An OpenMath based Unit Converter

David Prangle

Batchelor of Science in Computer Science with Honours
The University of Bath
May 2007

An OpenMath based Unit Converter

Submitted by: David Prangle

COPYRIGHT

Declaration

This dissertation is submitted to the University of Bath in accordance with the requirements of the degree of Batchelor of Science in the Department of Computer Science. No portion of the work in this dissertation has been submitted in support of an application for any other degree or qualification of this or any other university or institution of learning. Except where specifically acknowledged, it is the work of the author.

Signed:

Abstract

The OpenMath framework provides the protocol for interactive mathematics on the Internet; this is done through the use of Content Dictionaries that store the semantic meaning of mathematical symbols. This facilitates the transformation between the OpenMath representation and their corresponding internal representation within the application and as a result should increase the robustness of mathematical software.

This dissertation attempts to demonstrate the suitable of OpenMath as a basis for specialised applications. In the marketplace there are a number of standard mathematical unit converters, however I would like to analyse the efficiency of OpenMath as a platform for a mathematical unit converter.

OpenMath as a language is still very early in its development cycle and is yet to be utilised by the mathematical community. By demonstrating OpenMath as a suitable foundation for a unit converter I hope to create an extensible unit converter and increase the use of OpenMath in future projects.

Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank Professor Davenport and the rest of the computer science department, both staff and students, for all of their help and guidance.

I would also like to say thank you to my family and friends, their relentless support really pulled me through.

Thank You

1

Introduction

The Internet is growing rapidly and as a result is bringing around some exciting opportunities in the computing community. People are able to communicate at a much faster rate and as the Internet becomes an ever-increasing part of our everyday lives access to particular fields of interest is greatly improved. OpenMath is one of those fields, as a language it is relatively young; development was started just 13 years ago.

Due to the relatively minute amount of research made into OpenMaths it is highly under-utilised, research to date has mainly been focusing on the improvement of the language and as a result the development of OpenMath is yet to manifest itself in any programs that can truly demonstrate the advantages.

OpenMath describes the logical structure of mathematical formula and concepts (The OpenMath Society, 2006). It is widely accepted as a basic protocol for mathematical interaction and communication. It provides semantic-rich mathematical symbols through constructing a set of content dictionaries; content dictionaries are central to the OpenMath philosophy of transmitting mathematical information. It is the OpenMath content dictionaries which actually hold the meanings of the objects being transmitted (Buswell et al, 2004 Page 34). The content dictionaries are defined in two different ways: XML or binary encoding.

To support the computers mathematic interaction XML encoding is necessary. XML, The Extensible Markup Language, is a W3C-endorsed standard for document markup. It is a markup language for documents containing structured information; in essence it defines a generic syntax used to mark up data with simple, human readable tags (O'Reilly, 2001, Page 1). When XML was first introduced, it was hailed as the cornerstone of a new kind of technology that would permit interoperable Web services, this means you can write your own program that will interact with, message and manipulate data in XML documents.

To summaries, mathematical conversions can be formally defined in XML encoded content dictionaries. I am going to investigate the possibility of designing a system that can read and manipulate the content dictionaries to draw on a result for a conversion requested by the user.

The main strength of OpenMath is that applications can be used which can handle every Content Dictionary or a changeable number of Content Dictionaries, perhaps after being sent Content Dictionaries in some way (Buswell at al, 2004). This is the main reason OpenMath was decided upon as a foundation.

There are a number of unit converters written in many different languages that are readily available on the Internet; by not using OpenMath the converters have limitations. Current converts in the marketplace can only carry out conversions that have been written into the system. OpenMath is extensible, by basing a converter on OpenMath it means any number of content dictionaries can be added to the program at any time, these new Dictionaries can then be used to carry out more conversions.

2

Literature Survey

2.1 An Introduction to OpenMath

OpenMath as a language was formalized in The OpenMath Standard (Caprotti et al. 2004)

2.1.1 OpenMath, The Background

OpenMath is a standard for representing mathematical data in as unambiguous a way as possible, it can be used to exchange mathematical objects between computer programs, stored in databases, or published on the worldwide web (Caprotti, 2004). It is tightly focused on representing semantic information and is not intended to be used directly for presentation, although tools exist to facilitate this. OpenMath originated from the Computer Algebra community and since 1997 its development has been partially funded by the European Union under a multimedia ESPRIT project. Computer Algebra packages were getting bigger and more unwieldy and it seemed reasonable to adopt a generic "plug and play" architecture to allow specialised programs to be used from general purpose environments. There were plenty of mechanisms for connecting software components together, but no common format for representing the underlying data objects. It quickly became clear that any standard had to be vendor-neutral and that objects encoded in OpenMath should not be too verbose. In 1998, the Worldwide Web Consortium (W3C) produced its first recommendation for the Extensible Mark-up Language (XML), intended to be a universal format for representing structured information on the worldwide web. It was swiftly followed by the first MathML recommendation which is an XML application oriented mainly towards the presentation of mathematical expressions, OpenMath was then derived from these developments.
To date there have been three main projects focused on the development of OpenMath as a language, in 1995 the Editing and Computing Project was launched. This was the organisation of five workshops open to all that have been interested in the project. These initial meetings were mainly of a political nature, a steering committee was elected to management the OpenMath development. This project ran for 2 years and produced OpenMath v1.0 with prototype Phrasebooks. The Esprit project was funded by the European Union between 1997 and 2000; this project carried on with the further development of the language and was in turn replaced by the OpenMath Thematic Network. This ran from July 2001 until June 2004 and was also sponsored by the European Union; it was a logical follow on from the Esprit project. The projects main aims were to organise workshops bringing together people working on OpenMath from around the world, provide a continued focus-point for the development of the OpenMath Standard and Content Dictionaries, coordinate the development of OpenMath and MathML tools and applications and to disseminate information about OpenMath and MathML; the projects produced the OpenMath standard we use today, v2.0.

2.1.2 OpenMath, The Specifics

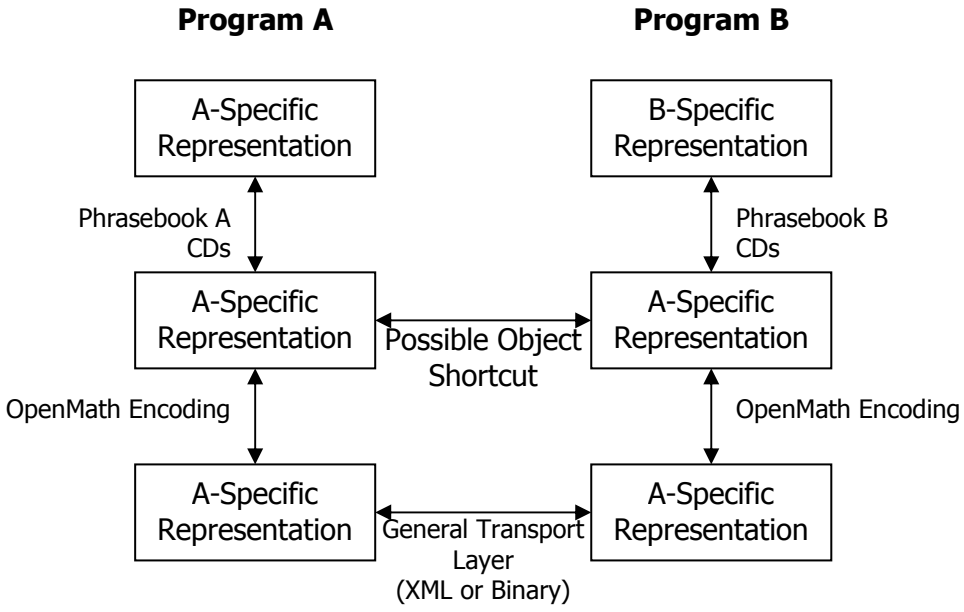
OpenMaths is a relatively new field of research, it is emerging as a standard for encoding the meaning of a mathematical object rather than just creating a visual representation. OpenMath allows the free exchange of mathematical objects between software systems and human beings; it enables the manipulation and computing of mathematical expressions embedded in web pages and is designed to be machine-generated and machine-readable.
Typical uses include:

- displaying in a browser
- exchanging between software systems
- cutting and pasting for use in different contexts
- testing to see if mathematically sound
- increasing document interaction

2.1.2.1 The Architecture

Within OpenMath there are three layers of representation of a mathematical object. The first is the internal representation used by an application, the private layer. The second is an abstract layer, this is the representation as an OpenMath object, these two layers may, in some cases, be the same. The third is a communication layer that translates the OpenMath object representation into a stream of bytes. An application dependent program manipulates the mathematical objects using its internal representation; it can convert them to OpenMath objects and communicate them by using the byte stream representation of OpenMath objects, shown below:

Figure 1. OpenMath Architecture



2.1.2.2 Content Dictionaries

Content Dictionaries (CDs) are central to the OpenMath philosophy of transmitting mathematical information; they actually hold the meanings of the objects being transmitted. For example if application *A* is talking to application *B*, and sends, say, an equation involving multiplication of matrices, then *A* and *B* must agree on what a matrix is, and on what matrix multiplication is, and even on what constitutes an equation (Buswell, 2004). A content dictionary holds the meanings of various mathematical "words", it is not content dictionaries themselves which are being transmitted, but some "mathematics" whose definitions are held independent to the application within the content dictionaries, with a set of symbol definitions A and B can now talk in a common "language".
In many cases, the content dictionaries that an application understands will be constant, and be built-in to the application's mathematical use. However applications can be developed that can handle every content dictionary or a changeable number of content dictionaries, perhaps after being sent content dictionaries in some way.

A Content Dictionary consists of the following mandatory pieces of information:

1. A name
2. A description of the Content Dictionary
3. A revision date, the date of the last change to the Content Dictionary. Dates should be stored in the ISO-compliant format YYYY-MM-DD, e.g. 1985-03-15.
4. A review date, a date until which the content dictionary is guaranteed to remain unchanged
5. A version number which consists of a major and revision part – If the status is not experimental the version number should be a positive integer. When the major number is increased the revision number is normally reset to zero
6. A status, which can be:
   - Official: approved by the OpenMath society
   - Experimental: under development and thus liable to change
   - Private: used by a private group of OpenMath users
   - Obsolete: kept only for archival purposes
7. A CD base which, when combined with the CD name, forms a unique identifier for the Content Dictionary. It may or may not refer to an actual location from which it can be retrieved
8. A series of one or more symbol definitions

2.1.2.3 Writing Content Dictionaries

The symbols contained in a content dictionary form the mechanism by which OpenMath achieves its goal of being an extensible framework for exchanging semantically rich representations of mathematical objects (Davenport, 2000). The motivation for writing a new CD would be that there are some semantics that one wishes to exchange. When writing a new content dictionary you need to ensure that:

- There must be a new piece of semantic information to convey
- It must be possible to informally write down the semantics that the author of the CD intends to convey
- There must be a motivation for wishing to convey the information in a content dictionary. An excellent example of this would be to define further units that could then be used in the unit converter.

A well written content dictionary should contain:

- Examples. Every symbol defined in a CD should have at least one example, either as such (i.e. <Example>), or via an FMP which uses it. In general FMPs are more useful because formal systems can also read the FMP whereas examples provide no further semantics. As examples illustrate uses unlike FMPs a combination of the two is usually compromised upon.
- If an FMP is given, then the equivalent CMP should also be given and if possible the conversion should be done as well. CMPs should be written in English and in full, no abbreviations.
- FMPs should be as comprehensive as sensibly possible whereas examples are generally a lot more concise.
- The CDUses field, which lists all of the other CDs which have symbols occurring in this CD should be correct.
- It is naive to expect a CD to be completely self-contained so restraint is necessary. Particularly, a CD should not be using other CDs which are less "official" that itself.
- The corresponding STS file should be written, these signatures should be as helpful as possible.

2.1.2.4 Phrasebooks

The Phrasebook is an internal interface that carries out the conversion of an OpenMath object to/from the internal representation in a software application. It is envisioned that a software application dealing with a specific area of mathematics declares which content dictionaries it understands, this means the translation can be governed by the content dictionaries and specifics of the application. Essentially, the Phrasebook of the application is able to translate OpenMath objects built using symbols from these content dictionaries to/from the internal mathematical objects of the application.

2.1.2.5 Objects

OpenMath uses objects as a representation of mathematical entities. The objects can then be communicated among various software applications whilst preserving their semantics.
Basic OpenMath objects are one of the following:

- Integer – No predefined range.
- IEE Floating point number – Double precision floating-point numbers.
- Character String – Corresponds to "Characters" in XML.
- Bytearray – Sequence of Bytes.
- Symbols – Encodes three fields of information,
    - Symbol name – Sequence of Characters
    - Content Dictionary name – the location of the symbol definition
    - Content Dictionary base URI or cdbase – used to disambiguate multiple Content Dictionaries. The URI usually follows the standard:
      URI = cdbase-value + '/' + cdvalue + '#' + name-value
      Example: Symbol sin in cd transcl and cdbase
      http://openmath.org/cd = http://openmath.org/cd/transcl#sin

Objects can also be derived:

- Foreign – Used to import a non-OpenMath object into an OpenMath application.
- Application – Constructs an object from a sequence of one or more OpenMath objects. The first child of an application is referred to as it's "head" while the remaining objects are called its arguments.
  Example: application(sin, x) corresponds to sin(x)
  application(Rational,1,2) corresponds to 1/2
- Binding – Is constructed from an object, a sequence of zero or more other variables followed by another object. The first object is the Binder.
  Example: binding(lambda,x,application(plus,x,2) corresponds to λ x.x + 2
- Attribution – Decorates an object with a sequence of one or more pairs made up of an OpenMath symbol, the "attribute" and an associate object, the "value of the attribute". The value of the attribute can be an attribute itself.

Example: attribution(attribution(A,S_1 A_1,.....,S_hA_h),$S_{h+1}A_{h+1}$,.....S_nA_n) is equivalent to attribution(A,S_1A_1,.....S_hA_h,$S_{h+1}A_{h+1}$,.....S_nA_n)

- Error – Is made up from an OpenMath symbol and a sequence of zero or more OpenMath objects.

Interesting Note **-** OpenMath objects do not specify any computational behaviour, they merely represent mathematical expressions. The OpenMath philosophy is to leave an application to decide what it does with an object once it has received it. OpenMath is not a query or programming language. Because of this, OpenMath does not prescribe a way of forcing "evaluation" or "simplification" of objects. Thus, the same object 2+3 could be transformed to 5 by a computer algebra system, or displayed as 2+3 by a typesetting tool.

2.1.2.6 Symbols

A symbol is used to construct an OpenMath object if it is the first child of an OpenMath application, binding or error object, or an even-indexed child of an attribution object (i.e. the key in a (key, value) pair). The role of an OpenMath symbol is the restriction on how it may be used to construct a compound OpenMath object. A symbol cannot have more than one role and cannot be used to construct a compound object in a way which requires a different role. The possible roles are:

- Binder - first child of a binding object.
- Attribution - used as key in an attribution object i.e. the first element of a (key, value) pair. This can be used as a clarification of how the attribution should be interpreted.
- Semantic-attribution - This is the same as attribution except that it modifies the meaning of the attributed OpenMath object and thus cannot be ignored by an application, without changing the meaning.
- Error - first child of an error object.
- Application - first child of an application object.
- Constant - The symbol cannot be used to construct a compound object.

2.1.2.7 OpenMath Vs MathML

MathML and the OpenMath development cycles have been strongly linked, there is an obvious overlap in both the people researching and developing the languages as well as the actual logic itself. MathML has been developed to provide both content and presentation mechanisms, it is a fixed set of mathematical operators that define a default presentation. The mathematical range of MathML can be extended by attaching semantic meanings to expressions. These external semantics may be in any form, but one obvious contender is to extend MathML using OpenMath to define the semantics. OpenMath, during the course of its development has matured and found a natural role as the encoding of Mathematics in situations where more

precision or richness is required than may easily be obtained by using MathML. The preciseness of OpenMath will be required ahead of MathML for the unit converter as it will enable more advanced and obscure conversions. This fixed range and default presentation makes it more suitable for being embedded in web browsers than OpenMath. For further detail about the conversion between OpenMath and MathML see Conversion between MathML and OpenMath (Carlisle, 2001).

2.1.2.8 eXtensible Markup Language (XML)

OpenMath defines two ways to encode its objects: XML and binary encoding. To support the computers mathematic interaction XML encoding is necessary. When XML was first introduced, it was hailed as the cornerstone of a new kind of technology that would permit interoperable Web services. XML provides a standardized way to represent structured and typed data. XML has been incorporated into every aspect of application and enterprise development. It is now a part of networking protocols, Web servers, operating systems, programming languages, application servers etc.

XML is a markup language for documents containing structured information. A markup language is a mechanism to identify structures in a document. XML defines a generic syntax used to mark up data with simple human-tags. It provides a standard format for computer documents. This format is flexible enough to be customised for domains as diverse as web sites, electronic data interchange, software configuration, remote procedure calls and more.

Since XML was first introduced it has been involved nearly everywhere information is managed. XML is intended to be understandable by both humans and applications. Therefore, the syntax for XML is very simple and can be understood easily by humans. At the same time the XML document is strictly formed with absolute minimum optional features so that the application can read it as well.

2.2 Parser APIs

A parser is essential to an XML-aware application. It takes the XML document as an input and generates data which can then be manipulated and handled by other XML tools or APIs. The three APIs I explored were SAX, DOM and JDOM.

2.2.1 SAX

SAX is an event driven API for XML, it provides a set of classes for inserting and manipulating. Put simply, when an XML document is in the process of parsing events occur. These events then provide the points that application code can insert into and manipulate. Its sequential parsing process does not allow for random access to an XML document so forward and backward movement among elements is difficult. The OpenMath content dictionaries are small XML documents, although SAX requires a lot less memory than DOM, given a project of this type DOM is definitely a more suitable choice.

2.2.2 DOM

The Document Object Model (DOM) is a tree structure based parsing API and part of W3C recommendation. SAX provides access to data during the parsing process whereas DOM has the facility to provide access to the data both during and after the parsing process.

2.2.3 JDOM

JDOM is a Java-based Document Object Model for XML that integrates with DOM and SAX and uses parsers to build the document. Similarly to DOM, JDOM provides the tree-structure representation of XML documents. JDOM has been developed specifically for Java and is often more convenient as it requires less coding in purely Java applications, as my project will focus on Java I have decided to use JDOM.

2.3 Existing Projects

OpenMath is still in its early stages of development and as a result there are very few projects utilising it.

2.3.1 OpenMath Browser

This project attempted to demonstrate the possibility of constructing an automatic query system, which can benefit both human readers and Phrasebooks or other OpenMath-aware applications that need to access content dictionaries remotely. The implemented system was developed as a Java XML Web Application and comprised of two parts:

- One part of the system was based on SOAP and handled queries from applications
- The second part was based on Cocoon XSP and handled queries from human readers.

The two parts were then designed to share the same core implementation based on DOM.

2.3.2 OpenMath Web Service Package

This was development in 2003 and had three main goals:

- Add functionality to the interface between the mathematical software applications and content dictionaries. The interface should be able to extract information from the content dictionaries and respond to the applications. The applications can then use the information and create new content dictionaries automatically.
- Build a user interface between human beings and content dictionaries, hence, the user can retrieve information they want from content dictionaries. This was meant as an aid to people who are interested in building some kind of OpenMath tool. For Example, a designer of a Phrasebook, to obtain information about content dictionaries more easily.
- Web services are still evolving; there is an overwhelmingly long list of proposed standards and the complex interactions between each of them. The final goal of the project was to try to implement these proposed standards and provide a successful practical example for the industry.

This project comprised of two parts, the first component handled user's requests on the Web via HTTP and the second handled requests from applications connected through the Internet. The purpose of the project was to allow mathematical software to create new content dictionaries automatically; this would facilitate the

transformation between the OpenMath representation and their corresponding representation within an application. As a result of this project the robustness of OpenMath was increased and it has now become more diverse to fit different kinds of mathematical software.

The purpose of these projects was to increase the robustness of OpenMath and make it more available to different kinds of mathematical software and projects being developed. In both systems there were issues with speed, reliability, security and error handling but on the whole they were deemed a success.

2.3.3 Algebra Interactive & ActiveMath

Algebra Interactive and ActiveMath are two pioneering systems that use OpenMath as a semantic foundation. They focus on web-learning HTML-based books for undergraduate students in all fields of science, e.g. mathematics, computer science, physics and engineering.

The main themes throughout both systems range from the elementary structures of the integers, polynomial rings in one variable to the abstract notions of groups, rings and fields; the systems tend to focus on an algorithmic approach to these algebraic structures.

The books provide the required knowledge for users of any aptitude as well as numerous exercises to work through. The programs also provide further, more novel features; these include interactive examples and tools, such as a Concept Map Tool or Plotting Tools.

2.3.4 Existing Unit Converters

There are a whole host of unit converters already available on the internet that have all been written in a number of different languages. As the back end of my system will be centred on OpenMath a review of how existing systems carry out conversions is not necessary, however, it is important to highlight some of the similarities and strength of the front ends. There tends to be a given format amongst the unit converters available on the internet:

- A text box is usually used to enter the amount you would like to convert into the system
- The source and destination units are typically stored in lists or drop down boxes.
- The design of the system is usually kept very simple with minimal distraction or confusion for the user.

3

Requirements

This project has two key goals:

1 To be able to read new Content Dictionaries into the system

2 To be able to read and utilize the information available in the content dictionaries to carry out conversions

The requirements process will begin with the requirements elicitation and analysis, all of the requirements highlighted will be gathered from the analysis of previous OpenMath projects and existing unit converters.

Once the requirements elicitation has been carried out I will draw up both the functional and non-functional requirements, as well as any optional requirements that are not core to the system but would add value.

To finish, the requirements validation will ensure that the requirements specification satisfies the original problem and that all requirements are complete, consistent, verifiable and realistic.

3.1 Requirements elicitation and analysis

3.1.1 Methodology

The requirements elicitation will focus on a few primary sources of requirements. Firstly, a number of requirements can be drawn from the core goals identified in the Literature Survey. Furthermore the literature survey has enabled me to review previous OpenMath projects and existing unit converters in the marketplace, by examining the flaws within these systems I will be able to draw upon further requirements and constraints for my system.

3.1.2 Hardware and Software Considerations

Most unit converters have been made available on the Internet; this is because they do not require a great deal of processing power and the Internet acts as a gateway to the largest concentration of users. It is important to note a few things, firstly for the converter to compete in the existing marketplace it should be easily transferable to become an Internet application and it should run on most standard PC's. The system is going to be written in XML and Java on a 4 year old laptop running a Pentium 4 processor so no obvious issues should arise with respect to these requirements.

3.1.3 Carrying out Conversions

When a user attempts to carry out a conversion the amount to convert, source unit and destination unit all need to be considered. The system will have to search through the content dictionaries to match the source and destination units with a valid definition and use that information to carry out the conversion.

3.1.4 Searching Content Dictionaries

When a conversion is carried out by the user the system will need to search through the bank of content dictionaries to ensure firstly the conversion is possible and secondly how the conversion will be carried out. This will need to work for a number of content dictionaries and shouldn't be constrained by the layout or design of any particular dictionary.

It may well be the case that for some conversions there is not a direct mapping from the source unit to the destination unit. In this case is will be necessary for the system to search through multiple content dictionaries to create a chain of conversions in order to find a route from the source to the destination unit.

3.1.5 Adding New Libraries

Having reviewed a number of unit converters on the Internet it was quite apparent that they all had limitations when it came to the number of units it is possible to convert between. When carrying out the literature review quite a few advantages of OpenMath were highlighted, the central reason for using OpenMath as a foundation for a specific application is that it is extensible. It is essential that the system be able to add new content dictionaries, the content dictionaries store all of the conversion details and by being able to add further content dictionaries to the system it will provide more possible conversions. In theory, as OpenMath grows as a language more content dictionaries will be produced, hence increasing the conversions available in the converter.

Secondly, when the content dictionaries are added to the system it is essential that the new units are made available to the user. This is a similar process to when a conversion is carried out but rather than extracting the conversion details the unit details will need to be gathered and added to the interface.

3.1.6 User Interface

It should be noted that at this point the design and completeness of the user interface is secondary in importance to the process of converting units using OpenMath. As I am using OpenMath as the foundation of the unit converter the

back end is going to be very different to that of the unit converters currently available, hence, there was very little merit behind me carrying out a thorough review of existing products. However, even the most advanced unit converter is pointless without the means to interact efficiently with it. A graphical interface will be design and implemented but some attention to styling will be sacrificed in order to allow additional effort to be placed upon the primary focus of the project.

3.2 Requirements Specification

3.2.1 Functional requirements

The functional requirements are a specification stating what the system will and won't do, the system will:

1. Be able to add new content dictionaries to the system
   1.1. Search and add content dictionary via the front end
2. (Simple Case)
   2.1. Carry out conversions using just one content dictionary achieving a 99% success rate
      2.1.1. Search through a content dictionary to find a conversion that matches the given source and destination units
      2.1.2. Extract the corresponding conversion amount from the content dictionary
      2.1.3. Interpret the conversion and apply it to the figure the user inputted
3. (Complex Case)
   3.1. Carry out conversions using multiple units achieving a 99% success rate
      3.1.1. Search through all content dictionaries to find conversions that include the source unit
      3.1.2. If a direct conversion is not available carry out an intermediate conversion
      3.1.3. Continue the loop taking the intermediate destination unit as your new source unit until the requested destination unit is reached
4. Provide a simple front-end that collates all of the functionality of the system
5. Display all available units on the front end
   5.1. When the system is initially ran it should search through all existing content dictionaries to add the units to the front end
   5.2. When adding new content dictionaries the system must make the new units available as source and destination units on the front end
6. Error Handling
   6.1. If a file you are attempting to add is not a content dictionary, return the appropriate error

6.2. If two selected units are not a compatible conversion, error handle accordingly

6.3. If any of the required details are not entered before the convert button is pressed, return an error.

7. Finalise the Process

7.1. When a conversion is carried out the result should be clearly displayed for the user and the system should be ready to carry out another conversion.

8. Time to start up

8.1. When the system is turned in it should take no longer than 15 seconds till it is fully operational and all the functionality that it provides is available to the user.

9. Time to execute user commands

9.1. When the user instructs the system to perform an operation the system should begin that operation immediately. The operation should have provided the user with the desired result within 2 seconds of the user executing the command.

10. Reliability

10.1. When the user instructs the system to perform an operation the system should begin that operation immediately. The operation should have provided the user with the desired result within 2 seconds of the user executing the command

11. Robustness

11.1. The system should be able to run for long periods of time without failing and causing the user disruption.

11.2. The system should recover from failure in under 15 seconds In the event that the system does fail then it needs to be able to be reset and quickly become functional to the user again in order to cause minimum annoyance.

3.2.2 Non-Functional requirements

The non-functional requirements specify those requirements that cannot be precisely measured by success or failure, they are subjective. They are:

1. The front end should be simple and easy to use. It should respond fast enough that it is usable
2. Future improvements should be catered for in the design
   2.1. New operations and modifications to improve efficiency should be easy to add to the system
3. The user manual should provide the user with clear and simple instructions on the functionality of the system, including illustrations of its use.

3.3 Requirements Validation

The goal of the requirements validation is to attempt to confirm the correctness and completeness of the requirements specification. The tools for carrying out this process are usually a structured or informal walk-through.

Once I had drawn up my requirements I carried out an informal walk-through with Professor Davenport, essentially, this involved me and Professor Davenport running through a summary of my completed requirements. The purpose of the informal walk-through was firstly for Professor Davenport to highlight and remove any glaring errors in the specification and from there he could then confirm that I am heading in the right direction and highlight any further work or ideas that are worth exploring.

The validation process was fairly successful, Professor Davenport was happy with the direction that I was taking and the work that I had already implement, however, there were a few extra points that Professor Davenport felt were worth addressing:

- I should explore how content dictionaries are written and ensure that my system can add a content dictionary written by myself
- Try to explore some more complex conversions and incorporate them into the system.

4

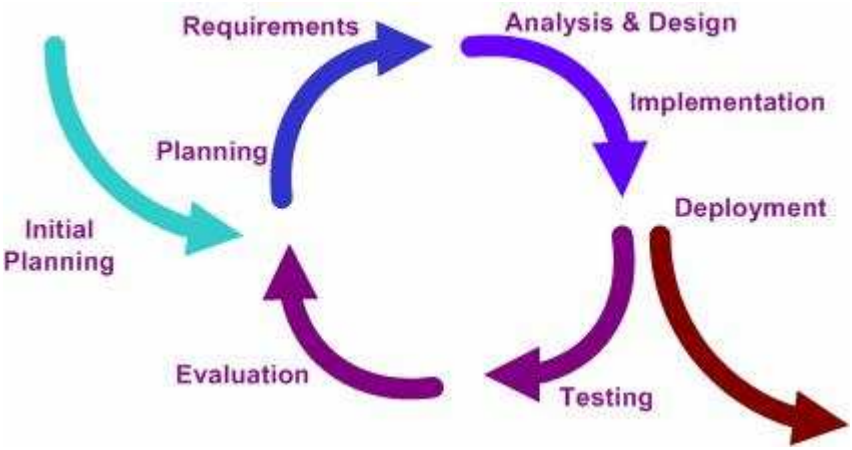
Design & Implementation

4.1 Design & Implementation Process

Having researched OpenMath and drawn up the requirements of my project, I now moved onto the design and implementation stage. Having not worked with OpenMath before I felt it would be best to adopt almost an iterative or evolutionary development processes. The basic idea behind iterative enhancement is to develop a software system incrementally. This would then allow me to take advantage of the skills learnt during the development and testing of earlier deliverables of the system.

Key steps in the process were to start with a simple implementation of the software requirements and iteratively enhance the sequence of versions until the full system is implemented. At each iteration, design modifications are made and new functional capabilities are added. This is demonstrated in the diagram below:

Figure 2. Evolutionary Design Model



4.2 Creating an OpenMath Content Dictionary

Having explored the content dictionaries online I felt the best approach to fully understand the task I am undertaking would be to design my own content dictionary. As it was a topic that has not been covered yet I produced a Pressure content dictionary.

4.2.1 Anatomy of a Content Dictionary

As previously mentioned in section 1.1.1.1 there is a standard motivation and process when designing content dictionaries. Below I am going to highlight some of the key areas of a content dictionary that I will need to manipulate in my system. The following example shows the body of a conversion between feet and metres.

```
<CDDefinition>
<Name> foot </Name>
<Description>
This symbol represents the measure of one foot. This is the standard
imperial measure for distance.
</Description>
<CMP> 1 foot = 0.3048 metres </CMP>

<FMP><OMOBJ>
  <OMA>
    <OMS name="eq" cd="relation1"/>
    <OMA>
      <OMS name="times" cd="arith1"/>
      <OMI> 1 </OMI>
      <OMS name="foot" cd="units_imperial1"/>
    </OMA>
    <OMA>
      <OMS name="times" cd="arith1"/>
      <OMF> 0.3048 </OMF>
      <OMS name="metre" cd="units_metric1"/>
    </OMA>
  </OMA>
</OMOBJ></FMP>

</CDDefinition>
```

As I have previously stated one of the main advantages of XML is that it is designed to be both human and machine readable, although a great deal of the content dictionary is simple to understand for completeness I will run through want each section. The previous example shows how to derive metres from feet. It states that 1 times foot equals 0.3048 times metre; most content dictionaries definitions follow the same format.

<CDDefinition> Element

This defines the starting point at which a definition will come together for a conversion.

<Name> Element

This simply defines the name given to the conversion that is about to be formalised.

<Description> Element

This is a standard written description of what conversion is about to be defined.

<CMP> Element

The CMP stands for the Commented Mathematical Properties. This is another written description of the conversion but in a more formalized manner.

<FMP> Element

The FMP stands for the Formal Mathematical Properties. This is where the formalized mathematical definition of an element is included in the dictionary.

<OMOBJ> Element

The OMOBJ element defines an OpenMath Object. Formally, an OpenMath object is a labelled tree whose leaves are the basic OpenMath objects integers, IEEE double precision floats, unicode strings, byte arrays, variables or symbols.

<OMA> Element

OMA is an application element. It is an element that contains a sub tree of elements and allows you to add additional information to an object.

<OMS> Element

The OMS element defines an OpenMath Symbol. These are interesting because they consist of a name and a reference to a definition in an external content dictionary. Using XML notation where the element name OMS indicates an OpenMath symbol, the above example: <OMS name="eq" cd="relation1"/> represents the standard equals function, as defined in the content dictionary "relation1".
<OMI> Element

The OMI is an OpenMath Integer, the OMI element encapsulates the value of the Integer.

<OMF> Element

The OMF is an OpenMath Float; the OMF element encapsulates the value of the Float.

<OMV> Element

The OMV is an OpenMath Variable; the OMV element encapsulates the value of the Integer.

Following the guidelines set out in On Writing OpenMath Content Dictionaries, (Davenport, 2000) I have produced my own content dictionary that outlines some pressure conversions, these includes Pascal, Newton's per square Metre, Pounds per square Metre and Pounds per square Foot; See Appendix D.

4.3 The Simple Case

The initial step in the design process is to interpret the content dictionaries and attempt to carry out a simple conversion. After review the content dictionaries the simplest conversion and starting point would be to pass the initial system:

- The amount to convert = 1
- The unit to convert from (Source Unit) = Foot
- The unit to convert to (Destination Unit) = Metre

Given these details the system should then:

- Read in one content dictionary
    - units_imperial1.ocd as it contains the required conversion from feet to metres
- Search through the content dictionary extracting a list of all the conversions that involve the source unit
- Search the list of conversions to find the conversion that involves both the source and destination unit.
- Extract the required details and carry out the conversion

4.3.1 Reading in a Content Dictionary

Reading in the content dictionaries is the job of the SAXBuilder. The SAXBuilder builds a JDOM document from files, streams, readers, URLs, or a SAX InputSource instance using a SAX parser. As it is only reading in a single file stored in a

convenient location it was just a case of initialising a new SAXBuilder and building a document using the .build() method shown below:

```
SAXBuilder parser = new SAXBuilder();
Document response = parser.build("units_imperial1.ocd");
```

4.3.2 Extracting a list of relevant Conversions

XPath is a W3C-defined language and an official W3C recommendation. The XPath language provides a simple, concise syntax for selecting nodes from an XML document. Using XPath I was able to search through the XML document produced by the SAXBuilder to extract all nodes that match the source unit, in this case foot, as each node was found it was added to a List. Finally the list was cast to an ArrayList for easier manipulation.

```
XPath x =
XPath.newInstance("/CD/CDDefinition/FMP/OMOBJ/OMA[OMA/OMS/@name='"+src
Unit+"']");
List srcNodesList = x.selectNodes(response);
ArrayList srcNodes = (ArrayList) srcNodesList;
```

As you can see from the code above, using XPath is just a case of entering the correct path so you can match the source unit to the predicate expression.

```
[OMA/OMS/@name='"+srcUnit+"']
```

In this case I want to add the OMA where the name within the OMS is equal to the srcUnit or Foot.

4.3.3 Searching the List of Nodes

Within our ArrayList we now contain a number of nodes; these correspond to all of the conversions within the content dictionary that involve the source unit. The next step is to iterate through the ArrayList to match both the source and destination units with a node. This will confirm that a conversion is possible and enable us to extract the corresponding details.

As the nodes within the List are in fact elements, when iterating through the ArrayList we must first cast each node to an element.

```
Element el = (Element) srcNodes.get(i);
```

By using XPath again, we are able to match the source and destination units with the names within each element and start to extract the required conversion details.

```
Element srcOma = (Element)
XPath.selectSingleNode(el,"OMA[OMS/@name='"+srcUnit+"']");
Element dstOma = (Element)
XPath.selectSingleNode(el,"OMA[OMS/@name='"+dstUnit+"']");
```

4.3.4 Extract the Details

Now the required elements have been established I need to extract all of the information about the destination unit so the conversion can be carried out. It is simple a case of applying XPaths to the element dstOma; extracting the conversion operator as an attribute and the conversion value as an element, which then needs to be cast from a String to a Double.

```
String dstOp   =
((Attribute)XPath.selectSingleNode(dstOma,"OMS/@name")).getValue();
String dstBase =
((Element)XPath.selectSingleNode(dstOma,"OMI")).getValue().trim();
dblBase        = (double) Integer.parseInt(dstBase);
```

InvertOp

Previously I stated that the content dictionary defined the conversion between feet and metres as 1 times foot equals 0.3048 times metre. This posses a problem because there is no guarantee that it will always be 1 times the source unit.

To solve this problem I had to gather the source unit information in the same manner as with the destination unit details, as you can see the same XPaths as before are used to extract the source operator as an attribute and the source unit value as an element.

```
String invOp =
invertOp(((Attribute)XPath.selectSingleNode(srcOma,"OMS/@name")).getValue());
String srcBase =
((Element)XPath.selectSingleNode(srcOma,"OMI")).getValue().trim();
double dblBase = (double) Integer.parseInt(srcBase);
double norm = normalise(srcVal,dblBase,invOp);
```

The difference comes when the source unit operator is then passed into the invertOp method; this is a straightforward method that returns the opposite of the operator you passed into it.

```
public static String invertOp(String operator)
{
  if (operator.equals("times")) return "divide";
  if (operator.equals("divide")) return "times";
  if (operator.equals("add")) return "minus";
  if (operator.equals("minus")) return "add";
  return "unknown";
}
```

Finally the amount to convert, source unit value and inverted operator are passed into the normalise function.

```
public static double normalise(double srcVal, double baseVal, String operator)
{
  if (operator.equals("divide")) {
     return srcVal/baseVal;
  }
  return 0;
}
```

This function will apply the inverted operator on the amount you wish to convert and the source unit value. By doing this it means that no matter what the source unit value may be it has been catered for, leaving a simple method to finish the conversion process.

Example: If I wanted to convert 10 feet into metres

The content dictionary defines the conversion as 1 times foot equals 0.3048 times metre but equally it could be defined as 2 times foot equals 0.6096.

The normalise function will do:

- 10/1 = 10, which will then be times by 0.3048 in the final method
- 10/2 = 5, which will then be times by 0.6096 in the final method

Resulting in the same answer.

4.3.5 Carry out the Conversion

By normalising the amount you want to convert it means the apply method is very simple. It takes in the normalised amount you want to convert, the destination conversion value and the destination operator and returns the answer.

```
public static double apply(double arg1, double arg2, String operator)
{
  if (operator.equals("times")) {
     return arg1*arg2;
  }
  return 0;
}
```

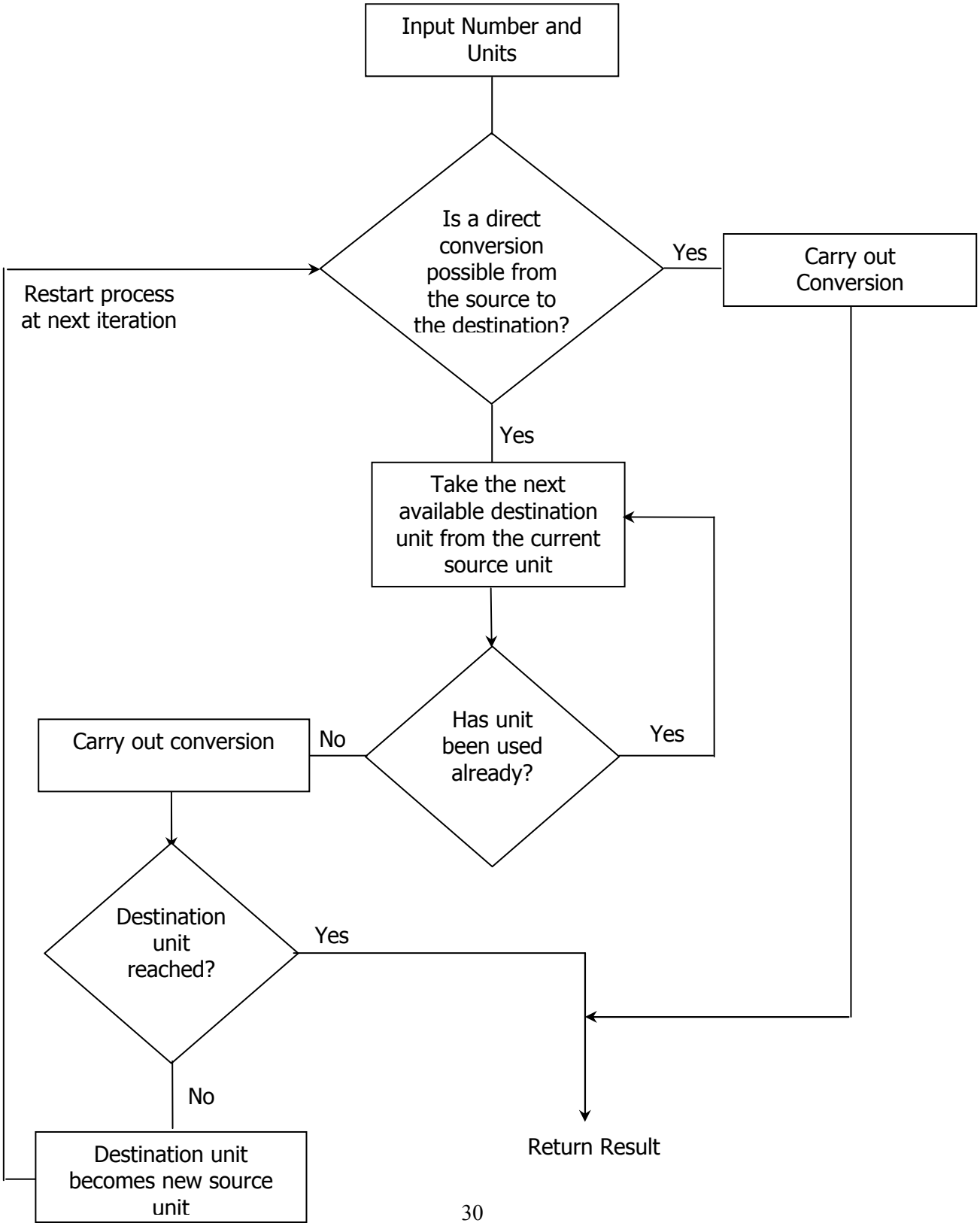
4.4 High Level Design

With the simple case completed it gave me a chance to have a better look at the real structure of the system. At which time I decided there were four key areas that I should focus on:

- Carrying out longer transformations
- Developing a GUI
- Adding content dictionaries
- Reading/Searching multiple dictionaries

The main body of the system will follow the structure of the follow chart over page:

Figure 3. Main System Structure



4.4.1 Carrying out longer transformations

Currently the system searches through all of the possible conversions to try and find an exact match with the source and destination units; however, it could be the case that a conversion is only possible with an intermediate step. For example using the imperial content dictionary a conversion from metres to yards would have to go through feet. The most suitable way of demonstrating this would be with an example:

Search(A, D, amount2conv, visitedlist(0))

Lookup(A)

D Not Found

B and E Returned

Search(B, D, convert(amount2conv, A, B), visitedlist(A))

Lookup(B)

D Not Found

A and C Returned, Ignore A

Search(C, D, Convert(Pre-value, B, C), visitedlist(A,B))

Lookup(C)

D Found

Return Convert(Pre-value, C, D))

This method carries out a conversion from A to D. Initially it searches from a conversion directly from A to D, if that is not possible it will look to see what other conversions are possible, in this case B and E. It then converts from A to B and adds A to the list of units that have already been used. The cycle then continues with an attempt to convert from B to D.

If a direct conversion is not possible then the Search method will enter into another for loop, this will enable us to iterate over all of the conversions that involve the source unit. Although we do not know what the destination unit will be for each intermediate conversion we can extract it using the XPath.

```
Attribute a = (Attribute)
XPath.selectSingleNode(N,"OMA/OMS[2][@name!="'+srcUnit+"']/@name");
String N_string = a.getValue();
```

By making an arbitrary variable (N_string) the new destination unit, the conversion method can now be called for a conversion from the source unit to the N_string. The search method will then call itself with the newly worked out amount to convert and the current destination unit (N_string) as the new source unit. The cycle continues until the conversion is carried out or there are no more conversions available, hence, it is not possible with the given content dictionaries.

There is some administration that needs to take place before this method will work. In the previous example there was a 'been visited' list. This is essential because without checking that a unit has already been used at an intermediate stage it could loop endlessly over the same units.

The beenVisited method is called before any intermediate steps are taken; it takes the list of units that have been visited and compares the contents with the N_string, or new destination unit. If the unit has already been visited, the method returns false and the loop in the search method starts its next iteration. The loop in search then finds another possible destination unit which is again checked by beenVisited, this cycle continues until a suitable destination unit is found or the conversion is just not possible.

```
public static boolean beenVisited(ArrayList visited, String unit)
{
  for(int k=0; k<visited.size(); k++)
  {
     if(visited.get(k).equals(unit))
     {
       return true;
     }
  }
  return false;
}
```

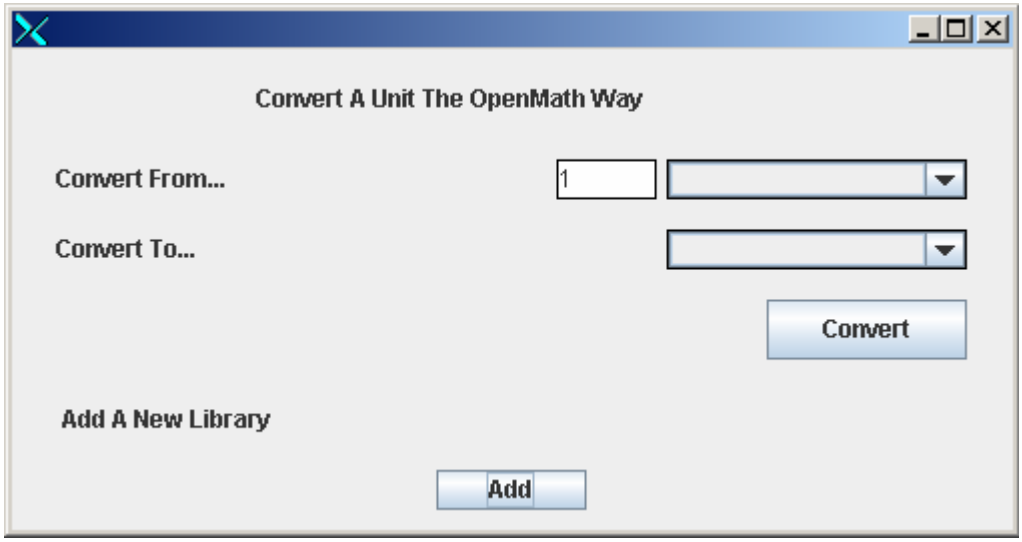
For this process to work successfully, before the search method calls itself to start another iteration the source unit needs to be added to the visited array.

4.4.2 Developing a GUI

Having reviewed some of the existing unit converters in the marketplace I was not going to spend a great deal of time producing the GUI. The GUI has a few basic requirements:

- Must be able to enter a number
- Must be able to select a source and destination unit to convert
- Must have a button for carrying out the conversion
- Must have a button to add content dictionaries
- Must have a way of presenting the conversion answer

Figure 4. Main system GUI




```
public static JTextField convertAmount = new JTextField("1")
public static JComboBox convertFrom = new JComboBox()
public static JComboBox convertTo = new JComboBox()
public static JButton processButton = new JButton("Add")
public static JButton convertButton = new JButton("Convert")
```

I toyed with a variety of approaches for the GUI but felt the best approach would be to keep it as simple as possible. The processes of carrying out a conversion could

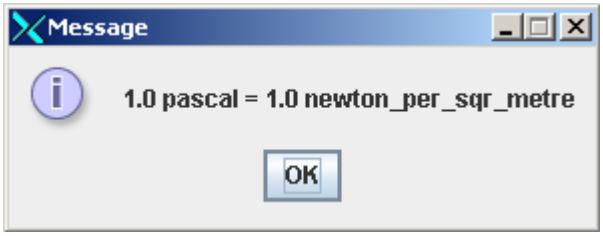
no be easier, enter the amount you would like to convert, select the unit you would like to convert from and to and press the convert button.

Once the convert button has been pressed the system implements an ActionListener and selects the conversion amount from the JTextField, which is then cast to a double. As well as that the AcvtionListener gathers the source and destination units from their Combo Boxes, each of them then being cast to a string. The search method is then called to start the conversion process.

```
if (event.getSource()== convertButton)
{
  ArrayList visited = new ArrayList();
  String srcUnit = (String) convertFrom.getSelectedItem()
  double srcVal = Double.parseDouble(convertAmount.getText().trim())
  String dstUnit = (String) convertTo.getSelectedItem()
  double result = search(srcUnit, dstUnit, srcVal, visited)
}
```

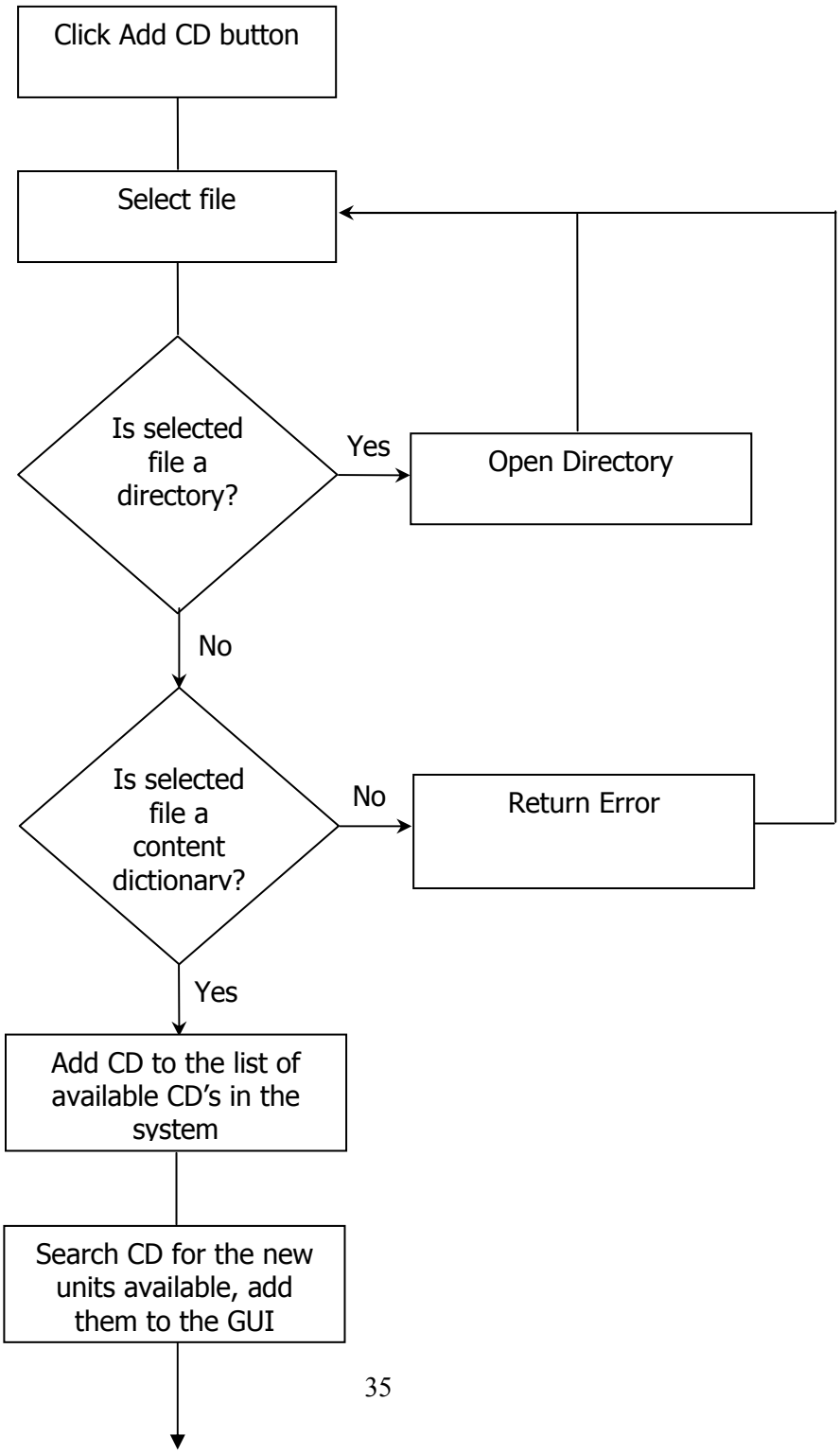
If a solution is found a new JOptionPane is shown with the solution otherwise an error message will appear informing the user that the conversion they are attempting is not possible.

Figure 5. GUI to displaying the result

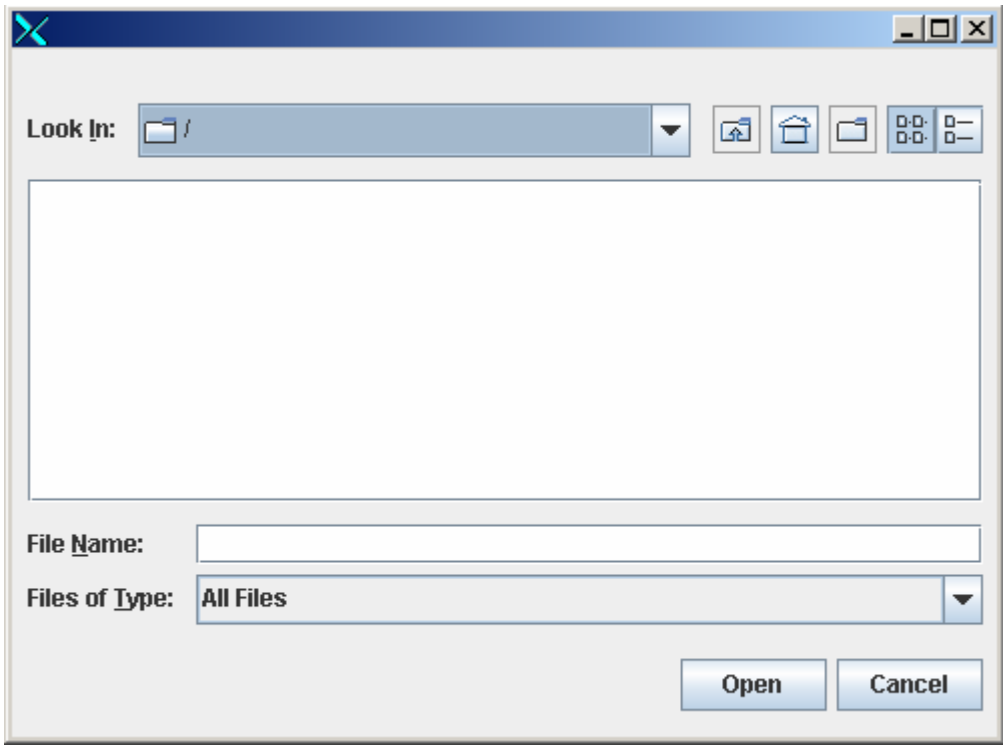


4.4.3 Adding Multiple Dictionaries

Figure 6. Adding a Content Dictionary



The add new library button also has an ActionListener, when this button is pressed a new instance of the class FileChoose() launches. This opens a new JPanel that contains a JFileChooser, a JFileChooser provides a simple mechanism for the user to choose a file.



The file chooser works in the standard manner, the user can search through the directories and when they have found the required content dictionary can use the Open button to load it into the system. When the Open button is pressed the approveSelection() method is called.

Once the file chooser has verified that the file selected is not a directory it will then check the length of the filename and last 4 digits. This is a verification process that checks to ensure the file selected is a content dictionary; the last 4 digits of a content dictionary should always be .ocd. A check for .ocd at the end of the file means if the user has selected the wrong type of file the system can return and error and the user can try again. See the code over page:

```
int filenameLength = filename.length();
if (filenameLength<4 || !filename.substring(filenameLength - 4,
filenameLength).equalsIgnoreCase(".ocd"))
{
  JOptionPane.showMessageDialog(null, "Error, Please add a valid Content
Dictionary")
}
else {
  Unit_Converter.addcd(getSelectedFile());
  chooser.dispose();
}
```

When the correct file has been opened the system will pass the content dictionary into the addcd() method in the Unit Converter class, and close the JPanel. The addcd method then adds the new content dictionary to the list of content dictionaries already in the system and passes the content dictionary into the readCD() method.

The purpose of the readCD method is to update the combo boxes on the GUI; the content dictionary is passed into the method and parsed by the SAXBuilder, XPath is used to extract the names of all of the available units in that content dictionary and they're added to a list of nodes. The list is checked for any duplicates and the resulting list is added to the combo boxes on the GUI, this now makes the new conversions available to the user.

4.4.4 Reading/Searching multiple dictionaries

Now there are multiple dictionaries in the system the search method needs to be edited to ensure all of the content dictionaries are being checked for relevant conversions. The content dictionaries are stored in a list so a loop is necessary at the start of the searching method so that the existing search process can be carried out on every content dictionary.
As well as that the SAXBuilder code needs to be edited so that a different content dictionary is parsed during each iteration, this is done by selecting the content dictionary at the location corresponding to the for loop and casting it to a string.

```
for(int y=0;y<=files.size();y++)
{
   SAXBuilder parser = new SAXBuilder();
   Document response = parser.build((String) files.get(y));
```

5

Testing & Results

5.1 Test Plan

The testing document aims to show that each of outlined requirements have been met successfully, this is going to be split into two parts. Initially I am going to test that my system has met the functional requirements highlighted in the specification stage. I am then going to carry out a usability study with some possible users to analyse the design and usability of the front end.

5.2 Testing Functional Requirements

The functional requirements are a specification stating what the system will and won't do; they are generally yes or no answers so I designed a number of tests to see if the system can carry them out, the results are as follows:

Tests 1 & 2: Requirements 1 & 5: Adding new content dictionaries to the system

Test Carried Out: Add the Pressure content dictionary via the front end

Expected Result: The content dictionary should be added to the system so the units; Pascal, Newton's per square Metre, Pound's per square Metre and Pound's per square Foot should all be made available for conversions

Successful: Yes, see Figure 5.1 in Appendix B

Test Carried Out: Using the front end add a number of different content dictionaries from the OpenMath website

Expected Result: The content dictionaries should be added to the system and all of the expected units are made available for conversions

Successful: No. Some of the content dictionaries where not compatible with the system, as a result the units were added but the conversions are not available.

Test 3, 4 & 5: Requirements 2 & 7: Carry out conversions using just one content dictionary achieving a 99% success rate

Test Carried Out: Using 1 as the conversion amount converted between:

Pascal and Newton's per square Metre

Pascal and Pound's per square Metre

Pascal and Pound's per square Foot

Expected Result: The answer should return for each conversion respectively

Successful: Yes, see Figures 5.2, 5.3, 5.4 in Appendix B

Test 6: Requirement 3: Carry out conversions using multiple units achieving a 99% success rate

Test Carried Out: Using 1 as the conversion amount converted between:

Miles and Metres

Expected Result: Although this conversion isn't formally defined it should use feet as an intermediate step, the answer should return for the conversion

Successful: Yes, see Figure 5.5 in Appendix B

Test 7: Requirements 4 & 8: Provide a simple front-end that collates all of the functionality of the system

Test Carried Out: Run the System

Expected Result: The front end should appear within 2 seconds

Successful: Yes, see Figure 5.6 in Appendix B

Tests 8, 9 & 10: Requirement 6: Error Handling

Test Carried Out: Attempt to carry out a conversion without a conversion amount filled in

Expected Result: An error should be returned

Successful: Yes, see Figure 5.7 in Appendix B

Test Carried Out: Attempt to add a Word file instead of a content dictionary

Expected Result: As the file is not a valid content dictionary an error should be returned

Successful: Yes, see Figure 5.8 in Appendix B

Test Carried Out: Carry out a conversion between Metres and Pascal

Expected Result: The conversion is not possible so an error should be returned

Successful: Yes, see Figure 5.9 in Appendix B

Requirements 9 & 10: Time to execute user commands & Reliability

Test Carried Out: All of the above

Expected Result: All of the tests should be successful and the operation should have provided the user with the desired result within 2 seconds

Successful: Yes

Requirement 11: Robustness

Test Carried Out: Keep the system running for 5 hours carrying out a number of different tests including failure cases

Expected Result: All of the tests should run successfully and in the failure case the system should recover and be operational within 15 seconds

Successful: Yes, the only test that went wrong was when adding certain content dictionaries. Although the dictionaries failed to add properly the system did not crash and was still operational.

5.3 Testing Non-Functional Requirements

The non-functional requirements specify those requirements that cannot be precisely measured by success or failure, they are subjective. For this reason, testing in real-world environments and user feedback often plays a key part in the validation of non-functional requirements.

5.3.1 Usability Study

There are a number of different ways to carry out a usability study; I decided to carry out some short exercises with a sample of possible users. Ten people were given the user guide and asked to 'play' with the system for five to ten minutes. To ensure the users tested all aspects of the system they were given a small list of tasks they were asked to complete; add at least one content dictionary, carry out at least one successful conversion and try to break the system. They were then asked to give it a mark out of ten for:

- How simple and easy to use the front end is
- How useable they felt the system is

Although the design and completeness of the user interface was always secondary in importance to the process of converting units using OpenMath the results of the usability study were still important because even the most advanced unit converter is pointless without a means to interact efficiently with it.

5.3.2 Usability Study Results

After all ten users had tested my system the results came out as follow:

Table 1 Usability Study Results

User No.	1	2	3	4	5	6	7	8	9	10
Front End /10	8	8	9	10	9	8	9	10	8	8
Usability /10	8	7	8	9	8	7	8	7	8	9

The final results were:

- How simple and easy to use the front end is… 8.7/10
- How useable they felt the system is… 7.9/10

Verbal discussions with the different users did not raise any major issues with the design and usability of the front end. Most felt that the system itself was simple enough to navigate and given the user guide as a backup they couldn't foresee any major problems occurring. The reason the usability had a slightly lower score compared with the actual design was because of the error reporting. Not all of the content dictionaries are compatible with the system and when a non-compatible content dictionary is added there was often some confusion because the user is not made aware of this.

5.3.3 Testing Summary

The original goal of this project was to create a system that would demonstrate OpenMath as a suitable foundation for a unit converter and in turn create an extensible unit converter; the requirements specification firmly defined these parameters. The majority of tests carried out on my functional requirements were a success; the testing demonstrated that the system could carry out conversions and add content dictionaries, hence, increasing the number of conversion available.

Despite the fact that that the non-functional requirements are harder to test than the functional requirements, they were also deemed to have been met. The usability study demonstrated that after a host of different users tested the system they felt that the front end and system usability were both more than adequate.

The main problem highlighted during both rounds of testing was the fact that not all content dictionaries are compatible with the system, this is discussed further in the conclusion.

6

Conclusions

6.1 Project Critique

At the beginning of this project the task set to me was to research the OpenMath language and investigate its suitability as the foundation for an extensible unit converter. This broke down into two major sections, the inclusion of content dictionaries into the system and evaluating the input to return the answer or an error.

I set out studying OpenMath as a language and given my lack of knowledge on OpenMath and XML followed what I deemed to be quite a sensible development process. By carrying out my implementation in an evolutionary manner it meant that as I developed the system I could learn about some of the more advanced features and incorporate them in the system at a later date.

The main draw back of an evolutionary approach is that it can be very easy to lose control of the development process. By carrying out quite an informal approach to evolutionary design and implementation it has had a negative affect on my coding. Java programs by definition are generally highly modularised with clear class diagrams and code layout. Unfortunately this is not the case within my system, a number of the methods within the main body of the system should really be separated into different classes; the program as a whole could really do with being modularised.

Due to my coding it could mean that future development of the system could well be slowed. Not only will it make it a lot harder for other programmers to read through and understand the processes I've implemented thus far but future changes will also be more difficult to incorporate. One of my non-functional requirements was to ensure that future improvements were catered for in the design, hence, new operations and modifications to improve efficiency should be easy to add to the system. Although I feel a better system was produced due to my implementation style it has resulted in this requirement not really being met.

I enjoyed researching and implementing this project, OpenMath is a fascinating language and it was refreshing to work in a field that still has scope for development.

To critique myself I would say that I have achieved my primary objectives. Although the system is limited by the number of content dictionaries it is compatible with, it is capable of reading in a variety of content dictionaries and based on the details stored in those dictionaries, carry out conversions.

6.2 The Future – The Unit Converter

6.2.1 Writing Content Dictionaries

One of the main disadvantages of OpenMath is that as yet is does not have mass appeal and as a result there is not wide spread understand of the language. The unique selling point behind an OpenMath based unit converter is that it is extensible. However, at this moment in time the only way users can add content dictionaries to the system is to search through the list of dictionaries, gather the required information and add the dictionary that relates to the conversion they want to perform. Although the content dictionaries are written in XML and that is designed to be both machine and human readable this is not a particularly practical approach for non-computer literate users.

To ensure all types of users fully utilise the system a sensible addition would be a section of the program that can write content dictionaries for the user. Phrase-books are available to convert other languages to OpenMath, for example:

```
OMA>
<OMS  cd="Basic" name="segment"/>
1
2
</OMA>
```

Is written by:

```
OMApplication om1 = new OMApplication(functor1,params1);
    Vector params2 = new Vector();
    params2.addElement(new OMInteger(1));
    params2.addElement(new OMInteger(3));
```

This means that if a conversion is not already available the user could enter the two units they want to convert between as well as some of the core conversion details, a phrase-book could then be used to produce the content dictionary on their behalf.

6.2.2 Formalising a Phrase-book

Suppose you have two applications communicating by sending OpenMath objects to each other, a client program and a computational server. It is very unlikely that the applications internal data structure will be OpenMath, hence, translations between the internal representation and OpenMath are going to have to take place. The piece of software that does this is usually called the phrase-book.

It is possible to write a phrase-book which can handle any piece of OpenMath, in practice this means you can write an OpenMath phrase-book that will handle a fixed set of content dictionaries and symbols. What "handle" means will vary from case to case: in my system it attempts to evaluate the input and return a result or an error.

At present my system is interpreting content dictionaries using a standard XPath location. This enables access to a number of different units and conversions but only those of the same format. If I were to continue developing this project the main area worth developing is to write a formal phrase-book. Within the phrase-book I would be able to define a number of ways of interpreting the content dictionaries and in doing so I would be able to access a great deal more conversions increasing the extensibility of the converter.

6.3 The Future – OpenMath

OpenMath avoids defining what the "right" behaviour is in a given circumstance and leaves that up to the phrase-book writer, it is also possible that a piece of software could have multiple phrase-books associated with it for different purposes (The OpenMath Society, 2006). This means that a whole host of different systems can be produced that utilize a mixture of content dictionaries.

There are a number of different content dictionaries already written and available on the Internet that cover a variety of topics. A good example of this is the dimension1 content dictionary. This content dictionary defines units such as length, area, velocity, time, speed etc. These content dictionaries really highlight the advantages of OpenMath, using a phrase-book that can interpret these content dictionaries a mathematical tutor or teaching aid could be produced. A mathematical tutor would be able to read the content dictionaries and return the relations in a more legible manner; furthermore the content dictionaries can be used to work out the result of equations for the user. For Example, given the length and time travelled the system can work out the acceleration.

6.4 Summary

OpenMath as a language is highly underutilised, it is an excellent way of formally defining the semantics of and relations between mathematical objects and as a result I can see the use of the language increasing greatly over the coming years. Due to the way that OpenMath is formalised it provides the perfect foundation for a mathematical system to build on. As more OpenMath based applications are being produced it means there will be more content dictionaries and phrase-books being built. This in turn will increase the understanding, functionality and usability of the language, greatly increasing its popularity.

Bibliography

- Bray, T., Paoli, J., 1998
  Extensible Markup Language (XML) 1.0, Available at:
  http://www.w3.org/TR/1998/REC-xml-19980210.pdf
- Buswell. S., Caprotti, O., Carlisle, D.P., Dewar, M.C., Gaëtano, M. and Kohlhase, M., 2004
  The OpenMath Standard. Available at:
  http://www.openmath.org/standard/om20-2004-06-30//omstd20.pdf
- Carlisle, D., Davenport, J. Dewar, M., Hur, N., Naylor, W., 2001
  Conversion between MathML and OpenMath, Available at:
  http://www.openmath.org/documents/om-mml.pdf
- Cohen, M.A, Cuypers, H., Sterk, H.
  Algebra Interactive Available at:
  http://www.win.tue.nl/~ida/home.html
- Davenport, J., 1999
  A Small OpenMath Type System, Available at:
  http://www.openmath.org/standard/sts.pdf
- Davenport. J, 2000
  On Writing OpenMath Content Dictionaries. Available at:
  http://www.openmath.org/documents/writingCDs.pdf
- Davenport, J., Naylor, W., 2003
  Units and Dimensions in OpenMath, Available at:
  http://www.openmath.org/documents/Units.pdf
- Harold, E.R., Means, W.S., 2001
  XML: In a Nutshell, United States: O'Reilly & Associates, Inc
- McLaughlin, B., 2001
  Java & XML, 2nd Edition, United States: O'Reilly & Associates, Inc

- Neimeyer, P., Knudsen, J., 2005
  Learning Java, 3rd Edition, United States: O'Reilly Media, Inc
- OpenMath Consortium, 1999
  The OpenMath Project Final Report, Available at:
  http://www.openmath.org/projects/esprit/final/index.html
- Piroumian, V., 1999
  Java GUI Development, United States: Sams Publishing
- Seppala, M., 1995
  Report of OpenMath Activities, Editing and Computing. Available at:
  http://www.openmath.org/projects/EditingAndComputingReport.pdf
- Sommerville, I., 2001
  Software Engineering, 6th Edition, United States: Addison Wesley
- Stiller, E., LeBlanc, C., 2002
  Project Based Software Engineering, United States: Addison Wesley
- Van Der Vlist, E., 2002
  XML Schema, United States: O'Reilly & Associates, Inc
- Walsh, N., 1998
  A Technical Introduction to XML, Available at:
  http://www.xml.com
- W3C, 2005
  Document Object Model (DOM), Available at:
  http://www.w3.org/DOM/
- Yu Terrance, 1998
  Poly Math, Available at:
  http://pdg.cecm.sfu.ca/openmath0.5/lib/phrasebook.html

Appendix A

User Documentation

Appendix B

Raw results output

Figure 7. Test 1 Results

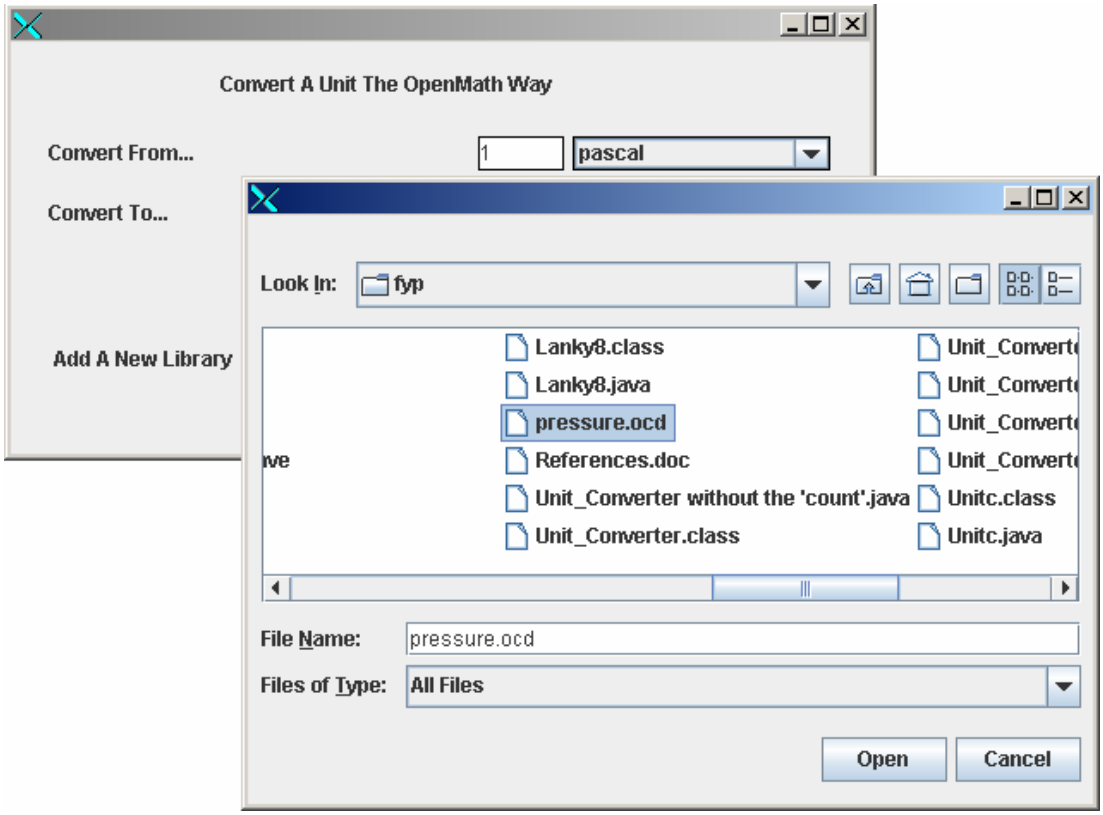


Figure 8. Test 3 Results

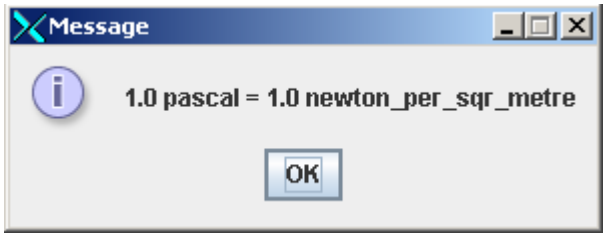


Figure 9. Test 4 Results

Figure 10. Test 5 Results

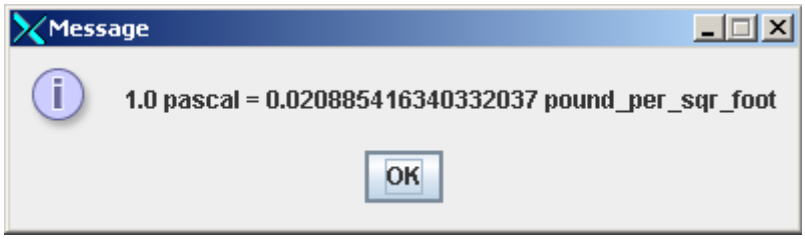


Figure 11. Test 6 Results

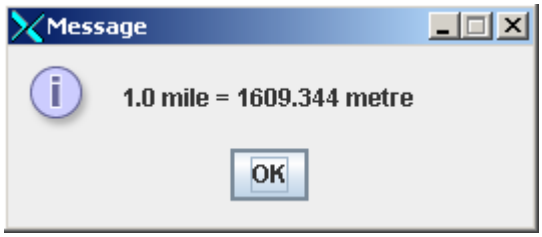


Figure 12. Test 7 Results

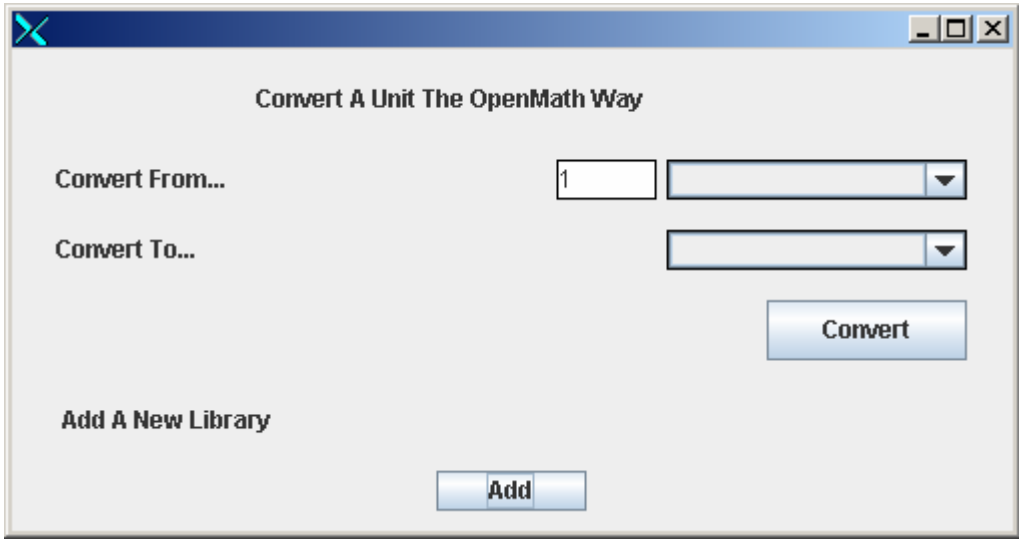


Figure 13. Test 8 Results

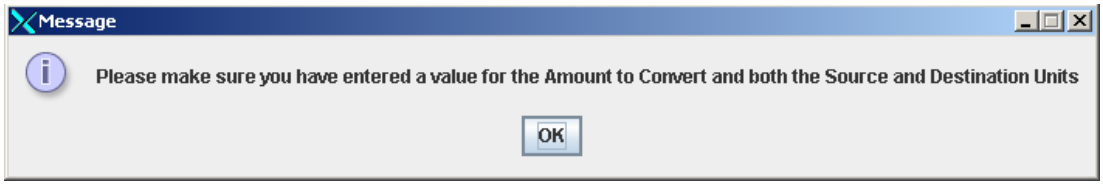
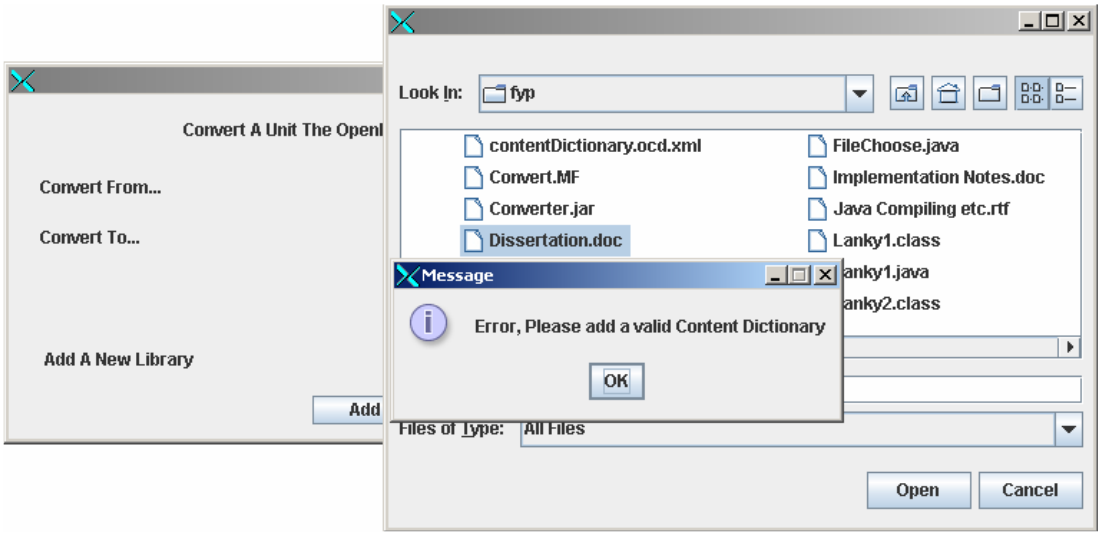


Figure 14. Test 9 Results

Figure 15. Test 10 Results



Appendix C

Code

Pressure.ocd

```
<CD>
<CDName> pressure </CDName>
<CDStatus> experimental </CDStatus>
<CDDate> 2007-04-11 </CDDate>
<CDVersion> 1 </CDVersion>
<CDUses>
<CDName>arith1</CDName>
<CDName>relation1</CDName>
</CDUses>

<Description>
This CD defines symbols to represent pressure measurements.
</Description>

<CDDefinition>
<Name> pascal </Name>
<Description>
This symbol represents the measure of one Newton per square metre.
This is the standard SI measure for pressure.
</Description>

<CMP> 1 pascal = 1 newton_per_sqr_metre </CMP>
<FMP><OMOBJ>
  <OMA>
    <OMS name="eq" cd="relation1"/>
    <OMA>
      <OMS name="times" cd="arith1"/>
      <OMI> 1 </OMI>
      <OMS name="pascal" cd="pressure"/>
    </OMA>
    <OMA>
      <OMS name="times" cd="arith1"/>
      <OMI> 1 </OMI>
      <OMS name="newton_per_sqr_metre" cd="pressure"/>
    </OMA>
  </OMA>
</OMOBJ></FMP>

</CDDefinition>

<CDDefinition>
<Name> pound_per_sqr_metre </Name>
<Description>
This symbol represents the measure of one Pound per square metre.
This is the standard SI measure for pressure.
</Description>
<CMP> 1 pound_per_sqr_metre  = 4.4483 pascal </CMP>

<FMP><OMOBJ>
```

```
  <OMA>
    <OMS name="eq" cd="relation1"/>
    <OMA>
      <OMS name="times" cd="arith1"/>
      <OMI> 1 </OMI>
      <OMS name="pound_per_sqr_metre" cd="pressure"/>
    </OMA>
    <OMA>
      <OMS name="times" cd="arith1"/>
      <OMF> 4.4483 </OMF>
      <OMS name="pascal" cd="pressure"/>
    </OMA>
  </OMA>
</OMOBJ></FMP>

</CDDefinition>

<CDDefinition>
<Name> pound_per_sqr_foot </Name>
<Description>
This symbol represents the measure of one Pound per square foot.
This is the standard SI measure for pressure.
</Description>
<CMP> 1 pound_per_sqr_foot  = 47.8803 pascal </CMP>

<FMP><OMOBJ>
  <OMA>
    <OMS name="eq" cd="relation1"/>
    <OMA>
      <OMS name="times" cd="arith1"/>
      <OMI> 1 </OMI>
      <OMS name="pound_per_sqr_foot" cd="pressure"/>
    </OMA>
    <OMA>
      <OMS name="times" cd="arith1"/>
      <OMF> 47.8803 </OMF>
      <OMS name="pascal" cd="pressure"/>
    </OMA>
  </OMA>
</OMOBJ></FMP>

</CDDefinition>

</CD>
```

FileChoose.java

```
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import javax.swing.*;

public class FileChoose extends JFrame
{
  public FileChoose()
  {
    super("");
    JPanel pane = new JPanel(new FlowLayout());

    Dave fc = new Dave(new File("."), this);
    Box box = Box.createVerticalBox();
    box.add(Box.createRigidArea(new Dimension(0, 10)));
    box.add(fc);
    pane.add(box);
    setSize(500, 370);
    setLocation(400,200);
    setContentPane(pane);
  }

  public void launchFrame()
  {
    setVisible(true);
  }

 private class Dave extends JFileChooser
 {
   FileChoose chooser;
   public Dave(File f, FileChoose chooser)
   {
     super(f);
     this.chooser = chooser;
   }
   public void approveSelection()
   {
   // If the selected file is a directory open it
   if (getSelectedFile().isDirectory()) {
     setCurrentDirectory(getSelectedFile());
   }
   else {
   // Otherwise check to make sure the last 4 digits are .ocd, if so
add the CD to the system.
     String filename = getSelectedFile().toString();
     int filenameLength = filename.length();
     if (filenameLength<4 || !filename.substring(filenameLength - 4,
filenameLength).equalsIgnoreCase(".ocd"))
     {
```

```
        // If the last 4 digits are not .ocd display an error
message.
        JOptionPane.showMessageDialog(null, "Error, Please add a
valid Content Dictionary");
     }
     else {
       Unit_Converter.addcd(getSelectedFile());
       chooser.dispose();
     }
   }

   }
   public void cancelSelection()
   {
     chooser.dispose();
   }
 }
}
```

Unit_Converter.java

```
import java.util.*;
import java.net.*;
import java.io.*;
import org.jdom.*;
import org.jdom.xpath.*;
import org.jdom.output.XMLOutputter;
import org.jdom.input.SAXBuilder;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;

public class Unit_Converter extends JFrame implements ActionListener
{
  // Initialise global variables
  public static JFrame frame = new JFrame();
  public static JPanel screen = new JPanel(new FlowLayout());
  public static JTextField convertAmount = new JTextField("1");
  public static JComboBox convertFrom = new JComboBox();
  public static JComboBox convertTo = new JComboBox();
  public static JButton processButton = new JButton("Add");
  public static JButton convertButton = new JButton("Convert");
  public static JFileChooser addNewLibrary = new JFileChooser(new
File("."));
  public static java.util.List files = new LinkedList();

  // Add the new Content Dictionary to the current list
  public static void addcd(File f)
  {
    files.add(f.toString());
    readCD(f.toString());
  }

  // Invert the operator applied to the srcUnit
  public static String invertOp(String operator)
  {
     if (operator.equals("times")) return "divide";
     if (operator.equals("divide")) return "times";
     if (operator.equals("add")) return "minus";
     if (operator.equals("minus")) return "add";
     return "unknown";
  }

  // Carry out the operator applied to the srcUnit
  public static double normalise(double srcVal, double baseVal,
String operator)
  {
     if (operator.equals("divide")) {
```

```
        return srcVal/baseVal;
     }
     return 0;
  }

  // Carry out the final operation
  public static double apply(double arg1, double arg2, String
operator)
  {
     if (operator.equals("times")) {
        return arg1*arg2;
     }
     return 0;
  }

  // Find the value within the Element, whether thats the source or
destination Element
  public static double getValue(Element oma)
  {
    // Search the OMA for the element value, this method makes it
independent of whether it's a float or integer
    try {
      Element ele;
      String value;
      // Integer
      ele = ((Element)XPath.selectSingleNode(oma,"OMI"));
      if (ele != null) {
         // Trim the content dictionary value
         value = ele.getValue().trim();
         return (double) Integer.parseInt(value);
      }
      // Float
      ele = ((Element)XPath.selectSingleNode(oma,"OMF"));
      if (ele != null) {
         value = ele.getValue().trim();
         return (double) Double.parseDouble(value);
      }
    }
    catch (Exception e) {
    }
    return 0.0;
  }

  // Do the maths
  public static double convert(Element el, String sourceUnit, String
destinationUnit, double value)
  {
    double result = 0.0;
    try {
        // Search the list to find the section that focuses on the
srcUnit & dstUnit
        // Match the Source and Destination values
```

```
        Element srcOma = (Element)
XPath.selectSingleNode(el,"OMA[OMS/@name='"+sourceUnit+"']");
        Element dstOma = (Element)
XPath.selectSingleNode(el,"OMA[OMS/@name='"+destinationUnit+"']");

        // Focuses on the source unit
        // Call the invertOp method - reverse the operation thats
applied to the source unit
        String invOp =
invertOp(((Attribute)XPath.selectSingleNode(srcOma,"OMS/@name")).get
Value());
        // Make srcBase a double
        double dblBase = getValue(srcOma); //(double)
Integer.parseInt(srcBase);
        // Carry out the normalise method
        double norm = normalise(value,dblBase,invOp);

        // Retrieve the destination value from the content
dictionary
        String dstOp   =
((Attribute)XPath.selectSingleNode(dstOma,"OMS/@name")).getValue();
        // Make dstBase a double
        dblBase        = getValue(dstOma); //(double)
Integer.parseInt(dstBase);

        // Carry out the apply method
        result = apply(norm,dblBase,dstOp);
    }
    catch (Exception e) {
      System.err.println(e);
      e.printStackTrace();
    }
    return result;
  }

  // Search for the correct conversion
  public static double search(String srcUnit, String dstUnit, double
srcVal, ArrayList visited)
  {
  // This loop ensures the method searches through all of the
  // content dictionaries in the system.
  for(int y=0;y<=files.size();y++)
  {
    try {
      SAXBuilder parser = new SAXBuilder();
      Document response = parser.build((String) files.get(y));

      // sourceNodes = lookup source unit
      // Extract all sections that focus on the srcUnit, add them to
a list
```

```
      XPath x =
XPath.newInstance("/CD/CDDefinition/FMP/OMOBJ/OMA[OMA/OMS/@name='"+s
rcUnit+"']");
      java.util.List sourceNodes = x.selectNodes(response);
      ArrayList srcNodes = (ArrayList) sourceNodes;

      // loop over all source units in list
      // for all nodes in sourceNodes
      for(int i=0;i<srcNodes.size();i++)
      {
        // Search the list to find the section that focuses on the
srcUnit & dstUnit
        Element el = (Element) srcNodes.get(i);
        Element dstOma = (Element)
XPath.selectSingleNode(el,"OMA[OMS/@name='"+dstUnit+"']");
        if (dstOma != null) {
          // If there is a match for the dstUnit, Carry out the
conversion method
          return convert(el, srcUnit, dstUnit, srcVal);
        }
      }

      // for all nodes N in sourceNodes
      for(int j=0;j<srcNodes.size(); j++)
      {
        // if srcUnit is not in visited
        if (!beenVisited(visited, srcUnit))
        {
          // Return the attribute in the element that isn't the
source unit
          Element N = (Element) srcNodes.get(j);
          Attribute a = (Attribute)
XPath.selectSingleNode(N,"OMA/OMS[2][@name!='"+srcUnit+"']/@name");
          // N_string is out new element, it becomes the new dstUnit
          String N_string = a.getValue();
          // A conversion is carried out from the srcUnit to N-
string
          double newSrcVal = convert(N, srcUnit, N_string, srcVal);
          // Every srcUnit is added to a list of all units visited
to no conversions are repeated
          visited.add(srcUnit);
          // N_string is now the srcUnit
          // Search is called again to try and find a conversion
from N_string to the original dstUnit
          double val = search(N_string, dstUnit, newSrcVal,
visited);
          if (val != -999)
          {
            return val;
          }
        }
      }
```

```
    }
    catch (Exception e) {
      System.err.println(e);
    }
   }
     return -999;
  }

  // Do a check to see if the unit is in the visited array list
  // This means it should be ignored
  public static boolean beenVisited(ArrayList visited, String unit)
  {
     for(int k=0; k<visited.size(); k++)
     {
        if(visited.get(k).equals(unit))
        {
          return true;
        }
     }
     return false;
  }

  // read new Content Dictionaries into the converter
  public static void readCD(String nameCD)
  {
    try {
      // Read in the file
      SAXBuilder parser = new SAXBuilder();
      Document cd = parser.build(nameCD);
      // Extract the information from the correct path and add the
names to the ArrayList
      XPath x =
XPath.newInstance("/CD/CDDefinition/FMP/OMOBJ/OMA/OMA/OMS[2]/@name")
;
      java.util.List nodes = x.selectNodes(cd);
      ArrayList allNodes = (ArrayList) nodes;

      // Search through the list removing any duplicates
      for(int k=0;k<allNodes.size();k++)
      {
        String name = ((Attribute) allNodes.get(k)).getValue();
        for (int d=(k+1);d<allNodes.size();d++)
        {
          String compare = ((Attribute) allNodes.get(d)).getValue();
          if (compare.equals(name))
          {
            allNodes.remove(d);
            allNodes.trimToSize();
          }
        }
        convertFrom.addItem(name);
        convertTo.addItem(name);
```

```
      }

   }
    catch (Exception e) {
      System.err.println(e);
    }
  }

  public Unit_Converter()
  {
      // Initialise the Instance Variables
      JPanel screen = new JPanel(new FlowLayout());
      JLabel title = new JLabel("Convert A Unit The OpenMath Way");
      JLabel convertF = new JLabel("Convert From...");
      JLabel convertT = new JLabel("Convert To...");
      JLabel addNewL = new JLabel("Add A New Library");
      JLabel spacer = new JLabel();

      // Set the sizes of the variables
      screen.setPreferredSize(new Dimension(500, 240));

      title.setPreferredSize(new Dimension(255, 40));
      convertF.setPreferredSize(new Dimension(245, 30));
      convertAmount.setPreferredSize(new Dimension(50, 20));
      convertFrom.setPreferredSize(new Dimension(150, 20));
      convertT.setPreferredSize(new Dimension(300, 30));
      convertTo.setPreferredSize(new Dimension(150, 20));
      addNewL.setPreferredSize(new Dimension(450, 40));
      spacer.setPreferredSize(new Dimension(350, 40));
      processButton.setPreferredSize(new Dimension(75, 20));
      convertButton.setPreferredSize(new Dimension(100, 30));

      // Set borders on the TextPanes
      convertAmount.setBorder(new LineBorder(Color.black));
      convertFrom.setBorder(new LineBorder(Color.black));
      convertTo.setBorder(new LineBorder(Color.black));

      // Add the elements to the Screen
      screen.add(title);
      screen.add(convertF);
      screen.add(convertAmount);
      screen.add(convertFrom);
      screen.add(convertT);
      screen.add(convertTo);
      screen.add(spacer);
      screen.add(convertButton);
      screen.add(addNewL);
      screen.add(processButton);

      frame.setTitle("");
      frame.show();
      frame.setLocation(400, 200);
```

```
      frame.setContentPane(screen);
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.pack();

      // Add action listeners to the relevant parts of the GUI
      convertFrom.addActionListener(this);
      convertButton.addActionListener(this);
      processButton.addActionListener(this);

    }

    public void actionPerformed(ActionEvent event)
    {
      if (event.getSource()== convertButton)
      {
        try {
          // Read in the details and start the conversion process
          ArrayList visited = new ArrayList();
          String srcUnit = (String) convertFrom.getSelectedItem();
          double srcVal  =
Double.parseDouble(convertAmount.getText().trim());
          String dstUnit = (String) convertTo.getSelectedItem();
          double result = search(srcUnit, dstUnit, srcVal, visited);
          // Return the result
          if (result != -999)
          {
            JOptionPane.showMessageDialog(null, srcVal + " " +
srcUnit + " = " + result + " " + dstUnit);
          }
          else {
            JOptionPane.showMessageDialog(null, "Sorry, that
conversion is not mathematically possible");
          }
        }
        catch (Exception e) {
          JOptionPane.showMessageDialog(null, "Please make sure you
have entered a value for the Amount to Convert and both the Source
and Destination Units");
        }
      }

      if (event.getSource()== processButton)
      {
        new FileChoose().launchFrame();
      }

    }

    public static void main(String[] args) {
       Unit_Converter uc = new Unit_Converter();
    }
}
```