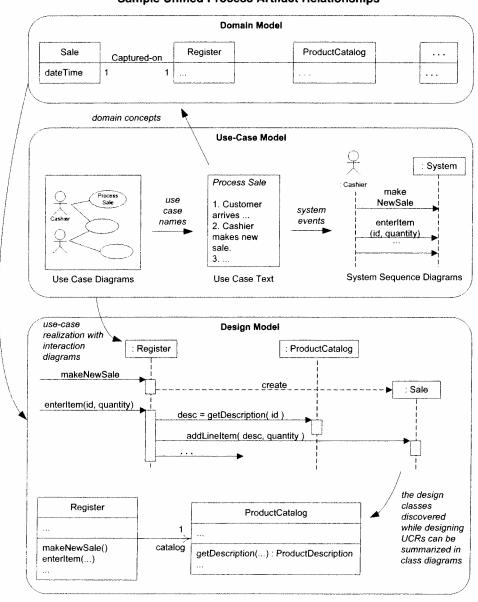
Sample Unified Process Artifacts and Timing (s-start; r-refine)

| Discipline        | Artifact                           | Incep.     | Elab. | Const. | Trans. |
|-------------------|------------------------------------|------------|-------|--------|--------|
|                   | Iteration→                         | <b>I</b> 1 | E1En  | C1Cn   | T1T2   |
| Business Modeling | Domain Model                       |            | s     |        |        |
| Requirements      | Use-Case Model                     | s          | r     |        |        |
|                   | Vision                             | s          | r     |        |        |
|                   | Supplementary Specification        | s          | r     |        |        |
|                   | Glossary                           | S          | r     | ·      |        |
| Design            | Design Model                       |            | s     | r      |        |
|                   | SW Architecture Document           |            | s     |        |        |
|                   | Data Model                         |            | s     | r      |        |
| Implementation    | Implementation Model (code, html,) |            | s     | r      | r      |

## Sample Unified Process Artifact Relationships



conceptual classes in the domain inspire the names of some software classes in the design General Responsibility Assignment Software Patterns or Principles (GRASP)

| Pattern/<br>Principle            | Description  |  |  |  |
|----------------------------------|--|--|--|--|
| Information<br>Expert            | A general principle of object design and responsibility assignment?  |  |  |  |
|                                  | Assign a responsibility to the information expert—the class that has the information necessary to fulfill the responsibility.  |  |  |  |
| Creator                          | Who creates? (Note that Factory is a common alternate solution.)   |  |  |  |
|                                  | Assign class B the responsibility to create an instance of class A if one of these is true:  1. B contains A  2. B aggregates A  3. B has the initializing data for A  4. B records A  5. B closely uses A   |  |  |  |
| Controller                       | What first object beyond the UI layer receives and coordinates ("controls") a system operation?  |  |  |  |
|                                  | Assign the responsibility to an object representing one of these choices:  1. Represents the overall "system," a "root object," a device that the software is running within, or a major subsystem (these are all variations of a facade controller).  2. Represents a use case scenario within which the system operation occurs (a use-case or session controller) |  |  |  |
| Low Coupling<br>(evaluative)     | How to reduce the impact of change?  |  |  |  |
| (evaluative)                     | Assign responsibilities so that (unnecessary) coupling remains low. Use this principle to evaluate alternatives.   |  |  |  |
| High<br>Cohesion<br>(evaluative) | How to keep objects focused, understandable, and manageable, and as a side-effect, support Low Coupling?   |  |  |  |
|                                  | Assign responsibilities so that cohesion remains high. Use this to evaluate alternatives.  |  |  |  |
| Polymorphism                     | Who is responsible when behavior varies by type?   |  |  |  |
|                                  | When related alternatives or behaviors vary by type (class), assign responsibility for the behavior—using polymorphic operations—to the types for which the behavior varies.   |  |  |  |
| Pure<br>Fabrication              | Who is responsible when you are desperate, and do not want to violate high cohesion and low coupling?  |  |  |  |
|                                  | Assign a highly cohesive set of responsibilities to an artificial or convenience "behavior" class that does not represent a problem domain concept—something made up, in order to support high cohesion, low coupling, and reuse.  |  |  |  |
| Indirection                      | How to assign responsibilities to avoid direct coupling?   |  |  |  |
|                                  | Assign the responsibility to an intermediate object to mediate between other components or services, so that they are not directly coupled.  |  |  |  |
| Protected<br>Variations          | How to assign responsibilities to objects, subsystems, and systems so that the variations or instability in these elements do not have an undesirable impact on other elements?  |  |  |  |
|                                  | Identify points of predicted variation or instability; assign responsibilities to create a stable "interface" around them.   |  |  |  |

Sample UML Notation Class Diagram AlternateUMLFor Abstract «interface» ClassX AbstractClass ClassX InterfaceX {abstract} - classOrStaticAttribute : Int operation1() + publicAttribute : String - privateAttribute assumedPrivateAttribute isInitializedAttribute : Bool = true interface generalization burgerCollection : VeggieBurger [ \* ] **FinalClass** implementation {leaf} attributeMayLegallyBeNull : String [0 1] finalConstantAttribute : Int = 5 { readOnly } ClassY /derivedAttribute Singleton operation1() + classOrStaticMethod() + publicMethod() assumedPublicMethod() - privateMethod() Whole AlternateUMLFor # protectedMethod() ImplOfInterfaceX InterfaceX ~ packageVisibleMethod() «constructor» SuperclassFoo( Long ) operation1() Composition methodWithParms(parm1 : String, parm2 : Float) methodReturnsSomething(): VeggieBurger Part methodThrowsException() {exception IOException} Associations: abstractMethod() abstractMethod2() { abstract } // alternate Association-name ClassA finalMethod() { leaf } // no override in subclass ClassB role-2 synchronizedMethod() { guarded } AssociationClass Multiplicity: zero or 1..40 Class more; one to Class Class one or exactly Class "many" forty more five Domain::ClassX o home package of class Runnable Clock myClock: Runnable myThread : Thread myClock active class run() navigability Use Case Diagram NextGen POS Process Sale **Extension Points:** «actor» CreditAuthorization Payment VIP Customer Service Cashier «extend» Payment, if Customer «include» presents a gift certificate Handle Gift Certificate Handle Credit Payment **Payment** 

