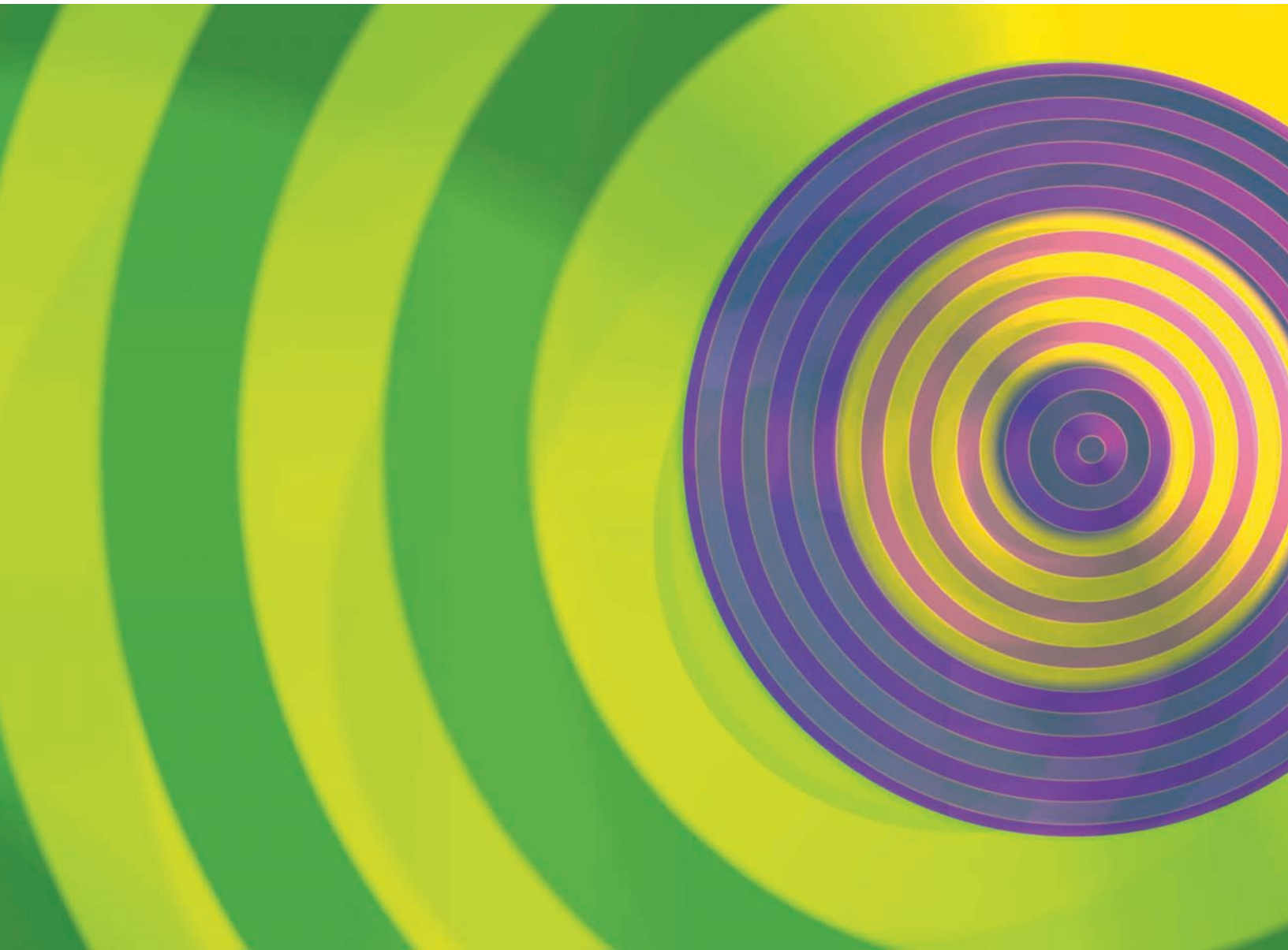


Pearson New International Edition



**Object Oriented Systems Analysis  
and Design**  
**Noushin Ashrafi Hessem Ashrafi**  
**First Edition**

# Pearson New International Edition

---

Object Oriented Systems Analysis  
and Design  
Noushin Ashrafi Hessam Ashrafi  
First Edition

**Pearson Education Limited**

Edinburgh Gate  
Harlow  
Essex CM20 2JE  
England and Associated Companies throughout the world

*Visit us on the World Wide Web at: [www.pearsoned.co.uk](http://www.pearsoned.co.uk)*

© Pearson Education Limited 2014

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 either the prior written permission of the publisher or a licence permitting restricted copying in the United Kingdom issued by the Copyright Licensing Agency Ltd, Saffron House, 6–10 Kirby Street, London EC1N 8TS.

All trademarks used herein are the property of their respective owners. The use of any trademark in this text does not vest in the author or publisher any trademark ownership rights in such trademarks, nor does the use of such trademarks imply any affiliation with or endorsement of this book by such owners.

**PEARSON®**

ISBN 10: 1-292-03960-4  
ISBN 13: 978-1-292-03960-2

**British Library Cataloguing-in-Publication Data**

A catalogue record for this book is available from the British Library

Printed in the United States of America

# Table of Contents

|  |            |
|--|------------|
| <b>References</b>                                    | <b>1</b>   |
| Noushin Ashrafi/Hessam Ashrafi                       |            |
| <b>1. Information Systems</b>                        | <b>7</b>   |
| Noushin Ashrafi/Hessam Ashrafi                       |            |
| <b>2. The Concept of Object Orientation</b>          | <b>36</b>  |
| Noushin Ashrafi/Hessam Ashrafi                       |            |
| <b>3. Methodology</b>                                | <b>61</b>  |
| Noushin Ashrafi/Hessam Ashrafi                       |            |
| <b>4. Gathering Requirements</b>                     | <b>109</b> |
| Noushin Ashrafi/Hessam Ashrafi                       |            |
| <b>5. Domain Analysis</b>                            | <b>153</b> |
| Noushin Ashrafi/Hessam Ashrafi                       |            |
| <b>6. Behavioral Modeling: Use Cases: The Basics</b> | <b>189</b> |
| Noushin Ashrafi/Hessam Ashrafi                       |            |
| <b>7. Behavioral Modeling: Developing Use Cases</b>  | <b>214</b> |
| Noushin Ashrafi/Hessam Ashrafi                       |            |
| <b>8. Structural Modeling</b>                        | <b>257</b> |
| Noushin Ashrafi/Hessam Ashrafi                       |            |
| <b>9. Dynamic Modeling</b>                           | <b>303</b> |
| Noushin Ashrafi/Hessam Ashrafi                       |            |
| <b>10. The Design Challenge</b>                      | <b>345</b> |
| Noushin Ashrafi/Hessam Ashrafi                       |            |
| <b>11. Application Design: Flow &amp; Control</b>    | <b>378</b> |
| Noushin Ashrafi/Hessam Ashrafi                       |            |
| <b>12. Application Design: The User Interface</b>    | <b>411</b> |
| Noushin Ashrafi/Hessam Ashrafi                       |            |

|   |            |
|---|------------|
| <b>13. Application Design: Database &amp; Persistence</b> |            |
| Noushin Ashrafi/Hessam Ashrafi                            | <b>453</b> |
| <b>14. Patterns</b>                                       |            |
| Noushin Ashrafi/Hessam Ashrafi                            | <b>508</b> |
| <b>15. Components &amp; Reuse</b>                         |            |
| Noushin Ashrafi/Hessam Ashrafi                            | <b>529</b> |
| <b>16. Architecture</b>                                   |            |
| Noushin Ashrafi/Hessam Ashrafi                            | <b>556</b> |
| <b>17. Implementation</b>                                 |            |
| Noushin Ashrafi/Hessam Ashrafi                            | <b>596</b> |
| <b>Index</b>  | <b>621</b> |

# REFERENCES

From References of *Object-Oriented Systems Analysis and Design*, First Edition. Noushin Ashrafi, Hessam Ashrafi. Copyright © 2009 by Pearson Education, Inc. Published by Pearson Prentice Hall. All rights reserved.

# REFERENCES

---

- [**Alexander 1977**] Alexander, Christopher, Sara Ishikawa, and Murray Silverstein, 1977. *Pattern Language: Towns, Buildings, Construction*. New York: Oxford University Press.
- [**Alhir 1998**] Alhir, Sinan Si, 1998. *UML in a Nutshell*. Sebastopol, CA: O'Reilly.
- [**American Heritage 1996**] *The American Heritage® Dictionary of the English Language*, Third Edition, 1996. Houghton Mifflin Company.
- [**Arlow 2004**] Arlow, Jim, and Ila Neustadt, 2004. *Enterprise Patterns and MDA: Building Better Software with Archetype Patterns and UML*. Boston: Addison-Wesley.
- [**Armour 2001**] Armour, Frank, and Granville Miller, 2001. *Advanced Use Case Modeling: Software Systems*. Boston: Addison-Wesley.
- [**Arrington 2003**] Arrington, C. T., and Sayed H. Rayhan, 2003. *Enterprise Java with UML*. New York: Wiley.
- [**Ashrafi 1995**] Ashrafi, Noushin, Hessam Ashrafi, and Jean-Pierre Kuilboer, "ISO-9000-3: Guidelines for Software Quality," *Information Systems Management*, Summer 1995.
- [**Avison 2003**] Avison, David E., and G. Fitzgerald, "Where Now for Development Methodologies?" *Communications of the ACM*, Vol. 46, No. 1 (January 2003).
- [**Avison 1995**] Avison, David E., and G. Fitzgerald, 1995. *Information Systems Development: Methodologies, Techniques and Tools, 2nd Edition*. London: McGraw-Hill Book Company.
- [**Awalt 2004**] Awalt, Don, and Rick McUmber, "Secrets of Great Architects," *Microsoft Architects Journal*, Vol. 3 (July 2004).
- [**Bahrami 1999**] Bahrami, Ali, 1999. *Object-Oriented Systems Development: Using the Unified Modeling Language*. Boston: Irwin McGraw-Hill.
- [**Basiura 2001**] Basiura, Russ, Mike Batongbacal, Brandon Bphling, Mike Clark, Andreas Eide, Robert Eisenberg, Brian Loesgen, Christopher L. Miller, Matthew Reynold, Bill Sempf, and Srinivasa Sivakumar, 2001. *Professional ASP.NET Web Services*. Birmingham, UK: Wrox Press.
- [**Bass 2003**] Bass, Len, Paul Clements, and Rick Kazman, 2003. *Software Architecture in Practice, Second Edition*. Boston: Addison-Wesley.
- [**Beck 2000**] Beck, Kent, 2000. *Extreme Programming Explained*. Boston: Addison-Wesley.
- [**Bennet 2002**] Bennett, Simon, Steve McRobb, and Ray Farmer, 2002. *Object-Oriented Systems Analysis and Design Using UML*. Englewood Cliffs, NJ: Prentice Hall.
- [**Bittner 2002**] Bittner, Kurt, and Ian Spence, 2002. *Use Case Modeling*. Boston: Addison-Wesley.
- [**Blum 1992**] Blum, Bruce I., 1992. *Software Engineering: A Holistic View*. New York: Oxford University Press.
- [**Booch 1999**] Booch, Grady, James Rumbaugh, and Ivar Jacobson, 1999. *The Unified Modeling Language User Guide*. Boston: Addison-Wesley.
- [**Booch 1994**] Booch, Grady, 1994. *Object-Oriented Analysis and Design with Applications, Second Edition*. Redwood City, CA: The Benjamin/Cummings Publishing Company, Inc.
- [**Brooks 1995**] Brooks, Frederick P., Jr., 1995. *The Mythical Man-Month: Essays on Software Engineering*. Reading, MA: Addison-Wesley Publishing Company.
- [**Brookshear 2003**] Brookshear, J. Glenn, 2003. *Computer Science: An Overview: 7th Edition*. Reading, MA: Addison-Wesley Publishing Company.
- [**Brown 2001**] Brown, Kyle, Gary Craig, Greg Hester, Jaime Niswonger, David Pitt, and Russell Stinehour, 2001. *Enterprise Java Programming with IBM WebSphere*. Upper Saddle River, NJ: Pearson Education.
- [**Brown 1998**] Brown, William H., Raphael C. Malveau, Hays W. "Skip" McCormick III, and Thomas J. Mowbray, 1998. *Anti-Patterns: Refactoring Software, Architectures, and Projects in Crisis*. New York: Wiley.
- [**Campbell-Kelly 2003**] Campbell-Kelly, Martin, 2003. *From Airline Reservations to Sonic the Hedgehog: A History of the Software Industry*. Cambridge, MA: The MIT Press.
- [**Chapin 2002**] Donald Chapin, "What's the Business in Business Rules?" *Business Rules Journal*, Vol. 3, No. 10 (October 2002). URL: <http://www.BRCommunity.com/a2002/b119.html>.
- [**Chessman 2001**] Chessman, John, and John Daniels, 2001. *UML Components: A Simple Process for Specifying Component-Based Software*, 2001. Boston: Addison-Wesley.
- [**Chiera 1938**] Chiera, Edward, 1938. *They Wrote on Clay*. Chicago: The University of Chicago Press. (Copyright Renewed 1966)

- [Chisholm 2002] Malcolm Chisholm, "The Black Box Problem," *Business Rules Journal*, Vol. 3, No. 3 (March 2002). URL: <http://www.BRCommunity.com/a2002/b100.html>.
- [Chonoles 2003] Chonoles, Michael Jesse, and James A. Schardt, 2003. *UML 2 for Dummies*. Indianapolis, IN: Wiley Publishing, Inc.
- [Clark 2001] Clark, Andy, 2001. *Mindware: An Introduction to the Philosophy of Cognitive Science*. New York: Oxford University Press.
- [Coad 1997] Coad, Peter, Mark Mayfield, and David North, 1997. *Object Models: Strategies, Patterns, and Applications, 2nd Edition*. Upper Saddle River, NJ: Prentice Hall.
- [Coad 1992] Coad, Peter, "Object-Oriented Patterns," *Communications of the ACM*, Vol. 35, No. 9 (September 1992).
- [Cockburn 2002] Cockburn, Alistair, 2002. *Agile Software Development*. Boston: Addison-Wesley.
- [Cockburn 2001] Cockburn, Alistair, 2001. *Writing Effective Use Cases*. Boston: Addison-Wesley.
- [Codd 1979] Codd, E. F., "Extending the Data Base Relational Model to Capture More Meaning," *ACM Transactions on Database Systems*, Vol. 4, No. 4 (December 1979), 397–434.
- [Codd 1970] Codd, E. F., "A Relational Model of Data for Large Shared Data Banks," *Communications of ACM*, Vol. 13, No. 6 (June 1970), 377–387.
- [Conallen 2003] Conallen, Jim, 2003. *Building Web Applications with UML, Second Edition*. Boston: Addison-Wesley.
- [Cook 1996] Cook, Melissa, 1996. *Building Enterprise Information Architecture: Reengineering Information Systems*. Upper Saddle River, NJ: Prentice Hall.
- [Cooper 1995] Cooper, Alan, 1995. *About Face: The Essentials of User Interface Design*. Foster City, CA: IDG Books.
- [Cooper 2003] Cooper, James W., 2003. *C# Design Patterns: A Tutorial*. Boston: Addison-Wesley.
- [Coplien 1997] Coplien, James O., 1997. "Domain Analysis and Patterns," Bell Labs. URL: <http://users.rcn.com/jcoplien/oopsla/OopslaDomainPatterns-1.html>.
- [Crnkovic 2002] Crnkovic, Ivica, Brahim Hnich, Torsten Jonsson, and Zeynep Kiziltan, "Specification, Implementation, and Deployment of Components," *Communications of ACM*, Vol. 45, No. 10 (October 2002).
- [D'Souza 1999] D'Souza, Desmond Francis, and Cameron Wills, 1999. *Objects, Components, and Frameworks with UML: The Catalysis Approach*. Boston: Addison-Wesley.
- [Dutka 1989] Dutka, Alan F., and Howard H. Hanson, 1989. *Fundamentals of Data Normalization*. Reading, MA: Addison-Wesley.
- [Evitts 2000] Evitts, Paul, 2000. *A UML Pattern Language*. Indianapolis: Macmillan Technical Publishing (MTP).
- [Faison 2002] Faison, Ted, 2002. *Component-Based Development with Visual C#*. New York: M&T Books (Wily).
- [Fielding 2000] Fielding, Roy Thomas, 2000. *Architectural Styles and the Design of Network-Based Software Architecture*. Doctoral Dissertation. University of California, Irvin. URL: <http://www.ics.uci.edu/~fielding/pubs/dissertation>.
- [Fowler 2003] Fowler, Martin, 2003. *Patterns of Enterprise Application Architecture*. Boston: Addison-Wesley.
- [Fowler 2000a] Fowler, Martin, with Kendall Scott, 2000. *UML Distilled, Second Edition: A Brief Guide to the Standard Modeling Language*. Boston: Addison-Wesley.
- [Fowler 2000b] Fowler, Martin, 2000. *Refactoring: Improving the Design of Existing Code*. Boston: Addison-Wesley.
- [Fowler 1997] Fowler, Martin, 1997. *Analysis Pattern: Reusable Object Models*. Boston: Addison-Wesley.
- [Freeman 2004] Freeman, Eric, and Elisabeth Freeman with Kathy Sierra and Bert Bates, 2004. *Head First Design Patterns*. Sebastopol, CA: O'Reilly.
- [Gamma 1997] Gamma, Erich, Richard Helm, Ralph Johnson, and John Vlissides, 1997. *Design Patterns: Elements of Reusable Object Oriented Software*. Reading, MA: Addison-Wesley.
- [Glass 2002] Glass, Robert L., 2002. *Facts and Fallacies of Software Engineering*. Boston: Addison-Wesley.
- [Goldfedder 2002] Goldfedder, Brandon, 2002. *The Joy of Patterns: Using Patterns for Enterprise Development*. Boston: Addison-Wesley.
- [Graham 2001] Graham, Ian, Alan O'Callaghan, and Alan Cameron Wills, 2001. *Object-Oriented Methods: Principles & Practice, Third Edition*. Boston: Addison-Wesley.
- [Groff 1999] Groff, James R., and Paul N. Weinberg, 1999. *SQL: The Complete Reference*. Berkeley: Osborne/McGraw-Hill.
- [Haggerty 2000] Neville Haggerty, 2000. "Modeling Business Rules Using the UML and CASE," Business Rules Community. URL: <http://www.brcommunity.com/cgi-bin/x.pl/features/b016.html>.
- [Halpin 2001] Halpin, Terry, 2001. *Information Modeling and Relational Databases: From Conceptual Analysis to*



- Logical Design, Using ORM with ER and UML*. San Francisco: Morgan Kaufmann Publishers.
- [Harmon 2000] Harmon, Paul, Michael Rosen, and Michael Guttman, 2000. *Developing E-Business Systems and Architectures: A Manager's Guide*. San Francisco, CA: Morgan Kaufmann.
- [Hilliard 1999] Hilliard, Rich, 1999. "Using the UML for Architectural Description," *UML 99: Proceedings of Second International Conference on the Unified Modeling Language*. New York: Springer-Verlag.
- [Jackson 2001] Jackson, Michael, 2001. *Problem Frames: Analyzing and Structuring Software Development Problems*. Harlow, England: Addison-Wesley.
- [Jackson 1995] Jackson, Michael, 1995. *Software Requirements and Specifications: A Lexicon of Practice, Principles and Prejudices*. Harlow, England: Addison-Wesley.
- [Jacobson 1999] Jacobson, Ivar, Grady Booch, and James Rumbaugh, 1999. *The Unified Software Development Process*. Boston: Addison-Wesley.
- [Jacobson 1995] Jacobson, Ivar, Maria Ericsson, and Agneta Jacobson, 1995. *The Object Advantage: Business Process Reengineering With Object Technology*. Wokingham, England: Addison-Wesley.
- [Jacobson 1992] Jacobson, Ivar, Magnus Christerson, Patrik Jonsson, and Gunnar Övergaard, 1992. *Object-Oriented Software Engineering: A Use Case Driven Approach*. Harlow, England: Addison-Wesley.
- [Johnson 2000] Johnson, Jeff, 2000. *GUI Bloopers: Don'ts and Do's for Software Developers and Web Designers*. San Francisco: Morgan Kaufmann Publishers.
- [Kean 2001] Kean, Liz, 2001. "Domain Engineering and Domain Analysis," Carnegie Mellon University: The Software Engineering Institute (SEI). URL: [http://www.sei.cmu.edu/str/descriptions/deda\\_body.html](http://www.sei.cmu.edu/str/descriptions/deda_body.html).
- [Kendall 1999] Kendall, Kenneth E., and Julie E. Kendall, 1999. *Systems Analysis and Design*, Fourth Edition. Upper Saddle River, NJ: Prentice Hall.
- [Kleppe 2003] Kleppe, Anneke, Jos Warmer, and Wim Bast, 2003. *MDA Explained, The Model Driven Architecture: Practice and Promise*. Boston: Addison-Wesley.
- [Kulak 2000] Kulak, Daryl, and Eamonn Guiney, 2000. *Use Cases: Requirements in Context*. New York: ACM Press.
- [Landay 1996] Landay, James A., and Brad A. Myers. "Sketching Storyboards to Illustrate Interface Behaviors," *Electrical Engineering and Computer Sciences, University of California at Berkeley*. URL: [http://www.cs.berkeley.edu/~landay/research/publications/CHI96/short\\_storyboard.html](http://www.cs.berkeley.edu/~landay/research/publications/CHI96/short_storyboard.html).
- [Leffingwell 2000] Leffingwell, Dean, and Don Widrig, 2000. *Managing Software Requirements: A Unified Approach*. Boston: Addison-Wesley.
- [Lin 2002] Nelson Lin, "Alternatives for Rule-based Application Development," *Business Rules Journal*, Vol. 3, No. 10 (October 2002). URL: <http://www.BRCommunity.com/a2002/n007.html>.
- [MacDonald 2002] MacDonald, Mathew, 2002. *User Interfaces in C#: Windows Forms and Custom Controls*. New York: APress (Springer-Verlag, New York, Inc.).
- [Malveau 2001] Malveau, Raphael, and Thomas J. Mowbray, 2001. *Software Architect Bootcamp*. Upper Saddle River, NJ: Prentice Hall.
- [Martin 1996] Martin, James, and James J. Odell, 1996. *Object-Oriented Methods: Pragmatic Considerations*. Upper Saddle River, NJ: Prentice Hall.
- [Martin 1995] Martin, James and James J. Odell, 1995. *Object-Oriented Methods: A Foundation*. Upper Saddle River, NJ: Prentice Hall.
- [Mayhew 1992] Mayhew, Deborah, J., 1992. *Principles and Guidelines in Software User Interface*. Englewood Cliffs, NJ: Prentice Hall.
- [McBreen 2002] McBreen, Pete, 2002. *Software Craftsmanship: The New Imperative*. Boston: Addison-Wesley.
- [McClure 1997] McClure, Carma, 1997. *Software Reuse Techniques: Adding Reuse to the System Development Process*. Upper Saddle River, NJ: Prentice Hall.
- [McConnel 1996] McConnel, Steve, 1996. *Rapid Development: Taming Wild Software Schedules*. Redmond, WA: Microsoft Press.
- [Muller 1999] Muller, Robert J., 1999. *Database Design for Smarties: Using UML for Data Modeling*. San Francisco: Morgan Kaufmann Publishers, Inc.
- [Muller 1997] Muller, Pierre-Alain, 1997. *Instant UML*. Birmingham, UK: Wrox Press.
- [Naiburg 2001] Naiburg, Eric J., and Robert A. MaksimChuk, 2001. *UML for Database Design*. Boston: Addison-Wesley.
- [Neighbors 1981] J. Neighbors, 1981. *Software Construction Using Components*. Ph.D. Thesis. Irvine: Department of Information and Computer Science, University of California.
- [Nock 2004] Nock, Clifton, 2004. *Data Access Patterns: Database Interactions in Object-Oriented Applications*. Boston: Addison-Wesley.
- [Norman 2005] Norman, Jeremy M. 2005. *From Gutenberg to the Internet: A Sourcebook on the History of Information Technology*. Novato, CA: Historyofscience.com.

## References

- [Norman 2004] Norman, Donald A. 2004. *Emotional Design: Why We Love (or Hate) Everyday Things*. New York: Basic Books.
- [Norman 2002] Norman, Donald A. 2002. *The Design of Everyday Things*. New York: Basic Books
- [Ommering, 2002] Ommering, Rob van, 2002. "Building Product Populations with Software Components," *International Conference on Software Engineering: Proceedings of the 24th International Conference on Software Engineering*. ACM, 2002. URL: <http://portal.acm.org>.
- [Orfali, 1997] Orfali, Robert, and Dan Harkey, 1997. *Client/Server Programming with Java and CORBA*. New York: John Wiley & Sons, Inc.
- [O'Rourke 2003] O'Rourke, Carol, Neal Fishman, and Warren Selkow, 2003. *Enterprise Architecture Using the Zachman Framework*. Boston: Course Technology.
- [Parnas 2001] Parnas, David L., 2001. *Software Fundamentals: Collected Papers by David L. Parnas*. Boston: Addison-Wesley.
- [Petroski 2003] Petroski, Henry, 2003. *Why There Is No Perfect Design*. New York: Vintage Books.
- [Petroski 1985] Petroski, Henry, 1985. *To Engineer Is Human: The Role of Failure in Successful Design*. New York: St. Martin Press.
- [Prieto-Diaz 1990] Prieto-Diaz, Rubén, "Domain Analysis: An Introduction," *Software Engineering Notes*, 15-2, April 1990. URL: <http://www.cs.jmu.edu>.
- [Ravichandran 2003] Ravichandran, T, and Marcus A. Rothenberger, "Software Reuse Strategies And Component Markets," *Communications of the ACM*, Vol. 46, No. 8. (August 2003).
- [Reeder 2001] Judi Reeder, 2001. "Templates for Capturing Business Rules," Business Rules Community. URL: <http://www.brcommunity.com/cgi-bin/x.pl/features/b056.html>.
- [Robertson 1999] Robertson, Suzanne, and James Robertson, 1999. *Mastering the Requirements Process*. Harlow, England: Addison-Wesley.
- [Rosenberg 1999] Rosenberg, Doug, and Kendall Scott, 1999. *Use Case Driven Object Modeling with UML: A Practical Approach*. Boston: Addison-Wesley.
- [Royce 1998] Royce, Walker, 1998. *Software Project Management: A Unified Framework*. Boston: Addison-Wesley.
- [Schneider 2001] Schneider, Geri, and Jason P. Winters, 2001. *Applying Use Cases, Second Edition: A Practical Guide*. Boston: Addison-Wesley.
- [Shneiderman 1986] Shneiderman, B., 1986. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Reading, MA: Addison-Wesley.
- [Shoemaker 2004] Shoemaker, Martin L., 2004. *UML Applied: A .Net Perspective*. New York: APress (Springer-Verlag, New York, Inc.)
- [Smith 1986] Smith, Sidney L., and Jane Mosier, 1986. "Guidelines for Designing User Interface Software," Report ESD-TR-86-278, Electronic System Division, MITRE Corporation, Bedford, MA. URL: <http://hcibib.org/sam/>.
- [Sparks 2001] Sparks, Geoffrey, 2001. "Database Modelling in UML," *Methods & Tools e-newsletter*. URL: <http://www.martinig.ch/mt/index.html>.
- [Spolsky 2001] Spolsky, Joel, 2001. *User Interface Design for Programmers*. New York: APress (Springer-Verlag, New York, Inc.).
- [Starr 2004] Starr, Paul, 2004. *The Creation of the Media: Political Origins of Modern Communications*. New York: Basic Books.
- [Stephens 2003] Stephens, Matt, and Doug Rosenberg, 2003. *Extreme Programming Refactored: The Case Against XP*. New York: APress (Springer-Verlag, New York, Inc.).
- [Strum 1999] Strum, Jake, 1999. *VB6 UML: Design and Development*. Birmingham, UK: Wrox Press Ltd.
- [Szyperski 2002] Szyperski, Clemens, 2002. *Component Software: Beyond Object-Oriented Programming, Second Edition*. Boston: Addison-Wesley.
- [Taylor 1998] Taylor, David A., 1998. *Object Technology, A Manager's Guide: Second Edition*. Boston: Addison-Wesley.
- [Tenner 1996] Tenner, Edward, 1996. *Why Things Bite Back*. New York: Vintage Books.
- [Togaf 2002] Open Group Architecture Forum (TOGAF), 2002. "Developing Architecture View," The Open Group. URL: <http://www.opengroup.org/architecture>.
- [Urman 2002] Urman, Scott, 2002. *Oracle 9i PL/SQL Programming*. New York: Oracle Press, McGraw-Hill/Osborne.
- [Van Gigch 1991] Van Gigch, John P., 1991. *System Design Modeling and Metamodeling (The Language of Science)*. New York: Plenum Pub Corp.
- [Van Slyke 2003] Van Slyke, Craig , and France Bélanger, 2003. *E-Business Technologies: Supporting the Net-Enhanced Organization*. New York: John Wiley & Sons, Inc.
- [Viera 2000] Viera, Robert, 2000. *Professional SQL Server Programming*. Birmingham, UK: Wrox Press Ltd.
- [Vitharana 2003] Vitharana, Padmal, "Risks and Challenges of Component-Based Software Development," *Communications of the ACM*, Vol. 46, No. 8 (August 2003).

## References

- [Wampler 2002]** Wampler, B. E., 2002. *The Essence of Object-Oriented Programming with JAVA and UML*. Boston: Addison-Wesley.
- [Webster 1995]** Webster, Bruce F., 1995. *Pitfalls of Object-Oriented Development*. New York: M&T Books.
- [Weisfeld 2000]** Weisfeld, Matt, 2000. *The Object-Oriented Thought Process: The Authoritative Solution*. Indianapolis, IN: Sams Publishing.
- [White 1997]** White, S. A., and C. Lemus, 1997. "The Software Architecture Process," University of Houston-Clear Lake. URL: <http://nas.cl.uh.edu/whites/webpapers.dir/ETCE97pap.pdf>.
- [Wiegiers 2003]** Karl E. Wiegiers, 2003. *Managing Software Requirements*. Redmond, WA: Microsoft Press.
- [Wirfs-Brock 2003]** Wirfs-Brock, Rebecca, and Alan McKean, 2003. *Object Design: Roles, Responsibilities, and Collaborations*. Boston: Addison-Wesley.
- [Wysocki 2000]** Wysocki, Robert K., Robert Beck Jr., and David B. Crane, 2000. *Effective Project Management, Second Edition*. New York: John Wiley & Sons, Inc.
- [Yourdon 1994]** Yourdon, Edward, 1994. *Object-Oriented System Design: An Integrated Approach*. Upper Saddle River, NJ: Prentice Hall.
- [Zachman 1987]** Zachman, J. A., "A Framework for Information System Architecture," *IBM Systems Journal*, Vol. 26, No. 3 (1987).

## References

# Information Systems

## 1. OVERVIEW

---

Information systems are systems that process data into information. We can view an information system from various perspectives: its goals, its processes or its components, that is, applications, information technology, people, and procedures.

Information systems are also products, and like other products they must satisfy their consumers and be developed by following a methodology that ensures the best possible quality and the best possible use of resources.

### Chapter Topics

- Information and its components.
- System and its components.
- An overview of information systems.
- An introduction to information technology.
- The core building blocks of information technology.
- The concept of “application.”
- Information systems as products.
- The business of developing information system products.
- Information system as the infrastructure of the business.
- The enterprise of software development.

### Introduction

To develop a modern information system, we must start with a clear understanding that an information system is primarily a *commercial product*. All products such as cars, houses, or computers might be built to satisfy demands or wishes that fall outside the domain of the marketplace, but it is the marketplace that defines their center of gravity and shapes their overall features.

From Chapter 1 of *Object-Oriented Systems Analysis and Design*, First Edition. Noushin Ashrafi, Hessam Ashrafi. Copyright © 2009 by Pearson Education, Inc. Published by Pearson Prentice Hall. All rights reserved.

Furthermore, we cannot arrive at an effective understanding of information systems unless we comprehend their **components**—what they do, how they relate to each other, and how they work together. Developing information systems and software applications involves highly *abstract* concepts that have very *concrete* outcomes and sometimes very serious consequences.

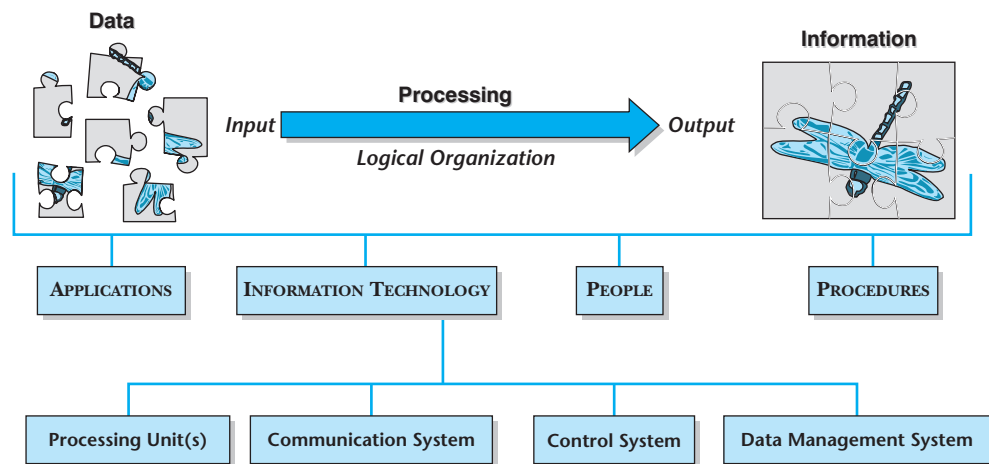
The term “information system” is relatively recent, but the concept is as old as history. As many historians have concluded, writing appeared with the need for accounting, the first application of information systems. Accounting has always exemplified the everlasting components of the information systems: data as *input*, statements and balances as *output*, arithmetic as the application of **logic** (processing), some sort of *storage* or data management system, **security** to prevent unauthorized access, and *communication* through oral and written symbols.

**Information automation**, however, has changed both the reach and role of information systems within the human civilization. Information automation means that a nonhuman device can apply information logic to data through a set of stored instructions or a “program.” This, in turn, means that information processing can become more capital-intensive and less labor-intensive. Another consequence has been the increasing *commoditization* of information systems. By packaging information logic into software, automation has allowed information systems and applications to become *market* products.

All commercial products have three basic traits in common: ❶ they must satisfy certain requirements or take advantage of opportunities, ❷ they are human artifacts and, therefore, must be built, and ❸ their development must follow a methodology that helps to lower costs, raise quality, and make success more likely.

To say, however, that the development of a product must follow a methodology is not to say all products must be developed following the *same* methodology. Even with the same product, methodology changes with technology, experience, theory, scale, and context.

**Figure 1**  
Information System and Its Components



An information system can be viewed from two angles: what requirements it must satisfy and how it satisfies those requirements.

## 2. INTRODUCING INFORMATION SYSTEMS

A concise but separate review of both “information” and “system” is necessary before we can effectively introduce information systems. The reach of both terms is very wide, but each one constraints the other: An information system is *not* concerned with every kind of information, but only those that can be obtained by processing data through a system; it is *not* any system, but an open system that accepts input, produces output, and has understandable logic.

Any information system requires an information technology, but the information technology is not the same as the information system. The latter is not one technology, but an interrelated and often rapidly changing *collection* of technologies and subsystems.

### Information

Information is an organized collection of data that allows its recipient to ① gain knowledge, ② draw meaning, ③ arrive at conclusions, or ④ execute a set of actions to reach an objective.

From this definition, it follows that the term “information” covers a very vast ground:

☞ A broad definition of “information” would also include works of art and entertainment—novels, poems, movies, etc.—even though their data are not “real.” These works, however, are outside the scope of this text, which is information systems, not information in general.

- A news report in the paper, on the radio, or on TV.
- The itinerary of your upcoming trip.
- The year-end balance sheet of a company.
- A business report.
- A fire alarm.
- A bank statement.
- A book on system analysis and design.

These items are all information—regardless of whether the data consists of words, sounds, numbers, images, or other symbols. Some, like alarms and traffic signs, are designed to communicate their intentions immediately, with a minimum amount of data and a minimal need for interpretation. At the other end of the spectrum, news reports, books, and documentaries paint a complex picture by offering a large amount of data within a narrative composed of numerous logical packages (sentences, paragraphs, pictures, dialogues, and scenes).

☝ Any information also relies on a **cultural context** to be meaningful.

In practice, how we characterize “information” depends on our judgment and on our expectations. However, regardless of how we designate it, any information has three main constituents: **data, purpose, and logical organization.**

### Data

Data are the building blocks of information.

☞ Properly speaking, *data* is the plural form of Latin *datum* and many insist that the separation between the plural and the singular should be maintained. But, for better or for worse, it seems that “data” will play both roles in the English language. Therefore, it is the verb that must distinguish between “data” as a single item and “data” as plural.

The original meaning of *datum* in English was “assumption.” And “fact,” at some point in the past, meant “crime” (a meaning which still applies to “accessory after the fact.”)

Data are the facts *or* assumptions that are structured within a logical framework to convey information. In other words, data are the *raw material* for information:

| Data   | Information       |
|--|-------------------|
| Deposits, withdrawals, interest, and service charges for a certain month, plus the forwarding balance from the previous month. | Bank Statement    |
| Moving images, dialog, music, and commentary.  | Television Report |
| Titles, subtitles, words, paragraphs, quotations, and pictures.  | Newspaper Report  |
| The red outline of a circle bisected by a red line.  | No Entry!         |
| Weigh, height, cholesterol, sugar level, age, symptoms, etc.   | Patient Profile   |

It is often said that data are meaningless by themselves until they are turned into information. (Hence the modifier “raw,” which is frequently added to “data.”) This is true in many cases, especially when the data consists of numbers, but the relationship between data and information is usually multileveled and subtle: A deposit to your bank account is meaningful by itself, even though it must be placed in the context of other data to present the monthly activity and the state of your bank account. (See *Data Hierarchy* later in this chapter.)

“Unfounded” is the usual term for the information that lacks data or is based on wrong or incomplete assumptions.

### Goal

Information has an objective.

Any information must have a *purpose*, a meaning that it wants to impart or a goal that it wants to achieve. Some goals are simple, some are ambitious, and some are open to multiple interpretations:

☞ A logical conclusion (or information) would prove incorrect if the data are wrong or unrelated. **Logic is a matter of form, not of content.** In other

| Information               | Purpose   |
|---------------------------|---|
| Bank Statement            | Reports how much money you had in your account at the beginning of the month, the amounts that you deposited or withdrew during the month, bank charges, and how much you have now. |
| Television Report         | Communicates (or tries to communicate) the what, when, where, how, and (perhaps) why of an event to its audience.   |
| Year-End Corporate Report | Attempts to tell shareholders (or anybody else who might be interested) how well the corporation did in the previous year (and why) and what to expect for the next year.           |

words, the form must be correct but its correctness does not ensure that the conclusions are true. Some logical fallacies can be detected easily, while others are more difficult to identify. (Understanding logic is very important to anybody who wants to work with information systems. But, alas, this text is not about logic.)

If the information lacks an understandable purpose, we call it “pointless,” “rambling,” or a similar term.

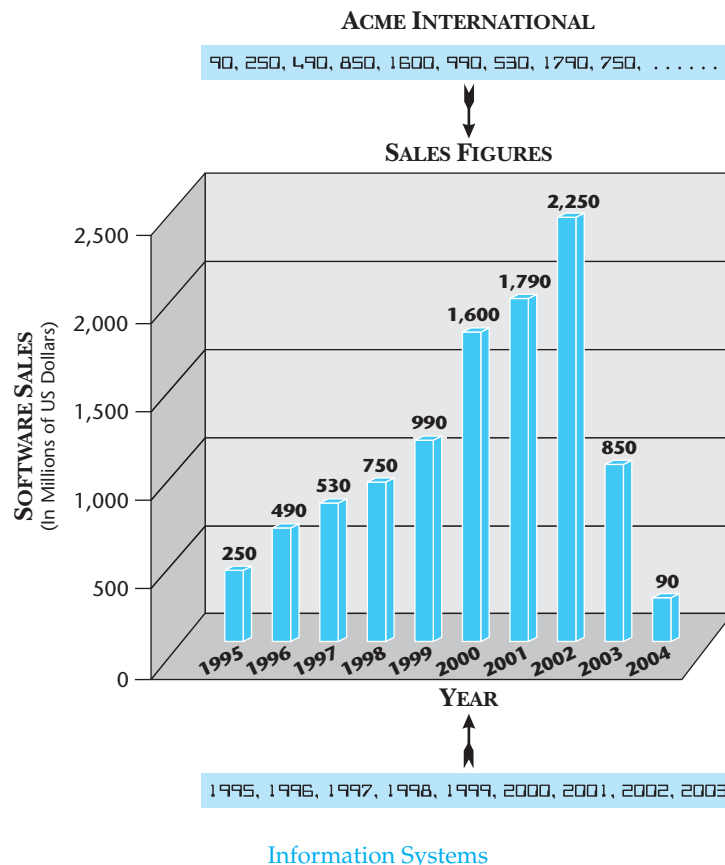
### Logic

The objective of information is achieved through logic.

To arrive from data to meaning, information must follow a logic—simple or complex, apparent or not readily apparent. (“Nonsense” is the term we apply to information if it lacks logic, and “misleading” or “sophistry” is applied if it follows an unacceptable logic.)

The purpose and the logical organization of information may be combined into an infinite number of packages. The only constant is that without either, data does *not* become information.

Figure 2 is an easy-to-understand chart. Acme International enjoyed a rapid growth from 1995 to 2002, but witnessed its fortunes reversed in the 2003–2004 period. Left to themselves, the underlying data would not enlighten us as to the fortunes of the company. By correlating the sales figures (the Y axis) and the years (the X axis) we arrive at certain conclusions (or knowledge).



**Figure 2**  
**From Data to Information**  
Information results from providing data with a logical organization that serves a purpose. In this chart, it is the logical correlation between the years and the sales figures for 10-year periods that allows us to conclude that Acme has fallen on hard times.



Two points are crucial for understanding how data is processed into information:

- ① **Information uses data selectively.** Since the process has a purpose, the data that underlies the information is always selective *because it must relate to the purpose*. For example, our chart does not represent *all* data about Acme International (which is not possible, in any case); nor does it even show *all* sales figures, but only *software* sales and only *yearly* totals for a specific period.
- ② **Information is only as valid as the logic that produces it.** Since the process organizes data based on a certain logic, if the logic is wrong, distorted, or incomplete, the resulting knowledge will be wrong, distorted, or incomplete as well, even if *every* assumption is an undisputable fact. The chart about Acme International suggests a company in trouble, but what if Acme has a thriving *hardware* business and is actually dismantling its software division by selling it off piece by piece? If the answer to this question is affirmative, then the logic of presenting *only* the software sales as a reference to the overall fortunes of Acme is misleading.

## Symbols

Both data and information are expressed as symbols.

The bulk of information that we send or receive is expressed through verbal symbols. But information is by no means limited to verbal messages. (The chart in Figure 2 is a hybrid message: Its verbal and visual elements reinforce each other.) Information can be delivered by sound, pictures, or multimedia as well: traffic signs, smoke signals, the sound of a bugle instructing soldiers what to do, skulls and crossbones (which can mean both “pirate” or “this is a dangerous place: stay clear”).

## Data Hierarchy

The relationship between data and information is hierarchical.

Something that is considered “information” at one level may be used as “data” in a higher level, and vice versa. The distinction depends on the purpose. For instance, the chart about Acme’s software sales can become “data” if we intend to merge it into a fuller report about Acme International or industrial trends between 1995 and 2004. (Each yearly sales figure or column in the chart, in turn, consists of many sales figures *within* each year.)

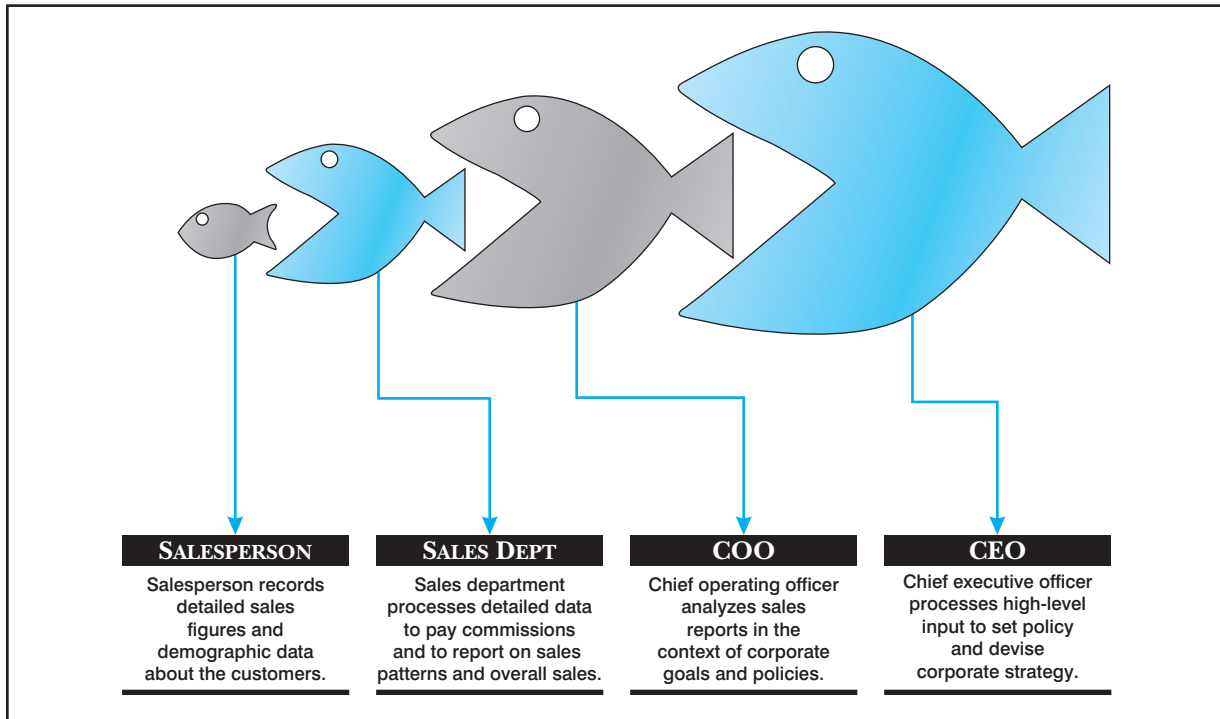
## System

A system is a set of interrelated elements organized into an identifiable whole.

[Van Gigch 1991, 30–31]

The majority of systems, natural or man-made, fulfill a function. Hence, we may also define “system” as a **collection of elements that work together to perform a task**.

**Figure 3 Data Hierarchy: One Person's Information Is Another Person's Data**



Information at one level often becomes input data for a higher level.

The elements may be few or many, they may be very similar or extremely different, the collection may be tightly knit or loosely connected, and the function may be simple or complex. Examples are plentiful:

| Elements  | System             |
|---|--------------------|
| Locomotives, wagons, tunnels, railroads, switches, engineers, conductors, etc.                  | Railroad System    |
| Microprocessor(s), printed circuitry, keyboard, monitor, mouse, operating system, storage, etc. | Computer           |
| Receipts, canceled checks, correspondence, folders, file cabinets, etc.                         | Filing System      |
| Canals, ditches, dams, sprinklers, etc.   | Irrigation System  |
| Organs, such as the lungs, that deliver oxygen to the circulatory system.                       | Respiratory System |

To correctly understand *information* systems, we must achieve a clear understanding of the parts that constitute the definition of *system*: elements, interrelationships, organization, and the “identifiable whole” (or the *distinct identity* of the system).

- ❶ **Elements of the System: A system is not monolithic, but consists of a set of elements.** The constituent elements of a system can be real objects (parts of a watch), virtual objects (characters in a computer game), concepts (words in a sentence), or a combination of all (an automated factory). The constituent elements of a system are also called **components**. A component, in turn, may consist of other elements or components.
- ❷ **Interrelated Elements: The elements making up a system must be both related and interrelated.** An element within a system must interact with at least one other element and the interactions must, directly or indirectly, link all elements. A fruit basket does not qualify as a system even though it consists of a set of related entities packaged as “an identifiable whole”; a class of students, on the other hand, is a system even if no word passes among the students because the teacher provides a focal point of interaction.
- ❸ **Organization: Elements within a system must have a formal structure.** Organized means “functioning with a *formal structure*, as in the coordination and direction of activities.” No formal structure means no system. In other words, we do not consider a random assembly of entities as a “system.”
- ❹ **Identifiable Whole: A system must have a distinct identity.** We have heard, often enough, that “a whole is greater than the sum of its parts.” If we cannot identify a whole that is more than a collection of elements, then, at best, we have identified a *set* of related elements, not a “whole.” Prime numbers (those divisible only by one and themselves) do not constitute a system, but a set.
- ❺ **Subsystems: A subsystem is a system that functions as a component of another system.** An element within a system can be a system in its own right, in which case it is called a **subsystem**. In fact, any system can be a subsystem of a bigger system. The human body consists of several subsystems—respiratory, alimentary, nervous, etc.—while, in turn, a human is a component of the society.
- ❻ **Open and Closed Systems:** The relationship of a system to the outside is identified by its place on a spectrum from open to closed. An perfect open system is one that ❶ accepts input, ❷ the logic of its internal workings can be understood and/or changed, ❸ can change as a result of interaction with the outside world, and ❹ produces output. A closed system is the opposite of an open system. No system, however, is “perfect.”

### Networks vs. Systems

A network is a cooperating set of relatively independent elements.

The terms “network” and “system” are often used interchangeably and, indeed, they overlap in many aspects: Like systems, networks consist of a set of interrelated elements. The differences appear when we move towards stricter definitions of both terms:

- ❶ While the elements within a system *cannot* function the same way if they are taken out of the system, elements within a network are more or less able to function independently. (The circulatory system of the human body is a typical example of the “system,” whereas workstations connected to the Internet are members of a “network.”)
- ❷ A network has a less sharply defined identity as a “whole” than a system.

These differences, however, depend on the nature and the organization of the system or the network in question. In a network of roads, individual elements (roads) are less dependent on each other than the components within a computer system. In addition, elements within a network are not necessarily of equal importance: in a computer network, the failure of a server can effectively halt the operation of the network.

A system or network in which the constituent elements are highly dependent on each other is described as *tightly coupled* (which, of course, is a relative term). The Internet is an example of a *loosely coupled* network in which the failure of a part does not bring the whole “system” to a halt.

In general, the more loosely coupled a system or a network is, the less it is prone to failure. As a result, however, as the coupling is loosened the role of a central authority is diminished—for good or for bad.

☞ As we shall see immediately below, any information system is *vital* dependent on storage and communication systems, and *vice versa*.

☞ Our purpose here is not to split hairs between information systems and information technology. But too many people involved in developing and using information systems are waylaid by the siren of technology. And, too often, technological solutions are offered as information system solutions.

So, if you are told that a company's information system consists of so many servers, such and such networks, etc., do not object to the term: After all, the usage is very widespread, easy to understand, and not entirely wrong as no information system exists *without* an information technology. But keep the question open: What does this system actually *do* (or *should it do*) for the information needs of the company?

## The Information System

An information system is a system that processes data into information according to a predictable set of logical assumptions.

As you would expect, our assertions about information and systems apply to information systems as well, but with qualifications that partly arise from the act of combining the two:

- An information system is an **open system**: It takes input and produces output based on a logic that is comprehensible. A **closed system**—with no definable input, output, and/or logic—is *not* an information system.
- An information system *processes* data (or input) *into* information (or output). A system that manages data is a **data management system** and a system that sends and receives data or information is a **communication system**. The two are components of an information system but neither is the information system itself.
- Since information has a *goal* and is *selective* regarding data, an information system must have goals and be selective as well.
- Information systems are *man-made*. Therefore, information systems share many properties with other human products: They serve a range of social and economic needs; they are built by using available methodologies, processes, and tools; and they are subject to rational and irrational forces (such as theorizing, experience, fashion trends, and prejudices).

Even though numerous sources of information exist, not all can be deemed “information systems.”

The major components of any information system are:

- **Applications** that perform specific tasks.
- **Information technology** that consists of processing and control units, and communication and data management systems.
- **People** who use the system or who provide it with services.
- **Procedures** that decide how the information system is operated and by whom.

## The Information Technology

Information technology is the know-how, methods, tools, and material used to support information systems.

Data and, as a result, information are always *virtual*. Anything virtual needs a *real* vehicle to be sent, received, stored, or manipulated. Therefore, any information system is shaped by, and is dependent on, an information technology or technologies that carry and maintain it. (As we mentioned, information systems and information technology are so intertwined that they are often used interchangeably, and distinguishing them in certain areas is rather difficult.)

As the definition indicates, information technology is not one thing, but a set of interrelated components:

- **Know-how** is the knowledge and the skill required to do something correctly. The technology would go to waste if the developers of a product do not know how to shape and exploit its material base.
- **Method** is a systematic way of accomplishing something. Methods result from both experience and the theoretical interpretation of experience. Methods function as guidelines to people with know-how to accomplish a task. In turn, new experiences must refine existing methods, or else methods lose their relevance.
- **Material** has a complex and varied relation to the technology that uses it. Increasingly, the invention of new material and methods is not left to chance, but is *targeted*. Modern research labs systematically experiment and search for material and solutions that are required by the expressed needs of the business or the marketplace.

Usually, a technology is not used to its full potential when it emerges. The Internet, the foundations of which were laid by DARPA (Defense Advanced Research Projects Agency) in 1973, was more than anything else an imaginative use of *existing* computer technology. In turn, the impact of the Internet would have been minor without improvements to the computer technology and the availability of rather inexpensive but reliable personal computers.

The technology to support an information system is, itself, composed of one or more **processing units** and three systems: **communication**, **data management**, and **control**. These systems are conceptually distinct but are intertwined in actuality. Furthermore, each system, in its own way, has a foot in the real world and another in the virtual world of the information system, bridging the gap between the two.

### Processing Unit(s)

A processing unit is an entity that applies the logical rules of the information system to data.

Until the invention of mechanical calculators and then electronic CPUs (central processing units), the processing units were humans employed to process data into information: mainly accountants (as information systems mostly performed book-keeping functions), but also surveyors, mapmakers, census takers, tax estimators, etc.

As previously mentioned, an information system is a system that processes data into information based on a set of logical rules. Processing units play a vital role in information systems, but they are not the whole story. Let us explain.

Banks, and the concepts of “bank account” and “double-entry bookkeeping,” predate modern computers by many centuries. But the logical rules are essentially the same: When a bank customer deposits money in the bank, the amount (**data**) is credited to the customer’s account (**processing**) and the balance of both the customer’s account and the overall bank’s debts and assets is adjusted accordingly (**information**). Before the advent of computers, bank clerks performed the required processing (with the help of abacuses and calculators). With the adoption of computers, the CPU applies the same logic and performs the calculations. But the processing units merely apply the logical rules with varying degrees of speed and accuracy: They do not set the rules by which the information system operates.

The *technology* of processing data, of course, is not irrelevant to the functioning of information systems. Far from it: The speed and the accuracy of electronic processing units have radically changed expectations and, consequently, the logical complexity of information systems has reached a point where resorting to any previous technology has become inconceivable. Modern businesses, governments, factories, and airlines (to name only a few) would grind to a halt without computers, even if they can employ thousands of clerks equipped with the best abacuses.

Still, it is crucial to remember that it is the creators of information systems who set the logic (simple or complex) that the technology executes.

## The Communication System

The communication system transmits data to the information system and carries information back to its users.

Without incoming data and outgoing information, a processing unit (and, as a result, the whole information system) would be useless. It is the task of a communication system to carry messages to and from an information system.

Communication systems (or networks) are many and varied: the telephone system, the postal service, the network of geocentric satellites, the telegraph network (that quietly passed away quite recently). But, regardless of technological sophistication, any communication system consists of communication devices, protocols, and the connections between those devices:


- **Devices.** A phone set, a fax machine, a modem, an envelope, a computer keyboard, and a monitor are all communication devices. The task of a communication device is to *encode* messages (data or information) into packages that can be carried by the system to another device that will *decode* the message for the benefit of the recipient. (The *content* of the message is irrelevant to the communication device or the communication system.)
- **Protocols.** Protocols are a set of rules or standards that allow two devices to communicate. Like data and information, the protocols are *virtual*, even though they are carried by real mediums: electrical current for the telephone and the computer, written symbols on the back of envelopes, or hand signals in the communication between the hearing-impaired.


- **Connections.** Communication devices cannot exchange messages unless they are connected through some medium: electrical current over copper wires, light waves over optical fibers, radio waves, visual spectrum, voice, etc.

Information systems are so intertwined with communication systems that often it is very difficult to distinguish where one ends and the other starts. The most important distinction is not technological but conceptual: If a system processes data into information, it is an information system; if it carries messages with disregard to their contents, it is a communication system.

## The Data Management System

A data management system is a set of rules, procedures, material, and tools that stores, organizes, protects, and retrieves data needed by the information system.

 The data that is stored “for keeps” is called **persistent** and the act of storing such data is **persistence**. Remember these terms, as they are used extensively in object-oriented theory and in this text.

 A **database** is a collection of related data, while a data management system includes not just data, but also rules, procedures, the material on which the data is stored, and, depending on the technology, people in one capacity or another.

All information systems need data management systems to safe-keep data—before processing and after, temporarily or permanently. But a data management system is more than just a cellar for storing data. It must also:

- **organize** the stored data in a manner that can be retrieved as needed;
- establish connections between **related** data;
- ensure the **integrity** of data against decay, misplacement, and erroneous identification; and
- **secure** data against unauthorized access and manipulation.

Modern database management systems have gone far beyond the mere management of data. Though conceptually we should still distinguish them from information systems in the strict sense of the term, they have incorporated many information system functions such as the enforcement of business rules (or logic). In effect, there is no longer a distinct and *actual* border between the two systems.

## The Control System

The control system ❶ directs and facilitates the interactions between the building blocks of the information technology, and ❷ provides the information system with the services of information technology.

Information systems have goals that are usually reflected in the names of their applications: accounts payable, college registration, stock trading, supply chain management, and so on. But, as we have said, information systems must use a set of technologies to connect to the real world. This requirement presents two challenges:

- how to achieve the connection between the virtual world of the information system and the real world, and
- how to maintain and manage the technology that supports such connections.

Answering these challenges is the task of the control system. For computer technology, the most important (and the best-known) component of the control system is the

☞ It is interesting to know that early electronic computers—circa 1940s—did *not* have operating systems. It was with the introduction of IBM’s System 360 in 1964 that a well-defined concept for “operating system” was established.

☞ Even though this text is not about technology per se, without the automation of information systems it would not have existed. The emergence of each significant technology allows (or forces) us to view many non-technological concepts in a new perspective. Without the need to program machines to process data, we would not have been required to examine information concepts under a microscope.

☞ Two of the most important and promising technologies in the effort to achieve interoperability are **Web services** and **XML**, which we will discuss later in this text.

*operating system*, nowadays known as a “brand name” software product (such as Linux, Windows, Unix, etc.) that controls the operations of a computer and network.

But the control system goes beyond the operating system. First, even in computers, the control system is not all software: Every computer has a hardware “control bus”—a set of lines or conductors that carry signals and instructions between the CPU and various devices. Second, no operating system can function without utilities (programs that carry out specific housekeeping tasks) and drivers (software that allows communication between the general-purpose operating system and specific devices).

In other words, a control system is an interconnected *collection* of subsystems and entities—hardware *and* software—that caters to the specific needs of the technology on which that information system operates. For non-automated technologies, the control system also includes people, actions that they must take, and procedures that they must follow to ensure the successful operation of the technology.

### Information Automation

Information automation is the application of information logic to data by a device that executes a program.

The term “automation” applies to numerous areas in which labor-saving machines that can (more or less) operate on their own have replaced (or nearly replaced) humans: manufacturing, telephone switches, military drones, automatic controls, and so on. In fact, automation has found its way into common household items as well: TV sets, video recorders, radios, and personal computers.

By now electronic computers are so common that conceiving any other automated device for processing data is difficult. Indeed, the terms “information system” and “computer” have become near synonyms.

In the past fifty years the progress in the automation of information technology has been impressive but the trend does not show any signs of slowing down. On the contrary, the existing technology is overhauled constantly and new areas appear continuously. One of the latest is *interoperability*, driven by strong market demand: Producers and consumers of information are searching for standards and technologies that would allow exchange of information free from the current obstacles resulting from incompatibilities among systems and lack of standards.

In the blizzard of new tools, standards and protocols for the technology of information automation, one important rule must not be overlooked:

The task of the information technology is to support information systems; the task of information systems is to support human enterprises.

As we shall explain later, the problems of information systems and the issues of the information technology belong to two different (although related) spheres: the “problem space” and the “solution space.”

### Applications and Systems

An application is a set of programs that performs a specific task.



Applications are software programs that help their users perform well-defined jobs: word processing, photo editing, contact management, accounting, etc. However, we started this chapter by introducing information *systems*, not applications. So what is the difference?

At one time, no great difference existed. Very early electronic computers were capable of solving only individual functions. Later, they grew to handle more complex tasks, from processing payroll to analyzing sales. The structure of the applications, however, remained the same: a *monolithic* piece of software that performed a task, from the beginning to the end, by itself. And these monolithic pieces were usually called applications.

At a certain point in time, the combination of rising expectations and increasingly powerful computing platforms resulted in monolithic applications that were more capable and complex but also more fragile and prone to failure. (Imagine, if possible, a modern passenger jet made up of one piece, not thousands of parts.)

The early efforts to address the issue of *complexity* in software were focused on the *inside* of the application: how to structure various functions within the application to facilitate its construction and maintenance. Next came the idea of *modularity*: the concept that an application can be constructed from a set of modules, conceived around a certain functionality—database module, calculation module, reporting module, etc., so that the code would become more reusable and reliable, teams of programmers could work on different parts of the application, and redundancy would decrease. This approach also resulted in the acceptance of general-purpose *libraries*: pieces of independent software that could be used by unrelated applications that, nevertheless, share certain common needs.

The realization that managing complexity needs a new vision appeared only gradually (and under the fire of failures). This vision holds that, to be successful, applications cannot be conceived as islands unto themselves.

Applications must be viewed and developed as integral parts of an information system.

In other words, applications can no longer remain as monolithic “pieces” of software, but must be constructed as components of a system (even if the system supports only one application). The “system” approach allows software to absorb complexity without falling victim to it.

Two other trends have supported this approach: *object orientation* and software development as an *architectural enterprise*.

☞ It is difficult to declare, with any certainty, when the awareness of the shortcomings of monolithic software became clear. The first major *practical* steps to remedy the situation, however, can be attributed to the late 1970s and early 1980s.

### 3. THE INFORMATION SYSTEM AS PRODUCT

---

Before information automation, the term “information system” was not applied to what we now associate with it. Companies had “accounting” (the oldest type of information system, undoubtedly), surveyors gathered geographical data, mapmakers made maps, spies gathered intelligence, and so on. Even after computers first appeared in the workplace, software was not considered as *merchandise*, a product that you build for selling in the marketplace and buy from the marketplace.