The Elements of UML Style

Scott W. Ambler
Ronin International, Inc.
Contents

Preface ............................................. xi

Purpose ............................................... xi

Features ............................................... xi

Audience ............................................. xii

Assumptions .......................................... xii

Acknowledgments .................................... xiii

1. Introduction ................................... 1
   1.1 Organization of This Book ..................... 2

2. General Diagramming Guidelines ............... 3
   2.1 Readability Guidelines ....................... 3
   2.2 Simplicity Guidelines ....................... 6
   2.3 Naming Guidelines ......................... 9
   2.4 General Guidelines ......................... 10
   2.5 Common UML Modeling Elements Guidelines .... 12

3. UML Use Case Diagrams ......................... 14
   3.1 Use Case Guidelines ....................... 15
   3.2 Actor Guidelines ......................... 17
   3.3 Relationship Guidelines ................... 19
   3.4 System Boundary Box Guidelines ............. 25

4. UML Class Diagrams .......................... 27
   4.1 ........................................... 27
   4.2 ........................................... 31
CONTENTS

4.3 Interface Guidelines ................................38
4.4 Relationship Guidelines ............................40
4.5 Association Guidelines .............................45
4.6 Inheritance Guidelines .............................48
4.7 Aggregation and Composition Guidelines .............50

5. Package Diagrams .......................... 53
5.1 Class Package Diagram Guidelines ...................53
5.2 Use Case Package Diagram Guidelines ............... 56
5.3 Package Guidelines ................................58

6. UML Sequence Diagrams .......................... 60
6.1 General Guidelines ................................61
6.2 Classifier Guidelines ...............................66
6.3 Message Guidelines ................................69
6.4 Return Value Guidelines ............................71

7. UML Collaboration Diagrams ..................... 74
7.1 General Guidelines ................................74
7.2 Message Guidelines ................................77
7.3 Link Guidelines ...................................79

8. UML State Chart Diagrams ......................... 82
8.1 General Guidelines ................................82
8.2 State Guidelines ...................................84
8.3 Substate Modeling Guidelines .......................85
8.4 Transition and Action Guidelines....................87
8.5 Guard Guidelines ..................................90

9. UML Activity Diagrams ........................... 92
9.1 General Guidelines ................................92
9.2 Activity Guidelines ................................95
9.3 Substate Modeling Guidelines .......................95
9.4 Transition and Action Guidelines....................96
5. Package Diagrams

A “package diagram” is any UML diagram, commonly a UML use case diagram or UML class diagram, composed only of packages. A package is a UML construct that enables you to organize model elements, such as use cases or classes, into groups. Packages are depicted as file folders and can be applied on any UML diagram. Although package diagrams are not official UML diagrams, their creation is common enough in practice to warrant discussion. Create a package diagram to

- depict a high-level overview of your requirements,
- depict a high-level overview of your design,
- logically modularize a complex diagram,
- organize source code, or
- model a framework (Evitts 2000).

5.1 Class Package Diagram Guidelines

113. Create Class Package Diagrams to Logically Organize Your Design

Figure 19 depicts a UML class diagram organized into packages. In addition to the package guidelines presented later in this chapter, apply the following heuristics to organize UML class diagrams into package diagrams:

- Classes of a framework belong in the same package.
- Classes in the same inheritance hierarchy typically belong in the same package.
Security <<infrastructure>>

Order <<domain>>

Shipping <<user interface>>

Customer <<domain>>

Security <<infrastructure>>

Persistence <<infrastructure>>

Corporate DB <<database>>

Online Ordering <<user interface>>

Apache Struts <<user interface framework>>

Figure 19. A class package diagram.
Classes related to one another via aggregation or composition often belong in the same package.

Classes that collaborate with each other a lot often belong in the same package.

### 114. Create UML Component Diagrams to Physically Organize Your Design

If you have decided on a component-based approach to design, such as that promoted by Enterprise Java Beans (EJB) (Roman et al. 2002) or Visual Basic, you should prefer a UML component diagram over a package diagram to depict your physical design. A version of Figure 19 as a UML component diagram is presented in Chapter 10 and, as you can see, that diagram is better suited for a physical design. Always remember to follow Agile Modeling’s (Chapter 12) Apply the Right Artifact(s) practice.

### 115. Place Inheriting Packages Below Base Packages

Inheritance between packages is depicted in Figure 19 and, as you can see, the inheriting package is shown below the base package. This approach is consistent with other inheritance guidelines.

### 116. Vertically Layer Class Package Diagrams

Dependencies between packages indicate that the contents of the dependent package depend on, or have structural knowledge of, the contents of the other package. In Figure 19, the packages are placed on the diagram to reflect the logical layering of your architecture. The user interface interacts with domain classes, which in turn use infrastructure classes, some of which access the database—which is traditionally depicted in a top-down manner.
5.2 Use Case Package Diagram Guidelines

Use cases are a primary requirement artifact in many object-oriented development methodologies. This is particularly true of instantiations of the Unified Process (Rational Corporation 2002; Ambler 2000). For larger projects, package diagrams are often created to organize these usage requirements.

117. Create Use Case Package Diagrams to Organize Your Requirements

In addition to the package guidelines presented below, apply the following heuristics to organize UML use case diagrams into package diagrams:

- Keep associated use cases together: included, extending, and inheriting use cases belong in the same package as the base/parent use case.
- Group use cases on the basis of needs of the main actors. For example, in Figure 20, the Enrollment package contains use cases pertinent to enrolling students in seminars, a vital collection of services provided by the university.

118. Include Actors on Use Case Package Diagrams

Including actors on package diagrams helps to put the packages in context, making diagrams easier to understand.

119. Horizontally Arrange Use Case Package Diagrams

The primary audience of Use Case package diagrams is project stakeholders; therefore, the organization of these diagrams should reflect their needs. The packages in Figure 20 are arranged horizontally, with dependencies drawn from left to right to reflect the direction that people in Western cultures read.
Figure 20. A UML use case diagram comprising mostly packages.
5.3 Package Guidelines
The advice presented in this section is applicable to packages on any UML diagram, not just package diagrams.

120. Give Packages Simple, Descriptive Names
In both Figure 19 and Figure 20, the packages have simple, descriptive names, such as Shipping, Customer, Enrollment, and Manage Loans, and Grants, which make it very clear what the package encapsulates.

121. Make Packages Cohesive
Anything that you put into a package should make sense when considered with the rest of the package contents. A good test to determine whether a package is cohesive is whether you can give your package a short, descriptive name. If you can’t, then you likely have put several unrelated things into the package.

122. Indicate Architectural Layers with Stereotypes on Packages
It is very common to organize your design into architectural layers such as user interface, business/domain, persistence/data, and infrastructure/system. In Figure 19 you see that stereotypes such as <<user interface>>, <<domain>>, <<infrastructure>>, and <<database>> have been applied to packages.

123. Avoid Cyclic Dependencies Between Packages
Knoernschild (2002) advises that you avoid the situation in which package A is dependent on package B which is dependent on package C which in turn is dependent on package A—in this case, A → B → C → A forms a cycle. Because these packages are coupled to one another, they will be harder to test, maintain, and enhance over time. Cyclic dependencies are a good indicator that you need to refactor one or more
packages, removing the elements from them that are causing the cyclic dependency.

124. Reflect Internal Relationships in Package Dependencies

When one package depends on another, it implies that there are one or more relationships between the contents of the two packages. For example, if it's a use case package diagram, then there is likely an include, extend, or inheritance relationship between a use case in one package and one in the other package.