

# **Large-Scale C++ Software Design**

**John Lakos**



**ADDISON-WESLEY PUBLISHING COMPANY**

Reading, Massachusetts Menlo Park, California New York  
Don Mills, Ontario Wokingham, England Amsterdam Bonn  
Sydney Singapore Tokyo Madrid San Juan  
Seoul Milan Mexico City Taipei

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book and Addison-Wesley was aware of a trademark claim, the designations have been printed with initial capital letters.

The authors and publishers have taken care in the preparation of this book, but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information of programs contained herein.

The publisher offers discounts on this book when ordered in quantity for special sales. For more information please contact:

Corporate & Professional Publishing Group  
Addison-Wesley Publishing Company  
One Jacob Way  
Reading, Massachusetts 01867

### Library of Congress Cataloging-in-Publication Data

Lakos, John, 1959--

Large-scale C++ software design / John Lakos.

p. cm. -- (Addison-Wesley professional computing series)

Includes bibliographical references and index.

ISBN 0-201-63362-0 (pbk. : alk. paper)

1. C++ (Computer program language) 2. Computer software--  
--Development. I. Title. II. Series.

QA76.73.C153L342 1996

005.13'3--dc20

95-52106

CIP

#### DEDICATION

*To my parents, Marci and Gene, who prepared and encouraged me.*

*To my wife, Cathy, who suffered through it with me.*

*To my daughter, Sarah, who was born in the middle of it all.*

Copyright © 1996 by Addison-Wesley Publishing Company, Inc.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior consent of the publisher. Printed in the United States of America. Published simultaneously in Canada.

ISBN 0-201-63362-0

Text printed on recycled and acid-free paper

1 2 3 4 5 6 7 8 9 10 CRW 99989796

First Printing, June 1996

# Contents

<b>Figure List</b>	<b>xiii</b>
<b>Preface</b>	<b>xxv</b>
<b>Chapter 0: Introduction</b>	<b>1</b>
0.1 From C to C++ .....	2
0.2 Using C++ to Develop Large Projects .....	2
0.2.1 Cyclic Dependencies .....	3
0.2.2 Excessive Link-Time Dependencies .....	5
0.2.3 Excessive Compile-Time Dependencies .....	7
0.2.4 The Global Name Space .....	10
0.2.5 Logical vs. Physical Design .....	12
0.3 Reuse .....	14
0.4 Quality .....	15
0.4.1 Quality Assurance .....	16
0.4.2 Quality Ensurance .....	17
0.5 Software Development Tools .....	17
0.6 Summary .....	18
<b>PART I: BASICS</b>	<b>19</b>
<b>Chapter 1: Preliminaries</b>	<b>21</b>
1.1 Multi-File C++ Programs .....	21
1.1.1 Declaration versus Definition .....	22
1.1.2 Internal versus External Linkage .....	23

1.1.3 Header (.h) Files .....	27
1.1.4 Implementation (.c) Files .....	29
1.2 typedef Declarations .....	30
1.3 Assert Statements .....	32
1.4 A Few Matters of Style .....	33
1.4.1 Identifier Names .....	34
1.4.1.1 Type Names .....	34
1.4.1.2 Multi-Word Identifier Names .....	35
1.4.1.3 Data Member Names .....	36
1.4.2 Class Member Layout .....	38
1.5 Iterators .....	40
1.6 Logical Design Notation .....	47
1.6.1 The IsA Relation .....	48
1.6.2 The Uses-In-The-Interface Relation .....	50
1.6.3 The Uses-In-The-Implementation Relation .....	52
1.6.3.1 Uses .....	55
1.6.3.2 HasA and HoldsA .....	56
1.6.3.3 WasA .....	57
1.7 Inheritance versus Layering .....	58
1.8 Minimality .....	59
1.9 Summary .....	61

## **Chapter 2: Ground Rules 63**

2.1 Overview .....	63
2.2 Member Data Access .....	65
2.3 The Global Name Space .....	69
2.3.1 Global Data .....	69
2.3.2 Free Functions .....	72
2.3.3 Enumerations, Typedefs, and Constant Data .....	73
2.3.4 Preprocessor Macros .....	75
2.3.5 Names in Header Files .....	77
2.4 Include Guards .....	80
2.5 Redundant Include Guards .....	82
2.6 Documentation .....	88
2.7 Identifier-Naming Conventions .....	91
2.8 Summary .....	93

## PART II: PHYSICAL DESIGN CONCEPTS 97

### Chapter 3: Components 99

3.1 Components versus Classes .....	99
3.2 Physical Design Rules .....	108
3.3 The DependsOn Relation .....	120
3.4 Implied Dependency .....	127
3.5 Extracting Actual Dependencies .....	134
3.6 Friendship .....	136
3.6.1 Long-Distance Friendship and Implied Dependency .....	141
3.6.2 Friendship and Fraud .....	144
3.7 Summary .....	147

### Chapter 4: Physical Hierarchy 149

4.1 A Metaphor for Software Testing .....	149
4.2 A Complex Subsystem .....	151
4.3 The Difficulty in Testing “Good” Interfaces .....	155
4.4 Design for Testability .....	157
4.5 Testing in Isolation .....	161
4.6 Acyclic Physical Dependencies .....	164
4.7 Level Numbers .....	166
4.7.1 The Origin of Level Numbers .....	167
4.7.2 Using Level Numbers in Software .....	169
4.8 Hierarchical and Incremental Testing .....	174
4.9 Testing a Complex Subsystem .....	181
4.10 Testability versus Testing .....	183
4.11 Cyclic Physical Dependencies .....	184
4.12 Cumulative Component Dependency .....	187
4.13 Physical Design Quality .....	193
4.14 Summary .....	201

### Chapter 5: Levelization 203

5.1 Some Causes of Cyclic Physical Dependencies .....	204
5.1.1 Enhancement .....	204
5.1.2 Convenience .....	208
5.1.3 Intrinsic Interdependency .....	213

5.2 Escalation .....	215
5.3 Demotion .....	229
5.4 Opaque Pointers .....	247
5.5 Dumb Data .....	257
5.6 Redundancy .....	269
5.7 Callbacks .....	275
5.8 Manager Class .....	288
5.9 Factoring .....	294
5.10 Escalating Encapsulation .....	312
5.11 Summary .....	324

## Chapter 6: Insulation

327

6.1 From Encapsulation to Insulation .....	328
6.1.1 The Cost of Compile-Time Coupling .....	333
6.2 C++ Constructs and Compile-Time Coupling .....	335
6.2.1 Inheritance (IsA) and Compile-Time Coupling .....	336
6.2.2 Layering (HasA/HoldsA) and Compile-Time Coupling .....	337
6.2.3 Inline Functions and Compile-Time Coupling .....	339
6.2.4 Private Members and Compile-Time Coupling .....	341
6.2.5 Protected Members and Compile-Time Coupling .....	342
6.2.6 Compiler-Generated Member Functions and Compile-Time Coupling .....	342
6.2.7 Include Directives and Compile-Time Coupling .....	344
6.2.8 Default Arguments and Compile-Time Coupling .....	346
6.2.9 Enumerations and Compile-Time Coupling .....	347
6.3 Partial Insulation Techniques .....	349
6.3.1 Removing Private Inheritance .....	349
6.3.2 Removing Embedded Data Members .....	352
6.3.3 Removing Private Member Functions .....	353
6.3.4 Removing Protected Members .....	363
6.3.5 Removing Private Member Data .....	375
6.3.6 Removing Compiler-Generated Functions .....	378
6.3.7 Removing Include Directives .....	379
6.3.8 Removing Default Arguments .....	381
6.3.9 Removing Enumerations .....	382
6.4 Total Insulation Techniques .....	385
6.4.1 The Protocol Class .....	386
6.4.2 The Fully Insulating Concrete Class .....	398
6.4.3 The Insulating Wrapper .....	405
6.4.3.1 Single-Component Wrappers .....	406
6.4.3.2 Multi-Component Wrappers .....	415

6.5 The Procedural Interface .....	425
6.5.1 The Procedural Interface Architecture .....	426
6.5.2 Creating and Destroying Opaque Objects .....	428
6.5.3 Handles .....	429
6.5.4 Accessing and Manipulating Opaque Objects .....	435
6.5.5 Inheritance and Opaque Objects .....	441
6.6 To Insulate or Not to Insulate .....	445
6.6.1 The Cost of Insulation .....	445
6.6.2 When Not to Insulate .....	448
6.6.3 How to Insulate .....	453
6.6.4 How Much to Insulate .....	460
6.7 Summary .....	468

## **Chapter 7: Packages** **473**

7.1 From Components to Packages .....	474
7.2 Registered Package Prefixes .....	483
7.2.1 The Need for Prefixes .....	483
7.2.2 Namespaces .....	486
7.2.3 Preserving Prefix Integrity .....	491
7.3 Package Levelization .....	493
7.3.1 The Importance of Levelizing Packages .....	494
7.3.2 Package Levelization Techniques .....	496
7.3.3 Partitioning a System .....	498
7.3.4 Multi-Site Development .....	500
7.4 Package Insulation .....	503
7.5 Package Groups .....	506
7.6 The Release Process .....	512
7.6.1 The Release Structure .....	514
7.6.2 Patches .....	520
7.7 The main Program .....	523
7.8 Start-Up .....	531
7.8.1 Initialization Strategies .....	534
7.8.1.1 The Wake-Up Initialized Technique .....	534
7.8.1.2 The Explicit <code>init</code> Function Technique .....	535
7.8.1.3 The Nifty Counter Technique .....	537
7.8.1.4 The Check-Every-Time Technique .....	543
7.8.2 Clean-Up .....	545
7.8.3 Review .....	546
7.9 Summary .....	546

## **PART III: LOGICAL DESIGN ISSUES** **551**

### **Chapter 8: Architecting a Component** **553**

8.1 Abstractions and Components .....	554
8.2 Component Interface Design .....	555
8.3 Degrees of Encapsulation .....	560
8.4 Auxiliary Implementation Classes .....	572
8.5 Summary .....	579

### **Chapter 9: Designing a Function** **583**

9.1 Function Interface Specification .....	584
9.1.1 Operator or Non-Operator Function .....	584
9.1.2 Free or Member Operator .....	591
9.1.3 Virtual or Non-Virtual Function .....	597
9.1.4 Pure or Non-Pure Virtual Member Function .....	603
9.1.5 Static or Non-Static Member Function .....	604
9.1.6 const Member or Non-const Member Function .....	605
9.1.7 Public, Protected, or Private Member Function .....	612
9.1.8 Return by Value, Reference, or Pointer .....	614
9.1.9 Return const or Non-const .....	618
9.1.10 Argument Optional or Required .....	619
9.1.11 Pass Argument by Value, Reference, or Pointer .....	621
9.1.12 Pass Argument as const or Non-const .....	629
9.1.13 Friend or Non-Friend Function .....	630
9.1.14 Inline or Non-Inline Function .....	631
9.2 Fundamental Types Used in the Interface .....	633
9.2.1 Using short in the Interface .....	633
9.2.2 Using unsigned in the Interface .....	637
9.2.3 Using long in the Interface .....	642
9.2.4 Using float, double, and long double in the Interface .....	644
9.3 Special-Case Functions .....	645
9.3.1 Conversion Operators .....	646
9.3.2 Compiler-Generated Value Semantics .....	650
9.3.3 The Destructor .....	651
9.4 Summary .....	655

### **Chapter 10: Implementing an Object** **661**

10.1 Member Data .....	662
10.1.1 Natural Alignment .....	662



10.1.2 Fundamental Types Used in the Implementation .....	665
10.1.3 Using typedef in the Implementation .....	667
10.2 Function Definitions .....	669
10.2.1 Assert Yourself! .....	669
10.2.2 Avoid Special Casing .....	670
10.2.3 Factor Instead of Duplicate .....	673
10.2.4 Don't Be Too Clever .....	680
10.3 Memory Management .....	681
10.3.1 Logical versus Physical State Values .....	681
10.3.2 Physical Parameters .....	685
10.3.3 Memory Allocators .....	691
10.3.4 Class-Specific Memory Management .....	698
10.3.4.1 Adding Custom Memory Management .....	702
10.3.4.2 Hogging Memory .....	705
10.3.5 Object-Specific Memory Management .....	711
10.4 Using C++ Templates in Large Projects .....	718
10.4.1 Compiler Implementations .....	718
10.4.2 Managing Memory in Templates .....	719
10.4.3 Patterns versus Templates .....	730
10.5 Summary .....	733

## Appendix A

### The Protocol Hierarchy

737

Intent .....	739
Also Known As .....	739
Motivation .....	739
Applicability .....	744
Structure .....	744
Participants .....	745
Collaborations .....	746
Consequences .....	746
Implementation .....	747
Sample Code .....	761
Known Uses .....	766
Related Patterns .....	767

<b>Appendix B</b>	
<b>Implementing an ANSI C-Compatible C++ Interface</b>	<b>769</b>
B.1 Memory Allocation Error Detection .....	769
B.2 Providing a Main Procedure (ANSI C Only) .....	778

<b>Appendix C</b>	
<b>A Dependency Extractor/Analyzer Package</b>	<b>779</b>
C.1 Using adep, cdep, and ldep .....	780
C.2 Command-Line Documentation .....	793
C.3 Idep Package Architecture .....	810
C.4 Source Code .....	813

<b>Appendix D</b>	
<b>Quick Reference</b>	<b>815</b>
D.1 Definitions .....	815
D.2 Major Design Rules .....	820
D.3 Minor Design Rules .....	821
D.4 Guidelines .....	822
D.5 Principles .....	824

<b>Bibliography</b>	<b>833</b>
---------------------	------------

<b>Index</b>	<b>835</b>
--------------	------------

0-1: Cyclic  
0-2: Oversi  
0-3: An In  
0-4: Unnec  
0-5: Unnec  
0-6: Typed  
1-1: Some  
1-2: Some  
1-3: What  
1-4: Typed  
1-5: Create  
1-6: Get/S  
1-7: Trailing  
1-8: A Sin  
1-9: Trivia  
1-10: Infe  
1-11: Atte  
1-12: Ano  
1-13: A St  
1-14: Whi  
1-15: Int  
1-16: The  
1-17: Not  
1-18: The  
1-19: Both