

Contents

<i>Preface to the second edition</i>	xv
<i>Preface</i>	xxi
<i>About the author</i>	xxvii
<i>About the contributing authors</i>	xxix
<i>Acknowledgements</i>	xxxi
PART ONE	
Motivation – components and markets	1
1 Introduction	3
1.1 Components are for composition	3
1.2 Components – custom-made versus standard software	4
1.3 Inevitability of components	6
1.4 The nature of software and deployable entities	8
1.5 Components are units of deployment	10
1.6 Lessons learned	12
2 Market versus technology	17
2.1 Creating a market	18
2.2 Fundamental properties of component technology	19
2.3 Market development	21
2.3.1 <i>Strategic Focus</i> (January 1995)	21
2.3.2 <i>Ovum</i> (1995)	22
2.3.3 IDC (May 1996)	22
2.3.4 Forrester Research (October 1996)	23
2.3.5 IDC (April 1999)	24
2.3.6 ComponentSource (2001)	25
2.3.7 Flashline (2001)	25
3 Standards	27
3.1 The utmost importance of (quasi) standards	27
3.2 Wiring standards are not enough	29
3.3 Too many competing standards are not useful	30
3.4 Where is software component technology today?	31
3.5 What's next?	32

PART TWO	Foundation	33
4	What a component is and is not	35
4.1	Terms and concepts	35
4.1.1	Components	36
4.1.2	Objects	37
4.1.3	Components and objects	38
4.1.4	Modules	39
4.1.5	Whitebox versus blackbox abstractions and reuse	40
4.1.6	Interfaces	42
4.1.7	Explicit context dependencies	44
4.1.8	Component "weight"	45
4.2	Standardization and normalization	46
4.2.1	Horizontal versus vertical markets	47
4.2.2	Standard component worlds and normalization	47
5	Components, interfaces, and re-entrance	49
5.1	Components and interfaces	50
5.1.1	Direct and indirect interfaces	50
5.1.2	Versions	52
5.1.3	Interfaces as contracts	53
5.1.4	Contracts and extra-functional requirements	54
5.1.5	Undocumented "features"	54
5.2	What belongs to a contract?	55
5.2.1	Safety and progress	55
5.2.2	Extra-functional requirements	55
5.2.3	Specifying time and space requirements	56
5.3	Dress code – formal or informal?	57
5.4	Callbacks and contracts	58
5.5	Examples of callbacks and contracts	59
5.5.1	A directory service	60
5.5.2	A client of the directory service	61
5.5.3	Same client, next release	62
5.5.4	A broken contract	64
5.5.5	Prevention is better than cure	65
5.5.6	Proofing the directory service	66
5.5.7	Test functions in action	66
5.6	From callbacks to objects	67
5.7	From interobject consistency to object re-entrance	74
5.8	Self-interference and object re-entrance: a summary	77
5.9	Processes and multithreading	79
5.10	Histories	79
5.11	Specification statements	81
6	Polymorphism	83
6.1	Substitutability – using one for another	83
6.2	Types, subtypes, and type checking	88
6.3	More on subtypes	90
6.4	Object languages and types	93

6.5	Types, interfaces, and components	93
6.6	The paradigm of independent extensibility	95
6.7	Safety by construction – viability of components	98
6.7.1	Module safety	99
6.7.2	Module safety and metaprogramming	99
6.7.3	Safety in a multilanguage environment	100
6.8	Safety, security, trust	101
6.9	Dimensions of independent extensibility	102
6.9.1	Bottleneck interfaces	103
6.9.2	Singleton configurations	104
6.9.3	Parallel, orthogonal, and recursive extensions	104
6.10	Evolution versus immutability of interfaces and contracts	105
6.10.1	Syntactic versus semantic contract changes	105
6.10.2	Contract expiry	106
6.10.3	Overriding law	106
6.11	Other forms of polymorphism	107
7	Object versus class composition or how to avoid inheritance	109
7.1	Inheritance – the soup of the day?	109
7.2	More flavors to the soup	111
7.2.1	Multiple inheritance	111
7.2.2	Mixins	113
7.3	Back to basic ingredients	115
7.4	The fragile base class problem	115
7.4.1	The syntactic fragile base class problem	116
7.4.2	The semantic fragile base class problem	116
7.5	Inheritance – more knots than meet the eye	117
7.6	Approaches to disciplined inheritance	122
7.6.1	The specialization interface	122
7.6.2	Typing the specialization interface	123
7.6.3	Behavioral specification of the specialization interface	125
7.6.4	Reuse and cooperation contracts	127
7.6.5	Representation invariants and method refinements	130
7.6.6	Disciplined inheritance to avoid fragile base class problems	131
7.6.7	Creating correct subclasses without seeing superclass code	131
7.7	From class to object composition	133
7.8	Forwarding versus delegation (or making object composition as problematical as implementation inheritance)	135
7.9	A brief review of delegation and inheritance	138
8	Aspects of scale and granularity	139
8.1	Units of abstraction	140
8.2	Units of accounting	141
8.3	Units of analysis	141
8.4	Units of compilation	142
8.5	Units of delivery	143
8.6	Units of deployment	143
8.7	Units of dispute	143
8.8	Units of extension	145

8.9	Units of fault containment	146
8.10	Units of instantiation	146
8.11	Units of installation	147
8.12	Units of loading	147
8.13	Units of locality	149
8.14	Units of maintenance	150
8.15	Units of system management	150
8.16	Summary	150
9	Patterns, frameworks, architectures	151
9.1	Forms of design-level reuse	152
9.1.1	Sharing consistency – programming languages	152
9.1.2	Sharing concrete solution fragments – libraries	153
9.1.3	Sharing individual contracts – interfaces	154
9.1.4	Sharing individual interaction fragments – messages and protocols	155
9.1.5	Sharing individual interaction architecture – patterns	156
9.1.6	Sharing architecture – frameworks	158
9.1.7	Sharing overall structure – system architecture	162
9.1.8	Systems of subsystems – framework hierarchies	164
9.2	Interoperability, legacies, and re-engineering	166
10	Programming – shades of gray	169
10.1	Different programming methods for different programmers	169
10.2	Programming to a system	172
10.3	Connection-oriented programming	172
10.4	Connection-oriented programming – advanced concepts	175
10.5	Events and messages	181
10.5.1	Message syntax and schema – XML	183
10.5.2	Events versus calls	185
10.5.3	Call syntax and protocol – SOAP	186
10.6	Ordering of events – causality, races, and glitches	187
10.7	Very late binding – dispatch interfaces and metaprogramming	189
10.8	Degrees of freedom – sandboxing versus static safety	192
10.9	Recording versus scripting	192
11	What others say	195
11.1	Grady Booch (1987)	195
11.2	Oscar Nierstrasz and Dennis Tsichritzis (1992 and 1995)	196
11.3	Gio Wiederhold, Peter Wegner, and Stefano Ceri (1992)	196
11.4	Ivar Jacobson (1993)	197
11.5	Meta Group (1994)	197
11.6	Jed Harris (1995)	197
11.7	Ovum Report on Distributed Objects (1995)	198
11.8	Robert Orfali, Dan Harkey, and Jeri Edwards (1995, 1996)	198
11.9	Johannes Sametinger (1997)	199
11.10	UML 1.3 Standard (1999)	200
11.11	Desmond D'Souza and Alan Wills (1999)	200
11.12	Krzysztof Czarnecki and Ulrich Eisenecker (2000)	201
11.13	Peter Herzum and Oliver Sims (2000)	202
11.14	CBSE Handbook (2001)	203

PART THREE	Component models and platforms	205
12	Object and component “wiring” standards	207
12.1	Where it all came from	207
12.2	From procedures to objects	209
12.3	The fine print	210
12.3.1	Specification of interfaces and object references	210
12.3.2	Interface relationships and polymorphism	211
12.3.3	Naming and locating services	211
12.3.4	Compound documents	212
12.4	On the wire – the rise of XML	214
12.4.1	XML, XML Namespaces, XML Schema	215
12.4.2	XML support standards	220
12.4.3	XML document object and streaming models	221
12.4.4	SOAP	222
12.4.5	XML web services: WSDL, UDDI, WSFL, XLANG	224
12.4.6	Web services and programming models	229
12.5	Which way?	230
13	The OMG way: CORBA, CCM, OMA, and MDA	231
13.1	At the heart – the object request broker	231
13.1.1	From CORBA to OMA	235
13.1.2	CORBA timeline	237
13.1.3	A bit of history – system object model (SOM)	238
13.2	Common object service specifications (CORBA services)	239
13.2.1	Services supporting enterprise distributed computing	240
13.2.2	Services supporting architecture using fine-grained objects	242
13.3	CORBA Component Model	247
13.3.1	Portable object adapter	247
13.3.2	CCM components	248
13.3.3	CCM containers	252
13.4	CORBA-compliant implementations	252
13.4.1	BEA's WebLogic	253
13.4.2	IBM's WebSphere	254
13.4.3	IONA's Orbix E2A Application Server Platform	255
13.4.4	Borland's Enterprise Server	255
13.4.5	Non-for-profit implementations	256
13.5	CORBA facilities	256
13.6	Application objects	257
13.7	CORBA, UML, XML, and MDA	258
13.7.1	Meta-object facility	259
13.7.2	Model-driven architecture (MDA)	259
14	The Sun way – Java, JavaBeans, EJB, and Java 2 editions	261
14.1	Overview and history of Java component technologies	261
14.1.1	Java versus Java 2	262
14.1.2	Runtime environment and reference implementations	263
14.1.3	Spectrum of editions – Micro, Standard, and Enterprise	265
14.2	Java, the language	270
14.2.1	Interfaces versus classes	273

14.2.2	Exceptions and exception handling	278
14.2.3	Threads and synchronization	279
14.2.4	Garbage collection	282
14.3	JavaBeans	284
14.3.1	Events and connections	286
14.3.2	Properties	288
14.3.3	Introspection	289
14.3.4	JAR files – packaging of Java components	292
14.4	Basic Java services	293
14.4.1	Reflection	293
14.4.2	Object serialization	296
14.4.3	Java native interface	298
14.4.4	Java AWT and JFC/Swing	299
14.4.5	Advanced JavaBeans specifications	300
14.5	Component variety – applets, servlets, beans, and Enterprise beans	302
14.5.1	Java server pages (JSP) and servlets	304
14.5.2	Contextual composition – Enterprise JavaBeans (EJB)	308
14.5.3	Data-driven composition – message-driven beans in EJB 2.0	316
14.6	Advanced Java services	316
14.6.1	Distributed object model and RMI	317
14.6.2	Java and CORBA	318
14.6.3	Enterprise service interfaces	319
14.6.4	Java and XML	323
14.7	Interfaces versus classes in Java, revisited	323
14.8	JXTA and Jini	324
14.8.1	Jini – federations of Java objects	325
14.8.2	JXTA – peer-to-peer computing	326
14.9	Java and web services – SunONE	328
15	The Microsoft way: COM, OLE/ActiveX, COM+, and .NET CLR	329
15.1	The first fundamental wiring model – COM	330
15.2	COM object reuse	335
15.3	Interfaces and polymorphism	338
15.3.1	Categories	339
15.3.2	Interfaces and versioning	340
15.4	COM object creation and the COM library	340
15.5	Initializing objects, persistence, structured storage, monikers	342
15.6	From COM to distributed COM (DCOM)	343
15.7	Meta-information and automation	345
15.8	Other COM services	346
15.8.1	Uniform data transfer	346
15.8.2	Dispatch interfaces (dispinterfaces) and dual interfaces	347
15.8.3	Outgoing interfaces and connectable objects	348
15.9	Compound documents and OLE	349
15.9.1	OLE containers and servers	350
15.9.2	Controls – from Visual Basic via OLE to ActiveX	351
15.10	Contextual composition and services	353
15.10.1	COM apartments – threading and synchronization	354
15.10.2	Microsoft transaction server – contexts and activation	355
15.10.3	COM+ – generalized contexts and data-driven composition	356

15.11	Take two – the .NET Framework	357
15.11.1	The .NET big picture	358
15.11.2	Common language infrastructure	358
15.11.3	COM and platform interoperation	361
15.11.4	Exemplary .NET language – C#	362
15.11.5	Visual Studio .NET	366
15.12	Assemblies – the .NET components	366
15.13	Common language frameworks	368
15.13.1	AppDomains, contexts, reflection, remoting	372
15.13.2	Windows Forms, data, management	375
15.13.3	Web Forms, Active Server Pages (ASP) .NET	376
15.13.4	XML and data	377
15.13.5	Enterprise services	378
15.13.6	Web services with .NET	378
16	Some further technologies	381
16.1	Computer Associates' Advantage Plex	381
16.2	Hitachi Appgallery	382
16.3	Groove Transceiver	382
17	Strategic comparison	385
17.1	Shared attributes	385
17.2	Differences	386
17.3	Consequences for infrastructure vendors	390
17.4	Consequences for component vendors	395
18	Efforts on domain standards	397
18.1	OMG Domain Technology Committee	397
18.1.1	OMG BODTF	398
18.2	W3C	398
18.3	Business processes and documents	399
18.3.1	OASIS and ebXML	399
18.3.2	RosettaNet and PIPs	400
18.3.3	BizTalk.org	401
18.4	DMTF's CIM and WBEM	402
18.5	Java domain standard efforts	403
18.6	OLE for process control	404
18.7	Industry associations	404
18.7.1	Information technology industry groupings	404
18.7.2	Trade associations	405
18.7.3	User associations	406
19	Ongoing concerns	407
19.1	Domain standards	407
19.2	Rethinking the foundations of software engineering	408
19.3	But is it object-oriented?	408
19.4	Object mobility and mobile agents	411
19.5	Foundations – better contracts for better components	412

PART FOUR	Components meet architecture and process	415
20	Component architecture	417
20.1	The roles of an architecture	417
20.2	Conceptualization – beyond objects?	418
20.3	Definitions of key terms	419
20.4	A tiered component architecture	421
20.5	Components and middleware	423
20.6	Components versus generative programming	424
21	Component frameworks	425
21.1	Contributions of contextual component frameworks	426
21.1.1	Foundation and roots	426
21.1.2	Component frameworks versus connectors	428
21.1.3	Component frameworks versus metaprogramming	430
21.1.4	Component frameworks versus aspect-oriented programming	430
21.2	Frameworks for contextual composition	431
21.2.1	COM+ contexts	432
21.2.2	EJB containers	433
21.2.3	CCM containers	434
21.2.4	CLR contexts and channels	434
21.2.5	Tuple and object spaces	436
21.3	BlackBox component framework	437
21.3.1	Carrier–rider–mapper design pattern	438
21.3.2	Directory objects	440
21.3.3	Hierarchical model view separation	441
21.3.4	Container modes	444
21.3.5	Cascaded message multicasting services	446
21.3.6	Advanced applications based on compound documents	448
21.4	BlackBox and OLE	449
21.5	Portos – a hard realtime component framework and its IDE	451
21.5.1	Structure of Portos	452
21.5.2	Realtime scheduler	453
21.5.3	Cross-development environment	455
22	Component development	457
22.1	The methodology – component-oriented programming	457
22.1.1	Problems of asynchrony	458
22.1.2	Multithreading	458
22.1.3	Learning from circuit design	459
22.1.4	Living without implementation inheritance	460
22.1.5	Nutshell classes	461
22.1.6	Language support	462
22.1.7	Dynamic base objects with forwarding semantics	462
22.1.8	Caller encapsulation	464
22.2	The environment – selecting target frameworks	467
22.3	The tools – selecting programming languages	467

23 Component distribution and acquisition	469
23.1 Building what sells – applications not components?	469
23.2 Product cataloging and description	470
23.3 Component location and selection	471
23.4 Superdistribution	472
23.5 Intermediaries	473
24 Component assembly	475
24.1 Systematic initialization and wiring	475
24.2 Visual component assembly	476
24.3 Compound documents to supersede visual assembly	476
24.4 Components beyond graphical user interface environments	477
24.5 Managed and “self-guided” component assembly	478
24.6 End-user assembly	478
24.7 Component evolution	479
25 On the horizon	481
25.1 Advanced object composition	481
25.1.1 Delegation	481
25.1.2 Split objects	482
25.1.3 Environmental acquisition	483
25.1.4 Dynamic inheritance	483
25.2 New forms of object and component abstraction	483
25.2.1 Subject-oriented programming	483
25.2.2 Aspect-oriented programming	484
25.2.3 XML components	485
PART FIVE Markets and components	487
26 Gamut of markets	489
26.1 Components	489
26.2 Component platforms and infrastructure	490
26.3 Tools	490
26.3.1 Component design and implementation tools	490
26.3.2 Component testing tools	491
26.3.3 Component assembly tools	491
26.3.4 Component system diagnosis and maintenance	492
26.4 Professional services	492
26.4.1 Component system and framework architects	492
26.4.2 Component assembly consultants	493
26.4.3 Component configuration management	493
26.4.4 Component warehouses, marketing, and consulting	494
26.4.5 Component operators, web services, application service providers	494
27 New professions	495
27.1 Component system architect	495
27.2 Component framework architect	496
27.3 Component developer	497
27.4 Component assembler	497

28 A component marketing paradox	499
28.1 Branding	500
28.2 Pay per use	500
28.3 Co-placement of advertisements	503
28.4 Leveraging on newly created markets	504
28.5 Leverage of integrative forces	505
Epilogue	507
Appendix A Java versus C# versus Component Pascal	509
<i>Useful addresses and bibliography</i>	515
<i>Glossary</i>	543
<i>Index</i>	571

Trademark notice

AppleScript, Cyberdog, HyperCard, Macintosh, Mac OS, NeXT, OpenStep, QuickTime and SANE are trademarks of Apple Computer, Inc., registered in US and other countries.

Tuxedo and WebLogic are registered trademarks of BEA Systems, Inc.

Borland, the Borland Logo, Delphi™, C++Builder™, Borland® VisiBroker® - RT are trademarks or registered trademarks of Borland Software Corporation in the United States and other countries.

Jbed is a registered trademark of esmertec, inc.

CBToolkit, CBConnector, ComponentBroker, DSOM, PowerPC®, REXX, SOM®, VisualAge® and WebSphere® are trademarks of International Business Machines in the United States, other countries, or both.

Orbix, OrbixCOMet Desktop and OrbixWeb are trademarks of IONA.

LEGO® is a trademark of the LEGO Group.

Netscape and the Netscape N and Ship's Wheel logos are registered trademarks of Netscape Communications Corporation in the US and other countries. Netscape Communicator and Netscape Navigator are also trademarks of Netscape Communications Corporation and may be registered outside the US.

Authenticode®, ActiveX®, Visual C#™, COM, COM+, DCOM, OLE, EXCEL®, Internet Explorer, Microsoft® Office, Word®, Microsoft®.NET™, PowerPoint®, Visual Basic®, Visual C++®, Visual J++®, Visual Studio® and Windows® are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

BlackBox, Component Pascal, Direct-To-COM and Safer OLE are trademarks of Oberon Microsystems, Inc.

CORBA®, CWM™, IIOP®, MOF™, OMA, OMG Interface Definition Language (IDL)™, UML™ and XMI® are either registered trademarks or trademarks of the Object Management Group, Inc. in the United States and/or other countries.

X/Open® and OSF/1® are registered trademarks of The Open Group in the US and other countries.

R/3® is a registered trademark of SAP AG in Germany and in several other countries all over the world.

Sun, Sun Microsystems, the Sun Logo, EJB™, Enterprise JavaBeans™, J2EE™, Java™, JavaBeans™, Java Naming and Directory Interface™, Java Naming and Discovery Service, Java™ Servlets, JavaMail™, JavaServer Pages™, JDBC™, JNDI™, JSP™, Java™ RMI, and Solaris™ are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

Texas Instruments Composer is a trademark of Texas Instruments.

W3C® and XML are trademarks or registered trademarks of the World Wide Web Consortium, Massachusetts Institute of Technology.