ONLINE EVALUATION SYSTEM

A Thesis
Presented to
The Graduate Faculty of The University of Akron

In Partial Fulfillment
of the Requirements for the Degree
Master of Computer Science

Jeromie L. Walters
May, 2005
ABSTRACT

This paper describes the design and implementation of a web-based online evaluation system (OES). The original concept of OES was to develop a generic evaluation system capable not only of allowing students to evaluate instructors, but also allowing instructors to evaluate teaching assistants, supervisors to evaluate employees, and so on. Additionally, the system would be openly available on the Internet, accessible to any number of users. New programming tools like .NET and SQL Server make web-based and database development easier than ever, but designing a project of this magnitude still required much effort. As with any major project, conceptual design changed throughout the development process, and numerous variables such as security and the presentation of data were taken into consideration. As such, OES is a culmination of user roles, file formats, and web-based and database programming. The development of OES, from initial concept to final implementation, along with a survey of similar evaluation systems, is discussed in detail throughout the remainder of this document.
DEDICATION

I dedicate this thesis to my wonderful parents, Barry and Shona Walters, without whose love and guidance I would never have realized this dream.

God bless you always, Mom and Dad.
ACKNOWLEDGEMENTS

First and foremost, I praise God for allowing me to pursue my dreams and hope I can someday use my talents for Him. I thank God for my loving family, for the encouraging friends He has placed in my life, and for making this thesis possible.

I would also like to acknowledge the professors who have most influenced me. Dr. Price, thank you for developing the original concept of OES and for being an advisor to me. Dr. Pelz, thank you for advising me throughout my entire university experience.

Dr. Xiao, thank you for your help as my professor and advisor, for motivating and encouraging me, and for genuinely caring for my personal and professional development.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>LIST OF FIGURES</th>
<th>viii</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAPTER</td>
<td></td>
</tr>
<tr>
<td>I. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>Overview and Survey of Online Evaluation</td>
<td>1</td>
</tr>
<tr>
<td>Benefits and Challenges of Online Evaluation</td>
<td>4</td>
</tr>
<tr>
<td>II. BACKGROUND</td>
<td>7</td>
</tr>
<tr>
<td>Unified Modeling Language</td>
<td>7</td>
</tr>
<tr>
<td>Technology and Tools</td>
<td>7</td>
</tr>
<tr>
<td>Web Services</td>
<td>8</td>
</tr>
<tr>
<td>III. DESIGN</td>
<td>9</td>
</tr>
<tr>
<td>UML Design Diagrams</td>
<td>9</td>
</tr>
<tr>
<td>Database Design Diagrams</td>
<td>17</td>
</tr>
<tr>
<td>IV. IMPLEMENTATION</td>
<td>21</td>
</tr>
<tr>
<td>Authentication and Authorization</td>
<td>21</td>
</tr>
<tr>
<td>OES File Formats</td>
<td>23</td>
</tr>
<tr>
<td>Form Administrator Authentication File</td>
<td>23</td>
</tr>
<tr>
<td>Form User Authentication File</td>
<td>25</td>
</tr>
<tr>
<td>Form Data File</td>
<td>27</td>
</tr>
</tbody>
</table>
Question Rows ........................................................................................................ 28
Response Rows .................................................................................................... 30
OES Form Builder ............................................................................................... 31
OES Web Services ............................................................................................... 34
V. SUMMARY AND FUTURE WORK .................................................................... 37
VI. REFERENCES .................................................................................................. 43
### LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 OES Use Case Diagram</td>
<td>10</td>
</tr>
<tr>
<td>2 OES Conceptual Class Diagram</td>
<td>12</td>
</tr>
<tr>
<td>3 OES Security State Diagram</td>
<td>14</td>
</tr>
<tr>
<td>4 OES State Diagram</td>
<td>16</td>
</tr>
<tr>
<td>5 OES Database Diagram</td>
<td>18</td>
</tr>
<tr>
<td>6 OES Account Database Diagram</td>
<td>19</td>
</tr>
<tr>
<td>7 Sample OES Form Administrator Authentication File</td>
<td>24</td>
</tr>
<tr>
<td>8 Sample OES Form User Authentication File</td>
<td>26</td>
</tr>
<tr>
<td>9 Sample OES Form Data File</td>
<td>27</td>
</tr>
<tr>
<td>10 OES Evaluation Form HTML Layout</td>
<td>33</td>
</tr>
</tbody>
</table>
CHAPTER 1
INTRODUCTION

Overview and Survey of Online Evaluation

Traditional course evaluations, specifically in institutions of higher education, require students to fill out at least one paper form to rate an instructor or course. These forms, such as the IDEA (Individual Development and Educational Assessment) form, are typically filled out by hand with a pencil at the end of every semester and are often very tedious to complete, forcing students to fill in bubbles to rate an instructor on a numeric scale. The purpose of the online evaluation system (OES) project was to implement a web-based system capable of accomplishing the same tasks as paper-based evaluations that would be easier and more cost-effective to use. Knowing that some other universities had already implemented online evaluation systems, a goal of this project was to go beyond what other schools had done with the creation of a “generic” system. The purpose of this generic system would be to allow evaluations to be performed for more than just university purposes; supervisors at a company could be evaluated, evaluations could be set up for online polls, and so on. It was discovered late in the development phase that such a generic system already existed in a product called 2WAY from Modern Mind Software, but the benefits of OES over this system shall be discussed. Before any code was written, it was necessary to determine what other
schools had done with regard to online course evaluation, and a summary of some of these systems follows.

Officials at Deakin University in Australia knew they had to do something to save time and money wasted on paper-based evaluation. Off-campus students were required to mail in their evaluation forms, and coding those forms into digital format would take up to three months to complete. In 1997, they implemented an HTML and CGI-based online system that yielded a forty to fifty percent student response rate. Deakin released their final online evaluation system, which provided different forms for on- and off-campus students, in 1998. Students could log in with a unique ID and choose a course to evaluate from their list of pending evaluations. Features of the system included language censoring, anonymity, and publicly-available response rate and summative (overall) ratings data. Only staff could view detailed formative information, such as student comments. Results showed that the online system yielded response rates at least as good as their former paper-based system, particularly for off-campus students. Instructors also noted that students provided longer, more useful, thought-out responses to online evaluations.

Drexel University also released an online evaluation system in 1998 that was based on HTML, SQL, and the Perl scripting language. Instructors would submit questions on a template email, which would be uploaded as an evaluation form into the system. Students used their name and birth date to log in and submit evaluations. Email was used as a main source of communication to students and faculty, reminding students to fill out
their evaluations and providing instructors with real-time response rate data during the evaluation period so they could encourage more participation if necessary.

Columbia University implemented their WCES (Web Course Evaluation System) in 1997. Their system, in addition to end of semester evaluation, was capable of performing midterm course evaluations. WCES allowed faculty to customize their surveys and was linked directly to the registrar’s office for security and for obtaining student course information. A public web site allowed anyone, including prospective students, to view evaluation results by professor or course.

Finally, three Chinese universities (HKUST, HKU, and HKPU in Hong Kong) collaborated for the creation of two online evaluation systems, COSSET (Centralized Online System for Student Evaluation of Teaching) and OSTEI (Online System for Teaching Evaluation of Instructors). COSSET contained many features of previously-discussed evaluation systems, while OSTEI was a formative questionnaire-based system. COSSET relied on registration information for student logins; OSTEI used a combination of instructor ID and questionnaire ID for students to log in and was considered less secure than COSSET. One major benefit of OSTEI was the instructor’s ability to create a custom questionnaire. An instructor would register on the system, creating their own user ID, password, and questionnaire ID. They also had access to built-in questionnaires and a question bank consisting of 800 questions that could be used on a custom questionnaire. A survey of students using COSSET showed that seventy-two percent preferred the online system to paper-based evaluation, and sixty-five percent found the
login method, using their registration ID, acceptable. In another survey, 129 of 455 instructors (twenty-eight percent) chose COSSET over paper-based evaluations for their appraisal.

**Benefits and Challenges of Online Evaluation**

Having implemented online evaluation systems, the schools noticed some obvious benefits of web-based evaluation over their former paper-based systems. Immediate and flexible feedback was one such benefit; instructors could view real-time results presented in various formats. The online systems were also much easier to use than paper-based evaluations; students could click radio buttons and type in textboxes instead of filling in bubbles and writing by hand. Instructors could customize evaluation forms and, since the forms were already in a digital format, time and money typically spent on administrative tasks (administering, collecting, and scanning paper forms) was saved. Digitally stored data could also be readily used for analysis and establishing historical trends. Online evaluations were more convenient for students, who provided more useful, thought-out responses to open-ended questions on the online forms. Using digital evaluations also saved the cost of paper, a small but noteworthy environmental benefit. Finally, online systems had the added benefit that users could be sent email with links to the evaluation forms and results, making it easier to send reminders to students and results to instructors or administrators. Emails could even be customized on a per user basis, such that instructors could see detailed formative results and department leaders or a dean could see summative statistical results.
The schools implementing the systems described above also noted some challenges of using online evaluation, most of which could be overcome with proper measures. For example, instructors were concerned that the online systems allowed any student, even students who may not have actually attended, to evaluate a course. Yet it would be unlikely that a student lacking initiative to attend a course would be motivated to go online to perform an evaluation. Instructors also worried that, since they performed the evaluations on their own time, students might influence each other to rate a course or instructor a particular way. Perhaps the largest difficulty observed by the schools was low response rates. A Drexel/Columbia non-response bias study found that women, upper-classmen, and those with higher GPAs were more likely to respond to online evaluations. Response rates were greatly improved through the use of email reminders and promotion from campus posters, campus newspapers, and instructors. Response rates were also reportedly higher when lab time was scheduled to perform evaluations or when evaluations were promoted with incentives, such as pizza parties and Palm pilot giveaways. A major concern of students was that their evaluation responses would not be confidential because of necessary login methods, but these concerns were alleviated when students were guaranteed their responses could not be identified and would not have any effect on grading. The last major challenge was “culture change”, a natural bias against the online system simply because it was not the familiar way of performing evaluations. Culture change was overcome with promotion, a common and necessary component of all successful online evaluation systems. Midterm evaluations and faculty training
seminars, to emphasize the importance of the evaluation process and how to encourage it in class, were also potential solutions to the challenge of culture change.
CHAPTER II
BACKGROUND

Unified Modeling Language

Having researched the basics of online evaluation systems, further research was conducted to learn the Unified Modeling Language (UML), a language defined to standardize conceptual design of software processes. There are a variety of UML diagram types, some of which are used to illustrate expected usage scenarios (activity diagrams), class relationships in an object-oriented environment (class diagrams), and flow of control in a software system (sequence and state diagrams). A full description of UML is beyond the scope of this paper, but a few specific UML diagram types were vital to the design process of OES, and those specific diagrams are explained in the following design chapter.

Technology and Tools

One of the many goals of OES was to use modern development technologies such as .NET and web services. More specifically, OES has been developed with Microsoft’s new Visual Studio .NET development environment. It has been written with the C# (pronounced “C-sharp”) programming language, which is often compared to the Java programming language. OES is an ASP.NET web application to be used on the Internet. Client code is written in HTML and JavaScript, with potentially embedded server-side
script written in C#. Code that executes on the server to process event handlers for events such as button clicks is also written in C#. Finally, a trial download of Microsoft’s SQL Server 2000 was used for database operations, including the storage and retrieval of login information, as well as evaluation form display and submission data. OES relies heavily on database access, as shall be seen in the following chapters.

**Web Services**

Web services are a new frontier in online development that focuses on providing useful functions to web-based applications. Web services are often referred to as “XML web services” because their functions use XML (eXtensible Markup Language) as their underlying layer for transferring data. XML web services have their own language, WSDL (Web Service Definition Language), which is used to describe the functions and parameters implemented by them. Web services add value to the Internet by allowing a web application on one server to call a function, such as a calculator or global time function, on a remote server. OES web services were implemented in C# to perform the data storage and retrieval operations necessary to implement an evaluation system, as described in the implementation chapter.
CHAPTER III
DESIGN

UML Design Diagrams

As with most projects, the design of OES was improved, and naturally evolved, during the course of development. Some driving factors behind using the .NET development environment were the ability to create web services and to use the .NET role-based security model. Working from these basics led to the development of three types of UML diagrams used to describe the operation of OES: a use case diagram, a conceptual class diagram, and two state diagrams.

A use case diagram describes the interactions of users and components in a software system. The following use case diagram was developed to show what basic actions could be performed by members of each OES role.
The idea of serving multiple companies, universities, and groups led to the development of making OES an account-based system. A single user, called the account administrator, would register a new OES account on behalf of their organization. An administrator (later in the development phase and henceforth referred to as a “form administrator”) could create and modify evaluation forms, which would be filled out by evaluators. An

Figure 1: OES Use Case Diagram
evaluatee was the user being evaluated by a specific evaluation form, and a reviewer would be able to view the results of an evaluation.

Once roles were defined, it was necessary to think about the objects that would be implemented in an object-oriented evaluation system. The following diagram shows the initial OES conceptual class diagram, a basic blueprint for the system’s implementation.
While the conceptual class diagram may appear complicated, there are a few concepts easily grasped from the illustration. First, the diagram follows the three-tier model of
web-based design; it makes use of a client web application, an intermediate layer (web services), and a server database. Connecting lines show the relationship between objects; for example, there is a one-to-one relationship between the web service and database layers because one service accesses one database, but there is a many-to-one relationship between the client application and web service layers because many clients on the Internet may access a single web service. The EvalItem, GUIItem, and GUIRenderer classes were initially designed to specify how an HTML evaluation form would be created, but these classes have been merged into the FormBuilder class, which shall be presented in the next chapter.

The last type of UML diagram created for OES was the state diagram. Having considered the roles to be implemented in OES, and wanting to use the .NET role-based security model, the following security state diagram was developed.
This security scenario is similar for the "Submit Evaluation" and "Review Evaluations" use cases as well. The text "Create/Modify Evaluation Form" would be replaced with the corresponding use case. Roles of "Evaluator" and "Reviewer" (instead of Administrator) would be used for the submit evaluation and review evaluations use cases, respectively.

Figure 3: OES Security State Diagram
The OES security state diagram shows the process of authentication and authorization executing from the top of the diagram toward the bottom of the diagram. A user must first be authenticated by having provided valid login information. Following authentication, a user may only perform their desired action if they are authorized (have the proper role) to do so.

Another state diagram was used to show the flow of control in OES. The OES state diagram below shows the operation of users in each role.
User may cancel or return to the "Login" state from any state before "Server Processing."

Self-transition on login state implies a failed login attempt.

Figure 4: OES State Diagram
The OES state diagram was designed to show the operation of each user of the various roles within the system. Execution would begin at the top of the diagram and proceed toward the bottom, starting with a login attempt and ending with server processing. Each role follows a specific line of execution; for example, an account administrator’s actions are shown along the left-hand side of the diagram. Note that a reviewer’s execution loops back from the server processing step when a reviewer chooses to view evaluations for more than one evaluatee. Additionally, an evaluatee’s execution path is not shown because the evaluatee role is basically a subset of the form administrator and reviewer roles (see the use case diagram above).

**Database Design Diagrams**

The last design diagrams to be drawn were OES database diagrams. At the start of the OES project’s development it was realized there would be a need for separate databases for each account, which would make accounts self-contained and confidential. Account administrators would find peace in knowing that their data was not kept together with data from any other organization, and keeping all data for an account in its own database would make account deletion as simple as dropping a single database. In order to keep track of the different accounts in OES, a single table was created in the OES database, and the information it contains is shown below.
The OES database contains just one table, which is called “Accounts”, as evident by the table’s heading. Bolded fields in the diagram are required; they may not contain the database value NULL. The “PK” symbol indicates that the field “AccountID” is a primary key, a unique identifier for each row in the table. Only basic registration information is stored in the “Accounts” table to uniquely identify each account registered with OES. Account databases are much more complex than the OES database, containing multiple tables for account authentication information, as well as evaluation form display and submission data. Tables were added as needed during the development phase to accommodate new requirements, yielding the final account database design diagram below.
Figure 6: OES Account Database Diagram

From the diagram it is clear that eleven tables are required to keep track of all user and form data for a single OES account. The “FormQuestionData”, “FormResponseData”, and “FormGuiData” tables are used to store information regarding how an evaluation form should be displayed. The “CreatedForms” table stores basic information about the forms created in the account. The “FormDrafts” and “SubmittedForms” table store evaluators’ form submission data. The remaining tables store data to authenticate and authorize users of the account when they log in to OES. Each table in the diagram is similar to the OES database’s “Accounts” table with respect to table name, required fields, and primary keys. However, some tables in this diagram use “clustered” keys,
where more than one field combines to uniquely identify each row in the table. For example, the “FormQuestionData” table uses its “FormID” and “QuestionNumber” columns to uniquely identify each row, which is indicated with a “PK” symbol next to both fields. Finally, this diagram uses arrows and “FK” symbols to show foreign key relationships among tables. Each arrow extends from a child (dependent) table to its parent table (the table it depends on), and the foreign key in the child table matches the primary key in the parent table. For example, the “FormEvaluators” table depends on the “CreatedForms” and “Users” tables. The “FormID” foreign key “FK1” matches the “CreatedForms” table’s primary key, and the “UserID” foreign key “FK2” matches the “Users” table’s primary key. Thus a table may have more than one foreign key, and these keys may also be clustered such that more than one field combines to create the foreign key, as in the case of the “FormDrafts” table. The practical reasoning behind foreign keys is the concept of referential integrity (RI). A new form administrator cannot be added to the “FormAdministrators” table unless that user already has a user ID stored in the “Users” table. Additionally, user data cannot be deleted from the “Users” table if the user has an entry in the “FormAdministrators” table; the “FormAdministrators” reference would have to be deleted before the user could be deleted from the “Users” table.
CHAPTER IV
IMPLEMENTATION

OES has been developed in the C# .NET programming language, using HTML and JavaScript for client-side code and C# server-side code (which includes server-side script embedded in HTML pages). OES takes full advantage of role-based authorization in the .NET security model and also implements its own web services that can be called by various other web applications on the Internet. The sections that follow describe the implementation of the major components of OES.

Authentication and Authorization

The .NET security model operates in two basic modes: windows authentication and forms authentication. Windows authentication uses a user’s login information, as provided when they log into the Windows operating system installed on their local machine, to authenticate a user. For web-based system like OES, the forms authentication method is much more practical. Forms authentication allows a user to log in with a username and password they provide on an HTML form. If valid login information for all users of a system is stored in a SQL Server database, as it is for OES, forms authentication can look up a username and password in the database to authenticate a user attempting to log into the system.
The sibling component of forms authentication is URL authorization. URL, or role-based, authorization can be thought of as folder-based protection. When a user is authenticated with forms authentication, their roles are assigned to their identity in the form of an Internet cookie that is used for authorization. A web configuration file is used to specify which folders users of specific roles may access. If a user is not authenticated (has not provided a valid login) or attempts to access a folder they are not authorized to access, they are redirected to a web page specified in the web configuration file.

When a user logs into OES, they specify which account they are logging into and provide a username and password. Their username and password are looked up in the specified account database’s “Users” table to authenticate the user. Furthermore, an authenticated user’s ID is looked up in each of the account database’s role tables (FormAdministrators, FormEvaluators, FormEvaluatees, and FormReviewers) to build a list of the user’s roles. The user’s ID is also looked up in the OES database’s “Accounts” table to determine if the user is the account’s administrator. All of the user’s roles, since users may be members of multiple roles, are attached to their identity in the form of a cookie used to authorize them on OES web pages. After successfully logging into OES, users are taken to an action page that dynamically presents options to them based on their roles. Users who are not authenticated or authorized are redirected to the OES home page, which is also the OES login page.
OES File Formats

OES uses Microsoft Excel files to specify authentication information, which is loaded into SQL Server tables in an account database. Additionally, OES uses a custom Excel file format to specify how a form should be displayed. Original design dictated that OES would allow a form administrator to build a form’s interface graphically, but it was discovered that Modern Mind Software’s 2WAY product already had this capability, and it seemed that using a custom Excel file format would make form creation and modification simpler. The file formats used by OES are discussed in further detail in the following sections.

Form Administrator Authentication File

This section describes the format of the Excel form administrator authentication file uploaded by an account administrator during the account registration process. The uploaded form contains login information for an account's form administrators and contains just a single sheet named "Sheet1", which is created by default when opening a new Excel workbook.

In order to create an OES account, the account administrator must specify which users may use their account to administer evaluations. The form administrators are able to create evaluation forms and specify which users may log in to the account to evaluate and review their forms. Additionally, form administrators may assign users to the evaluatee role; evaluatees may add a subform to a form administrator's form and review partial evaluation results. A form administrator may choose to assign users to multiple
roles, such as making an evaluatee also a reviewer so they may view complete evaluation results. A sample form administrator authentication file is shown below.

Figure 7: Sample OES Form Administrator Authentication File

The Excel form administrator authentication file should have two columns on sheet "Sheet1", one for form administrator IDs and another for passwords. The first cell of the ID column must contain the text "UserID" and the first cell of the password column must contain the text "Password". Note that the term “UserID” in this context actually refers to a username used during the login process and is not related to the “UserID” of an account database’s “Users” table, which is actually an integer value used to uniquely identify the user. Finally, form administrator IDs and passwords may not contain single or double quotes and are limited to sixteen characters in length.
**Form User Authentication File**

This section describes the format of the Excel form user authentication file uploaded by a form administrator during the form creation process. The uploaded form contains login information for the users of a form and contains just a single sheet named "Sheet1", which is created by default when opening a new Excel workbook.

In order to create an OES evaluation form, a form administrator must specify which users may use their form. Users may have three different roles: evaluator, reviewer, or evaluatee. An evaluator may evaluate the form, perhaps saving drafts of their form before their final submission. A reviewer may view full response data for an evaluation and has the ability to page through submitted evaluation forms. An evaluatee may attach a subform to the form any time before the beginning of the evaluation period. Evaluatees also have limited reviewing capability - they may page through submitted forms but can only view response data for their subforms. A sample form user authentication file is shown below.
A form user authentication file should contain three columns for each user of an evaluation form: UserID, Password, and Roles. UserID and Password correspond to the login information required for a user to log in to the form administrator’s OES account and use their form; user IDs and passwords may not contain single or double quotes. User IDs in the form user authentication file correspond to usernames used to log into OES and are not related to the integer ID created for the user in the account database’s “Users” table. The Roles column may contain either of "evaluator", "reviewer", or "evaluatee", and corresponds to the roles described above. Users may be assigned to multiple roles by separating the roles with a "|" character. Also note that the headings "UserID", "Password", and "Roles" are required, and user ID and role values are case-insensitive.
Form Data File

This section describes the format of the Excel form data file uploaded by a form administrator during the form creation process. The uploaded form represents an evaluation form in OES and contains just a single sheet named "Sheet1", which is created by default when opening a new Excel workbook.

Sheet1 specifies which questions, response options, and user controls should appear on an evaluation form and how they should be displayed. The following displays an example form data sheet, a hypothetical music course survey demonstrating some typical form requirements. A full explanation of form syntax follows.

![Microsoft Excel - FormData.xls](image)

Figure 9: Sample OES Form Data File
Question Rows

Sheet1 can be thought of as a series of question rows followed by rows of response options. A question row must precede its response rows. Before delving into the details of specific column values, it can already be seen on the sample Sheet1 that the first question is "Who sang 'Strawberry Fields'?" and the possible response options are "Beatles", "Simon and Garfunkel", "Bob Dylan", and "The Monkees". The columns in a form data file have different meanings for question rows than they do for response rows and are referred to by the letters shown at the top of each column.

Each question row starts with a question number in column A, which determines the question's placement on the form. A question that appears first in Sheet1 does not have to be displayed first on an evaluation form; question numbers are sorted in increasing order, so a question’s position on a form can be easily changed by changing its number. Also, although question numbers are shown starting from one in the sample, they may span any range of numeric values and skip numbers as well. Finally, question numbers are text values, which means custom number formats such as {"Q1", "Q2", "Q3"} and sub-level numbers such as {"1.1", "1.1.1", "1.1.2"} can be used. Note, however, that no two questions can have the same question number.

Columns B and C correspond to a question’s parent question number and parent response ID, respectively, and combine to allow question nesting. If columns B and C are both zero, a question is a top-level question; it is not nested within any other question. If B and C are numbers other than zero, the question is a "child" question nested within
another question (called the "parent" question) and will only be visible when one of the
parent question's response options is chosen. Column B specifies which question will be
the parent of the current question, and column C specifies which parent response option
must be chosen for the current question to be displayed. Multiple levels of nesting are
allowed.

Column D of a question row contains the text of the question as it should be
displayed on the evaluation form. Question text will be displayed to the right of the
question's number. Column E, the question’s response type, specifies what kind of input
is expected from an evaluator. If response options are all text values, then the response
type should be "text". If response options for a question are all numeric, such as in the
case of rating an item from one to five, then the response type should be "number".

Columns F and G allow a form administrator to specify what kind of control should
be used to obtain an evaluator's input on the form. Column F may have only one of two
values, either "select" or "textbox". The select control type will render a different control
set depending on its mode specified in column G. If its control mode is "single", the
select control will produce a set of radio buttons for the group of response options listed
for the question. If the select control's mode is "multiple", a set of checkboxes will be
created for the question's response options. Other select control modes which render
checkboxes include "max = n", "min = n", and "exactly = n", where 'n' is any integer
greater than zero. Max mode allows the evaluator to choose at most 'n' items, min mode
requires them to choose at least 'n' items, and exactly mode requires them to choose
exactly 'n' items. Finally, the textbox control renders a textbox so the evaluator may type
their input. Control mode "single" renders a single-line textbox and control mode
"multiple" renders a multi-line textbox.

Any column after G may specify additional properties for the controls displayed on
the evaluation form. Currently only two properties are supported: select controls may
have only a "range" property and textbox controls may have only a "maxlength" property.
The select control's range property allows a form administrator to create a range of
numeric response options instead of typing out all of the options beneath the question
row. The range property's format is "range = start, stop, step", where start is the numeric
starting value, stop is the numeric ending value, and step is the numeric step between
values. Start, stop, and step may be integer or floating-point numbers. An example
property "range = 1, 5, 1" would create the response options \{1, 2, 3, 4, 5\}.
Alternatively, a range can be specified in decreasing order such as "range = 5, 1" to create
the response options \{5, 4, 3, 2, 1\}. Note that an omitted step value defaults to one, and
step values should always be greater than zero. The textbox "maxlength" property is
used to specify a maximum number of characters that may be entered into a textbox
control. All form inputs are limited by default to 8000 characters, but input could be
limited to 200 characters by specifying the property "maxlength = 200".

Response Rows

Unlike question rows, response rows are not always necessary. When a control type
of "textbox" is specified in the question row or when a select control has a range
property, there is no need for response rows. Any response rows provided after a question whose control type is "textbox" will be ignored. Response rows appearing after a question with a range property will be added as response options after the range values, which could be useful for adding options that could not be directly included in the numeric range.

Column C of a response row specifies a response option's ID. Unlike question numbers, response IDs have no effect on a response option's position on an evaluation form. Response options are placed on an evaluation form in the order they appear in on Sheet1. Response IDs are used strictly for nesting as described in regard to columns B and C of question rows above. They are provided so that if a large number of questions are children of a specific response option, changing the response option's position on the form (by moving it to a different row on Sheet1) does not cause the link between nested questions and their parent response option to be lost. Note that when a range property is used in a question row, the response IDs are equivalent to the numeric response values. Any additional response rows after a question row with a range property must specify response IDs that do not conflict with the numeric range values. Finally, column D is the text of a response option as it should be displayed next to its radio button or checkbox on an evaluation form.

**OES Form Builder**

Towards the end of the development phase it was apparent that some OES functions were similar and were actually causing code duplication. For example, reviewing an
individual evaluation form is the same as viewing the form, with the extra step of filling in an evaluator’s selections or text. Resuming an evaluation form draft is the same as reviewing the form, except that input controls should not be disabled (as they are when reviewing forms). Viewing evaluation forms, and performing similar tasks such as reviewing forms and resuming drafts, required a lot of code to generate the dynamically-created HTML used to render a form. This code was encapsulated in a separate .NET class called “FormBuilder”.

The OES FormBuilder class resides in its own .NET namespace, a grouping of classes, called “OesFormBuilder”. The benefit of creating this namespace and putting the form building functions in their own class was code reusability. Any code page in the OES project needing to dynamically render an HTML evaluation form can simply declare its intention to use the OesFormBuilder namespace, at which point it has full access to the public functions in the FormBuilder class. Another major reason for implementing form building functions in their own namespace is the ability to compile the FormBuilder class in a separate project, which would create a FormBuilder dynamic link library (DLL). The web services described in the next section can be used to implement an evaluation system with regard to data storage and retrieval, but do not assist the developer of an evaluation system with any interface functionality. Making the FormBuilder DLL publicly available for download from the OES web site would allow developers to quickly and easily set up their own online evaluation systems by providing, in addition to OES data web services, an HTML evaluation form renderer. The FormBuilder DLL is
not yet available for release and would require documentation explaining its use in custom evaluation systems.

The last point of interest regarding the OES form builder is the HTML layout of evaluation forms. While HTML forms can be created with dynamically positioned elements, it was decided early in the development phase that using HTML tables would create a cleaner interface capable of dynamically rendering form controls. Specifically, as described in the OES file formats section, it is possible to make certain questions appear only when a parent question’s particular response option is chosen. HTML tables are useful for dynamically hiding and showing these “child” question rows. The following illustration shows the HTML layout of an OES evaluation form as built by the OES form builder.

![Figure 10: OES Evaluation Form HTML Layout](image)

The left side of the diagram above shows the structure of HTML tables required to generate the evaluation form shown on the right. Each box in the diagram above
represents an HTML table cell element. Notice that some table cells span two or more columns and that each question table (shown as Q1 and Q3 in the figure) ends with a blank row to separate it from the previous question table. A “question table” consists of a single question followed by its question rows, including any nested questions that may be contained by the question’s response options. Each question table is itself an HTML table containing rows and cells. Nested questions, like question two in the figure above, are also question tables. Thus the cell in the figure labeled “Nested Question 2” would in fact contain a question table similar to Q3. Child question tables such as question two can be shown or hidden dynamically by changing the visibility of the row in which they appear. In the example above, the nested question would only be visible when “Response Option 1B” was chosen as the response to question one. Selecting any other option for question one would cause the nested question to be hidden.

**OES Web Services**

One of the core goals of OES was the ability to implement web services. During the development phase, all database access was restricted to SQL query strings called from server-side code pages. Near the end of the development phase the many data functions used for user authentication and evaluation form data storage and retrieval were migrated to OES web services for one main reason: code reuse. Similar to the concept of the OES FormBuilder class, OES web services reduce the amount of code in the web application by allowing similar functions to be repeated with a call to a single web service. Additionally, OES web services allow external developers to create their own custom
online evaluation systems without worrying about data access. OES web services completely handle data storage and retrieval for evaluation system-based operations (creating and submitting evaluation forms, retrieving results, and so on). Thus a developer can easily create their own evaluation system and use OES as an evaluation system host, meaning that OES would store all data for the custom developer’s evaluation system. Providing OES web services on the Internet will make it possible for anyone to create their own online evaluation system without having to perform all of the database work that has already been implemented in OES. The four OES web services are based on OES user roles.

The OES account administrator web service performs functions that would typically be available to an OES account administrator on their action page. The web service implements database functions for registering a new OES account, modifying an existing account, and deleting an account. It implements a custom data type called “OESRegistrationInfo” that can be used by a web service client for storing and retrieving registration information. The account administrator web service also contains functions for authenticating users of the account, determining whether a user is a member of a specific role, and returning information for the forms with which a user may perform some action (such as an evaluator’s list of pending evaluation forms).

The OES form administrator web service manipulates data to perform the functions of a form administrator. These functions include the creation, viewing, modification, and deletion of evaluation forms. The form creation data functions are unique in that they
must parse the user’s uploaded form authentication and form data Excel files (see the file formats section above) to distribute data to the appropriate account database tables. After authentication and form display data are stored, the uploaded Excel files are deleted from the server. With regard to viewing evaluation forms, it is the developer’s responsibility to generate the HTML to render an evaluation form unless the OES FormBuilder DLL is used; the form viewing functions of the form administrator web service only return the data needed to view a specific evaluation form.

The last two web services are the OES evaluator web service and the OES reviewer web service. The evaluator web service performs functions for storing evaluation form submissions and drafts and for retrieving evaluation form drafts. The reviewer web service currently only performs functions for reviewing individual evaluation forms, specifically getting a list of form submission IDs and storing the date when a form was reviewed by a reviewer. Pending further enhancements, the reviewer service will also perform data functions for computing overall statistical results to be viewed by a reviewer. Note that there is no evaluatee web service because evaluatee web pages can use a subset of the form administrator and reviewer web service functions.
OES is an online evaluation system, developed as an ASP.NET web application in C# server-side code, HTML, and JavaScript for use on the Internet. OES consists of five user roles: account administrator, form administrator, evaluator, evaluatee, and reviewer. After an account administrator registers an account with OES, form administrators may login to create evaluation forms and, through their form authentication files, specify which evaluatees, evaluators, and reviewers may use their forms. Evaluatees may attach a subform to a form administrator’s evaluation form to obtain extra information from evaluators. They may also review submitted evaluations but may currently only view the results of their subforms. Evaluators may submit evaluation forms or save drafts of an evaluation form, which they may resume at any time before the end of the form’s evaluation period. Reviewers may view full evaluation results at any time during or after an evaluation form’s evaluation period. OES uses the .NET role-based forms authentication and URL authorization security model. It also uses custom Excel file formats for user authentication and form creation. OES web services make it possible for other developers to implement their own custom evaluation systems, using OES as an evaluation data host to avoid implementing their own database functionality. Additionally, the OES FormBuilder class may eventually be made publicly available in
the form of a DLL to prevent developers of custom evaluation systems from having to write their own dynamic HTML evaluation form renderers.

OES has accomplished many of its initial goals, and its design has evolved throughout the development process to accommodate new and changing requirements. One of the most difficult aspects of development was determining how to ensure confidentiality in a system that requires users to log in to use an account, a concept particularly important in university evaluations. OES accomplishes confidentiality through its account database structure (see Figure 6). The FormEvaluators table stores the date when an evaluator submitted their evaluation but does not in any way tie their user ID to their submitted data, which is stored in the SubmittedForms table. The “EvalID” field of the SubmittedForms table is an automatically generated value used only to keep an evaluator’s responses together for reviewing purposes but is not in any way tied to an evaluator’s login or user ID. Students can be assured that their responses on OES evaluation forms are truly anonymous, and the system allows only evaluators with valid logins to perform evaluations. Another challenge that arose late in the development phase was the comparison of OES with Modern Mind Software’s 2WAY product, which already had a graphical interface builder (part of the original design of OES). Yet it was believed that using a custom Excel file format, which is easily saved and modified on a user’s computer, would be more useful than 2WAY’s interface builder. Additionally, unlike 2WAY, OES does not require a client download. With the exception of using Microsoft Excel files for authentication and form data, OES is completely online. Any
user with Microsoft Office installed on their local machine can register an OES account or create evaluation forms, and anyone in general may submit or review OES evaluation forms online.

While OES has accomplished many of the goals for which it was designed and is indeed a usable system as it is, there is much work that could yet be done to improve OES. Most of the items listed as yet to be implemented in OES are considered enhancements but could be critical depending on the environment in which OES is used. These enhancements are prioritized into three categories: system-based enhancements, developer-based enhancements, and user-based enhancements.

The system-based enhancements listed here include issues with system setup and security. Currently, OES can only perform authentication via the use of Excel authentication files that are uploaded to the OES server. But it may be desirable to allow the system to connect to an external authentication data source, such as a university’s registration system, PeopleSoft, LDAP, or another SQL Server database on the Internet. It would be desirable to allow authentication, file uploads, and web service calls to be performed securely via SSL (Secure Sockets Layer); OES does not currently have security measures in place to safeguard the transmission of data over the Internet. Also, although it is not too common, there are occasions where a reference to a SQL Server database prevents it from being renamed or deleted, which occurs when an account administrator attempts to modify account information or delete their account. Database deletion and renaming could be made more reliable. Finally, it would be useful to
generate a random number or hash value for evaluation draft IDs. When an evaluator submits a draft of an evaluation form, they receive a draft ID number that is one greater than the previous draft number issued to another evaluator. Clever evaluators could possibly resume another evaluator’s draft, which would be less likely with a draft ID generated by random or created as a hash value.

Developer-based enhancements are those that would ease the maintenance of OES. For example, the OES error system could be much improved through the use of an “Errors” database table. OES currently stores error information in server session variables which are passed to the OES error page when program exceptions are thrown (when an unexpected error occurs). Sometimes these events occur at a point in the application where the session object is not yet available, in which case storing information in a database would be very helpful for debugging. Database values could then be used to display error information on the OES error page. Another useful developer-based enhancement would be the conversion of the functions in OES web services to use stored procedures. Stored procedures are functions stored in SQL Server that perform database operations and are capable of performing database logic not available to the web service functions. OES web services currently use strings of SQL commands, which would need to be converted to stored procedures, to perform database operations.

The last category, user-based enhancements, consists of enhancements that would improve an OES user’s experience. For example, users may want more form controls in
addition to the “select” and “textbox” control types currently provided by OES. One example, currently termed a “rate-set” control, would allow form administrators to easily specify a set of questions on their form that should all be rated on the same numeric scale and that would be displayed with radio buttons for the numeric values to the left or right of each question in the set. Additional properties, such as allowing the form administrator to specify a textbox control’s width (or number of columns) and height (or number of rows), could also be very useful. It may also be desirable to implement the concepts of form and account settings in OES. For example, a form administrator may wish to specify how many times each user may submit their form; currently only one form may be submitted per evaluator. An account administrator may wish to specify whether evaluatees are allowed to submit their own evaluation forms (which can be allowed by the form administrator making the evaluatee also an evaluator of the form), whether form administrators may also participate in evaluations, and whether form data submitted by evaluators should be deleted when a form administrator deletes a form.

Finally, it may be useful for the account administrator to be able to view usage statistics for their account, such as how many users are active in their account, how many evaluation forms have been created, how many evaluations have been submitted, and response rate data.

It should be clear that although OES is a working product, there is much that could yet be done to make it better. OES has been written to be easily extended and has the potential to evolve into a universal online evaluation system. As it is, OES can easily be
used for midterm and end-of-semester evaluations at universities. It could also be used for mid-course feedback (with an enhancement allowing evaluators to submit multiple evaluations), for anonymous surveys or quizzes to grasp overall student understanding of coursework, for company supervisor/employee evaluations, or even just for informal polls such as a group of friends voting on which movie to see on a Friday night. The benefit of OES being an online system is that it is accessible to anyone willing to setup and perform an evaluation, and developers can use OES web services to create their own evaluation systems to meet their custom needs. As easy to use and customizable as it is, perhaps OES will someday be the standard by which all other systems are evaluated.
REFERENCES


4) http://home.ust.hk/~eteval/cosset/qtIconf.pdf

5) http://home.ust.hk/~eteval/cosset/ncitt98.pdf


7) http://www.drexel.edu/provost/ir/conf/webeval.pdf

8) http://www.able-consulting.com/dotnet/adonet/Data_Providers.htm

9) http://www.kamath.com/tutorials/tut007_identity.asp

10) http://www.4guysfromrolla.com/webtech/091201-1.shtml

11) http://www.action-links.com/clip/

12) http://www.developerfusion.com/show/4410/4/

13) http://www.c-sharpcorner.com/asp/Articles/CachingInASPDPPL.asp


15) http://weblogs.asp.net/pwilson/archive/2004/05/11/129844.aspx


19) http://samples.dotnet.com/quickstart/aspplus/samples/webforms/ctrlref/webctrl/da
tagrid/doc_datagrid.aspx
4F33-A062-D165078E32B1&displaylang=en
us/vbcon/html/vbtskaddingcontrolstowebformspageprogrammatically.asp
=4&dir=next
us/dnnetsec/html/SecNetHT04.asp
5836&exp=0&select=949902
28) http://www.w3schools.com/sql/sql_update.asp
fileupload.asp
31) http://www.codeproject.com/vb/net/wsfileserver.asp