# INTEGRATION-READY ARCHITECTURE AND DESIGN

What would you do if your IT job was no longer performed in your country? Your survival does not lie in limiting global collaborative engineering. IT workers will survive and prosper because of their ability to innovate, to quickly learn and change directions, and to evolve from Information Technology into Distributed Knowledge Technology. You have no choice but to be pro-active, learn to stay current, and even run ahead of the game.

*Integration-Ready Architecture and Design* shows how to build presentation factories and seamless integration of VoiceXML, WAP, and Web technologies, providing access to corporate data and services not only through PCs and corporate workstations, but also through multiple types of wired and wireless devices and PDAs. The author integrates theory and practice, going from foundations and concepts to specific applications and architectures. Through deep insights into almost all areas of modern CIS and IT, he provides an entry into the new world of integrated knowledge and software engineering. Readers will learn the "what's, why's, and how's" on: J2EE, J2ME, .NET, JSAPI, JMS, JMF, SALT, VoiceXML, WAP, 802.11, CDNA, GPRS, CycL, XML, and multiple XML-based technologies including RDF, DAML, SOAP, UDDI, and WDSL.

For Internet and wireless service developers, this book contains unique recipes for creating "integration-ready" components. Architects, designers, coders, and even management will benefit from innovative ideas and detailed examples for building multi-dimensional worlds of enterprise applications. Throughout, the book provides a "unified service" approach while creating a core of business frameworks and building applications for the distributed knowledge marketplace.

**Jeff Zhuk** is the President of Internet Technology School. A software architect and developer with more than twenty years of experience and numerous patents and publications, he teaches at the University of Phoenix and DeVry University, and he conducts corporate consulting and training. He has pioneered IPServe.com and JavaSchool.com, which promote collaborative engineering and a distributed knowledge marketplace. An expert in distributed enterprise applications and wireless, XML, and Java technologies, his current focus is on the integration of software and knowledge engineering in a new development paradigm.

Look out for code examples and updates on the book's Web site www.cup.org/Titles/0521525837.htm

This book is dedicated to my parents, Lubov and Veniamin, and to my wife, Bronia.

# INTEGRATION-READY ARCHITECTURE AND DESIGN

## SOFTWARE ENGINEERING WITH XML, JAVA, .NET, WIRELESS, SPEECH, AND KNOWLEDGE TECHNOLOGIES

JEFF ZHUK
Internet Technology School, Inc.

CAMBRIDGE UNIVERSITY PRESS

PUBLISHED BY THE PRESS SYNDICATE OF THE UNIVERSITY OF CAMBRIDGE
The Pitt Building, Trumpington Street, Cambridge, United Kingdom

CAMBRIDGE UNIVERSITY PRESS
The Edinburgh Building, Cambridge CB2 2RU, UK
40 West 20th Street, New York, NY 10011-4211, USA
477 Williamstown Road, Port Melbourne, VIC 3207, Australia
Ruiz de Alarcón 13, 28014 Madrid, Spain
Dock House, The Waterfront, Cape Town 8001, South Africa

http://www.cambridge.org



First published 2004

Printed in the United States of America

*Typefaces* ITC Berkeley Oldstyle 10.75/13 pt. and ITC Franklin Gothic *System* LaTeX 2ε [TB]

*A catalog record for this book is available from the British Library.*

*Library of Congress Cataloging in Publication Data*
Zhuk, Jeff.
Integration-ready architecture and design : software engineering with XML, Java, .NET, Wireless, speech, and knowledge technologies / Jeff Zhuk.
p. cm.
Includes bibliographical references and index.
ISBN 0-521-52583-7 (pb.)
1. Software engineering. 2. Expert systems (Computer science) I. Title.
QA76.758.Z48 2004
005.1 – dc22 2003065381

ISBN 0 521 52583 7 paperback

# Contents

# Preface

## Who Should Read This Book?

*Integration-Ready Architecture and Design* strives for a union of theory and practice. Teaching the latest wired and wireless software technologies, the book is probably the first entry into "the next big thing," a new world of integrated knowledge and software engineering. Written by a software architect and experienced trainer, this book is for:

- **Software architects, designers, and developers**
- **Internet and wireless service providers**
- **IT managers and other IT professionals, as well as amateurs**
- **Subject matter experts** who will directly participate in a new development process of integrated software and knowledge engineering
- **Students and educators**, who will find up-to-date materials for the following courses:

1. Software Architecture,
2. Software Engineering,
3. Programming Concepts,
4. Information Technologies,
5. Smart Card and JavaCard Technologies,
6. Wireless Technologies,
7. J2ME and Wireless Messaging,
8. XML Technologies,
9. Speech Technologies,
10. Java Language and Technology,
11. C# and .Net Technology,
12. Integration Technologies,
13. Business Communications and Collaborative Engineering,
14. Web Technologies,
15. Introduction of Ontology, and
16. Integrated Software and Knowledge Engineering (introduced in the book)

- **Peers: students, instructors, consultants, and corporate team players** who might start using a peer-to-peer educational tool offered in the book as their entrance to the distributed knowledge marketplace
- **All of the above who want to know** how things work, should work, and *will* work in the IT world

## Why Did I Write This Book?

Of course, I wanted to solve several global-scale problems. Divided by corporate barriers and working under "time-to-market" pressure, we often replicate data and services and produce software that is neither *soft* nor *friendly*. Working as fast as possible in this mode, we deliver products that lack flexibility and teamwork skill and are hardly ready for integration into new environments. These products strictly target user requirements—which become obsolete by the time the project ends. Producing "more of the same" and raising the number of product choices (instead of moving to new horizons), we actually increase entropy and slow down the progress of technology, which depends heavily on inventions, new usage, or new combinations of existing tools and methods.

The famous formula "write once" is not working *anywhere* today. One of the reasons is the absence of a mechanism capable of accepting, classifying, and providing meaningful information about new data or services created by knowledge producers.

**We have not changed our way of writing software during the past twenty years.**

**We have not moved far from the UNIX operational environment (which was a big hit thirty years ago).**

**Our computers are much faster, but for the regular user, they are as stupid as they were forty years ago. We add power but we fail to add common sense to computers, we cannot help them learn, and we routinely lose professional knowledge gained by millions of knowledge workers.**

Meanwhile, best practices in software and knowledge engineering are reaching the point of critical mass. By learning, understanding, and *integrating* them, we can turn things around. We might be able to improve the reliability of quickly changing environments by using distributed self-healing networks and knowledge base–powered application solutions.

We can finally stop rewriting traditional address book, scheduling, inventory, and order applications. We will shift our focus from ironing out all possible business cases in our design and code to creating flexible application mechanisms that allow us to change and introduce new business rules on the fly. Coming changes are similar to the transition from structural to object-oriented programming. We are going back to school.

## How to Use This Book

I hope you find this book on your list of recommended reading or as "the best gift for yourself, your spouse, and your friends." Just buy two copies and let *them* figure out what to do with the book. In the worst case, it can be used for self-defense. It is almost as heavy as other good books.

If your gift list includes yourself, you might want to read this book in the bookstore first—at least some selected chapters, starting from the back.

For example, Chapter 10, about a JavaCard key that opens all doors, can be very handy the next time you lock your keys in your car. If this happens too often to you or your close relatives, you might find Chapter 11, on J2ME and wireless messaging, very practical.

Armed with the knowledge of wireless technologies from Chapters 8, 9, and 11, you can create your own communication service and finally stop switching from AT&T to Sprint and back to Verizon. Serve your friends and neighbors, compete with T-Mobile, and someday I'll be happy to buy your integrated "wireless portal communicator" product.

If you are a serious developer or plan to become one, you might prefer to start from the beginning and read all the way through. Search for long-term, secure, and exciting IT directions. Find out why all the pieces of the puzzle, as well as the glue, are almost equally important. Teach yourself to see every technology (component) as an object with three dimensions: what, why, and how. After reading the book, you might even become less serious and more efficient.

If you want to increase your business clientele from the 20% of the population who are fluent in current computer interfaces to the rest of us, including those who hate computers or cannot bear their stupidity—just go for it! Read Chapters 4, 5, 12, and 13, on speech and knowledge technologies, and create a natural user interface, a bridge from your business to humankind.

A professional hacker (whose average age is 15 but ranges from 6 to 66) might start with Appendix 3 ("Source Examples"). Find examples that can help to build collaborative and location-based services, screen/voice instant sharing and security monitoring, and speech and distributed knowledge alliance applications. Look there for spam killer hints to be ahead of the game.

If you just want to speak more languages, go to Appendix 1 ("Java and C#: The Saga of Siblings"). You can get two for the price of one, including the latest JDK1.5 language innovations.

If you would like to include XML in your repertoire, add Appendix 2 ("XML and Web Services"), which covers several dialects of the XML family.

Chapters 5 and 13 are not only for computer folks. The elusive category of "knowledge workers"—anyone who has gained knowledge and never had a chance to share—might be looking at the Promised Land. Subject matter experts (SME)—who used to talk to developers about *what* and *why*—can find in those chapters new ways to say *how*.

There is also a downloadable software product with this book. Students and educators can use the tool for collaborative work in team projects. The tool helps to connect students and instructors with educational knowledge resources. This can elevate the visibility and quality of student projects and transmit the best of them into industry contributions. The software can be handy in academic/corporate alliances.

## What Is This Book About?

- The what's, why's, and how's on: J2EE, J2ME, .NET, JSAPI, JMS, JXTA, JMF, SALT, VoiceXML, WAP, 802.11, CDMA, GPRS, CycL, XML, and multiple XML-based technologies, including RDF, DAML, SOAP, WSDL, and UDDI. The book turns these abbreviations into understandable concepts and examples
- The distributed knowledge marketplace
- Collaborative engineering methods and technology

- XML and Web technology architecture, design, and code patterns
- Ontological (knowledge) engineering and natural user interface
- XML-based application scenarios that integrate dynamic user interface and traditional services with speech and knowledge technologies
- Unique recipes for creating integration-ready components across a wide range of client-server, multi-tier, and peer-to-peer distributed architectures for Internet and wireless service developers
- Innovative ideas, methods, and examples for building multidimensional worlds of enterprise applications
- Privilege-based access to corporate data and services, not only through PCs and workstations but also through multiple types of wired and wireless devices and personal digital assistants with seamless integration of wireless, Web, speech, and knowledge technologies
- A unified approach to architecture and design that allows for J2EE and .NET implementations with code examples in Java (most of the source code) and C# languages
- Integration of software and knowledge engineering in knowledge-driven architecture
- Union of theory and practice. As many of us do, I divide my professional time between the education (30%) and development (70%) worlds. Time sharing is not always the best option in real estate, but I hope it works for this book by providing more cross-connections between these parallel worlds. Like most parallel worlds, they have (almost) no intersections according to Euclidian geometry, but we can find many by moving into Riemann's space
- Questions and exercises, case study assignments, design and code samples, and even a learn-by-example self-training system that allows you to enter the distributed knowledge marketplace

Bridging the gap for a new generation of software applications, the book teaches a set of skills that are becoming extremely valuable today and that will certainly be in high demand tomorrow.

This book is designed to walk a reader through the peaks of current software, with a focus on foundations, concepts, specifications, and architecture. The hike then goes down to the valley of implementation, with detailed design examples and explanations, and finally flies to new horizons of software and knowledge technologies. There lies the happy ending, where software and knowledge engineering ride off into the sunset together.

# Contributors

Many special thanks to the people who made direct contributions to this text:

**Ben Zhuk**—Cowrote a number of sections, edited the entire book for both content and style, and created all diagrams and illustrations (except those mentioned below). He was a sounding board for ideas throughout the writing process and was an invaluable resource. I am indebted to him, more than I can express, for his tireless efforts and the countless hours he put into this project.

**Dmitry Semenov**—Senior Software Architect who read the manuscript carefully and thoroughly. Dmitry's remarks and criticism helped me to clarify content and add significant parts to Chapters 3 and 13.

**Olga Kaydanov, artist**—Provided great design ideas and artistic inspiration for illustrations (http://artistandart.com).

**Irina Zadov, artist**—Illustrated Fig. I.1, Fig. I.2, Fig. I.5, and Fig. 1.4 (http://ucsu.colorado.edu/~zadovi).

**Inna Vaisberg, designer**—Illustrated Fig. 3.1 and Fig. A3.18 (http://javaschool.com/skills/Inna.Vaisberg.Portfolio.pdf ).

My former students, talented software developers:
**Masha Tishkov**, who wrote and tested several XML parser methods in the Stringer.java source and helped to prepare the uploader package;

**Slava Minukhin**, who teamed up with me to write C# sources for Appendix 3: TextToSpeech.cs, SocketTTS.cs, Recognizer.cs;

**Alex Krevenya**, who helped write email- and spam-related sources for Appendix 3; and **Dina Malkina**, who wrote C++ sources for Chapter 12: ListeningClient.cpp and TalkingClient.cpp.

**Thank you so much!**

# Acknowledgments

This is a great opportunity to say thank you to so many people without whom this book would be impossible. Thanks to my parents; if not for them, you might be holding a different book right now.

To my friends (many of them former students), who assisted and supported me with many essential steps in the book's production.
To Candi Hoxworth, IT Manager, who read most of the book and provided great suggestions on improving my American accent in it.
To Stuart Ambler, Mathematician and Software Architect, who reviewed and provided important feedback for Chapters 8, 10, and 12.
To Michael Merkulovich, Software Team Lead, who read and provided valuable suggestions for Chapter 6 and Appendix 1.
To Nina Zadov, Senior Software Engineer, who read and approved Chapter 7.
To Roman Zadov, Mathematician, who reviewed Chapter 5 (Ontology).
To Bryan Basham, Java Instructor, who reviewed a section (multipart/form upload) from Appendix 3.
To Linda Koepsell, Course Development Project Manager, who reviewed a section from Chapter 11 (J2ME).
To Jason Fish, Enterprise Learning, President, who invited me to teach for Java University at an international conference, where I met Lothlorien Homet from Cambridge University Press. Thank you Jason and Lothlorien, you got me started with this book.
To Cambridge University Press editors Lauren Cowles and Katherine Hew and TechBooks project manager Amanda Albert and copy editor Georgetta Kozlovski for their work and dedication.
To Cyc Corporation knowledge experts John De Oliveira, Steven Reed, and Dr. Doug Lenat, who taught me ontology and the Cyc Language and reviewed several sections from Chapters 5 and 13.
To my colleagues from the University of Phoenix, Mary A. Martin, Ph.D., Blair Smith, Stephen Trask, Adam Honea, Ph.D., and Carla Kuhlman, Ph.D., who reviewed sections from Chapter 2 (Software Architecture) and Notes for Educators.

To my colleagues from DeVry University, Ash Mahajan, Karl Zhang, Ph.D., and Mike Wasson, who reviewed the Preface and Introduction.
To Victor Kaptelinin, Ph.D., Umea University (Sweden), professor with whom I discussed multiple ideas for collaborative environments.
To Jay DiGiovanni, Director of Software Development, who reviewed and provided encouraging remarks on a section from Chapter 13.
To Vladimir Safonov, Ph.D., St. Petersburg University, Professor, who offered excellent suggestions for Chapter 2 (Software Architecture).
To Vlad Genin, Ph.D., Stanford University and University of Phoenix, Professor of Engineering, who gave me important notes on introductory sections.
To Robert Gathers, GN President, with whom I discussed the future of distributed networks and who reviewed several sections from Chapters 13 and 14.
To Rachel Levy, whose reviews of and ideas for the Introduction and Preface were inspired and right on-target.
To my children, Julie and Ben, for their moral support and phenomenal help during the entire process.
And finally, and most importantly, to my wife, Bronia, who makes everything in my life possible.

# Introduction

One might think that the software industry is performing very well because it is armed with object-oriented approaches, Web services, Java and .NET technologies, and so forth. Unfortunately, this is not true.

There may be something wrong with the way we write programs. The process has not changed much during the past twenty years, except that applications and tools are getting bigger. Yet are they better and more scalable? Do they require any common sense? Can they be reused in different circumstances?

If these things were true, I do not think we would be rewriting the address book, schedule, order, and inventory applications over and over again instead of moving to new, untouched tasks. We would be able to accumulate the professional knowledge gained by millions of knowledge workers (everyone who manages information flow on a daily basis) instead of routinely losing it, as we do today. We would also not be facing the current IT crisis.

We could even have had more precise and direct access to the market's supply and demand, which would have reduced the glaring inefficiencies of the software marketplace of the 1990s. A big change is required to return investors' confidence to IT, and, hopefully, the change is coming.

Yes, technology can help economic stability if applied with precision. Sometimes I wonder why big companies are constantly growing bigger while small ones tend to disappear. Why do corporations prefer doing business with a few vendors, or often a single vendor, even when it is an expensive one? One of the reasons is that the integration of multiple vendors' products would be even more expensive.

## Why Is Product Integration Such a Pain?

Products and services are currently designed to cover *predefined* tasks and work with *known* data.

Are they ready for *unknown* tasks and new data sources?
Are they ready for integration with other products and services?
Are they ready to be extended into the wireless world?


**FIGURE I.1. Cooking Applications on Demand.**

We think we know what is going on in our industry, but how close are we, really, to even asking the right questions?

Where can you look for information? Google.com is one answer. Where can you find any tangible product to buy? Ebay.com is a good solution. Can you find a service you need? This question is a bit harder, and the short answer is no, except for the technologies to register *Web services*, which we explore in following chapters. The ocean of industries and markets is filled with a myriad of services, but we notice only the brightest and loudest fish on the surface and have no equipment to help us swim the depths and explore this underwater universe.

If we somehow find a service, then we have a chance to take a closer look. We may subsequently find that it is not exactly what we *really* want.

Can we modify a product or service? Can we easily plug it into another service? Can I cook my application as easily as my lunch, from products and services selected right now for a single usage, as in Fig. I.1?

What does it mean to build *integration-ready* products?

Why and how do we use XML, Java, .NET, wireless, and speech technology components?

What is the next big thing? Is it about collaborative engineering, distributed knowledge marketplace, *knowledge-driven architecture*, or all of the above?

This book answers these questions, walks readers through the edge of current technologies, explains their fundamentals and interdependencies, helps to integrate the best practices of existing technologies into a new development paradigm, and provides an entrance into the world of knowledge alliances.

Examples, case studies, and the self-training application offered by this textbook arm students and instructors with ammunition for successful teamwork.

FIGURE I.2. A Mysterious Planet.

## AN ILLUSTRATIVE PASSAGE FROM "BEN'S REAL AND DREAM MEMOIRS"

A hint: even if Ben were a real person (I doubt it) his stories are not necessarily real.

> *"If you are not a lucky person, you'd better get some skills." I suddenly appreciated this aphorism praising the advancement of knowledge as I rushed down the runway. I had just realized that, having failed to check the schedule, I was about to miss the last kicker-ship and now risked spending the night on the tiny planet known only as H.235.*
>
> *This was strictly forbidden by travel rules. H.235 was one of the infamous "black holes" rarely visited by tourists. There were rumors about strange forms of nightlife on the planet. No, this was not related to sex, but to unions of magicians, the legendary "global mind," and disappearing travelers. I tried to flag down a couple of H'sters, but they did not understand English.*

*They immediately threw many extremely strange and incomprehensible words at me and then moved on when my face showed no understanding. I had always suspected that the people here spoke a different language, but I never expected it to go this far.*

*The street was now empty except for an old man coming my way. My memory struggled to find the right words, and a single phrase came out: "Sprechen sie Deutch?" This was the sole gem in my collection of foreign language knowledge.*

*The old man stopped and smiled while taking out his cell phone. My first thought was, "He is calling the police!" Then the second, "He is calling a gang!" The man spoke the strange language into the phone, repeating a word that sounded almost, but not quite, completely unlike the word* Instructions. *Suddenly, he handed the phone to me.*

*A voice from the phone spoke clear English. "This is the Common Brain help line. You are granted a guest level of permission to ask any question, except those related to H.235 security. Your question will be translated into our language. Please end your question with the words* exit and translate."

*I started asking rapid-fire questions. The phone worked like magic.*

*After the first question, the voice from the phone took a leading role. It navigated me through a question-and-answer session, helping me express my needs in system-specific terms. At the end of the session, the phone commanded: "Mister Ben, please pass the phone to the answering party."*

*The old man, whose name turned out to be Paul, played the role of the answering party. He listened to the phone's version of my queries and made strange sounds that came to me via the phone in perfect English. At the end of the call, Paul used his phone as a camera to photograph my smiling face.*

*The voice from the phone explained that I had missed the regular night shuttle, but could still make the special shuttle reserved for just such occasions. My picture was in the system now to allow me past the auto flight attendant. I was provided with precise directions over the phone, and at one point, a map was even displayed on the phone's screen.*

*I felt so happy I even tried to ask a couple of extra questions, but the only answer I received was "Questions about religion or sex cannot be answered, as they are related to H.235 security."*

*While on the way home in the shuttle, I thought about the Common Brain. Was it a high-tech company or a psychic organization? Or maybe both would be needed to help people understand each other?*

*Where was the catch? What technology was used? Could the old man have been a magician? I had heard the rumor that some magicians had access to a legendary "global mind." Why was there a special flight just for me? And why was this flight taking so long?*

*My eyes closed as a white fog slowly surrounded me.*

## Back to Reality: Mosaics of Technologies

That story sounds like science fiction, except for the final questions. These questions bring us back to reality, introducing *mosaics of technologies*.

Let us answer one question at a time. Was it technology or magic? Figure I.3 illustrates three related technologies that could be used today to make that science fiction story a reality.

Web technology is a respected parent: young, but mature. It has already raised a couple of very promising kids, and more are expected. A Web client or Hypertext Markup Language (HTML) browser talks directly to a Web server using HTML over the Hypertext Transfer Protocol (HTTP). HTML pages deliver text, graphics, and sound files over wired networks.

One of the Web's children is based on the wireless application protocol (WAP) and wireless markup language (WML). The small screen of a cellular phone serves as a destination for

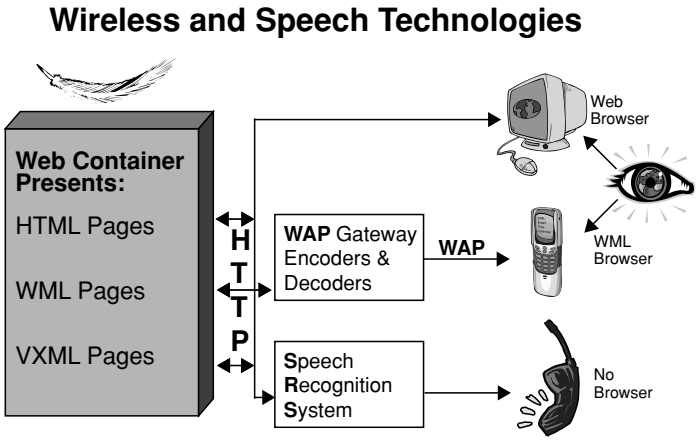


**FIGURE I.3. Children of Web Technology.**

WML pages. Not all wireless phones are capable of displaying WML pages; the phone must include a WML browser that provides this capability.

The other child of Web technology is based on a speech recognition system (SRS) and Voice Extensible Markup Language (VoiceXML or VXML) to deliver content to human beings in the most natural way possible. In all cases, the content is kept on the Web server or is generated dynamically by a Web container that includes a Web server and the software responsible for dynamic content generation.

In every case the content is presented by some kind of XML page, as XML is a common denominator for HTML, WML, and VXML. (See *http://w3.org* for a further description of this family.) SRS/VXML can be used for language translation. WAP/WML can be used for electronic document exchange: the photo ID that was created on the spot was sent to the server, and a map with directions was sent from the server back to the phone.

## FROM CENTRALIZED COLLECTIONS TO INTEGRATION-READY SERVICES

Here is an interesting puzzle stated in a simple question. It looks like we are moving more and more resources to the server side. Is this the way technology is going? Will we face a huge service repository that continues to grow and collect more and more services?

Not necessarily. Another answer to our question is to improve the capacity for integration. This means that a service (in the simplest case, an application) might be made flexible enough to allow it easily to adapt to novel contexts. A library of services can be integrated on the fly and used on demand in real time and in a fashion transparent to the end user. For example, a wireless communication channel service, a map service, a shuttle-scheduling service, and a personal security authentication service might configure themselves to provide a new, integrated service to the end user, such as booking a flight.

This solution is more than a theoretical possibility. Businesses are already beginning to turn the focus of development from "more products in a package" to "friendly (easy to integrate and customize) products." Figure I.4 represents a multidimensional view of integration-ready services.

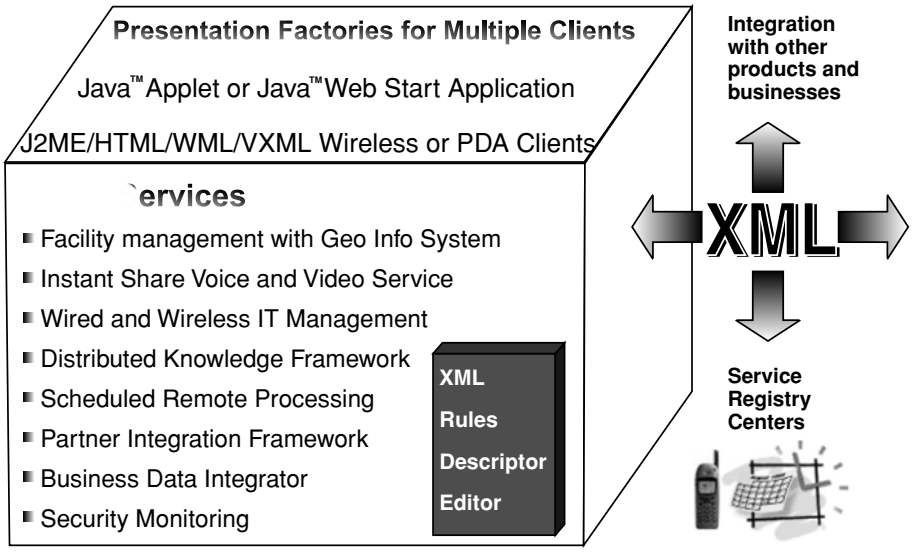


**FIGURE I.4. Multidimensional Views of Enterprise Applications.**

Perhaps you have heard about the "80/20" rule, which states that every service is focused on just the 20 percent of the possible scope that will produce 80 percent of the desired results.

This approach becomes even more beneficial when architecture, design, and development are done in such a manner that customization is an inexpensive process, when it is even possible to add or customize services without changing the core code.

What is important for corporate businesses that are selecting the right technologies?

System flexibility, the capacity to recapture business rules and structures, and readiness for integration with other products and systems are all rapidly becoming priorities. How can these goals be accomplished?

There are several methods that can be used for system integration. Java-specific integration techniques, such as JAIN (Java application program interfaces [APIs] for integrated network), introduce system integration. There is also a new world of Java-based Jini devices capable of promoting or announcing their skills/capabilities over the network. We consider these technologies and talk about a *unified service and data* integration approach.

This XML-based approach to integration interfaces defines system behavior as well as the data integration process. It paves the road for creating integration-ready services, defines unified integration protocols and policies, and helps to connect them into distributed business alliances.

Business clients access corporate repositories with multiple types of wired and wireless devices and personal digital assistants (PDAs). Corporate workstations still outnumber other devices, but this is changing rapidly.

We do not want to redevelop business services for every type of access. It is important to understand that we need only to build more ways to present/access the same content of enterprise data and services. This leads to one of the most important architecture rules: Separate business from presentation logics. This separation is very visible in Fig. I.4.

Presentation factories (software responsible for visual/audio depictions or presentations for different types of clients) allow us to reuse a set of framework services (e.g., facility management) **and** deliver service results properly formatted for multiple types of devices from corporate workstations to wireless phones or PDAs.

In the preceding example, common business problems are solved with several "core" frameworks. Each service was focused on just 20 percent of the possible scope that produced 80 percent of the results. Architecture, design, and development are accomplished in a manner that makes it possible to add or customize services without changing the core code.

This allows a business to transform into an e-business with the ability to connect to a global business alliance or distributed knowledge and service marketplace.

This marketplace instantly provides information on products, data, and services, along with their values (some services might be more valuable than others), allows multiple parties/systems to negotiate multiple forms of collaboration, and contains sufficiently flexible levels of access security.

Businesses will be able to leverage their existing application assets by making them available to others via Web service and distributed technologies. Business frameworks, presentation factories, Web service, and distributed technologies (and much more) are discussed further in separate chapters.

## Software Architecture and Integration-Ready Systems

The past decade has moved software architecture into the center of system development. It has also provided many (perhaps too many) examples of inefficient "quick and dirty" solutions prompted by a lack of integration among the three worlds of what, why, and how.

The latest buzzwords, industry trends, and components have often been used without necessity and without understanding that they are beneficial in specific circumstances under specific requirements. At the same time, some projects have not benefited from new standards and technologies, simply because developers have not been aware of their existence or have not had enough skills to apply them.

The importance of developing integration-ready systems and components is another lesson we continue to learn from the past.

The book teaches the reader to understand design patterns and architectural styles, to be able to see multiple sides of a system through the prism of industry standards and specifications, to communicate this vision via multiple architectural views, and finally to transform this vision into design and code that can make the "write once" dream a reality.

## Enterprise Service Components and Challenges

### Wired and Wireless Telecommunications

Enterprise communications include telecommunication services as well as a variety of IT and data services.

Telecommunication services are competing for a limited market. (Though we called this market "unlimited" in the past, this clearly did not help matters much.) Telecommunication vendors offer new products and services that balance reliability, superiority of features, and price. Here is the client's usual question: "Will your new program work with my old one?" In this case, "I am not sure" is not the right answer.

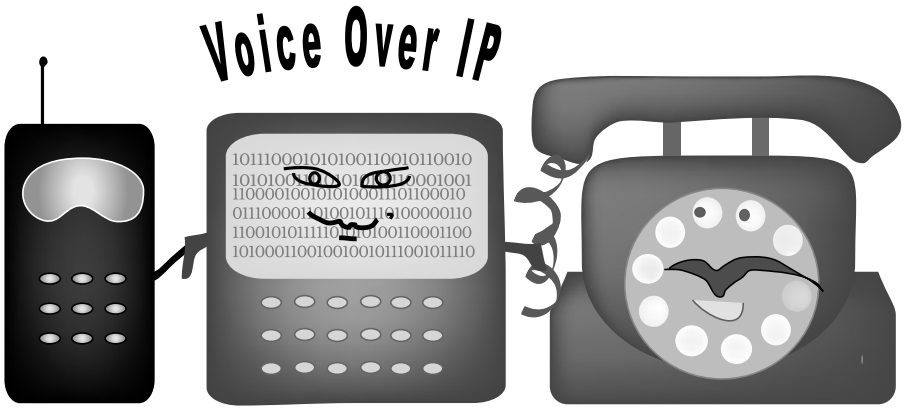


**FIGURE I.5. Phone Generations Hand in Hand.**

There are "9-to-5" employees who work only in the office and use fixed telephony. There is also a growing community of telecommuters: mobile employees who, for example, work one day in the office and four on the road, and who use wireless telephony. Corporations have multibillion dollar investments in Legacy PBXs that run regular telephony. This is the best guarantee of their longevity. Voice over IP and wireless communications will have to find ways to coexist in the mixed-enterprise telecom environment. Interoperability and a smooth transition can be achieved with a unified approach to wired and wireless application architecture.

The increasing quantity and variety of wireless devices sets new challenges for companies competing for multiple markets that run multiple wireless technologies. The winners will be those wireless Internet providers (WISPs) and wireless application service providers (WASPs) who can offer reliable connections and content services and optimize development solutions with a unified approach across multiple client devices.

Wireless clients are truly mobile. Client environments, locations, connections, and other parameters can be changed more frequently and drastically than those of fixed-wired network clients. We have to develop smart location-based services for our PDAs, cars, and robots equipped with GPS (global positioning system). We will have to switch from centralized, fixed computing to *distributed self-healing networks* with *flexible service redistribution*. We'll deploy *knowledge base–powered* application solutions: quick and smart adjustments to client locations, connections, and overall environment changes (Fig. I.5).

## Consolidation of Multiple Data Sources

Multiple data sources were historically created in multiple departments for different and often unrelated purposes. Company mergers and the process of building Web umbrellas for company services bring a new focus to an old task. Data consolidation is becoming one of the highest enterprise business priorities. Enterprise Web applications require data integration and consistency. Data integration is especially relevant to telecom applications, which by their nature must be connected to multiple enterprise data storages.

Data consolidation is an expensive, nontrivial process in which XML and Java technologies can play an essential role. In Chapter 6 we consider a unified approach to multiple data sources (Fig. I.6).

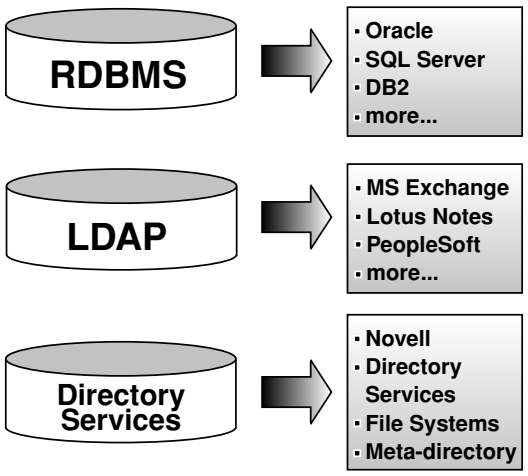


FIGURE I.6. Multiple Faces of Data Storage.

## *Natural* User Interface and Knowledge Technologies

There are dependencies between the natural user interface and knowledge technologies. For example, current speech technologies are not quite ready to understand natural language speech initiated by a person. There are similar difficulties with machine translation from foreign languages. Can tasks like machine vision, speech recognition, and foreign language translation, which seem very different, have a common solution?

The first common denominator is easy to find: all these tasks belong to the field of data processing. The next move is to find common steps for all three activities. What do people do when they are engaged in image or speech recognition, when they translate text from Hebrew to English?

Can we build a computer program to perform similar steps? We know that data we usually keep in relational databases can hardly help us. What types of data are needed for the process?

Distributed yet connected, specific yet based on generic roots, different in themes yet non-conflicting: these are the qualities required for information to be inherently useful. Creating this quality data is extremely difficult; however, it has been proven possible.

We discuss architecture, design, and code samples that integrate knowledge technologies into enterprise applications and give the keys to a new breed of software—"softology," a mix of software and ontological engineering.

Because the knowledge engine is a new element that plays an important role in such architecture, I label it *knowledge-driven architecture*. It is very likely that this type of application will become the next big thing (after service-oriented architecture and event-driven architecture) in the near future.

This promising direction can raise the IQ level of our products; add more flexibility, intelligence, and common sense to applications; and even change the development paradigm by providing business experts with better tools that can eliminate several layers of the current development process.

## Location-Based Services (Topology/Maps)

Topology data or geographical information systems (GIS) are necessary for mobile environments, facility management, military operations, and more.

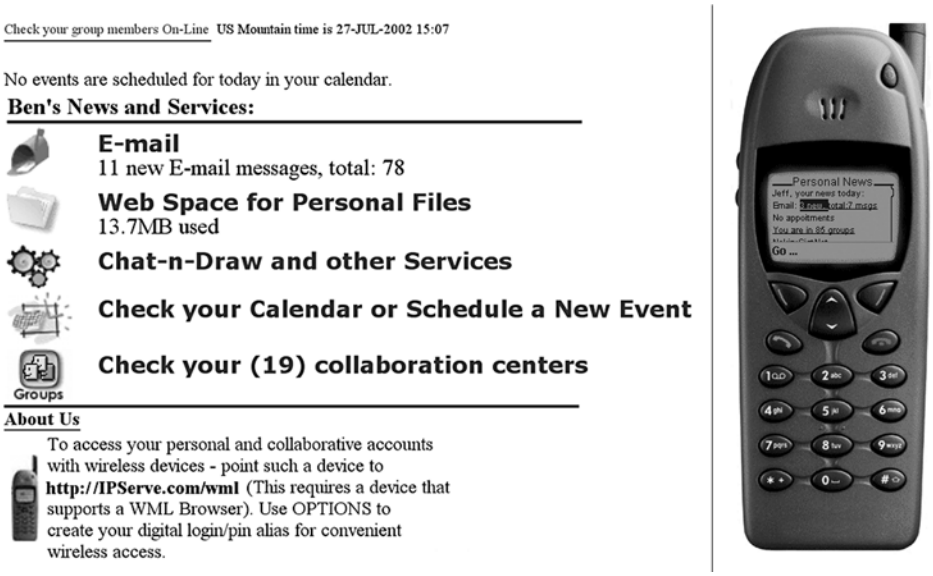


**FIGURE I.7. Integrated Wired and Wireless Portals.**

In a very peaceful example, GIS can be used to locate things such as office equipment and to optimize sharing of resources such as books, computers, facsimile machines, copying machines, or microwaves, and services based on their locations.

Topology data also can be used by robot-based moving services to determine hallway widths and produce the best or perhaps only way to move an item.

## IT Provisioning, Corporate Portals, and the Distributed Knowledge Marketplace

IT provisioning and collaboration technologies, which are often implemented with corporate portals, increase productivity and encourage the creation of a distributed knowledge and service marketplace. There, data as well as services can be measured and can represent a new currency.

People produce knowledge and services every day. Unfortunately, most of the knowledge gained by individuals is rarely shared. We have no direct access to this greatest treasure of humankind. Distributed knowledge technologies can improve the situation, drastically decreasing the time between a service's supply and the market's awareness of it, providing a more precise estimate of data and service usability values, and introducing a new marketplace where information and service contributions are rewarded according to their values.

Collaboration or openness must be supported by security mechanisms. Multiple security domains with multiple member roles and multilevel privilege-based access are a necessity to encourage data sharing and active teamwork. Collaboration is not just about corporate employees working at corporate workstations. Wireless devices using WAP or speech technology (phone) clients, as well as regular workstations, will also be looking to access the rich content of enterprise data and services. The full content can be made available via corporate workstations while team members with thin client devices use a subset of the data and services (Fig. I.7).

## Integration of Software and Knowledge Engineering

There is a gap in the current development process between initial business input provided by business people and the final implementation. The current process requires multiple transformations to simplify the complex world of reality into low-level program functions and tables expressed with current programming languages. This gap is filled with multiple filters/layers and sometimes multiple teams.

The shift to service-oriented architecture together with recent advances in knowledge technologies has paved the way for a new development paradigm.

In the new development paradigm, knowledge technologies play an essential role in the development process. Business domain experts will be able to directly participate in the development agenda by writing business rules and application scenarios in a more "natural" language. This will also help to focus developers (currently distracted by technological details) on the business aspects of applications.

Some time ago, we thought of the C language as a language for application development, whereas Assembly was the language for system development. It is about time to move the current programming languages, such as C# and Java, to the system level and make applications that are driven by knowledge-based rules and scenarios.

Knowledge-driven architecture is not a dream but a reality.

## Start with Good Design and Repeatable Process

There is always a temptation to jump ahead several steps. Sometimes this is possible, but it is never safe. (This is my experience, at least, as a former mountain climber and an inventor.) The plan for this book is to learn the best current development practices in the software industry, climb close to the top, and then see the horizon. We focus on Object-Oriented Programming (OOP) and we also look beyond. There are Aspect-Oriented Programming and Generative Programming directions. OOP is not the end of the story.

Current technologies have brought significant power to developers. These technologies can be successfully used to implement good design ideas. They can also be used to unsuccessfully implement bad ones. How do you start walking in the right direction before you can see the end of the road?

Enterprise applications are often complex enough to offer us this challenge. Building a robust development process and building a team capable of following this process are probably the most important tasks of development. Good teams produce good products. It is appropriate, then, to start Chapter 1 with a discussion of teamwork and development processes and then to move on to technology.

# Notes for Educators: AMA Teaching Methods

The AMA teaching methods are Activate, Motivate, and Assess.

## Activate

Interact with the group and solicit each student's contribution. Engage your students in *collaborative learning* (let them work in teams) and *collaborative teaching*—give the stage to your students so they can review the material and share their best practices. One learns twice while teaching! Reward the best contributors.

## Motivate

Use all possible means to connect the course materials to the students' areas of expertise, thus increasing their motivation level. Motivation is the key to opening students' eyes and ears and unlocking their memories.

## Assess

Facilitate a question-and-answer session, let students work in groups, and help them keep precise focus and timing.

Use the online assessment and evaluation forms periodically from the very beginning of the course.

This book provides a special section at the end of each chapter to reinforce the AMA methods.

The **Integrating Questions** are designed to help students build a thorough understanding of the relevance, relationships, and application of the content in the real world. To ensure

that students can relate course theories to the workplace, illustrate points with examples drawn from your professional experience and the experiences of the students.

The **Case Study** serves as a guide for student assignments. The instructor can modify and personalize assignments based on instructor–student interactions and early assessments of a student's current and desired skills.

### Recap Presentations and Midterm Assessments

Conduct two- or three-student recap presentations (ten to fifteen minutes each) starting with the second class. The content of each presentation is the student's work on the Case Study–based assignments. The presentations may serve as midterm assessments. They also increase students' motivation level and their understanding of the material (you learn twice while teaching).

### Delivery Format for Assignments and Presentations

It is recommended to format assignments and presentations as illustrated white papers using, for example, Web pages, MS Word, PDF, or PowerPoint with notes. The content can be presented in three levels: definition (high level), functionality and applicability (middle level), and examples of implementations (low level). These three levels roughly map the "why," "what," and "how" of the selected subject. Students will enhance their ability to clearly express their ideas and will learn the art of publishing and presentation with text, images, diagrams, and code extracts.

### Final Team Projects

Students may integrate selected parts of their weekly work assignments into team projects. This approach helps the student increase the quality of his or her work and consistently focus on the course materials.

Integration is not a copy-and-paste process. The team project allows students to exercise the downloadable software tool that comes with this book and helps them share their work with a distributed knowledge repository. The high visibility of a student's achievement adds to the perfection of her or his work and makes it possible to directly tie students' efforts to industry needs.