THE Elements OF UML 2.0 Style

Scott W. Ambler

CAMBRIDGE www.cambridge.org/9780521616782

This page intentionally left blank

The Elements of UML™ *2.0 Style*

For Beverley

The Elements of UML™ *2.0 Style*

Scott W. Ambler

CAMBRIDGE
UNIVERSITY PRESS

CAMBRIDGE UNIVERSITY PRESS
Cambridge, New York, Melbourne, Madrid, Cape Town, Singapore, São Paulo

Cambridge University Press
The Edinburgh Building, Cambridge CB2 2RU, UK

Published in the United States of America by Cambridge University Press, New York

www.cambridge.org
Information on this title: www.cambridge.org/9780521616782

First published in print format 2005

ISBN-13 978-0-511-12389-4 eBook (Adobe Reader)
ISBN-10 0-511-12389-2 eBook (Adobe Reader)

ISBN-13 978-0-521-61678-2 paperback
ISBN-10 0-521-61678-6 paperback

Contents

Preface

Models are used by professional developers to communicate their work to project stakeholders and to other developers. The Unified Modeling Language (UML) has been an important part of the software development landscape since its introduction in 1997. We've seen the UML evolve over the years and it is now into its 2.x series of releases. Modeling style, however, has remained constant and will continue to do so. By understanding and following these common modeling style guidelines, you can improve the effectiveness of your models.

I've updated this book to include the new diagrams in UML 2, to use the terminology of UML 2, and to include hand-drawn diagrams. The vast majority of models are drawn on whiteboards and I think that it's time that modeling books, including this one, reflect that reality.

Purpose

This book describes a collection of standards, conventions, and guidelines for creating effective UML diagrams. They are based on sound, proven principles that will lead to diagrams that are easier to understand and work with.

These simple, concise guidelines, if applied consistently, will be an important first step in increasing your productivity as a modeler.

Features

This guide attempts to emulate Strunk and White's (1979) seminal text, *The Elements of Style*, which lists a set of rules describing the proper application of grammatical and compositional forms in common use within the written English language.

Using a similar style of presentation, this book defines a set of rules for developing high-quality UML diagrams. In doing so, this guide

- employs existing standards defined by the Object Management Group (OMG) whenever possible,
- provides a justification for each rule, and
- presents standards based on real-world experience and proven principles.

Audience

This guide targets information technology (IT) professionals who are interested in

- creating effective UML diagrams,
- increasing their productivity, and
- working as productive members of a software development team.

Assumptions

In this book I make several assumptions:

- You understand the basics of the UML and modeling. If not, then I suggest *UML Distilled* (Fowler 2004) if you are looking for a brief overview of the UML, or better yet *The Object Primer*, third edition (Ambler 2004) for a

more comprehensive discussion. *UML Distilled* is a great book but is limited to the UML; *The Object Primer*, third edition, on the other hand, goes beyond the UML where needed, for example, to include user interface, Java, and database development issues. It also covers agile software development techniques in detail.

- You are looking for style guidelines, not design guidelines. If not, then I suggest the book *Object-Oriented Design Heuristics* (Riel 1996).
- Your focus is on business application development. Although these guidelines also apply to real-time development, all of the examples are business application–oriented, simplifications of actual systems that I have built in the past.
- You belong to a Western culture. Many of the layout guidelines are based on the Western approach to reading—left to right and top down. People in other cultures will need to modify these guidelines as appropriate.

Acknowledgments

The following people have provided valuable input into the development and improvement of this text: James Bielak, Chris Britton, Larry Brunelle, Lauren Cowles, Beverley Dawe, Caitlin Doggart, Doug English, Jessica Farris, Scott Fleming, Mark Graybill, Alvery Grazebrook, Jesper R. Jensen, Jon Kern, Kirk W. Knoernschild, Hubert Matthews, Les Munday, Sabine Noack, Paul Oldfield, Marco Peters, Scott W. Preece, Neil Pitman, Edmund Schweppe, Leo Tohill, Tim Tuxworth, Michael Vizdos, and Robert White.

1.

Introduction

One of Agile Modeling's (AM) practices (discussed in Chapter 17) is *Apply Modeling Standards*, the modeling version of Extreme Programming (XP)'s *Coding Standards* (Beck 2000). Developers should agree to and follow a common set of standards and guidelines on a software project, and some of those guidelines should apply to modeling. Models depicted with a common notation and that follow effective style guidelines are easier to understand and to maintain. These models will improve communication internally within your team and externally to your partners and customers, thereby reducing the opportunities for costly misunderstandings. Modeling guidelines will also save you time by limiting the number of stylistic choices you face, allowing you to focus on your actual job – to develop software.

> A lot of the communication value in a UML diagram is still due to the layout skill of the modeler.
>
> —Paul Evitts, *A UML Pattern Language* (Evitts 2000)

When you adopt modeling standards and guidelines within your organization, your first step is to settle on a common notation. The Unified Modeling Language (UML) (Object Management Group 2004) is a good start because it defines the notation and semantics for common object-oriented models. Some projects will require more types of models than the UML describes, as I show in *The Object Primer 3/e* (Ambler 2004), but the UML will form the core of any modern modeling effort.

Your second step is to identify modeling style guidelines to help you to create consistent and clean-looking diagrams. What is the difference between a standard and a style guideline? For source code, a standard would, for example, involve naming the attributes in the format *attributeName*, whereas a style guideline would involve indenting your code within a control structure by three spaces. For models, a standard would involve using a squared rectangle to model a class on a class diagram, whereas a style would involve placing subclasses on diagrams below their superclass(es). This book describes the style guidelines that are missing from many of the UML-based methodologies that organizations have adopted, guidelines that are critical to your success in the software development game.

The third step is to enact your modeling standards and guidelines. To do this, you will need to train and mentor your staff in the modeling techniques appropriate to the projects on which they are working. You will also need to train and mentor them in your adopted guidelines, and a good start is to provide them with a copy of this book. I've been amazed at the success of *The Elements of Java Style* (Vermeulen et al. 2000) with respect to this—hundreds of organizations have adopted that book for their internal Java coding standards because they recognized that it was more cost-effective for them to buy a pocketbook for each developer than to develop their own guidelines.

1.1 Organization of This Book

This book is organized in a straightforward manner. Chapter 2 describes general diagramming principles that are applicable to all types of UML diagrams (and many non-UML diagrams for that matter). Chapter 3 describes guidelines for common UML elements such as stereotypes, notes, and frames.

Chapters 4 through 16 describe techniques pertinent to each type of UML diagram. Chapter 17 provides an overview of the values, principles, and practices of AM, with a quick reference to this popular methodology.

2.

General Diagramming Guidelines

The guidelines presented in this chapter are applicable to all types of diagrams, UML or otherwise. The terms "symbols," "lines," and "labels" are used throughout:

- Symbols represent diagram elements such as class boxes, object boxes, use cases, and actors.
- Lines represent diagram elements such as associations, dependencies, and transitions between states.
- Labels represent diagram elements such as class names, association roles, and constraints.

2.1 Readability Guidelines

1. Avoid Crossing Lines

When two lines cross on a diagram, such as two associations on a UML class diagram, the potential for misreading a diagram exists.

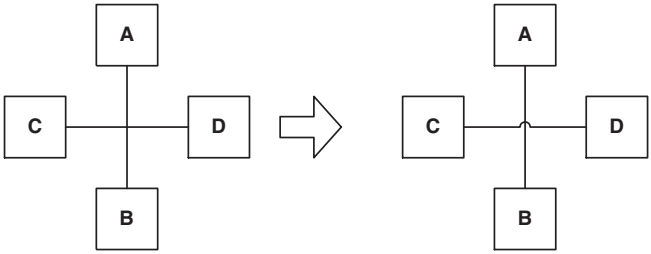

Figure 1. Depiction of crossing lines.

2. Depict Crossing Lines as a Jump

You can't always avoid crossing lines; for example, you cannot fully connect five symbols (try it and see). When you need to have two lines cross, one of them should "hop" over the other as in Figure 1.

3. Avoid Diagonal or Curved Lines

Straight lines, drawn either vertically or horizontally, are easier for your eyes to follow than diagonal or curved lines. A good approach is to place symbols on diagrams as if they are centered on the grid point of a graph, a built-in feature of many computer-aided system-engineering (CASE) tools. This makes it easier to connect your symbols by only using horizontal and vertical lines. Note how three lines are improved in Figure 2 when this approach is taken. Also note how the line between *A* and *C* has been depicted in "step fashion" as a line with vertical and horizontal segments.

4. Apply Consistently Sized Symbols

The larger a symbol appears, the more important it seems to be. In the first version of the diagram in Figure 2, the *A* symbol is larger than the others, drawing attention to it. If that isn't the effect that you want, then strive to make your symbols of uniform size. Because the size of some symbols is

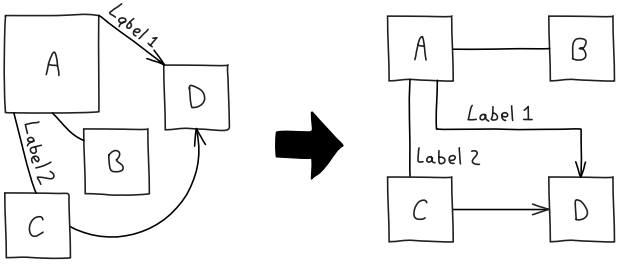

Figure 2. Improving the attractiveness of a diagram.

determined by their contents—for example, a class will vary in size based on its attributes and operations—this rule is not universally applicable. Ideally you should only deviate if you want to accentuate an aspect of your diagram (Koning, Dormann, and Van Vliet 2002).

5. Attach Lines to the Middle of Bubbles

As you can see in Figure 2, the *Label 1* line between *A* and *D* is much more readable in the updated version of the diagram.

6. Align Labels Horizontally

In Figure 2 the two labels are easier to read in the second version of the diagram. Notice how *Label 2* is horizontal even though the line it is associated with is vertical.

7. Arrange Symbols Symmetrically

Figure 3 presents a UML activity diagram (Chapter 10) depicting a high-level approach to enterprise modeling. Organizing the symbols and lines in a symmetrical manner makes the diagram easier to understand. A clear pattern will make a diagram easier to read.

8. Don't Indicate "Exploding" Bubbles

The rake symbol in the upper right corner of each activity in Figure 3 is the UML way to indicate that they "explode" to

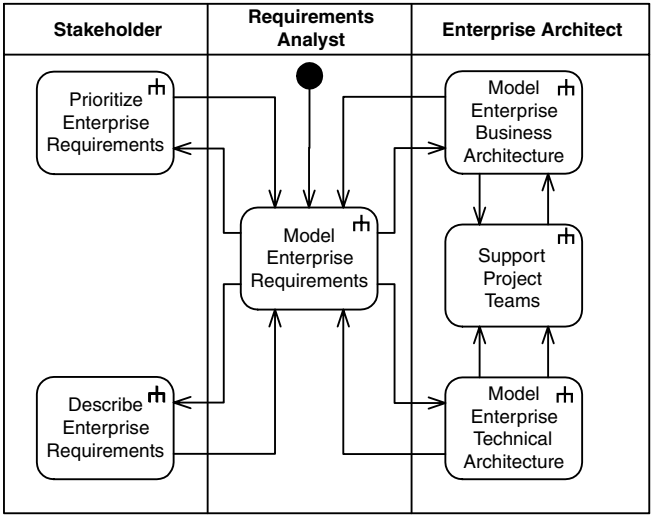

Figure 3. UML activity diagram for a software process.

another diagram showing a greater level of detail. Although this seems like a good idea, the reality is that people using a CASE tool know enough to double click on it, or whatever strategy the tool implements, to get more detail. The rake isn't adding any extra value.

9. Minimize the Number of Bubble Types

Koning, Dormann, and Van Vliet (2002) recommend that you have six or fewer bubbles on a diagram; any more risks overwhelming the user of the model.

10. Include White Space in Diagrams

White space is the empty areas between modeling elements on your diagrams. In the first version of Figure 2 the symbols are crowding each other, whereas in the second version, the symbols are spread out from one another, thus improving the