Developments in Business Simulation and Experiential Learning, Volume 34, 2007

APPLYING .NET REMOTING TO A BUSINESS SIMULATION

Precha Thavikulwat
Towson University
pthavikulwat@towson.edu

Jimmy Chang
Hong Kong Polytechnic University
tcchangj@inet.polyu.edu.hk

ABSTRACT

We consider Microsoft's .NET Framework in relation to its alternative, Sun Microsystems' Java Virtual Machine. We discuss principal issues for business simulation developers who would use .NET remoting technology. We explain how we applied the technology to an existing computer-assisted business simulation such that the application is available in both a local-area-network pure-Windows version and an Internet-based .NET version. Problems are discussed, and suggestions given for first-time and experienced developers of computerized business simulations.

INTRODUCTION

From time to time, developers of computerized business simulations are confronted with new technologies to which they must respond. Thus, the mainframe computer was superseded by the microcomputer and Microsoft's Disk Operating System (DOS) was superseded by Microsoft Windows. Web browsers followed, and with it, Sun Microsystems' Java Virtual Machine (JVM) and Microsoft's .NET Framework (DNF).

Both JVM and DNF are software components that can be added to a computer's operating system. JVM, first released in late 1995 (Byous, 1998), is available for installation with Linux, Microsoft Windows, and other operating systems. DNF, first released a few years later, is available for fully featured installation with Microsoft Windows only (Richter, 2000), although the company apparently plans to make it available in the future for other operating systems. Both JVM and DNF are attempts to make advanced programming easier for developers, especially in applications that involve the Internet.

Programs written for JVM and DNF differ from their predecessors in that the predecessor programs were compiled directly into machine instructions that would be read by the computer's processor, whereas the successor programs are compiled into intermediate instructions that would be preprocessed by the JVM or the DNF, as the case may be, before being sent to the processor. The preprocessing step allows the preprocessor to assure security by screening out disallowed instructions and to insert installation-specific instructions to perform featured tasks, such as optimizing memory usage and accessing data through the Internet. Assuring security and optimizing memory usage are nice features, but the feature of JVM and DNF that developers of business simulations should find the most attractive is their simplification of data access through the Internet.

Pillutla (2003) has classified applications that access data through the Internet into two categories: Web-based and Internet-based. Web-based applications are embedded within a browser. Internet-based applications access data through the Internet without a browser. Browsers themselves are Internet-based applications. JVM and DNF simplify the development of both Web-based and Internet-based applications. As such, they make programming such applications easier, allow applications to link more effectively with databases and research tools, and democratize usage by enabling mass participation from distant sites (Dasgupta & Garson, 1999).

Both JVM and DNF are designed for object-oriented programming, wherein data and their associated operations are packaged together into base classes that can be incorporated into more refined classes. The concept is relatively new, but one already in use by some developers of computerized business simulations (Duserick, Enke, Huang, & Robana, 1999; Helge, Michael, & Joerg, 2004). In these instances, the developers worked with JVM. To date, the business simulation literature apparently contains no report of developers working with DNF.

JVM and DNF are incompatible. To benefit from JVM, developers must write code in the Java programming language. To benefit from DNF, developers may write code in any of the languages supported by Microsoft through its respected suite of programming tools, Visual Studio. These languages include Visual Basic, Visual C++, Visual C# (pronounced "C sharp"), and Visual J# (pronounced "J sharp"), but not Java. Thus, an application based on computer code written in a language supported by Microsoft before the advent of DNF can be modified to take advantage of DNF features, but the same application will have to be completely recoded in Java to take advantage of JVM features. Complete recoding is more costly for larger programs than it is for smaller ones, so developers of large programs written in a language supported by Microsoft may be inclined to forgo JVM in favor of DNF for that reason alone.

For the developer of a new application, JVM offers the allure of an open-source package supported by a strong open-source community, whereas DNF offers an integrated package supported by Microsoft, a company with a distinguished reputation for well-integrated products. Each side has many loyal adherents. Essentially, however, developers of JVM applications have many choices of vendors for development tools and services, whereas developers of DNF applications have Microsoft's Visual Studio, which, as Thilmany (2004) notes, is found almost everywhere and does almost everything.

Herein we show how DNF has been applied to an Internet-accessing business simulation such that the simulation runs as a Windows program independent of a Web browser, without requiring installation on the client computer. Accordingly, the simulation is an Internet-based application. Moreover, the simulation accesses its data through the Internet using an efficient data transmission arrangement known as *remoting*. Principal issues of the implementation will be considered first. Then the characteristics of the business simulation will be described, followed by an explanation of how the simulation has been structured to use DNF for its Internet-accessing functions. We describe the DNF-related problems that we encountered, and we conclude with observations and suggestions for first-time and experienced developers.

PRINCIPAL ISSUES

The principle issues of an application are those that are decisive in determining how well the application functions. For an application that will access its data over the Internet, we identify four principal issues: deployment method, data accessing method, server object lifetime management, and program responsiveness.

DEPLOYMENT METHOD

A program that access data over the Internet can be deployed as a Web-based application dependent on a browser, or as an Internet-based application independent of a browser. The Web-based application generally consists of *applets*, packages of computer code on the client side that interface with the user, and *servlets*, packages of computer code on the server side that access data stored on the file server. The Internet-based application consists of client-side and server-side executable files, which in the Microsoft Windows environment are files with EXE and DLL extensions. Internet-based applications generally are less cluttered and more responsive to user actions, because the program is better able to control what the user sees and what the user may do. Conversely, the browser of a Web-based application surrounds the application with a frame that reduces the application's working area on the computer's screen, and gives the user Back and Forward buttons that overrides the program's control of the user's view.

A common misconception is that the Internet-based application must be installed on a computer before it can be run, whereas the Web-based application can be run simply by navigating to its Web site. For those using Microsoft products, this is incorrect. Microsoft's Internet Explorer (IE) will run an application directly when the application's address is entered into IE's address box, in the same way that IE will bring up a Web site when that Web's site address is entered therein. Thus, entering http://pages.towson.edu/precha/geonetd5.exe suffices to launch the GEONETD5 application, provided the file, GEONETD5.EXE, exists on the Web site http://pages.towson.edu/precha. To guard against a malicious application, IE and the operating system will prompt the user several times before launching the application. If the user responds affirmatively to the prompts, the application will be launched as if it had previously been installed. Installation before launch is unnecessary. As for other browsers, such as Mozilla Firefox, the user may be required to save the program file, which in the above case would be GEONETD5.EXE, before launching the application from the program file in a two-step process.

Accordingly, the Internet-based application can be used in all instances where a Web-based application is useful, but the converse is not true, especially not without clever programming. As a rule, however, clever programming should be avoided whenever possible, because it tend to give rise to code that is fragile, hard to understand, and hard to maintain.

DATA ACCESSING METHOD

An Internet-based application can access the Internet by two methods, Web servicing and remoting (Gregory, 2004). In Web-servicing, data transmitted between the client program running on the user side and the server program running on the file server side are encapsulated in packages prefaced by metadata, that is, information on the type of data within the package. In remoting, data is transmitted raw. Raw data is more compact. To assure that the remoted data are meaningful to both programs, DNF requires both client and server to share a common reference. In the simplest arrangement, the common reference is a full copy of the server program that is packaged with the client program, but this arrangement makes the client unnecessarily large.

A less known but more efficient and safer arrangement is for both programs to share an interface, a small binary file that contains only information sufficient for both programs to understand the data they transmit (Rammer & Szpuszta, 2005). This arrangement is safer because it limits access to the server program. Access would enable hackers to disassemble the program in search of vulnerabilities.

SERVER OBJECT LIFETIME MANAGEMENT

Managing server object lifetimes is among the most difficult of issues in an Internet-accessing application. When a program is launched, program instructions are placed into the computer's memory as memory objects. For programs that access local data, these memory objects are removed by the operating system when the program terminates normally. If the program terminates abruptly, often because of an error in the instructions, its memory objects continue to occupy memory space until the computer is rebooted. Rebooting a single-user computer is generally a minor concern, because most users turn off their computers every day in any case. Rebooting a file server computer, however, is a serious undertaking, because all clients that connect to the server will be disrupted by the process. For this reason, DNF objects that reside on the server have a five-minute default lifetime that is renewed for two minutes whenever the object is called into use by the client program.

If the client program should call the object after it has been destroyed, because its lifetime has ended, the server will respond with a no-object-available error message in place of the data that the client program may have expected to receive. If the programmer has not encoded a response to the error message, the client program will terminate abruptly, to the chagrin of the user. An effective response by the client program to the no-object-available error message is to call a "factory" object on the server, with instructions to re-create the server objects required by the client program and set the state of those objects to appropriate values (Rammer & Szpuszta, 2005).

PROGRAM RESPONSIVENESS

Program responsiveness refers to user's perception of the time between the issuance of a command by the user and the completion of that command by the computing system. This perception is affected by processing speed and program features that make the wait seem shorter.

Processing speed mostly depends upon how frequently the program accesses its data. Before disk caching became ubiquitous, programmers sought ways to minimize reads and writes to disks, because these operations took so long that they caused noticeable delays in program responsiveness. Disk caching eliminated the problem in most cases. It managed reads and writes, converting most disk-accessing operations to memory-accessing operations so that an operation that read from disk many times to compute a result was not noticeably less responsive that an operation that read from disk with less frequency.

When a program accesses its data through the Internet, the physical distance between the client program and its data files coupled with the burden of packing and unpacking the data for transmission through the Internet slows down reads and writes so that the delays of these operations have become noticeable again. The program must work around these delays by performing data-accessing operations in the background, encoding data-intensive routines into the server program, and stripping unessential items from the data that is transmitted. These operations are difficult to program because they are difficult to debug. A development system that includes a powerful debugger will make the task much easier.

THE BUSINESS SIMULATION

The Internet-based business simulation that we developed with DNF is GEO. It is a *computer-assisted* simulation (Crookall, Martin, Saunders, & Coote, 1986). Like the more common *computer-controlled* simulation, a computer-assisted simulation supports a high level of participant-to-participant interaction. The difference between the two is in the control of events. Control rests with the computer in the computer-controlled simulation; it rests with the participants in the computer-assisted simulation.

The purpose of giving participants control over events is to give them direct experience with critical processes, such those of trade, investment, employment, and superior-subordinate relations. In the simulation, these processes were represented genotypically (Thavikulwat, 2004). Thus, when participants sold products, they sold them to other participants, rather than to an algorithm. When they invested, they bought shares in each other's companies. When they employed, they employed each other. When they related to each other as executives of a company, their roles were computer-enforced so that superiors controlled the subordinate's compensation, could review the subordinate's work, and could dismiss the subordinate at will. Accordingly, the markets for products, stocks, and employment are real (Thavikulwat, 1997). The approach is constructivistic, emphasizing social interaction, rather than instructivistic, which emphasizes description and examples (Leemkuil, de Jong, de Hoog, & Christoph, 2003).

A computer-controlled simulation generally requires decisions to be submitted by predetermined deadlines, but a computer-assisted simulation generally processes each decision as it is submitted. As a consequence, easy access by participants to the computer is essential in a computer-assisted simulation. Ideally, access should be available at any time, from anywhere, and simultaneously to all participants. Given today's technology, this means that the simulation software should be able to access its data over the Internet.

GEO has been in use for over a decade. It was originally coded in the C programming language for the DOS platform, and later upgraded to Windows C++ for the Windows platform. When DNF became available, the simulation was modified so that it could be compiled into two executable versions, a pure-Windows version without DNF that accessed its data through a local area network and an Internet-based version with DNF. Compiling the same code into the two different executable versions was made possible by a feature of DNF that Microsoft calls It Just Works (IJF). IJF is available for programs coded in

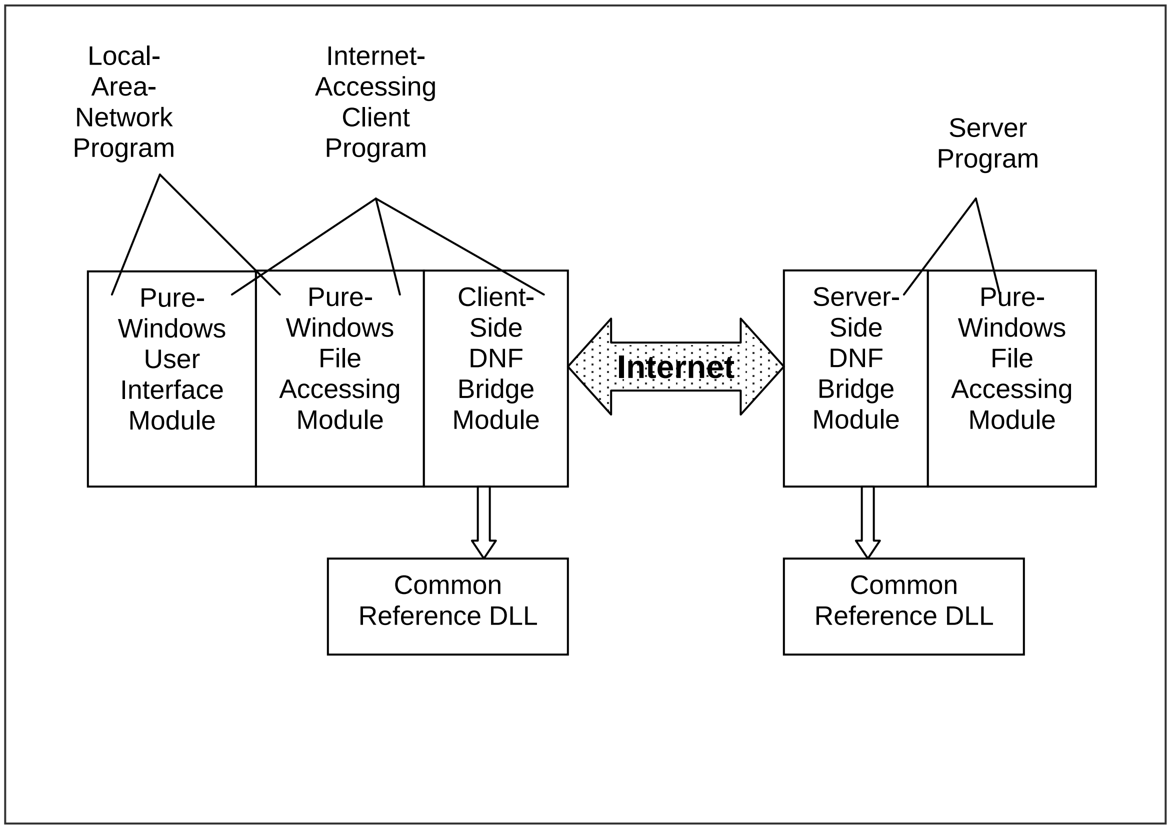
Local-Area-Network Program
Internet-Accessing Client Program
Server Program
Pure-Windows User Interface Module
Pure-Windows File Accessing Module
Client-Side DNF Bridge Module
Internet
Server-Side DNF Bridge Module
Pure-Windows File Accessing Module
Common Reference DLL
Common Reference DLL

Figure 1: Two-Version Internet Remoting System

Windows C++ only. Programs coded in Windows Basic, generally accepted as the most popular of the Microsoft-supported languages, cannot use this feature.

A schematic diagram of the two-version system is shown in Figure 1. As the figure shows, the system consisted of three programs: a local-area-network program, an Internet-accessing client program, and a server program. The local-area-network program links together a user-interface module and a file-accessing module to form an executable file with an EXE extension. The Internet-accessing client program adds the client-side DNF bridge module to the two modules of the local-area-network program, forming another executable file with an EXE extension also. Unlike the local-area-network program, however, the Internet-accessing client must be accompanied by the common-reference DLL, an essential component of remoting. The server program consists of the server-side DNF bridge module and the same file-accessing module of the other two programs. The linked combination is a program file with a DLL extension. This server program also must be accompanied by the common-reference DLL.

The server program was designed to work with Microsoft's Internet Information Services (IIS). This arrangement allows the program to be updated by simply replacing it with a new copy, without needing the administrative privileges necessary to restart a Windows service. Administrative privilege is necessary only initially, to set up the server application within its own folder on the server. Subsequently, those given full rights to the folder are able to modify the folder's content at will. IIS automatically detects the modifications and makes them effective immediately.

The two-version system greatly simplified debugging, generally considered the most time-consuming stage of computer program development. The most complex codes were associated with the pure-Windows file accessing module. These codes were tested first in the pure-Windows local-area-network program, where operations could be tracked with finer granularity than is possible with the DNF-based server program. Codes that involved the DNF were isolated into a few files, thereby facilitating the tracing of errors resulting from those codes.

The two-version system also enabled the gradual deployment of the Internet-accessing DNF version. Participants were given access to both versions. They could use either version when they had access to a computer connected to the local area network. They were able to use the Internet-accessing version otherwise. Thus, the Internet-accessing version made access more convenient. Its responsiveness and reliability were not critical to the exercise.

The Internet-accessing version was put to a critical test when it was used as the sole version in a two-month exercise that involved 42 participants from the opposite side of the world, where the distance between the client program and the server program was the greatest. The major DNF-associated problems encountered in this test were as follows:

- Failure to launch
- Outdated client
- Locked data
- Server busy message

FAILURE TO LAUNCH

A directly executable DNF file appears no different from a directly executable pure-Windows file. Both have an EXE extension. Invoking a DNF executable when DNF has not been installed on the client gives rise no response. Although participants were advised that DNF must be installed on the computer before the executable will run, some either missed the instructions or were unable to do the needed installation.

This problem could not be resolved by having the program pop-up a dialog box that would inform the user that DNF was needed, because no part of the program would run without DNF. It can be resolved by having the user launch a pure-Windows pre-application program that would check for the presence of DNF before launching the DNF program itself. We chose not to implement this solution, however, as we deemed it to be not worth the added complication to the package. The problem will resolve itself over time as operating systems are upgraded, because Microsoft has already made DNF a prepackaged part of its Windows operating system.

OUTDATED CLIENT

When participants run a local-area-network program through a link to the program, the program can be updated with assurance that all participants will subsequently be running the updated program. The same assurance does not apply to a universal resource locater (URL) address that points, over the Internet, to the DNF program. In this latter case, when the URL is entered on the browser's address box, the browser will either prompt the user to choose between running the program and saving it, or the browser will give the user the single option of only saving the program. If the user saves the program before running it, the user may subsequently choose to run the saved program, which may then be an outdated version if the original version has been updated.

The possibility that the client may be outdated made the task of updating the application while the exercise was in progress difficult. Changes to the server program had to allow for the possibility that it would be serving outdated client programs as well as current ones. In some instances, the required updates were such that they could not be made compatible with outdated client programs. Nevertheless, these instances did not appear to greatly inconvenience participants, as only a few complaints traceable to an outdated client were received, the participants having been warned early on that they should not run a client program that was downloaded at an earlier time.

LOCKED DATA

The easiest way to adapt a program that accesses its files through a local area network to one that accesses its files through the Internet is to redirect all low-level file-accessing operations, such as those that open, lock, read, write, unlock, and close files, to operations at a distance through the Internet. When all low-level operations have been re-directed, the program is functional over the Internet without the need to redirect high-level operations, such as those for computing the book values of a collection of companies. The problem with this method is that high-level operations can take so much time that the user, in frustration, aborts the process, either through Window's Task Manager or by rebooting or turning off the computer. If the process is aborted after data in a file has been locked but before the data is unlocked, the file will remain locked until either the server is rebooted or until an indeterminate time after the server object's lifetime has ended. Locked data can neither be read nor updated by any process other than the one that set the lock, effectively stopping all activities that must access the data.

Anticipating this problem, we coded the locking and unlocking operations so that they were performed on proxy values that represented no data, rather on real data values. This permitted a data recovery program to read all of the data in the locked files, from which it could reconstruct the files. Accordingly, our immediate resolution of the problem was to use the data recovery program to recreate files and folders of the simulation at another location on the server, after which we redirected the server program to the new location. The solution is workable, because the circumstances that lead to locked data are infrequent. Over the long term, as an increasing number of high-level file-accessing operations are moved from the client program to the server program, the problem of locked data will fade. Then, whenever the server program performs an operation that requires locking data, the same program will unlock the data before returning values to the client program, in which case the lock-out state will never be for an extended time.

SERVER BUSY MESSAGE

The Windows operating system pops-up a message box with a server-busy message when the wait for a response from the server is long. Long server response times were infrequent when participants lived within automobile driving distance of the file server, so the occasional server-busy message was tolerable. In the situation when participants were at the opposite side of the world from the file server, however, the server-busy message appeared frequently. It became a substantial irritant.

The usual Windows documentation does not cover this message box. An Internet search of the issue led to a posting with the advise that the dialog box could be turned off by setting a little-known Enable-Busy-Dialog operation to *false*.

This is an example of the many poorly documented issues that one inevitably encounters in applying a new technology. Fortunately, when the issue involves a technology associated with Microsoft, the likelihood that others have posted a resolution of the issue somewhere on the Internet is high. An Internet search based on the key words that describe the issue usually leads to advice on how the problem should be resolved, as it did in this case.

CONCLUSION

DNF is among the latest of the technologies that have become available to developers of computerized business simulations. We have discussed the principal issues that developers who choose to incorporate the technology into their applications must resolve, and we have given an account of the problems that we encountered when we used an Internet-based simulation with DNF in an exercise that involved participants from one side of the world accessing the simulation data on a server at the other side. We have found that the problems are manageable.

DNF is an attractive technology for developers of business simulations who know a Microsoft-supported computer language well. For many, the language of choice is Visual Basic. Although the application presented here was developed in Visual C++, the issues and problems we have discussed apply just as well to DNF-based programs written in Visual Basic, except that all code written in Visual Basic for a pure-Windows program must be recoded for DNF if the program will use DNF technology. Microsoft's IJW feature, which permits linking together pure-Windows modules with DNF modules into a single DNF-based program, is not supported for code written in Visual Basic.

DNF substantially eases the problem of having a program access its data over the Internet. Making the program functional is done with only a few lines of code, so it can be said to be easy. Making the program function well, however, remains difficult, because server objects have lifetimes that must be managed, because operations must be carefully divided between those performed by the client and those performed by the server to maximize program responsiveness and minimize server resource requirements, and because literature on DNF is fragmented. Nonetheless, the difficulties are surmountable, as has been our experience.

To developers, we offer the following suggestions. First-time developers of business simulations should try C#. The language is simple, like Java, but powerful, like C++. Although Visual C++ is our preferred language, learning to use it requires an extensive effort that is difficult to justify given the C# alternative. Developers who have worked with JVM, as we also have, should try DNF. The promise of JVM is that it will run everywhere; the promise of DNF is that it will do everything. Neither completely lives up to its promise, but for the usual business gaming simulation application, running everywhere is superfluous. Running on Windows is enough. Doing everything, however, is a godsend.

REFERENCES

Byous, J. (1998). Java technology: The early years. Sun Microsystems, Inc. Available: http://java.sun.com/features/1998/05/birthday.html.

Crookall, D., Martin, A., Saunders, D., & Coote, A. (1986). Human and computer involvement in simulation. *Simulation & Gaming, 17*, 345-375.

Dasgupta, S., & Garson, G. D. (1999). Guest editorial: Internet simulation/gaming. *Simulation & Gaming, 30*, 20-22.

Duserick, F., Enke, E., Huang, W., & Robana, A. (1999). Financial simulation using distributed computing technology. *Developments in Business Simulation and Experiential Learning, 26*, 52-57. Available http://www.absel.org.

GEO. Thavikulwat, P. (2006). Demo available: http://pages.towson.edu/precha/geonetd5.exe. (Department of Management, Towson University, 8000 York Road, Towson, MD, USA).

Gregory, K. (2004). *Visual C++ .NET 2003*. Indianapolis, IN: Sams Publishing.

Helge, F., Michael, J., & Joerg, B. (2004). Controlling the complexity and orienting target groups by a modular, server-based business game system. *Developments in Business Simulation and Experiential Learning, 31*, 1-7. Available http://www.absel.org.

Leemkuil, H., de Jong, T., de Hoog, R., & Christoph, N. (2003). KM QUEST: A collaborative Internet-based simulation game. *Simulation & Gaming, 34*, 89-111.

Pillutla, S. (2003). Creating a Web-based simulation gaming exercise using PERL and JavaScript. *Simulation & Gaming, 34*, 112-130.

Rammer, I., & Szpuszta, M. (2005). *Advanced .NET remoting: Everything you need to know about .NET Remoting.* Berkeley, CA: Apress.

Richter, J. (2000, September). Microsoft .NET Framework delivers the platform for an integrated, service-oriented Web. *MSDN Magazine*. Available: http://msdn.microsoft.com/msdnmag/issues/0900/framework/print.asp.

Thavikulwat, P. (1997). Real markets in computerized top-management gaming simulations designed for assessment. *Simulation & Gaming, 28*, 276-285.

Thavikulwat, P. (2004). The architecture of computerized business gaming simulations. *Simulation & Gaming, 35*, 242-269.

Thilmany, C. (2004). *.NET Patterns: Architecture, design, and process.* Boston: Addison-Wesley.