The DocConversion framework.

Eric D. Schabell
ericS@cs.kun.nl

University of Nijmegen, Computing Science Institute, P.O. Box
9010, 6500 GL Nijmegen, The Netherlands

Abstract

Continuing where the *PRONIR Conversion Clearinghouse* left off (Sch03), this technical report presents the basic framework of the DocConversion tool. In its first phase our DocConversion tool will support single step document conversions, provide a Broker that negotiates the conversion process and a conversion Server. This is the first of three phases to be traversed on our way to a complete document conversion system. This work is part of the ongoing research project *Profile Based Retrieval Of Networked Information Resources (PRONIR)*.

0.1 Introduction

Our search for a a generic information retrieval software architecture has been outlined previously in (Sch02). Continuing work on this project has resulted in (GPB03b), (GS03) and (GPB03a).

The ultimate goal of this part of the overall software architecture is to provide a comprehensive document conversion system, one that works both locally and across networked environments[1]. To meet this goal, the four previously discussed *Phases* have been defined to allow for incremental evolution of the provided conversion functionality. For clarity, an overview of all four *Phases* will be provided first with a detailed explanation of *Phase one* being covered in the rest of this report.

0.1.1 Phase zero

A web based portal has been created to offer access to a collection of existing one step conversion routines. It also provides a mechanism for submitting conversion routines for addition to the collection. Finally, this web portal provides a means for a browsing client to obtain needed conversion tools from the conversion clearinghouse collection (Sch03).

0.1.2 Phase one

Phase one adds to the functionality of the conversion clearinghouse. Here we want to build a simple document conversion system that can make use of the one step conversion tools available in phase zero. This phase will be covered in detail in the rest of this report.

0.1.3 Phase two

This phase will expand on the previous one to include additional *recipes*. A conversion process should not only take a document from a to b, but also provide for possible a to b to ... to n. The client will be able to make use of these conversions without regard for quality of the conversion.

0.1.4 Phase three

At this stage, we will provide for a more complex management of the conversion process between a user and one or more servers in the conversion network.

The Broker will be expanded to provide for more functionality. The network will be able to host multiple Servers with each participating Server providing a listing of its own conversion services.The Broker will be able to negotiate for the client in a network of conversion servers, attempting to offer the quickest (distance to server) conversion. The Broker will offer the most efficient (quality comes into play) conversion possible.

[1] As soon as hosting is arranged, a central conversion server will be made available.

Finally, conversion results should be cached. This caching will not be of the converted document (higher level component caching), but at a more elemental level (low level component caching) as described in (Sch02).

0.1.5 Building blocks

The entire DocConversion tool is implemented in C++, currently consisting of three classes; Server, Broker and Document. It is able to make use of the *PRONIR Conversion Clearinghouse* PostgreSQL database. The various existing conversions available in the *PRONIR Conversion Clearinghouse* were the starting point for our offered conversion system. These will involve only single step conversions, ones that move a document from one type to another. For example, postscript to pdf format.

Currently there are several supporting libraries that are depended upon to provide certain functionalities. For retrieving the documents for conversion, there is support for both Curl(CUR03) and Wget(GNU03). One or the other is a requirement. Furthermore, to connect and interact with the conversion database we make use of the library libpqxx(Lib03) which provides a very clean API.

A Broker and Server interface has been created to provide the user with a seamless conversion experience. This can occur locally or invisibly via a networked conversion server. The Document class is used to provide a mechanism to maintain the state information with regards to the document being converted during its life cycle in the conversion system.

DocConversion has been added to the source-based Linux distribution Source Mage GNU/Linux(Sou03) as a software package. The package management software provides a dependency resolution feature that allows a user to manage the install of not only the software itself (configure and build), but also to setup the conversion database (tables created, user, passwords and fill with data). Our hope is to create a wider test base of users through Source Mage GNU/Linux.

0.2 Design

The first design decision was to to apply parts of the *Extreme Programming (XP)* technique as outlined by (BF00). One of the more interesting aspects of applying XP techniques we found to be the concept of *Just-In-Time design* (Mil02). This is the idea of designing only for what you need at a particular moment to satisfy the presented testing criteria, usually a use case.

In the following section we will present the various use cases that define the phase one functional requirements. Following this, the implementation will be discussed for each and every use case, showing the tests written and eventual code additions to the needed classes. At the end of the implementation we will provide class diagrams and figures detailing the resulting framework.

0.2.1 Requirements

The DocConversion requirements have been distilled in the form of use cases. Each use case is the basic form for one requirement. To validate each require-

ment we will show the unit test(s) written for each use case. These are kept constantly running at timely intervals during our development cycle to ensure that any new changes do not break existing code or functionality. First we present the phase one requirements in the form of use cases.

0.2.2 Use cases

The following table shows the use cases as defined for completing DocConversion phase one functionality. The use cases are all numbered, have and Action as seen from the users point of view and include a few Remarks providing some clarity where necessary:

Number	Action	Remarks
1.	The user can request the use of a specific conversion server for her conversion requests.	The user must submit a valid IP address or resolvable hostname.
2.	The user should be able to retrieve a complete list of all of available conversion routines that a server has to offer.	Display complete listing of available server conversions.
3.	Multiple users should be able to access the conversion server at any given time.	Allow for multiple Brokers and keep track of them.
4.	The user can submit a specific request for a conversion on a document.	The user must submit a valid conversion routine and a valid URI referencing the document to be converted.
5.	All converted documents should be made available on the users local machine.	Report all work done in /tmp directory.
6.	The user should be able to query for possible conversion routines based on representation types (pdf, ps, text, html). The user should be able to specify a 'from' type and a 'to' type which will result in a list of possible conversion routine(s).	Supply a list of available conversion routines based on give 'from' and 'to' representation types.
7.	The user should be able to query for possible conversion routines based on feature types (binary, full-content, abstract, summary). The user should be able to specify a 'from' type and a 'to' type which will result in a list of possible conversion routine(s).	Supply a list of available conversion routines based on give 'from' and 'to' feature types.

0.2.3 Use case details

The following is a more detailed description for each use case, with extra information or comments relating to the input and output from each use case.

Use case 1

This is related to a future vision that will allow a user to select any available conversion server, be that one located on-site or on the other side of the world. The user can specify either an IP address or a resolvable hostname.

The input from a user is an IP/hostname, the resulting output should be a Broker that has registered the given Server as its location for conversion requests. Default is to use the localhost for all conversions.

Use case 2

The user wants to know what the conversion server has to offer before submitting a request. Who knows, this might not be a server you want to even consider using if it can only convert HTML to text?

The user inputs some flag to request a listing of the servers database. If the database has been setup (this is optional during initial software build), then it will provide a listing of the conversions available. Should the database not be setup, then a message will be returned stating this fact.

Use case 3

There should be no limit (other than the machines physical capacity) to the number of Brokers allowed to be working for users at any one time.

Users should not notice the fact that more than one conversion Broker is working at a time. The internal workings will have to keep track of the number of existing Brokers and this is then accessable for reporting/limiting the systems load.

Use case 4

A request for a specific conversion routine along with a URI reference to the location of the document should result in the retrieval, conversion and reporting of process back to the user.

Input from the user is in the form of a valid conversion routine name along with the URI reference to the actual document for conversion. This will result in the retrieval of the given document, attempted conversion of the document through use of the given conversion routine (with out verification of document compatibility) and reporting back to the user as to the location of the conversion.

Use case 5

All conversions should be made available to the user locally.

After a conversion is completed the user will be able to obtain the converted document locally. As of this writing we will place all results in the /tmp directory.

Use case 6

A request for possible conversion routines (currently only for one-step conversions) should be possible in the form of 'from' a documents feature type, 'to' a documents feature type. A few examples of feature types are binary, full-content, abstract, summary, etc.

The input will be defined by offering flags to identify whether the user would like to search on feature types consisting of a valid 'from' type to a valid 'to' type. The results will be a list of available conversion routine(s) as described in the available database. As noted in use case 2, if the database has been setup (this is optional during initial software build), then it will provide a listing of the conversions available. Should the database not be setup, then a message will be returned stating this fact.

Use case 7

A request for possible conversion routines (currently only for one-step conversions) should be possible in the form of 'from' a documents representation type, 'to' a documents representation type. A few examples of representation types are pdf, postscript, text, html, etc.

The input will be defined by offering flags to identify whether the user would like to search on representation types consisting of a valid 'from' type to a valid 'to' type. The results will be a list of available conversion routine(s) as described in the available database. As noted in use case 2, if the database has been setup (this is optional during initial software build), then it will provide a listing of the conversions available. Should the database not be setup, then a message will be returned stating this fact.

0.3 Implementation

In this section the coded implementations of the previously shown design will be provided. All code examples have been pushed through Doxygen for document generation and it should be noted that care was taken to document the code as concisely as possible. This was done with an eye on the possible help that can be obtained by releasing this as a free software project and making the learning curve as gentle as possible.

0.3.1 Unit tests

Our first step after examining the use cases was to write a complete unit test for the use cases required functionality. To facilitate the management and running of unit tests it is desireable to use existing frameworks to manage the testing overhead. A good starting point for taking a look at the C++ unit testing frameworks is found at (Jef03). We have chosen to use the Unit++ Testing Framework (Uni03) based on ease of use, simplicity and good initial documentation.

Each test, created based on a single use case, remains in place during code actual code development and continues to monitor the use case functionality in the future. This is an ongoing quality insurance that ensure any changes in

the future that affect this particular functionality are caught immediately by the unit test(s)[2].

Use case 1

The sole purpose of this unit test is to ensure that a Broker will be created that can handle a remote conversion server request. The default is to use the localhost so we see here that a Broker is created with the name of the remote conversion server passed as an argument. A second method is needed to check what the Broker as filed away as the conversion server it will be dealing with.

Listing 1: Unit test for use case 1

```
//-----------------------------------------------------------------------
//Purpose:          Test the construct of remote Server request.
//Preconditions:    none.
//Postconditions:   server name checked.
//Arguments:        none.
//Returns:          0.
//Called Funcs:     Broker(), Broker.getServerName(), assert_true().
//-----------------------------------------------------------------------
void testBrokerRemoteServer()
{
    Broker myBroker( "my.conversion.server" );
    string myServerName = myBroker.getServerName();
    if ( myServerName == "my.conversion.server" )
    {
        assert_true( "broker remote server check", true );
    }
    else
    {
        assert_true( "broker remote server check", false );
    }
}
```

Use case 2

To test our listing request, we have created a Broker, used a method to call a listing request and if the returned query from our database is not empty we consider it a passed test. This should enable this test to run unchanged in the future without regard for what tuples are in the database. Note that the Broker has been chosen to intercept a request and check itself. This functionality could be put into the Server, but would then require one more call from Broker to Server and back which we deemed not necessary.

Listing 2: Unit test for use case 2

```
//-----------------------------------------------------------------------
//Purpose:          Test Broker/Server listConversionDB  method.
//Preconditions:    Supply Broker and Server.
//Postconditions:   A complete listing of all available conversion
//                  routines available in the Clearinghouse DB.
//Arguments:        none.
//Returns:          0.
//Called Funcs:     Server(), Broker(), Broker.listConversionDB().
//-----------------------------------------------------------------------

void testListConversionDB()
{
```

[2]This testing framework is provided along with the actual DocConversion source package, but it is not built as part of the standard build. For interested developers there is an apart README-testing file that has been supplied explaining how to build the unit tests.

```
    Broker myBroker;
    Result myDBQuery = myBroker.listConversionDB();
    if ( myDBQuery.size() > 0 )
    {
        assert_true( "broker list conversions check", true );
    }
    else
    {
        assert_true( "broker list conversions check", false );
    }
}
```

Use case 3

To test for mmultipleinstances of the Broker we created three sseparatetests, all related to a counter wwhichneeds to be kept up to date to monitor the number of Brokers alive at any one time. The first test checks if the counter is set at all and retrievable, the second test ensures that the counter can be increased and the third test ensures that the counter can be decreased. The third test makes a point to create a second Broker to ensure that this is indeed possible.

Listing 3: Unit test for use case 3

```
//-------------------------------------------------------------------
//Purpose:          Test the getCounter method.
//Preconditions:    Broker object.
//Postconditions:   Broker counter is checked.
//Arguments:        none.
//Returns:          0.
//Called Funcs:     Broker(), Broker.getCounter(), assert_eq().
//-------------------------------------------------------------------
void testGetCounter()
{
    Broker myBroker;
    assert_eq("broker counter check", 1, myBroker.getCounter() );
}

//-------------------------------------------------------------------
//Purpose:          Test the increaseCounter method.
//Preconditions:    Two Broker objects.
//Postconditions:   Broker counter is increased and checked.
//Arguments:        none.
//Returns:          0.
//Called Funcs:     Broker(), Broker.getCounter(), assert_eq().
//-------------------------------------------------------------------
void testIncreaseCounter()
{
    Broker myBroker;
    assert_eq("initial counter check", 1, myBroker.getCounter() );
    Broker anotherBroker;
    assert_eq("increase counter check", 2, anotherBroker.getCounter() );
}

//-------------------------------------------------------------------
//Purpose:          Test decresing of Broker Counter.
//Preconditions:    Two Broker objects.
//Postconditions:   Broker counter is increased, decreased and checked.
//Arguments:        none.
//Returns:          0.
//Called Funcs:     Broker(), ~Broker(), Broker.getCounter(), assert_eq().
//-------------------------------------------------------------------
void testDecreaseCounter()
{
    {
        Broker myBroker;
        Broker anotherBroker;
        int mycounter = anotherBroker.getCounter();
        assert_eq("pre decrease counter check", 2, mycounter );
```

```
    }
    Broker yetAnotherBroker;
    int yetanothercounter = yetAnotherBroker.getCounter();
    assert_eq("decreased counter check", 1, yetanothercounter );
}
```

Use case 4

The user submits a valid URI and cconversionroutine to the system. This will be the actual conversion taking place so we test not only for the conversion itself, but for the fetching of the document to be converted.

The tests show that a Document class was needed to hold the various elements that would be needed to manage a documents conversion. The first test is for the Broker method that calls to the Server to convert a dodocument-The second is the Server converting the document based on the call from the Broker.

Listing 4: Unit test for use case 4

```
//-------------------------------------------------------------------
//Purpose:          Test the convertDocument method.
//Preconditions:    Broker and Document objects.
//Postconditions:   Document convertDocument method checked.
//Arguments:        none.
//Returns:          0.
//Called Funcs:     Document( URI, routine ), Broker(),
//                  Broker.convertDocument(), assert_true().
//-------------------------------------------------------------------
void testConvertDocument()
{
    Broker myBroker;
    Document myDocument(
        "http://infolab.uvt.nl/people/erics/docs/tech_clearinghouse.pdf",
        "pdf2ps");
    if ( myBroker.convertDocument(myDocument) )
    {
        assert_true( "Conversion results", true );
    }
    else
    {
        assert_true( "Conversion results", false );
    }
}

//-------------------------------------------------------------------
//Purpose:          Test Server requestConversion method.
//Preconditions:    Supply a correctly instantiated Document.
//Postconditions:   Returns true if conversion successful, false
//                  if not.
//Arguments:        none.
//Returns:          0.
//Called Funcs:     Server(), Document(), Server.requestConversion().
//-------------------------------------------------------------------

void testServerRequestConversion()
{
    Server myServer;
    Document myDocument(
        "http://infolab.uvt.nl/people/erics/docs/tech_clearinghouse.pdf",
        "pdf2ps");
    if ( myServer.requestConversion( myDocument ) )
    {
        assert_true( "server request conversion check", true );
    }
    else
    {
        assert_true( "server request conversion check", false );
```

```
    }
}
```

Use case 5

The results of all conversions should show up in on the users local machine in the /tmp directory. The first test shows this check being performed on the Document class (internally DocConversion will have hard coded usage of the /tmp dir, so here it is assumed by using the set method). The second test shows the converted documents location which is tested with a local url.

Listing 5: Unit test for use case 5

```
//-----------------------------------------------------------------------
//Purpose:            Test the getFileLocation method.
//Preconditions:      Document object.
//Postconditions:     Document getFileLocation method checked.
//Arguments:          none.
//Returns:            0.
//Called Funcs:       Document( URI, routine ),
//                    Document.setFileLocation(),
//                    Document.getFileLocation(), assert_eq().
//-----------------------------------------------------------------------
void testGetDocumentFileLocation()
{
    Document myDocument(
        "http://infolab.uvt.nl/people/erics/docs/tech_clearinghouse.pdf",
        "pdf2ps");
    myDocument.setFileLocation("/tmp");
    string myFileLocation = myDocument.getFileLocation();
    int match;  // to check matching strings.
    if ( myFileLocation == "/tmp" )
    {
        match = 0;
    }
    else
    {
        match = 1;
    }
    assert_eq("get Document file location check", 0, match);
}

//-----------------------------------------------------------------------
//Purpose:            Test the getConversionLocation method.
//Preconditions:      Document object.
//Postconditions:     Document getConversionLocation method checked.
//Arguments:          none.
//Returns:            0.
//Called Funcs:       Document( URI, routine ),
//                    Document.setConversionLocation(),
//                    Document.getConversionLocation(), assert_eq().
//-----------------------------------------------------------------------
void testGetDocumentConLocation()
{
    Document myDocument(
        "http://infolab.uvt.nl/people/erics/docs/tech_clearinghouse.pdf",
        "pdf2ps");
    myDocument.setConversionLocation("http://localhost/conversions");
    string loc = myDocument.getConversionLocation();
    int match;  // to check matching strings.
    if ( loc == "http://localhost/conversions" )
    {
        match = 0;
    }
    else
    {
        match = 1;
    }
    assert_eq("get Document conversion location check", 0, match);
```

```
}
```

Use case 6

The user wants to search for a specific representation type (PDF, HTML, text, etc.) in our conversion database. We provide for an answer that includes any routine that has the given representation type listed in it's 'to' or 'from' fields. This means we get an overview of all routines supplying a conversion from the given representation type, or to the given representation type.

Listing 6: Unit test for use case 6

```
//-----------------------------------------------------------------------
//Purpose:            Test Broker/Server Representaion type query method.
//Preconditions:     Supply Broker.
//Postconditions:  A listing of all available convconversion
//                        routines based on given representation type in
//                       the Clearinghouse DB (both from and to).
//Arguments:       none.
//Returns:           0.
//Called Funcs:    Broker(), Broker.getRepresentaionTypeListing( string repType).
//-----------------------------------------------------------------------
void
testServerRepresentationTypeQuery()
{
    Broker myBroker;
    string representationTypeTester = "pdf";

    Result myRepresentationQuery =
        myBroker.getRepresentationTypeListing( representationTypeTester );
    if ( myRepresentationQuery.size() > 0 )
    {
        assert_true( "representation types listing", true );
    }
    else
    {
        assert_true( "representation types listing", false );
    }
}
```

Use case 7

The user wants to search for a specific feature type (binary, key-words, etc.) in our conversion database. We provide for an answer that includes any routine that has the given feature type listed in it's 'to' or 'from' fields. This means we get an overview of all routines supplying a conversion from the given feature type, or to the given feature type.

Listing 7: Unit test for use case 7

```
//-----------------------------------------------------------------------
//Purpose:            Test Broker/Server feature type query method.
//Preconditions:  Supply Broker.
//Postconditions: A listing of all available conversion routines
//                            based on given feature type in the Clearinghouse
//                            DB (both from and to).
//Arguments:        none.
//Returns:            0.
//Called Funcs:    Broker(), Broker.getFeatureTypeListing( string featureType).
//-----------------------------------------------------------------------
void testServerFeatureTypeQuery()
{
    Broker myBroker;
    string featureTypeTester = "binary";
```

```
    Result myFeatureQuery =
        myBroker.getFeatureTypeListing( featureTypeTester );
    if ( myFeatureQuery.size() > 0 )
    {
        assert_true( "feature types listing", true );
    }
    else
    {
        assert_true( "feature types listing", false );
    }
}
```

Extra tests

There are a few tests that are used to verify internal methods that are used by the rest of the use cases listed above. They are grouped here for completeness and provide verification for various elements that make up a Document object.

Listing 8: Extraneous unit tests

```
//-------------------------------------------------------------------
//Purpose:            Test the getDocumentUrl method.
//Preconditions:      Document object.
//Postconditions:     Document getDocumentUrl method checked.
//Arguments:          none.
//Returns:            0.
//Called Funcs:       Document( URI, routine ), Document.getDocumentUrl(),
//                    assert_eq().
//-------------------------------------------------------------------
void testGetDocumentUrl()
{
    Document myDocument(
        "http://infolab.uvt.nl/people/erics/docs/tech_clearinghouse.pdf",
        "pdf2ps");
    string myUrl = myDocument.getDocumentUrl();
    int match;  // to check matching strings.
    if ( myUrl ==
        "http://infolab.uvt.nl/people/erics/docs/tech_clearinghouse.pdf" )
    {
        match = 0;
    }
    else
    {
        match = 1;
    }
    assert_eq("get Document url check", 0, match);
}

//-------------------------------------------------------------------
//Purpose:            Test the getDocumentFile method.
//Preconditions:      Document object.
//Postconditions:     Document getDocumentFile method checked.
//Arguments:          none.
//Returns:            0.
//Called Funcs:       Document( URI, routine ), Document.getDocumentFile(),
//                    assert_eq().
//-------------------------------------------------------------------
void testGetDocumentFile()
{
    Document myDocument(
        "http://infolab.uvt.nl/people/erics/docs/tech_clearinghouse.pdf",
        "pdf2ps");
    string myFile = myDocument.getDocumentFile();
    int match;  // to check matching strings.
    if ( myFile == "tech_clearinghouse.pdf" )
    {
        match = 0;
    }
    else
```

```
    {
        match = 1;
    }
    assert_eq("get Document file check", 0, match);
}

//-------------------------------------------------------------------
//Purpose:          Test the getDocumentConversion method.
//Preconditions:    Document object.
//Postconditions:   Document getDocumentConversion method checked.
//Arguments:        none.
//Returns:          0.
//Called Funcs:     Document( URI, routine ),
//                  Document.getDocumentConversion(), assert_eq().
//-------------------------------------------------------------------
void testGetDocumentConversion()
{
    Document myDocument(
        "http://infolab.uvt.nl/people/erics/docs/tech_clearinghouse.pdf",
        "pdf2ps");
    string myConversion = myDocument.getDocumentConversion();
    int match;  // to check matching strings.
    if ( myConversion == "pdf2ps" )
    {
        match = 0;
    }
    else
    {
        match = 1;
    }
    assert_eq("get Document conversion check", 0, match);
}
```

0.3.2 Classes

In this section we provide the class diagrams and implemented interfaces for each of the three classes.

Broker class

Listing 9: Broker Class

```
// ********************************************************************
// Broker.h
// ********************************************************************
//
//  DESCRIPTION: API for Broker class. Used by clients of the
//                       DocConversion project to broker a conversion
//                       of a given document.
//  NOTE:              This program makes use of strings and Document.h.
//
/*************************************************************************
 *   Copyright (C) 2003 by drs. Eric D. Schabell                         *
 *   erics@cs.kun.nl                                                     *
 *                                                                       *
 *   This program is free software; you can redistribute it and/or modify *
 *   it under the terms of the GNU General Public License as published by *
 *   the Free Software Foundation; either version 2 of the License, or   *
 *   (at your option) any later version.                                 *
 *************************************************************************/
#ifndef _BROKER_H_
#define _BROKER_H_

//-------------------------------------------------------------------
//#includes
//-------------------------------------------------------------------
#include <string>           // STL strings.
#include <pqxx/all.h>       // postgresql api.
```

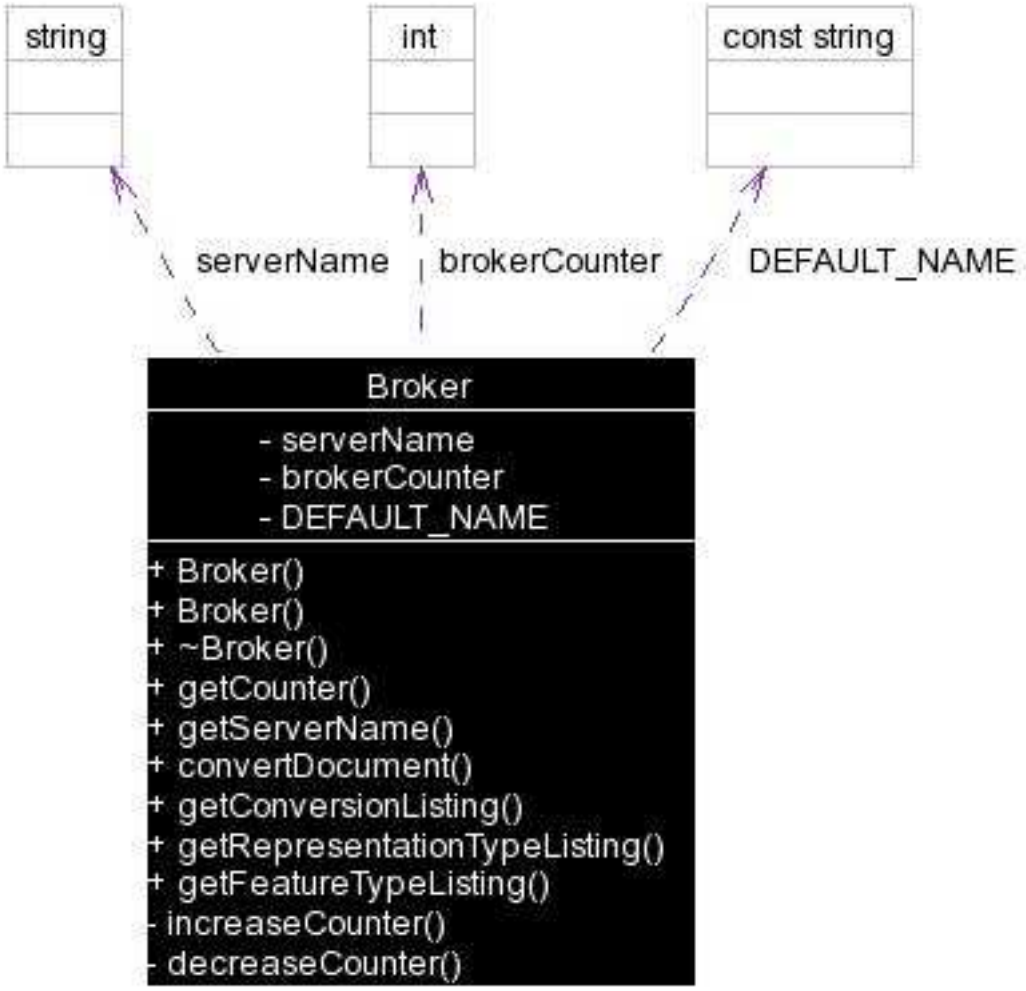

Figure 1: Broker Class

```
#include "Document.h"

using namespace std;
using namespace pqxx;


class Broker
{

public:

    //-------------------------------------------------------------------------
    //Purpose:            default constructor.
    //Preconditions:     none.
    //Postconditions:   a Broker object is instantiated.
    //Arguments:         none.
    //Returns:            Broker.
    //Called Funcs:     increaseCounter().
    //-------------------------------------------------------------------------
    Broker ();

    //-------------------------------------------------------------------------
    //Purpose:            constructor with a (remote) server name given.
    //Preconditions:     must supply a valid server hostname.
    //Postconditions:   a Broker object is instantiated with a
    //                        remote server hostname.
    //Arguments:         string name.
    //Returns:            Broker.
    //Called Funcs:     increaseCounter().
    //-------------------------------------------------------------------------
    Broker ( const string name );

    //-------------------------------------------------------------------------
    //Purpose:            default destructor.
    //Preconditions:     Broker object.
    //Postconditions:   Broker object destroyed.
    //Arguments:         none.
    //Returns:            0
    //Called Funcs:     decreaseCounter().
    //-------------------------------------------------------------------------
    ~Broker();

    //-------------------------------------------------------------------------
    //Purpose:            obtain the Broker Counter's current value.
    //Preconditions:     none.
    //Postconditions:   returns brokerCounter value.
    //Arguments:         none.
    //Returns:            const int brokerCounter.
    //Called Funcs:     none.
    //-------------------------------------------------------------------------
    const int getCounter();

    //-------------------------------------------------------------------------
    //Purpose:            obtain the name of the server which the Broker
    //                        is brokering with.
    //Preconditions:     none.
    //Postconditions:   returns serverName.
    //Arguments:         none.
    //Returns:            const string serverName.
    //Called Funcs:     none.
    //-------------------------------------------------------------------------
    const string getServerName();

    //-------------------------------------------------------------------------
    //Purpose:            prepare for conversion of the provided Document,
    //                        and pass to Server for actual conversion.
    //Preconditions:     Document object.
    //Postconditions:   returns true if conversion successfull, false if not.
    //                        Converted document location is registered in the
    //                        Document.conLocation attribute.
    //Arguments:         Document& doc.
    //Returns:            bool.
    //Called Funcs:     Document.getDocumentConversion(),
    //                        getServerName(), Server(),
```

```
    //                      Server.requestConversion().
    //-------------------------------------------------------------------------
    bool convertDocument ( Document& doc );

    //-------------------------------------------------------------------------
    //Purpose:          obtain list of conversions in conversion database.
    //Preconditions:    Server must be available.
    //Postconditions:   returns Result object contianing the list of available
    //                        conversions.
    //                      This is a 5 column answer:
    //                        routine_name,
    //                        from_feature_type,
    //                        to_feature_type,
    //                        from_representation_type,
    //                        to_representation_type.
    //Arguments:        none.
    //Returns:          Result.
    //Called Funcs:     Server.listConversionDB().
    //-------------------------------------------------------------------------
    Result getConversionListing();


    //-------------------------------------------------------------------------
    //Purpose:          obtain list of available conversions in conversion
    //                      database based on given representation type..
    //Preconditions:    Server must be available.
    //Postconditions:   returns Result object contianing the list of available
    //                        conversions.
    //                      This is a 5 column answer:
    //                        routine_name,
    //                        from_feature_type,
    //                        to_feature_type,
    //                        from_representation_type,
    //                        to_representation_type.
    //Arguments:        none.
    //Returns:          Result.
    //Called Funcs:     Broker(), Server.listRepresentaionTypes(
    //                                        string representationType ).
    //-------------------------------------------------------------------------
    Result getRepresentationTypeListing( string representationType );

    //-------------------------------------------------------------------------
    //Purpose:          obtain list of available conversions in conversion
    //                      database based on given feature type..
    //Preconditions:    Server must be available.
    //Postconditions:   returns Result object contianing the list of available
    //                        conversions.
    //                      This is a 5 column answer:
    //                        routine_name,
    //                        from_feature_type,
    //                        to_feature_type,
    //                        from_representation_type,
    //                        to_representation_type.
    //Arguments:        none.
    //Returns:          Result.
    //Called Funcs:     Broker(), Server.listFeatureTypes( string featureType ).
    //-------------------------------------------------------------------------
    Result getFeatureTypeListing( string featureType );



private:
    static int brokerCounter;   // keeps track of  nr. activerokers.
    static const string DEFAULT_NAME; // localhost used if notinitialized.
    string serverName;          // name of conversion server..

    //-------------------------------------------------------------------------
    //Purpose:            to increase the brokerCounter attribute.
    //Preconditions:    none.
    //Postconditions:   brokerCounter attribute has increased by one.
    //Arguments:        none.
    //Returns:            0.
    //Called Funcs:     none.
    //-------------------------------------------------------------------------
```

```
    void increaseCounter();

    //-------------------------------------------------------------------
    //Purpose:            to decrease the brokerCounter attribute.
    //Preconditions:     none.
    //Postconditions:  brokerCounter attribute has decreased by one.
    //Arguments:        none.
    //Returns:            0.
    //Called Funcs:     none.
    //-------------------------------------------------------------------
    void decreaseCounter();
};

#endif /* _BROKER_H_ */
```

Document class

Listing 10: Document Class

```
// ********************************************************************
// Document.h
// ********************************************************************
//
//  DESCRIPTION: API for Document class. Used to manipulate a
//                          given document.
//  NOTE:               This program makes use of strings.
//
/*********************************************************************
 *   Copyright (C) 2003 by drs. Eric D. Schabell                     *
 *   erics@cs.kun.nl                                                 *
 *                                                                   *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 2 of the License, or     *
 *   (at your option) any later version.                             *
 *********************************************************************/


//-------------------------------------------------------------------
//conditional compilation
//-------------------------------------------------------------------
#ifndef _DOCUMENT_H_
#define _DOCUMENT_H_

//-------------------------------------------------------------------
//#includes
//-------------------------------------------------------------------
#include <string>

using namespace std;

class Document
{
public:

    //-------------------------------------------------------------------
    //Purpose:            default constructor.
    //Preconditions:     none.
    //Postconditions:  a Document object is instantiated and all
    //                        attributes are zeroed.
    //Arguments:        none.
    //Returns:            Document.
    //Called Funcs:     none.
    //-------------------------------------------------------------------
    Document();

    //-------------------------------------------------------------------
    //Purpose:            constructor with url and conversion given..
    //Preconditions:     must supply a string document url and a
    //                        string conversion routine.
```

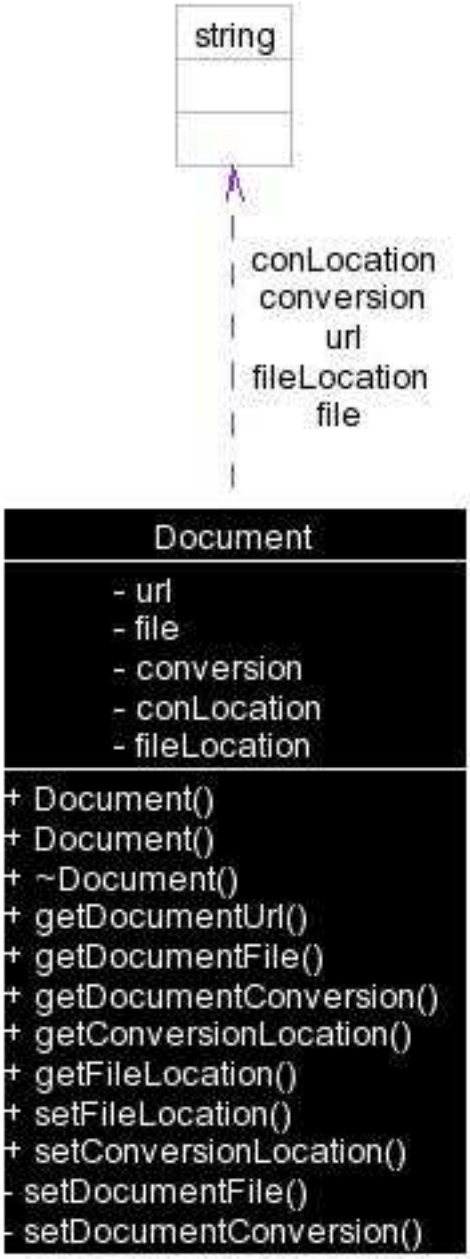

Figure 2: Document Class

```
//Postconditions:  a Document object is instantiated with a
//                       document url, file name and conversion
//                       routine defined. File location and Conversion
//                       location are zeroed.
//Arguments:       string doc_url, string routine.
//Returns:           Document.
//Called Funcs:    setDocumentFile(), setDocumentConversion().
//-------------------------------------------------------------------
Document( string doc_url, string routine );

//-------------------------------------------------------------------
//Purpose:           default destructor.
//Preconditions:    Document object.
//Postconditions:  Document object destroyed.
//Arguments:       none.
//Returns:           0
//Called Funcs:    none.
//-------------------------------------------------------------------
~Document();

//-------------------------------------------------------------------
//Purpose:           obtain the documents complete url.
//Preconditions:    Docuemnt object exists.
//Postconditions:  returns url attribute.
//Arguments:       none.
//Returns:           string.
//Called Funcs:    none.
//-------------------------------------------------------------------
string getDocumentUrl();

//-------------------------------------------------------------------
//Purpose:           Return the file name if instantiated, otherwise
//                       returns the url attribute.
//Preconditions:    Docuemnt object created.
//Postconditions:  success = returns instantiated file attribute.
//                       failure    = returns url if file has not been instantiated,
//                                       which signals a problem with the
//                                       document construction.
//Arguments:       none.
//Returns:           string
//Called Funcs:    none.
//-------------------------------------------------------------------
string getDocumentFile();

//-------------------------------------------------------------------
//Purpose:           Return the conversion routine attribute.
//Preconditions:    Docuemnt object created.
//Postconditions:  returns instantiated file attribute.
//Arguments:       none.
//Returns:           string
//Called Funcs:    none.
//-------------------------------------------------------------------
string getDocumentConversion();

//-------------------------------------------------------------------
//Purpose:           Return the conLocation attribute.
//Preconditions:    Docuemnt object created.
//Postconditions:  returns conLocation attribute.
//Arguments:       none.
//Returns:           string
//Called Funcs:    none.
//-------------------------------------------------------------------
string getConversionLocation();

//-------------------------------------------------------------------
//Purpose:           Return the docuemnt file location.
//Preconditions:    Docuemnt object created.
//Postconditions:  returns fileLocation attribute.
//Arguments:       none.
//Returns:           string
//Called Funcs:    none.
//-------------------------------------------------------------------
string getFileLocation();
```

```
    //-------------------------------------------------------------------
    //Purpose:            Sets the file location to given location.
    //Preconditions:    Docuemnt object created, location provided.
    //Postconditions:  fileLocation attribute is set with given file
    //                        location.
    //Arguments:        string file_location.
    //Returns:            0.
    //Called Funcs:     none.
    //-------------------------------------------------------------------
    void setFileLocation( string file_location );

    //-------------------------------------------------------------------
    //Purpose:            Sets the converted files location to given location.
    //Preconditions:    Docuemnt object created, location provided.
    //Postconditions:  conLocation attribute is set with given file
    //                        location.
    //Arguments:        string conversion_location.
    //Returns:            0.
    //Called Funcs:     none.
    //-------------------------------------------------------------------
    void setConversionLocation( string conversion_location );

private:
    string url;          // url to location of document.
    string file;           // document basename.
    string conversion;     // conversion being requested on this docuement.
    string conLocation; // location of converted docuement (url or local file system)
    string fileLocation;  // location of basename (local file system).

    //-------------------------------------------------------------------
    //Purpose:            Determine and set the file name by basnaming
    //                        the url attribute.
    //Preconditions:    Docuemnt object created.
    //Postconditions:  success = a set file attribute.
    //                        failure    = returns error
    //Arguments:        none.
    //Returns:            0.
    //Called Funcs:     basename().
    //-------------------------------------------------------------------
    void setDocumentFile();

    //-------------------------------------------------------------------
    //Purpose:            Instantiates the conversion attribute with the
    //                        given convertion routine.
    //Preconditions:    Docuemnt object created, convertion routine
    //                        provided..
    //Postconditions:  instantiated conversion attribute.
    //Arguments:        string convertion_request..
    //Returns:            0.
    //Called Funcs:     none.
    //-------------------------------------------------------------------
    void setDocumentConversion( string conversion_request );
};

#endif /* _DOCUMENT_H_  */
```

Server class

Listing 11: Server Class

```
// ****************************************************************
// Server.h
// ****************************************************************
//
//  DESCRIPTION:  contains the function prototypes and the #includes
//                          used in implementation of the DocConversion Server.
//
//  NOTE:               This program makes use of the STL string.
//
//  CHANGE LOG:
```

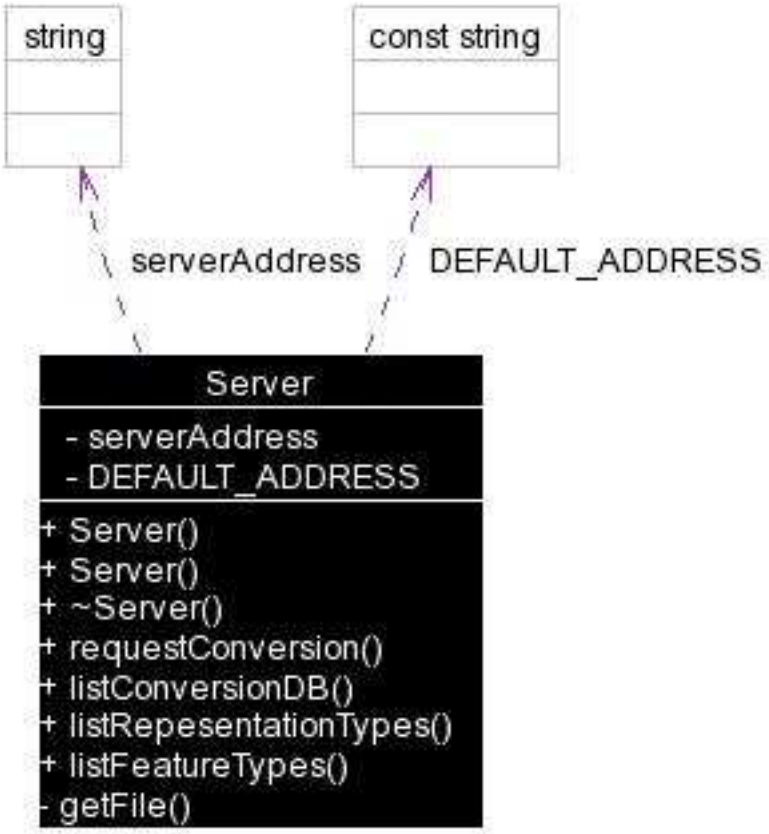

Figure 3: Server Class

```
//                          2003-10-15  Initial creation.
//
/*************************************************************************
 *   Copyright (C) 2003 by drs. Eric D. Schabell                         *
 *   erics@cs.kun.nl                                                     *
 *                                                                       *
 *   This program is free software; you can redistribute it and/or modify *
 *   it under the terms of the GNU General Public License as published by *
 *   the Free Software Foundation; either version 2 of the License, or   *
 *   (at your option) any later version.                                 *
 *************************************************************************/

//-------------------------------------------------------------------------
//conditional compilation
//-------------------------------------------------------------------------
#ifndef _SERVER_H
#define _SERVER_H


//-------------------------------------------------------------------------
//#includes
//-------------------------------------------------------------------------
#include <string>
#include "Document.h"
#include <pqxx/all.h>       // postgresql api.

using namespace std;
using namespace pqxx;



class Server
{
public:

    //-------------------------------------------------------------------
    //Purpose:           Default constructor.
    //Preconditions:    none.
    //Postconditions:  Server object created.
    //Arguments:        none.
    //Returns:           0.
    //Called Funcs:    none.
    //-------------------------------------------------------------------
    Server();

    //-------------------------------------------------------------------
    //Purpose:           Constructo with server address.
    //Preconditions:    none..
    //Postconditions:  Server object created, serverAddress set.
    //Arguments:        string address.
    //Returns:           0.
    //Called Funcs:    none.
    //-------------------------------------------------------------------
    Server( string address );

    //-------------------------------------------------------------------
    //Purpose:           Destructor.
    //Preconditions:    Server object exists..
    //Postconditions:  Server object destroyed.
    //Arguments:        none.
    //Returns:           0.
    //Called Funcs:    none.
    //-------------------------------------------------------------------
    ~Server();

    //-------------------------------------------------------------------
    //Purpose:           conversion of the provided Document, based
    //                       on the objects registered conversion routine.
    //Preconditions:    Document object.
    //Postconditions:  returns true if conversion successfull, false if not.
    //                      Converted document location is registered in the
    //                      Document.conLocation attribute.
    //Arguments:       Document& doc.
    //Returns:           bool.
```

```
    //Called Funcs:    Document.getDocumentConversion(),
    //                 Document.getFileLocation(), getFile(),
    //                 Document.getDocumentFile(),
    //                 Document.setFileLocation(),
    //                 Document.getDocumentConversion(), system(),
    //                 c_str(), Document.setConversionLocation().
    //-------------------------------------------------------------------
    bool requestConversion( Document& doc );

    //-------------------------------------------------------------------
    //Purpose:          obtain list of conversions in database.
    //Preconditions:    DB must be available.
    //Postconditions:   returns Result object contianing the list of available
    //                      conversions.
    //                    This is a 5 column answer:
    //                      routine_name,
    //                      from_feature_type,
    //                      to_feature_type,
    //                      from_representation_type,
    //                      to_representation_type.
    //Arguments:        none.
    //Returns:          Result.
    //Called Funcs:     Connection(), Transaction(), Result(),
    //                      Transaction.Exec(), Transaction.Commit().
    //-------------------------------------------------------------------
    Result listConversionDB();

    //-------------------------------------------------------------------
    //Purpose:          obtain list of conversions in database based on
    //                      given representation type.
    //Preconditions:   DB must be available.
    //Postconditions: returns Result object contianing the list of available
    //                      conversions.
    //                      This is a 5 column answer:
    //                      routine_name,
    //                      from_feature_type,
    //                      to_feature_type,
    //                      from_representation_type,
    //                      to_representation_type.
    //Arguments:       none.
    //Returns:           Result.
    //Called Funcs:    Connection(), Transaction(), Result(),
    //                       Transaction.Exec(), Transaction.Commit().
    //-------------------------------------------------------------------
    Result listRepesentationTypes( string representationType );


    //-------------------------------------------------------------------
    //Purpose:          obtain list of conversions in database based on
    //                      given feature type.
    //Preconditions:   DB must be available.
    //Postconditions: returns Result object contianing the list of available
    //                      conversions.
    //                      This is a 5 column answer:
    //                      routine_name,
    //                      from_feature_type,
    //                      to_feature_type,
    //                      from_representation_type,
    //                      to_representation_type.
    //Arguments:       none.
    //Returns:         Result.
    //Called Funcs:    Connection(), Transaction(), Result(),
    //                       Transaction.Exec(), Transaction.Commit().
    //-------------------------------------------------------------------
    Result listFeatureTypes( string featureType );


private:
    string serverAddress;      // address of server, either dns name or ip.
    const string DEFAULT_ADDRESS;  // localhost if not initialized.

    //-------------------------------------------------------------------
    //Purpose:            retrieve the Document file from its url location.
    //Preconditions:    Document object.
```

```
    //Postconditions:  returns true if file retrieval successfull, false if not.
    //                        Only does remote retrieval after checking for local
    //                        presence of the file. The location of the file is set
    //                        for the Document.fileLocation attribute.
    //Arguments:        Document& doc.
    //Returns:            bool.
    //Called Funcs:    Document.getDocumentFile(), ifstream.open(),
    //                       c_str(), system(), Document.getDocumentUrl(),
    //                       Document.setFileLocation(), ifstream.close().
    //-------------------------------------------------------------------
    bool getFile ( Document& doc );
};

#endif /* _SERVER_H */
```

This concludes the discussion on phase one of the DocConversion project. We will continue further with our plans to expand the functionality in phases two and three. The resulting improvements and implementations will be discussed in future technical reports.

0.3.3 Developer access

As previously hinted at, this project was organized from the beginning to be released into the Free Software community. Early in the development cycle time was taken to setup a Sourceforge project, import the code into a CVS repository, setup the documentation, forums and bug reporting tools. Furthermore, in the interest of spreading the news of future code releases a Freshmeat project was also started. These two will provide for a very wide exposure, support the project with professional tools and hopefully provide interested developers with the chance to participate. For more information see the DocConversion project homepage and Freshmeat project page:

`http://docconversion.sourceforge.net`
`http://freshmeat.net/projects/docconversion`

Interested developers are encouraged to take part by checking out the code, submitting patches, bugs or comments requests at the above project site. We hope to welcome you soon to the DocConversion project as a user, developer or just as someone interested in document conversions.

Bibliography

K. Beck and M. Fowler. *Extreme Programming Explained*. Addison-Wesley, Reading, Massachusetts, USA, 2000.

cUrl – Grok those URLs, 2003.
`http://curl.haxx.se`

GNU. *GNU wget*, 2003.
`http://wget.sunsite.dk`

B. van Gils, H.A. Proper, and P. van Bommel. A conceptual model for information suppy. Technical Report NIII-R0313, Nijmegen Institute for Information and Computing Sciences, University of Nijmegen, Nijmegen, The Netherlands, EU, 2003. Accepted for publication in Data & Knowledge Engineering.

B. van Gils, H.A. Proper, and P. van Bommel. Towards a general theory for information supply. In C. Stephanidis, editor, *Proceedings of the 10th International Conference on Human-Computer Interaction*, pages 720–724, Crete, Greece, EU, 2003. ISBN 0805849300

B. van Gils and E.D. Schabell. User-profiles for information retrieval. In *Proceedings of the 15th Belgian-Dutch Conference on Artificial Intelligence (BNAIC'03)*, Nijmegen, The Netherlands, October 2003.

R. Jeffries. *Unit Test*, 2003.
`http://c2.com/cgi/wiki?UnitTest`

LibPQXX Website, 2003.
`http://pqxx.tk`

R. Miller. *Demistifying Extreme Programing: Just-in-time design*, 2002.
`http://www-106.ibm.com/developerworks/library/j-xp07013.html?ca=%dnt-426#resources`

Eric D. Schabell. Resource access in generic information retrieval systems. Master's thesis, Vrije Universiteit, Amsterdam, Netherlands, 2002.

E.D. Schabell. Building the pronir conversion clearinghouse. Technical Report NIII-R0317, Nijmegen Institute for Information and Computing Sciences, University of Nijmegen, Nijmegen, The Netherlands, EU, 2003.

Source Mage Website, 2003.
`http://www.sourcemage.org`

Unit++ Testing Framework, 2003.
`http://unitpp.sourceforge.net`