

[▶ subscribe](#)[▶ contact us](#)[▶ submit an article](#)[▶ rational.com](#)[▶ issue contents](#)[▶ archives](#)[▶ mission statement](#)[▶ editorial staff](#)

You Build Software -- Not Houses

by [Robert A. Maksimchuk](#)

Senior Evangelist
Rational Software

"Oh no, not again!" I thought as I read through the latest technical journal. "Not another article comparing software development to building a house. Has any analogy been overused as much as that one?"

Not that there is anything wrong with the software and houses analogy. (In fact, I must confess that I've used it myself.) It's just very limited -- and long in the tooth. The building analogy compares architecting buildings to architecting software. But is software development just about architecture? Hardly. The analogy is too simple.



So I started to look for a new analogy that would encompass more characteristics of software development. Software is complex, challenging, and sometimes frustrating. Serious software development involves many different people with specific roles -- each with different responsibilities, each important -- all working together. And there are different approaches to development; some are tried and true, and some are new and exciting. You really need to know your goals to be successful. Change is constant. Surprises abound. And development takes a long time to master. HmMMM.

Chess! Yes, chess: the game of kings! Chess has all these same characteristics. The more I thought about it, the more I saw that this ancient game shares other characteristics as well with software development.

The Right Tool for the Job

Chess has many playing pieces, just as software development has many participants. Each piece has different, highly specialized capabilities. Yet they all play together on the same board. Although other games have

boards and playing pieces, they may also have card decks, a bank, score sheets, dice, spinners, and so on. You might say these are all part of the game; however, they are all used differently and are only loosely (procedurally) integrated. In chess, everything you need is on the same board. The board is a complete "chess environment" in which a player leverages the unique capabilities of each piece -- whether it is a rook, a knight, a bishop, or a pawn -- to achieve a strategic goal: win the game.

Frictionless Design and Development

In a similar way, software developers can benefit from an integrated environment that allows them to leverage the unique capabilities of the tools they are using to achieve a strategic goal: produce great code. Rational® XDE™ Professional provides such an environment. It is an "eXtended Development Experience" that tightly integrates with your Integrated Development Environment (IDE) of choice (Microsoft® Visual Studio.NET, IBM® WebSphere® Studio Application Developer, or a standalone environment based on the Eclipse platform). It brings all your design and development tools onto one "playing board," so you don't need to switch context each time you switch tools. Design and coding become "frictionless" because all your tools make use of the same menus, gestures, and usage patterns. This results in reduced learning curves and accelerated development. You never have to leave your IDE.

Many Strategic Options

In chess, there is no single "correct" way to open the game. Whether you start by moving a pawn or moving out a knight, for example, is up to you. In development, your point of view (e.g., project developer, enterprise architect) often influences how you approach software development. Rational XDE allows you to design and code the way you want. You can be code-centric or design-centric (see Figure 1). You can automatically synchronize your code (C#, VB.NET, ASP.NET, Java™) and your design models as you work. Or you can wait until you're done and synchronize with a single button push.

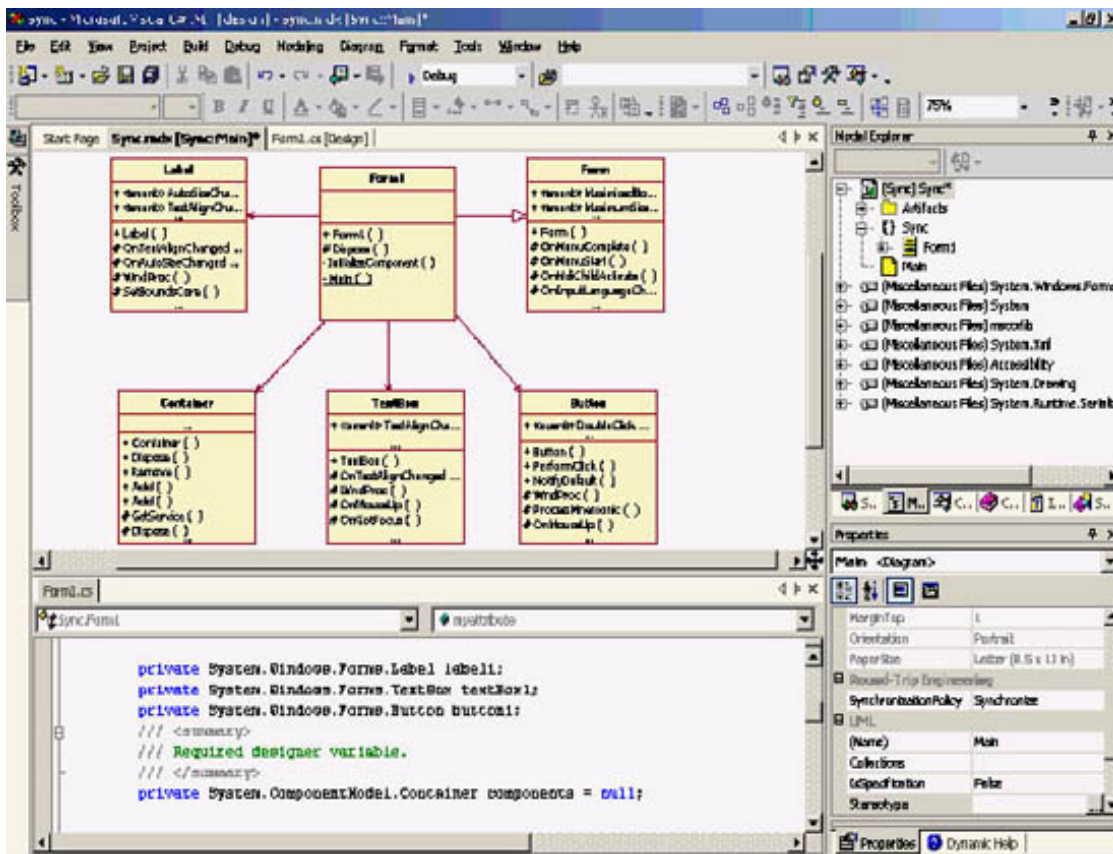


Figure 1: Rational XDE Supports Both Code-Centric and Design-Centric Development. It's Your Choice!

The "Blank Slate" Syndrome

In chess, every new game starts on the same board, with the same pieces configured in the same way. Also, there are numerous standard strategies for starting the game (openings) and ending the game (closings). And there are many mid-game strategies: Forced Move, Knight Fork, Pin, X-Ray Attack, Trap, Castling, and so forth. These standard strategies are well known in the chess community, as are particularly successful, unique, or powerful strategies named after places or the people who popularized them: Ruy Lopez, Sicilian, Budapest, Caro Kann, Alekhine, to name a few.

Until recently, developers have not had the benefit of standard strategies. Often they found themselves starting from scratch on projects. Although they knew they had already built great solutions to common problems, there was no way for them to reuse those solutions effectively and efficiently on subsequent projects.

Unleash the Power of Your Patterns

Now, however, developers can leverage the standard strategies embodied in software design patterns, including the well-known "Gang-of-Four"¹ patterns. Rational XDE provides you with these patterns, and others, to use in your designs (see Figure 2). Plus, it enables you to capture your own specialized patterns and reuse them across your development projects. (You can even claim your moment of fame by naming them after yourself!)

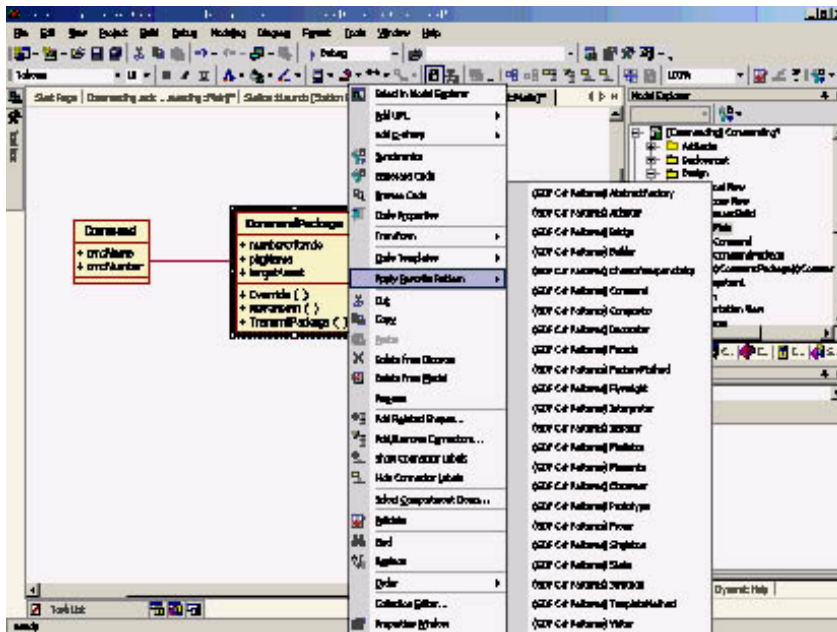


Figure 2: Using Proven Patterns Improves Productivity and Quality

This Stuff Is Hard

Chess is an extremely difficult game to master. It takes a bit of study to get started, and it can take a lifetime to become an expert player. The challenge is multi-faceted. You must learn how each piece moves and how to use it in combination with other pieces. And you must learn the various strategies, exceptions to the rules, and so forth. Chess even has "best practices": Don't "block in" your powerful pieces; learn the three ways to escape "check"; understand the relative value of different areas on the chessboard, and so forth. As with everything else, you need experience to become proficient. But you have to begin somewhere. Most people start with books, many of which do a good job of explaining the game and its moves, using a standard notation system that chess players understand.

Software development is similar: It's complex and hard to master; acquiring expertise takes time; following best practices can help you win. Many software developers also use a standard notation: the Unified Modeling Language (UML). Just as you learn standard chess moves from books, you can learn the UML from books and training classes if you have the time (How does that development schedule look?), but many developers don't have this luxury. So why not learn it while you code?

Instant UML

Rational XDE gives you a faster and easier way to learn UML. When you code, Rational XDE instantly generates UML models that you can use to document your design and communicate project information to other stakeholders. In addition, Rational XDE gives you access to Rational Developer Network, an online resource offering Web-based training, technology guidance, and articles, as well as countless other educational resources (see Figure 3). So you can not only learn while you code, but also benefit from the experience of industry leaders via Rational Developer

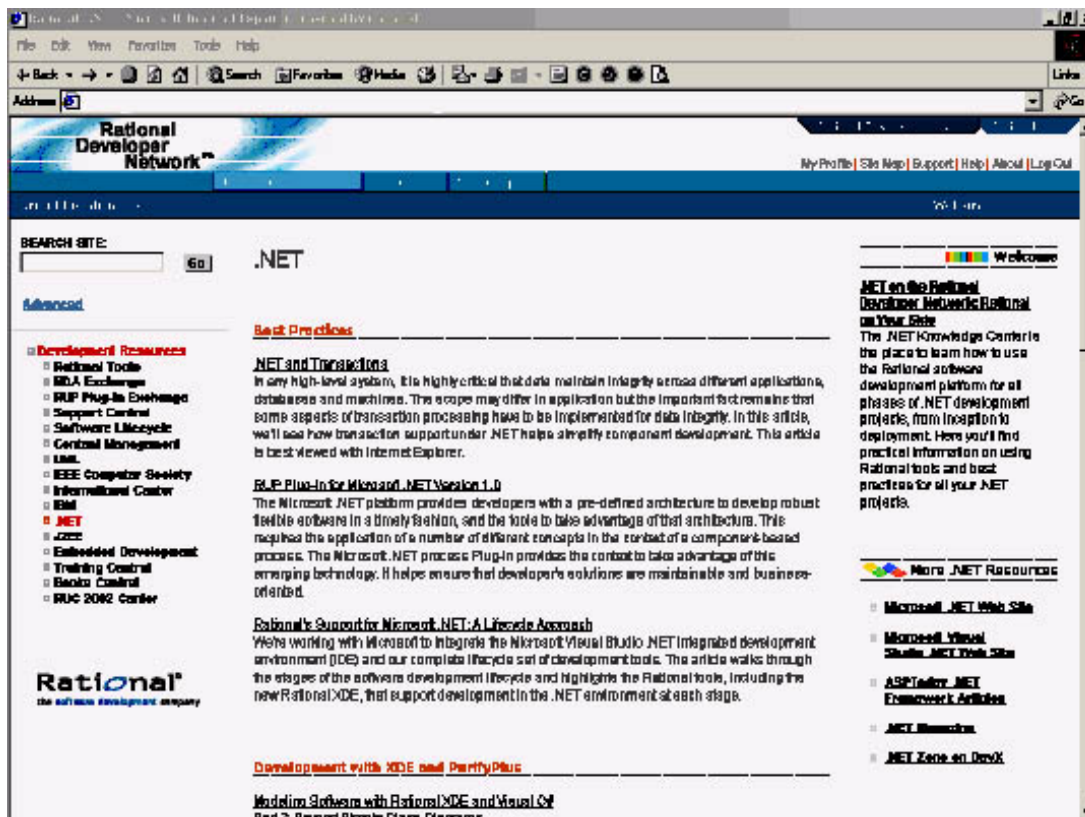


Figure 3: Rational Developer Network Is a Key Resource for Development

The Need to Communicate: An Important Difference

Of course, chess and software development are not alike in every respect. Except in formal competitions, typically only one key set of people gets involved in playing a particular chess game: the players. They manage how and when to move which pieces. Nobody watches them to spot errors in their play and ensure it is of high quality. Nobody complains that their game is affecting the game others are playing across the hall. The two opponents are the only people who need to understand what is happening in their game.

Although the same might have been true long ago for software development, today software is a team sport. Often, developers must explain their designs to people who have never written a single line of code. All the stakeholders -- customers, managers, team leaders, developers, testers, marketing, sales, and so forth -- have a valid need to understand your work. How can you explain it to people who are not code warriors?

Get More from Your Models

Rational XDE allows you to create design models that are both powerful *and* flexible. By combining the semantic power of the UML with capabilities for free-form diagramming, using expressive shapes and icons, it lets you

communicate your ideas to anyone, at the appropriate level of complexity and abstraction (Figure 4).

For your more technical teammates, Rational XDE provides capabilities in numerous technology specialties. And with Rational's proven, UML-based data modeling technology, even non-technical people will be able to understand your conceptual, logical, and physical database designs. In addition, Web team members can design and generate code for Web applications and Web Services, using the new Web stereotypes in XDE. They can also reverse engineer existing Web applications and Web Services to help them understand interactions and complexities in the Web designs.

And finally, with Rational XDE's multi-model capability, you can organize your projects and designs for ease of use. Your design solution can have multiple models open at the same time -- requirements models, architecture models, design models -- whatever you need to make your designs clear, scalable, and manageable.

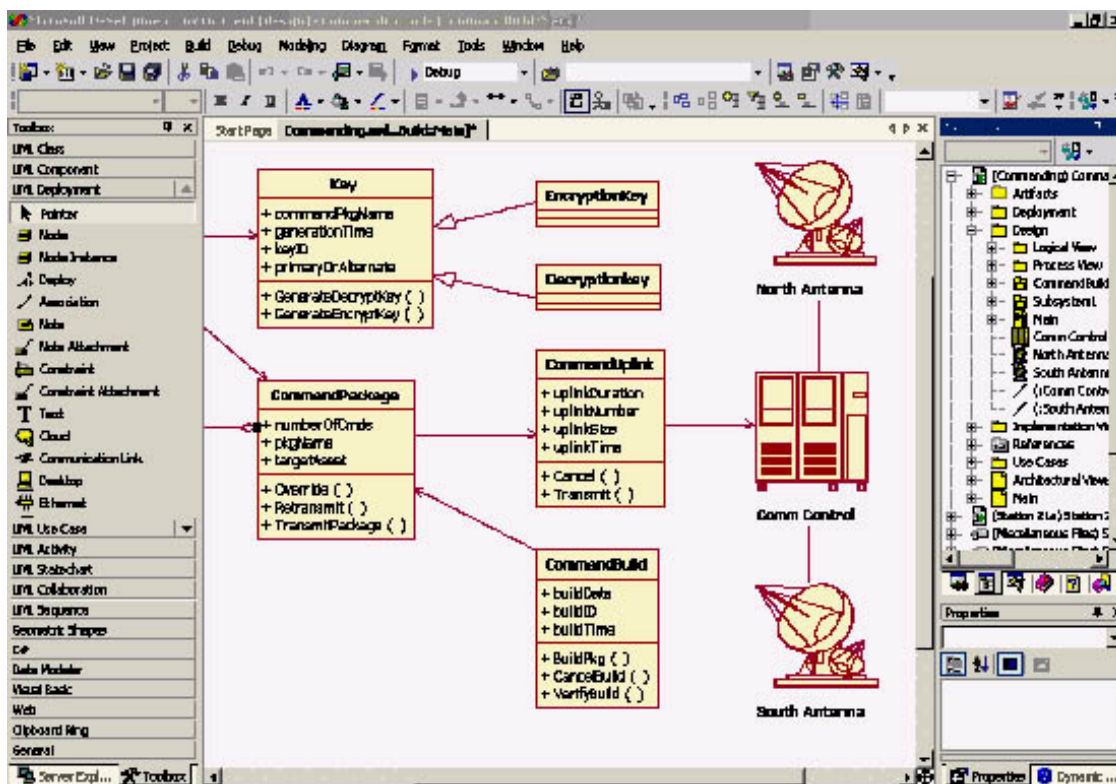


Figure 4: Free-Form Modeling Enables Clear Communication with Multiple Stakeholders

The Endgame

Chess and software development have at least one more important similarity: Neither activity will ever be "easy." If you play the game, you will win some and lose some. For software practitioners, the stakes can be high: A failed project might bring heavy losses for the business. But just as in chess, if we keep striving to improve -- using our skills, proven strategies and best practices, and powerful, innovative tools like Rational XDE -- then we can win. And like the often-underestimated pawn, we may

even reach the other side of the board where we can be "promoted."

Acknowledgments

Thanks to Lisa Dornell, Carolyn Hakansson-Johnston, and David Hauck for their editorial assistance.

Notes

¹ Gamma, Helm, Johnson, and Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison Wesley, 1995.



For more information on the products or services discussed in this article, please click [here](#) and follow the instructions provided. Thank you!