

프로페셔널  
**엔터프라이즈 닷넷**  
Professional Enterprise .NET

# Professional Enterprise .NET

by Jon Arking, Scott Millett

Copyright © 2009 by Wiley Publishing, Inc., Indianapolis, Indiana  
This translation published under license with the original publisher John Wiley & Sons, Inc.  
Korean translation rights arranged with John Wiley & Sons, USA  
through Danny Hong Agency, Korea

Trademarks: Wiley, the Wiley logo, Wrox, the Wrox logo, Wrox Programmer to Programmer, and related trade dress are trademarks or registered trademarks of John Wiley & Sons, Inc. and/or its affiliates, in the United States and other countries, and may not be used without written permission. All other trademarks are the property of their respective owners. Wiley Publishing, Inc. is not associated with any product or vendor mentioned in this book.

Korean translation copyright © 2010 by J-Pub

이 책의 한국어판 저작권은 대니홍 에이전시를 통해 저작권자와의 독점 계약으로 제이펍에 있습니다.  
신저작권법에 의해 한국 내에서 보호를 받는 저작물이므로 무단전재와 무단복제를 금합니다.

## 프로페셔널 엔터프라이즈 닷넷

초판 1쇄 발행 2010년 9월 27일

저자인 존 아킹, 스콧 밀렛  
옮긴이 장현희 | 펴낸이 장성두 | 책임편집 안주연

본문디자인 초심디자인 | 표지디자인 Arowa & Arowana

주소 경기도 파주시 교하읍 파주신도시 에이15-1블록 한빛마을 휴먼빌 201-502  
전화 070-8201-9010 | 팩스 02-6280-0405  
홈페이지 [www.jpup.kr](http://www.jpup.kr) | 펴낸곳 제이펍

출판신고 2009년 11월 10일 제406-2009-000087호

용지 신승지류유통 | 인쇄 한승인쇄 | 제본 춘산제본

ISBN 978-89-94506-04-3 (13560)

값 32,000원

- ※ 이 책은 저작권법에 따라 보호를 받는 저작물이므로 무단 전재와 무단 복제를 금지하며,  
이 책 내용의 전부 또는 일부를 이용하려면 반드시 저작권자와 제이펍의 서면동의를 받아야 합니다.
- ※ 잘못된 책은 구입하신 서점에서 바꾸어 드립니다.

제이펍은 독자 여러분의 책에 관한 아이디어와 원고 투고를 기다리고 있습니다. 책으로 펴내고자 하는 아이디어나 원고가 있으신 분은  
책에 대한 간단한 개요와 차례, 구성과 저(역)자 약력 등을 메일로 보내주세요. (보내실 곳: [jeipub@gmail.com](mailto:jeipub@gmail.com))

엔터프라이즈  
개발을 위한  
최고의 가이드

# 프로페셔널 엔터프라이즈 닷넷

## Professional Enterprise .NET

존 아킹, 스콧 밀렛 지음 / 장현희 옮김



Jpub  
제이퍼블릭



친절함과 창의력, 아름다움 덕분에 스스로를 모든 남성들이 결혼하고 싶은  
여자로 만드는 나의 아내 안드리아Andrea에게 이 책을 바칩니다.

\_ 존 아킹(Jon Arking)



아름다운 린지Lynsey, 글을 쓰는 동안 당신이 가끔 마련해 준 토스트와  
한 잔의 차, 그리고 당신의 사랑에 감사다. 애거사Agatha와 콜롬보Columbo,  
너희들은 아마 세상에서 가장 귀여운 녀석들일 거야.

\_ 스콧 밀렛(Scott Millett)

## 지은이 소개

존 아킹Jon Arking은 필라델피아 지역에서 근무하는 엔터프라이즈 소프트웨어 아키텍트이다. 그는 14년 이상 다계층 시스템을 디자인, 개발 및 관리해왔으며, 특히 시스템 마이그레이션과 분산 아키텍처의 디자인 전문가이다. 존은 여러 언어와 플랫폼 상에서 프로그래밍을 경험했으며, 그의 경력 대부분은 시스템 디자인과 팀 관리, 강의, 집필 및 인터뷰 활동, 다양한 기술적인 주제를 다루는 서적 출간 등의 분야에서 쌓아왔다. 그가 운영하는 아킹 테크놀로지Arking Technologies는 필라델피아 지역의 대기업을 위한 엔터프라이즈 시스템 디자인 전문 회사이다.



스콧 밀렛Scott Millett은 영국 남부 포츠머스Portsmouth의 사우스시 southsea에 거주하고 있으며, 사이클과 철인 3종 경기에 특화된 전자상거래 회사인 Wiggle.co.uk의 중견 개발자로 근무하고 있다. 그는 .NET 1.0 시절부터 프로그래밍 경험을 쌓아왔으며, 마이크로소프트 공인 웹 개발자Microsoft Certified Professional Web Developer 자격을 보유하고 있다. ASP.NET 포럼의 정식 기고자이기도 하며, .NET 개발을 하거나 .NET 개발과 관련된 글을 쓰는 외에는 글래스턴베리Glastonbury에서 음악을 즐기고 있거나 혹은 여름에는 영국의 주요 음악 페스티벌에 참석하고 있는 그를 발견할 수 있다. 스콧과 이 책, 그리고 .NET과 관련된 모든 질문과 영국의 음악 페스티벌에 대한 이야기를 나누고 싶다면 언제든지 scott@elbandit.co.uk로 메일을 보내거나 그의 블로그 [www.elbandit.co.uk/blog](http://www.elbandit.co.uk/blog)를 방문해 보기 바란다.



# 차례

지은이 소개 • v  
옮긴이 후기 • xiv  
베타리더 후기 • xvi  
감사의 글 • xvii  
이 책에 대하여 • xviii

## CHAPTER 1 :: 엔터프라이즈 디자인

---

엔터프라이즈 아키텍처란 무엇인가? .....	5
엔터프라이즈 개발이란 무엇인가? .....	7
신뢰성 .....	8
유연성 .....	8
역할의 분리 .....	9
재사용성 .....	9
유지보수성 .....	9
마이크로소프트의 엔터프라이즈 기술은 어디에 있는가? .....	11
COM .....	13
자바로의 이동 .....	14
.NET .....	15
요약 .....	19

## CHAPTER 2 :: 엔터프라이즈 코드

---

코드를 바라보는 새로운 방법 .....	21
모듈화 .....	23
느슨하게 결합된 클래스들 .....	25
단위 테스트 .....	33
제어 역행화 컨테이너 .....	35
요약 .....	41

## CHAPTER 3 :: 보다 나은 클래스 디자인

---

코드의 의존성을 평가하기 .....	46
경직성 .....	59
유연성 .....	59

역할의 분리 .....	59
재사용성 .....	60
유지보수 용이성 .....	60
<b>역할의 분리와 모듈화의 이해 .....</b>	<b>62</b>
<b>의존성 전도의 원칙 .....</b>	<b>79</b>
<b>의존성 주입 패턴의 활용 .....</b>	<b>83</b>
경직성 .....	90
유연성 .....	90
역할의 분리 .....	91
재사용성 .....	91
유지보수 용이성 .....	91
<b>요약 .....</b>	<b>92</b>

## CHAPTER 4 :: 테스트 주도 개발

<b>TDD 예제: 틱택토 게임 .....</b>	<b>97</b>
틱택토 게임의 요구사항 .....	97
테스트 프레임워크 .....	140
테스트가 가능한 요소들을 정의하기 .....	141
유용한 단위 테스트 작성하기 .....	143
<b>리팩토링 .....</b>	<b>149</b>
<b>리팩토링 도구 .....</b>	<b>151</b>
ReSharper .....	151
Refactor Pro .....	151
<b>테스트 주도 개발 환경에서 의존성 처리하기 .....</b>	<b>151</b>
<b>가상 객체 프레임워크 .....</b>	<b>164</b>
Rhino Mocks .....	164
Moq .....	165
NMock .....	165
<b>요약 .....</b>	<b>165</b>

## CHAPTER 5 :: 간결한 코드를 위한 제어 역행화 기법

---

의존 객체의 생성 .....	169
Factory 패턴 .....	178
Service Locator .....	182
제어 역행화와 IoC 컨테이너 .....	185
의존성 주입과 제어 역행화의 비교 .....	187
IoC 컨테이너의 선택 .....	187
StructureMap 프레임워크 .....	192
Fluent Interface 패턴을 이용한 객체 연결 .....	193
Plugin Family 특성을 이용한 객체 연결 .....	197
메타 데이터를 이용한 객체 연결 .....	201
XML이냐 아니냐, 그것이 문제로다 .....	204
다른 IoC 프레임워크들 .....	205
요약 .....	206

## CHAPTER 6 :: 미들웨어 구축하기

---

미들웨어에서는 어떤 일이? .....	211
혼돈의 시대 .....	212
계층적 디자인 .....	213
인터넷 시대 .....	215
엔터프라이즈 미들웨어의 시대 .....	218
WCF 웹 서비스 .....	221
메시징 모델 .....	233
SOA에 대한 단상 .....	234
요약 .....	235

## CHAPTER 7 :: 미들웨어 구현하기

<b>비즈니스 로직 계층</b> .....	<b>237</b>
<b>비즈니스 로직 계층을 위한 패턴</b> .....	<b>239</b>
트랜잭션 스크립트 .....	239
액티브 레코드 패턴 .....	241
도메인 모델 패턴 .....	244
어떤 패턴을 사용해야 할까? .....	249
<b>서비스 계층의 구현</b> .....	<b>249</b>
서비스 계층 .....	250
패턴 실습하기 .....	251
모기지 론 자격 심사 애플리케이션 .....	252
도메인 언어로 의사소통하기 .....	253
<b>도메인 주도 디자인에 대한 간단한 소개</b> .....	<b>254</b>
엔티티 .....	254
값 객체 .....	255
객체 집합과 객체 집합 루트 .....	255
도메인 전문가와 대화하기 .....	256
도메인 모델의 구현 .....	259
객체 집합의 정의 .....	260
애플리케이션의 구현 .....	261
저장소 구현하기 .....	307
도메인 서비스의 구현 .....	309
<b>요약</b> .....	<b>316</b>

## CHAPTER 8 :: 비즈니스 구축하기

---

데이터 액세스 계층이란? .....	319
데이터 액세스 계층 구현하기 .....	320
객체 관계 매핑 .....	321
DataContext 객체 .....	322
Entity Framework .....	341
LinqToSQL과 Entity Framework .....	358
모기지 론 애플리케이션을 NHibernate를 이용하여 매핑하기 .....	377
요약 .....	401

## CHAPTER 9 :: 프론트 엔드의 구현

---

방치된 프론트 엔드 .....	403
초기의 프론트 엔드 패턴들 .....	405
자바 스트럿츠 .....	407
ASP.NET .....	409
모델-뷰-프리젠퍼 패턴 .....	413
MVC 패턴 다시 보기 - 레일즈 방식 .....	418
요약 .....	422

## CHAPTER 10 :: MVP 패턴

---

MVP 패턴에 대한 간단한 소개 .....	424
모델 .....	425
뷰 .....	425
프리젠퍼 .....	425
MVP 모기지 계산기 - 웹 예제 .....	426

플랫폼의 전환 - 팻 클라이언트 예제 .....	452
WPF 애플리케이션 프로젝트 준비하기 .....	452
요약 .....	457

## CHAPTER 11 :: MVC 패턴

MVC의 기초 .....	460
모델 .....	462
컨트롤러 .....	463
뷰 .....	463
모기지 론 애플리케이션 .....	464
모델 구현하기 .....	468
컨트롤러의 구현 .....	471
뷰 구현하기 .....	474
간단한 저장소 객체의 구현 .....	478
모델의 생성과 수정 .....	484
모델 확장하기 .....	503
전체 소스 코드 .....	516
요약 .....	517

## CHAPTER 12 :: 최종 실습

2보 전진을 위한 1보 후퇴 .....	520
우리가 학습했던 개념들 .....	520
신뢰성 .....	520
유연성 .....	521
역할의 분리 .....	521
재사용성 .....	521
유지보수성 .....	521
총평 .....	522

<b>우리가 작성했던 코드들</b> .....	<b>522</b>
총평 .....	523
<b>우리가 사용했던 패턴들</b> .....	<b>524</b>
미들웨어 .....	524
영속성 .....	526
사용자 인터페이스 .....	527
총평 .....	529
<b>전체적으로 다시 살펴보기</b> .....	<b>529</b>
모기지 서비스 .....	530
<b>최종 마무리</b> .....	<b>531</b>
<b>요약</b> .....	<b>532</b>

## 부록 A :: C#.NET 기초

---

<b>.NET의 기본 개념</b> .....	<b>534</b>
다중 언어 지원 .....	534
유연한 런타임 환경 .....	535
가비지 컬렉션 .....	535
COM의 쇠퇴 .....	536
C# 프로그래밍 .....	536
시스템 정의 타입들 .....	537
사용자 정의 타입들 .....	538
범위와 선언 .....	538
속성, 대리자 그리고 이벤트 .....	540
네임스페이스 .....	544
C# 3.0의 언어적 특징들 .....	547
묵시적으로 형식화된 지역 변수 .....	548
람다 식 .....	548
확장 메서드 .....	549
객체 및 컬렉션 초기자 .....	550

객체지향 개념 .....	552
클래스와 객체 .....	553
상속 .....	555
캡슐화 .....	557
다형성과 추상화 .....	560
<b>C#과 웹 SDK .....</b>	<b>563</b>
System.Web 네임스페이스 .....	563
System.Web.UI 네임스페이스 .....	564
System.Web.UI.WebControls 네임스페이스 .....	564
System.Web.UI.HtmlControls 네임스페이스 .....	565
System.Web.Services 네임스페이스 .....	565
System.Web.Security 네임스페이스 .....	565
System.Web.Mobile 네임스페이스 .....	566
System.Net.Mail 네임스페이스 .....	567
System.Web.Mvc 네임스페이스 .....	567
System.Web.Mvc.Ajax 네임스페이스 .....	567
System.Web.Mvc.Html 네임스페이스 .....	567
System.Data 네임스페이스 .....	567
<b>요약 .....</b>	<b>569</b>
<b>찾아보기 .....</b>	<b>571</b>

# 옮긴이 후기

.NET 프레임워크가 세상에 등장한 지도 어느새 10년의 세월이 흘렀다. 그 동안 .NET 프레임워크는 해외는 물론 국내 시장에서 활발히 그 영역을 넓혀왔으나 안타깝게도 국내 시장에서의 영향력은 해외 시장과 비교해 볼 때 훨씬 미치지 못하는 것이 사실이다. 여러 가지 원인이 있을 수 있지만 그 중에서도 가장 중요한 이유는 정보의 공유가 원활하지 않았다는 것이다. 사실 마이크로소프트의 MVP(Most Valuable Professional) 제도는 개발자 커뮤니티에 얼마나 많은 정보를 공유했는지가 중요한 지표 중 하나이다. 그만큼 해외 개발자들은 정보의 공유를 중요하게 생각한다는 반증이기도 하다. 우리가 가진 문제는 정작 중요하고 실질적인 정보보다는 단편적인 정보의 공유가 월등히 많다는 점이다.

그런 의미에서 이 번역서의 출간은 국내 .NET 개발자들에게 알찬 정보와 중요한 의미를 제공하고, 생각할 거리들을 던져주게 될 것이라고 믿는다. 솔직히 역자는 국내에 출간된 책 중 엔터프라이즈 개발, 더 폭넓게 이야기하자면 소위 말하는 중급 개발자가 소프트웨어 개발 업무를 수행하는 데 있어 반드시 익히고 고려해야 할 정보들을 이 정도로 쉽고 자세하게 설명한 책을 본 적이 없다.

한때 역자는 마이스페이스([www.myspace.com](http://www.myspace.com))라는 세계 최대의 SNS 서비스의 한국 지사에서 근무한 적이 있다. 당시 역자가 그들의 소스 코드를 통해 느낀 것은 우리가 지금까지 열심히 학습해 왔지만 실상은 어디에 어떻게 사용해야 할지 몰라 어쩔 줄 몰라 하던, 수많은 패턴들을 필요한 곳에 정확하게 구현하고 있었다는 점이다. 혹시 여러분이 지금 그렇다면 반드시 이 책을 읽어보기 바란다. 이 책을 통해 우리 나라의 .NET 개발자들도 코드를 작성하는 태도부터 습관, 이해력, 타인에 대한 배려심, 개발자로서의 역량 등이 한 단계 더 성숙해질 수 있기를 바란다. 적어도 이 책의 내용을 처음부터 끝까지 정독한다면 개발자로서의 기본 소양은 충분히 갖추게 될 수 있으리라 본다.

본론으로 들어가서 이 책은 소프트웨어 개발자가 하나의 소프트웨어를 구현해 나가는 데 필요한 배경 지식부터 범용적으로 활용되는 디자인 패턴과 그 사례, 그리고 완성된 소프트웨어까지 총망라하고 있다. 저자가 말했듯이 이 책은 처음부터 끝까지 정독을 해도 되고 어느 정도 수준에 오른 개발자라면 그저 옆에 두고 필요한 부분만 골라 읽어도 된다. 중요한 것은 한 장 한 장 읽어가면서 그 내용을 완벽하게 이해하고 실무에 적용하도록 노력해야만 한다는 점이다. 비단 이 책에만 해당되는 것은 아니겠지만 그렇게 함으로써 여러분은 이 책이 가진 가치를 피부로 느끼게 될 것이다.

지금까지 10권이 넘는 책을 집필/번역해온 역자에게 있어 이 책은 지금까지 역자가 출간해온 책 중에 가장 가치 있고, 영향력 있으며, 오래도록 소장할 가치가 있는 책이다. 이 책을 끝까지 읽은 뒤 여러분도 역자와 같은 생각을 하게 된다면 그것이야말로 역자에게는 보람이요, 자랑이 될 것이다.

마지막으로 이 책의 번역 기회를 주신 제이펍의 장성두 실장님께 깊은 감사의 말씀을 드리며, 밤낮으로 회사 업무와 번역에 매달려 있는 남편을 이해하고 기다려 준 사랑하는 아내, 그리고 예린이와 은혁이에게 진심어린 사랑을 담아 이 책을 바친다.

2010년 8월

웁긴이 장현희

## 베타리더 후기

**저는** 4년차 .NET 초급 개발자입니다. 아직 코딩도 미숙하지만 아키텍트에 관심이 있어 이 책의 베타리더를 신청했습니다. 처음에는 백지에서 시작해서 이해하기가 어려웠지만 순차적인 진행 방법과 내실 있는 예제로 인해 많은 도움을 얻었습니다. 특히 디테일한 예제를 사용함으로써 독자들의 이해력을 증강시키는 이 책을 저와 같은 미숙하지만 이 분야에 관심이 있는 초급 개발자에게 추천하고 싶습니다.

단, 손을 게을리 하는 개발자라면 이 책을 추천하고 싶지 않습니다. 책의 상당 부분이 예제로 되어 있어서 직접 따라서 해보지 않고는 초급 개발자로서는 이해하기가 다소 어려울 수 있기 때문입니다. 책을 읽다가 어렵거나 이해가 잘 되지 않는 부분이 나오면 예제를 통해서 실제적으로 어떻게 적용되는지를 알 수 있어 다른 책보다 이해하기가 수월하고 닷넷 엔터프라이즈 개발의 전체적인 그림을 그려 줄 수 있다는 점에서 닷넷 초중급 개발자들에게 이 책을 추천합니다.

마정건 m78mjg@hotmail.com

**이 책은** 평소 개인적으로 관심을 갖고 있던 여러 가지 주제들에 대하여 설계 및 예제들로 풀어서 설명하는 방식을 취하고 있습니다. 이러한 주제들을 유연하고 확장 가능한 코드로 작성하고 테스트하는 방법을 제안하고 있는데, 엔터프라이즈 아키텍처 및 개발에 관한 패턴, 방법론 등에 대한 포괄적인 안내 지침이 되지 않을까 생각해 봅니다. 반드시 코드를 따라해 보기를 권해드리며, 가치 있는 정보를 찾게 될 거라 생각합니다.

마이크로소프트 MVP로 5년간 활동해 오면서 다수의 책을 집필/번역하신 역자의 깔끔한 번역 또한 돋보입니다. 저의 짧은 경험으로 국내 번역서 중 이 정도의 번역 품질을 제공하는 책은 그다지 많지 않았으며, 그렇기에 쉽지 않은 내용을 쉽게 읽을 수 있어 더욱 좋았던 것 같습니다.

김용환 kimyh00@gmail.com

나는 이 책을 쓰면서 많은 사람들에게 도움을 받았다. 내가 바쁜 나날을 보내고 있는 동안 참고 이해해 준 Ed Connor와 Jim Minatel에게 감사의 말씀을 전한다. 공저자인 스콧 역시 나의 비전을 이해하고 이를 현실화하는 데 도움을 준 것에 대해 정말로 감사하고 있다. “스콧, 언젠가 필라델피아에 오면 미국에서의 생활을 맛보여 줄게요.” 토요일 밤, 바에서 이 책을 쓸 수 있도록 해 준 Vince, Sandra, John 그리고 Jug Handle Inn의 Ed에게 감사한다. 내가 항상 한결같은 수 있도록 곁에서 힘을 주신 어머니와 성질 더럽고 고지식한 괴짜이자 바보인 나를 다듬어주신 아버지에게 감사드린다. 무엇보다 아무런 불평 없이 나를 이해해준 사랑하는 나의 아내 Andrea와 나의 아이들, Emma 그리고 Jake에게도 고맙다는 말을 전한다.

존 아킹 Jon Arking

먼저 Wrox 출판사를 통해 이 책을 쓸 기회를 주고 나를 존에게 소개해준 Wiley 출판사의 Jim Minatel에게 감사의 말씀을 전한다. 이런 멋진 프로젝트에 참여할 기회를 주고 글을 쓰는 동안 물심양면으로 도와준 존에게도 감사한다. “존, 당신이 다시 영국을 방문한다면 영국의 전통적인 맛이 살아있는 선술집에 데려가 줄게요.” 기술 편집장인 Doug Parsons의 노고와 이 책의 책임 편집자라는 달갑지 않은 일을 선뜻 맡아주고 존과 내가 끝까지 이 일을 해낼 수 있도록 도와준 Ed Connor에게도 감사의 말씀을 전한다. 마지막으로 이 책을 쓰기 전에도, 쓰는 동안에도 그리고 탈고한 이후에도 많은 사랑과 지원을 아끼지 않았으며 내 인생에서 가장 소중한 것이 무엇인지 다시 한 번 깨닫게 해준 사랑하는 나의 아내 Lynsey에게 감사드린다.

스콧 밀렛 Scott Millett

# 이 책에 대하여

**컴퓨터** 프로그래밍과 비즈니스 소프트웨어 개발은 전혀 다른 일이다. 많은 사람들은 코드를 누가 작성하는지 혹은 누구를 위한 코드를 작성하든지에 관계없이 코드를 작성하는 일 자체는 동일하다고 생각하지만 사실은 그렇지 않다. 물론 비즈니스 소프트웨어 개발이라는 매력적이고 뭔가 더 나아보이는 일을 하기 위해서는 우선 코드 작성 실력을 더욱 향상시켜야겠지만 그것은 시작에 불과하다. 오늘날에는 비즈니스를 위한 소프트웨어를 개발하기 위해서는 여러 가지 언어에 대한 지식과 수양이 필요하다. 고수준 프로그래밍 기술과 저수준 프로그래밍 기술을 모두 연마해야 하는 것은 물론 복잡한 소프트웨어 디자인에 대한 경험도 필요하다. 무엇보다 인내와 새로운 개념에 대한 포용력이 가장 중요하다.

물론 우리의 생각에 동의하지 않는 사람도 많을 것이다. 우리 중 대부분은 복잡한 비즈니스 세계와는 담을 쌓은 채 골방에서 자신만의 작업에 몰두해야 최상의 결과를 만들 수 있는 개발자를 최소한 한두 명은 알고 있다. 어떤 경우에는 이런 방법이 좋을 수도 있다. 그러나 누구나가 회사 전체가 만족할 수 있는 소프트웨어를 만들고자 하지는 않는다. 소규모 애플리케이션 개발은 항상 커뮤니티 내에서 적절한 위치를 확보할 것이다. 이러한 소프트웨어들이 대규모의 엔터프라이즈 시스템으로 발전하면 여러분은 반드시 문제에 봉착하게 된다. 대체로 소규모 소프트웨어 개발을 위한 접근법은 놀랍도록 짧은 기간에 결과물을 창출하지만 그 기능과 사용법을 확장하기란 매우 어렵다. 기업에 필요한 엔터프라이즈 시스템을 구축할 때 엔터프라이즈 디자인에 대한 지식이 지속적으로 필요한 이유는 바로 이 때문이다.

잘 디자인된 시스템을 위한 노력은 언제나 새롭다. 컴퓨터 시대의 초기 IT 전문가들은 신속한 애플리케이션 개발과 소프트웨어 디자인 사이의 타협안을 마련하기 위해 열띤 논쟁을 벌였다. 이는 최상의 디자인과 개발자를 위한 생산성 사이에서 벌어진 진정한 논쟁이었다. 비즈니스가 IT와 더욱 가까워지면서 비용 절감에 대한 기대와 함께 엔터프라이즈 디자인에 대한 선행 투자를 시작했다. 그러한 가능성이 현실화되면서 개발자들은 클라이언트의 요구에 적합한 엔터프라이즈 패턴을 올바르게 적용할 수 있는 방법에 대해 알아야만 했다.

마이크로소프트 개발자들은 이와 같은 상황에 익숙하다. 플랫폼이 거의 독점적으로 신속한 애플리케이션 개발에 초점을 맞추고 있기에 마이크로소프트 애플리케이션은 오랫동안 형편없는 디자인이 적용되어 왔으며, 확장성과 유연성은 시장의 빠른 대응을 위해 뒷전으로 밀려나 있었다. 엔터프라이즈 방법론이 더욱 광범위하게 적용됨에 따라 마이크로소프트 기술을 다루는 새로운 세대의 개발자들은 이러한 새로운 패턴에 대한 학습이 필요하며, 이러한 패턴을 자신들의 기술과 애플리케이션에 적용해야 하는 상황에 직면했음을 깨달았다. 잘 디자인된 소프트웨어와 애플리케이션의 신속한 출시에 대한 수요 사이의 불합리한 격차는 수년간 마이크로소프트 커뮤니티를 괴롭혔다. 새롭게 학습해야 할 엔터프라이즈 패턴이 너무도 많았기 때문에 마이크로소프트 개발자들은 이러한 패턴들을 익히고 코드를 변경하는 최적의 방법에 대한 가이드가 필요했다.

이 책 『프로페셔널 엔터프라이즈 닷넷』은 최신의 엔터프라이즈 개발 방법론을 학습하고자 하는 마이크로소프트 개발자를 위한 최선의 가이드를 제공한다. 이 책의 목표는 개발자들이 서로 다른 패턴들을 이용하여 코드를 더욱 간결하고 유지보수가 용이하도록 구현하는 방법을 교육하는 것이다. 또한 자신들의 애플리케이션뿐만 아니라 개발 기술을 보다 새롭고 유연한 엔터프라이즈 방법론에 따라 발전시킬 수 있는 방법을 찾는 중/고급 마이크로소프트 개발자들에게 로드맵을 제시하고자 한다.

## 이 책이 다루는 범위

이 책은 대중적인 소프트웨어 개발 패턴과 방법론에 대해 소개하는 책이며, 이러한 패턴과 방법론을 마이크로소프트 애플리케이션 개발에 대한 경험을 가진 독자, 특히 C#과 ASP.NET 개발 경험을 가진 개발자들을 위주로 소개한다. 이 책은 엔터프라이즈에 필요한 모든 내용들을 소개하는 책은 아니다. 엔터프라이즈 디자인은 너무나도 광범위한 주제이며, 각각의 하위 주제만으로도 책 한 권 분량은 족히 넘는 것들이다. 대신 이 책은 유용하게 활용할 수 있는 광범위한 주제들을 포괄적인 시각에서 다룬다. 테스트 주도 개발이나 미들웨어 디자인 패턴 혹은 ASP.NET에서의 웹 MVC 패턴의 활용 등과 같은 주제에 관심이 있는 독자라면 이러한 주제들을 전문적으로 다루는 다른 서적을 참고하기를

권한다. 이 책은 엔터프라이즈 개념에 대한 설명을 거쳐 테스트 가능한 코드를 작성하는 방법론과 디자인 패턴에 대한 소개로 시작하기 때문에, 여러분은 코드를 느슨하게 결합시키며 테스트 가능한 형태로 구성할 수 있는 조금 다른 방법에 익숙해지게 될 것이다. 그런 후에는 Spring.NET이나 NHibernate, ASP.NET MVC 등 이러한 방법론들이 적용된 다양한 도구에 대해 학습하게 된다. 이러한 주제들을 조합함으로써 개발자들이 서로 다른 패턴을 결합하여 활용하는 방법을 학습할 수 있도록 하며, 궁극적으로는 개발자들은 자신들의 작업에 가장 적합한 것들을 찾아낼 수 있게 된다. 이 책은 이야기를 전개해 나가는 스타일이지만 엔터프라이즈 개발자가 되고자 하는 독자들에게는 유용한 참고서적이 될 수 있다. 이 책의 각 장들은 읽는 순서에 관계없이 읽을 수 있도록 독자적인 내용들로 구성되어 있다. 이러한 점을 염두에 두고 각 장의 주제들에 대해 살펴보자.

## 제1장 엔터프라이즈 디자인

이 책은 엔터프라이즈 개발의 핵심 개념에 대한 설명으로 시작한다. 이러한 핵심 개념들은 실질적인 코딩 양을 줄이는 동시에 엔터프라이즈 디자인의 철학을 더하기에 엔터프라이즈 디자인을 적용해야 하는 이유가 된다. 제1장은 엔터프라이즈 아키텍처의 개념에서 출발하여 그 의미는 무엇이며 누가 적용해야 하는지에 대한 정의를 설명하는 데 상당한 지면을 할애한다. 그런 후 안정성, 유지보수 용이성 및 역할의 분리와 같은 엔터프라이즈 개발의 핵심 가치들을 토대로 엔터프라이즈 개발의 개념에 대해 학습한다. 이후에는 소프트웨어 디자인의 역사를 거슬러 올라가 이러한 핵심 가치들이 어떻게 발전해 왔는지를 독자들에게 설명한다. 마지막으로 엔터프라이즈 개발자들이 반드시 활용해야 할 유명한 도구들에게 대해 설명한다.

## 제2장 엔터프라이즈 코드

제2장에서는 엔터프라이즈 개발과 관련된 새로운 코딩 개념을 소개한다. 모듈화나 느슨한 결합과 같은 개념들을 몇 가지 간단한 예제를 통해 상세히 설명한다. 단위 테스트는 엔터프라이즈 아키텍처에서는 매우 권장되는 개념이며, 이 개념은 모듈화 및 느슨한 결합과 같은 문맥 내에서 세심하게 설명하고 있다. 제2장에서는 제어 역행화에 대해서도 설명하고 있으며, 테스트가 용이한 기반 코드를 구축하기 위해 제어 역행화를 통해 객체 인스턴스를 생성하는 코드를 유지보수가 용이하도록 만드는 방법을 소개한다. 마지막으로 엔터프라이즈 코딩을 위해 필요한 NUnit이나 Resharper 및 Spring.NET과 같은 도구들을 소개한다.

## 제3장 클래스의 해방

제3장에서는 의존성 전도 개념에 초점을 맞춘다. 이 원칙은 상위 수준과 하위 수준 모듈들의 관계를 뒤집어 실제 구현에 있어 이들 사이의 의존도를 낮춘다. 제3장에서는 하위 수준의 객체를 상위 수준의 객체에 주입하는 의존성 주입의 형태로 의존성 전도 원칙을 활용하는 방법을 소개한다.

## 제4장 테스트 주도 개발

제4장에서는 느슨하게 결합되며 고도의 테스트 용이성을 확보한 코드를 구현하기 위한 테스트 주도 개발 방법론을 소개한다. 제4장에서는 복잡한 예제를 통해 단위 테스트를 토대로 애플리케이션의 디자인을 이끌어 내기 위한 방법을 보여주기 위해 TDD를 활용한다. 또한 단위 테스트를 생성하는 과정에 대해서도 설명하며, TDD에 대한 순수한 접근법과 현실적인 애플리케이션 개발 방법론 사이의 격차를 절충하는 방법에 대해서도 설명한다. 마지막으로는 비용이 큰 자원에 대한 의존성을 가진 모듈의 단위 테스트에 대해 설명한다. 이로 인해 가상 객체에 대한 개념이 등장하며, 대중적인 가상 객체 프레임워크를 활용하는 몇 가지 예제를 소개한다.

## 제5장 간결한 코드를 위한 제어 역행화 기법

제5장에서는 제3장에서 설명했던 개념들을 엔터프라이즈 개발을 보다 쉽게 만들어 주는 도구와 패턴을 중심으로 다시 살펴본다. 우선은 팩토리 패턴과 서비스 로케이터 패턴의 장단점을 살펴본 후 절차적 프로그래밍과는 반대로 시스템의 흐름이 전도되는 디자인을 설명하는 추상적인 원칙인 제어 역행화 원칙에 대해 소개한다. 마지막으로 독자적인 제어 역행화 컨테이너를 구현해 보고 유명한 오픈 소스 제품인 StructureMap에 대해 간략히 소개한다.

## 제6장 미들웨어 구축하기

제6장에서는 엔터프라이즈 개발과 관련된 핵심 개념과 코딩 원칙에서 벗어나 패턴에 대해 학습한다. 제6장에서는 잘 디자인된 엔터프라이즈 시스템에 필요한 대중적인 디자인 패턴과 프레임워크들에 대해 소개한다. 또한 미들웨어 개념이 최초로 소개된다. 우선 메인프레임 시스템 모델에서 시작하여 클라이언트-서버 아키텍처를 거쳐 웹 개발 환경에 이르기까지 계층화 디자인의 역사에 대해 살펴본다. 그런 후에는 오늘날 분산 시스템에서 활용되는 서비스와 메시지 기반 미들웨어를 구현하는 몇 가지 디자인 패턴에 대해 살펴본다.

## 제7장 미들웨어 구현하기

제7장은 애플리케이션에 구현하는 비즈니스 로직에 초점을 맞춘다.

독자 여러분은 제7장에서 트랜잭션 스크립트, 액티브 레코드 및 도메인 모델 패턴 등 세 가지 유명한 미들웨어 패턴에 대해 학습하게 된다. 그런 후에는 최근 급속히 대중화되고 있는 디자인 방법론인 도메인 주도 디자인에 대해 간략하게 살펴본다. 이 최신의 패턴은 애플리케이션의 비즈니스 로직을 위한 것으로 비즈니스 로직을 기술적인 인프라스트럭처와 철저히 분리한다.

제7장의 후반부에서는 가상의 모기지 론 승인 애플리케이션을 위한 도메인 모델을 구성하며, 이 프로젝트는 나머지 장에서도 계속해서 활용

된다. 독자는 도메인 주도 디자인의 몇 가지 핵심 원칙을 활용하여 요구사항을 수집하고, 도메인 전문가와 대화하며, 테스트 주도 개발 원칙에 따라 도메인 모델을 구현한다.

## 제8장 비즈니스 구축하기

제8장에서는 영속성에 대한 개념을 소개한다. 이 장에서는 데이터 액세스 계층의 역할과 이를 ADO.NET을 이용해 구현하는 전통적인 접근법을 간략하게 소개한다. 그 후 객체 관계 매핑에 대해 소개하며, 이러한 프레임워크를 사용하는 방법과 개발자가 이를 직접 구현하는 방법 사이의 장단점을 비교한다.

그런 후에는 영속성 관리를 위한 두 가지 방법에 대해 학습하게 된다. 첫 번째는 마이크로소프트의 LinqToSql을 이용한 데이터 모델 접근법이며, 두 번째는 마이크로소프트의 Entity Framework 객체 관계 매핑을 이용하여 데이터 액세스 객체 패턴을 구현하는 방법을 살펴보게 된다.

제8장의 마지막 부분에서는 제7장에서 NHibernate 프레임워크를 이용해 구현했던 저장소 계층을 이용하여 모기지 론 신청 애플리케이션을 구현한다.

## 제9장 프론트 엔드의 구현

시스템 디자인에 대한 설계에 이어서 제9장에서는 사용자 인터페이스 개발의 세계로 안내한다. 제6장과 마찬가지로 제9장에서는 사용자 인터페이스 디자인과 프로그래밍의 역사에 대한 설명으로 시작한다. 독자들은 신속한 애플리케이션 디자인과 이를 위한 몇 가지 도구들이 미친 영향에 대해 학습하게 된다. 그런 후 끌어다 놓기 방식의 사용자 인터페이스 디자인과 웹 프로그래밍으로의 진화의 시기를 거쳐 최근 각광받으며 궁극적으로 엔터프라이즈 시스템 내에서 고유의 길을 개척하고 있는 모델-뷰-컨트롤러 패턴과 같은 대중적인 디자인 패턴들에 대해 학습하게 된다.

## 제10장 MVP 패턴

제10장에서는 모델-뷰-프리젠티 패턴을 이용한 심도 있는 사용자 인터페이스 디자인에 대한 이야기가 펼쳐진다. 테스트가 가능한 사용자 인터페이스를 구현하기 위해서는 느슨한 결합 패턴을 활용해야 한다. 그러나 이런 패턴을 애플리케이션에 미치는 영향을 최소화하면서 기존의 사용자 인터페이스 코드에 적용하는 것은 매우 어려운 일이다. 올바른 패턴을 적용하면 아무런 문제없이 사용자 인터페이스 코드에 대한 단위 테스트를 수행할 수 있다. 이런 것이 바로 MVP 패턴의 목적이다. MVP 패턴은 두말할 필요 없이 엔터프라이즈 사용자 인터페이스 디자인 분야에서는 가장 대중적인 패턴이며, 이를 통해 사용자 인터페이스 코드를 실제 요소들과 분리할 수 있다. 제10장에서는 MVP 패턴을 자세히 설명하고 개별적인 컴포넌트들과 이들 사이의 상호 작용 그리고 이들이 제공하는 다양한 테스트 가능성에 대해 살펴본다. 또한 웹과 팻 클라이언트를 위한 간단한 모기지 계산기를 MVP 패턴을 토대로 구현하는 간단한 예제를 제공한다.

## 제11장 MVC 패턴

제9장과 제10장에서 제시했던 주제에 이어 제11장에서는 모델-뷰-컨트롤러 패턴에 대해 심도 있게 살펴본다. 우선은 MVC 패턴의 역사에서 Ruby on Rails 프레임워크와 이 프레임워크가 미친 영향에 대해 간략히 살펴본다. 다음으로 MVC 패턴을 구성하는 핵심 컴포넌트들과 그 역할에 대해 설명한다. 그리고 최근 마이크로소프트가 Windows 플랫폼 기반의 웹 개발자를 위해 배포한, 웹을 위한 MVC 프레임워크인 ASP.NET MVC에 대해 소개한다. 그 후에는 웹 MVC 모델의 장단점을 제10장에서 소개한 MVP 패턴과의 비교를 통해 살펴본다. 마지막으로 제7장과 제8장 그리고 제10장의 내용을 토대로 모기지 론 모델의 코드 예제를 구현한다.

## 제12장 최종 실습

제12장은 이 책의 마지막 장으로 지금까지 논의했던 모든 개념과 방법론, 그리고 디자인 패턴 등이 복합적으로 활용된다. 제1장에서 설명했던 엔터프라이즈 디자인의 핵심 가치를 다시 한 번 되짚어보고 이전의 각 장들이 소개했던 내용들과 주요 요점들, 그리고 이들과 관련된 엔터프라이즈 시스템의 목표에 대해 요약한다. 또한 적절한 타입의 시스템에 올바른 패턴을 적용하는 방법과 시스템의 요구사항에 적절한 프레임워크와 모델을 선택하는 방법에 대해 설명한다. 마지막으로 이 책이 소개했던 대부분의 주제와 패턴들을 사용한 폭넓고 다면적인 모기지 신청서 애플리케이션을 살펴본다.

## 이 책의 구성

이 책은 단계별 학습 가이드이자 지속적으로 참고할 수 있는 두 가지 역할을 모두 제공하는 책이다. 이 책은 개별적인 섹션으로 분리되어 독자는 전체 내용을 모두 읽거나 혹은 필요한 부분만 골라서 읽을 수 있다. 이 책의 첫 번째 파트에서는 엔터프라이즈 개발에 내포된 철학에 대해 설명한다. 두 번째 파트에서는 코딩 패턴에 대해 심도 있게 학습한다. 이 과정에서 느슨한 결합의 개념과 기존의 코드를 분리하는 최적의 방법, 그리고 테스트 주도 디자인의 이점에 대해 학습하게 된다. 세 번째 파트에서는 엔터프라이즈 시스템에서 사용되는 보다 일반적인 디자인 패턴들을 광범위하게 다룬다. 이 과정에서는 미들웨어를 구축하기 위한 보편적인 방법과 데이터의 최소화 및 영속성 기법, 엔터프라이즈 UI 디자인의 배경 등에 대해 학습한다. 각 장의 내용들은 먼저 주제에 대해 설명한 후 관련된 코드 예제를 살펴보는 형식으로 진행된다. 코드 예제는 한 장의 내용만으로 구성되기도 하지만 보다 완성도 높은 시스템을 구현하기 위해 몇 개의 장에 걸쳐 구현되기도 한다.

## 이 책에 적용된 규칙

독자들이 책의 내용을 더욱 잘 이해할 수 있도록 돕기 위해 이 책에는 다음과 같은 몇 가지 규칙이 적용되어 있다.

저자가 덧붙이는 말이나 힌트, 편법 및 본문의 내용에 대한 첨언 등은 본문과 약간 띄우고 고딕체로 표기한다.

본문에 적용된 글꼴의 스타일은 다음과 같다.

- 새로운 용어와 중요한 단어는 고딕체로 표기한다.
- 키보드의 단축키는 Ctrl+A와 같이 표기한다.
- 파일명, URL 및 코드는 `persistence.properties`와 같이 Consolas 글꼴로 표기한다.
- 코드는 다음과 같이 두 가지 방식으로 표기한다.

그다지 중요하지 않은 대부분의 코드 예제는 아무런 표시 없이 표기한다.

현재 문맥상 의미가 있는 코드는 회색 배경을 삽입하여 표기한다.

## 소스 코드

독자들은 이 책의 예제 코드를 따라하면서 전체 코드를 직접 입력하거나 혹은 이 책이 제공하는 소스 코드 파일을 사용할 수도 있다. 이 책에서 사용된 모든 소스 코드는 [www.wrox.com](http://www.wrox.com)에서 다운로드할 수 있다. 이 사이트에서 (검색 상자나 도서 목록에서) 이 책의 제목을 선택하고 상세 보기 페이지에서 [Download Code] 링크를 클릭하면 소스 코드를 다운로드할 수 있다(출판사\_제이펍 블로그 <http://www.jpub.kr>의 이 책 소개 페이지에서도 제공합니다).

유사한 제목의 다른 책들이 많기 때문에 ISBN을 이용해 검색하는 것이 가장 손쉬운 방법이다. 이 책의 ISBN은 978-0-470-44761-1이다.

또한 Wrox의 코드 다운로드 페이지([www.wrox.com/dynamic/books/download.aspx](http://www.wrox.com/dynamic/books/download.aspx))에서 이 책과 다른 Wrox 도서들의 소스 코드를 확인할 수 있다.

## 정오표

우리 저자들은 본문이나 소스 코드에 오류가 없도록 하기 위해 최선의 노력을 기울였다. 그러나 완벽한 사람은 아무도 없기 때문에 언제든지 실수는 일어날 수 있다. 만일 이 책의 내용에서 오타자나 코드의 오류와 같은 문제를 발견한다면 우리 저자들에게 연락해 주기 바란다. 정오표를 공유함으로써 여러분은 다른 독자들의 불만을 줄이는 동시에 보다 고품질의 정보를 제공할 수 있도록 우리를 도와줄 수 있다.

이 책의 정오표는 [www.wrox.com](http://www.wrox.com)에서 검색 상자에 책 제목을 입력하여 검색하거나 도서 목록에서 책을 찾은 후 도서 상세 정보 페이지에서 [Book Errata] 링크를 클릭하면 확인할 수 있다. 이 페이지에서는 Wrox의 편집자가 올려둔 모든 정오표를 확인할 수 있다. 모든 도서의 정오표 페이지에 대한 링크를 모아둔 페이지는 [www.wrox.com/misc-pages/booklist.shtml](http://www.wrox.com/misc-pages/booklist.shtml)에서 확인할 수 있다.

만일 여러분이 발견한 오류가 정오표 페이지에 존재하지 않는다면 [www.wrox.com/contact/techsupport.shtml](http://www.wrox.com/contact/techsupport.shtml) 페이지를 방문하고 양식을 작성하여 여러분이 발견한 오류를 우리에게 알려주면 된다. 보고된 내용을 검토하여 오류가 맞다면 여러분이 보내준 정보를 정오표 페이지에 등록하고 다음에 출간될 책에서 오류를 수정할 것이다(출판사\_제이펍 블로그 <http://www.jpub.kr>의 정오표 카테고리에서 확인할 수 있으며, 오류에 관한 새로운 정보는 다음의 이메일로 보내주시기 바랍니다. 출판사: [jeipub@gmail.com](mailto:jeipub@gmail.com) 역자: [geniex@msn.com](mailto:geniex@msn.com)).

## p2p.wrox.com

저자와의 직접적인 교류를 원한다면 [p2p.wrox.com](http://p2p.wrox.com)에서 제공하는 P2P 포럼에 참여하기 바란다. 이 포럼은 웹 기반의 시스템으로 관련된 Wrox의 책이나 기술들에 대해 다른 독자들과 논의하기 위한 메시지를 작성할 수 있다. 이 포럼은 전자 메일을 이용한 구독 기능을 제공하기 때문에 여러분이 관심있는 포럼에 새로운 글이 등록되면 곧바로 알 수 있다. Wrox의 저자와 편집자, 다른 산업 전문가 및 다른 독자들이 모두 이 포럼에서 활동하고 있다.

<http://p2p.wrox.com>에서는 여러분이 읽고 있는 이 책뿐만 아니라 여러분의 애플리케이션을 구현하는 데 도움을 얻을 수 있는 여러 가지 포럼들이 제공된다. 포럼에 참여하려면 아래의 절차를 밟으면 된다.

1. [p2p.wrox.com](http://p2p.wrox.com)을 방문하고 [Register] 링크를 클릭한다.
2. 최종 사용권 계약 동의서를 읽고 [Agree] 버튼을 클릭한다.
3. 시스템이 필요로 하는 정보 및 여러분이 추가로 제공하고자 하는 정보를 입력한 후 [Submit] 버튼을 클릭한다.
4. 계정을 인증하기 위한 전자 메일이 발송되며 이를 통해 가입 단계를 완료할 수 있다.

P2P포럼에 가입하지 않아도 포럼의 글들을 읽을 수는 있지만 직접 글을 등록하려면 반드시 가입을 해야 한다. 일단 가입하게 되면 새로운 글을 등록하거나 다른 사람의 글에 대해 답변을 등록할 수 있다. 또한 웹을 통해 언제든지 포럼의 글들을 확인할 수 있다. 특정 포럼에 새로운 글이 등록될 때마다 메일을 받고 싶다면 포럼 목록에서 포럼의 제목 옆에 있는 [Subscribe to this Forum] 아이콘을 클릭하면 된다.

Wrox P2P 포럼을 사용하는 보다 자세한 내용은 P2P FAQ 페이지를 통해 포럼이 동작하는 방식과 P2P 포럼 및 Wrox의 도서들과 관련된 질문에 대한 답변을 읽어보기 바란다. FAQ 페이지는 P2P 포럼 페이지의 FAQ 링크를 통해 열람할 수 있다.

## 기대 효과

만일 여러분이 개발에 매우 능하며 블로그와 포럼에 많은 시간을 할애하는 오픈 소스 전문가라면 지금 당장 이 책을 내려놓기 바란다. 이 책은 엔터프라이즈의 모든 것을 다루는 완벽한 참고서가 아니다. 이 책은 테스트 우선 방법론을 강요하지 않는다. 또한 애자일 선언문을 내세우지도 않으며 엔터프라이즈 전문가에게 어필하는 책도 아니다.

이 책은 독자들에게 좋은 시스템 디자인의 가치를 소개하고 대형 시스템 프로젝트에 이러한 디자인을 어떻게 적용하는지를 알리는 책이다.

우리가 엔터프라이즈/오픈 소스 커뮤니티의 반감을 사는 것을 원하는 것은 아니지만 이 책이 그들을 제외한 나머지 프로그래밍 커뮤니티를 대상으로 한다는 것이 중요하다. 비록 이 책은 마이크로소프트 개발자들에 초점을 맞추고는 있지만 이 책이 설명하는 경험이나 방법론은 모든 종류의 기술적 환경을 토대로 하는 모든 소프트웨어 개발에 적용될 수 있다. 많은 도구와 프레임워크들은 자바나 Ruby on Rails와 같은 다른 개발 플랫폼에서 차용되었다. 마이크로소프트도 몇 가지 도구들을 제공하지만 이들에 대한 언급은 시애틀 레드몬드에서 있는 '모션'을 지원하기 위한 것이라기보다는 좋은 소프트웨어 디자인을 위한 것이다. 이 책의 저자들은 오픈 소스 커뮤니티에서 활발히 활동하고 있다. 두 저자는 모두 어떤 문제를 해결하기 위한 최선의 해결책을 하나의 회사가 제공할 수 없다고 굳게 믿고 있다.

그 후 몇 달간 우리는 유사한 주제를 다룬 수많은 책들을 발견했으며, 이들 중 대부분은 마이크로소프트 솔루션을 위한 것들이었다. 이 책은 우선 좋은 시스템 디자인의 본질에 대해 이해하고 그런 후 여러분의 필요에 맞는 적절한 방법을 찾도록 한다. 우리는 마이크로소프트 기술의 강점과 약점을 모두 다룬다. 서로 다른 도구들의 장단점을 파악하는 것은 여러분의 프로젝트에 적절한 결정을 내리는 데 도움이 되며, 궁극적으로는 보다 나은 시스템 디자인을 이끌어 낼 것이라 믿는다.