Word of Thanks

Of course we could not have done this project all by ourselves. First of all we want to thank our mentor Torben for his valuable feedback, his guidance and the hard disks. Also Brecht was a great help, he provided us with a profound insight in the problem domain and in the existing application. Marie Arneberth helped us translating the application to Swedish. Thanks Marie. Walter Schurmans, the coordinator for international student affairs at Hogeschool Gent, for maintaining the relations with Halmstad University, without him we wouldn't be here at all.

Contents

Introduction

The Team

We are Mel Gerats & Joren Six. We are in our last year Bachelor of Computer Science at Hogeschool Gent. We both chose the programming option. We have continued the work that our colleagues, Brecht Desmijter & Bastiaan Lambrey, started here. They also studied at Hogeschool Gent. Brecht was still in Sweden during our project and assisted us during our project.

Students

Name: Melchior Gerats
Address: Kortrijksesteenweg 37, 9000 Gent, Belgium
E-mail: mel.gerats@pandora.be

Name: Joren Six
Address: Bollestraat 68, 8820 Torhout, Belgium
E-mail: joren.ovning@0110.be

Mentor

Title: Associate Professor, Business Information Systems
Name: Torben Svane
E-mail: torben.svane@ide.hh.se

The Place

Halmstad, Sweden. We had the opportunity to study abroad, discover another country and other cultures so we did. We really did get in touch with a great number of different cultures: in our corridor alone there are people from Poland, Nigeria, USA, the Netherlands, Lithuania, South-Africa, Austria, Finland and Spain.

We could have gone to almost any western European country, but we chose Sweden for its great IT infrastructure, appealing way of live and beautiful language. Ok, and the blondes had something to do with it also.

We do our internship at Högskolan i Halmstad or Halmstad University. This is how they present themselves on their website, http://hh.se:

> ***Halmstad University is a university which crosses boundaries - for innovation and creativity! The university offers a wide choice of study programmes to approximately 7,000 students. Most of the study programmes leads to either a Bachelor's or Master's degree. The university has well-developed research environments, most with a unique national or international profile.***

Company

U-Quest is an international company that mainly works virtually, so they have small offices all over the world and no real head-office. The director of the company is also our mentor: Mr. Torben Svane, he also works as a professor at Halmstad University. The main working place of U-Quest is located in Halmstad. The other cities U-Quest is represented in are Birmingham (U.K.), Mobile (AL, U.S.A.), Brisbane (Australia), Auckland (N.Z.) and Taichung (Taiwan). U-Quest is specialized in web based survey systems and have a lot of experience in that field. They are used to working on the Microsoft Windows platform using ASP 3.0 and want to know the advantages of ASP.NET more specified: ASP.NET 2.0 using VB.NET 2.0.

Assignment

Our task was to develop a war game supporting system. War games are large scale disaster simulations.

Take for example a terrorist attack on a nuclear power plant. A certain scenario is planned: Terrorists take hostage a couple of employees and threaten to blow up the plant. That situation has to be dealt with by every stakeholder: police, SWAT teams, the media make fake news bulletins, the employees of the plant, the management of the plant…

During this simulation the stakeholders fill out surveys to give feedback. With the data from these surveys the management can see how well (or bad) everything goes. Those surveys are held on the secure website we developed. This data is used to generate reports.

The assignment consists mainly of two parts:

1. Optimize and refactor the existing ASP.NET 1.1 application: We upgraded the application to the ASP.NET 2.0 platform with an access database system.
2. Add new functionality: report generation from the survey data to different file types (doc, rtf, xls, ppt, pdf….).

Övning.nu

Övning.nu is a web based war game support system. The website allows organizations to set up and manage surveys and documentation to get a better overview of the war games.

The website is fully internationalized. When a user visits for the first time the local browser language is used, if supported. If the language is not supported English is used as the default. Once logged in, a user can select his preferred language in case he wants to change it.

Throughout the website we made as much as possible use of intuitive icons instead to help the user find his way.

The website consists of three separate parts. The administrative part for the Super User (or website administrator), the part for the War Game Organizer and a part for the normal user.

Website administrator

The website administrator can view a list of all registered users, view and delete hacklog events and manage the help messages for each page of the website.

Hacklog entries are created when something happens that ideally shouldn't happen. When a user tries to log in with a wrong username or password, when a user is temporarily banned because he tried to log on 5 consecutive times with the wrong password or when a user requests his login data through password recovery.

Additionally, the website administrator can create and update help messages for each page. The help messages provide additional help information about pages and can be viewed from each page.

The help message management interface provides the ability to fully customize the help, in different languages.

War game organizer

The part for the war game organizer is by far the biggest part of the website.

There are three mayor parts: User management, Question management and War game management.

Under user management the war game organizer can add users to the organisation. A username and e-mail address can be entered and a rank can be specified. An e-mail containing the username, a generated easy to remember but hard to guess password and the user's role will be sent to the e-mail address to notify the user.

Under question management the war game organizer can create questions. Each organization has its own pool of questions to use in surveys. Questions can be used in different surveys so the same questions do not have to be created every time.

A question consist of a title, a category, question text, an optional description and optional comments and, in the case of multiple choice or multiple select questions a list of answers. The number of answers to choose from is fully customizable.

Once created, questions can be changed and updated, or deleted. If a question is already part of a survey, a warning will be given. If the question is deleted anyway it will also be removed from the existing surveys.

Under war game management war games can be created. Once created, a war game gets its own node in the menu so it is easily selectable and clearly visible. Then you can: manage surveys, participants, documents and reports.

To create a survey you have to give it a name, specify a start date and end date, select a type, and specify which roles will be able to take the survey.

The type of survey can be “fill out once” or “edit allowed”. If editing is allowed users can edit their answers after filling out the survey for the first time. If not, they can only review their answers without editing them.

Previously created questions can also be added to the survey here. You can choose questions from the organizations question pool and link them to the survey. The questions are only shown by name but more information is available by double clicking on them. A popup with the question, as the user will see it in the survey, will be shown. Even when the organisation has a large database of questions it will still be easy to choose the right ones. Adding questions to the survey or removing them is as simple as clicking a button.

You also can see a preview of the survey here. All questions that are part of the survey will be shown in the order the user will see them, allowing you to get a great overview of the survey.

Under participant management you participants can be added and removed to and from the war game. The process is very similar to adding questions to a survey. Participants are chosen from a list of all users of the organisation. Like with questions, additional information can be viewed by double clicking the names.

You can also manage the participant’s roles. Within a war game, each participant is assigned a role. Roles can be for example mentor, counter player, internal participant or manager. They show what the role of the participant will be in the war game. It is possible to create different surveys based on the participant’s role, so information can be kept separate. The default role is internal participant, as that is the most likely one.

Under document management the war game organizer can write or upload documents that give more information about the war game. Each document has a category and a title, and can either we written in a text area or uploaded if it is already written.

Once created, they can be viewed (if uploaded), updated and deleted.

Under reports you can view and export the results of the surveys.

Charts provide a visual representation of the answers. For each multiple choice or multiple select question a chart will be generated that shows the number of participants that gave each answer. Answers on open questions can not really be represented in a chart, as each answer can, and probably will be different. However, a chart will be shown that shows the percentage of participants that answered the question and the number that didn’t.

It is also possible to view the answers per participant and answers per question. This allows you to get a detailed overview of the answers, and also provides an overview of the answers to open questions, questions that can not be represented in a chart.

Finally, reports can be exported to various file formats:

- doc, with charts
- rtf, with charts(in a zip file to keep the file size reasonable)
- xml
- html
- xls (Microsoft Excel)
- pdf

Normal user

Normal users basically have to do one thing: fill out surveys.

The user is given a list of surveys that are applicable to him, and he can fill them out. Depending on the type of the survey and whether or not the user already has answered the survey they will be represented with a different icon.

Once the survey is started it is just a matter of answering the question and clicking 'next' until all questions are answered.

All users

Part of the web site is available for all users.

The first time someone visits the website he is greeted with the splash page. The philosophy behind this is that people are more likely to try to get (unauthorized) access to a site if the first thing they see is a login field.

After logging in, all users have access to a profile page where they can change their e-mail, password, preferred language and printer settings.

Each page has a help page available should users need more information.

Requirements

Use cases

There are three main use cases. Two for the war game organizer ('register' and 'organize war game' and one for the participants of a war game ('fill out survey'). The use case “Organize a war game” is divided in several sub use cases.

Register

Primary actor: War game organizer

Pre condition: None.

Post condition: The organisation is registered, the war game organizer has an account, an invoice has been sent

Main success scenario:

1. The user enters the organisation details
2. The system validates the data
3. The user enters the war game organizers details
4. The user enters a Captcha code
5. The system validates the data
6. The system creates the war game organizer account
7. The system creates the organisation database
8. The system send the account details to the war game organizer by e-mail
9. The system sends an invoice to the war game organizer by e-mail

Alternative flows:

2a. the system notifies the user that the organisation data is invalid

1. The system shows the user which fields are invalid and clarifies how to fix the errors.
2. The user corrects the input.
3. The process continues at step 2.

5a. the system notifies the user that the organisation data is invalid

1. The system shows the user which fields are invalid and clarifies how to fix the errors.
2. The user corrects the input.
3. The process continues at step 5.

Organize war game

Primary actor: war game organizer

Stakeholder: war game participant

Pre condition: the user is authenticated and has the rights to create a war game.

Post condition: a war game is organized. It has one or more surveys, the surveys have at least one question and the war game has one or more participants.

Main success scenario:

1. The user creates a war game (see “Create war game” use case)
2. The user creates a survey (see “Create survey” use case)
3. The user adds one or more questions to a survey (see “Add questions to survey”)
4. The user adds one or more participants to a war game (see “Add participants to war game”)

Alternative flows:

2a. multiple surveys can be added by repeating step 2.

Create war game

Primary actor: War game organizer

Pre condition: the user is authenticated and has the rights to create a war game.

Post condition: a war game is created.

Main success scenario:

1. The user enters the war game details (start date, name,…) into the system
2. The user gives the command to create the war game.
3. The system creates the war game
4. The system notifies the user that the war game was successfully created.

Alternative flows:

*. The system fails, shows the error message and tries to recover its previous state.

3a. the system notifies the user that the war game details are invalid.

4. The system shows the user which fields are invalid and clarifies how to fix the errors.
5. The user corrects her or his input.
6. The process continues at step 2.

Create survey

Primary actor: war game organizer

Pre condition: the user is authenticated and has the rights to create a survey. At least one war game exists.

Post condition: a survey is created.

Main success scenario:

1. The user enters the survey details (start date, end date, name,…) into the system
2. The user gives the command to create the survey.
3. The system creates the survey
4. The system notifies the user that the survey was successfully created.

Alternative flows:

*. The system fails, shows the error message and tries to recover its previous state.

3a. the system notifies the user that the survey details are invalid.

1. The system shows the user which fields are invalid and clarifies how to fix the errors.
2. The user corrects the input.
3. The process continues at step 2.

Delete survey

Primary actor: war game organizer

Pre condition: the user is authenticated and has the rights to delete surveys. At least one survey exists.

Post condition: one the survey is deleted.

Main success scenario:

1. The system shows the available surveys.
2. The user selects the survey to delete.
3. The user gives the command to delete the survey.
4. The system deletes the survey.
5. The system notifies the user that the survey was successfully deleted.

Alternative flows:

*. The system fails, shows the error message and tries to recover its previous state.

4a. One or more participants have already taken this survey.

1. The system notifies the user of this and asks for confirmation.
2. The user confirms the deletion.

   2a. the user cancels the action.

   2b. the system cancels the action and the process restarts.
3. The system deletes the answers.

4. The process continues at step 4.

Add documents to a war game

Primary actor: war game organizer

Pre condition: the user is authenticated and has the rights to add documents. At least one war game exists.

Post condition: one or more documents are added.

Main success scenario:

1. The user enters the document details (name, category…) and text.
2. The user gives the command to add the document to the war game.
3. The system creates the document
4. The system notifies the user that the document was successfully created.

Alternative flows:

*. The system fails, shows the error message and tries to recover its previous state.

1a. the user selects an existing document to upload.

1. The process continues at step 2.

3a. the document details are invalid.

1. The system shows the user which fields are invalid and clarifies how to fix the errors.
2. The user corrects the input.
3. The process continues at step 2.

Create Question

Primary actor: war game organizer

Pre condition: the user is authenticated and has the rights to create questions.

Post condition: a question is created.

Main success scenario:

1. The user enters the question details (title, setting, question, comments, category…).
2. The user selects the type of question.
3. The user enters the possible answers.
4. The user gives the command to create the question.
5. The system creates the question.
6. The system notifies the user that the question was successfully created.

Alternative flows:

*. The system fails, shows the error message and tries to recover its previous state.

2a. the user selects “Open question”.

    1. The process continues at step 4.

5a. the question details are invalid.

    1. The system shows the user which fields are invalid and clarifies how to fix the errors.
    2. The user corrects the input.
    3. The process continues at step 2.

Delete Question

Primary actor: war game organizer

Pre condition: the user is authenticated and has the rights to add questions to a war game. At least one question exists.

Post condition: one or more questions are deleted.

Main success scenario:

1. The system shows the available questions.
2. The user selects the questions to delete.
3. The user gives the command to add the questions.
4. The system deletes the questions.
5. The system notifies the user that the questions were successfully deleted.

Alternative flows:

*. The system fails, shows the error message and tries to recover its previous state.

4a. the questions is already used in a survey.

1. The system notifies the user of this and asks for confirmation.
2. The user confirms the action.

   2a. the user cancels the action.

   2b. the system cancels the action and the process restarts.
3. The system removes the questions from the survey.
4. The process continues at step 4.

Add questions to war game

Primary actor: war game organizer

Pre condition: the user is authenticated and has the rights to add questions to survey. At least one survey exists. At least one question exists.

Post condition: one or more questions are added to a war game.

Main success scenario:

1. The system shows the available questions.
2. The user selects the questions to add to the survey.
3. The user gives the command to add the questions to the survey.
4. The system adds the questions to the survey.
5. The system notifies the user that the questions were successfully added to the survey.

Alternative flows:

*. The system fails, shows the error message and tries to recover its previous state.

3a. the questions could not be added to the survey.

1. The system notifies the user of the failure.
2. The process continues at step 1.

Add user to organization

Primary actor: war game organizer

Pre condition: the user is authenticated and has the rights to add users to an organization.

Post condition: a user is added to the organization.

Main success scenario:

1. The user enters the new user's details (name, e-mail…).
2. The user selects the new user's role.
3. The user gives the command to create the new user.
4. The system creates the new user.

Alternative flows:

*. The system fails, shows the error message and tries to recover its previous state.

3a. the new user's details are invalid.

1. The system shows the user which fields are invalid and clarifies how to fix the errors.
2. The user corrects the input.
3. The process continues at step 3.

Add participants to war game

Primary actor: war game organizer

Pre condition: the user is authenticated and has the rights to add participants to a war game. At least one war game exists. At least one participant exists.

Post condition: one or more participants are added to a war game.

Main success scenario:

1. The system shows the available participants
2. The user selects the participants to add to the war game.
3. The user gives the command to add the participants to the war game.
4. The system adds the participants to the war game.
5. The system notifies the user that the participants were successfully added to the war game.

Alternative flows:

*. The system fails, shows the error message and tries to recover its previous state.

3a. the participants could not be added to the war game.

1. The system notifies the user of the failure.
2. The process continues at step 1.

Take Survey

Primary actor: war game participant

Pre condition: the user is authenticated and has the rights to take a survey. The user participates in at least one war game, which has at least one survey.

Post condition: The user has taken the survey, the answers are saved.

Main success scenario:

1. The system shows the available surveys
2. The user selects the survey to take.
3. The system shows a question.
4. The user enters an answer.
5. The system notifies the user that the participants were successfully added to the war game.

Alternative flows:

*. The system fails, shows the error message and tries to recover its previous state.

3a. the participants could not be added to the war game.

1. The system notifies the user of the failure.
2. The process continues at step 1.

Legacy System

There were a few constraints on what we could and could not do. First of all the databases were already designed. We were allowed to add fields and add integrity rules but we could not rename or remove fields from the databases. We did remove a very small number of fields but those changes had to be approved explicitly. This is because there are a number of legacy applications that perform read operations on the databases. The naming guidelines for the database fields were also fixed; these had to be followed when adding new fields to the database. For more information on which changes we made please consult appendix C.

Another requirement was that we had to use not one but multiple databases. Each organization has its own Access database that can be downloaded off the website easily. This also adds an extra layer of security. We strongly discouraged the use of an Access database on the server side. Even Microsoft self discourages the use of access for 24/7 applications because Access is not designed for multi-threaded applications or heavy use. You can read more about why using Access is not a good idea in appendix B.

The system that was developed last year was build for the ASP 1.1 platform. The idea was that we had to port it to the ASP 2.0 platform and add extra functionality. This sounds good in theory but in practice we almost rewrote everything, mainly because there are a lot of changes involved in the transition of .NET 1.1 to .NET 2.0. First of all the language used has changed a lot. VB.NET 2.0 has a lot of features that were not present in the 1.1 version. These changes include operator overloading, unsigned types, partial classes, default instances and last but not least generics. The underlying framework also changed a lot. The .NET framework has a complete new collection library, using generics and it includes many additional and improved ASP.NET web controls. Because we used all these new features we were able to recreate the project of last year and add the requested functionality within the proposed timeframe.

Design

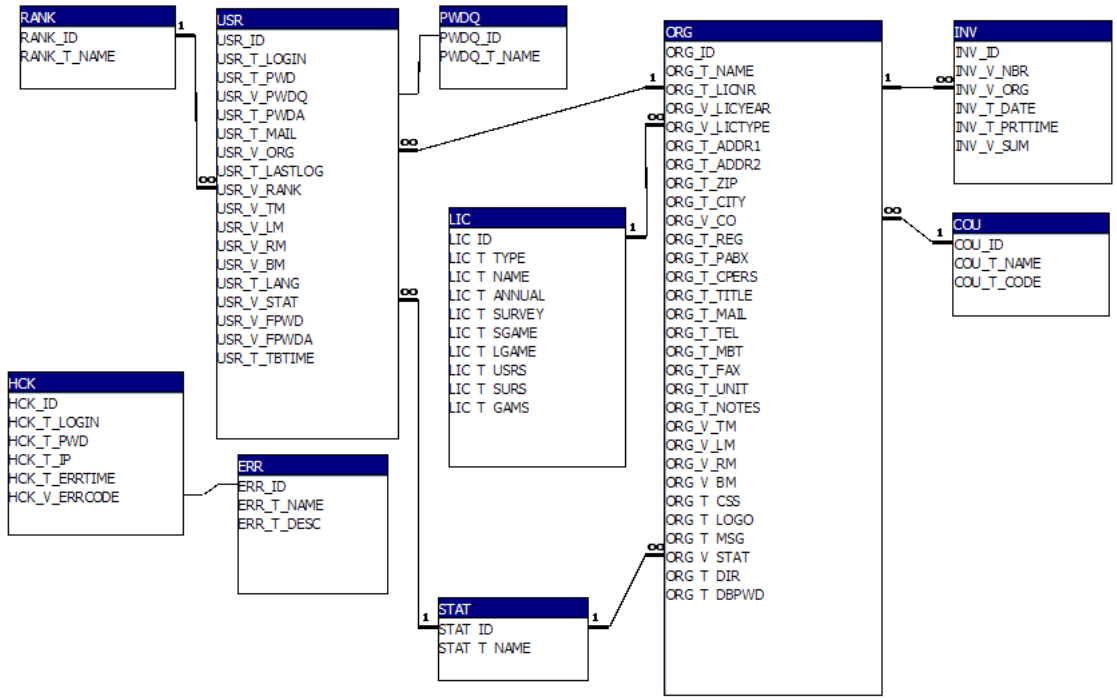
In this chapter we are going to highlight the most interesting parts of the design of our application. This is not a complete overview of all the techniques used in our applications but the most important concepts and aspects are covered. Please browse the code and the code documentation, included on the CD-ROM, if you want a more detailed insight.

Database Diagram

We have two types of databases in our system: one for the system data and one for each organization.

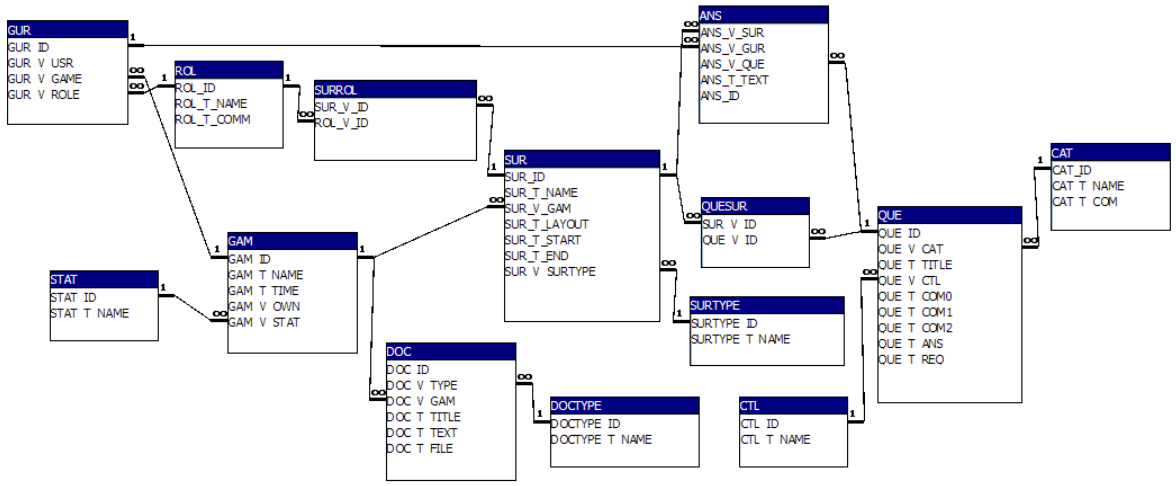
System database

The system database is used to store all the data used for system administration (HCK, ERR), user authorization (USR, STAT) and data about organizations (ORG, INV, COU). The data of the organization itself is stored in another database. For more information please consult Appendix B. Every table and field is documented and the relations are explained in that appendix.



Organization database

Each organization has its own database with only their data. The data for an organization includes their war games (GAM), surveys (SUR), questions (QUE), answers (ANS), war game participants (GUR) and their roles (ROL) and related data. For more information please consult Appendix B. Every table and field is documented and the relations are explained in that appendix.



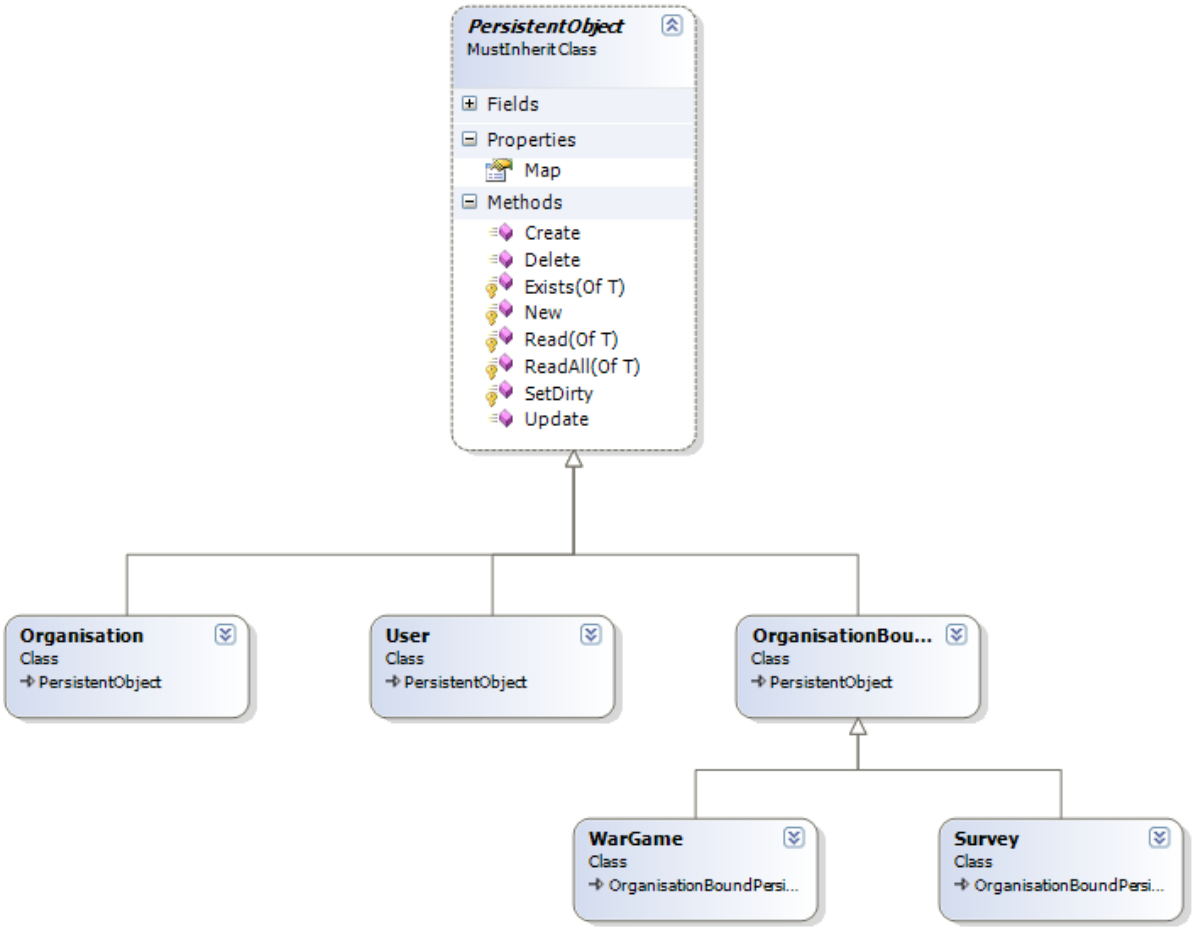
Class Diagrams

Data Layer

The data layer is one of the most interesting parts of our application; it leverages reflection, generics and attributes extensively to perform its task. It had to be very resource friendly and it had to perform well under heavy load. One of the main challenges was that some data had to be stored in the system database and other data had to be saved in an organization specific database. To handle this gracefully and consistently we had to create our own object to relational mapping (ORM) system because there is not one ORM system that supports access and multiple databases.

PersistentObject

Because all of our data objects had to have the same behavior we used inheritance to bundle the common behavior in one class. The `PersitentObject` class is an abstract class that is implemented by all the data objects that are stored in the system database e.g. users. For the objects that are stored in the database of an organization – e.g. a survey or a war game - an extra layer is added. They inherit form the abstract class `OrganizationBoundPersistentObject` which in return inherits from `PersistentObject`. In UML this looks thusly:



In `PersistentObject` the `Create()`, `Update()` and `Delete()` methods are public and have a default implementation:

```
Public Overridable Sub Update()
    Dim queryBuilder As New SimpleQueryBuilder
```

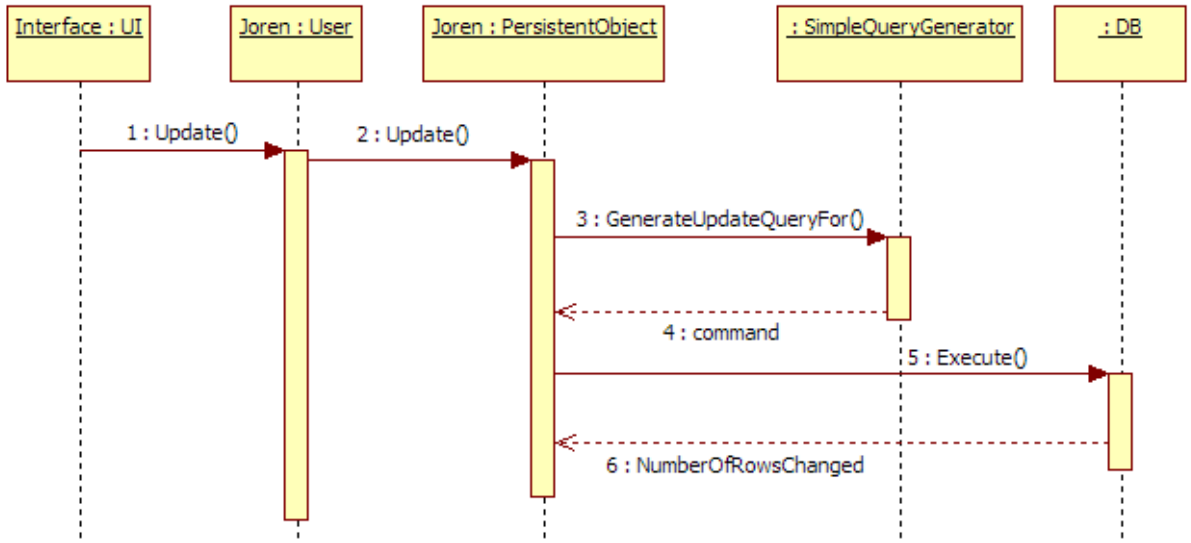
```
        Dim command As IDbCommand = _
          queryBuilder.GenerateUpdateCommandFor(Me)
        DB.SystemDatabase.Execute(command)
    End Sub
```

This default implementation is sufficient for almost all objects and if it is not, the methods can be overridden. `PersistentObject` has a `Map` in which the dirty properties are saved, a property is dirty when it is changed and that change has not been saved in the database. The dirty properties are set with the `SetDirty(string property)` method. The `SimpleQueryBuilder` uses the map to build an `IDbCommand`. This command represents a parameterized query (update, delete, select, …). The `SimpleQueryBuilder` inspects the map to know which fields have to be updated.

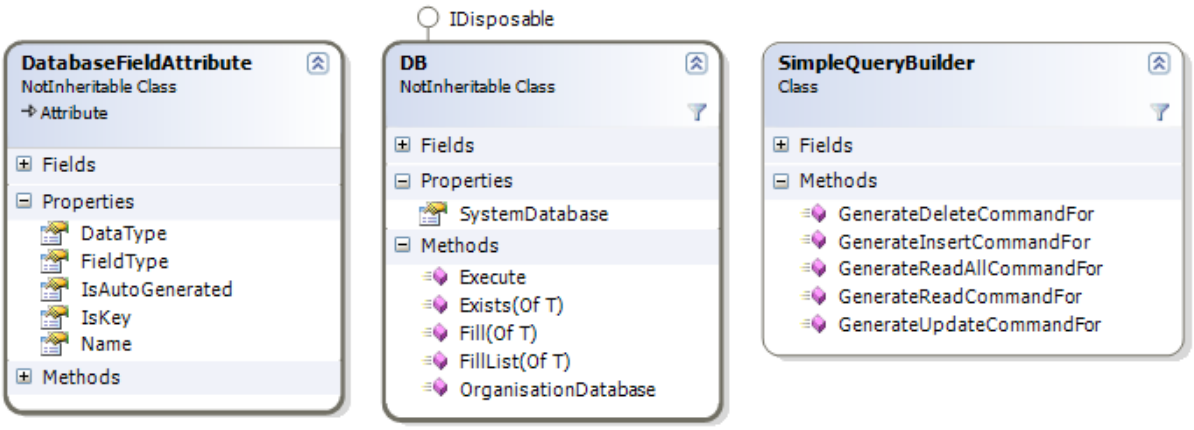
As previously stated the `PersistentObject` class saves objects in the system database To save objects in the organization database the class `OrganizationBoundPersistentObject` is used. This class works identical to `PersistentObject` only the database in which the changes are saved differs.

Database action

This is how a generic database action is executed:



The user interface sends a message - update - to a domain object `User`, the message is automatically delegated to its super class: `PersistentObject`. `PersistentObject` asks the `SimpleQueryBuilder` to generate an update query and when it receives a command it is executed by the `DB`. The `DB` returns the number of rows changed, so the domain object can check if everything went as expected. These are, next to `PersistentObject`, the main classes of the data layer:



`DB` is used as an interface to the database it executes commands and fetches objects. It also knows all the databases: the system database and the organization databases.

`DatabaseFieldAttribute` is an attribute that can be used by classes and properties. It is meta-data for a data object. This may sound strange but an example will make everything clear:

```
<DatabaseFieldAttribute("USR")> _
Public Class User
    Inherits PersistentObject

   <DatabaseFieldAttribute("USR_T_LOGIN", DbType.String)> _
    Public Property Username() As String
        …
    End Property

    <DatabaseFieldAttribute("USR_ID", DbType.Int32, _
         DatabaseFieldType.DatabaseGeneratedKey)> _
    Public Property Identifier() As Integer
        …
    End Property

    <DatabaseFieldAttribute("USR_T_LASTLOG", DbType.DateTime)> _
    Public Property LastLogin() As DateTime
        …
    End Property
End Class
```

This is a small part of the `User` class, major parts have been omitted to improve readability. This code demonstrates the use of attributes. Every property that has to be saved in the database has a `DatabaseFieldAttribute`. The attribute defines how, the type, and where, the column name, the objects data is saved. The class itself is also decorated with an attribute to define the table name for the object. The `SimpleQueryBuilder` reflects these attributes and generates parameterized queries with the gathered data.

This approach has major advantages:

> It is easy to add fields to the database and the objects without any effort. The system can grow iteratively without problems.
>
> The RDBMS can be changed very easily all database dependent code is concentrated in one class (`DB`) . The rest of the classes are using an abstraction.
>
> It is invulnerable to SQL-injection because everything is parameterized automatically.

There is no SQL to debug. The developer can focus on creating new functionality instead of tediously debugging and adapting SQL.

It is type safe: the database returns a perfectly type safe object. All the casts, from `String` to `Integer`, from String to `DateTime`, … are performed automatically. This prevents errors since the developer is freed from the responsibility to write and check casts.

By using query-by-example it is easy to request any information from the database, the following code snippet searches for the user “Jan” and deletes his data from the database. The same approach can be used when searching for multiple properties.

```
Dim jan as new User
jan.Name = “Jan”
Dim queryBuilder As New SimpleQueryBuilder
Dim command As IDbCommand
command = queryBuilder.GenerateReadCommandFor(jan)
jan = DB.SystemDatabase.Fill(Of User)(command)
jan.Delete()
```

There are also some drawbacks:

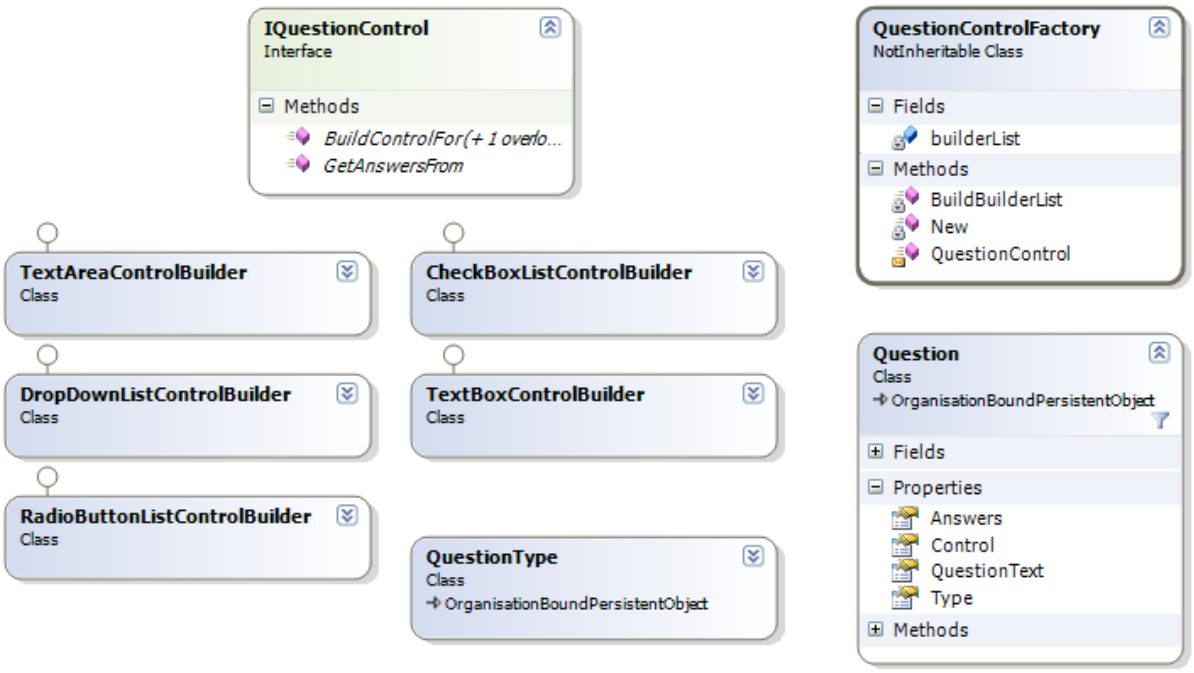
Every time an object is requested a class has to be reflected. This is not as fast as a direct SQL-query. This proved to be an acceptable, almost negligible overhead.

Only simple queries, queries with one table, are built automatically. We did not implement a way to model relations between tables. The queries with joins are done manually. The results of those queries are converted into type safe objects by the data layer. The number of simple queries is much larger than the queries with multiple tables.

Question control

Another interesting part of our application is how the different questions are represented in their own controls. When creating a survey a number of questions are to be added to the new survey. A question has a type: multiple-select, multiple-choice or open question. When a user wants to fill out a survey these questions have to be presented in a familiar form. The representation of the question depends on the type of the question. Every type has another control. This is done using the builder pattern. The common behavior is defined in an interface: `IQuestionControl`. For each type an implementation of that interface is created. E.g. a multiple-select is rendered using a `CheckBoxListControlBuilder`. The question type and the control are linked in `QuestionControlFactory`. `IQuestionControl` has to be able to do these three things:

1. Render an empty control representing a question: `BuildControlFor(Question)`
2. Render a control with the answers provided by the user: `BuildControlFor(Question,Answers)`
3. Get the answers out of the control: `GetAnswersFrom(Control)`



By using this approach it is easy to change the controls rendered or even add new question types and controls. Because the user interface knows nothing about these controls they can be changed without any changes in the user interface. The interface just renders a control: `Question.Control` without knowing its type or any other details.

Template System

In this chapter we are going to explain which technologies and practices we used to make sure that the layout is completely adaptable. It could even be possible to create a layout that runs on smart phones by changing only a couple of files. We made our pages as accessible as possible and followed the W3C standards in the html we wrote. The html ASP.NET generates is sometimes of questionable quality, but there is nothing that can be done about that.

Master Pages

Master pages, a new feature of ASP.NET 2.0, enable you to apply the same page layout to multiple content pages in a Web application. Master pages provide you with an easy method of creating a consistent look and feel for a web site. Because the pages in most web applications have standard elements, such as logos, navigation menus and copyright notices, you can place all those elements within a single Master Page. If you base the content pages in your application on this master page, then all of the content pages will automatically include the same standard elements. In our application we used three master pages: one with the menu, header and footer; a master page for popups and a master page for the splash page. In the image above you can see the concept. The menu, header and footer are provided by the master page, the contents by the content page.

Themes

ASP.NET 2.0 includes support for themes. The idea is that all the images, CSS files, skin files, and all the other files that define a theme end up in a special directory with the name of that theme. If you create a number of those themes the user can switch to her favorite one. We have included two themes. Only one of them is complete, the other one is an incomplete proof of concept.

The concept of themes has a number of advantages, even when only one is created. It is a good thing to define the look and feel of a website in a central place. This look and feel can be replaced or new themes can be added. It is possible to convert our application to a web application for smart phones only by adding a new theme directory.

A theme also defines how ASP.NET controls are rendered. This is done using skin files. With those skin files it is possible to circumvent the annoying fact that Internet Explorer does not support CSS attribute selectors. The following code is the complete skin file for our application it assigns a CSS class to a number of controls:

```
<asp:Button runat="server" CssClass="button" />
<asp:RadioButtonList runat="server" CssClass="flatlabel" />
<asp:CheckBoxList runat="server" CssClass="flatlabel checkboxlist" />
<asp:TextBox runat="server" CssClass="textbox"/>
```

```
<asp:Wizard runat="server">
    <CancelButtonStyle CssClass="button" />
    <FinishCompleteButtonStyle CssClass="button" />
    <NavigationButtonStyle CssClass="button" />
</asp:Wizard>
```

CSS and standards

Using CSS results in reusable html and separates style from content. The entire layout in our web application is done using CSS. These are the most important advantages of CSS:

1. It is easier to make site-wide changes. One only has to change one CSS file rather than all pages
2. Smaller files result in a faster download, and the CSS file is cached by the client
3. It is easier to code html, there is less code on the page
4. It allows users to customize to their own needs if they provide their own CSS file, or use style switchers
5. Separating style from content makes life very easy for visitors who prefer to view only the content of a web page, or to modify the content. These could be blind or visually impaired people who might use a screen reader to interpret a page
6. More accessible to wider variety of mediums. Build one page, but write separate CSS files to suit particular mediums like hand-held devices, web TV, printers etc.

The CSS files follow the standards except for a few hacks for internet explorer. Since the CSS support in Internet Explorer is flaky, at best, we have included a couple of Internet Explorer only style rules. All our pages are valid XHTML 1.0 Transitional, or have minor errors due to the nature of ASP.NET, we can not control the HTML it generates. The master page is standards compliant, only the content pages can have some incorrect, generated, XHTML.

Internationalization

One of the requirements for our project was that it had to be available in at least three languages: Swedish, English and Dutch. It was also required that adding new languages had to be easy. To provide this functionality we used the built-in ASP.NET 2.0 internationalization framework. Using this framework makes supporting multiple languages a lot easier. Not only the language of the user is taken in account, the culture is also handled. For example, people from Great-Britain use another notation system for their dates and times than someone from the United States, but they both use the same language: English. These subtle differences are handled automatically by the framework if the correct language and culture is set.

If a user visits our web application for the first time the language of the website is automatically set to the language of the user's browser. If the language is not supported the users sees the English version.

We used the ASP.NET 2.0 model to internationalize the website. This makes use of resource files (.RESX). RESX files are XML files that contain all the texts for each language. By naming the files in a specified way, they are automatically used for the right language. The files are named by taking the name of the page it is for, followed by the language code, followed by "*.resx*". The default language, English in our case, doesn't need a language code. The three resource files for the page Default.aspx are called Default.aspx.resx, Default.aspx.nl.resx and Default.aspx.sv.resx.

To support more languages all that has to be done is create more resource files with the right language code.

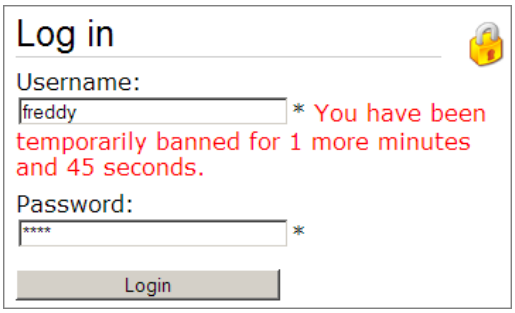
Role-based authentication

Because classified data is stored in the databases of our website and that data is shown on the pages, security is an important issue. To prevent unauthorized users from accessing this confidential data we used the built-in role based security model. There are three roles in our application:

1. Su: super user, the website administrator
2. Admin: the administrator for an organization, can add users to his organization, organize war games, view reports, …
3. Participant: a war game participant, can only log in and fill out surveys.

Every role has its own pages it can access. These restrictions are handled by securing directories using web.config files. The following code excludes everyone from accessing all files in a directory except the users with the “SU” role:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration
xmlns="http://schemas.microsoft.com/.NetConfiguration/v2.0">
<system.web>
  <!--Allows only users in the "SU" role-->
  <authorization>
    <allow roles="SU"/>
    <deny users="*"/>
  </authorization>
</system.web>
</configuration>
```

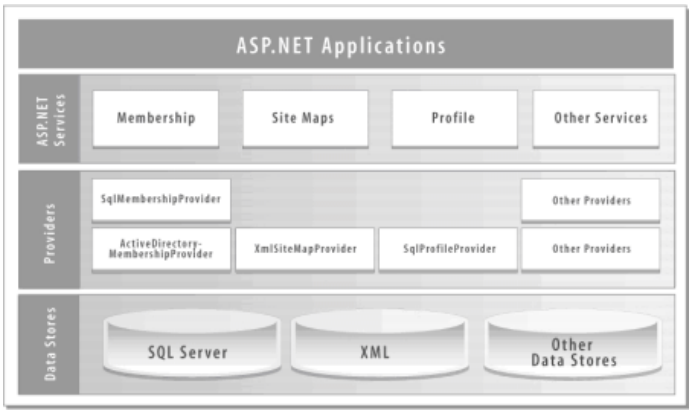


To prevent dictionary attacks on the login system we have provided a security mechanism that discourages these hacking attempts. We have implemented a hacklog and a temporary locking mechanism. When a user tries to login and provides a faulty password five times the user is banned for five minutes. This event is logged in the hack log together with the users IP address. When an alleged hacker tries to login with a user name that does not exist this is also logged. When a banned user tries to log in this is logged as wel. Also when a visitor uses the password recovery feature the system registeres this action. In other words the system administrator knows what the users are doing and when and if something goes wrong the IP address of the user is logged.

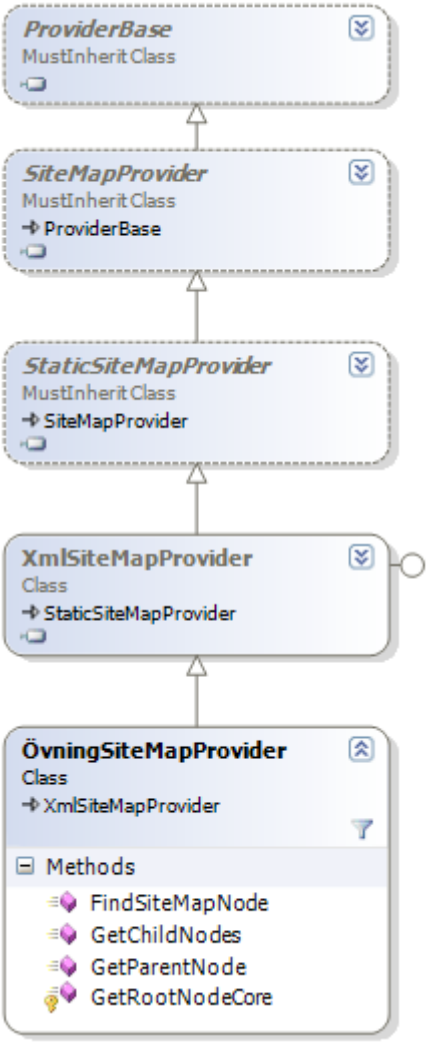
Another security feature is the sliding session. When a user logs in his session starts, he can do whatever he is allowed to do. If he forgets to logout another malicious user could use his computer and change his data. To prevent this from happening the user is logged out automatically after a certain time of being inactive. Every time he requests a page his session is extended for five minutes. Of course it is still advisable to log out manually on a public computer.

Menu and sitemap

ASP.NET 2.0 includes a number of services that store state in databases and other storage media. For the majority of applications, the built-in storage options are sufficient. However, the need sometimes arises to store state in other media such as PostgreSQL databases, DB2 databases, or even XML files.



In ASP.NET 2.0 the provider design pattern is used throughout to offer this flexibility. It is used for membership, profiles, sitemaps and many other services. The idea is that you have a common interface and a couple of implementations performing a specific task. For example you can save the membership data in active directory using the `ActiveDirectoryMembershipProvider` or in Microsoft's SQL Server using the `SqlMembershipProvider`. If you want to store the membership data in a PostgreSQL database you will have to write a `PostgreSqlMembershipProvider`, this is an implementation of the `BaseMembershipProvider`. Because existing ASP.NET components communicate with the provider interface you can use your own provider implementation, with custom functionality, and still be able to use the ASP.NET components.



The ASP.NET Sitemap service is used to populate menus and the breadcrumbs on top of the page. We used the provider design pattern in combination with the sitemap. We wrote an `ÖvningSiteMapProvider` to be able to generate dynamic menus.

The SiteMapProvider used in Övning.nu fetches all nodes out of an XML-file and then adds nodes per war game. These war games are read out of the database. All the functionality of `XmlSiteMapProvider` was needed so we sub classed it and, overrode the methods and added nodes when needed. The XML file has a structure like this one:

```
<siteMap>
  <siteMapNode title="Home" url="home.aspx">
    <siteMapNode title="Om" url="om.aspx">
    </siteMapNode>
  </siteMapNode>
</siteMap>
```

Methodology

Test Driven Development

We used test driven development for all the domain classes. When we had to implement a new class we implemented empty method and property stubs wrote a unit test for the new class. Only after that we wrote the implementation. When the unit test passes the class is finished. Visual Studio 2005has a built-in unit test framework, so the tests can be run from within the IDE. The unit tests made refactoring a much easier task. If all the unit tests pass after a serious refactoring you know you didn't break something. If you did change the external interface you would know, by checking the unit tests, where changes have to be made in the unit tests and in the production code, the website. All the tests are included in a stand alone class library and another namespace: `Övning.Domain.Test`. The tests are written in Visual C# 2.0.

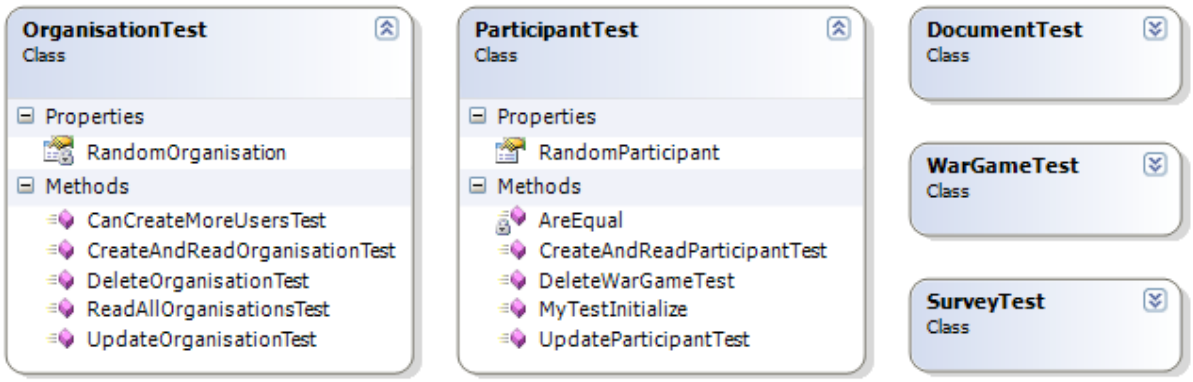
Example of a unit test

What follows is an example of a class that tests the Organisation domain class. The real class includes a number of other tests. Each method in the class tests a method in the organisation class. After the operation is done the results are checked and everything is cleaned up. E.g. deleting the newly created organisation.

```
[TestClass()]
public class OrganisationTest
{
    [TestMethod()]
    public void CreateAndReadOrganisationTest()
    {
        Organisation target = RandomOrganisation;
        target.Create();
        Organisation actual = Organisation.Read(target.LicenceNumber);
        Assert.AreEqual(target.Name, actual.Name);
        target.Delete();
    }
    …

    private Organisation RandomOrganisation
    {
        get
        {
            Organisation target = new Organisation();
            target.Name = Guid.NewGuid().ToString();
            target.Address1 = Guid.NewGuid().ToString();
            target.License = License.BronzeLicence;
            return target;
        }
    }
    …
}
```

We have created more than fifty different tests to assure the quality of the code. These tests are all atomic, they are organized according to their function. In the following class diagram you can see the rest of the methods in `OrganisationTest` and some other test classes.



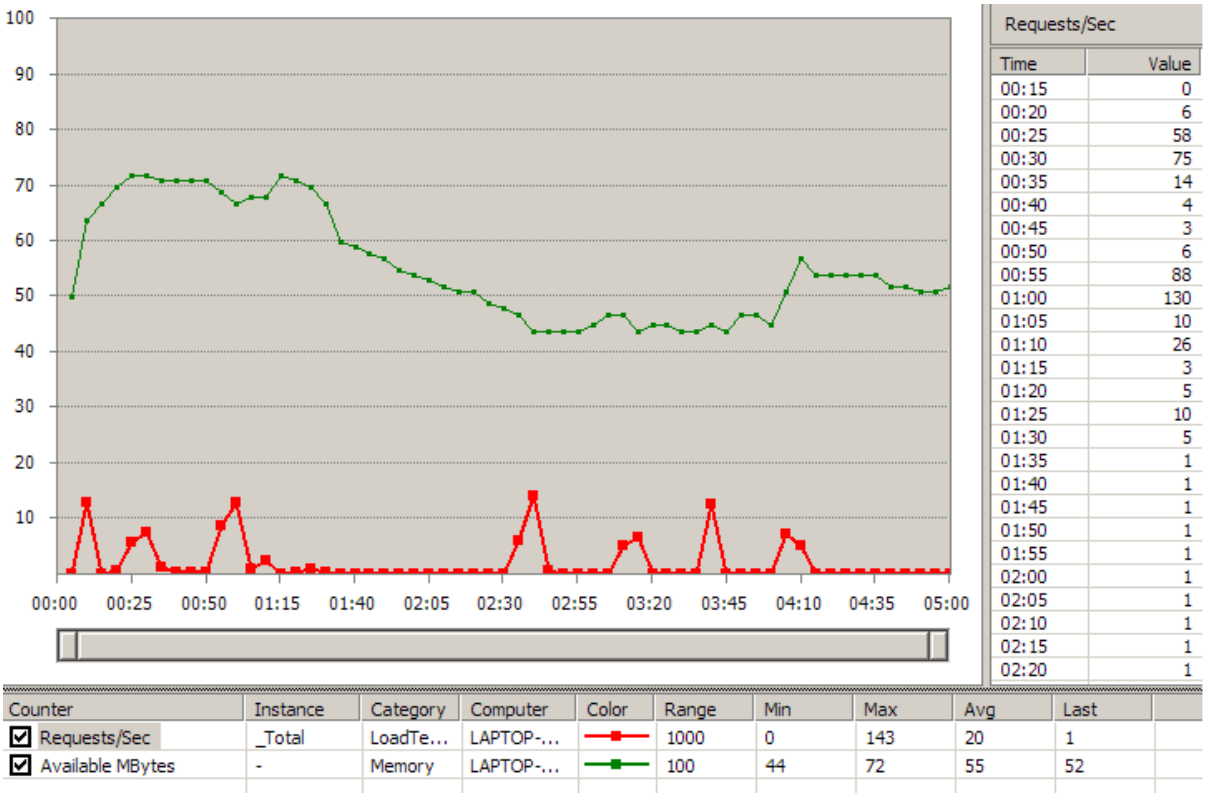
Load tests

Because a lot of users can use our application at the same time and we did not know how our application reacted to such a situation we had to do load tests. The built-in testing framework also includes load testing facilities. It can be used to execute a number of unit tests simultaneously, to simulate a multi user environment.

A lot of properties can be set: duration of the test, number of concurrent users, the test mix (e.g. 90% test1 , 5% test2 and 5% test3) … This gives a valuable insight in the performance of the application and helped us fix numerous multi-threading bugs.

Since Executing unit tests is not what the end-user is going to do this gives an unrealistic view on the performance. To have a realistic view on the performance, however, it is possible to create a web test. A web test can be recorded while surfing the application. In our application the most important and performance sensitive part is filling out a survey. Many users will be doing this at the same time. We recorded a regular session for a user: log in, fill out a survey and logout. Once recorded, this web test can be used in a load test so you can see how well your application performs in a real live situation.

These are the results of the load test described in the previous paragraph. It has a constant load pattern of 50 users, all using internet explorer 6 and a LAN connection to access the web site. During the load test there were more than 6000 requests in five minutes. The system is capable of handling at least 20 requests per second. In the graph you can see the amount of memory used and the number of requests per second. Other stats that can be shown include processor usage, load on the network interface, time to first and last byte, etc.



Requests/Sec

Time	Value
00:15	0
00:20	6
00:25	58
00:30	75
00:35	14
00:40	4
00:45	3
00:50	6
00:55	88
01:00	130
01:05	10
01:10	26
01:15	3
01:20	5
01:25	10
01:30	5
01:35	1
01:40	1
01:45	1
01:50	1
01:55	1
02:00	1
02:05	1
02:10	1
02:15	1
02:20	1

Counter	Instance	Category	Computer	Color	Range	Min	Max	Avg	Last
☑ Requests/Sec	_Total	LoadTe...	LAPTOP-...		1000	0	143	20	1
☑ Available MBytes	-	Memory	LAPTOP-...		100	44	72	55	52

Code Analysis

After every build our code is automatically analyzed by a built-in version of FxCop:

> ***"FxCop is a code analysis tool that checks .NET managed code assemblies for conformance to the Microsoft .NET Framework Design Guidelines. It uses reflection, MSIL parsing, and call graph analysis to inspect assemblies for more than 200 defects in the following areas: library design, localization, naming conventions, performance, code maintainability, reliability and security."***

When FxCop detects an error it shows a warning message in the to-do list. These warnings are very specific and detailed. E.g. this is a warning that you should use a `StringBuilder` instead of a string concatenation:

```
CA1818 : Microsoft.Performance : Change
QuestionRepresentation.Concat():Void to use StringBuilder instead of
String.Concat or +=
```

This is the code that triggers the '*Do not concatenate strings inside loops*' warning:

```
Public Sub Concat()
    Dim out As String = ""
    For i As Integer = 0 To 1000
        out += "this is bad code"
    Next
End Sub
```

The FxCop Warnings are very well documented and that makes fixing them easy. Every FxCop warning has a specific cause:

*"A method iteratively builds a string inside an iteration statement, such as a **for** or **while** loop, using the System.String.Concat method or using the addition (+ or &) or addition assignment (+=) operators."*

A rule description, a description why you should avoid the code that invokes the warning:

*"A System.String object is immutable. When two strings are concatenated, a new **String** object is created. Iterative string concatenation creates multiple strings that are un-referenced and must be garbage collected. For better performance, use the System.Text.StringBuilder class. **StringBuilder** maintains a mutable character buffer, which allows string concatenation and other changes to occur without the overhead of object creation and garbage collection. The initial capacity of the **StringBuilder** is an important performance consideration. When the capacity is no longer sufficient to hold the string, the original buffer is discarded and another buffer is created with twice the capacity of the original."*

FxCop gives also a hint on how to fix the violation:

*"To fix a violation of this rule, use an instance of the **StringBuilder** class to build the string inside the iteration statement. After the string is built, use the System.Text.StringBuilder.ToString method to access the string."*

And optionally it says when the warning can be ignored:

"Exclude a warning from this rule if the temporary strings are used inside the iteration statement. In this case, StringBuilder does not offer a performance gain because its ToString method also creates a new String object on each iteration."

This is a trivial example but the main part of the warnings are less obvious, obscure even. By using FxCop you can learn a lot on how to create state-of-the-art .NET libraries and how to avoid the main pitfalls of the .NET framework and languages.

External libraries

There are a number of libraries integrated in our application. In this chapter we are going to give an overview of those libraries and inform you about which tasks they fulfill.

ZedGraph

ZedGraph is a set of classes, written in C#, for creating 2D line and bar graphs. The classes provide a high degree of flexibility, almost every aspect of the graph can be modified. At the same time, usage of the classes is kept simple by providing default values for all of the graph attributes. We use it in our application to draw bar and pie charts. ZedGraph is open source and licensed under the Lesser GNU Public License, LGPL.

NFOP

NFOP is a Formatting Objects Processor (FOP) for XSL-FO that runs on the .NET Framework. It is a port from the Apache XML Project's FOP Java source to .NET's Visual J#. This makes it great for pure .NET reporting modules. We use NFOP in our application to generate PDF files of an XSL-FO representation of a survey. The XSL-FO file is build by transforming an XML representation of a survey using XSLT. NFOP is open source and licensed under an BSD license.

GDI+

GDI+ is an imaging library build by Microsoft.

> ***"Microsoft Windows GDI+ is the portion of the Windows XP operating system or Windows Server 2003 operating system that provides two-dimensional vector graphics, imaging, and typography. GDI+ improves on Windows Graphics Device Interface (GDI) (the graphics device interface included with earlier versions of Windows) by adding new features and by optimizing existing features."***

We use this library to convert bitmap images into enhanced metafile images, EMF images. The .NET framework is not capable of doing this conversion so we had to use this COM library for this task. We needed this type of conversion because the RTF file format only supports EMF images.

SharpZipLib

SharpZipLib is a free, open-source compression library for the .NET Framework. Data can be handled using the following formats:

- Zip: A format that archives multiple files into one and compresses the results that uses the deflate algorithm.
- Gzip: A format often used in Unix/Linux. Does not archive multiple files, so it is frequently used with Tar (see below). Also uses the deflate algorithm.
- Tar ("Tape archive"): Archives a set of files and directories into one file, but does not compress them. Often used with Gzip to compress.

- Bzip2: Uses the same file format as Gzip, but a different compression algorithm called BWT. Compresses better than Gzip, but is much slower.

The library and its source code are distributed under the GNU General Public License. We use it in our application to compress reports before sending them. Especially the RTF report with images can be compressed a lot.

JavaScript Libraries

To enhance the user experience we have included a number of JavaScript libraries in our application. These libraries are not used for core functionality. If a user has JavaScript disabled in his browser it is still possible to use our website.

Prototype

Prototype is a JavaScript framework that aims to ease development of dynamic web applications. Featuring a unique, easy-to-use toolkit for class-driven development and the nicest Ajax library around, Prototype is quickly becoming the code base of choice for web application developers everywhere. This library is needed to support two other libraries: Effects and Window. We do not use it directly in our application.

Effects

Web applications don't have to stay behind the current graphical trends in ultra-rich-desktop-applications. And they don't have to resort to fancy plug-ins that break usability and accessibility. Browsers and standards have reached a stage where advanced animations are finally possible to do with JavaScript. This is what the Effects JavaScript library tries to do. It is needed for the window library but we also use it directly. E.g. on the survey – add questions – preview.

Window

We use this library to show pop-ups. We show the help for a page in a pop-up and use it also to present the user with more information about a user. In the image below a popup with help is shown:

StyleSwitcher

To increase the text size dynamically we included this library. Depending on the preferences of the user it chooses one of three CSS files. Each one with another text size. It stores this preference in a cookie. In the image above you can see the links to the different text sizes.

Conclusion

Sweden is a great country. The weather can be very nice, and the schools are technologically highly advanced.

We learned a great deal about the .NET 2.0 framework. We used the new generics feature to great effect in the database layer of our website. We also learned a lot about Object Databases, even though we didn't use it beyond the prototyping stage. We had to work with the *good* old Access database, but we managed to make it an interesting and educative experience by designing and implementing a robust and easy to use framework that doesn't require the use of SQL statements.

Extensive use of test driven development turned out to be a very valuable decision and saved us lots of unnecessary debugging and frustrations, because errors were found early and automatically. Load testing also proved valuable. We found several errors that only occurred with multiple users simultaneously accessing the website. Without load testing it would have been almost impossible to find them.

Our knowledge about various file formats (DOC, RTF, PDF) greatly increased by creating them programmatically. We were amazed by the power of XSLT for transforming XML to different formats.

We also got a look into the new Microsoft Office Open XML file format. It will greatly improve the ability to generate and manipulate Microsoft file formats.

Appendices

Appendix A: The Case Against Access

Why are we writing this?

First of all because we want to make sure everyone who reads this is fully aware of the dangers involved in using Access as a database for web applications. We also want to make a strong statement that database corruption, scaling issues, data loss and random errors under 'heavy' load are very likely to occur and we can do absolutely nothing to prevent this. What follows is an attempt to explain why we can not prevent any of those disastrous things to happen.

Drivers, drivers, drivers

There are two options to connect to an Access database to a website. ODBC and Microsoft Jet. What follows is an overview of why both of these approaches are not suitable for a web application environment.

ODBC

The golden oldie ODBC. In the .NET Framework this binding is represented by the `System.Data.Odbc` namespace. The connection string, in that case, looks somewhat like this [1]:

```
Driver={Microsoft Access Driver(*.mdb)};
Dbq=C:\accessdb.mdb;
```

All right that looks good, let's give it a go. We wrote Unit tests and performed a load test. The unit tests do basic read/write operations: creating a record, deleting a record and querying for multiple records:

```
[TestMethod()]
public void ExecuteTest()
{
   OdbcCommand command;
   int rowCount;

   command = new OdbcDbCommand(
      "Insert INTO  COU(COU_T_NAME,COU_T_CODE)
   VALUES('te','te')"
   );
   rowCount = DB.Execute(command);
   Assert.IsTrue(rowCount == 1);

   command = new OdbcDbCommand(
   "DELETE FROM COU where COU_T_CODE= 'te'"
   );
   rowCount = DB.Execute(command);
}
```

Let's see how well this performs under a constant load of 25 users for three minutes. It quickly became clear that the ODBC driver is a source of descriptive error messages like `[Microsoft][ODBC Microsoft Access Driver] Unspecified Error`. There was also a logical one: `[Microsoft][ODBC Microsoft Access Driver] Too many client tasks`.

After a quick look at Google it appeared that there can be 10 concurrent connections at a certain time. This can be solved by connection pooling. But alas the ODBC driver does not support this. Another solution is programming a connection pool yourself, so we did. The "Too many client tasks" error disappeared; the unspecified error occurred at random intervals. We googled some more and found this on microsoft.com [2]:

> ***"When running Microsoft Jet in an IIS environment, it is recommended that you use the native Jet OLE DB Provider in place of the Microsoft Access ODBC driver. The Microsoft Access ODBC driver (Jet ODBC driver) can have stability issues due to the version of Visual Basic for Applications that is invoked because the version is not thread safe. As a result, when multiple concurrent users make requests of a Microsoft Access database, unpredictable results may occur. The native Jet OLE DB Provider includes fixes and enhancements for stability, performance, and thread pooling (including calling a thread-safe version of Visual Basic for Applications)."***

Microsoft Jet

So we tried the other driver: the Jet OLE DB provider, in the .NET Framework represented by the `System.Data.OleDb` namespace. This driver has built in support for multi-threading and connection pooling. That looked promising; we swiftly adapted our unit tests and db classes and ran the load test again: success! We tried to push our luck and unleash 50 users on the database, well that was not a wise thing to do:

"Övning.Domain.Test.DBTest.ExecuteTest threw exception: System.Data.OleDb.OleDbException: Unspecified Error"

So back to Google then we found this [3] and [4]:

"Microsoft Jet is deprecated.
Microsoft's current recommendation is to not use Jet at all. "No new feature enhancements will be made to Jet... It is highly recommended that while developing new applications, avoid using these [deprecated] components. Additionally, while upgrading or modifying existing applications, remove any dependency on these components" (Source: Microsoft's MDAC Road Map, under "Deprecated Components"). "

And in [1] we read:

"it [Microsoft Jet] was not intended (or architected) for the high-stress performance required by 24x7 scenarios, ACID transactions, or unlimited users, that is, scenarios where there has to be data integrity or very high concurrency."

Conclusion

So here we are, using a deprecated technology for something it was not designed for. Even the vendor self, Microsoft, has absolutely no trust in this technology and this is clearly stated on numerous places on its website. I hope you can understand we have troubles with our conscience to programming for Access. The question that is running trough our head during our sleepless nights is this one: Is it worth to invest even more time and effort on this obsolete technology and is it wise to continue using it for a task that it was not designed for? If you can give us an affirmative answer to this question we will weep and continue our work, albeit slightly less motivated.

Sources

[1] Connectionstrings, http://www.connectionstrings.com/

[2] KB299973, "*Using Microsoft Jet with IIS",* September 14 - 2005, http://support.microsoft.com/kb/299973/,

[3] PETER JOHNSON, "*Microsoft Jet database engine advice*", http://authors.aspalliance.com/PeterJohnson/JetAdvice.aspx

[4] PRASH SHIROLKAR, "*Data Access Technologies Road Map*", September 2005 http://msdn.microsoft.com/library/default.asp?url=/library/en-us/Dnmdac/html/data_mdacroadmap.asp

Further reading material:

[1] *"Reasons to use the native Jet OLEDB Provider "* http://www.adopenstatic.com/faq/whyOLEDB.asp

[2] *"ADO.NET Connection Pooling Explained"* http://www.ondotnet.com/pub/a/dotnet/2004/02/09/connpool.html

[3] *"Choosing the Appropriate Database"* http://support.microsoft.com/kb/168549/EN-US/

[4] "*How to keep a Jet 4.0 database in top working condition*" http://support.microsoft.com/?scid=kb;en-us;303528

Appendix B: Database Schema

Contents

System Database

COU Table

Description

This table stores a list with countries.

Fields

Name	Type	Attributes	Default	Description
COU_ID	Long			
COU_T_NAME	Varchar(50)	Nullable		The country name
COU_T_CODE	Varchar(50)	Nullable		county code

Indexes

Name	Columns	Unique	Clustered
COU_ID	COU_ID	True	False
COU_T_CODE	COU_T_CODE	False	False

Relationships

Master		Detail	
Table	Columns	Table	Columns
COU	COU_ID	ORG	ORG_V_CO

ERR Table

Description

This table stores a list of errors. The errors that are logged in the Hacklog. E.g. User Data Request, Wrong Password, Wrong Username, Banned user login attempt …

Fields

Name	Type	Attributes	Default	Description
ERR_ID	Long			
ERR_T_NAME	Varchar(50)	Nullable		The error name
ERR_T_DESC	LongText(0)	Nullable		An error description

Indexes

Name	Columns	Unique	Clustered
ERR_ID	ERR_ID	True	False

Relationships

Master		Detail	
Table	Columns	Table	Columns
ERR	ERR_ID	HCK	HCK_V_ERRCODE

HCK Table

Description

This is the hack log. It is a list of hack attempts or other events the administrator should know about.

Fields

Name	Type	Attributes	Default	Description
HCK_ID	Long			
HCK_T_LOGIN	Varchar(255)	Nullable		The name of the user who created this error.
HCK_T_PWD	Varchar(255)	Nullable		The faulty password
HCK_T_IP	Varchar(255)	Nullable		IP address of the client
HCK_T_ERRTIME	DateTime	Nullable		Date and time for event
HCK_V_ERRCODE	Short	Nullable	0	Type of error

Indexes

Name	Columns	Unique	Clustered
ID	HCK_ID	True	False

HLP Table

Description

Each and every page can have a number of help messages, in different languages. This is the message shown in the popup.

Fields

Name	Type	Attributes	Default	Description

HLP_ID	Long			The identifier for the message
HLP_CODE	Varchar(10)			The language identifier for the message
HLP_MSG	LongText(0)			The message itself
HLP_URL	Varchar(255)			The url of the page

Indexes

Some extra indexes are defined to prevent double entries. It is impossible to create a help message for a page in the same language twice.

Name	Columns	Unique	Clustered
PrimaryKey	HLP_ID	True	False
Url_CODE	HLP_URL	True	False
Url_CODE	HLP_CODE	True	False

LANG Table

Description

This is a list of languages

Fields

Name	Type	Attributes	Default	Description
LANG_ID	Long			
LANG_T_NAME	Varchar(50)	Nullable		Nederlands, English or Swedish
LANG_T_CODE	Varchar(50)	Nullable		The country code e.g. Be

Indexes

Name	Columns	Unique	Clustered
LANG_CODE	LANG_T_CODE	False	False
LANG_ID	LANG_ID	True	False

LIC Table

Description

This table stores a list of licenses and their restrictions. E.g. the bronze license can create 12 surveys with 3 war game participants.

Fields

Name	Type	Attributes	Default	Description
LIC_ID	Long			
LIC_T_TYPE	Long	Nullable	0	The license type
LIC_T_NAME	Varchar(50)	Nullable		The name of the license
LIC_T_ANNUAL	Long	Nullable	0	annual fee (€)
LIC_T_SURVEY	Long	Nullable	0	fee for survey (€)
LIC_T_SGAME	Long	Nullable	0	fee for Small Game (€)
LIC_T_LGAME	Long	Nullable	0	fee for Large Game (€)
LIC_T_USRS	Long	Nullable	0	Number of users allowed
LIC_T_SURS	Long	Nullable	0	Number of surveys allowed
LIC_T_GAMS	Long	Nullable	0	Number of war games allowed

Indexes

Name	Columns	Unique	Clustered
LIC_ID	LIC_ID	True	False

Relationships

Master		Detail	
Table	**Columns**	**Table**	**Columns**
LIC	LIC_ID	ORG	ORG_V_LICTYPE

ORG Table

Description

This table stores the data of an organisation.

Fields

Name	Type	Attributes	Default	Description
ORG_ID	Long			
ORG_T_NAME	Varchar(255)	Nullable		

ORG_T_LICNR	Varchar(255)	Nullable		User license number
ORG_V_LICYEAR	Long	Nullable	0	License valid until
ORG_V_LICTYPE	Long	Nullable	0	0 = blue 1 = bronze 2 = silver 3 = gold
ORG_T_ADDR1	Varchar(255)	Nullable		
ORG_T_ADDR2	Varchar(255)	Nullable		
ORG_T_ZIP	Varchar(255)	Nullable		
ORG_T_CITY	Varchar(255)	Nullable		
ORG_V_CO	Long	Nullable	0	ref to COU_ID
ORG_T_REG	Varchar(50)	Nullable		Region
ORG_T_PABX	Varchar(255)	Nullable		Telephone number to main switchboard
ORG_T_CPERS	Varchar(255)	Nullable		Contact person
ORG_T_TITLE	Varchar(255)	Nullable		Title of contact person
ORG_T_MAIL	Varchar(255)	Nullable		Email address
ORG_T_TEL	Varchar(255)	Nullable		
ORG_T_MBT	Varchar(255)	Nullable		Mobile telefone
ORG_T_FAX	Varchar(255)	Nullable		
ORG_T_UNIT	Varchar(255)	Nullable		department
ORG_T_NOTES	Varchar(255)	Nullable		Our private memo/notes field
ORG_V_TM	Long	Nullable	0	Default settings for printing (topmargin)
ORG_V_LM	Long	Nullable	0	Left margin
ORG_V_RM	Long	Nullable	0	Right margin
ORG_V_BM	Long	Nullable	0	Bottom margin
ORG_T_CSS	Varchar(255)	Nullable		Css file
ORG_T_LOGO	Varchar(255)	Nullable	"gfx/logo_UQ.gif"	Logo
ORG_T_MSG	LongText(0)	Nullable		System message for this org

ORG_V_STAT	Long	Nullable	1	System status 1 =Active 2 = revoked
ORG_T_DIR	Varchar(255)	Nullable		Directory name under /usr (8 chars, A-Z, 0-9)
ORG_T_DBPWD	Varchar(255)	Nullable		Envryption key for customer db (8 chars, A-Z, 0-9)

Indexes

Name	Columns	Unique	Clustered
LICORG	ORG_V_LICTYPE	False	False
ORG_ID	ORG_ID	True	False

Relationships

Master		Detail	
Table	**Columns**	**Table**	**Columns**
ORG	ORG_T_LICNR	INV	INV_V_ORG
ORG	ORG_T_LICNR	USR	USR_V_ORG

RANK Table

Description

This table stores a list of ranks or roles. Administrator, su and participant are the different ranks.

Fields

Name	Type	Attributes	Default	Description
RANK_ID	Long			
RANK_T_NAME	Varchar(50)	Nullable		

Indexes

Name	Columns	Unique	Clustered
RANK_ID	RANK_ID	True	False

Relationships

Master		Detail	
Table	**Columns**	**Table**	**Columns**

RANK	RANK_ID	USR	USR_V_RANK

STAT Table

Description

This table stores a list of statuses. A user can be active , revoked or temporarily banned.

Fields

Name	Type	Attributes	Default	Description
STAT_ID	Long			
STAT_T_NAME	Varchar(50)	Nullable		

Indexes

Name	Columns	Unique	Clustered
STAT_ID	STAT_ID	True	False

Relationships

A status can be assigned to users and organizations.

Master		Detail	
Table	**Columns**	**Table**	**Columns**
STAT	STAT_ID	ORG	ORG_V_STAT
STAT	STAT_ID	USR	USR_V_STAT

USR Table

Description

This table stores the data of a user.

Fields

Name	Type	Attributes	Default	Description
USR_ID	Long			
USR_T_LOGIN	Varchar(255)	Nullable		8 chars a-z, 0-9
USR_T_PWD	Varchar(255)	Nullable		8 chars a-z, 0-9
USR_V_PWDQ	Long	Nullable	0	Password question

USR_T_PWDA	Varchar(255)	Nullable		
USR_T_MAIL	Varchar(50)	Nullable		persons email address
USR_V_ORG	Varchar(50)	Nullable		Pointer to ORG_T_LICNR
USR_T_LASTLOG	DateTime	Nullable		Login previous to this one
USR_V_RANK	Long	Nullable	0	#RT#RT Different Rank Types checked with Instr()
USR_V_TM	Long	Nullable	0	Private printer settings (topmargin)
USR_V_LM	Long	Nullable	0	Left margin
USR_V_RM	Long	Nullable	0	Right margin
USR_V_BM	Long	Nullable	0	Bottom margin
USR_T_LANG	Varchar(50)	Nullable	"en-US"	Def lang for this user
USR_V_STAT	Long	Nullable	1	ref to STAT_ID
USR_V_FPWD	Long	Nullable	0	number of failed password attempts
USR_V_FPWDA	Long	Nullable	0	number of failed secret answer attempts
USR_T_TBTIME	DateTime		#1/1/2000#	the date time the user has been temporarily banned

Indexes

Some extra indexes have been defined to speed up the system. A user is searched for using his name, his status, his rank or organization.

Name	Columns	Unique	Clustered
Login	USR_T_LOGIN	True	False
PrimaryKey	USR_ID	True	False
ORGUSR	USR_V_ORG	False	False
RANKUSR	USR_V_RANK	False	False
STATUSR	USR_V_STAT	False	False

Organization Database

ANS Table

Description

The answer table stores answers from game users (GUR) on surveys (SUR).

Fields

Name	Type	Attributes	Default	Description
ANS_V_SUR	Long	Nullable	0	Foreign key SUR
ANS_V_GUR	Long	Nullable	0	Foreign key GUR
ANS_V_QUE	Long	Nullable	0	Foreign key QUE
ANS_T_TEXT	LongText(0)	Nullable		The answer itself
ANS_ID	Long			Identifier

Indexes

There are two indexes in this table. First of all the primary key. The other one is a clustered index spanning three columns. This to prevent an answer by the same user to the same question in the same survey.

Name	Columns	Unique	Clustered
ANS_ID	ANS_ID	True	False
GURANS	ANS_V_GUR	True	True
QUEANS	ANS_V_QUE	True	True
SURANS	ANS_V_SUR	True	True

CAT Table

Description

This table stores a list of question categories.

Fields

Name	Type	Attributes	Default	Description
CAT_ID	Long			
CAT_T_NAME	Varchar(50)	Nullable		

CAT_T_COM	Varchar(50)	Nullable		Comment on category

Indexes

Name	Columns	Unique	Clustered
CAT_ID	CAT_ID	True	False

Relationships

Master		Detail	
Table	Columns	Table	Columns
CAT	CAT_ID	QUE	QUE_V_CAT

CTL Table

Description

This table stores a list of ways of rendering a question using controls.

Fields

Name	Type	Attributes	Default	Description
CTL_ID	Long			
CTL_T_NAME	Varchar(50)	Nullable		name of control

Indexes

Name	Columns	Unique	Clustered
CTL_ID	CTL_ID	True	False

Relationships

Master		Detail	
Table	Columns	Table	Columns
CTL	CTL_ID	QUE	QUE_V_CTL

DOC Table

Description

This table stores a list of documents for a wargame.

Fields

Name	Type	Attributes	Default	Description
DOC_ID	Long			
DOC_V_TYPE	Long	Nullable	0	The type of the document
DOC_V_GAM	Long	Nullable	0	The war game the document is for.
DOC_T_TITLE	Varchar(50)	Nullable		
DOC_T_TEXT	Varchar(50)	Nullable		Comments
DOC_T_FILE	Varchar(50)	Nullable		The path to the file

Indexes

Name	Columns	Unique	Clustered
DOC_ID	DOC_ID	True	False
DOC_V_TYPEID	DOC_V_TYPE	False	False
DOCTYPEDOC	DOC_V_TYPE	False	False
GAMDOC	DOC_V_GAM	False	False

DOCTYPE Table

Description

The doc table stores a list of document types.

Fields

Name	Type	Attributes	Default	Description
DOCTYPE_ID	Long			
DOCTYPE_T_NAME	Varchar(50)	Nullable		

Indexes

Name	Columns	Unique	Clustered
DOCTYPE_ID	DOCTYPE_ID	True	False

Relationships

Master		Detail	
Table	**Columns**	**Table**	**Columns**

DOCTYPE	DOCTYPE_ID	DOC	DOC_V_TYPE

GAM Table

Description

This table stores the data for a war game.

Fields

Name	Type	Attributes	Default	Description
GAM_ID	Long			
GAM_T_NAME	Varchar(50)	Nullable		The name
GAM_T_TIME	DateTime	Nullable		Start date and time
GAM_V_OWN	Long	Nullable	0	ref to USR_ID
GAM_V_STAT	Long	Nullable	0	ref to STAT_ID

Indexes

Name	Columns	Unique	Clustered
GAM_ID	GAM_ID	True	False
STATGAM	GAM_V_STAT	False	False

Relationships

A war game has a number of documents, game users and belongs to a survey.

Master		Detail	
Table	**Columns**	**Table**	**Columns**
GAM	GAM_ID	DOC	DOC_V_GAM
GAM	GAM_ID	GUR	GUR_V_GAME
GAM	GAM_ID	SUR	SUR_V_GAM

GUR Table

Description

The participants or game users data is stored in this table.

Fields

Name	Type	Attributes	Default	Description
GUR_ID	Long			Game/User/Role id
GUR_V_USR	Long	Nullable	0	The user (ref to USR_ID)
GUR_V_GAME	Long	Nullable	0	ref to the Game (GAM_ID)
GUR_V_ROLE	Long	Nullable	0	ref to The role of this user in this game

Indexes

A clustered index is defined to prevent a user to be enrolled in a war game with the same role twice.

Name	Columns	Unique	Clustered
GAMGUR	GUR_V_GAME	False	True
GUR_ID	GUR_ID	True	False
ROLGUR	GUR_V_ROLE	False	True

Relationships

Master		Detail	
Table	Columns	Table	Columns
GUR	GUR_ID	ANS	ANS_V_GUR

QUE Table

Description

The question table stores questions.

Fields

Name	Type	Attributes	Default	Description
QUE_ID	Long			
QUE_V_CAT	Long	Nullable	0	ref to CAT_ID
QUE_T_TITLE	Varchar(50)	Nullable		
QUE_V_CTL	Long	Nullable	1	ref to CTL_ID
QUE_T_COM0	LongText(0)	Nullable		Situation the Problem
QUE_T_COM1	LongText(0)			Question itself

QUE_T_COM2	LongText(0)	Nullable		Comment field
QUE_T_ANS	LongText(0)	Nullable		
QUE_T_REQ	Bit(2)		0	Defines if the question has to be answered

Indexes

Name	Columns	Unique	Clustered
CATQUE	QUE_V_CAT	False	False
CTLQUE	QUE_V_CTL	False	False
QUE_ID	QUE_ID	True	False

Relationships

A question has a number of answers and belongs to one or more surveys.

Master		Detail	
Table	**Columns**	**Table**	**Columns**
QUE	QUE_ID	ANS	ANS_V_QUE
QUE	QUE_ID	QUESUR	QUE_V_ID

QUESUR Table

Description

This table defines the relations between the question and survey tables.

Fields

Name	Type	Attributes	Default	Description
SUR_V_ID	Long	Nullable	0	
QUE_V_ID	Long	Nullable	0	

Indexes

It has a clustered primary key to prevent doubles and two other indexes to enforce the relationship between QUE, QUESUR and SUR

Name	Columns	Unique	Clustered
QUE_V_ID	QUE_V_ID	True	True
QUEQUESUR	QUE_V_ID	False	False

SUR_V_ID	SUR_V_ID	True	True
SURQUESUR	SUR_V_ID	False	False

ROL Table

Description

The role table defines the roles of a participant in a wargame.

Fields

Name	Type	Attributes	Default	Description
ROL_ID	Long			
ROL_T_NAME	Varchar(50)	Nullable		
ROL_T_COMM	Varchar(50)	Nullable		

Indexes

Name	Columns	Unique	Clustered
ROL_ID	ROL_ID	True	False

Relationships

The role is assigned to a number of users. A survey can restrict access to users in a certain role. E.g. a survey only for internal participants.

Master		Detail	
Table	**Columns**	**Table**	**Columns**
ROL	ROL_ID	GUR	GUR_V_ROLE
ROL	ROL_ID	SURROL	ROL_V_ID

STAT Table

Description

The status table defines the status of a war game. A war game is either ongoing, created or finished.

Fields

Name	Type	Attributes	Default	Description
STAT_ID	Long			
STAT_T_NAME	Varchar(50)	Nullable		

Indexes

Name	Columns	Unique	Clustered
STAT_ID	STAT_ID	True	False

Relationships

Master		Detail	
Table	**Columns**	**Table**	**Columns**
STAT	STAT_ID	GAM	GAM_V_STAT

SUR Table

Description

The survey table stores the data for a survey.

Fields

Name	Type	Attributes	Default	Description
SUR_ID	Long			
SUR_T_NAME	Varchar(50)	Nullable		Name of the survey
SUR_V_GAM	Long	Nullable	0	ref to GAM_ID
SUR_T_LAYOUT	Varchar(50)	Nullable		
SUR_T_START	DateTime	Nullable		Start date of the survey
SUR_T_END	DateTime	Nullable		End date of the survey
SUR_V_SURTYPE	Long	Nullable	0	ref to SURTYPE_ID

Indexes

Name	Columns	Unique	Clustered
GAMSUR	SUR_V_GAM	False	False
SUR_ID	SUR_ID	True	False
SURTYPESUR	SUR_V_SURTYPE	False	False

Relationships

A survey has a list of answers, a list of questions and a list of roles, it needs references to the tables storing that data.

Master		Detail	
Table	**Columns**	**Table**	**Columns**
SUR	SUR_ID	ANS	ANS_V_SUR
SUR	SUR_ID	QUESUR	SUR_V_ID
SUR	SUR_ID	SURROL	SUR_V_ID

SURROL Table

Description

The survey – role table stores the relationships between survey and question.

Fields

Name	Type	Attributes	Default	Description
SUR_V_ID	Long	Nullable		a reference to survey
ROL_V_ID	Long	Nullable		a reference to role

Indexes

Name	Columns	Unique	Clustered
ROL_V_ID	ROL_V_ID	True	True
ROLSURROL	ROL_V_ID	False	False
SURSURROL	SUR_V_ID	False	False

SURTYPE Table

Description

The SURTYPE table stores a list of survey types.

Fields

Name	Type	Attributes	Default	Description
SURTYPE_ID	Long			
SURTYPE_T_NAME	Varchar(50)	Nullable		

Indexes

Name	Columns	Unique	Clustered
SURTYPE_ID	SURTYPE_ID	True	False

Relationships

Master		Detail	
Table	**Columns**	**Table**	**Columns**
SURTYPE	SURTYPE_ID	SUR	SUR_V_SURTYPE

Appendix C: Database Changes

In this document we explain the changes we made to the system database and the organization database. Please use the schema's as a reference.

System Database

We deleted the TXT and LANG tables since we are not using the same system to internationalize our application. We use resource files instead. This is the standard ASP.NET approach.

We added a RANK table since the rank is a list of ranks and should logically be in its own table. It is referenced by the user table. A user has a rank. This was done to make changes to rank names easier, and to improve the data integrity: by making this chance it became impossible for a user to have no rank or an invalid rank.

The HLP table was also added, in this table the data for the help information is stored for each page.

The type of USR_T_LASTLOG was changed from the type string to `DateTime` because, well because it is a `DateTime`. This improves data integrity by making it impossible to insert an invalid value and it makes localization easier: the `DateTime` is saved in a culture independent format.

In USR two fields were added: USR_V_FPWD and USR_V_FPWA for the failed password and failed password answer attempt count.

Organization Database

We added a DOC table to store documents attached to a war game. Doc has a reference to a war game and a certain type. The type of the document is stored in the new DOCTYPE table.

Instead of adding a table with the intuitive name X where x is a any integer to save answers of a survey we added a table with the slightly more intuitive name ANS. An answer is created by a war game user (GUR) and is for a certain question (QUE) in a survey (SUR) . This means that ANS has relations with GUR, QUE and SUR. By adding these relations it became impossible to insert corrupt data, the integrity of the data is enforced by these relations, this in stark contrast to the old system where you could add answers of non-existing users to non-existing questions. An other advantage of these relationships is that joins of multiple tables are much faster because the database system knows how the data relates.

To store the roles of participants allowed to take a survey we added the table SURROL. This table stores and enforces the relation between a role and a survey.

Appendix D: User manual for war game organizer

Table of Contents

Introduction

This document is the user manual for war game organizers. It gives an overview of the task a war game organizer can perform and explains how to perform them.

1. Registration

To register your organisation with Öving.nu, click the register link on the login page. You will be presented with a form to fill out.

Once the form is submitted you will receive two e-mails: one with the details of your war game organiser account and one with the invoice.

You can immediately start to set up and manage war games. You do not need to wait until the invoice has been paid.

2. Menu

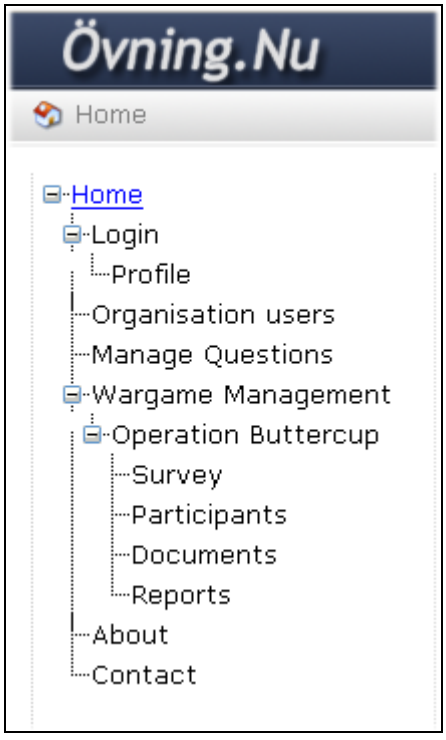


Figure 1. The menu

The menu shows the available management modules. Managing multiple war games at the same time is not a problem as each war game has its own menu entry. The different tasks are clearly visible and easily selectable.

3. User management

The user management page allows you to add users to your organisation.

Figure 2. User management

To add a user to your organisation enter a username and an e-mail address, and select a role.

Once created, the user will recieve his account details by e-mail.

4. War game management

Under War game management you can create and delete war games. Once you have created a war game it will appear in the menu. Selecting the war game allows you to change its name or start date and update the war game.

5. Question management

Before you can add questions to a survey you will have to create them first.

Title and Question Text are required fields, Description and comments are optional.

Then select the Question Type. Radio, select and checkbox will require at least one predefined answer, Text area and Textbox are open questions. If you need more than 5 answers to choose from you can add additional fields.

You can also specify whether or not a question has to be answered or not, with the

“Is required” checkbox.

Updating a question works exactly the same way.

To delete a question go to the delete tab and select the question(s) to delete. Be warned, if the question is part of a survey already, it will be removed from the survey.

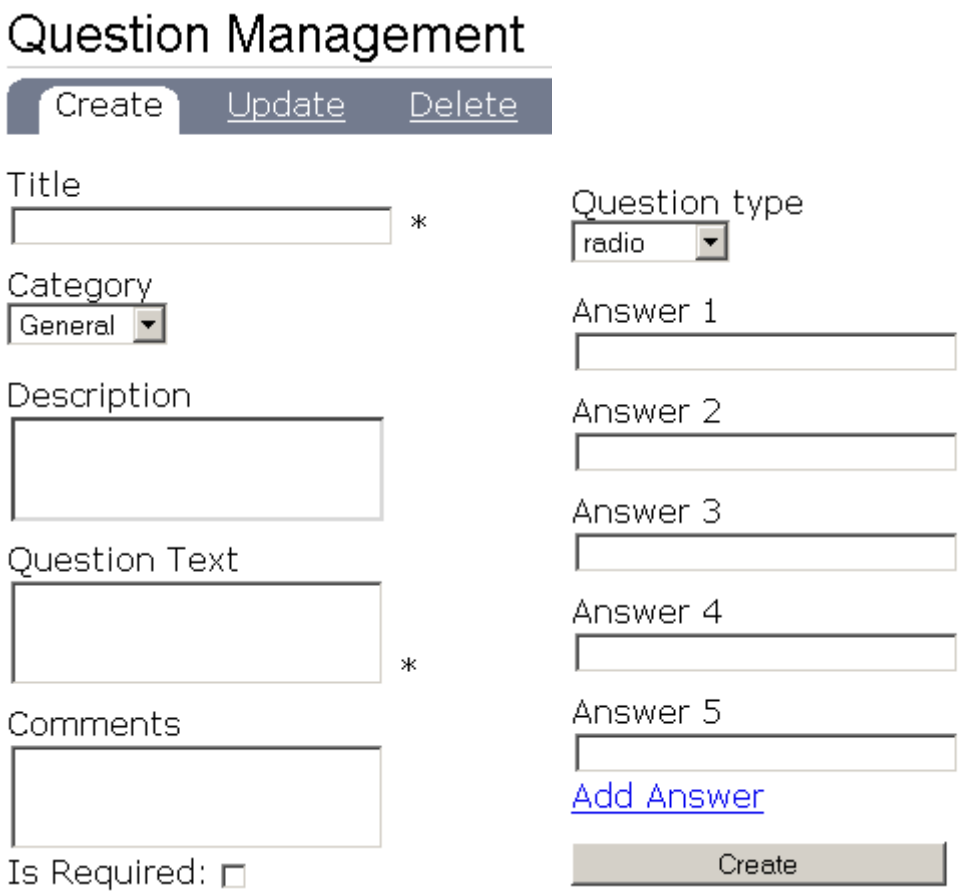


Figure 4. Question management

6. Survey management

Each war game has a survey link in the menu, which will bring you to the survey management page.

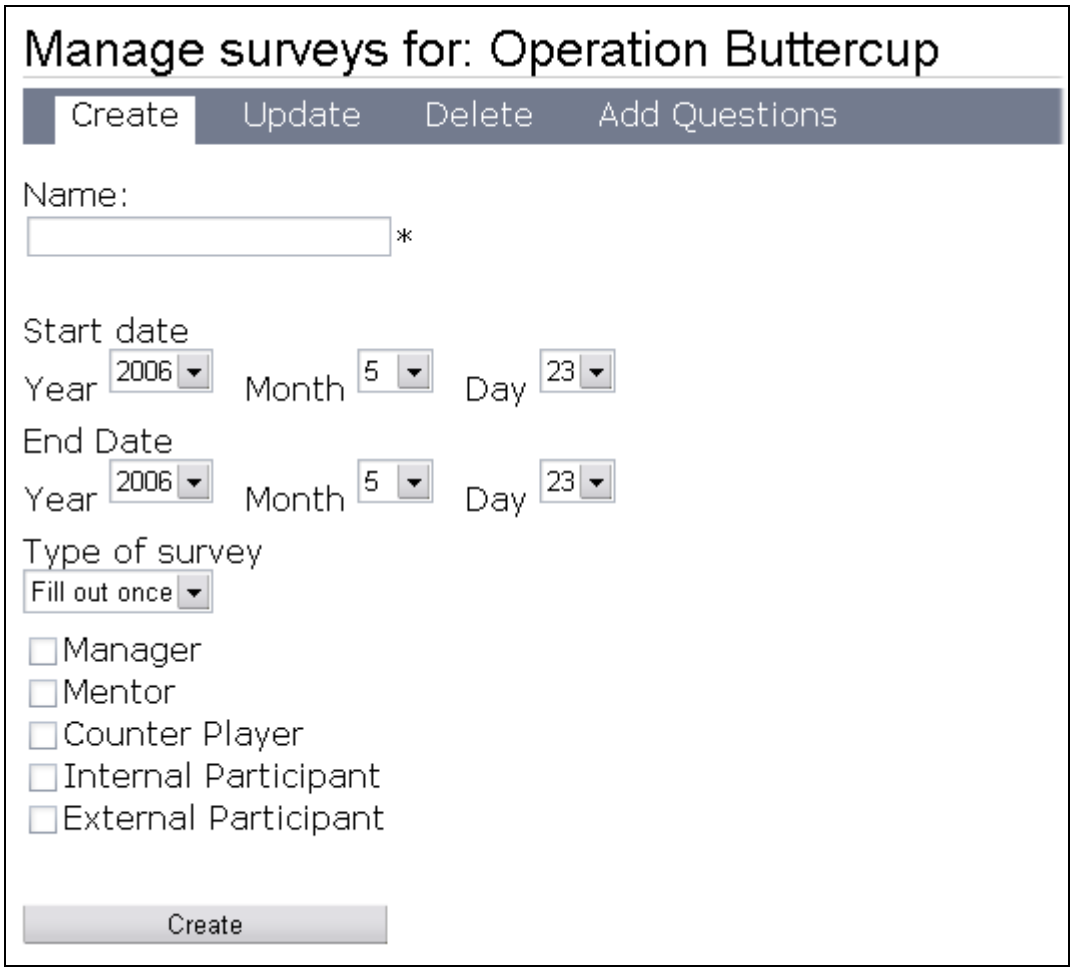


Figure 5. Survey management

To create a survey give it a name, select a start and end date, select a type, and select the war game roles that should be able to take the survey. Then click create.

Updating a survey is performed in exactly the same way and allows you to change the settings.

Deleting a survey is as simple as selecting a survey and clicking the delete button. Be warned however, if participants already answered the survey the answers will be lost.

Finally you can add questions to the survey.

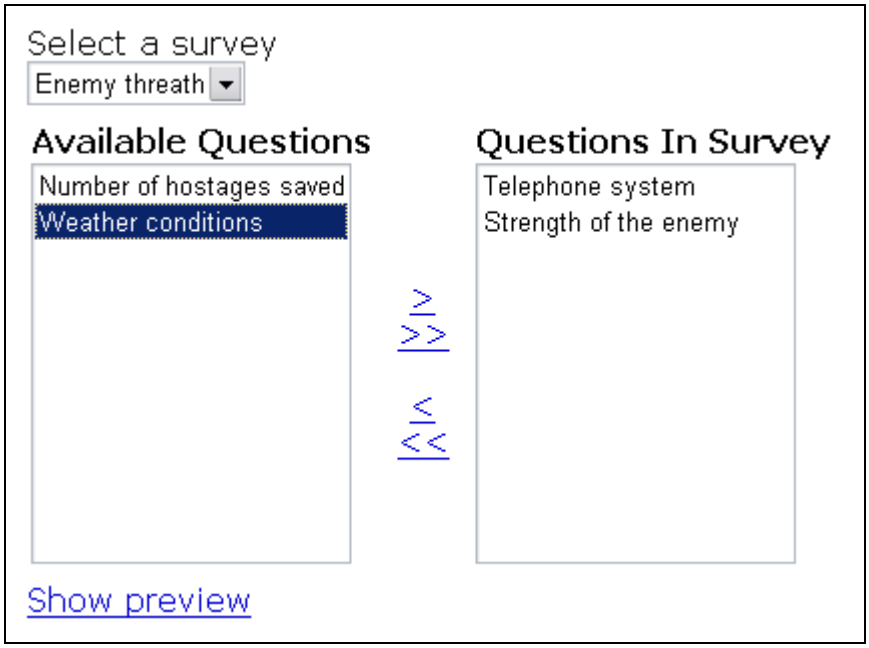


Figure 6. Add and remove questions to a survey

First select the survey you want to manage the questions or.

Next select the question you want to add or remove, and click the single arrow. The available questions are shown left, the questions that are already part of the survey on the right.

To add or remove all questions click the double arrows.

To get more information about a question you can double click on it. The question will be shown in a popup.

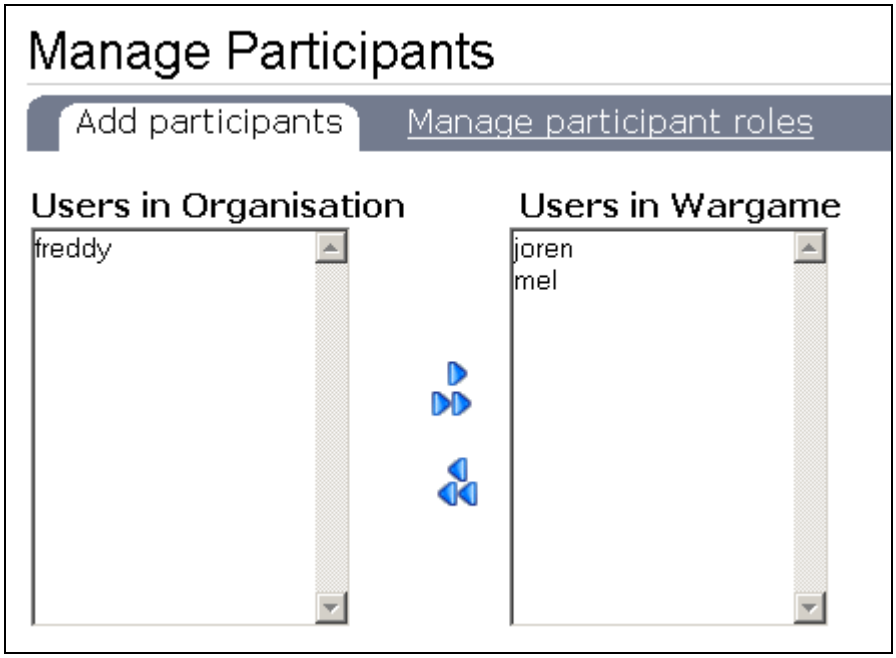
“Show preview” gives you a preview of the survey. All questions will be shown the way normal users will see them when taking the survey.

7. Participants

Here you can add participants to your war game. On the left you see a list with all users in the organisation, on the right a list with all participants in the war game.

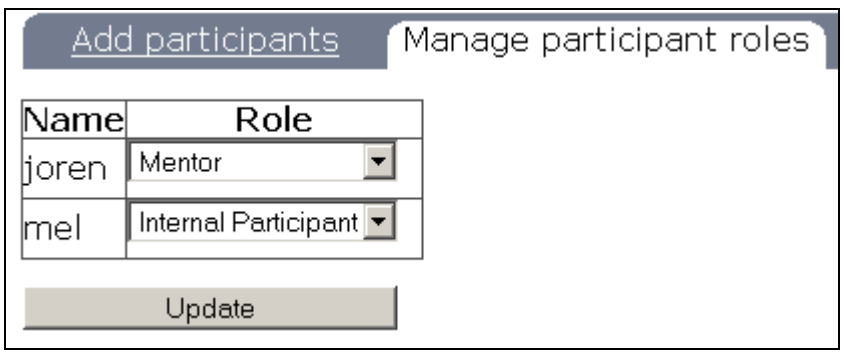
To add a participant to the war game first select a user from the organisation and move it to the war game with the right arrow. To remove a participant select a participant from the war game and remove him with the left arrow. You can also add or remove all wit the double arrows.

Double clicking on a user brings up a screen with additional details about this user.



Figuur 7. Add participants

Under the Manage Participant Roles tab you can change the roles of participants in the war game. The default role is internal participant.



Figuur 8. Manage participant roles

8. Document management

Here you can manage the documents that accompany the war game.

First select a category, then write a text or upload an existing document.

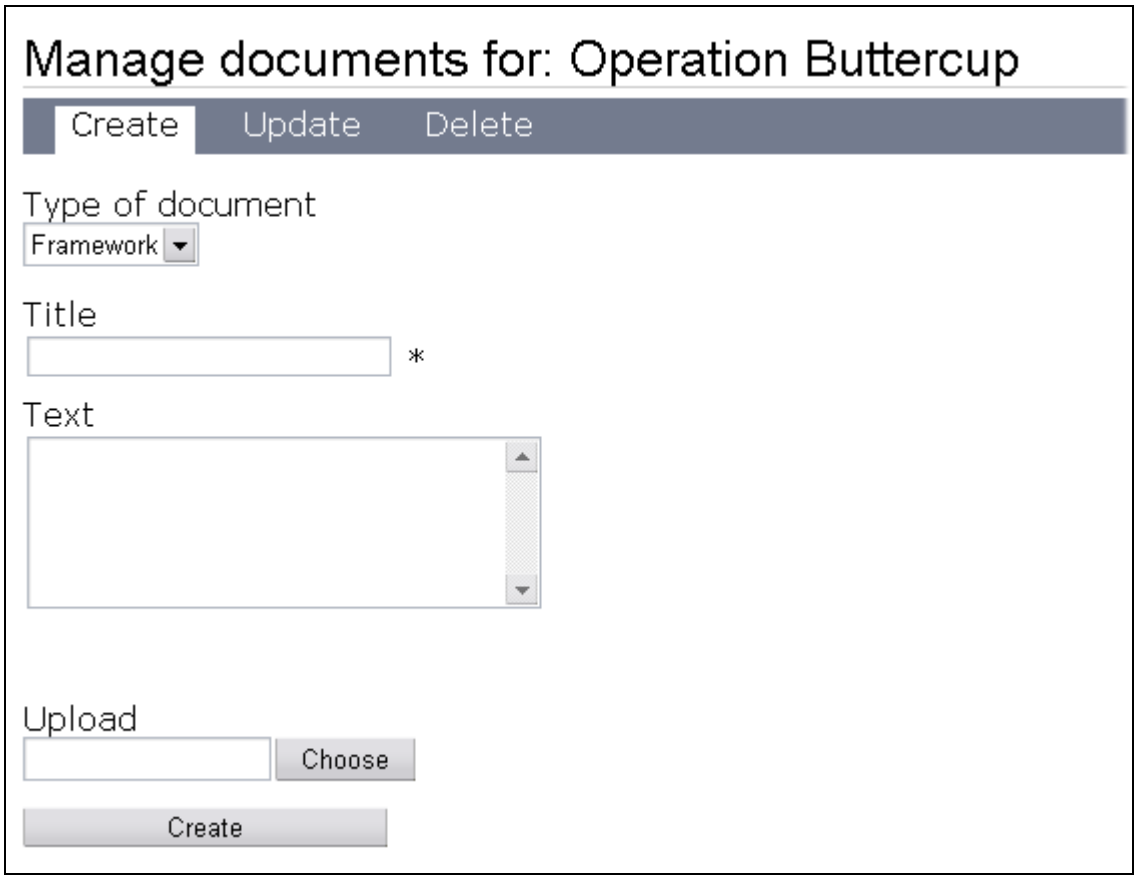


Figure 9. Document management

9. Reports

In the reports section you can view reports of the surveys.

Select a survey and a report will be generated.

You can download the reports in various file formats (doc, rtf, xls, html, xml and pdf).

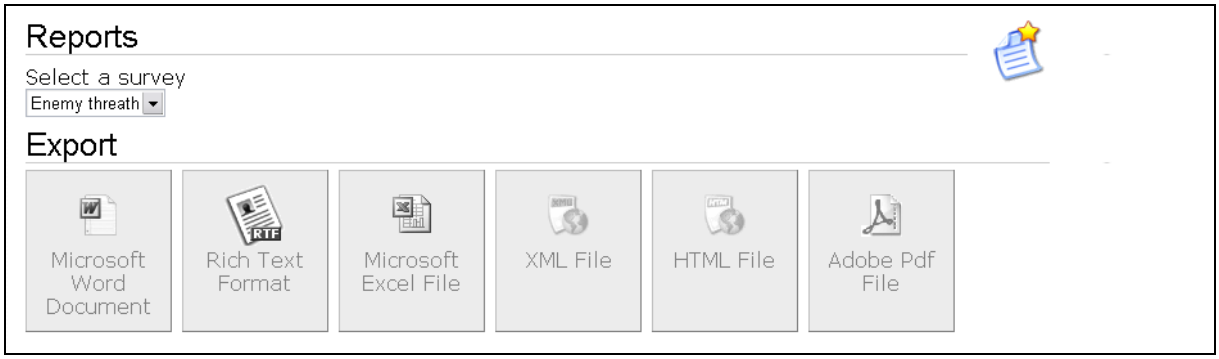


Figure 10. Reports

Appendix E: User manual for participants

Introduction

Övning.nu is a war game management system. As a war game participant you can help your organisation get a better overview of the progress of a war game by filling out surveys.

1. Login

To log in you simply enter the username and password that have been sent to you by the war game organiser.

2. Password recovery

If you forgot your password you can retrieve your credentials by submitting your username. A new password will be sent to your e-mail address. If you lost your e-mail address please contact your war game organiser.

3. Surveys

Once logged in you can use the menu to select the war game you are participating in. Available surveys will be shown as an icon.

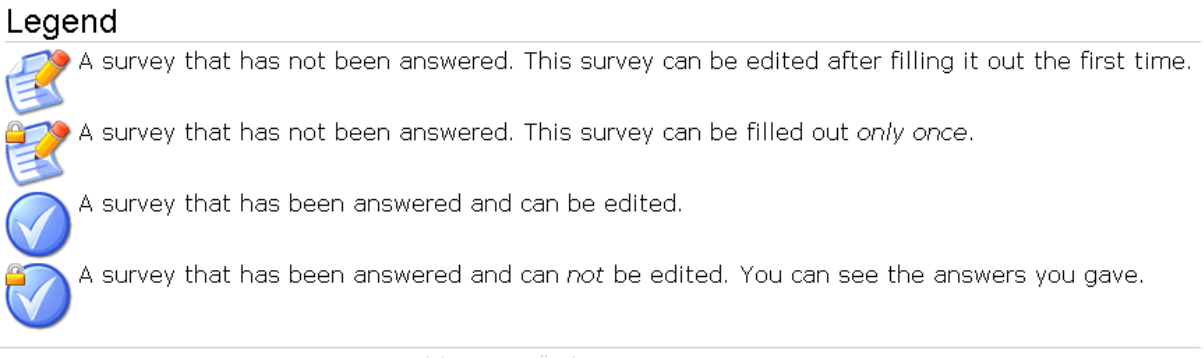


Figure 1. The different possible surveys

Depending on the state and the type of the survey you will see different icons, as shown above.

You can either:

- fill out a survey if you have not done so yet
- edit your answers if the type of survey allows this
- review your answers if editing is not allowed

To access the survey simply click on the icon.

You answer each question by selecting one or more answers, or entering your answer, depending on the type of question. The 'Next' button will bring you to the next question. Once you answered all

required questions you press finish to submit your answers. You will be taken back to the survey overview page.

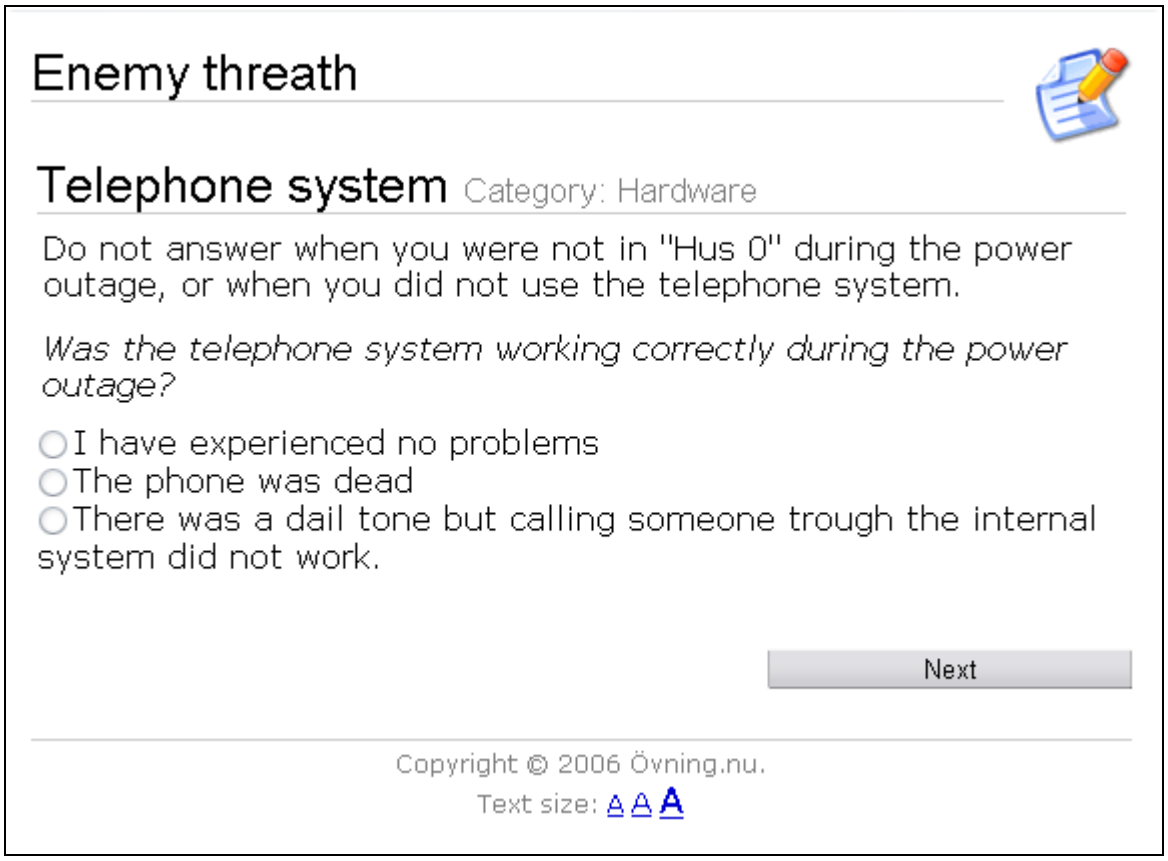


Figure 2. Example of a question

4. Profile

You can assess your profile through the link in the menu. In your profile you can:

- Change your e-mail address
- Set your preferred language
- Change your password
- Set your printer margins

5. Languages

Övning.nu is fully internationalized and supports several languages. You can select your language either in your profile or on the bottom of each page.

Appendix F: Logbook

Tuesday 28-02-2006

Lunch with our project coordinator Mr. Torben Svane.

Introduction to the project.

Friday 03-03-2006

Short meeting with Torben. We received the documentation of last year's project and access to the source code.

The rest of the day we spent reading the documentation and analyzing the code.

Brecht Desmeijter, who worked on this project last year, introduced us to the project and explained the general design.

Monday 06-03-2006

Our morning meeting with Torben had to be cancelled. Torben had other engagements in Göteborg.

We started to investigate ways to improve the project: The authentication process can be handled in a much more robust manner by the .NET 2.0 framework, as can the internationalization.

Started working on a prototype website to test the new features of .NET 2.0.

Tried to set up version control (Visual Source Safe)

Tuesday 07-03-2006

Meeting with Torben:

- We made a schedule for all future meetings.
- Torben explained the goals, requirements, and possible problems.
  - Improve the existing application
  - Generate reports in several document types
  - Security and privacy are key features

The rest of the day we continued prototyping, getting to know the framework and experimenting with new features (Internationalization support, Membership providers)

Wednesday 08-03-2006

Set up source control, and got it fully functional.

Experimented with db4o, an Object Database, trying several possible ways to store and retrieve data, comparing their speed and usefulness.

Refactored the existing object model into something more generic.

Thursday 09-03-2006

Created Unit tests for the refactored model and Load tests to see how well the new database handles heavy usage.

Friday 10-03-2006

Redesigned the layout to use more features present in the .NET 2.0 framework: Master pages, dynamic navigation, internationalization using resource bundles and make better use of CSS.

Monday 13-03-2006

Continue prototyping: getting some basic things (logging in, adding a war game to the database, …) to work so we could show them during the meeting on Wednesday.

Tuesday 14-03-2006

Finished the prototype: removed dead links, added breadcrumbs, tested everything in the prototype, extended the layout to support the new functionality (adding and removing a war game)

Wednesday 15-03-2006

Meeting with Torben. He was positive over the things we showed him but wanted us to use another database: Access instead of MSDE 2005 and multiple databases instead of only one. He gave us a better insight in what the system should look like, added a number of requirements, told us which different reports should be generated (Framework document, War game rules, Scenario document, Evaluation report) , what they should look like and how they are structured.

In the afternoon we started redesigning the prototype to work with the other databases (Access).

Thursday 16-03-2006

We started designing and implementing a basic persistence framework to store our objects in the database. We also did some research on the available report generating frameworks for ASP.NET.

Friday 17-03-2006

Continued working on the persistence framework, and created unit tests and did load tests to

see if the access database was capable of dealing with heavy load.

Monday 20-03-2006

10.30 – 17.00: First lessons BIT4: a course on edutainment, educational software.

Tuesday 21-03-2006

Creating a solution for the problem of multiple databases: One of the requirements is that each organisation has its own database so the data layer must be able to communicate with different databases within the same session. Also a way of creating those databases has to be designed and implemented: when someone registers an organisation a new database has to be created automagically.

Wednesday 22-03-2006

Meeting with Torben. He was positive over the changes we made. We showed him practically the same as the week before only with another database: multiple Access databases instead of one MSDE 2005 database. In the afternoon we improved the data layer to handle multiple databases.

Thursday 23-03-2006

Extended the functionality: added new classes to the domain to support creating war games and changed the data layer to handle this new functionality elegantly. Improved the reflection capabilities of the data layer. Added Unit Tests to test the new functionality.

Friday 24-03-2006

Created a help function with JavaScript pop-ups and a system to protect registration with bots, with image verification. Prototyped different approaches to report generation. One possibility is using the access report functionality and import it in the website with XML and XLS as XHTML.

Monday 27-03-2006

10.30 – 17.00: Classes.

Uploaded the website to test it on a server. Almost everything went ok.

http://0110.be/Övning.website/

Tuesday 28-03-2006

Explored the possibilities of using Crystal Reports to generate documents from the database. We created some test reports, with graphs.

Added Survey management. Surveys can be created, updated and deleted.

Made the menu and breadcrumbs fully dynamic. For each organisation it shows a list of their own War games

Wednesday 29-03-2006

Meeting with Torben:

- Surveys and Questions have become a priority for next week.
- Passwords shouldn't be too hard to remember. They have to be generated from a text file and have a format of WORD###, a word followed by some numbers. Zeroes, ones, O's and I's should be excluded to improve readability.

Question management added. Questions can be created, updated and deleted.

Implemented password generation.

Theme support added. Switching between various visual themes is very easy now.

Thursday 30-03-2006

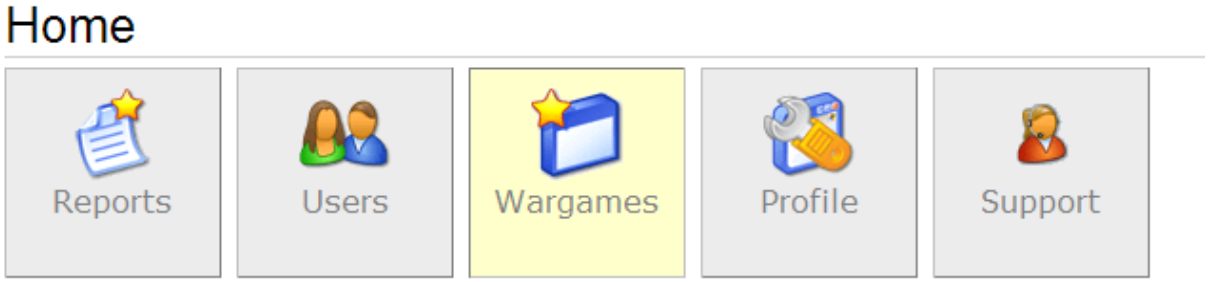
A War game Organiser can now add participants to a war game. Participants can get different roles (Manager, Mentor, Counter Player, External Participant…)

Questions can be added to a Survey.

Both actions use the same user control.

Various improvements to the GUI:

- Created a new Theme.
- Added buttons to make navigating easier.



- Added Style switcher script. Users can now easily adjust the text size, and their preferences are kept.

Friday 31-03-2006

Refactoring and Documenting.

Several changes to the data layer. Made reading objects from the database generic, greatly reducing the database related code. Our unit tests proved very useful here.

We went through all classes and added comments.

Monday 03-04-2006

10:15 – 17:00: Classes

Tuesday 04-04-2006

Enhanced Question and Survey management:

- Added Preview functionality for both. Preview shows a non-editable preview of the selected Question. Previewing a Survey gives a preview of all its Questions.
- Added the possibility to make Questions required or optional. Required questions must have at least one answer

Wednesday 05-04-2006

Meeting with Torben.

When assigning Participants to a War game or Questions to a Survey, it should be possible to view more information about the selected User or Question.

We did some more research on the MS PowerPoint File Format. Currently very little information is available. More information can be received from Microsoft upon buying a license. Office 12, the next version of Microsoft Office, will have completely new File Formats for most document types, based on XML. This will make dynamic document generation much easier in the future.

Thursday 06-04-2006

Altered the Login system: Depending on the role of the user that wants to log in various levels of security are applied:

- Normal users can log in by their username
- War game Administrators and Organisers have to enter a password
- Super Users (Site Administrator) have to enter a randomly generated verification code to prevent abuse from bots.

Failure to supply valid credentials is logged.

Implemented “Take Survey”. Participants of a war game can now fill out the survey. Answers are saved.

Friday 07-04-2006

First implementation of Document management. Documents can be added to a game to give more information about several aspects of the war game (Rules, Framework, Follow-up, Evaluation…)

Documents can either be written on the website as plain text or uploaded as a document.

Partly implemented the information window for Users and Questions, which shows additional information about the selected item. Sadly IE doesn't support the <options> tag.

In the afternoon we had an introduction to Flash 8 (13.00 – 16.45)

Monday 10-04-2006

10:15 – 17:00: Classes

Tuesday 11-04-2006

Continued writing the survey system until it was in a working state. Started our report: created a basic structure and wrote an introduction. Implemented a working prototype that changes text in a PowerPoint presentation.

Wednesday 12-04-2006

Meeting with Torben. He cleared up some issues concerning the survey system: it should be possible to set a timeframe when a survey can be taken. He explained how it could be possible to generate charts in PowerPoint trough the manipulation of excel files.

Thursday 12-04-2006

Easter holiday till Tuesday 18-04-2006

Monday 17-04-2006

Easter Monday.

Tuesday 18-04-2006

Added a start and end date to a survey. Changed the help system. Now it possible to add and edit a help-text to each and every webpage. Created a about and a contact page. Improved the overall look & feel by adding visual symbols to guide the user trough the website.

Wednesday 19-04-2006

Implemented a hack-log when a user tries to log in and provides a wrong password the system logs his mistake and blocks the user after 5 retries. These messages can be viewed and deleted by the site administrator.

Meeting with Torben: He insisted that we created a page where a user can change his or her preferences, a profile page: password, language and printer margins for the reports.

Thursday 20-04-2006

Created a profile page where a user can change his or her preferences. Started researching how to create excel files and charts for the survey answers. We found third party libraries: Aspose.Cells and Aspose.Charts and started looking into what we could do with these libraries.

Wrote the main use cases in our report, the use case “organize wargame” with its sub use cases “create war game”, “create user”, “create question”, ….. Another main use case is the use case “Take Survey”.

Friday 21-04-2006

Started implementing a chart prototype. Refactored the code that manages surveys and questions: getting things ready to generate charts and xml for the surveys.

Looked into the way PowerPoint links to excel files to create charts in PowerPoint trough excel. Found out that it is possible to link the two files without altering the PowerPoint but the files can not be dynamically linked and that’s a showstopper.

13:00-16:00 classes: group discussion about different games

Monday 24-04-2006

10:15 – 12:00: Classes

Started working chart generation with a third party library. Changed the way surveys are filled out: now it is possible for a user to change its answers if the survey is still accessible for the user. It can be made inaccessible by setting an end date or by defining that the user can not change its answers after he has filled out the survey. Started working on an XML representation of a survey. Looking in to a way to generate reports (PDF,DOC) with XML.

Tuesday 25-04-2006

Continued to work on the XML representation of a survey. Created a number of XSLT files to transform the XML into various document formats. More specifically a survey can be represented in: XML, XHMTL, RTF, DOC and XLS.

Added chart generating functionality based on our prototype. When the survey has answers these answers can be shown as a bar chart or pie chart.

Wednesday 26-04-2006

Made sure everything was functioning: fixed numerous bugs mainly in the chart generating code (e.g. handling an answer with frequency 0)

Meeting with Torben: Torben said that the charts where too sophisticated and that we should use another approach to generate charts. Discussed the use cases and the changes to the database. He requested some changes: he wants the user to see a splash page when they enter our site.

Thursday 27-04-2006

Looked for an other library to generate charts and stumbled upon ZedGraph which is open source. Converted the chart generating code to the new API, basically rewriting everything.

Implemented PDF generation using the XSL-FO library NFOP also with the XML representation of a survey: first the xml file is transformed to an XSL-FO file then NFOP creates a PDF file using the XSL-FO file. Improved the XML representation of a survey to include more data.

Friday 28-04-2006

Looked into a way of adding the graphs to the RTF and DOC file: tried to include PNG files in an RTF file but only Word supports PNG files in an RTF file. The only file format that all the RTF readers support is EMF. So we had to convert the PNG files to EMF files. This is not supported in the .NET framework, but it is possible by using an old fashioned COM library (gdiplus.dll).

EMF files have the tendency of becoming really large. We do not want to send 12MB files to the user so we compress the RTF file before sending it to the user.

Monday 01-05-2006

1 May: vacation.

Tuesday 02-05-2006

Some adjustments and improvements to the generated documents.

Refactored the document generation code to a more easy-to-use state.

Implemented a splash page as requested. Users that are not logged in see this page and not the actual site layout.

Wednesday 03-05-2006

Meeting canceled. Moved to Monday 08/05

Worked on our report.

Translated several pages so internationalisation is fully supported.

Thursday 04-05-2006

Several bugs fixed and cleaned up some code.

Wrote some more documentation. All code documentation is compiled using InDoc to present it in an MSDN-like way.

Friday 05-05-2006

Worked on our report.

Translated some more pages.

Published the preliminary website and fixed some configuration settings to make it work.

See http://www.oving.nu

Monday 08-05-2006

Morning: Classes.

Afternoon: Meeting with Torben.

He had several suggestions about the look of the site and the methods used to handle user input. Some radiobuttonlists and textboxes should be changed to dropdownlists.

Tuesday 09-05-06

Several optimizations. Reduced the number of database actions as much as possible to maximize performance.

Added a thourough description of the database layer to our report.

Wednesday 10-05-2006

Added extra validation in several places to ensure data integrity. It is no longer possible to create Users, War games, Surveys or Questions with the same name.

Thursday 11-05-06

Applied the requested changes to the user interface.

Documented all code that wasn't documeted yet.

Friday 12-05-06

Changed some more aspects of the user interface. The site is now more consistent and more user friendly.

Some small fixes and improvements.

Worked on our report.

Monday 15-05-2006

Morning: Classes.

Afternoon: Meeting with Torben.

Changed the layout of the survey management page. The preview is now in a more logical place.

Changed the representation of surveys when they are filled out. They no longer appear in a popup window but in a normal page instead.

Fixed several redirects to make the page flow more logical.

Tuesday 16-05-06

Added the possibility to assign roles withing a wargame to participants. Depending on their role participants can get different surveys to answer.

Added role dependent surveys. A participant only sees the surveys he can take.

Some changes to the splach page.

Wednesday 17-05-2006

Translated part of the website to Dutch and Swedish. The part for normal users (participants) is completely internationalized.

Worked on our report.

Thursday 18-05-06

Several more bugfixes and improvements. Added several icons to improve the usability and intuitivity of the Gui. Fixed some icons to support transparancy in Internet Explorer.

Friday 19-05-06

More fixes. Reports will no longer be generated for a survey if the survey contains no questions.

Worked on our report.

Monday 22-05-2006

Classes: presentation of the edutainment projects.

Meeting with Torben.

Worked on our report.

Tuesday 23-05-06

Added “answers per participant” and “answers per question” overviews to the reporting part. This gives an overview of the answers per question or per participant. Answers to open questions can now also be viewed.

Several fixes and improvements.

Wednesday 24-05-2006

Added feedback after successfully completed actions so it is more obvious that something went right.

Worked on our report.

Thursday 25-05-06

Worked on our report, some more fixes and improvements.

Friday 26-05-06

Wrote the user manuals

Preparation for the presentation.

Worked on our report.

Monday 29-05-06

Final presentation and delivery of our project.