TURING 图灵程序设计丛书 微软技术系列

Microsoft

.NET软件架构设计圣经

Microsoft .NET
Architecting Applications for the Enterprise

.NET软件架构之美
（英文版）

MVP Microsoft Most Valuable Professional

[意] Dino Esposito
Andrea Saltarello 著

- Amazon全五星图书
- 紧贴实战，透过实例探讨架构设计最佳实践
- 深刻阐述软件开发思想

人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵程序设计丛书 微软技术系列

Microsoft .NET
Architecting Applications for the Enterprise

.NET软件架构之美
（英文版）

[意] Dino Esposito
Andrea Saltarello 著

人民邮电出版社
北 京

图书在版编目（CIP）数据

.NET 软件架构之美 = Microsoft .NET: Architecting Applications for the Enterprise：英文 /（意）埃斯波西托（Esposito, D.），（意）索尔塔雷罗（Saltarello, A.）著 . —北京：人民邮电出版社，2009.9
（图灵程序设计丛书）
ISBN 978-7-115-20018-1

Ⅰ. N… Ⅱ. ①埃…②索… Ⅲ. 计算机网络－程序设计－英文 Ⅳ. TP393.09

中国版本图书馆CIP数据核字（2009）第104199号

内 容 提 要

本书出自两位具有多年软件开发经验的 ASP .NET 专家、作者和培训师之手，内容涉及多层架构、设计模式以及设计原则。第一部分简要介绍 UML、设计原则及模式；第二部分从技术架构角度讨论分层设计。本书行文流畅，语言通俗易懂，阐述了各种架构设计技术方案的优与劣，并讲述了如何在优与劣中做出权衡。书中设计了真实的场景，展示了如何将这些设计原则更加具体地应用到 .NET 应用程序中。

本书适合各层次 .NET 开发人员阅读。

图灵程序设计丛书

.NET软件架构之美（英文版）

◆ 著 ［意］Dino Esposito Andrea Saltarello
责任编辑 傅志红

◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街14号
邮编 100061 电子函件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
北京鑫正大印刷有限公司印刷

◆ 开本：800×1000 1/16
印张：28.5
字数：547千字 2009年9月第1版
印数：1－2 000册 2009年9月北京第1次印刷
著作权合同登记号 图字：01-2009-3797号

ISBN 978-7-115-20018-1/TP

定价：69.00元

读者服务热线：(010)51095186 印装质量热线：(010)67129223
反盗版热线：(010)67171154

版 权 声 明

To Silvia, Francesco, and Michela who wait for me and keep me busy.
But I'm happy only when I'm busy.

—Dino

To Mum and Depeche Mode.

—Andrea

"Any sufficiently advanced technology is indistinguishable from magic."

—Arthur C. Clarke

Acknowledgments

For at least two years, Andrea didn't miss any opportunity to remind Dino about the importance of a .NET-focused architecture book covering the horizontal slice of a multitier enterprise system. And for two years Dino strung Andrea along with generic promises, but absolutely no commitment. Then, suddenly, he saw the light. During a routine chat over Messenger, we found out that we repeatedly made similar statements about architecture—too many to mark it down as a simple coincidence. So we started thinking, and this time seriously, about this book project. But we needed a team of people to do it right, and they were very good people, indeed.

Ben Ryan was sneakily convinced to support the project on a colorful Las Vegas night, during an ethnic dinner at which we watched waiters coming up from and going down to the wine-cellar in transparent elevators.

Lynn Finnel just didn't want to let Dino walk alone in this key project after brilliantly coordinating at least five book projects in the past.

Kenn Scribner is now Dino's official book alter ego. Kenn started working with Dino on books back in 1998 in the age of COM and the Active Template Library. How is it possible that a book with Dino's name on the cover isn't reviewed and inspired (and fixed) by Kenn's unique and broad perspective on the world of software? The extent to which Kenn can be helpful is just beyond human imagination.

Roger LeBlanc joined the team to make sure that all these geeks sitting together at the same virtual desktop could still communicate using true English syntax and semantics.

We owe you all the (non-rhetorically) monumental "Thank you" for being so kind, patient, and accurate.

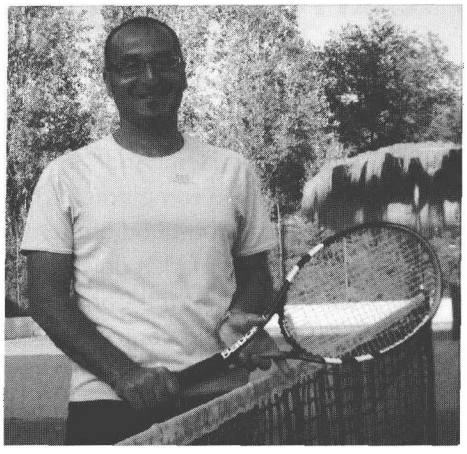
Only two authors and a small team for such a great book? Well, not exactly. Along the project lifetime, we had the pleasure to welcome aboard a good ensemble of people who helped out in some way. And we want to spend a word or two about each of them here.

Raffaele Rialdi suggested and reviewed our section in Chapter 3 about design for security. **Roy Osherove** was nice enough to share his enormous experience with testing and testing tools. **Marco Abis** of ThoughtWorks had only nice words for the project and encouraged us to make it happen. **Alex Homer** of Microsoft helped with Unity and Enterprise Library. And the whole team at Managed Design (our Italian company) contributed tips and ideas—special thanks go to **Roberto Messora**.

It's really been a pleasure!

—Andrea and Dino

Dino's Credits



This is the first book I have co-authored in 8 or 9 years. I think the last was a multi-author book on data access involving COM and OLE DB. In the past, co-authoring a book for me meant accepting to write a few chapters on specific topics, while having only a faint idea of what was coming before and after my chapters.

This book is different.

This book has really been written by a virtual author: a human with the hands of Dino and the experience of Andrea. I actually did most of the writing, but Andrea literally put concepts and ideas into my keyboard. If it were a song, it would be described as lyrics by Dino and music by Andrea.

This book wouldn't exist, or it wouldn't be nearly as valuable, without Andrea. Andrea has been my personal Google for a few months—the engine to search when I need to understand certain principles and design issues. The nicest part of the story is that I almost always asked about things I (thought I) knew enough about. My "enough" was probably really enough to be a very good architect in real life. But Andrea gave me a new and broader perspective on virtually everything we covered in the book—ISO standards, UML, design principles, patterns, the user interface, business logic, services, and persistence. I've been the first hungry reader of this book. And I've been the first to learn a lot.

It was so fun that I spent the whole summer on it. And in Italy, the summer is a serious matter. I smile when I get some proposals for consulting or training in mid-August. There's no way I can even vaguely hint to my wife about accepting them.

So, on many days, I reached 7 p.m. so cloudy minded that running, running, and running—which was more soothing than my favorite pastime of trying to catch up to and hit a bouncing tennis ball—was the only way to recover a decent state of mind. On other days, my friends at **Tennis Club Monterotondo** helped a lot by just throwing at me tons of forehands and passing shots. One of them, **Fabrizio**—a guy who played Boris Becker and Stefan Edberg and who now wastes his time with my hopeless backhand slice—has been my instructor for a while. He also tried to learn some basic concepts of Web programming during what often became long conversations while changing ends of the court. But just as I keep on twirling the wrist during the execution of a backhand slice, he still keeps on missing the whole point of HTTP cookies.

My friend **Antonio** deserves a very special mention for organizing a wonderful and regenerative vacation in the deep blue sea of Sardinia, and for being kind enough to lose all the matches we

played. It was just the right medicine to rejuvenate a fatigued spirit after a tough book project. He tried to initiate me into the sport of diving, too, but all I could do was snorkel while the kids got their Scuba Diver certification.

My kids, **Francesco** and **Michela**, grow taller with every book I write, and not because they just hop on the entire pile of dad's books. They're now 10 and 7, and Michela was just a newborn baby when I started working on my first .NET book for Microsoft Press. I really feel a strong emotion when attendees of conferences worldwide come by and ask about my kids—loyal readers of my books have been seeing their pictures for years now.

For me, this book is not like most of the others that I have written—and I do write about one book per year. This book marks a watershed, both personal and professional. I never expressed its importance in this way with **Silvia**, but she understood it anyway and supported me silently and effectively. And lovingly. And with great food, indeed!

Life is good.

—Dino

Andrea's Credits

This is my first book. More precisely, this is my first serious publication. The seeds for this book were sowed in November 2004 when a rockstar like Dino approached me and proposed that we work together.

We started a successful business partnership, and we delivered a number of classes and some articles—including one for MSDN Magazine—and took a number of industry projects home to ensure our customers were happy.

In all these years, Dino impressed me especially with his unique ability of going straight to the point, and being a terrifically quick learner of the fundamentals of any topics we touched on. More, he also showed an unparalleled ability to express any concept precisely and concisely. Countless times during this book project, I found my own wording hard to read, nebulous, and even cryptic. A few days later, instead, massaged by Dino, the same text looked to me magically fluent and perfectly understandable—just like any technical text should always be.

(OK, I admit. Sometimes I thought "I hate this man," but it was an unusual and unconfessed way to look up to Dino with admiration.)

More than everything else, what initially was a simple although successful professional collaboration turned into friendship. This book, therefore, is not a finish line. It is, instead, the starting point of a common path. I really don't know either where we're going or how long it will take, but I'm going to be happy to take the walk.

Being a full-time consultant, it was very hard for me to set aside the time needed for writing this book. So I had to start living a double life, resorting to writing in what you would define as "spare time": evenings and weekends, and suddenly the summer also became standard working time. Every now and then, it has been a little frustrating, but I found new strength and inspiration due to the love and support I was blessed with by my guardian angels: my **mom** and **Laura**. I'd like to say to them that words cannot express how precious your caring is. I love you.

Now, this is fun.

—Andrea

Introduction

Good judgment comes from experience, and experience comes from bad judgment.

—Fred Brooks

Every time we are engaged on a software project, we create a solution. We call the process *architecting*, and the resulting concrete artifact is the *architecture*. Architecture can be implicit or explicit.

An *implicit* architecture is the design of the solution we create mentally and persist on a bunch of Microsoft Office Word documents, when not on handwritten notes. An implicit architecture is the fruit of hands-on experience, the reuse of tricks learned while working on similar projects, and an inherent ability to form abstract concepts and factor them into the project at hand. If you're an expert artisan, you don't need complex drawings and measurements to build a fence or a bed for your dog; you can implicitly architect it in a few moments. You just proceed and easily make the correct decision at each crossroad. When you come to an end, it's fine. All's well that ends well.

An *explicit* architecture is necessary when the stakeholder concerns are too complex and sophisticated to be handled based only on experience and mental processes. In this case, you need vision, you need guidance, and you need to apply patterns and practices that, by design, take you where you need to be.

What Is Architecture?

The word *architecture* has widespread use in a variety of contexts. You can get a definition for it from the Oxford English Dictionary or, as far as software is concerned, from the American National Standards Institute/Institute of Electrical and Electronics Engineers (ANSI/IEEE) library of standards. In both cases, the definition of architecture revolves around planning, designing, and constructing something—be it a building or a software program. Software architecture is the concrete artifact that solves specific stakeholder concerns—read, *specific user requirements*.

An architecture doesn't exist outside of a context. To design a software system, you need to understand how the final system relates to, and is embedded into, the hosting environment. As a software architect, you can't ignore technologies and development techniques for the environment of choice—for this book, the .NET platform.

Again, what is architecture?

We like to summarize it as the art of making hard-to-change decisions correctly. The architecture is the skeleton of a system, the set of pillars that sustain the whole construction.

The architect is responsible for the architecture. The architect's job is multifaceted. She has to acknowledge requirements, design the system, ensure the implementation matches the expectation, and overall ensure that users get what they really need—which is not necessarily what they initially accept and pay for.

Software architecture has some preconditions—that is, design principles—and one post condition—an implemented system that produces expected results. Subsequently, this book is divided into two parts: principles and the design of the system.

The first part focuses on the role of the architect: what he does, who he interacts with and who he reports to. The architect is primarily responsible for acknowledging the requirements, designing the system, and communicating that design to the development team. The communication often is based on Unified Modeling Language (UML) sketches; less often, it's based on UML blueprints. The architect applies general software engineering principles first, and object-oriented design principles later, to break down the system into smaller and smaller pieces in an attempt to separate what is architecture (points that are hard to change) and what is not. One of the purposes of object-oriented design is to make your code easy to maintain and evolve—and easy to read and understand. The architect knows that maintainability, security, and testability need to be built into the system right from the beginning, and so he does that.

The second part of the book focuses on the layers that form a typical enterprise system—the presentation layer, business layer, and data access layer. The book discusses design patterns for the various layers—including Domain Model, Model-View-Presenter, and Service Layer—and arguments about the evolution of technologies and summaries of the new wave of tools that have become a common presence in software projects—O/R mappers and dependency injection containers.

So, in the end, what's this book about?

It's about the things you need to do and know to serve your customers in the best possible way as far as the .NET platform is concerned. Patterns, principles, and techniques described in the book are valid in general and are not specific to particularly complex line-of-business applications. A good software architecture helps in controlling the complexity of the project. And controlling the complexity and favoring maintainability are the sharpest tools we have to fight the canonical Murphy's Law of technology: "Nothing ever gets built on schedule or within budget."

The expert is the one who knows how to handle complexity, not the one who simply predicts the job will take the longest and cost the most—just to paraphrase yet another popular Murphy's Law.

Who This Book Is For

In the previous section, we repeatedly mentioned architects. So are software architects the ideal target audience for this book? Architects and lead developers in particular are the target audience, but any developers of any type of .NET applications likely will find this book beneficial. Everyone who wants to be an architect may find this book helpful and worth the cost.

What about prerequisites?

Strong object-oriented programming skills are a requirement, as well as having a good foundation of knowledge of the .NET platform and data access techniques. We point out a lot of design patterns, but we explain all of them in detail in nonacademic language with no weird formalisms. Finally, we put in a lot of effort into making this book read well. It's not a book about abstract design concepts; it is not a classic architecture book either, full of cross-references and fancy strings in square brackets that hyperlink to some old paper listed in the bibliography available at the end of the book.

This is (hopefully) a book you'll want to read from cover to cover, and maybe more than once—not a book to keep stored on a shelf for future reference. We don't expect readers to pick up this book at crunch time to find out how to use a given pattern. Instead, our ultimate goal is transferring some valuable knowledge that enables you to know what to do at any point. In a certain way, we would happy if, thanks to this book, you could do more *implicit* architecture design on your own.

Companion Content

In the book, we present several code snippets and discuss sample applications, but with the primary purpose of illustrating principles and techniques for readers to apply in their own projects. In a certain way, we tried to teach fishing, but we don't provide some sample fish to take home. However, there's a CodePlex project that we want to point out to you. You find it at *http://www.codeplex.com/nsk*.

This book also features a companion Web site where you can also find the CodePlex project. You can download it from the companion site at this address: *http://www.microsoft.com/mspress/companion/9780735626096*.

The Northwind Starter Kit (NSK) is a set of Microsoft Visual Studio 2008 projects that form a multitier .NET-based system. Produced by Managed Design (*http://www.manageddesign.it*), NSK is a reference application that illustrates most of the principles and patterns we discuss in the book. Many of the code snippets in the book come directly from some of the projects in the NSK solution. If you're engaged in the design and implementation of a .NET layered application, NSK can serve as a sort of blueprint for the architecture.

Refer to the Managed Design Web site for the latest builds and full source code. For an overview of the reference application, have a look at the Appendix, "The Northwind Starter Kit," in this book.

Hardware and Software Requirements

You'll need the following hardware and software to work with the companion content included with this book:

- Microsoft Windows Vista Home Premium Edition, Windows Vista Business Edition, or Windows Vista Ultimate Edition
- Microsoft Visual Studio 2008 Standard Edition, Visual Studio 2008 Enterprise Edition, or Microsoft Visual C# 2008 Express Edition and Microsoft Visual Web Developer 2008 Express Edition
- Microsoft SQL Server 2005 Express Edition, Service Pack 2
- The Northwind database of Microsoft SQL Server 2000 is used by the Northwind Starter Kit to demonstrate data-access techniques. You can obtain the Northwind database from the Microsoft Download Center (http://www.microsoft.com/downloads/details.aspx?FamilyID=06616212-0356-46A0-8DA2-EEBC53A68034&displaylang=en).
- 1.6 GHz Pentium III+ processor, or faster
- 1 GB of available, physical RAM.
- Video (800 by 600 or higher resolution) monitor with at least 256 colors.
- CD-ROM or DVD-ROM drive.
- Microsoft mouse or compatible pointing device

Find Additional Content Online

As new or updated material becomes available that complements this book, it will be posted online on the Microsoft Press Online Developer Tools Web site. The type of material you might find includes updates to book content, articles, links to companion content, errata, sample chapters, and more. This Web site is available at *www.microsoft.com/learning/books/online/developer* and is updated periodically.

Support for This Book

Every effort has been made to ensure the accuracy of this book and the contents of the companion CD. As corrections or changes are collected, they will be added to a Microsoft Knowledge Base article.

Microsoft Press provides support for books and companion CDs at the following Web site:

http://www.microsoft.com/learning/support/books

Questions and Comments

If you have comments, questions, or ideas regarding the book or the companion content, or questions that are not answered by visiting the sites above, please send them to Microsoft Press via e-mail to

mspinput@microsoft.com

Or via postal mail to

Microsoft Press
Attn: *Microsoft .NET: Architecting Applications for the Enterprise* Editor
One Microsoft Way
Redmond, WA 98052-6399

Please note that Microsoft software product support is not offered through the above addresses.

Table of Contents