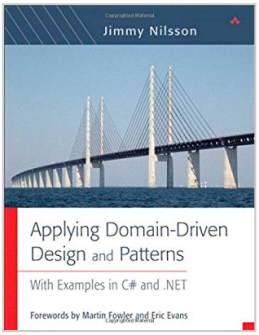

Applying Domain-Driven Design and Patterns: With Examples in C# and .NET

By Jimmy Nilsson

Applying Domain-Driven Design and Patterns: With Examples in C# and .NET By Jimmy Nilsson

Patterns, Domain-Driven Design (DDD), and Test-Driven Development (TDD) enable architects and developers to create systems that are robust and maintainable. While the examples in this guide are in C# and .NET, the principles can be used by developers using any language and IDE.

Applying Domain-Driven Design and Patterns: With Examples in C# and .NET

By Jimmy Nilsson

Applying Domain-Driven Design and Patterns: With Examples in C# and .NET By Jimmy Nilsson

Patterns, Domain-Driven Design (DDD), and Test-Driven Development (TDD) enable architects and developers to create systems that are robust and maintainable. While the examples in this guide are in C# and .NET, the principles can be used by developers using any language and IDE.

Applying Domain-Driven Design and Patterns: With Examples in C# and .NET By Jimmy Nilsson Bibliography

- Sales Rank: #1126346 in Books
- Published on: 2006-05-18
- Original language: English
- Number of items: 1
- Dimensions: 9.51" h x 1.42" w x 7.32" l, 2.41 pounds
- Binding: Hardcover
- 576 pages

Preface: Bridging Gaps

On the cover of this book is a picture of the Øresund Bridge that connects Sweden and Denmark. It seems that all software architecture books must have a bridge on the cover, but there are some additional reasons the bridge is appropriate for this book.

This bridge replaced a ferry that I took many times as a child. I enjoy very much driving over it even after dozens of times.

On a personal note, my father was on the team that built the highest parts of the bridge.

But beyond these, the main reason is that this book is very much about bridging gaps; bridging gaps between users and developers; bridging gaps between business and software; bridging gaps between logic and storage. Bridging gaps between "DB-guys" and "OO-guys"...

I will refrain from making a joke about the Bridge pattern GoF Design -Patterns. Hey, how geeky can a preface be?

Focus of This Book

The main focus of the book is how a Domain Model could be constructed to be clean, yet still be persistence-friendly. It shows what the persistence solution could look like for such a Domain Model and especially how to bridge that gap between the Domain Model and the database.

Put another way, my vision has been to provide a book that will put Eric Evans' *Domain-Driven Design* Evans DDD and Martin Fowler's *Patterns of Enterprise Application Architecture* Fowler PoEAA in context.

DDD might be perceived as a bit abstract. Therefore, more concrete examples are helpful regarding persistence, for example. Mine may be fairly basic, but it is a platform to start from. This book not only explains how to use the patterns, but also how the patterns are used in O/R Mappers, for example.

It has become very clear to me that "one size does not fit all" when it comes to architecture. Having said that, patterns have proven to be general enough to use and reuse in context after context.

The focus isn't on the patterns themselves, but this book uses patterns in every chapter as a tool and language for discussing different design aspects. A nice side effect is that patterns-ignorant readers will also gain some insight and interest into patterns along the way.

That also goes for TDD. Not all developers have become interested in this yet. I think it's especially common in the .NET community that TDD (just as patterns) is considered a niche technique at best, or it might even be totally unknown. Readers will learn how to apply TDD.

Why This Book?

Writing my first book Nilsson NED was a really tough project on top of all my other ordinary projects and obligations. I was pretty sure I wouldn't write another, but the time came when I thought I had something to say that I couldn't leave unsaid.

My change of heart started when I read two recent books that inspired me and changed my thinking. First, there was Martin Fowler's *Patterns of Enterprise Application Architecture* Fowler PoEAA. This book

inspired me to give the Domain Model pattern another try after having failed with several earlier attempts.

Then I read Eric Evans' book *Domain-Driven Design* Evans DDD. This book provided me with insights about how to think and act regarding development with a strong domain focus and with a certain style of how to apply the Domain Model pattern.

Another important influence was all that I learned from teaching my patterns course over a couple of years. As I interacted with students and the material evolved, I had insights myself.

My views of DDD transformed as I worked on an ambitious (though unfortunately unfinished) open source project called Valhalla, which I developed in collaboration with Christoffer Skjoldborg. (Christoffer did by far the most work.)

To summarize all this, I felt that a book that dealt more with application than theory was needed, but one that was based on solid ground, such as the DDD and PoEAA books. "Applying" feels close to my heart because I consider myself a developer above anything else.

Target Audience

This book is aimed at a wide target audience. It will help if you have *some* knowledge of

- Object-orientation
- .NET or a similar platform
- C# or a similar language
- Relational databases; for example, SQL Server

However, interest and enthusiasm will compensate for any lack of prior experience.

I'd like to elaborate on my statement that the target audience is wide. First, we can think about the way we put people into platform boxes. The book should serve .NET people who want a more core-based approach than drag-till-you-drop (if I may use some weak generalizations). Java people should get something out of the discussions and examples of how to combine DDD and O/R Mapping.

I think the chosen language/platform is less and less important, so it feels a little strange to talk about .NET people and Java people. Let's try to describe the target audience by using another dimension. Then I think that the book is for developers, team leaders, and architects.

Choosing yet another dimension, I think there might be something in this book both for intermediate and advanced readers. There's probably also something for beginners.

Organization of This Book

The book is arranged in four parts: "Background," "Applying DDD," "Applying PoEAA," and "What's Next?"

Part I: Background

In this part, we discuss architecture and processes in general terms. There is a lot of emphasis on Domain Models and DDD Evans DDD. We also introduce patterns and TDD. The chapters include the following:

- Chapter 1, "Values to Value"This chapter discusses properties of architecture and process to value for

creating quality results when it comes to system development. The discussion is influenced by Extreme Programming.
- Chapter 2, "A Head Start on Patterns"This chapter focuses on providing examples and discussions about patterns from different families, such as design patterns, architectural patterns and domain patterns.
- Chapter 3, "TDD and Refactoring"Chapter 1 talks quite a lot about TDD and refactoring, but in this chapter there is more in-depth coverage with pretty long examples and also different flavors of TDD.

Part II: Applying DDD

In this part, it's time to apply DDD. We also prepare the Domain Model for the infrastructure, and focus quite a lot on rules aspects.

- Chapter 4, "A New Default Architecture"This chapter lists a set of requirements of an example application, and a first-try model is created as a start for the coming chapters. A Domain Model-based architecture is used.
- Chapter 5, "Moving Further with Domain-Driven Design"The requirements set up in the prior chapter are used in this chapter as the basis for slowly, with TDD, starting to build the Domain Model in a DDD-ish style.
- Chapter 6, "Preparing for Infrastructure"Even though we try to push the infrastructure aspects as far off in the future as possible, it's good to think a little bit ahead and prepare the Domain Model for the infrastructure needs. In this chapter, there is a lot of discussion about pros and cons of Persistence Ignorant Domain Models.
- Chapter 7, "Let the Rules Rule"This chapter talks about business rules in the form of validation and how a Domain Model-based solution can deal with the need for such rules, connecting back to the requirements set up in Chapter 4.

Part III: Applying PoEAA

In this part, we put several of the patterns in Fowler's *Patterns of Enterprise Application Architecture* Fowler PoEAA into context by discussing what we need from the infrastructure for providing persistence support to our Domain Model. We will take a look at how those requirements are fulfilled by an example tool.

- Chapter 8, "Infrastructure for Persistence"When we have a fairly good Domain Model, it's time to think about infrastructure, and the main type of infrastructure in this book is infrastructure for persistence. This chapter discusses different properties of persistence solutions and how to categorize a certain solution.
- Chapter 9, "Putting NHibernate into Action"This chapter uses the categorizations of the prior chapter with an example of a persistence solution, namely NHibernate NHibernate.

Part IV: What's Next?

In this part, there is a focus on other design techniques to keep an eye on and start using. The other focus is on how you can deal with the presentation layer when it comes to bridging that gap to the Domain Model, but also how to deal with developer testing of the UI. This part is almost exclusively written by guest authors.

- Chapter 10, "Design Techniques to Embrace"After a short discussion about Bounded Context, this chapter discusses design techniques to keep an eye on now and for the future, such as Service Orientation, Dependency Injection/Inversion of Control, and Aspect Orientation.
- Chapter 11, "Focus on the UI"This chapter focuses on how the UI can be connected to the Domain Model and how to increase testability for the user interface when using a Domain Model, both for rich client applications and Web applications.

Appendices

There are two appendices providing further examples of Domain Model styles and an overview-type patterns catalog.

Why C# for the Examples?

In no way is this a book for teaching C#. But I still need a language (or possibly several, but I have chosen one) for the examples, and that's where C# comes in.

The reasons for the choice are mainly that C# is my current main language and that most VB.NET and Java developers can read C# code pretty easily.

Regarding the version, most of the code examples work in both C# 1.1 and 2.0, but there are some rare sections that are focused on 2.0.

Topics That Aren't Covered

There are loads of topics that aren't covered in the book, but I think there are two missing topics that spring to mind. They are distribution and advanced modeling.

Distribution

It was in my early plans of the book to include thorough coverage of the distribution aspects, but later on I came to the conclusion that the book would become too unfocused. Still, there is some coverage here and there.

Advanced Modeling

The title of the book might suggest that you find advanced and interesting examples of modeling of certain problem areas. That's not exactly the case. Instead, the application focus is more about applying TDD and adding infrastructure to DDD.

finally{}

As you can appreciate, a book project like this is hardly ever the work of only one person. On the contrary, it's a joint effort. See the list of people in the Acknowledgments section, and also remember that even more people have been involved, especially during production. That said, any errors we didn't catch before the book went to print are mine and mine alone.

I will post information of interest to readers of the book at http://www.jnsk.se/adddp.

Getting back to bridging gaps, the photo of the Øresund Bridge was taken by my friend Magnus von Schenck on one of his sailing trips.

Even though this book has not been as tough to write as the first one, there has been a fair amount of blood, sweat, and tears. I hope the book might save you some of that. Have fun and good luck!

Jimmy Nilsson
http://www.jnsk.se/weblog/

Listerby, Sweden
September 2005

Read Applying Domain-Driven Design and Patterns: With Examples in C# and .NET By Jimmy Nilsson for online ebook

Applying Domain-Driven Design and Patterns: With Examples in C# and .NET By Jimmy Nilsson Free PDF d0wnl0ad, audio books, books to read, good books to read, cheap books, good books, online books, books online, book reviews epub, read books online, books to read online, online library, greatbooks to read, PDF best books to read, top books to read Applying Domain-Driven Design and Patterns: With Examples in C# and .NET By Jimmy Nilsson books to read online.

Online Applying Domain-Driven Design and Patterns: With Examples in C# and .NET By Jimmy Nilsson ebook PDF download

Applying Domain-Driven Design and Patterns: With Examples in C# and .NET By Jimmy Nilsson Doc

Applying Domain-Driven Design and Patterns: With Examples in C# and .NET By Jimmy Nilsson Mobipocket

Applying Domain-Driven Design and Patterns: With Examples in C# and .NET By Jimmy Nilsson EPub

F9EUXB7I6AW: Applying Domain-Driven Design and Patterns: With Examples in C# and .NET By Jimmy Nilsson