System Analysis:
Automated Electronic Registration System

Name: Harley Green
Project Supervision: Alta van der Merwe
Organisation: UNISA
Email: harleygreen@unedspeed.net & Vdmeraj@unisa.ac.za
Date: 15 July 2003
System Design: Automatic Electronic Registration System

Table of Contents

Design Overview 3
System Level Use Case Diagram 4
Architecture Diagram 5
  Business Logic Layer 5
  Data Access Layer 8
  User Interface 9
    Overview 9
    Individual Pages 10 - 18
Subsystems 19
Use Cases 21
  Individual Use Cases 21 - 32
References 33
Design Overview

The Automatic Registration System has been designed for implementation using the Microsoft .Net platform, running on IIS. The development tool to be used to develop the system is Microsoft Visual Studio©v2003, and the language to be used will be C#. The system will interface with the existing Unisa student database. However, for the purposes of testing and evaluation, the system will use a Microsoft SqlServer©database, with a schema developed specifically for this project.

The choice of this technology was made primarily due to my desire to learn this new language and technology. Further, my initial assessment of the development environment was that it was not only suitable for large-scale web development, but that it provides excellent support for such development. It is developed and supported by one of the largest players in the software industry, and thus is not likely to disappear from the market place in the foreseeable future.

The architecture makes typical use of layering, i.e. the system is divided into 3 layers, namely the User Interface Layer, the Business Logic Layer, and the Data Access Layer. Due to the stateless nature of http, the application is designed to exploit the advantages of statelessness.

The User Interface Layer takes advantage of the Asp.Net architecture, and uses a modified version of the Model-View-Controller pattern (Gamma, 1995), the Page Controller (Microsoft, 2003).

Due to the minimal domain logic required, many of the Business Logic Layer classes implement the Active Record pattern (Fowler, 2002).

The Data Access Layer is a thin layer between the BLL and the physical database. As many of the BLL classes implement Active Record, a single class, which provides access to the database, should be sufficient. In the interests of code reuse, a class available for download from the Microsoft Developer Network, for use with Microsoft’s SqlServer database, is used. The design of the data access layer has been heavily influenced by the availability of this code. However, as it provides all its functionality through static methods, it is a fairly inflexible solution on its own. Porting to an alternative database at a later stage would require extensive modifications to the Business Logic Layer.

To increase database independence, a facade (Gamma, 1995) is used to shield the BLL from using this class directly. This is not a typical implementation of this pattern, as the façade itself is created using a singleton factory class (Gamma, 1995).

Of further interest, due to the native support provided by the .net platform for controlling access to sub-directories, the Verify Student use case, which was present in the original analysis, has been removed, and replaced with the .net functionality.
System-Level Use Case Diagram

The system-level use case diagram gives a broad overview of the scope of the project, and the different sub-systems and actors involved.

The actors identified are:

1. User
   a. Prospective Student
   b. Registered Student
2. Administrator
3. Despatch Department
4. Expert System
5. Student Database
6. Payment Gateway
Architecture Diagram

The static class diagram depicts the primary classes identified in the analysis. There are a number of auxiliary classes, the majority of which are simply attributes of the various classes. These have been omitted from this diagram for the sake of simplicity and brevity. All, however, are identified in these diagrams either directly, or as attributes of the classes depicted.

Business Logic Layer

```
+Constructor()
+Verify()
+RegisterForCourses()
+Save()
+CheckQualifications()
+MakePayment()

-StudentNumber : string
-IdentityNumber : string
-FirstNames : string
-First Names : string
-Initials : string
-ProposedQualification : ProposedQualification
-Specialisation : Specialization
-MaidenName : string
-Gender : Gender
-Disability : Disability
-TelephoneNumbers : TelephoneNumber
-CollaborateWithOthers : bool
-PostalAddress : Address
-PhysicalAddress : Address
-CorrespondenceLanguage : Language
-ExamCentre : ExamCentre
-HomeLanguage : Language
-Nationality : Nationality
-PopulationGroup : PopulationGroup
-Occupation : Occupation
-EconomicSector : EconomicSector
-PreviousInstitutions : Institution
-MostRecentInstitution : int

+VerifyApplication()

+Constructor()
+GetBalance()

+Constructor()
+Save()

+MakePayment()

+Constructor()
+GetBalance()
```

All the private attributes shown on this diagram will be exposed via getter/setter methods, which have not been included on the diagram for the sake of brevity.
The structure of the student class is given more clearly below:
A number of other class relations have been identified, as depicted below:

All classes deriving from LookupClass are simple, domain restricted classes that offer a number of choices. They model lookup tables in the underlying database.
Data Access Layer

```
«factory»
DAL
  - _dataHelper : DataHelper
  + Initialize(in database : string) : void
  + GetHelper() : AccessDataHelper

«facade»
DataHelper
  + ExecuteNonQuery() : int
  + ExecuteDataSet() : DataSet
  + ExecuteScalar() : object

SQLDataHelper

SQLHelper
  + ExecuteNonQuery() : int
  + ExecuteDataSet() : DataSet
  + ExecuteScalar() : object
```
**User Interface:**
The individual user interfaces are intended as a purely functional prototype. They are not aesthetically pleasing, and are intended as a point of reference for both programmer and graphic web designer.

**Overview**

[Diagram showing the flow of the user interface with various pages such as Welcome.htm, StudentNoApplication.aspx, Login.aspx, CreateOnlineAccount.aspx, MyUnisa.aspx, AccountBalance.aspx, Payment.aspx, ConfirmPayment.aspx, SSL, ThankYouPayment.aspx, StudentDetails.aspx, Register.aspx, ConfirmRegistration.aspx, ThankYouRegistration.aspx, Online Payment Pages, Online Registration Pages.]
Welcome.htm

Welcome to the Online Registration.
If you have already registered as an online user, please CLICK HERE.

If you already have a student no, and would like to register online, view your account balance, or make an online payment, please CLICK HERE.

If you do not have a student number yet, you will need to apply for one. Please CLICK HERE.

StudentNoApplication.aspx

The full page is not visible in a browser. The complete page is available in the repository, under its name.
CreateUserName.aspx

Welcome to MyUnisa. To access your account online, please create a username and password:

Student No

Last Name

Please choose a Username

Please choose a password

Please confirm your password
Login.aspx

Welcome to MyUnisa. Please enter your username and password

User Name

Password

MyUnisa.aspx

Welcome, Harley Green

Update Personal Details

My Account

Make Online Payment

Register Online
### AccountBalance.aspx

**Student No.** 32440952  
**Name** Harley Green

#### Transactions

<table>
<thead>
<tr>
<th>Reference</th>
<th>Debt</th>
<th>Credit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>12345</td>
<td>880.00</td>
<td></td>
<td>INF 412</td>
</tr>
<tr>
<td>54321</td>
<td></td>
<td>120.00</td>
<td>Receipt</td>
</tr>
</tbody>
</table>

**Balance:** XXXXX
Payment.aspx

The full page is not visible in a browser. The complete page is available in the repository, under its name.
ConfirmPayment.aspx

The full page is not visible in a browser. The complete page is available in the repository, under its name.
ThankYouPayment.aspx

Payment of Study Fees

Dear Mr Green,

Thank you. Your payment of R500 has been accepted.

Your reference no is #123456789

Please make a note of this number, or print this page for your records. A receipt will be sent to you via email.

Department of Finance
University of South Africa
StudentDetails.aspx

The full page is not visible in a browser. The complete page is available in the repository, under its name.

Register.aspx

To select courses for registration, please select the course in the drop-down box below, and click the "Add" button. Repeat for each course you wish to add. Click "Submit" once you have selected all your chosen courses.

Courses: INF 412 - Honours Project Module
ConfirmRegistration.aspx

You have chosen the following courses. Please check these courses, and press "Confirm".

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INF 412</td>
<td>Project Honours Module</td>
</tr>
<tr>
<td>INF 412</td>
<td>Project Honours Module</td>
</tr>
</tbody>
</table>

<< Back  Confirm

ThankYouRegistration.aspx

Thank you.

Your course selection will be evaluated. You will be informed via email of the success of your registration application.

Undergraduate Registrations
University of South Africa

Return to MyUnisa
Subsystems
The Automated Registration System can be seen as two largely independent subsystems, viz. the Online Registration Subsystem, and the Online Payments Subsystem. These are illustrated by their respective use case diagrams.

![Use case diagram of the Automated Registration System]
Online Payment System

Registered Student

Check Account Balance

Student Database

Make Online Payment

Payment Gateway
Use Cases

Use Case: Apply for Student Number

Brief Description
This is one of the primary use cases in the system. All prospective students who wish to make use of the Automated Online Registration System use it. This use case covers the steps involved in allocating a student no to a prospective student.

Primary Actor
Prospective Student

Other Actors
Student Database.

Uses
None

Extends
None

Context Diagram

Preconditions
The prospective student must have an email address for notification purposes.

Flow of Events
Normal Flow of Events
1. The use case begins when a prospective student enters the StudentNoApplication web page.
2. The system displays an enrollment form.
3. The prospective student enters his/her personal information.
4. The system validates the information to ensure it is complete.
5. The system displays a message informing the applicant of the success of his/her submittal.
6. The system verifies and validates the qualifications entered by the prospective student.
7. The system allocates the prospective student a student number.
8. The system persists the information to the student database.
9. The system emails the prospective student, informing him/her of their new student number.
10. The use case ends.

Alternative Flow of Events
Alternative 1: The qualifications are not accepted.
   1. The system rejects the qualifications that are entered.
   2. The system routes the information to a human analyst.
   3. The human analyst rejects the qualifications.
   4. The system emails the applicant, informing him/her of the reason for the rejection of his/her application.
   5. The use case ends.

Alternative 2: A human analyst accepts the qualifications.
   1. This alternative begins at step 3 of alternative one.
   2. The analyst accepts the qualifications.
   3. The system returns to step 6 in the normal flow of events.

Postconditions
The prospective student has been informed of the result of his/her application via email.
Sequence Diagram

User Interface
This use case uses the Welcome.aspx and the StudentNoApplication.aspx pages.

Outstanding Issues
None
Use Case: Check Account Balance

Brief Description
This use case may be used as a stand-alone use case, or may be called by another use case. It provides information to a student or the system regarding their current balance.

Primary Actor
Student

Other Actors
Student Database

Uses
None

Extends
None

Context Diagram

Preconditions
The student must be a registered student.

Flow of Events
Normal Flow of Events
1. This use case begins when the student requests his/her current student account balance.
2. The system determines the student’s balance from the database.
3. The system displays the balance.

Alternative Flow of Events
Alternative 1: System request.
1. This alternative begins when the system requires the account balance.
2. The system determines the student’s balance from the database.
3. The system returns the balance.

**Postconditions**
None

**Sequence Diagram**

![Sequence Diagram Image]

**User Interface**
This use case uses the Login.aspx, MyUnisa.aspx and AccountBalance.aspx pages.

**Outstanding Issues**
None
Use Case: Make Online Payment

Brief Description
This use case is central to the Automated Registration System. The use case covers the procedure for enabling students to make online payments towards their student accounts.

Primary Actor
Student

Other Actors
Payment Gateway
Student Database.

Uses
Check Account Balance.

Extends
None

Context Diagram

Preconditions
The student is registered with Unisa.
The student has an email account.

Flow of Events
Normal Flow of Events
1. This use case begins when the student requests the MakeOnlinePayment page.
2. Use Check Account Balance.
3. The system displays the current account balance, and requests the amount to be paid.
4. The Student enters the amount he/she wishes to pay.
5. The system requests the credit card information.
6. The system validates the entered information.
7. The system forwards the information to the Payment Gateway.
8. The system displays an acknowledgement of the payment to the student on a web page.
9. The Payment Gateway processes the credit card information, and returns the status of the payment.
10. The system updates the student's account.
11. The use case ends when the system emails a receipt to the student.

Alternative Flow of Events
Alternative 1: The Payment Gateway rejects the payment.
1. This alternative begins after step 9, if the Payment Gateway returns an unsuccessful status.
2. The use case ends with the system emailing a notice to the student, informing the student of the failed payment.

Postconditions
The student database is updated as necessary to reflect the correct balance.

Sequence Diagram

User Interface
This use case uses the Login.aspx, MyUnisa.aspx, AccountBalance.aspx, Payment.aspx, ConfirmPayment.aspx and ThankYouPayment.aspx pages.
Outstanding Issues

None
Use Case: Register Online

Brief Description
This is the central use case in the system. All students who wish to register via the Automated Online Registration System use it. This use case covers the steps involved in registering a student for a specific list of courses.

Primary Actor
Student

Other Actors
Student Database.
Dispatch Department.
Expert System

Uses
Update Personal Details.

Extends
None.

Context Diagram

Preconditions
The Student has a Student Number.
The Student has an email address.

**Flow of Events**

Normal Flow of Events
1. This use case begins when the student requests the Register web page.
2. Use Update Personal Details use case.
3. The system displays the Register web page, which allows the student to enter the details of the courses for which he/she wishes to register.
4. The student enters the course details.
5. The system verifies the course details against the Expert System actor.
6. If the expert system accepts the proposed courses, the system persists the information to the student database.
7. The system emails confirmation to the student.
8. The system emails the dispatch department, informing them of the registration, and requesting that the required documentation is sent.
9. The use case ends with the system displaying a confirmation page to the student.

Alternative Flow of Events

Alternative 1: The Expert System rejects the application
1. This alternative begins after step 7, if the expert system rejects the application.
2. The system displays a notice informing the student of the problem.
3. The system returns to step 4.

Alternative 2: The Student cancels the operation.
1. This alternative may begin at any step.
2. The student explicitly ends the process by clicking cancel on any of the pages, or implicitly by simply not stopping.
3. If the cancel is implicit, the system retains the information entered until the end of the default session time is reached, at which point the information is disposed of.

**Postconditions**

The student database has been updated to reflect any changes in the student’s status, and is in a consistent state.
Sequence Diagram

User Interface
This use case uses the Login.aspx, MyUnisa.aspx, StudentDetails.aspx, Register.aspx, ConfirmRegistration.aspx and ThankYouRegistration.aspx pages.

Outstanding Issues
None
**Use Case: Update Personal Details**

**Brief Description**
This use case provides additional functionality to the system. It allows registered students to update their details as recorded in the Student Database directly. This is necessary for students reregistering, and provides additional functionality beyond the direct scope of this system.

**Primary Actor**
Student

**Other Actors**
Student Database

**Uses**
None

**Extends**
None

**Context Diagram**

**Preconditions**
The student is registered with Unisa.
The student has a student no.
The student has an email address.

**Flow of Events**
Normal Flow of Events
1. This use case begins when the student selects the StudentDetails page.
2. The system requests the students student no, name and current address.
3. The system displays the student’s current information.
4. The student updates his/her information as required.
5. The system validates the input.
6. The system persists the data.
7. The use case ends.
Alternative Flow of Events
Alternative 1: Invalid input.
1. This use case begins after step 6 of the normal flow of events.
2. The input is not valid.
3. The system displays a message to the effect, and requests new input.
4. The student updates the necessary inputs.
5. The system returns to step 6 of the normal flow of events.

Postconditions
The database is in a consistent condition, with up-to-date data regarding the relevant student.

Sequence Diagram

User Interface
This use case uses the Login.aspx, MyUnisa.aspx and StudentDetails.aspx pages.

Outstanding Issues
None
References


Gamma, E. Helm, R., Johnson, R., Vlissides, J. 1995 Design Patterns. Addison-Wesley