ISSUES IN PORTING A LAN-BASED TOTAL ENTERPRISE SIMULATION GAME TO A WEB-BASED ENVIRONMENT

Sharma Pillutla
Towson University
spillutla@towson.edu

ABSTRACT

Business Simulation games have been used in capstone courses and other business courses for more than four decades. During this time games have morphed from paper and pencil based approaches to a web-based approach. The most recent incarnation of simulation games was a client-server based approach where a client program is installed on the user's computer (players program on the student's computer) and a server program (game administrator's program on the faculty/instructor's computer). The data exchange between these programs was done using either disks/flash drives or using a Local Area Network (LAN). Since the advent of the World Wide Web many games are moving from a LAN-based environment to a web-based environment. In this paper, we examine the issues involved in making such a move. The paper first lays out the different types of simulation games and narrows down the scope to the type of simulation game considered in this paper. We then explain the functioning of the game on a LAN-based environment. It then explicates how the game would be configured on the Web. We then list issues that need to be addressed in such a structure. We use the Microsoft .NET framework in explaining how these issues can be addressed. We conclude the paper with summary and future directions.

INTRODUCTION

Simulation games in the business field have been around for more than four decades. In this time, the games have undergone various changes and the field of simulation gaming itself has become more sophisticated. Thavikulwat (2004) developed a framework that delineated in broad terms the architecture of a simulation gaming solution. Since this is such a complex area, classifying business games can be done on various dimensions including purpose, control & interaction, representational systems, timing, hosting and performance scoring. While all of the above dimensions provide a comprehensive taxonomy of gaming solutions, this paper does not deal with such a broad array of games. We briefly discuss each of the above dimensions and explain the specific focus of this paper by narrowing down the category of simulation games considered herein.

A wide variety of simulation games have been developed including total enterprise (TE) simulations, functional simulations, concepts simulations, planning simulations, analysis simulations, computer enhanced role-plays (Hall, 2005). Each category addresses a specific structural aspect of the simulation. The focus, in this paper, is more on the TE simulation though a lot of the aspects could conceivable carry over to the other types of simulators. Crookall et al (1986) look at the two axes of (a) control – whether the game is controlled by the computer or by the participant in the game – and (b) interaction – whether there is more computer-participant interaction or more participant-participant interaction. Based on these axes values he classifies games into four classes – computer-directed (akin to a teacher-less blackboard), computer-based (aircraft simulation), computer-controlled (standard TE simulations) and computer-assisted (real market simulations). In the paper, we focus primarily on the computer controlled variety of games, though the analysis could be extended to the other types with facility and minimum loss of applicability. In any event, as Thavikulwat (2004) states, most of the TE games today tend to not use a real market environment. So restricting the analysis to computer-controlled and computer –based games would be addressing the bulk of the gaming solutions in use today.

Thavikulwat (2004, 1999) also discuss two types of representational systems – genotypical and phenotypical representations. A phenotypical representation is a reflection of the process and a genotypical representation tends to be a subset of the former. "Thus, a phenotypical representation of employment would have participants employ fictitious persons; a genotypical representation would have them employ each other. (Thavikulwat, 1999, p. 362). Most of the concepts discussed in this paper would be equally applicable to either form of representation.

This issue of timing is a little more nuanced. Thavikulwat (2004) uses scaling (fixed versus flexible), drive (administrator-driven, activity-driven, clock-driven) and synchronization (synchronized, unsynchronized) as the three parameters that distinguish between various gaming solutions. These three parameters also tend to have significant interaction with one another. A fixed scaling simulation proceeds in broad time segments such as months, quarters, etc. whereas a flexible scaling could vary from broad time segments to smaller time segments to the ultimate extreme of continuous time. Any real-time simulation (such as aircraft flight simulation) would tend to fall more towards the continuous time end of the continuum. Most extant TE and functional simulations tend to follow fixed scaling in that the gaming design only permits one type of scaling as programmed into the application. In terms of drive, clock and activity-driven simulations, in general,

tend to use finer segmentation of the time scaling. Thus a clock-driven activity could be segmented as low as a second or a minute at which time the game progresses to the next time unit. An activity driven game would most probably have multiple decisions that might be made by the participants (and each decision could constitute an activity which progresses the time unit to the next one). A synchronized game would require all participants to have all decisions input by a certain deadline at which time the game administrator would "run" the game as opposed to an unsynchronized game that does not have synchronized time segments driving the decision-making. Design issues in a web-based game might be significantly affected by the timing parameters. In this paper we address only one sub-segment of games, i.e., those that have a fixed scaling, are administrator-driven and are synchronized. Again, to the best of the knowledge of the author, most TE and functional simulations in use today tend to follow the aforementioned sub-segment and hence focusing on this segment does not diminish the importance of this subsequent discussion.

The next classification set forth by Thavikulwat (2004) is based on the dimension of hosting and he lays out three categories – stand-alone, LAN and Internet. In addition to the above hosting categories, Pillutla (2003) expands the third category further into Web-based and Internet-based applications. Applications in the former category are embedded in a web browser, while the latter use the Internet (just like a LAN) primarily as a network medium to exchange data. The focus of this paper is squarely in web-based arena and we examine issues raised by the move from a LAN based environment to a web-based environment. Thavikulwat (2004) states that "a program written for a stand-alone system can readily be extended to take advantage of a LAN's capabilities because a single programming language can be used for both hosting systems". However, a move from a LAN based environment to a Web-based environment is non-trivial.

Thavikulwat and Chang (2007) discuss issues in moving an existing LAN-based game to an internet-based application. They use "remoting" to retain the client-server flavor of the game while using the Internet (instead of a LAN) to accomplish the communication and data exchange between the client and the server. While there have been games that have been developed from the scratch for a web-based environment (Forio Simulations, 2007), many of the games that were originally developed to be played in a stand-alone mode or a LAN environment may have to be moved to a web-based environment. The popularity of the WWW and its universal availability and accessibility as well as the user-friendliness of the GUI as typified by standard browsers all provide evidence that such a move is inevitable.

We discuss next the architecture of a typical LAN-based client-server configuration of a business simulation game (keeping in mind the specific sub-segment of simulation games that are the subject of this paper as set forth earlier). Following that we lay out the issues that need to be addressed when porting such a game to the web-based environment. We finally conclude the paper by summarizing the issues addressed and pointing to aspects not considered in this paper that need to be addressed.

A LAN-BASED ENVIRONMENT

Most Total Enterprise and functional simulations that were developed in the previous decades followed a stand-alone or a LAN-based architecture. The business environment modeled in such games tends to include game players or participants that manage a functional area or the total enterprise. As mentioned in the previous section, we focus on those games that follow a synchronized, administrator-driven, fixed scaling architecture. These games constitute the bulk of the existing games in this category. The primary process in such games involves a student entering decisions for a particular period by a specified deadline as stated by the instructor or game administrator. At the conclusion of this deadline, the game

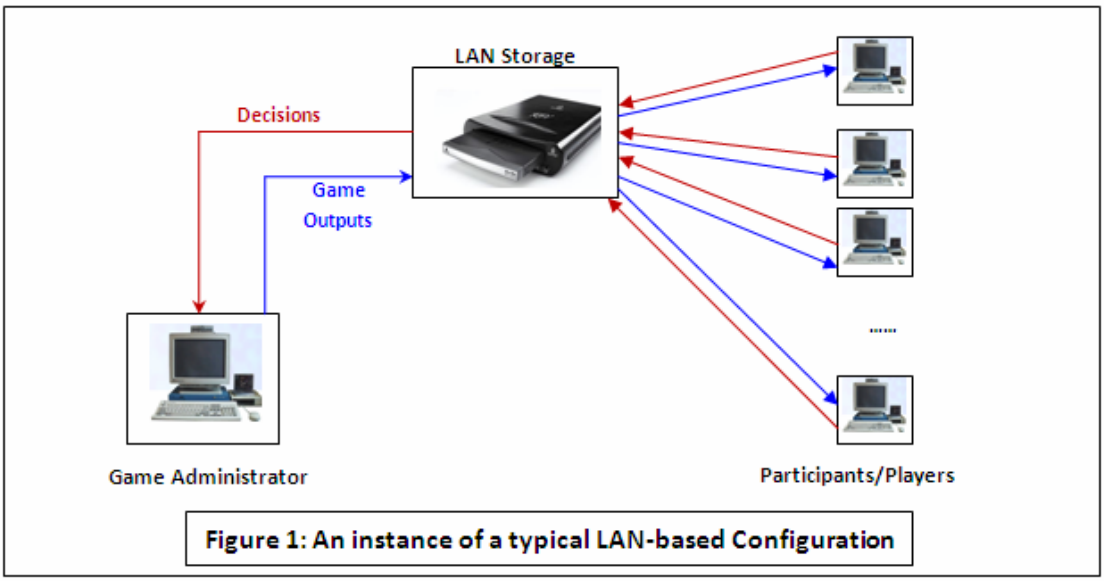

Figure 1: An instance of a typical LAN-based Configuration

administrator consolidates decisions made by participants according to the structure laid out in the game (by industry, or by world) and "run" the game resulting in the generation of outputs. These outputs could include a variety of financial and other performance measures. Standard outputs include income statements, balance sheets, cash flow statements and other team and individual performance measures arrived at by using various measures such as the balance scorecard, pro-scores, z-scores etc. The primary communication task that utilizes the LAN involves sending the player decisions to the game administrator and sending the game outputs to the player participants. **Figure 1** lays out one instance of a simulation game that follows a typical LAN-based architecture.

Historically, the complete application involved (a) a game administrator component and (b) a player/participant component. These two components were two stand-alone programs. The administrator application resided on the game administrator's computer and the player's application resided on the player's computer. The integration of the two was done solely through the exchange of data (decision inputs and performance outputs) over the LAN. The LAN-based storage acts as the data exchange medium that links the two application programs. Thus, if one wanted to replicate the above instance of a particular simulation game on the same network, the administrator component would allow the creation of multiple instances of the simulation game with the appropriate game-pertinent parameters stored in appropriate folders within the LAN storage environment. Thus game-specific aspects such as number of industries, number of companies, number of students in a company (if the game tracks that) and industry, economic, product initialization parameters would all be stored in a configuration file on the LAN storage. The recognition of a particular participant as belonging to one specific instance would be done by storing the parameters on the LAN storage and initializing the player application with the relevant information.

The same architecture would be repeated on multiple LANs which were each independent of the other. We recognize that the primary characteristic of a LAN-based configuration is the inherent de-centralization evident in the implementation. Since each instance on one LAN would be completely independent of another, the application would not need the functionality for tracking and maintaining a distinction between these multiple instances of the simulation game. A move to a web-based environment with a centralized web-server would thus entail additional program logic that is lacking or less important in a LAN-based implementation.

A WEB-BASED ENVIRONMENT

The primary advantage of a web-based environment is its ubiquity and accessibility in today's technological environment. As long as one has Internet connectivity on any computer at any location, one can potentially have access to the simulation game. The familiarity of the web and the browser indicates an unparalleled degree of user-friendliness (both for the game-administrator and the

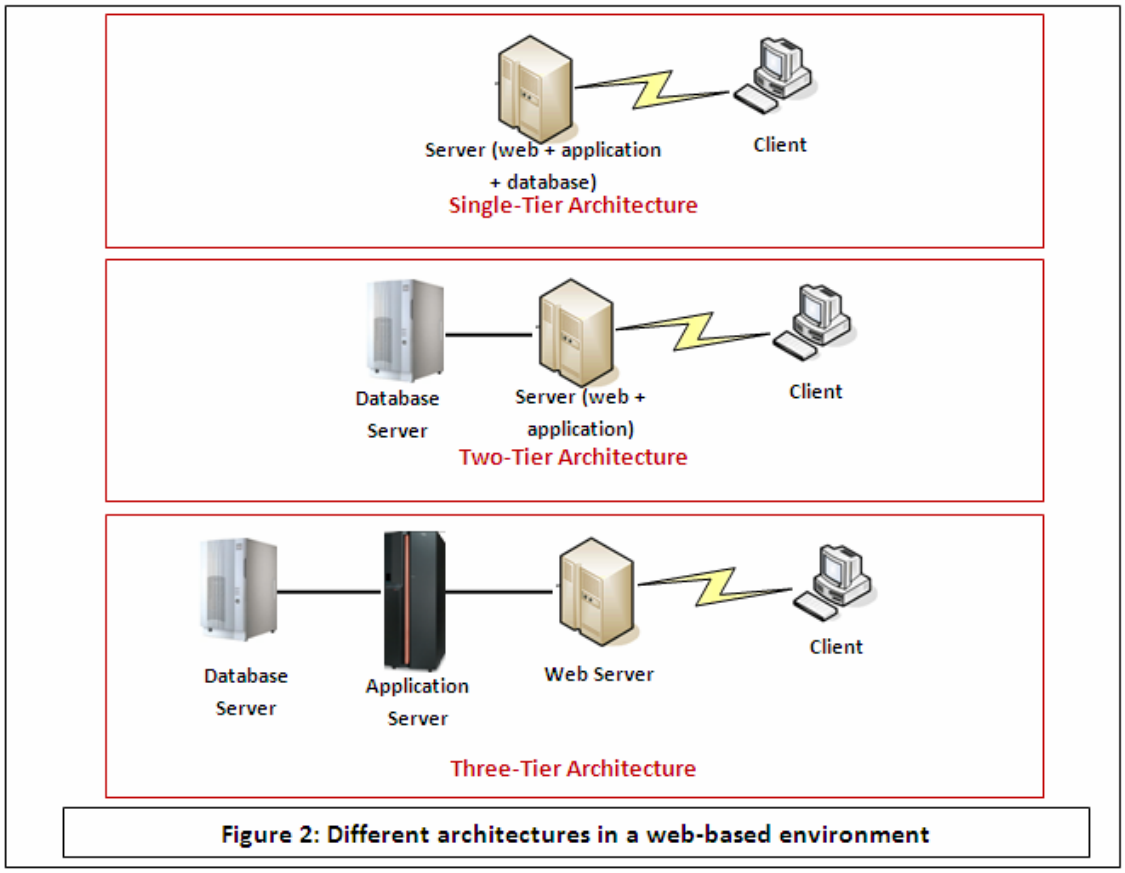

Figure 2: Different architectures in a web-based environment

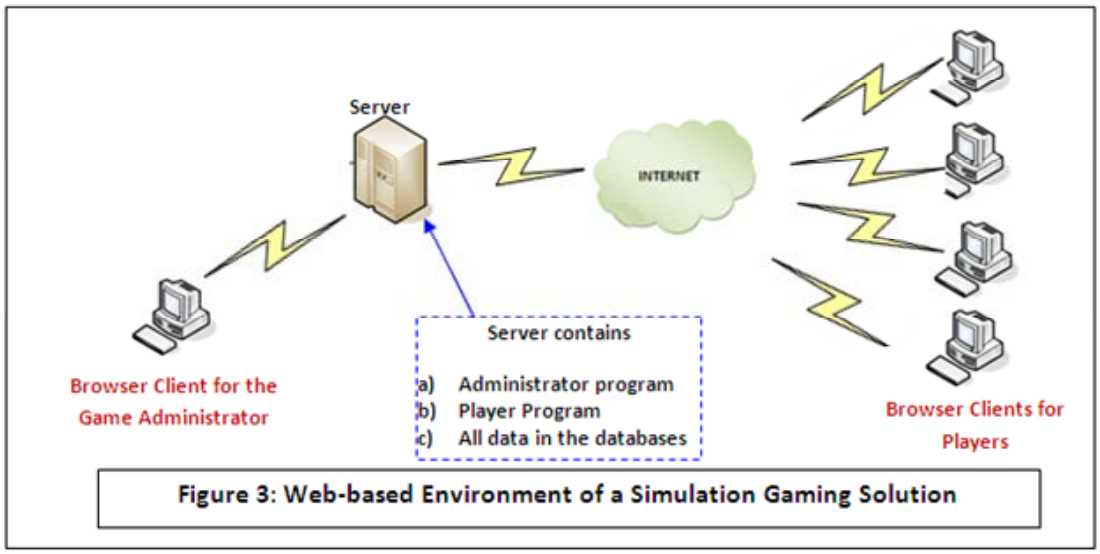

Figure 3: Web-based Environment of a Simulation Gaming Solution

player/participant) that might not have been available on a LAN-based application (depending, of course, on the specific application). Both of these characteristics make a web-based environment infinitely more attractive to the game administrator and the player/participant alike.

The universal availability characteristic, however, is premised on the notion that all aspects of the game are centralized on a single web-server. Admittedly, the actual architecture might vary from a single-tier to a two-tier to a three-tier architecture. See **Figure 2** for representative illustrations of each type of architecture. Nevertheless, regardless of this, a web-based implementation presumes a degree of centralization that did not exist in a LAN-based environment.

We notice therefore that unlike the LAN-based environment, in a web-based environment, all applications and data are centralized on the server (the actual location being dependent on the specific architecture as depicted in Figure 2). This characteristic raises a myriad of issues that were non-existent in a LAN-based environment. **Figure 3** depicts a typical web-based environment of a simulation gaming solution. A few aspects of this environment deserve explication.

Firstly, all application programs and data are centralized on the web-server. This centralization implies that while maintaining the multiple instances of a simulation game on a LAN was not a major issue, differentiating multiple instances of the game on a centralized web-server becomes a major issue. If there are ten different game administrators, each running two instances of the game, maintaining each of these instances separate and distinct involves identifying the instructor, the player or players (if a team-based player participation is employed), ensuring that each of them are presented with appropriate interfaces depending on their role.

Secondly, with newer architectures such as this, there tends to be an explicit database as opposed to file-based systems of the past. Most LAN-based simulations tend to be file-based representing the technological era during which they were developed. The usage of relational databases tends to provide for the explicit separation of data from program rendering the development of applications much more flexible. The availability of powerful query languages like SQL provides for more flexibility and power in data manipulation. Admittedly file-based systems – especially those applications that use binary files – may have an edge in terms of processing speed. However, since the data definitions are tightly coupled with the program any changes would be tedious and cumbersome. Most applications today tend to use relational databases. Developing the application can be done in a variety of Integrated Development Environments (IDE). Dreamweaver and Microsoft Visual Studio are two popular IDEs available. The advantage of using MS Visual Studio is that it provides a choice of programming languages including Visual Basic, Visual C++, Visual J++ (Microsoft's version of Java), and Visual C #(pronounced C-sharp). The IDE also comes with the SQL server express database engine which permits all aspects of the application to be developed within the IDE. We will use the Visual Studio 2005 using ASP.NET (.NET framework version 2.0) technology in explaining the issues that need to be considered and ways of addressing them.

SECURITY AND IDENTIFICATION ISSUES

Moving from a LAN-based environment to a web-based environment as depicted above raises a variety of issues. Hart et al (2006, p.106) raise the issue of web applications that involve membership with the associated concepts of identity, authentication and authorization.

IDENTITY

There would be various types of users arriving at the website. Game administrators, Player/participants, Website administrator and casual browsers are some of the categories of people who would be utilizing the website. In order to present a customized environment to each user, it is

imperative to identify who the person is that is accessing the website.

AUTHENTICATION

In order to establish the identity of the person, some sort of authentication process needs to be employed. Certain credentials such as email address and password combination are normally used credentials. In order to authenticate a user, these credentials must be stored on the server.

AUTHORIZATION

Once the login credentials have been verified, the authorization process has to grant permissions to the user based on the type of user it is. Thus a game administrator will have access to all the simulation game instances that she is managing. She will have access to all the input data as well as performance results for all industries and/or companies. She should also have access to game parameters including initializing the game and modifying them. She should have the ability to structure two different game instances with different parameters though both instances may be concurrently running. The player needs to have access to only his inputs and his performance results.

PERSONALIZATION

The capability to personalize a site to reflect the preferences of the currently logged-in user is necessary to give the site a sense of community and belonging.

Thavikulwat and Chang (2007, p. 114) who adopt an Internet-based approach (rather than a web-based approach) state that "a common misconception is that "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 Website". They further state that "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 without clever programming". While the above statements may be true in single-instance simulation gaming solutions, it may not hold for situations where multiple instances of the same simulation game are running. If the server program is being run from different IP addresses, (as would be the case if multiple game administrators are running their own simulation game instances on their computers) there would need to be some sort of "pre-installation" on the client-side computer that would identify the server that the client program should connect to. In all of the web-based environments with centralized servers, however, the above issues of identity, authentication and authorization become critical.

ISSUES TO BE ADDRESSED IN PORTING TO A WEB-BASED ENVIRONMENT

In this paper, we discuss the issues raised in the earlier section and others using Microsoft's Visual Studio IDE as the development environment. This presumes that the web environment uses Microsoft-related infrastructure, i.e., the operating system is a Windows-based system, the web-server used is Internet Information Server (IIS), the .NET framework 2.0 is installed on the Web Server and the database used is MS-SQL server.

ISSUE 1: CONSISTENT SITE DESIGN

Consistent site design is a critical factor in the user friendliness of a site and most websites on the Internet tend to have a generic site layout. Most websites have a header with the site/company logo and title, a menu bar on the left (or top) with links that allow various functionality, a footer with copyright information and other general information.

ASP.NET 2.0 provides tools such as Master page and

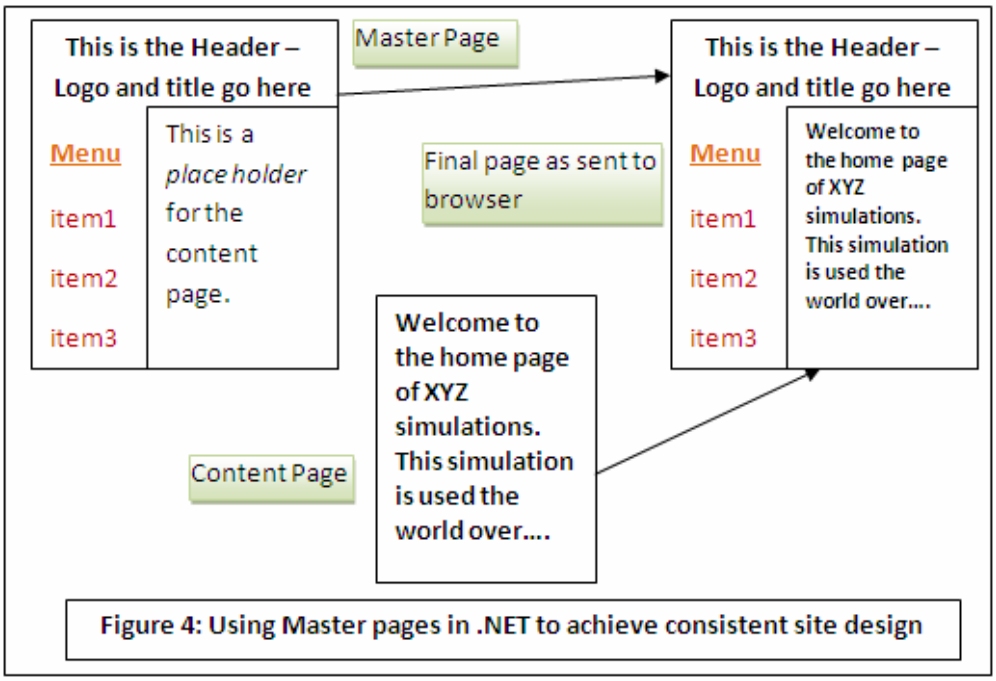

Figure 4: Using Master pages in .NET to achieve consistent site design

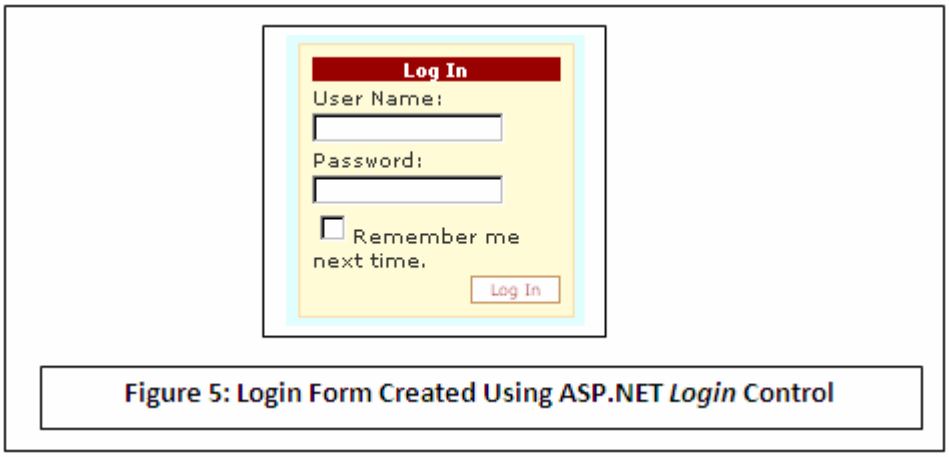

Figure 5: Login Form Created Using ASP.NET *Login* Control

Content page system to generate this consistent layout. **Figure 4** shows the process of creating pages for a website which have a consistent look and feel. So regardless of who the user is or where the user is in the application, consistency is maintained just in a traditional client-server program on a LAN-based or standalone program. The menu is created in ASP.NET by creating an XML file that contains the items in a *web.sitemap* file. More of this is explained in the section pertaining to Issue # 3.

ISSUE 2: AUTHENTICATION

ASP.NET provides three forms of authentication – Forms authentication, Windows authentication and Passport authentication. Forms authentication is the most widely used method which does not place any restrictions on the client infrastructure nor does it require a hook into other servers such as Microsoft Passport server or Windows Active Directory. **Figure 5** shows how a login form that can be created by using the ASP.NET *Login* Control.

In this case, the username and password need to be in the database along with other information related to the user profile. The user profile would include the user preferences, user permissions and the user role –game administrator, website administrator or player. Once logged in, it is this user profile that will be used to provide appropriate authorization and customization/personalization of the web interface.

ISSUE 3: AUTHORIZATION

It is to be kept in the mind that generic information about the simulation game will be provided in the main page for casual browsers. It would be unreasonable to require casual visitors to the website to login. A login should be required only for users with specific roles. ASP.NET accomplishes this by allowing access to entire folders using permissions as listed in the *Web.config* file. Below is the snippet of code that allows this functionality.

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <system.web>
    <authorization>
      <allow roles="gameadministrator,
gameplayer" />
      <deny users="*">
    </authorization>
  </system.web>
</configuration>
```

This file resides in the folder that contains all the game functionality for administrators and users alike. All other users are denied from entry into this folder. This is the basic level of authorization that distinguishes between causal visitors to the site and other users who actually play the game or administer it. The other issues that arise are visibility, access and authorization. Visibility and access are control by both the *Web.sitemap* and the *Web.config* file. Below is the snippet of code in the *Web.sitemap* file that shows how a particular page or link is made visible and/or accessible to specific people.

```
<siteMapNode title "Enter Decisions"
    url="player/enterdecision.aspx" roles="gameplayer,
    gameadministrator">
<siteMapNode title "View Results"
    url="player/viewresults.aspx" roles="gameplayer,
    gameadministrator">
<siteMapNode title "Initialize Game Parameters"
    url="administrator/initialize.aspx"
    roles="gameadministrator">
<siteMapNode title "Run Game"
    url="administrator/rungame.aspx"
    roles="gameadministrator">
<siteMapNode title "Change Game Parameters"
    url="administrator/changeparameters.aspx"
    roles="gameadministrator">
```

The above code will ensure that the "Enter Decisions" and "View Results" links are visible to both the game player and the game administrator. However the other links such as running the game or initializing the game parameters are visible only to the game administrator. In addition to visibility of the link, access to the functionality is also restricted at the folder level by the entries in the *Web.config* file. This can be done as shown below.

```
<location path="gameadministrator/rungame.aspx">
  <system.web>
    <authorization>
      <allow roles="gameadministrator" />
    </authorization>
  </system.web>
</location>
```

If the above snippet of code is added to the *Web.config* file, this ensures that only those people whose role is defined as a "gameadministrator" will be able to access this page.

ISSUE 4: DATABASE DESIGN

All of the above discussion assumed that some basic information already exists in the database. Primary tables in the database will include:

1. A user table which has all user related information including user preferences and roles.
2. A parameter table with all initial values for the game to be started.
3. A game administrator and game instance table that tracks each game instance created and the associated administrator.

Beyond this there would be no other information in the database. Each time a new instance of the simulation game is created, a new table will have to be created that would have the structure of the game including the number of industries and/or companies as well as the teams as well as individuals associated as game player/participants with that industry/company. This functionality would have to be available to each game administrator. For each such game instance, as the simulation progresses there will be tables created for decisions, performance results, and history information. The new tables would therefore be

1. An industry/company table – which would contain data about the number of industries/companies and the associated teams and individual users.
2. A decision table – which would include values for every decision for every period.
3. A performance results table – which would include all the performance parameters
4. A history table – which would store all history information needed to advance the simulation to the next period.

Thus each simulation game instance will generate multiple tables and each of them could be relatively large depending on the number of industries and the number of periods simulated for. A decision needs to be made whether each simulation instance will therefore be an independent database or will all these tables be in one large database with the different tables associated with various game instances being appropriately labeled and identified. There are pros and cons to each approach. Creating a single database would mean that all information is available in one database and the capability exists to access any and all information tempered, of course, by the persons role and the attendant permissions. This kind of flexibility might allow a game instructor to perform cross-simulation-instance analyses and thereby gain valuable insights into differences in performance. This might, in turn, aid her in her quest to further refine her pedagogy. On the other hand databases that tend to grow large might falter in the area of responsiveness. This would be a serious issue that would argue against adopting a large database approach. Responsiveness could be maintained and tremendous scalability would exist regardless of the number of simulation game instances that were created. It might still be possible to do cross-instance analyses though the coding might be quite cumbersome and tedious. Another issue is the licensing of databases. Many ISPs provides rates depending on the number of databases used. If the number of databases keeps increasing as time goes on, the cost of maintaining such a large number might be prohibitively expensive. Admittedly, one could retire older databases and take them offline to mitigate the cost impact of this approach.

CONCLUSION

We have identified some significant issues that need to be addressed in any move from a LAN-based simulation gaming environment to a web-based environment. While the latter offers convenience, universal access, powerful technology and functionality such is move is a non-trivial issue. Any game developer would be well advised to consider all the issues and design the web-based environment with proper fore-thought. We have identified four primary issues that need to be considered including site design, authentication, authorization and database-related issues.

The paper has confined its analysis to simulation gaming environments that fall under the fixed scaling, administrator-driven, synchronized simulations. The above analysis would nevertheless, apply to most other simulation games regardless of the category they fall under vis-à-vis the other dimensions including purpose, control and interaction, representational system or scoring. While this paper has identified some key issues that need to be addressed, other issues not discussed here might arise if one were to consider an unsynchronized simulation or one that was clock or activity-driven or one that had flexible scaling, the most complex one being continuous scaling simulations. To the best of the author's knowledge most business simulation games extant today would fall under the ambit of this paper. However, to further advance the theoretical knowledge of simulation gaming, other categories not considered here are interesting research areas that beckon.

REFERENCES

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

Forio Simulations. (2007). http://forio.com/resources/category/forio-simulations/, accessed on October 31st, 2007.

Hall, Jeremy J.S.B. (2005). "Computer Business Simulation Design: The Rock Pool Method". *Developments in Business Simulations and Experiential Learning*, Vol. 32, 144-154.

Hart, C., Kauffman, J., Sussman, D., Ullman, C. (2006). *Beginning ASP.NET 2.0*. Wiley Publishing, Indianapolis, Indiana.

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

Thavikulwat, P. (1999). Developing computerized business gaming simulations. *Simulation and gaming*, Vol. 30, 361-366.

Thavikulwat, P. (June 2004). "The architecture of computerized business gaming simulations". *Simulation and Gaming*, Vol. 35, No. 2, 242-269.

Thavikulwat, P. and Chang, J. (2007). "Applying .NET Remoting to a Business Simulation". *Developments in Business Simulations and Experiential Learning*, Vol. 34, 113-118.