The Design of a Web-based Process Simulator

Alistair Marshall
0453136

Chemical Engineering
School of Engineering and Electronics
University of Edinburgh

April 2009

Abstract

During the early stages of process engineering design, one of the key tasks that needs to be performed is the development of a mass balance of component flowrates and its distribution to all members of the design team. This project has developed a web based solution which aims to allow users to quickly develop a process layout and calculate an initial mass balance.

Whilst this is the first release of the application, it has already shown that it could be useful in a number of situations. With further development, this application has the potential to increase its scope and enter other markets.

All the code and the lab book associated with this project can be found on the attached CD and is also available at `http://www.thatscottishengineer.co.uk/code/impress/`

Acknowledgements

I would like to thank the following people for their assistance throughout this project:

- Professor Jack Ponton, my supervisor, for his support and guidance during the project.
- Dr Lev Sarkisov, my second supervisor, for providing an objective review during several stages of the design.
- Eddison Ruswa, Kevin McMullan, Jim Ross and James Adley for providing valuable feedback as beta testers of the application.
- Michael Gordon and the other members of the school's computing services team for providing server access to host the application
- My friends and family for their sustained interest, support and comments throughout the project.

Contents

SECTION 1

Introduction

During the early stages of process engineering design, one of the key tasks that needs to be performed is the development of a mass balance and its distribution to all members of the design team. Only after the approximate flowrates have been obtained, can design decisions regarding alternative process methods be compared[1].

Once the detailed design of process units has begun, changes may need to be made to the mass balance which could then affect the design of other process units. This procedure would continue iteratively until the changes being made have stabilised and a final mass balance has been produced. During this iterative design period, it is important that all members of the team have access to the most up-to-date version of the mass balance.

This project has developed a solution which aims to allow users to quickly develop a process layout and calculate an initial mass balance. This can then be tailored as more details of the design process continues and provide a simple method that allows users within a group to obtain the most up-to-date version.

1.1 Existing Tools

There are a number of products currently available which can be used to obtain a mass balance. Full simulation programs are capable of carrying out rigorous simultaneous heat and material balances, and preliminary equipment design[2].

Examples of simulation packages include ASPEN Plus, HYSYS, PRO/II, CHEMCAD[2]. To use a one of these packages requires that it is installed locally on a computer.

There are several methods for sharing simulations, from simply transferring copies of the simulations (using email, or other file transfer solutions) or using networked storage. Network storage has the advantage that keeps all users working with the same files however requires more set-up and configuration. Files shared over network storage require that files are locked when one user is accessing them, ensuring no other users can edit at the same time.

At this early stage of a project, the use of a full simulation package is often not justified and a simple material balance program is more suitable[2]. Alternatively, a designer may perform the calculations manually using a spreadsheet or some other general purpose program. Manual calculations are difficult and time consuming, especially if recycles are involved.

1.2 Web Applications

A web application or webapp is an application accessed via a web browser over a network such as the Internet or an Intranet[3]. Services that were traditionally provided by desktop applications are now available through the web browser. Such applications include email clients[4,5], photo collection management[6,7], personal journals*[8,9], calendars[10], point-of-sale terminals[11], word processing, spreadsheets and presentations[12] are all available as web applications.

There are numerous advantages to using a web application over traditional desktop programs. A user can access the application from any computer with a connection to the network without requiring any local software installation. The user's data is stored on a central server, removing the need for the user to be concerned with transferring data between workstations or other users[13]. Given the web based nature of the application, it is architecture-independent and may be accessed from any current operating system and may be used on cheaper low-end computers such as netbooks or smart phones.

One major disadvantage of delivering services over the Internet is that the application may only be used whilst connected to the Internet. This is not an issue for many desktops due to the high availability of always-on, fast Internet connections. When used within an office or house, most laptops will have access to the Internet. However, when traveling, Internet access is less accessible.

*These have become replaced by blogs

If access to the application is critical, there are a number of ways to work around this problem. One solution is to purchase a wireless modem which would allow the user to access the Internet over the mobile phone networks[14]. Another solution is to design the web application to run locally using client-side scripting and data storage such as JavaScript and Google Gears[15].

Another concern with web applications is data security. Firms may have issues with transferring potentially proprietary data over the Internet. This can be improved by transferring the data using encrypted HTTPS protocol rather that the standard unencrypted HTTP[16]. Furthermore, it may be possible to deploy the application on an internal, company-owned server on an Intranet, removing the liability of storing the data on an external server.

SECTION 2

Technology Review

A web application framework is a collection of tools and scripts that are designed to support the development of web applications by providing a base to build upon[17]. The framework aims to reduce the overheads associated with web development. Many frameworks provide libraries for database access, templating frameworks and promote code reuse. Frameworks often promote the use of design architectures and paradigms such as the model-view-controller, discussed in section 3.1.

The development of the main application will make use of one of these frameworks. The project needs to be completed within six months and so time can not be wasted on learning the framework. Similarly it is hoped that this project can be continued by other people and so the final application needs to encourage new developer participation. The key factors used for selecting a framework include: a shallow learning curve, ample concise documentation, support and quick development times.

Another key factor will be the ease of deployment. The final application should be simple to deploy onto most web servers without complex configuration.

2.1 Overview of Available Frameworks

Many frameworks were considered before a final selection was made. Due to the the high number of frameworks available, only a selection of the more popular were considered—those that were recommended by colleagues, friends and recent reviews.

After a short evaluation, frameworks with potential were then tested by producing a simple web application. The test application chosen was an on-line project planning tool

capable of listing tasks and identifying the critical path. This application was used to examine how simple it was to learn and implement features.

The application consists of activities. Each activity has a name, description and a duration. Activities also need to have some method of detailing which other activities it succeeds and is dependent on. Having collected this information, the application can calculate the earliest and latest start times which would not affect the critical path. This test application was chosen because it contains a number of key requirements that the final application would also have.

One of the key requirements that the final application has to deliver is the storage of the topology network of process unit operations. The storage of tasks and the details of which tasks depend on other tasks will be used to test this requirement. Another requirement of the final application is the ability to perform calculations based on the data stored in the system. This ability is tested by calculating the critical path.

Frameworks are available in a number of programing languages. The frameworks language will also affect the choice of framework.

2.1.1 PHP

Developed in 1995 by Rasmus Lerdorf, PHP is a scripting language designed for producing dynamic web pages[18]. PHP is widely available on most web servers and used by many high profile websites including wikipedia[19], flickr[20] and the user-facing portion of Facebook[21].

PHP has a similar syntax to that of C, perl and Java. However, PHP will only parse code that is between PHP delimiters the most common of which are `<?php` and `?>`. Any text not within PHP delimiters will be sent directly to the output. Delimiters are used to help split up PHP code from other code such as HTML[22].

Symfony

A PHP based web framework, Symfony makes use of the model-view-controller architecture.[23] Symfony also makes use of many existing open source PHP projects as part of the framework.

Symfony makes heavy use of convention over configuration, a software practice that aims to reduce the quantity of code the developer has to write by assuming conventional methods and only specifying the unconventional parts of the application.[23] Whilst this

design paradigm is useful for an expert developer, it has a higher initial learning curve making it more difficult to get started.

Similarly symfony has a high number of files many of which contain duplicate code. There are several tools which help produce and update these files supplied with the application however it was difficult to learn which tools to use and where to look for the correct code. During the creation of the test application, the maze of extra files and the difficulty in configuring the non-symmetric links between units meant that this framework was discarded.

2.1.2 Python

Python is an open source, general-purpose, high-level programming language[24]. Designed by Guido van Rossum and first released in 1991, it is used by NASA[25], Google[26] and CERN[27]. Python's design philosophy emphasizes code readability and makes use of significant white-space[28].

With most programming languages, white-space such as spaces, tabs and carriage returns are ignored by the compiler. With a program that includes significant white-space, white space characters are used as part of the code. In the case of Python, the indentation of code is used to define code blocks such as if-statements and loops, rather than make use of brackets.

Example

Java/C style:

```
list_of_names = ['Fred','Bert','Alan']
for (i=0,i=length(list_of_names),i++){
    name = list_of_names[i]
    print 'hello ',name
    if (name == 'Bert'){
        print 'You are a winner!'
    }
}
```

Python:

```
list_of_names = ['Fred','Bert','Alan']
for name in list_of_names:
    print 'hello', name
    if (name == 'Bert'):
        print 'You are a winner!'
```

This use of indents to determine blocks of code can help reduce errors due to missing brackets and also enforces a consistent indentation style making the code more readable.

Running Python-based web applications is not available on all web servers[29]. Apache, the most popular webserver[30], is able to serve dynamic web pages using Python if one of two modules is installed[29].

web2py

Designed as a teaching aid at DePaul University, this framework aims to have a shallow learning curve and provides many security features by default[31]. During the testing of this framework, an attempt was made to write a small test web application. It was found that whilst the learning curve for developing very simple projects was shallow, more complex and non-standard applications were much more difficult. The ability to create non-symmetric links between units was found to be complicated and help was difficult to obtain. Due to the relatively small community and lack of documentation, this framework was rejected.

Google App Engine

Whilst not strictly a framework in its own right, the Google App Engine (GAE) is a platform which can be used to deploy Python-based web applications[32]. GAE provides a free service up to a certain level of usage, which can be used to start a service. GAE aims to take the hassle out of deploying and scaling web applications.

Google App Engine is supplied with an old version of Django and has the ability to deploy most Python based frameworks on it after some modifications. There are restrictions to the way the database can store and retrieve data. Models created with other frameworks have to be heavily modified to ensure they would work[33].

Though applications created for the Google App Engine are capable of being deployed on other servers, this requires more modification, making deployment of smaller scale versions on Intranets more difficult[34]. This framework was rejected due to these restrictions and the extra configuration requirements

2.1.3 Other Frameworks

A number of other frameworks from other languages were considered.

Ruby on Rails

Ruby on Rails (RoR), first released in 2004, is a web development framework written in the ruby programming language[35]. Ruby on Rails follows the convention-over-configuration design paradigm, uses the model-view-controller architecture and attempts to aid rapid development of web application[35].

This framework was disregarded as it did not provide any substantial advantages over other frameworks written in languages familiar to the author and would have required significant resources to obtain the same proficiency.

TiddlyWiki

TiddlyWiki is non-linear personal notebook written entirely within HTML, JavaScript and CSS. This web page produces an editable wiki capable of running off-line[36]. There are many plugins and applications built on TiddlyWiki proving that it could be extended beyond a simple notebook[37]. TiddlyWiki was designed to be used by a single user. This would limit the applicability of a final application to teams and was therefore discarded.

2.2 Django Web Framework

Django, a Python based open source web framework, was designed to ease the creation of complex database driven websites. Originally developed to manage newspaper sites for World Online, it was released under an open source licence in 2005[38].

2.2.1 Overview

The design principals behind Django have developed with regard to the newspaper origins. Specifically the need to produce high quality, secure and robust sites for heavy public use within hours of a story breaking. Compared to development time, hardware is cheap and so the framework has been designed to be as simple and quick for the developer as possible with the intention of adding more computing power when required[38].

2.2.2 Framework Features

The Django framework has a number of built-in features which make development simpler including:

An object-relational mapper (ORM) which mediates between a database and data models. Models are defined as Python classes and allow the rest of the program to interact with the data in a simple manner without involving the database code. This abstraction of database and code handling has the advantage that it allows multiple database back ends allowing the user to switch which database type is used.

A URL dispatcher for translating incoming HTTP requests to views. This converts URLs such as '`http://example.com/process/+new/`' into calls for the appropriate sections of code - in this case the function to create a new process.

Template system allowing the designer to develop HTML pages and insert tags to be replaced with data when the template is rendered.

A form handling system which can automatically generate HTML forms and validate submitted forms obtaining suitable values for storage in the database. This automatic generation and validation system removes a lot of effort in terms of writing code and preventing security fixes.

Other features include a flexible caching framework which can be used to reduce loads on servers and an internationalisation system, allowing the translation of the site into many other languages. Whilst these last two features may not be used with the application produced during this project, they may be useful if the application is developed further later.

2.2.3 Documentation and Community

Django has an extensive documentation system covering almost all aspects of the framework[39]. There are also several books written about the framework including one free online book that gives a thorough introduction to Django[40]. During development of the small test web application, it was found that many problematic issues had already been encountered by other people who had documented their problem and solution through the use of blogs, mailing lists and other online sources.

2.2.4 Pinax Project

Pinax is a collection of reusable applications for the Django web framework that are released under an open source licence and aim to provide many of the things that sites have in common. This allows web developers to work on what makes their sites different[41].

Applications included in Pinax provide support for openid, email verification, password management, site announcements, a notification framework, user-to-user messaging, friend invitation, interest groups (called tribes), projects with basic task and issue management, threaded discussions, wikis with multiple markup support, blogging, bookmarks and tagging.

Most of the development is focused around a simple social networking site and therefore many of the applications may not be relevant to the final application.

2.2.5 Conclusion

Django provides a comprehensive framework with strong community support. Django struck a nice balance between ease of configuration without over complicating the platform. The Pinax project provides a selection of applications which could be easily integrated into a project providing a more useful final application.

SECTION 3

Design of Application

Aims of the system are to provide an easy-to-use and easy-to-learn interface allowing the user to quickly obtain a mass balance. Accuracy is not the main concern at this stage of the project and this should be used to get an idea of the flowrates through the process.

Wherever possible, the system should supply sensible default values to minimise user interaction when filling in forms. Similarly, if possible, the system should attempt to look up physical properties and allow the user to edit when finished.

Multiple-user interaction should also be considered during the program's design to ensure that users can collaborate easily without causing errors.

3.1 Model-View-Controller Architecture

Whilst working on Smalltalk in 1979, Tygve Reenskuge first described the Model-View-Controller (MVC), an architectural pattern for use in software engineering.[42] This architecture is used to separate the 'layers' of the program isolating the business logic from the user interface.

Given that, in programming, data is the stored unit and information is data with its context: The model refers to a representation of the information used by the application. The view normally refers to the representation of data presented to the user and handles the user interface. The controller is used to process and respond to events.

Django doesn't follow the traditional MVC method, however it still makes use of separating the code into separate layers. The Django architecture has been described as Model-Template-View (MTV). In this architecture, the model is similar to that described

in the MVC architecture[38]. A model defines a Python class which can have functions and classes within it used to store data. A view is used to describe what information is presented to the user. A view is a Python function which takes a request object and returns a HTTP response. The template is to describe how the information is presented. A template is usually a partial HTML file with tags used as place holders for data specified by the view. The role of the controller is taken up by a combination of the view and the Django framework itself.

3.2 System Units

It was decided that the system would work with SI units and store all data in the database using SI units. This should make internal calculations simpler (without having to worry about units internally).

To improve the user interface experience, the user should have the option to work with alternative units and have the application convert them during the data validation step. It would also be useful to have the data converted to the users preferred units during the rendering stage.

3.3 Equation Solver

The application aims to produce a quick calculation of a process flowrate during the early stages of design. During these early stages, there is no need to generate a rigorous set of calculations to generate accurate flowrates[1,2]. It was decided to restrict this application to linear equations and make use of a Gaussian solver. This produces a more robust system than dealing with an iterative solution generator.

Once the flowsheet is complete, a simulation can be run by collecting all the equations from all the units and solving them simultaneously. This then produces the flowrates of all components in all streams which can be inserted back into the database.

3.3.1 Form of Equations

Each variable represents the molar flowrate (in moles per second) of a particular component through a particular stream. Every unit must provide one equation per component per product stream. One equation may be provided by a material balance such as mixers and splitters. An equation takes the form of a three element tuple.

The first element is a list of the variables used in the equation. A variable is the molar flowrate of a particular component through a particular stream. This is represented as a two-element tuple containing the stream id and the component id.

The second element is a list of the coefficients for each of the variables defined in the first element.

The last element is a constant. In most cases this constant will be zero with the exception of feed streams where this constant contains the feed flowrate.

3.3.2 Solver Algorithm

The Gaussian solver was originally written by Ponton and modified by Andrews. The algorithm was first written in Fortran and converted to JavaScript and used in several linear equation solvers[43].

For this project the algorithm was converted from JavaScript to Python. Whilst converting, the algorithm was improved by automatically calculating the number of equations, removing the need to state the number of equations explicitly.

3.4 Authorisations

During the design of the process, security concerns were raised. If all users are accessing the same server to view their processes, and if multi-user design is required, how do you go about deciding who can access which views.

Django provides a built-in authentication system which can be used to determine who a user is. It does not provide any help regarding who can view/edit a process, therefore an authorisation system was built to determine rights access to different parts of a process.

For each process two levels of access were obtained:

View The user has the right to view the process, download an excel spreadsheet, copy the process and if the process simulation is complete, view the flowrates.

Edit The user has all rights contained with View plus may create, remove and edit any models and may run the simulation.

Every view in the process module checks first if the user has the right to view the appropriate process and, for views which require editing any of the process units, whether the user has the right to edit a process.

For this demonstration system, it was decided to keep user rights simple, allowing more expansion later if required. Each process has a field containing "team members" who are users with edit access. Each process also has a boolean field, "publicly viewable", used to determine if anyone can view the process. Using this method, there is no way to allow one specific user to view but not edit a process. It would be desirable to integrate groups and be able to associate processes with groups. These groups could represent teams or companies.

Two helper functions were written to determine the user's rights, passed to the process and user in question. These two functions could be extended to offer more complex rights management, should the need arise.

SECTION 4

Process Engineering Objects

There are many models used by the program as defined in Python classes. There are 4 primary models: Process, Components, Units and Streams. All components and units are associated with a process and all units are connected using streams.

Extra data is often stored using complementary models associated with one of the main models.

Data is stored in the database using fields. There are a number of types of fields:

Foreign key Every instance of an object has a primary key to quickly find it in the database. A foreign key field stores the primary key of another object.

Many-to-many field Stores multiple foreign keys for a particular object type.

Text field A text field holds a short section of text.

Slug field A slug field is an text field consisting of only alphanumeric characters, hyphens and underscores. A slug field is used when a field will make up part of the URL.

Float field A float field consists of a real number.

Boolean field A boolean field is a two state value and can be either true or false.

4.1 Process

The basic process object is used to store options about the overall process and provide a central object which all other objects can be associated with.

A process stores the following data in the database:

Name Slug field used to describe the component.

Description A text field used to hold a short description of the process.

Team Members A many-to-many field containing other users that may edit the process.

Public A boolean field to determine whether users who are not team members can view the process.

Simulated A boolean field used to store the status of the process simulation. This field is not editable by the user.

4.2 Components

The component model stores data for an individual chemical component. A chemical component is an element or compound such as 'oxygen' or 'methane' and not a mixture like 'air' or 'kerosene'.

The component stores the following data in the database:

Process A foreign key relating to the process to which it belongs

Name Slug field used to describe the component.

CAS Registry Number A text field containing the CAS Registry Number.

Formula A text field containing the chemical formula. When displayed, all numbers will be subscript.

Molecular Weight A float field containing the molecular weight of the component in grams/mole. This value must be non-zero. If it is not known, a value of 1 should be used.

Boiling Point A float field containing the boiling point of the component in Kelvin at 1 atmosphere.

Critical Temperature A float field containing the components critical temperature in Kelvin.

Critical Pressure A float field containing the critical pressure of the component in Pascals.

Other names A text description field for holding other names.

With the current implementation, the last three fields are not used, however they may be incorporated at a later time.

4.3 Units

A unit is a node in the process. Each unit represents an item of equipment performing a unit operation. Units are used to generate the linear equations which are then solved to obtain the flowrates. Each unit must produce one equation for each component for each of the units exit streams. One of these equations may be a material balance where the unit has no component generation or removal such as mixers, splitters and separators.

4.3.1 Generic Unit

The basic unit from which all other units are subclasses. A subclass unit will store all the following fields and may contain extra fields.

The unit stores the following information in the database:

Process A foreign key relating to the process of which it belongs.
Name Slug field used to describe the component.
Description A text field used to hold a short description of the unit.
Feeds A many to many field of feed units to this unit. This field should not be edited directly, instead it should be edited using a stream object.

A unit also has a number of functions associated with it. These are often modified by the sub-classed units.

Test flowsheet When called, this function counts the number of feed and product units and ensures this is a legal number for the particular unit. In the case of a generic unit, it must have between 1 and 10 feeds and again between 1 and 10 products. If the unit does not have the correct number of feed or product streams, then an error message is returned letting the user know what is wrong with the process. Otherwise the function returns false. This function is usually called on all units in a process to test the overall process flow sheet.
Get equations When called, this function produces a list of equations as defined is section 3.3.1. This is usually called for every unit in a process to get a list of all equations used in the process.
Get related This is an internal function used to get the sub-classed form of the unit if it has one.

4.3.2 Feed

A feed unit is used to insert material into the process. The user must specify a flowrate of each component fed into the process. This unit does not have any extra fields. The component flowrates are stored in a complementary model 'FeedFlowrate'. A feed unit must have zero feed streams and exactly one product streams.

The complementary model 'FeedFlowrate' stores the foreign key of the feed unit which it is associated with, the foreign key of component it refers to and the flowrate of component in moles per second.

4.3.3 Product

A product unit is used to remove material from the process. This unit does not have any extra fields. A feed unit must have one feed stream and zero product streams. There are no product streams from this unit, therefore this unit does not generate any equations.

4.3.4 Mixer

A mixer unit is used to combine two streams. This unit does not have any extra fields. A mixer unit must have two feed streams and exactly one product stream. A mixer does not generate or remove any material and therefore forms a material balance. The mixer has only one product stream therefore no more equations need to be specified.

4.3.5 Reactor

A reactor unit is used to simulate a chemical reaction in the process. This unit does not deal with complex non-linear systems. Instead the equations are based on a per pass conversion of a key component. To specify the reaction stoichiometry, a complementary model 'Stoichiometry' is used. A reactor must have exactly one feed stream and exactly one product stream.

The reactor unit has two extra fields:

key component The foreign key of the key component.

conversion Float field representing the fraction of key component entering the reactor that is consumed.

The complementary model Stoichiometry stores the foreign key of the reactor unit which it is associated with, the foreign key of the component it refers to and the stoichiometric coefficient. This is positive for products and negative for reactants.

Eg. In the equation:

$$CH_4 + 2O_2 \rightarrow CO_2 + 2H_2O$$

The stoichiometric coefficients are CH_4: -1, O_2: -2, CO_2: 1, H_2O: 2

4.3.6 Splitter

A splitter unit is used to split a stream without any component separation. Both exit streams will have the same chemical composition. A splitter does not generate or remove any material and therefore forms a material balance. The user selects a fraction of flow to be directed to the key stream.

This unit stores two extra fields:

key stream The foreign key of the key stream.
fraction A float field representing the fraction of flow diverted into the key stream.

A splitter must have exactly one feed stream and exactly two product streams.

4.3.7 Arbitrary Separator

An arbitrary separator unit is used to split a stream into two separate streams and a specified fraction of each component is recovered in the key stream. A separator does not generate or remove any material and therefore forms a material balance. The recovery of each component is defined by the user and stored in a complementary model SplitFraction. An arbitrary separator unit must have exactly one feed stream and exactly two product streams.

This unit stores one extra field:

key stream The foreign key of the key stream.

The complementary model 'SplitFraction' stores the foreign key of the separator unit which it is associated with, the foreign key of component it refers to and the fraction of component recovered in the key stream.

4.3.8 Relative Volatility Separator

A relative volatility separator, also known as an alpha separator, splits a stream into two separate streams based on user-specified component relative volatilities. A separator does not generate or remove any material and therefore forms a material balance. The recovery for a key component is set by the user and the recoveries of all other components are calculated using the number of stages and the component's relative volatility. A relative volatility separator unit must have exactly one feed stream and exactly two product streams. The specified relative volatilities are stored in a complementary model 'RelativeVolatility'.

This unit stores four extra fields:

key stream The foreign key of the key stream.
key component The foreign key of the key component.
recovery A float field representing the fraction of of the key component recovered by the key stream.
stages The number of stages used to separate the components. This should be set to one for a single stage flash operation.

The complementary model 'RelativeVolatility' stores the foreign key of the separator unit which it is associated with, the foreign key of component it refers to and the relative volatility of component in the separator.

4.3.9 Ideal VLE Separator

An ideal VLE separator splits a stream into two separate streams and the composition of the streams is calculated assuming an ideal vapour–liquid equilibrium behaviour. A separator does not generate or remove any material and therefore forms a material balance. The user specifies the recovery of a key component and the number of stages. The recoveries of all other components are calculated using their respective boiling points. All components must have a boiling point specified for this unit to work. An ideal VLE separator unit must have exactly one feed stream and exactly two product streams.

This unit stores four extra fields:

key stream The foreign key of the key stream.
key component The foreign key of the key component.

recovery A float field representing the fraction of of the key component recovered by the key stream.

stages The number of stages used to separate the components. This should be set to one for a single stage flash operation.

4.4 Streams

Streams are used to link units together.

The stream object stores the following information in the database:

Source The foreign key of the source unit.
Destination The foreign key of the destination unit.
Name Optional text box used to label the stream.
Temperature A float field intended to store the temperature of the stream in Kelvin.
Pressure A float field intended to store the pressure of the stream in Pascals.

The Temperature and Pressure fields are currently not used and are hidden from the user.

There are two calculated properties of the stream:

molar flow Calculates and returns the total molar flowrate through the stream in mol/s
mass flow Calculates and returns the total mass flowrate through the stream in kg/s

The flowrate for each component in a stream is stored in a complementary model - 'StreamFlow'.

The complementary model 'StreamFlow' stores the foreign key of the stream it is associated with, the foreign key of component it refers to and the molar flowrate of the component in mol/s. The user is not able to set the flowrate directly, this is set when the simulation is run.

The 'StreamFlow' model also has a number of calculated properties:

Mass flow Calculates and returns the mass flow of the component using the molar flow and the molecular weight.
Mass fraction calculates and returns the mass fraction of the component in the stream.
Mole fraction calculates and returns the mole fraction of the component in the stream.

SECTION 5

User Interface

The user interface is the method which allows the human to interact with the application. The design of a good user interface should minimise the amount of effort required by the user to interact with the application to obtain the desired result[44,45]. The final application makes use of several features to aid the user.

5.1 Browser Compatibility

Throughout development, browser compatibility has remained a high priority. It was intended to be compatible with as many browsers as possible. It was decided that the default browser for the main operating systems, Internet Explorer (Windows), Firefox (Linux*) Safari (Mac) should be supported. Together, these browsers comprise more than 97% of Internet users[47].

Several other browsers were identified as useful, however not as necessary due to the availability of other supported browsers for their platforms and their relatively low usage. Amongst tese browsers are Opera, Google Chrome, Konqueror, and Elinks (text-based browser)

Whilst it is not expected that users would use this program with a text-based browser, this provides a useful view of the site from an accessibility perspective. Blind users would use the site with a screen reader which would read the site in the same way a text-based browser would display the site.

The application has been tested on the following browsers:

*There are several different distributions of Linux. Ubuntu, the most popular[46], will be used in this case.

- Internet explorer (versions 7 & 8)
- Firefox
- Safari
- Opera
- Google Chrome
- Elinks
- Safari on the iPhone
- Android (Google phone)

There are issues with Internet Explorer 6 due to unsupported transparency in PNG images. Excess effort to fix this issue was not considered necessary, considering users can upgrade or install a different browser.

The ability to run the application on the iPhone and Android smart phones is an example of running the application on low-powered devices. Similarly, it provides a method of allowing users to access their most up-to-date data from any location where their mobile phone can be used.

This application has been written to produce XHTML 1.1 strict pages. These are a set of guidelines set out by the World Wide Web Consortium (W3C). Valid XHTML code assures consistency in document code, which in turn eases processing, but does not ensure consistent rendering by browsers[48].

5.2 NIST Interface

The National Institute of Standards and Technology (NIST), is an American federal agency whose aim is to promote U.S. innovation and industrial competitiveness by advancing measurement science, standards, and technology[49]. One of the services offered is the NIST Chemistry WebBook containing thermochemical data for over 7000 organic and small inorganic compounds[50].

When adding new components into a process the application requests a number of component properties such as formula, boiling point, critical temperature, et cetera. In an attempt to minimise the user interaction, the application offers a look up feature when creating components. When the user enters a component name, there is an option to search for that component in the NIST database. The user can enter one of the components names for example the user can enter “sulphur dioxide” (British), “sulfur dioxide” (American) or “Schwefeldioxyd” (German) and the system will fill in the rest of the form with the chemical properties of “sulfur-dioxide”. The system works by browsing

the American NIST website and downloading the component's web page. Once the page has been downloaded, the application extracts the properties using several regular expressions to produce the required data. This is then used to populate the form before returning the data to the user.

The user also has the option of entering the CAS number or the chemical formula; however there is a chance that more than one component will be found. If more than one component is found, the user is offered to look up the NIST website themselves and find the component they require. After they have found the correct component, they can copy and paste the CAS number back into the original form and re-run the look up. The rest of the components properties will then be completed in the form.

There are occasions when the NIST database does not have all the required information, such as when looking up ammonia. In this case, most of the information is filled in, however the boiling point is not found. In this case, the user can manually enter the missing information. It is conceivable that this feature can be extended to include the addition of different sources and databases.

5.3 Process Layout

During the early stages of development, it was noted that whilst the user could create a process, it was difficult to get an overall view of how the process units fitted together. Graphviz, a graph drawing program, was used to produce a graphical representation linking all the units together. The graphics produced by the program are not ideal for a process flow diagram, the diagram produces straight lines going directly from source to destination or curved lines as shown in figure 5.1. Ideally the diagrams would consist of a series of orthographic (horizontal and vertical) lines as shown in the block flow diagram figure 5.2, however to produce this image would require either complete reimplementation or heavy modification of the original program and was not considered suitable for this project.

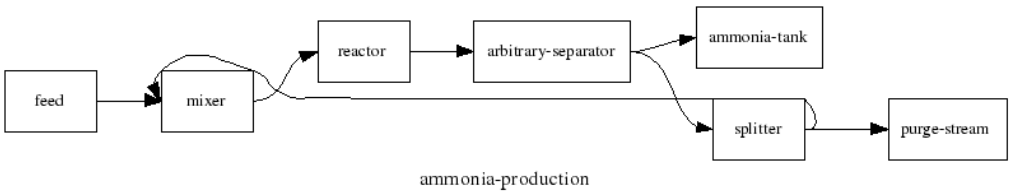

Figure 5.1: Automaticly generated layout for example ammonia process

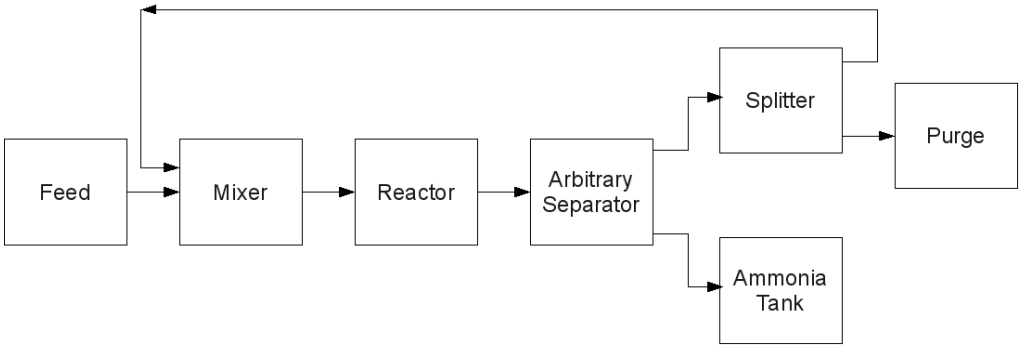

Figure 5.2: Block flow diagram of example ammonia process

The module is implemented using an external library pygraphviz, which provides a Python API for graphviz. The original implementation of this feature provided a simple graphic which was used to display the layout, however after usability testing as discussed in section 5.7 it was found that the users clicked a unit on the layout image and expected to be taken to the unit's page. It was found that graphviz has the ability to include hyperlinks and use them in image maps which can be embedded in the web page.

If the pygraphviz module is not installed, this feature is disabled and the rest of the application should operate as normal.

5.4 Export to Excel

One of the features provided by the application is the ability to export all the component flowrates to an excel spreadsheet after the simulation has been completed. This feature provides a simple method to allow the user to process the information and represent it in the manner the user wishes.

5.4.1 Why Excel

Excel is a proprietary spreadsheet program produced by Microsoft. Excel documents (.xls) are not an open format; however, the decision was made to export using the excel format due to the high market dominance of excel and the familiarity of excel files[51]. Another format considered was a comma separated variable (.csv) file however there is no facility to store formulas using CSVs.

Concern was raised over the proprietary nature of the excel format however the specification is freely downloadable and can be implemented under the open specification promise patent licensing[52].

5.4.2 Implementation

The export to excel function was implemented using the PyExcelerator module. PyExcelerator is an open source Python library allowing other Python programs to read and write excel files including formatting and formulas. If the PyExcelerator module is not installed, this feature is disabled and the rest of the application should operate as normal.

5.5 Flexible Inputs

All flowrates are stored as moles per second; however, it is unlikely that a process design team is likely to wish to use these units. The application can allow the user to input flowrates in a number of units. If no units are supplied, the application assumes default units of mol/s are used. If the user enters units, the program attempts to parse the units. If the units are mass based, they are first converted to kg, then converted to moles using the molecular weight specified for the component.

Unit class	Available units
Molar	'mol', 'mole', 'kmol', 'kmole'
Mass	'mg', 'g', 'kg', 'te', 'tonnes', 'ton', 'tons', 'lb', 'pound', 'oz', 'ounce'
Time	'ms', 's', 'm', 'min', 'h', 'hr', 'hour', 'day', 'wk', 'week', 'yr', 'year'

Table 5.1: Available units

It is a relatively simple task to add more units if required.

The application also makes use of fractions stored as decimal fractions (such as 0.75). Since the user may wish to enter the fraction as a percentage (75%) or a vulgar fraction (3/4), the application is capable of converting percentages and vulgar fractions to decimals before storing the data.

The project aims intended to allow the user to view the outputs of the flowrates using whatever units they wished but it was not possible to produce this in the required time. The output is always in SI units and the user can convert them using the export to excel function should this be required.

This system could be extended to pressures, temperatures and all other information used by the system.

5.6 Internationalisation

Django comes with built in internationalisation code which can allow a web application to be translated into many languages. Throughout design, it was attempted to write compliant internationalised code, allowing the program to be translated at a later stage.

Whilst the ability to translate the program into other languages is useful, it was not considered essential and as a result minimal effort was put into ensuring that the internationalisation code is correct. It is expected that there are several places in the code where strings are not translated before being passed to the user. Similarly there may be occasions where strings that are translated may cause errors with other parts of the code.

Unfortunately these errors will remain hidden until the whole application is translated, at which point these bugs can be found and fixed.

5.7 UI Testing

Testing is necessary to ensure that the application is indeed capable of being used by engineers at the end of the day. During early stage design, the application was not tested outwith the author and the project supervisor.

Once the all the basic features had been implemented, the application underwent alpha testing in a one-to-one environment with several 4th and 5th year students. During these testing sessions, the participant was asked to produce a simple process with minimal instruction. Once the participants ran into issues, they were prompted by the author and the issue was noted to be fixed later.

After the period of alpha testing and all major user interface issues were resolved, a period of beta testing was initiated whereby all 4th and 5th year students and faculty of chemical engineering were invited by email to use the application. It was hoped that this period of beta testing would have attracted more interest than it did; despite this, the feedback received was adequate in further developing the user interface.

Whilst the application was designed to handle multiple users simultaneously, this was not fully tested due to the lack of beta volunteers. There is a potential that bugs will arise once multiple users start to use the system however these cannot be identified without heavier usage.

SECTION 6

Case Studies

The previous sections describe a web application for the construction of a process mass balance. This section examines a number of places where it could be used.

6.1 First Year Students

During a first year course, the application could provide a valuable teaching resource when trying to explain certain concepts and design patterns. One case involves explaining the difference between mass and molar flowrates and similarly mass and molar fractions. The application clearly displays the the flowrates and fractions of all the streams on the same page so the user can easily compare the differences.

The application also offers a simple mass balancing tool which can be used to show the construction of a process, first with a single-pass reactor, then what happens when a recycle is added. After the basic recycle has been inserted, the process can be used to explain what happens if there are inserts present and how a purge stream operates.

With further improvements, the application could be further tailored to suit teaching needs depending on feedback from those involved with the course.

6.2 Fourth Year Design

During forth year, students need to design a chemical plant. This application can be used help them with this design process.

One of the main problems encountered with the design project is the creation of the mass balance - the earlier this can be sorted out, the sooner the other members of the group can

start producing their detailed design analyses. The problem arises when recycle loops are involved in the process making generation of the mass balance a complicated task.

Another frequently encountered problem is ensuring data is synchronised between all group members. That is, making sure that everyone in the group has access to the most up-to-date copy of the mass balance. This is usually done with group emails which have the potential to become difficult when trying to track down the most recent version.

For a 4th year group to use the application, all members of the group should first sign up to the application and one member should create a new process, making all other people in the group members.

The next step is to create all the components used in the process. Whilst this list can be added to at a later date, it should contain as many components that are known.

After the components have been added, the next step is to add the units into the process. The best method is to add the units in a forward method, from feed to product tank. After the main process line has been completed, the recycle loops should be completed.

At this stage, it should be possible to run an initial simulation and obtain flowrates. The parameters of individual units can then be modified to obtain the desired flowrates and concentrations.

SECTION 7

Discussion

This section will examine the final application produced during the course of this project and compare with the aims. It will also examine future work to improve the system.

7.1 Evaluation of Application

7.1.1 Time to Learn and Operate

During the course of alpha testing described in section 5.7, a number of areas where users got stuck attempting to use the system were identified. As a result those areas were examined and streamlined in an attempt to simplify the learning process.

Whilst the final application is relatively straightforward for someone to start using, it was not a quick process, and requires a lot of time to set up a simple model. For this application to be used in a production environment, a quicker method of initiating a system would be required.

7.1.2 Security and Robustness

Security was considered throughout development of the final application. The use of automatic forms generation and validation helps minimise the risk of many vulnerabilities due to code injection in forms.

The current version of the application is being run on a standard HTTP server. This transmits the data unencrypted across the Internet and may be intercepted by third parties. If the application is hosted on a secure server and the data transmitted over HTTPS, then the data is encrypted and can not be read by third parties. Serving data over HTTPS is

often more expensive than HTTP in shared hosting environments and requires slightly more processing during the encryption and decryption stages.

In cases where encryption is not considered secure enough, the application may be hosted on a private internal company server and served over an Intranet. This solution would require more setup by a company and may prove unattractive to many companies.

Due to the limitation of the simulation to linear equations, this makes the application relatively robust when dealing with multiple recycle loops unlike iterative solution approaches. Currently, short delays are noticed when running the application with a large number of units and components. After a quick investigation, it appears that most of the delay is caused by accessing the database. Enabling the caching framework bundled with Django may help reduce the level of processing required and should be investigated before deploying the application in a production environment.

A bug has been reported regarding change of unit types and inserting units into streams. If one of these operations is cancelled part way thorough, the process is not reverted to the previous state and instead the unit or stream is deleted. This bug has been documented and should be fixed as a high priority.

7.1.3 Deployment

This section examines the software requirements to deploy the application on a web server.

Server

To deploy the final application, the application files need to be stored on a server with Python 2.4+ and Django 1.0+ installed. To serve the data, a WSGI compliant web server such as Apache with the mod_wsgi module enabled is also required.

The application should be deployed on its own sub-domain:

‘`http://impress.example.com`’

It can not be deployed on a subdirectory of another site:

‘`http://example.com/impress/`’

To enable the layout feature which generated diagrams of the process layout, pygraphviz 0.35+ must be installed on the server. To enable the export to excel feature, pyexcelerator needs to be installed. If these optional modules are correctly installed on the server, the features should be automatically enabled.

Database

The application needs an SQL database to store the data. The default database used is SQLite, a lightweight database supplied with Python 2.5, and stores the data in one file. Whilst this is a useful database for lightweight testing, it is not suitable for production environments and should be replaced with a production database such as MySQL or PostgreSQL.

Configuration

Any configuration of the application should take place in a file stored in the main folder called `local_settings.py`. This way, it is not overwritten when installing updates. To configure the Apache web server, an example configuration file is provided in the deploy folder.

7.2 Future Work

This section details features that were considered but not implemented through the course of the project due to time constraints.

7.2.1 Energy Balance

The current application is capable of generating a rough mass balance for the process. Once the mass balance has been generated, it may be useful to produce an energy balance to highlight areas where heat transfer into and out-of the process. The key aspect of forming an energy balance would be the collection of physical property data for the streams and mixtures.

Other features that could be provided by the application may involve assisted heat exchanger sizing and pressure drop through a pipe calculations.

7.2.2 Revision Control

During the design process it is often necessary to revisit earlier stages of design and change a decision. The current application effectively deletes earlier versions of the process when it is updated. It would be usefully to generate a revision control system whereby all changes are logged and allows the user to revert to earlier versions of the process.

There are tools available to provide automatic revision control for Django applications, however the key would be linking versions of the units, components and streams with the version of the overall process.

7.2.3 Faster Process Generation

As explained in section 7.1.1, it was noted that the time to initiate a simple process was too long. Two alternative input methods could be considered.

Natural Language Parser

Previous work on a similar project by Andrews[53] made use of a natural language parser to quickly construct a process, which can then be corrected and refined by the user. Natural language parsing examines a portion of text provided by the user and scans for keywords like 'separa' which would match separate, separator, separates and separated. This is then used to indicate that a separator unit should be used in the process at this section.

Graphical construction

A different method could make use of a graphical construction where the user selects a unit and places it on the layout. This method could make use of one of many JavaScript libraries available which provide drag and drop support.

SECTION 8

Conclusion

This report has shown that a web based application can assist process engineers during the early stages of design. The main advantages of a web based application include the immediate transfer of results to all team members and the ability to run the application on many different computer systems including mobile Internet devices.

The application was limited to linear equations to increase the robustness of calculations at the expense of accuracy. This was considered a reasonable compromise for use during the early stages of design.

The integration of an authentication and authorisation system can be used to keep processes private and secure yet share with authorised users. Higher levels of security can be provided by using a secure server or hosting the application on an internal Intranet.

The application employs a number of features to improve the user experience including automatic collection of component physical property data and flexible unit inputs, allowing the user to insert data into the application using their preferred units.

Future work on the application should look at implementing an energy balance over the process and adding a revision control feature to allow users to revert to earlier versions of the process. The current method user interface is suitable for editing an already existing process, however it is cumbersome to generate a new process. A faster process generation method should be added.

Whilst this is the first release of the application, it has already shown that it could be useful in a number of situations including the 4th year design project and teaching process calculations in the earlier years. With further development, this application has the potential to increase its scope and enter other markets.

Glossary

Apache A popular open source web server, 31

CAS Registry Number A unique chemical identifier for chemical compounds and elements assigned by a division of the American Chemical Society, the Chemical Abstracts Service (CAS), 16

HTTP Hyper Text Transfer Protocol, Used to transfer webpages over the Internet, 3

HTTPS Hyper Text Transfer Protocol Secure. An encrypted version of HTTP, 3

Netbook Ultra portable low specification laptops, 2

Regular Expression A regular expression is a string of characters that is used to specify a set of strings that match it., 24

Server A server can refer to a physical computer assigned to a particular task or a piece of software to run a task, 3

SQL Structured Query Language - A database computer language to manage relational databases, 32

Web Server A software server used to serve webpages, 3

WSGI Web Server Gateway Interface. A Python standard used to define a common method of communicating with a web server, 31

References

[1] JM Douglas. *Conceptual Design of Chemical Processes*. McGraw-Hill, 1988. ISBN 0070177627.

[2] RK Sinnott. *Chemical Engineering Design*, volume 6 of *Coulson & Richardson's Chemical Engineering Series*. Elsevier Butterworth-heinemann, 4th edition, 2005. ISBN 9780750665384.

[3] M. Jazayeri. Some trends in web application development. In *International Conference on Software Engineering*, pages 199–213. IEEE Computer Society Washington, DC, USA, 2007.

[4] Gmail. URL `http://mail.google.com`.

[5] Hotmail. URL `http://www.hotmail.com`.

[6] Flickr. URL `http://www.flickr.com`.

[7] Picasa. URL `http://picasaweb.google.co.uk`.

[8] LiveJournal. URL `http://www.livejournal.com/`.

[9] Blogger. URL `http://www.blogger.com`.

[10] Google calendar. URL `http://www.google.com/calendar`.

[11] XTS (eXtensible Ticketing System). URL `http://www.opentheatre.org.uk`.

[12] Google documents. URL `http://docs.google.com`.

[13] B Hayes. Cloud computing. *Commun. ACM*, 51(7):9–11, 2008. ISSN 0001-0782. doi: http://doi.acm.org/10.1145/1364782.1364786.

[14] GSMA - mobile broadband. URL `http://www.gsmworld.com/our-work/programmes-and-initiatives/mobile-broadband/index.htm`.

[15] Gears. URL http://gears.google.com/.

[16] A. Goldberg, R. Buff, and A. Schmitt. A comparison of HTTP and HTTPS performance. *Computer Measurement Group*, 1998.

[17] D. Schwabe, L. Esmeraldo, G. Rossi, and F. Lyardet. Engineering web applications for reuse. *Multimedia, IEEE*, 8(1):20–31, Jan-Mar 2001. ISSN 1070-986X. doi: 10.1109/93.923950.

[18] History of PHP and related projects. URL http://www.php.net/history.

[19] MediaWiki. URL http://www.mediawiki.org/wiki/MediaWiki.

[20] C Henderson. Flickr and PHP, October 2004. URL http://www.iamcal.com/talks/flickr_php.pdf.

[21] B Shire. PHP and Facebook, May 2007. URL http://blog.facebook.com/blog.php?post=2356432130.

[22] Your first PHP-enabled page, February 2009. URL http://www.php.net/manual/en/tutorial.firstpage.php.

[23] Symfony project. URL http://www.symfony-project.org/.

[24] Python programming language – official website, . URL http://www.python.org/.

[25] Python streamlines space shuttle mission design, January 2003. URL http://www.python.org/about/success/usa/.

[26] Quotes about Python, . URL http://python.org/about/quotes/.

[27] Python : the holy grail of programming, . URL http://cdsweb.cern.ch/record/974627?ln=no.

[28] About Python, . URL http://www.python.org/about/.

[29] Web servers and Python, November 2008. URL http://wiki.python.org/moin/WebServers.

[30] The apache HTTP server project. URL http://httpd.apache.org/.

[31] web2py: Enterprise web framework. URL http://www.web2py.com/.

[32] Google App Engine. URL http://code.google.com/appengine/.

[33] J Everling. Unleash Django with app-engine-patch, February 2009. URL http://code.google.com/appengine/articles/app-engine-patch.html.

[34] A Albrecht. Running App Engine applications on Django, October 2008. URL http://code.google.com/appengine/articles/pure_django.html.

[35] Ruby on Rails. URL http://rubyonrails.org/.

[36] TiddlyWiki, . URL http://www.tiddlywiki.com/.

[37] Plugins - TiddlyWiki.org, . URL http://tiddlywiki.org/wiki/Plugins.

[38] Django: FAQ, 2008. URL http://docs.djangoproject.com/en/dev/faq/general/.

[39] Django documentation. URL http://docs.djangoproject.com.

[40] Adrian Holovaty and Jacob Kaplan-Moss. *The Definitive Guide to Django: Web Development Done Right*. Apress, 2007. ISBN 1590597257.

[41] Pinax project. URL http://www.pinaxproject.com/.

[42] T Reenskaug. Models - Views - Controllers. Xerox PARC technical note, December 1979.

[43] JW Ponton and SR Andrews. Linear equation solver. URL http://www.chemeng.ed.ac.uk/people/jack/newWork/allMSO/MSO/lab/linear6_ol.html.

[44] W.O. Galitz. *The essential guide to user interface design: an introduction to GUI design principles and techniques*. Wiley, 2007.

[45] A Raskin. Don't make me click. In *Google Tech Talks*, April 2008. URL http://www.youtube.com/watch?v=EuELwq2ThJE.

[46] 2007 desktop linux market survey, August 2007. URL http://www.desktoplinux.com/cgi-bin/survey/survey.cgi?view=archive\&id=0813200712407.

[47] Browser market share, March 2009. URL http://marketshare.hitslink.com/browser-market-share.aspx?qprid=0.

[48] Why validate?, 2009. URL http://validator.w3.org/docs/why.html.

[49] NIST general information, . URL http://www.nist.gov/public_affairs/general2.htm.

[50] NIST chemistry WebBook, . URL http://webbook.nist.gov/.

[51] Featured spreadsheet software, 2008. URL http://spreadsheet.software.informer.com/.

[52] Microsoft Office Excel 97 - 2007 binary file format specification (*.xls 97-2007 format), 2007. URL http://download.microsoft.com/download/0/B/E/0BE8BDD7-E5E8-422A-ABFD-4342ED7AD886/Excel97-2007BinaryFileFormat(xls)Specification.xps.

[53] SR Andrews. *Chronicling Process Model Construction using World Wide Web Technology*. PhD thesis, University of Edinburgh, 2001.