

Technical Manual for SimUniversity

Student: Mark Cunningham

Student Number: 95309021

Course: Computer Applications Final Year

Last updated: 30th May 1999

Email: mcunni.ca4@compapp.dcu.ie

Please refer to *User Guide and Reference to SimUniversity*

(<http://www.compapp.dcu.ie/~mcunni.ca4/SimUniversity/doc/configuration.html>) and *Abstract Description and Web Guide*

(<http://www.compapp.dcu.ie/~mcunni.ca4/SimUniversity/doc/abstract.html>) and *Functional Specification*

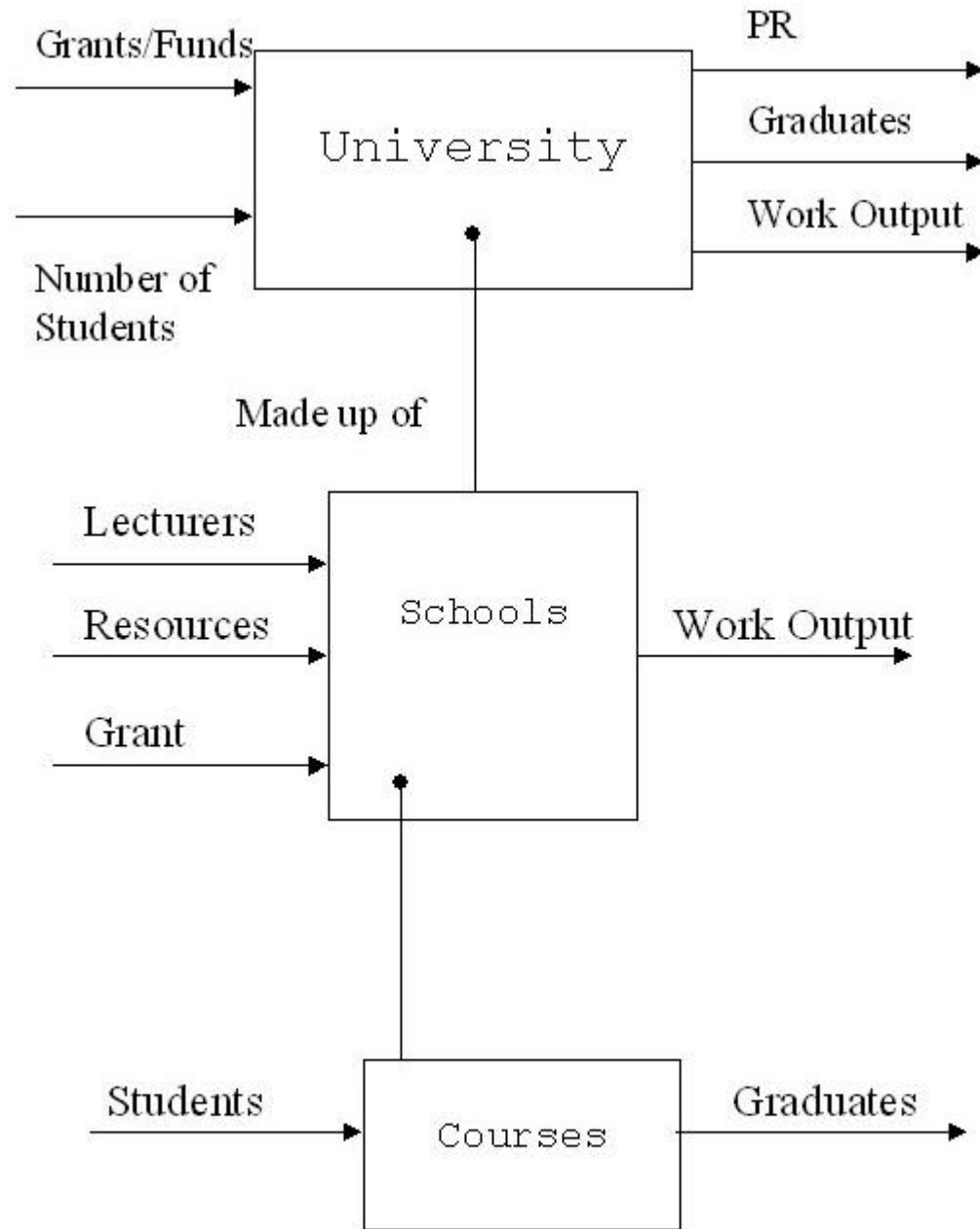
(<http://www.compapp.dcu.ie/~mcunni.ca4/funcspeg.html>) and *SimUniversity*

(<http://www.compapp.dcu.ie/~mcunni.ca4/SimUniversity/SimUniversity.html>) itself.

Overview of Project from a Design/Implementation Perspective

The program SimUniversity is made up of two distinctive parts. The Interface/Frontend and the University Model engine.

I tried to keep the University Model simple but it does appear quite complicated. In simple terms, a University is made up of Schools. A School is made up of Lecturers, Resources and, optionally, Courses. A Course is made up of Students. That's the model in a nutshell. But to expand this a bit...



We could also expand it further to say that Lecturers and Students are in themselves objects, but I think that would be overcomplicating the model needlessly.

From this I built up several series of Maths to model the interactions of each of the inputs and outputs and further refined it as I built the University Engine. The Maths is calculated in a linear fashion, where some equations are depending upon what happens in different objects so it doesn't really lend itself well to Object Oriented approach. I will approach this subject later.

A University has 3 variables, Facilities (will call it F) and Image (PRu) and Funds (G as in Grants though that's not what it means).

A School has several more. Lecturing Resources (LR), Research Resources (RR) and New Lecturers (NL), Quality of Lecturing (QL), Dedicated Lecturers to Research (DLtoR), Dedicated Lecturers to Lecturing (DLtoL), Work Output (W), Income (G'), Lecturers (L), the University takings (UG) and the number of Students in the School (NSS).

A Course then has number of Lecturers assigned to it (L'), Resources assigned to it (LR'), Number of Graduates (NG), Number of New Students (NS), Image of Course (PRc), Quality of Graduates (QG) and Difficulty (D) and Number of Students in a Course (NSC).

A lot of the equations are modified by factors. I've tried to give these intelligible names for sake of understanding but remember they are only labels; **Deprecation_Factor**, **Lecturer_Salary**, **Lecturer_Costs**, **F_Division**, **Student_Fees**, **Student_Costs**, **Marketable_Value**, **School_General_Expenses**, **Optimal_Student_Lecturer_Ratio**, **Expected_Graduate_Salary**, **General_Expenses**, **School_Cover**, **Course_Cover** and **Basic_Grant**.

These are abstract factors that effect equations.

Deprecation_Factor is the factor which decides how quickly resources devalue. (A value of 5 means it loses 1/5 every year)

F_Division is how University Facilities is divided off when including it into equations for LR and RR. (A value of 3 means that a 1/3 goes towards LR etc.)

Marketable_Value is how much of the work output of a School goes back into Income. (A value of 2 means that 1/2 goes to income)

Optimal_Student_Lecturer_Ratio affects the calculations involving DLtoL. It is meant to represent the *ideal* number of students to lecturer ratio.

These values are all pure money values.

Lecturer_Salary is the Salary of a Lecturer.

Lecturer_Costs is the Cost of equipment and it's maintain for each Lecturer.

Student_Fees is the "fee" gained from each student.

Student_Costs is the Cost to maintain equipment etc for each student..

School_General_Expenses the basic expenses incurred by each School each cycle/year.

Expected_Graduate_Salary is an abstract valuation of how much a Graduate is worth. It effects Quality of Graduates which in turn effects New Students into a Course.

General_Expenses is basic expenses incurred by the University each cycke/year.

School_Cover and **Course_Cover** is the covering/upkeep charge for each School and Course respectively.

New_School_Cost and **New_Course_Cost** is the cost to buy/create a New School and Course respectively.

Also, **Money_Allocated_to_F**, **Money_Allocated_to_LR**, **Money_Allocated_to_RR**, **Money_Allocated_to_NL**, **Course_Length**, **Student_Ceiling**, **DLtorR** (number of Lecturers dedicated to research) and **DRtoL** (number of Lecturers dedicated to Lecturing) are user defined.

The variables **number_of_courses** and **number_of_schools** and **number_of_new_schools** and **number_of_new_courses** are just keep track of by the program.

I broke down the calculations into steps so that variables that need to be calculated first or before are done.

Step 0

This is the user stage where values are allocated depending on values from the previous year. The user must decide how much funds from the University to allocated to Facilites (money value) and from there how much to additionally allocate to School's Lecturing/Research Resources (money value) and New Lecturers (money value). Also it must be decided how much to take from each school (percentage). Also new schools must be created now.

Then from a School perspective, how much of it's funds to allocated to Lecturing/Research Resources(money value) and New Lecturers (money value). How many Lecturers to dedicated to Research or Lecturing or to the two. How much Lecturing Resources to dedicated to each Course and how many students to accept next year into each course. Also new courses must be created now.

Mainly Resources are calculated here. There is no covering charge for Repairs or maintaince but the value of resources is depreicated every year by a certain factor.

University Level:

$$F = \text{Money_Allocated_to_F} - F_{i-1}/\text{Deprecation_Factor} + F_{i-1}$$

$$F \geq 0$$

School Level:

$$LR_i = \text{Money_Allocated_to_LR} - LR_{i-1}/\text{Deprecation_Factor} + LR_{i-1}$$

$$LR_i \geq 0$$

$$RR_i = \text{Money_Allocated_to_RR} - RR_{i-1}/\text{Deprecation_Factor} + RR_{i-1}$$

$$RR_i \geq 0$$

$$NL_i = \text{Money_Allocated_to_NL} / \text{Lecturer_Salary}$$

$$NL_i \geq 0$$

Note with NL there may be some left over Money which should go towards funds so =>

$$G'_i = \text{Money_Allocated_to_NL} \% \text{Lecturer_Salary}$$

Step 1

University Level:

Note that F_Division is meant to represent how Facilites are split up into Research/Resources/Other. One particular aspect of the University will have more relevance to a student then the other parts.

$$PRu_i = F_i/\text{F_Division}$$

School Level:

Note that an assumption was made here that carear life of a lecturer is 20 years

$$L_i = \sum_i^{i-20} NL$$

Note that the PR of a school is dependant on what they did last year instead of this year

$$PRs_i = W_{i-1} + PRu_{i-1}$$

Course Level:

Here we decide the New Students. The PR of a course is the attractor for students. Intially this is dependant on the PR of the school, after that then it is dependant on the QG and NG but if there is no Graduates then the PR depends on the QL and supposed QG

$$\text{year} \leq \text{Course_Length} \quad PRC_i = PRs_i$$

$$NG_{i-1} = 0 \quad PRC_i = QG_{i-1} * QL_{i-1}$$

$$PRC_i = NG_i * QG_i$$

Note Modifier here to simulate the varying of student numbers

$$NS\text{Modifier} = 1;$$

$$\text{Age_of_Course} > \text{Course_Length} + 2 \quad NS\text{Modifier} = \text{Math.sin}(\text{Age_of_Course})$$

$$NS\text{Modifier} < 0 \quad NS\text{Modifier} = -NS\text{Modifier}$$

$$NS_i = ((PRC_{i-1} * D_{i-1}) / \text{Student_Costs}) * NS\text{Modifier}$$

$$NS_i > 0$$

Implement Student Ceiling here

$$NS_i > \text{Student_Ceiling} \quad NS_i = \text{Student_Ceiling}$$

$$NG_i = NS_i - (\text{Course_Length} + 1) * \text{Product}_i^{i-1} D$$

Step 2

Course Level:

$$NSC_i = \sum_{i=0}^{\text{Course_Length}} (NS_i * \text{Product}_{j=0}^{j=i} D_j)$$

Step 2a

School Level:

$$NSS_i = \sum_{\text{for each course}} NSC_i$$

Step 3

School Level:

Note that if there is a debt, then it must be covered by UG. As in the University pays the deficit. UG is generally calculated as a percentage of G'_i before applying G'_{i-1}

$$G'_i = G'_i + (NSS_i * (\text{Student_Fees} - \text{Student_Costs})) + (W_{i-1} / \text{Marketable_Value}) - UG_i - (L_{i-1} * (\text{Lecturer_Salary} + \text{Lecturer_Costs})) - (\text{Course_Cover} * \text{Number_of_Courses}) - \text{School_General_Expenses} - (\text{Cost_New_Course} * \text{number_of_new_courses}) + G'_{i-1}$$

I initially used Lecturer to number of students ratio to represent time available to Lecturers for Research or Lecturering. When I added Dedicated Lecturering and Dedicated Research the ratios had to be modified for Work output and Quality of Lecturering

$$\begin{aligned} \text{if } NSS_i \leq 0 \quad L_NSS_ratio &= L_i - DLtoL \\ \text{if } L_i - DLtoL \leq 0 \quad L_NSS_ratio &= 0 \\ \text{if } (DLtoL * \text{Optimal_Student_Lecturer_Ratio}) \geq NSS_i \quad L_NSS_ratio &= L_i - DLtoL \\ \text{else } L_NSS_ratio &= ((L_i - DLtoR) / (NSS_i - (DLtoL * \text{Optimal_Student_Lecturer_Ratio}))) + DLtoR \end{aligned}$$

This is used to vary the work output though it can be switched off

$$\begin{aligned} W\text{Modifier} &= \text{Math.sin}(\text{Age_of_School}) \\ W\text{Modifier} < 0 \quad W\text{Modifier} &= -W\text{Modifier} \end{aligned}$$

$$W_i = (((RR_{i-1} + (F_{i-1} / F_Division)) / \text{Lecturer_Costs})) * L_NSS_ratio * W\text{Modifier};$$

$$\begin{aligned} NSS_i \leq 0) \quad L_NSS_ratio &= L_i - DLtoR \\ L_i - DLtoR \leq 0 \quad L_NSS_ratio &= 0 \\ (DLtoL * \text{Optimal_Student_Lecturer_Ratio}) \geq NSS_i \quad L_NSS_ratio &= L_i - DLtoR \\ \text{else } L_NSS_ratio &= (L_i - DLtoR) / (NSS_i - (DLtoL * \text{Optimal_Student_Lecturer_Ratio})) \end{aligned}$$

$$QL_i = L_NSS_ratio$$

Step 4

University Level:

Now calculate income to university

$$G_i = \sum_{\text{for each school}} UG_i - \text{General_Expenses} - (\text{number_of_schools} * \text{School_Cover}) - (\text{Cost_New_Schools} * \text{number_of_new_schools}) + G_{i-1}$$

Course Level:

Difficult is set between 0.1 and 1 (0.1, so that some could get through and 1 to prevent generation of more students). If no lecturers difficult is set to 0, as in NO students can get through

$$\text{NSC}_i \leq 0 \quad D_i = \text{QL}_i$$

$$\text{else } D_i = ((\text{LR}_i / \text{Student_Costs}) / \text{NSC}_i) * \text{QL}_i$$

$$D_i < 0.1 \quad D_i = 0.1$$

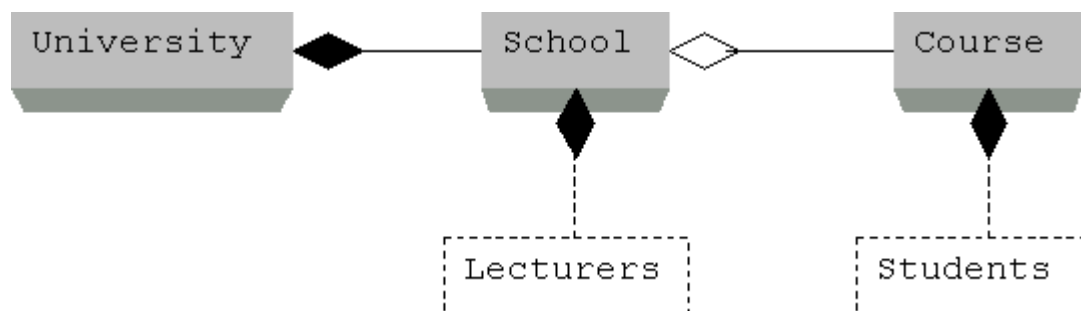
$$D_i > 1 \quad D_i = 1$$

$$L'_i \leq 0 \quad D_i = 0$$

$$\text{QG}_i = \text{QL}_i * \text{Expected_Graduate_Salary}$$

Also, important to note is that a Course is dead if after it has run for its Course_Length (or at least 2 cycles/years) and has no students or that it has no resources available to it or it has no Lecturers. A School is dead simply when it has no Lecturers and can not afford to get more. A University is dead when it has no Schools and no funds to purchase new Schools.

To further refine the model here it is in UML notation modeling the above description of the life of each of the main objects; A University is made up of and depends upon Schools. A School is made up of Lecturers and Courses. It depends upon Lecturers. A Course is made up of Students depends on Students for its survival. It is slightly simplified, but covers the aspects of the model not recognised before.

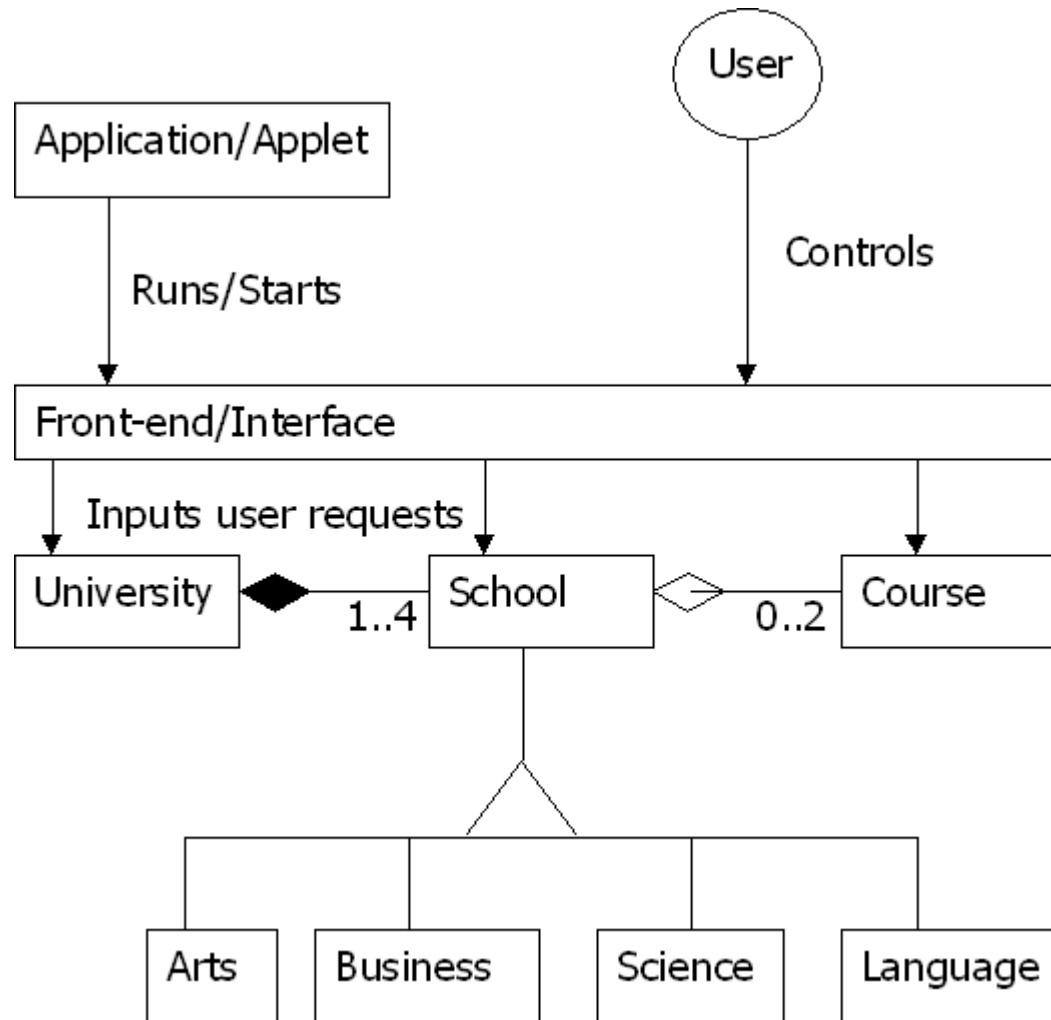


The Implementation Model

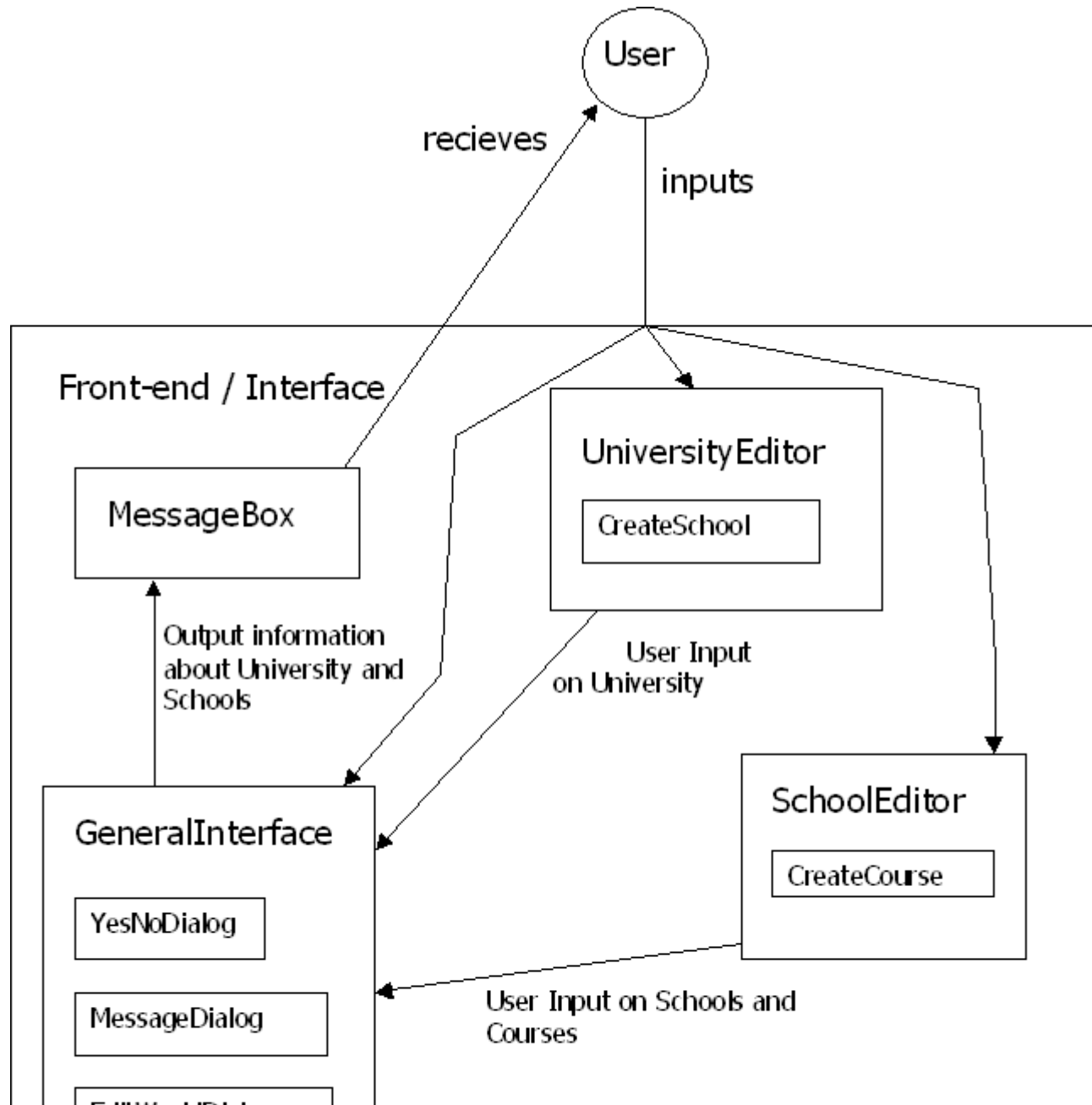
- [Course.java](#) Course.java is the most basic unit of the University Engine. It represents a single course.
- [School.java](#) School.java represents a single School in a University. Schools are important. They provide the income, the Lecturers, resources and courses that feed the University.
- [University.java](#) University.java represents the main focus of the University Engine. It is the University which must contain Schools.
- [SchoolHistory.java](#) This class is contained in SystemHistory and stores values from each cycle for each school and its courses.
- [SchoolStatisticsDisplay.java](#) This is contained in SchoolHistory and displays a Graph of different values that SchoolHistory has stored
- [SystemHistory.java](#) SystemHistory keeps a history/track of the University Engine
- [UniversityStatisticsDisplay.java](#) This is contained in SystemHistory and displays a Graph of values for the University stored in SystemHistory
- [SimUniversityApplet.java](#) This allows SimUniversity to be started from an Applet
- [SimUniversityApplication.java](#) This allows SimUniversity to be started from an Application
- [GeneralInterfaceFrame.java](#) These class represent the API/ Frontend to SimUniversity. GeneralInterface is the manager/controller of all these class
- [CreateCourseDialog.java](#)
- [CreateSchoolDialog.java](#)
- [EditStatDialog.java](#)

- [EditWorldDialog.java](#)
- [MessageBoxFrame.java](#)
- [MessageDialog.java](#)
- [UniversityEditorFrame.java](#) This allows a user to edit/setup values for the University Object
- [SchoolEditorFrame.java](#) This allows a user to edit/setup values for the School and Course Object
- [YesNoDialog.java](#)

A general overview of the final implementation of the project. The API went through 3 different phases before this was reached. The first implementation had a basic command line operation and had no flexibility towards number of courses and schools. The second implementation had no graphical frontend, only some editors.



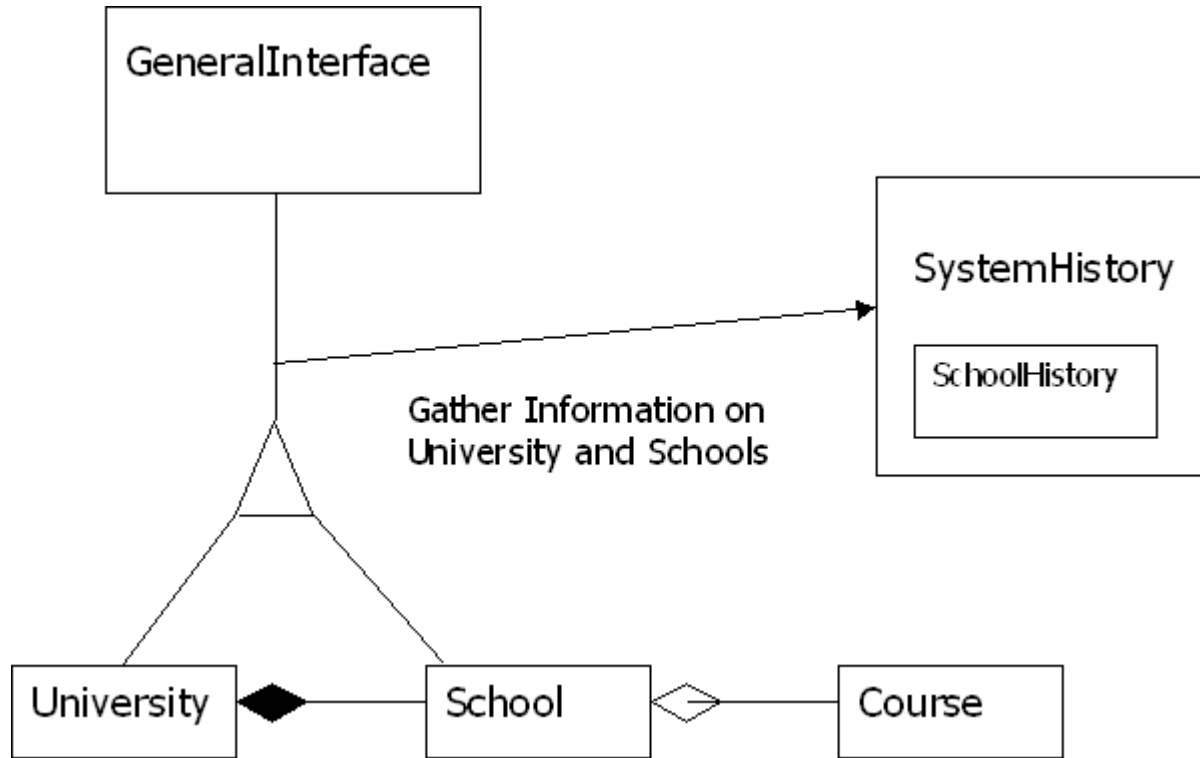
It's important to note that there is no real difference between the Arts, Business, Science and Language School, but it's important to note that there can be a distinct between the 4. There is also some other limits. For a start, there is only a maximum of four courses allowed and a maximum of 2 courses allowed.



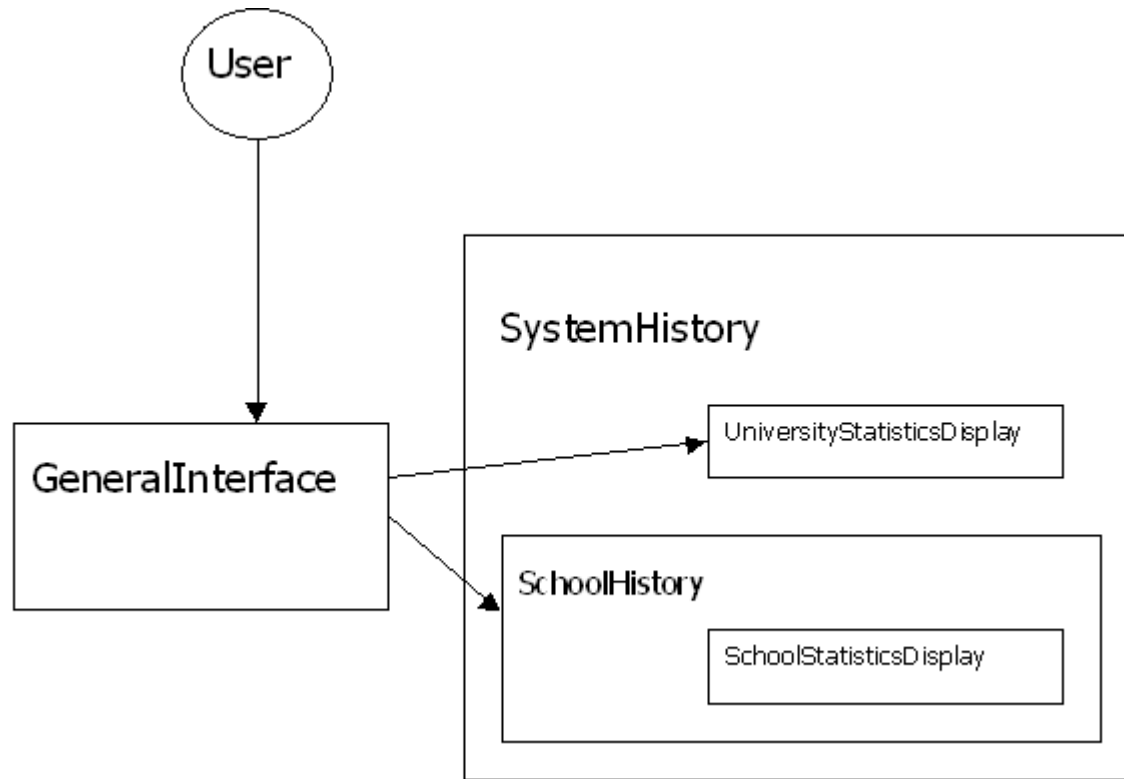
The interface is made up of 4 major components, the General Interface, the University Editor, the School Editor and the MessageBox. Please refer to the [*User Guide and Reference for SimUniversity*](#) for details on how to use these interfaces and their appearances. The University Editor and School Editor allow the user to control the various elements of the University and Schools and using the CreateSchool/CreateCourse dialogs they can also allow a user to create Schools and Courses. The Message box is the main way of displaying data to the user on the current University and School. The University and School Editors hold all the necessary user inputs. When the user wishes to update, the GeneralInteface takes those values, inputs them to the University engine and then updates the University engine.

I used Java version 1.1.7a API to create the frontend. I found it frustrating and time consuming, complicating some the overall structure needlessly.

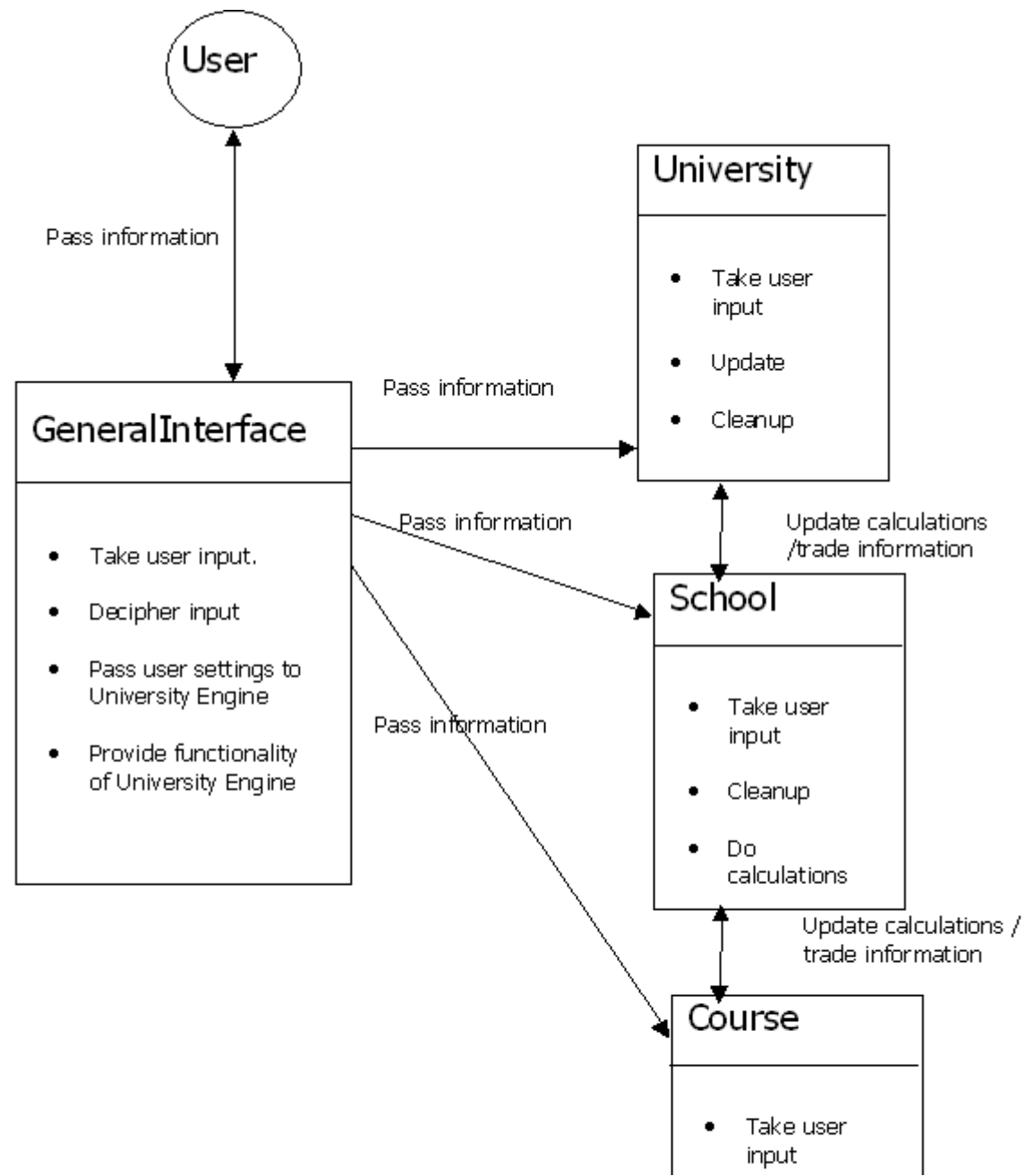
The GeneralInteface.EditWorldDialog allows the user to edit all the factors of the University Engine, thereby customising the Simulation.



The SystemHistory class collects the latest information about the University, School and Courses and stores it. It also provides a Graphical frontend to show user the history of the school in a graphically form. See [User Guide and Reference for SimUniversity](#) for details on how to use it.



A brief overview of interactions:



Please refer to the [User Guide and Reference](#) for an understanding of the Interface/Frontend while running.

The Devolpment History

Intially I create a simple model. It contained only a University and School. From that devolped a list of variables and their dependences on each other. This went through several iterations.

After that, a system of maths was generated and modified until it seemed satisficator. Several test runs were done to work out, in what order calculations were to be done and to avoid *loops* in the calculations. A simple basic implementation was done in C using typed commands. After this I was forced to redesign the model to take in that a Course should be treated as an individual object. I also considered that Java would lend itself well to the idea of having multiple schools and courses.

Another implementation, but this time in Java, with the maths again being refined.

Devolpment of first interface. Redesign of interface and implementation, refining of maths, specifically to do with allocation of funds and tracking. Devolpment and design of Pictures using Lightwave, Paintshop Pro and The Gimp.

Added SystemHistory to provide a history and graphical functions.

Completed Interface and University Engine.

Documentation. (The Gimp/Paintshop Pro were used to create Screenshots in the Installation Guide and User Guide, Microsoft word was used to create OO diagrams and Model Diagrams for this document)
