

**ENHANCE TO DATA CLENCHING FEATURES IN
IBM DATA INTEGRATIONS SUITE**

BY

**S.M.WIJESEKARA
(01/AS/067)**

This thesis is submitted in partial fulfillment of the requirements for the degree of Bachelor of Science in Physical Sciences.

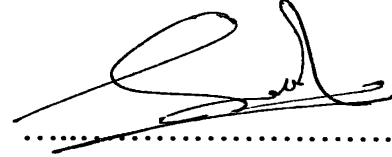
Department of Physical Sciences,
Faculty of Applied Sciences,
Sabaragamuwa University of Sri Lanka.
Buttala.

August 2006

Declaration

The content described in this thesis was practically implemented by me at the Virtusa Corporation and the Faculty of Applied Sciences under the supervision of Mr. N. Jayathilake and Mr. J.B. Ekanayake and the report described on this thesis has not been submitted by any one for another degree.

S.M. Wijesekara.
(01/AS/040)

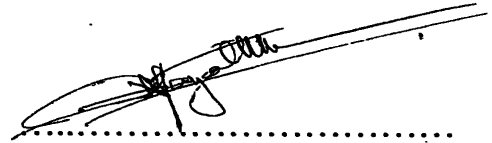


Signature

Date: 13-09-2006

Certified By,

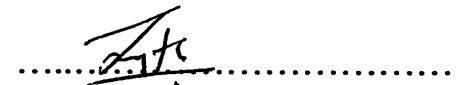
Mr. N. Jayathilake,
External Supervisor,
Associate Technical Lead,
Virtusa Corporation,
Sir Chittampalam A. Gardiner Mw,
Colombo 2.
Sri Lanka.



Signature

Date: 13/09/2006

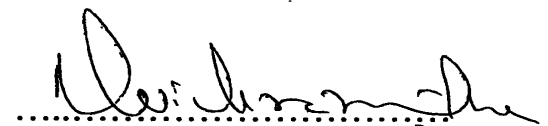
Mr. J.B. Ekanayake,
Internal Supervisor,
Lecturer/ Department of Physical Sciences,
Faculty of Applied Sciences,
Sabaragamuwa University of Sri Lanka,
Buttala.



Signature

Date: 16/09/2006

Dr (Mrs.) N. Wickramaratne,
Head/Department of Physical Sciences,
Faculty of Applied Sciences,
Sabaragamuwa University of Sri Lanka,
Buttala.



Signature

Date: 15/09/2006

Acknowledgement

First I should express my sincere thanks to my internal supervisor Mr. J.B. Ekanayake, Lecturer, Department of Physical Sciences, Faculty of Applied Sciences for his encouragement and guidance through this study.

And also I wish to express my deepest gratitude to my external supervisor Mr. N. Jayathilake, Associate Technical Lead, Virtusa Corporation, for his advice, encouragement and guidance through the study and for sparing his valuable time in bringing this study to a successful completion and also to my project manager Mrs. Inosha Sirithunga, Senior Team Lead, Virtusa Corporation for her encouragement and guidance through this project.

I express my sincere gratitude to all my team mates in the training and development team, fellow virtusans and Virtusa Corporation for providing me the opportunity to carry out my industrial training at Virtusa Corporation.

And I express my sincere gratitude to Dr. Mahinda Wickramaratne, The Dean, Faculty of Applied Sciences, Sabaragamuwa University of Sri Lanka, and Dr (Mrs.) Nirmali Wickramaratne, Head, Department of Physical Sciences, Faculty of Applied Sciences, Sabaragamuwa University of Sri Lanka, for guiding me toward a successful completion.

Finally, I express my heart-felt gratitude towards the lectures for their cooperation through out my study and my colleagues for their individual help and guidance at all times.

Abstract

Virtusa Corporation is a leading global technology innovation services provider and it has grown beyond being an efficient provider of product and application development services to being the partner of choice in creating competitive advantage for its clients using technology solutions. IBM Information Integration Solutions - Quality Stage team is one of the development teams in the Virtusa that goes under IBM Information Integration Solutions account. Prior to joining with the Virtusa this project was developed by the company Ascential software corporation. Now the maintenance part of the project is going on. Under this stage the aim of this project was to enhance data clenching features in the system IBM IIS - QualityStage Designer with the technology of VB6 Enterprise Edition.

IBM IIS - QualityStage is a comprehensive development environment for building applications that re-engineer data. This data re-engineering environment is designed to help programmers, programmer analysts, business analysts, and others re-engineer data to meet business objectives and data quality management standards. It provides a set of integrated modules for accomplishing data re-engineering tasks such as investigating, conditioning, matching, building relationships, resolving conflicts, and formatting data.

The software development process used for this project was Rational Unified Process. Due to the maintenance part of the project is going on, the requirements of this project were mainly achieved through user feedbacks. Unified Modeling Language was used to convert the requirement into an analysis model. Then the analysis model was translated into a design model. To verify this system, logical system architecture diagram, design class diagram, high level use case diagram, component diagram, user interfaces and Entity-Relational diagram was created. An iterative approach was applied to each phase above to give the problem owner to be involved in the development of the system. Evaluative feedbacks were acquired through mails from problem owners. Refinements and Modifications were carried out after each meeting to meet the client's requirements.

CONTENT

Declaration	I
Acknowledgement	II
Abstract	III
List of Abbreviations	IV
List of Figures	V
List of Tables	VI
Content	VII

Chapter 1 : Introduction 1

1.1	Introduction of Virtusa Corporation	1
1.2	Introduction of IBM IIS	1
1.3	Introduction of IBM IIS QualityStage	2
1.4	Major Challenges	3
1.5	Aims and Objectives	3
1.6	Deployment	3
1.7	Development Approach	4
1.8	Structure of the Report	5

Chapter 2 : Review of Literature 6

2.1	Introduction	6
2.2	IBM IIS QualityStage.....	6
2.2.1	IBM IIS QualityStage and IBM IIS QualityStage Real Time.....	6
2.2.2	Two Modes for Running QualityStage Jobs.....	7
2.2.3	Supports Data Quality Management Standards.....	8
2.2.4	How IBM IIS QualityStage Works.....	9
2.2.5	Matching Concepts.....	11
2.2.5.1	Individual Matching Examples.....	12
2.2.5.2	Geocoding Applications Examples.....	13
2.2.5.3	Many-to-One Matching Examples.....	14
2.2.5.4	File Unduplication Examples.....	14
2.2.5.5	File Grouping Examples.....	15

2.3	IBM IIS QualityStage Clerical Review.....	15
2.4	Visual Basic 6 Enterprise Edition.....	16
2.4.1	Interface Styles.....	18
2.4.2	Working with Multiple Projects.....	20
2.4.3	Managing Application Settings.....	21
2.4.4	Using Conditional Compilation.....	21
2.4.5	Working with Resource Files.....	23
2.4.6	Using Enumerations to Work with Sets of Constants.....	23
2.4.7	The Many (Inter)Faces of Code Reuse.....	24
2.5	Design Patterns.....	25
2.5.1	Singleton Design pattern.....	27
2.5.2	Logical Model.....	28
2.5.3	Physical Model.....	28
2.6	Usability.....	29
 Chapter 3 : Software Development Process		30
3.1	Introduction	30
3.2	Rational Unified Process	30
 Chapter 4 : Requirement Analysis		32
4.1	Introduction	32
4.2	Requirement Elicitation.....	34
4.3	Functional Requirements.....	34
4.4	Design Goals and Constraints.....	38
4.5	Non-Goals.....	39
4.6	Assumptions.....	39
 Chapter 5 : Design		40
5.1	Introduction	40
5.2	High Level Architecture.....	40
5.2.1	QualityStage Designer Process.....	41
5.2.1.1	Specify Use of CRS.....	41
5.2.1.2	Specify Relational Database to store clerical data.....	42

5.2.2	QualityStage Server Process.....	42
5.2.2.1	Create CRS Data tables.....	43
5.2.2.2	Populate CRS tables with Clerical data.....	44
5.3	User Interface Description.....	44
5.4	Process Logic Description.....	45
5.4.1	Decomposition View.....	46
5.4.1.1	CRS Components.....	46
5.4.1.1.1	UI Component.....	46
5.4.1.1.2	CRS Operations Component.....	47
5.4.1.1.2.1	DB Operations Component.....	47
5.4.1.1.3	Match Operation Component.....	47
5.4.1.2	Class Diagrams.....	47
5.4.1.2.1	Review and Reconciliation Classes.....	48
5.4.1.2.2	CRS Operations Classes.....	48
5.4.1.2.3	Logger Component Class.....	49
5.4.2	Process View.....	49
5.4.2.1	Review and Reconciliation Process.....	49
5.5	CRS DBAccess Library design.....	51
5.5.1	Over view.....	51
5.5.2	Class Diagram.....	51
5.5.3	Functionality.....	52
5.5.4	Supporting Concurrent Users.....	53
5.5.5	Cursor creation and database update.....	54
5.5.5.1	Clerical Reconciliation – Select Record Screen.....	55
5.5.5.2	Un-Duplication Match Screen.....	56
5.6	CRS Database Design, Load and Extract.....	57
5.6.1	CRS Database Design.....	57
5.6.2	CRS Database Load.....	60
5.6.2.1	QS Server Extract Modifications Preparing for Database Load.....	60
5.6.2.2	Database Load Platform Support.....	60
5.6.2.3	QS Server Database Load Overview.....	60
5.6.2.4	QS Server Creates CRS Database.....	61
5.6.2.5	QS Server Populates CR Database with Clerical Data.....	63
5.6.3	CRS Database Extraction.....	63

5.6.3.1	Clerical Reconciliation Extract Stage Control File.....	63
5.6.4	Clerical Set Identifier.....	64
5.7	CRS Weight Calculations.....	65
5.7.1	Re-Packaged QS Server Match Weigh Calculation Utility.....	65
5.7.1.1	QS Server Weight Calculation.....	65
5.7.1.2	CRS use of crs.dll.....	65
5.7.1.3	Updating Composite Weights in the CR Database.....	67
5.8	Quality Stage Designer Changes.....	67
5.8.1	Process.....	67
 Chapter 6 : Development Environment.....		69
 Chapter 7 : Deployment Environment		70
 Chapter 8 : Conclusion		71
8.1	Summary of objectives, achievements, problems	71
8.2	Future Consideration	72
 References		73
 Appendix A Reflection		74

List of Abbreviations

IBM	Intel Business Mach
IIS	Information Integration Solutions
CRS	Clerical Reconciliation Station
CR	Clerical Reconciliation
QS	Quality Stage
UnDup	Type of matching that groups records in data file that have similar attributes
Match	1-to-1 matching where a record in data file is matched with only one record in reference file and a record from the B file can participate in only one match
GeoMatch	Many-to-one matching in which any number of records on data file could match the same record on reference file.
Match Set	A record set that was assigned a weight equal to or above the match cutoff weight
Clerical Set	A record set that was assigned a weight in the clerical range (between the clerical cutoff and match cutoff weights)
Residual	A record that did not match another record
Match Score	Composite weight calculated by the Match stage which is the sum of the individual weights for field comparisons which measures the probability that two records represent the same entity
Match Weight	Value computed by a match comparison for a matching field
Match Comparison	Algorithms provided by the MATCH stage for calculating the contributed weight for a field for example, the CHAR comparison is a character-by-character comparison, NUMERIC comparison is an algebraic numeric comparison
A File	Data file used during the Match Stage.
B File	Reference file used during the Match Stage.
A records	Records from input file A
B records	Records from input file B for all match types except Undup.
Working Set	The set of records you are currently reviewing.
UML	Unified Modeling Language
RUP	Rational Unified Process

List of Figures

	Page Number
Chapter 2 : Review of Literature	
Figure 2.1 WordPad, a single-document interface (SDI) application	18
Figure 2.2 Microsoft Excel, a multiple-document interface (MDI) application... ..	19
Figure 2.3 The Windows Explorer, an explorer-style interface	20
Figure 2.4 Singleton pattern logical model.....	28
Figure 2.5 Singleton pattern physical model from design patterns	28
Chapter 3 : Software Development process	
Figure 3.1 A graphical overview of the RUP	31
Chapter 4 : Requirement Analysis	
Figure 4.1 Requirement Analysis Process	32
Figure 4.2 System Use case Diagram	35
Figure 4.3 Use case for Review Match Set	35
Figure 4.4 Use case for Review Unduplication Clerical Set	35
Figure 4.5 Use case for Review GeoMatch Clerical Set	35
Figure 4.6 Use case for Review GeoMatch Multiple Clerical Set.....	35
Figure 4.7 Use case for Review GeoMatch Duplicate Clerical Set	35
Chapter 5 : Design	
Figure 5.1 Clerical Reconciliation Station high level architecture view.....	41
Figure 5.2 Clerical Reconciliation Station's screen layout	45
Figure 5.3 CRS Decomposition View	46
Figure 5.4 CRS Review and Reconciliation Class Diagram	48
Figure 5.5 CRS Operation Classes	48
Figure 5.6 CRS Logger Component Class	49
Figure 5.7 Sequence Diagram for GetMatchSet process	49
Figure 5.8 Sequence Diagram for SwapMasterAndDuplicate process	50
Figure 5.9 Sequence Diagram for Create a Match set process	50

Figure 5.10 CRS Database Access Library Class Diagram.....	51
Figure 5.11 Entity Relationship Diagram for CRS Database Access Library	59

List of Tables

	Page Number
Chapter 2 : Review of Literature	
Table 2.1 Visual Basic Functions for Accessing Registry	21
Table 2.2 Scope of each Conditional Compilation Constants	22
Chapter 5 : Design	
Table 5.1 public interfaces provided by the CRS DBAccess Library and their expected functionality	53
Table 5.2 Record Level Locking Support in each Database	54
Chapter 6 : Development Environment	
Table 6.1 CRS Development Environment	69
Chapter 7 : Deployment Environment	
Table 7.1 CRS Deployment Environment	70

Chapter 1 : Introduction

1.1 Introduction of Virtusa Corporation

Virtusa Corporation is a leading global technology innovation services provider that creates competitive advantage for its clients. Virtusa was founded in 1996 by the prominent technology entrepreneur, Kris Canekeratne, who has assembled a strong leadership team from well-known companies like Infosys, IBM, Aether, 3Com and John Keels.

Previously known as eRUNWAY, Inc., Virtusa has grown beyond being an efficient provider of product and application development services to being the partner of choice in creating competitive advantage for its clients using technology solutions.

Headquartered in Westborough, MA, Virtusa employs the finest global technology talent, spread across its Advanced Technology Centers in the US, India and Sri Lanka. It also has sales and marketing offices in several locations around the world.

1.2 Introduction of IBM IIS

IBM Information Integration Solutions is to integrate and leverage data across all transactional, operational, and analytical applications with confidence in the accuracy, completeness and timeliness of critical information. IBM Information Integration Solutions Software's powerful data profiling, data quality, data transformation, parallel processing, meta data and connectivity solutions enable customers to reduce total cost of ownership and increase return on IT investment.

The IBM Information Integration Solutions, which includes

- IBM IIS DataStage
- IBM IIS DataStage TX
- IBM IIS ProfileStage
- IBM IIS QualityStage
- IBM IIS's Enterprise Integration Platform

1.3 Introduction of IBM IIS QualityStage

IBM IIS QualityStage is a client/server application. IBM IIS QualityStage Designer provides a client interface for defining and customizing data re-engineering jobs. This runs on a Windows workstation. Whereas the IBM IIS QualityStage Designer defines how source data will be processed, the IBM IIS QualityStage server accesses the source data, and processes them into the target re-engineered data. It maintains the data fields and executes the IBM IIS QualityStage data re-engineering jobs that you build with IBM IIS QualityStage Designer.

The IBM IIS QualityStage server runs on the following systems:

- Windows
- OS/390
- UNIX

IBM IIS QualityStage is a comprehensive development environment for building applications that re-engineer data. This data re-engineering environment is designed to help programmers, programmer analysts, business analysts, and others re-engineer data to meet business objectives and data quality management standards.

Large stores of data can quickly and easily be processed by using IBM IIS QualityStage, selectively conform the data as needed. IBM IIS QualityStage provides a set of integrated modules for accomplishing data re-engineering tasks such as:

- Investigating
- Conditioning (standardizing)
- Matching
- Building relationships
- Resolving conflicts
- Formatting data

1.4 Major Challenges

The major challenges of this project were:

1. To enhance the data clenching features in IBM IIS Quality Stage Designer and its Sub components.
2. To fix Bugs of the IBM IIS Quality Stage Designer and its sub components.
3. To acquire technical skills which are to be applied to the IBM IIS Quality Stage Designer and its Sub components.

1.5 Aims and Objectives

The aim of this project was to enhance data clenching features in the system IBM IIS - Quality Stage Designer with the technology of VB6 Enterprise Edition.

This would help users to process clerical records within the IBM IIS Enterprise Application environment and also users will elevate clerical sets to match sets, demote clerical sets to residuals or divide them in to subsets as is meaningful for the type of match.

The objectives of this project were:

1. To capture documents and prioritize requirements for the Clerical Reconciliation Station.
2. To learn and apply the technology of VB6 Enterprise Edition
3. To apply an iterative approach to increase user involvement.
4. To design the Clerical Reconciliation Station while paying specific attention to usability and maintainability.
5. To implement and test the system.
6. To evaluate the system with some user trials.

1.6 Deployment

After enhancing the system in each iteration system is deployed to the customer with latest possible tested source code.

1.7 Development Approach

The first step in acquiring the necessary background knowledge to support the project included:

- Rational Unified Process
- Rapid Application Development
- Design Patterns
- Visual Basic 6 Enterprise Edition
- IBM IIS Quality Stage Designer
- IBM IIS Quality Stage Designer - Clerical Review
- IBM IIS Quality Stage Server
- Providing maintainability and Usability

The software development process used for this project was Rational Unified Process. The requirements of this project were achieved by interviewing with the problem owner, through user feedback mails, by reviewing similar applications and reading sample documents.

Unified Modeling Language was used to convert the requirement into an analysis model. For potential users and their corresponding tasks, the terms of UML diagram were identified.

An analysis model was then translated into a design model. To verify this system, logical system architecture diagram, design class diagram, component diagram, user interfaces and Entity-Relational diagram was created.

An iterative approach was applied to each phase above to give the problem owner to be involved in the development of the system. Evaluative feedbacks were acquired through mails from problem owners. Refinements and Modifications were carried out after each meeting to meet the client's requirements.

A description of the Rational Unified Process – the software development process used for this project will be shown in chapter 3 and a full description of each phase of the system was shown in chapter 4-7.

1.8 Structure of the Report

This report is structured in chapters as follows:

1. Chapter One – A brief Introduction of the project.
2. Chapter Two – Background research on relevant topics.
3. Chapter Three – Software development process of the system.
4. Chapter Four – Requirement and analysis of the system.
5. Chapter Five – Design of the system.
6. Chapter Six – Development environment of the system
7. Chapter Seven – Deployment environment of the system
8. Chapter Eight –A summary of objectives, problem encountered achievements, challenges, and future work of the project.

Chapter 2 : Review of Literature

2.1 Introduction

To achieve the objectives of this project, in-depth knowledge of the following topics will be conducted.

- IBM IIS QualityStage
- IBM IIS QualityStage-Clerical Review
- Visual Basic Technology
- Design Patterns
- Usability

2.2 IBM IIS QualityStage

IBM IIS QualityStage is a client/server application. IBM IIS QualityStage Designer provides a client interface for defining and customizing data re-engineering jobs. IBM IIS QualityStage Designer runs on a Windows workstation. Whereas the IBM IIS QualityStage Designer defines how source data will be processed, the IBM IIS QualityStage server accesses the source data, and processes them into the target re-engineered data. It maintains the data fields and executes the IBM IIS QualityStage data re-engineering jobs that you build with IBM IIS QualityStage Designer.

The IBM IIS QualityStage server runs on the following systems:

- Windows
- OS/390
- UNIX

2.2.1 IBM IIS QualityStage and IBM IIS QualityStage Real Time

In its standard configuration (as described in this guide), IBM IIS QualityStage jobs run as batch processes. That is, we use a complete input file to run a IBM IIS QualityStage job in a stand-alone processing environment. IBM IIS QualityStage Designer lets us directly invoke the IBM IIS QualityStage server. IBM IIS QualityStage Real Time can be

used to integrate IBM IIS QualityStage functionality within other software applications. This feature is provided with APIs that let us run IBM IIS QualityStage jobs (built with IBM IIS QualityStage Designer) remotely on the IBM IIS QualityStage server. This option requires the purchase of an IBM IIS QualityStage Real Time license

2.2.2 Two Modes for Running QualityStage Jobs

Most QualityStage jobs can be run in either file mode or data stream mode on an OS/390 server.

1. File Mode

If file mode is used, QualityStage inputs data file into the first stage in the QualityStage job and processes the entire file before handing the results to the next stage in the QualityStage job. QualityStage also generates interim files while processing, which remain on the system.

Running in file mode has several advantages:

- Preserves intermediate results; this is a useful feature for debugging
- Allows us to generate Match reports and extracts
- May be faster if processing with a single CPU (Central Processing Unit)
- Uses less memory than data stream mode

2. Data Stream Mode

If data stream mode is used; QualityStage puts the first record in the input files into the first stage. QualityStage then passes the first record in the input files to the second stage and inputs the second record into the first stage. This process continues until all of the records have been processed by all of the stages. The only operation that is processed differently is the Sort stage, which must read and process all of the records together. Therefore, having a large number of Sort stages can slow processing considerably.

There are several advantages to using data stream mode:

- Does not produce intermediate files. Since data stream mode does not produce any intermediate files, the process takes up much less space on disk.
- Can better take advantage of multiple CPUs. Data stream mode takes advantage of multiple CPUs and parallel processing to increase the speed of processing.

However, data stream mode cannot be used:

- For Investigation stages
- To generate reports

2.2.3 Supports Data Quality Management Standards

IBM IIS QualityStage helps us to manage and maintain data quality so that any company can rely upon its corporate data investment. It helps organizations produce high-quality data as well as lower the cost of data management.

IBM IIS QualityStage enables organizations to produce high-quality, consolidated information by:

- Correlating and matching selected entities to uncover relationships.
- Automating data re-engineering tasks according to an organization's business rules and information requirements.
- Standardizing data in free-form fields and across disparate data sources.

IBM IIS QualityStage lowers costs of data management by:

- Providing control over data and the resulting quality of information.
- Uncovering information buried in free-form fields and identifying relationships between data values.
- Offering the capability to view data as related information.
- Ensuring that the data populating the new system is of the highest data quality.

2.2.4 How IBM IIS QualityStage Works

IBM IIS QualityStage provides control over data quality and enables companies to re-purpose operational data into strategic enterprise intelligence. It does this through a process that applies lexical analysis (parsing and pattern processing) and probabilistic matching according to the business rules. The process comprises:

1. Investigation

Data investigation gives 100 percent visibility into the actual condition of data, providing a sound understanding of the information in legacy sources. This lets us identify and correct data problems before they corrupt new systems.

Investigation parses and analyzes free-form and single domain fields, determining the number and frequency of unique values and classifying or assigning a business meaning to each occurrence of a value within a field. As a result, investigation:

- Uncovers trends, potential anomalies, metadata discrepancies, and undocumented business practices
- Identifies invalid or default values
- Reveals common terminology
- Verifies the reliability of fields proposed as matching criteria

2. Conditioning (standardization)

Based on the understanding of the data gleaned from investigation, conditioning standardizes and reformats data from multiple systems to create a consistent data representation with fixed, discrete fields, according to the company rules. As a result, conditioning:

- Creates fixed-field, addressable data
- Facilitates effective matching
- Enables output formatting

In conditioning, IBM IIS QualityStage provides the ability to transform any data type into desired standards. It standardizes data — applying consistent representations, correcting misspellings, and incorporating business or industry standards — and formats it, placing each value into a single domain field and transforming each single domain field into a standard format.

3. Matching

Matching ensures data integrity. Matching:

- Identifies duplicate entities (such as customers, suppliers, products, parts, etc.) within one or more files
- Creates a consolidated view of an entity
- Performs householding
- Establishes cross-reference linkage
- Enriches existing data with new attributes from external sources

IBM IIS QualityStage applies probabilistic matching technology to any relevant attribute evaluating user-defined full fields, parts of fields, or even individual characters and assigns agreement weights and disagreement weights to key data elements, based on a number of factors such as frequency distribution, discriminating value, and reliability.

It can also gauge the number of differences in a field, to account for errors such as transpositions (for example, in Social Security numbers). It can match records character by character exactly or find and match even nonexact record matches (and provide a probable likelihood that two records match), in the absence of common keys. It matches records faster and more accurately than any visual inspection or other matching tool on the market and produces auditable and legally defensible match results.

4. Survivorship and formatting

Survivorship and formatting ensure that the best available data survives and is correctly prepared for the target destination.

- Survivorship consolidates duplicate records, creating a “best of breed” representation of the matched data, enabling companies to cross-populate all data sources with the best available data.
- Formatting implements the business and mapping rules, creating the necessary output structures for the target application and identifying fields that don’t conform to load standards.

Performed on groups of matched records, survivorship creates a “best of breed” data representation. Survivorship:

- Supplies missing values in one record with values from other records on the same entity
- Resolves conflicting data values on an entity according to your business rules
- If desired, enriches existing data with data from external sources.

The result is complete, accurate surviving data. Formatting customizes the output to meet specific business and technical requirements.

2.2.5 Matching Concepts

Match is a generalized record linkage system that provides:

- Individual matching
- Geographic coding
- Many-to-one matching
- Single file grouping
- Unduplication services

All of the data management, report generation, clerical review and probability analysis functions are also provided. Match matches flat ASCII or DBF files.

Individual record linkage involves two files called *file A* and *file B*. The purpose of a record linkage application is to find records pertaining to the same person, household, event, etc., despite the fact that the files can contain missing or incorrect information.

Geographic coding involves matching a single data file to a reference file with the object of obtaining geographic location information on the data file. Match is different from previous geocoding systems in that a mathematically-justifiable methodology is used to perform the matching. This provides increased precision and flexibility.

File unduplication involves identifying all records on a single file that pertain to the same individual, household, event, etc.

2.2.5.1 Individual Matching Examples

The methodology used by Match was extensively tested by matching a sample of individuals counted in the U.S.Census to a Post Enumeration Survey with the object of determining which individuals and households were present in both the census and survey. For this application, very high precision matching is required.

Matching highway traffic accident reports filled out by the police, to injury records completed by (Emergency Medical Service) EMS and hospital personnel provides the National Highway Traffic Safety Administration (NHTSA) with a means of measuring the impact of highway safety countermeasures on medical outcome.

Matching files of cancer cases to files containing information about exposure to certain risks or to death index files can provide data about morbidity and mortality of various forms of cancer and correlations between occupation and various forms of cancers.

Matching files containing administrative data can be useful in a number of areas. For example, if juries for a court are taken from both drivers' license records and voter registration rolls, the two files should be matched to determine which individuals are on both files, so they are not twice as likely to be selected for a jury as persons on only one file.

Matching administrative files could provide data missing from one of the files. For example, a driver's license file could provide information regarding age, gender, race, etc., while a voter's file might provide Social Security Number.

As it can be seen, applications for individual matching are limited only by the imagination.

2.2.5.2 Geocoding Applications Examples

Geographic coding is used for matching a single data file against a reference file with the object of providing information that the data file lacks. In this case, a reference file record can match to many records on the data file. This is different from individual matching, where matching an individual record means that the record cannot match any other record.

A major application of geocoding is called address matching, where a data file containing street addresses is matched to a geographic reference file to obtain latitude-longitude coordinates, census tract numbers, special area codes, etc.

Reference files providing this information are readily available. Examples include the Census TIGER files, The GDT reference files, the Etak reference files, the ZIP code files, etc.

Examples of address matching applications include:

- Matching a file of cancer cases to produce a computer-generated map showing where the cases occurred
- Matching a file of park visitors to the census reference files to obtain the census tracts where the visitors reside. This can be used to prepare a demographic profile of the visitors.
- Matching a customer file to a geographic coordinate file to produce an automated map showing the location of the customers.
- Matching a file of ambulance calls or fire incidents to a file containing fire district codes to produce a map showing locations of such incidents.

- Matching a file containing address to a post office ZIP code file to correct the addresses or postal codes

To perform address matching, the files containing the addresses must be standardized so that each address component is in a separate field. The Standardize stage can be used to accomplish this.

2.2.5.3 Many-to-One Matching Examples

Many-to-one matching involves independent records on file A that can match to any record on file B. Geographic coding is an example of many-to-one matching, since any number of records on file A could match to the same record on the reference file B.

An example of many-to-one matching would be matching a purchased mailing-list against a list of customers. The mailing list can have duplicates, or many lists can have been integrated. Consequently more than one record on the mailing list can match to the list of customers.

Matching a file of sales activities, accounts receivable, etc., to the file of customers is also a many-to-one match since there might be more than one sale for a given customer

2.2.5.4 File Unduplication Examples

File unduplication is used to purge an individual file of multiple records pertaining to the same individual, household, event, etc.

The most obvious example of unduplication is in purging mailing lists of duplicates.

Other examples include:

- Removing duplicates from a file before updating or creating a database
- Removing duplicates to obtain a clean file
- Identifying households (since all individuals are duplicates within a household address)

2.2.5.5 File Grouping Examples

A variation of file unduplication is file grouping. Examples include finding all members of a household, all hospital charges associated with one patient, all customers in the same building, all affiliated customers, etc.

2.3 IBM IIS QualityStage Clerical Review

The Clerical Reconciliation Station enables users to process clerical records within the IBM IIS Enterprise Application environment. Currently the IBM IIS Enterprise applications do not provide a mechanism for reviewing and reconciling clerical records. Clerical records are records that were assigned a match weight between the clerical cutoff weight and the match cutoff weight by QualityStage Server's automatic matching.

Through the Clerical Reconciliation Station, users will elevate clerical sets to match sets, demote clerical sets to residuals or divide them in to subsets as is meaningful for the type of match.

Currently the QualityStage Server provides functionality to extract clerical records to clerical file. Users must then manually sift through these records or create their own clerical review programs to handle these clerical records.

Clerical Review allows us to closely examine clerical sets and determine if they are match sets or residuals. After examining clerical sets, users can choose to partition them into subsets meaningful to the type of match, or switch the Master Record for a set with any of its duplicates.

Using Clerical Review users can:

- Review match sets that have been designated for clerical review.
- Filter match sets to easily focus on a subset.
- Mark records as dups or residuals.

2.4 Visual Basic 6 Enterprise Edition

VB6 is a programming tool that allows programmers to design applications and system that can be stand-alone, or multi-user. With the enterprise edition programmers can produce systems that are Web-Enabled. VB6 is an excellent programming tool featuring many great wizards that will allow programmers to do much of the design work without actually writing any code.

The key to VB6 is the use of controls. Controls range from text boxes, and buttons, to data grids that combine with database to produce some excellent forms and screens. Like many of the modern day languages, VB6 allows programmers to install '3rd-party' controls thus expanding the capabilities of VB6.

The key to programming in VB6 is that the main program will only contain a few lines; this main program being used to open forms, call procedures and functions etc. Much of the code is attached to the objects that are called from the main program. Where possible, common functions or routines will be separated from the controls allowing for multiple controls to use these routines.

VB6 is a very structured language that uses much of the modern technologies, such as Classes, COM, ODBC, ASP, etc. Because a Visual Basic application is based on objects, the structure of its code closely models its physical representation on screen. By definition, objects contain data and code. For each form in an application, there is a related form module (with file name extension .frm) that contains its code.

Each form module contains event procedures — sections of code where place the instructions that will execute in response to specific events. Forms can contain controls. For each control on a form, there is a corresponding set of event procedures in the form module. In addition to event procedures, form modules can contain general procedures that are executed in response to a call from any event procedure.

Code that isn't related to a specific form or control can be placed in a different type of module, a standard module (.BAS). A procedure that might be used in response to events

in several different objects should be placed in a standard module, rather than duplicating the code in the event procedures for each object.

A class module (.CLS) is used to create objects that can be called from procedures within the application. Whereas a standard module contains only code, a class module contains both code and data — you can think of it as a control without a physical representation.

- **Form Modules**

Form modules (.FRM file name extension) are the foundation of most Visual Basic applications. They can contain procedures that handle events, general procedures, and form-level declarations of variables, constants, types, and external procedures. The code that write in a form module is specific to the particular application to which the form belongs; it might also reference other forms or objects within that application.

- **Standard Modules**

Standard modules (.BAS file name extension) are containers for procedures and declarations commonly accessed by other modules within the application. They can contain global (available to the whole application) or module-level declarations of variables, constants, types, external procedures, and global procedures. The code that write in a standard module isn't necessarily tied to a particular application; a standard module can be reused in many different applications.

- **Class Modules**

Class modules (.CLS file name extension) are the foundation of object-oriented programming in Visual Basic. Code can be written in class modules to create new objects. These new objects can include our own customized properties and methods. Actually, forms are just class modules that can have controls placed on them and can display form windows.

2.4.1 Interface Styles

There are two main styles of user interface: the single-document interface (SDI) and the multiple-document interface (MDI). An example of the SDI interface is the WordPad application included with Microsoft Windows (Figure 2.1). In WordPad, only a single document may be open.

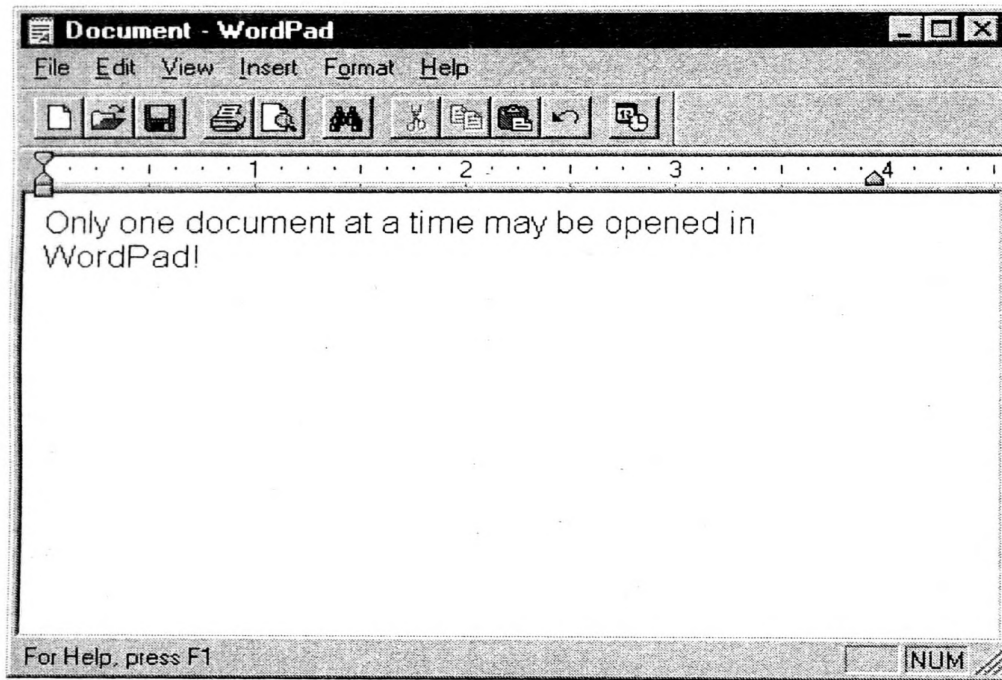


Figure 2.1 WordPad, a single-document interface (SDI) application

Applications such as Microsoft Excel and Microsoft Word for Windows are MDI interfaces; they allow to display multiple documents at the same time, with each document displayed in its own window (Figure 2.2). A MDI application can be recognized by the inclusion of a Window menu item with submenus for switching between windows or documents.

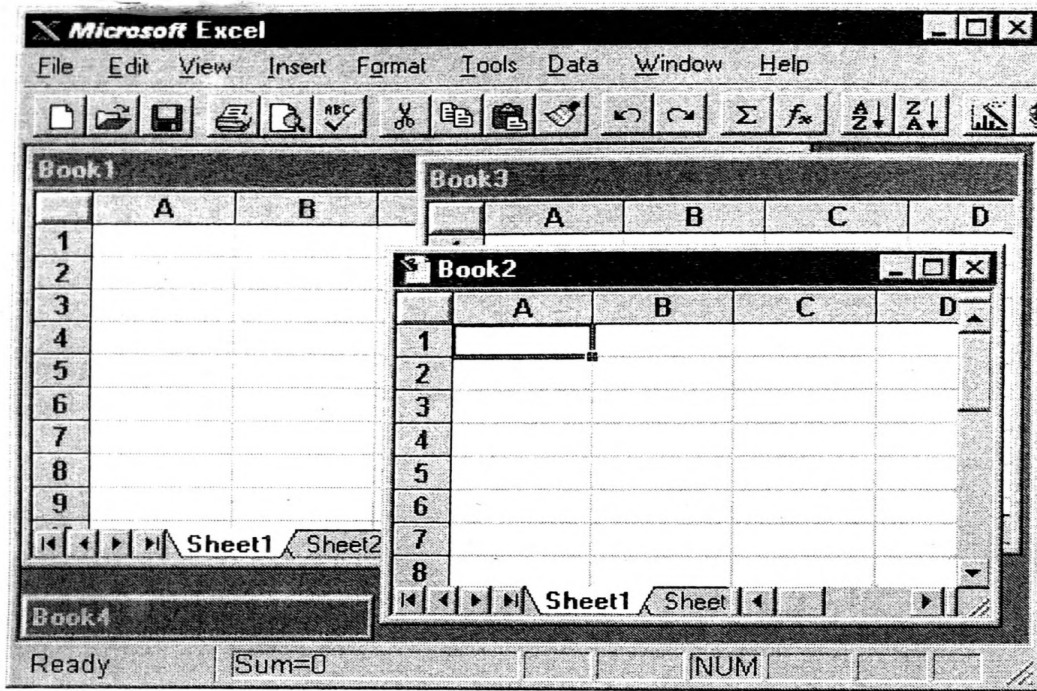


Figure 2.2 Microsoft Excel, a multiple-document interface (MDI) application

In addition to the two most common interface styles, SDI and MDI, a third interface style is becoming more popular: the explorer-style interface (Figure 2.3). The explorer-style interface is a single window containing two panes or regions, usually consisting of a tree or hierarchical view on the left and a display area on the right, as in the Microsoft Windows Explorer. This type of interface lends itself to navigating or browsing large numbers of documents, pictures, or files.

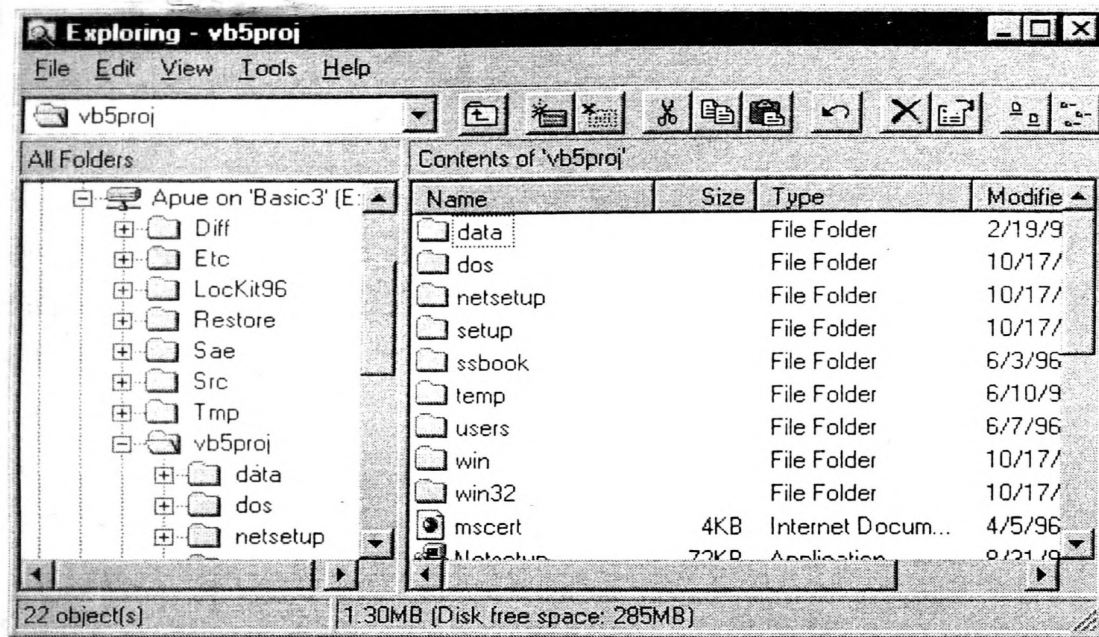


Figure 2.3 The Windows Explorer, an explorer-style interface

2.4.2 Working with Multiple Projects

Many applications can be created by working with a single project. However, as applications become more complex, programmers may want to work with multiple projects in the same session of the programming environment. For example, programmers may want to use one project to build an application's executable file, and a second project to serve as a "scratch pad" for testing code before add it to the application.

A new or existing project can be added to the current editing session by adding it to a project group. Then the project group can be saved and can be worked with it in subsequent editing sessions. Either the project group or an individual project in the project group can be opened, or the project group or its individual projects can be added to another project group.

In a project group, one executable project serves as a startup project. project groups can be used to create and debug multiple-component applications. For example, project groups containing standard executable projects, ActiveX executable projects, ActiveX dynamic-link library projects, or ActiveX control projects can be created and debugged.

2.4.3 Managing Application Settings

In Microsoft Windows 3.1 and earlier versions of Windows, program settings like window positions, files used, and other items were commonly stored in .ini files. In Windows NT, Windows 95, and later versions of Windows these program settings are stored in the system registry.

Visual Basic provides a standard registry location for storing program information for applications created in Visual Basic:

HKEY_CURRENT_USER\Software\VB and VBA Program Settings\appname\section\key

Visual Basic also provides four statements and functions to manipulate the settings stored in application's registry location.

Function or Statement	Description
GetSetting function	Retrieves registry settings.
SaveSetting statement	Saves or creates registry settings.
GetAllSettings function	Returns an array containing multiple registry settings.
DeleteSetting statement	Deletes registry settings.

Table 2.1 Visual Basic Functions for Accessing Registry

2.4.4 Using Conditional Compilation

Conditional compilation lets selectively compile certain parts of the program. Specific features can be included of the program in different versions, such as designing an application to run on different platforms, or changing the date and currency display filters for an application distributed in several different languages.

Structuring Code for Conditional Compiling

To conditionally compile a code, enclose it between `#If...Then` and `#EndIf` statements, using a Boolean constant as the branching test. To include this code segment in compiled code, set the value of the constant to `-1` (True).

For example, to create French language and German language versions of the same application from the same source code, embed platform-specific code segments in `#If...Then` statements using the predefined constants `conFrenchVersion` and `conGermanVersion`.

```
#If conFrenchVersion Then
  ' <code specific to the French language version>.
#Elseif conGermanVersion then
  ' <code specific to the German language version>.
#Else
  ' <code specific to other versions>.
#End If
```

Declaring Conditional Compilation Constants

There are three ways to set conditional compilation constants: in the Conditional Compilation Arguments field of the Make tab on the Project Properties dialog box, on a command line, or in code.

Conditional compilation constants have a special scope and cannot be accessed from standard code. How programmer set a conditional compilation constant may depend on the scope programmer want the constant to have.

How Set	Scope
Project Properties dialog box	Public to all modules in the project
Command line	Public to all modules in the project
Const statement in code	Private to the module in which they are declared

Table 2.2 Scope of each Conditional Compilation Constants

2.4.5 Working with Resource Files

A resource file allows collecting all of the version-specific text and bitmaps for an application in one place. This can include icons, screen text, and other material that may change between localized versions or between revisions or specific configurations.

Adding Resources to a Project

A resource file can be created using the Resource Editor add-in. The compiled resource file will have a .res file name extension. Each project can contain only one resource file.

The actual file consists of a series of individual strings, bitmaps, or other items, each of which has a unique identifier. The identifier is either a Long or a String, depending on the type of data represented by the resource. Strings, for example, have a Long identifier, while bitmaps have a Long or String identifier. The function parameters referring to the resources can use the Variant data type.

2.4.6 Using Enumerations to Work with Sets of Constants

Enumerations provide a convenient way to work with sets of related constants and to associate constant values with names. For example, an enumeration can be declared for a set of integer constants associated with the days of the week, and then the names of the days can be used in code rather than their integer values.

An enumeration can be created by declaring an enumeration type with the Enum statement in the Declarations section of a standard module or a public class module. Enumeration types can be declared as Private or Public with the appropriate keyword. For example:

```
Private Enum MyEnum
```

```
-OR-
```

```
Public Enum MyEnum
```

By default, the first constant in an enumeration is initialized to the value 0, and subsequent constants are initialized to the value of one more than the previous constant.

```
Public Enum Days
    Sunday
    Monday
    Tuesday
    Wednesday
    Thursday
    Friday
    Saturday
End Enum
```

The following code refers to the Saturday constants in the Days and WorkDays enumerations, displaying their different values in the Immediate window.

```
Debug.Print "Days.Saturday = " & Days.Saturday
Debug.Print "WorkDays.Saturday = " & WorkDays.Saturday
```

After an enumeration type is declared, a variable of that type can be declared, and then the variable is used to store the values of enumeration's constants

```
Dim MyDay As WorkDays
MyDay = Saturday ' Saturday evaluates to 0.
If MyDay < Monday Then ' Monday evaluates to 1,
    MsgBox "It's the weekend. Invalid work day!" ' so Visual Basic displays a message box.
End If
```

2.4.7 The Many (Inter)Faces of Code Reuse

There are two main forms of code reuse — binary and source. Binary code reuse is accomplished by creating and using an object, while source code reuse is achieved by inheritance, which isn't supported by Visual Basic. (Source code reuse can also be achieved by copying and modifying the source code, but this technique is nothing new, and has many well-known problems.)

The code is reused in a control by placing an instance of the control on the form. This is known as a containment relationship or a has-a relationship; that is, the form contains or has a CommandButton.

Delegating to an Implemented Object

Implements provides a powerful new means of code reuse. An abstract class can be implemented, or the interface of a fully functional class can be implemented. Inner object (that is, the implemented object) can be created in the Initialize event of the outer object (that is, the one that implements the inner object's interface).

An interface is like a contract — must be implemented all the members of the inner object's interface in the outer object's class module. However, It is very selective in the way it is delegated to the properties and methods of the inner object. In one method it might delegate directly to the inner object, passing the arguments unchanged, while in another method it might execute some code of its own before calling the inner object — and in a third method it might execute only its own code, ignoring the inner object altogether!

For some of the methods, implementation may delegate directly to the inner Cacophony object, while for others may be interposed its own code before and after delegating — or even omit delegation altogether, using entirely its own code to implement a method.

COM provides another mechanism for binary code reuse, called aggregation. In aggregation, an entire interface is reused, without any changes, and the implementation is provided by an instance of the class being aggregated. Visual Basic does not support this form of code reuse.

2.5 Design Patterns

A design pattern is a template for software development. The purpose of the template is to define a particular behavior or technique that can be used as a building block for the construction of software - to solve universal problems that commonly face developers. Think of design code as a way of passing on some nifty piece of advice, just like your mother used to give. "Never wear your socks for more than one day" might be an old family adage, passed down from generation to generation. Its common sense solutions

that are passed on to others. We considered a design pattern as a useful piece of advice for designing software.

A pattern is a 'best practice solution to a common recurring problem'. That is, a pattern documents and explains an important or challenging problem that can occur when designing or implementing an application, and then discusses a best practice solution to that problem. Over time, patterns begin to embody the collective knowledge and experiences of the industry that spawned it.

There are many uses of patterns, but the following are some of the most important benefits:

1. Helps provide a high-level language for discussing design issues.
2. Provides much of the design work upfront.
3. Combinations of patterns lend themselves to reusable architectures.

Although not a magic bullet (if such a thing exists), design patterns are an extremely powerful tool for a developer or architect actively engaged in any development project. Design patterns ensure that common problems are addressed via well-known and accepted solutions. The fundamental strength of patterns rests with the fact that most problems have likely been encountered and solved by other individuals or development teams. As such, patterns provide a mechanism to share workable solutions between developers and organizations. Regardless of where these patterns find their genesis, patterns leverage this collective knowledge and experience. This ensures that correct code is developed more rapidly and reduces the chance that a mistake will occur in design or implementation. In addition, design patterns offer common semantics between members of an engineering team. As anyone who has been involved in a large-scale development project knows, having a common set of design terms and principles is critical to the successful completion of the project. Best of all, design patterns—if used judiciously—can free up your time.

2.5.1 Singleton Design pattern

Often in designing a system, we want to control how an object is used, and prevent others (ourselves included) from making copies of it or creating new instances. For example, a central configuration object that stores setup information should have one and one only instance - a global copy accessible from any part of the application, including any threads that are running. Creating a new configuration object and using it would be fairly useless, as other parts of the application might be looking at the old configuration object, and changes to application settings wouldn't always be acted upon. It's a common enough design criteria (not used everyday, but we'll come across it from time to time

In software engineering, the singleton design pattern is used to restrict instantiation of a class to one (or a few) objects. This is useful when exactly one object is needed to coordinate actions across the system. Sometimes it is generalized to systems that operate more efficiently when only one or a few objects exist. It is also considered an anti-pattern since it is often used as a euphemism for global variable. Before designing a class as a singleton, it is wise to consider whether it would be enough to design a normal class and just use one instance.

The singleton pattern is implemented by creating a class with a method that creates a new instance of the object if one does not exist. If an instance already exists, it simply returns a reference to that object. To make sure that the object cannot be instantiated any other way, the constructor is made either private or protected. Note the distinction between a simple static instance of a class and a singleton. Although a singleton can be implemented as a static instance, it can also be lazily constructed, requiring no memory or resources until needed.

The singleton pattern must be carefully constructed in multi-threaded applications. If two threads are to execute the creation method at the same time when a singleton does not yet exist, they both must check for an instance of the singleton and then only one should create the new one. If the programming language has concurrent processing capabilities the method should be constructed to execute as a mutually exclusive operation.

2.5.2 Logical Model

The model for a singleton is very straightforward. There is (usually) only one singleton instance. Clients access the singleton instance through one well-known access point. The client in this case is an object that needs access to a sole instance of a singleton. Figure 2.4 shows this relationship graphically.

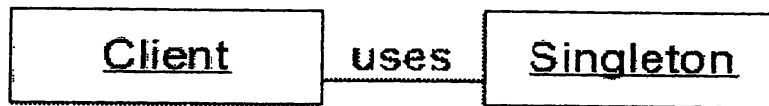


Figure 2.4 Singleton pattern logical model

2.5.3 Physical Model

The physical model for the Singleton pattern is also very simple. However, there are several slightly different ways that singletons have been implemented over time. Figure 2.5 shows a UML model of the original Singleton pattern as defined in Design Patterns.

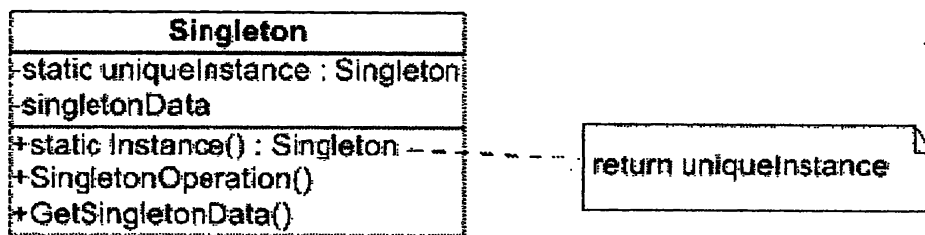


Figure 2.5 Singleton pattern physical model from design patterns

What we see is a simple class diagram showing that there is a private static property of a singleton object as well as public method Instance() that returns this same property. This is really the core of what makes a singleton. The other properties and methods are there to show additional operations that may be allowed on the class.

2.6 Usability

Shackel (1991) defines the usability of a system as 'the capability in human functional terms to be used easily and effectively by the specified range of users to fulfill the specified range of tasks, within the specified range of environmental scenarios'.

Ratner (2003) states "Measures of usability, are measures of support, that is provided by the software, for tasks, which the users carry out. During the development process, potential user participation can facilitate the integration of task knowledge and user requirements into software design." These factors should be taken into consideration for this project

- User motivation to use the Clerical Reconciliation Station.
- User acceptance of the design after the tries in the prototype.
- Validation of task- the level to which the task represented by the interface matches the task understood by user and the task supported by the system.
- Task coverage- the level to which system facilities cover all the users' task.
- Context fit- the level to which interaction between user and system matches environmental demands.

Chapter 3 : Software Development Process

3.1 Introduction

A software development process is a set of activities to transform the user's requirements into a software system. The software development process used for this project was Rational Unified Process. "The Unified Process is a generic process framework that was designed with a wide scope of applicability to accommodate variations in project size, application domain, organizations, competence levels and technologies used".

To be effective, an iterative and incremental process, adopted from Unified Process, was used throughout each phase.

3.2 Rational Unified Process

The Rational Unified Process is a software development process. The Unified Process is adapted to the project because of its special features such as Use Case driven, component-based, iterative and incremental.

"Use-case driven" means that the development process follows a series of workflows that derive from Use Cases. (Jacobson et al 1999) A Use Case is a piece of functionality in the system that gives a user a result of value. Use Cases capture functional requirements. All Use Cases together form a Use Case Model which describes the complete function of the system.

In this project, requirements will be captured by referencing from related applications. Also, by interviewing problem owners these will then be converted into Use Cases.

The Unified Process is component-based, which means that the software system being built is made up of software components interconnected via well-defined interfaces. (Jacobson et al 1999) Amendments and Improvements can be easily made in individual components to improve usability and maintainability.

The Unified Process suggests dividing work into smaller slices or mini projects. Each mini project is an iteration that results in an increment. Iterations refer to steps in the workflow while increments refer to growth in the system. (Jacobson et al 1999)

Figure 3.1 shows the emphasis of workflows and the time schedule for each workflow. In the inception phase, more time is suggested to spend on requirements in early iteration and less time should be spent on implementation. Analysis, design and implementation are emphasized in elaboration phase. Concentration should be put in the implementation, and testing phase and deployment in construction phase while the transition stage should deal with the changes in the deployed product.

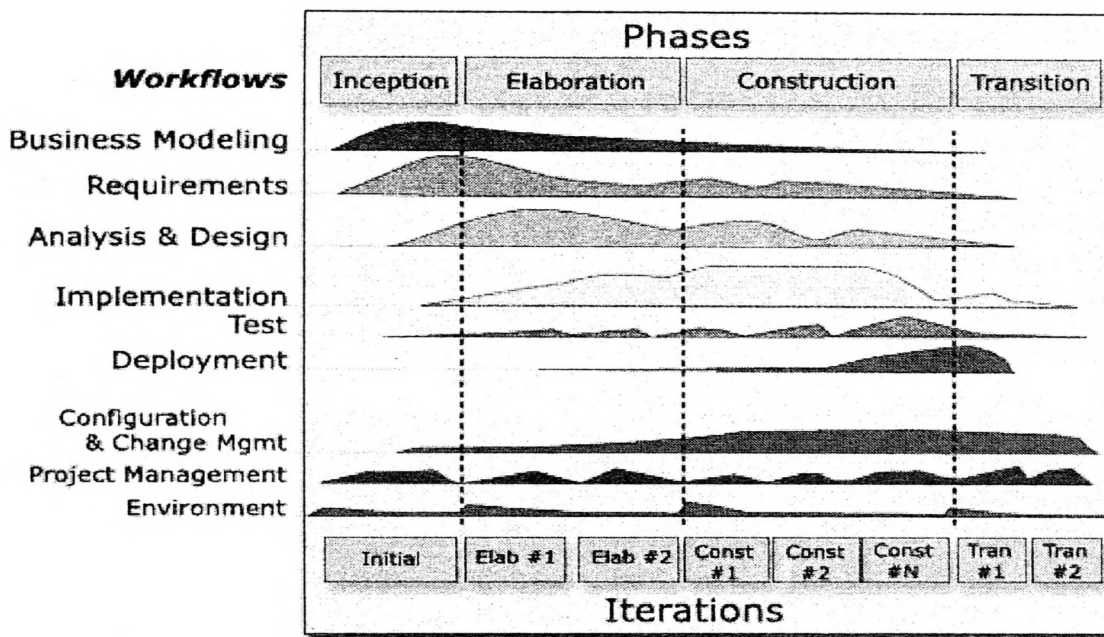


Figure 3.1 A graphical overview of the RUP

The full description of each phase of the project will be shown in chapter 4-7.

Chapter 4 : Requirement Analysis

4.1 Introduction

This chapter specifies the requirements for Enhancement to data clenching features in IBM data integration suit. To be effective, an iterative and incremental process, adopted from Unified Process, is to be used throughout this phase.

Due to the maintenance stage is currently going on, it was difficult to estimate the number of iterations was taken throughout this project. Throughout each meeting with problem owner, who is also a potential user of this system, requirements of the system and uncovered issues which would be incorporated into the system were discussed. Based on this comments, changes of functional and non-functional requirements were carried out as iteration.

This is done on the basis of information provided by the client in the form of documents, existing systems and process specs, on-site analysis interviews with end-users, market research and competitor analysis. This stage has the following steps:

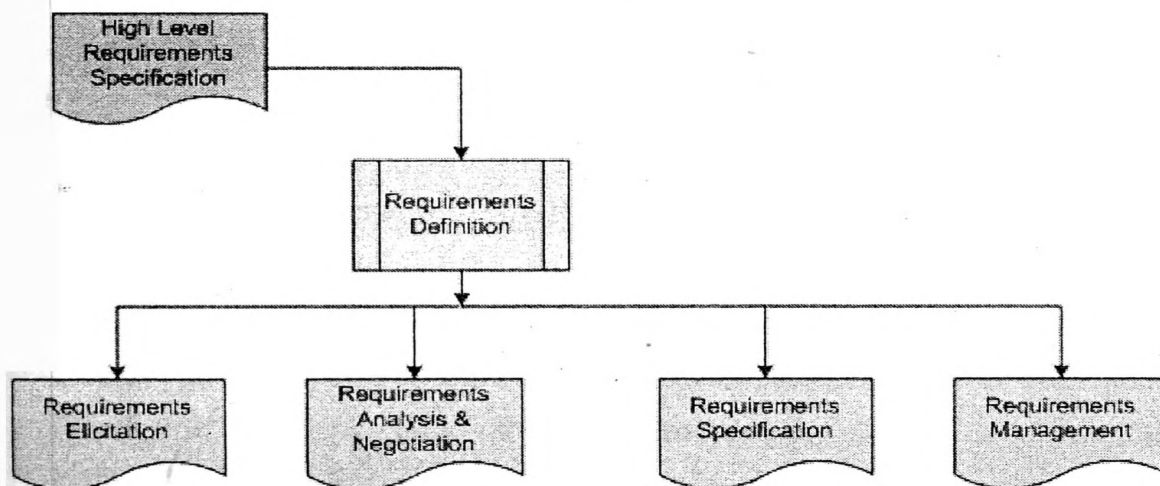


Figure 4.1 Requirement Analysis Process

1. Requirements Elicitation

This is the process of gathering customer needs. This involves asking the customers, users and others about the objectives of the system, what is to be accomplished, how the system fits into the needs of the business and finally how the system will be used.

2. Requirements Analysis

This is the process of understanding the problem and the requirements for a workable solution.

Once the requirements have been gathered they become the basis for “Requirements Analysis”. Analysis categorizes requirements and organizes them into related subsets, explores each requirement in relationship to others, examines requirement for consistency, omissions and ambiguity, and prioritizes requirements based on the needs of the customer. Rough estimates of development are made and used to assess the impact of each requirement on project cost and delivery time. Using an iterative approach, requirements are eliminated, combined, and/or modified so that each party achieves some measure of satisfaction. The requirements are used to generate Business Process Flows, Use Cases Modeling and Data Flow diagrams which facilitates a clearer understanding of the requirements and its solution, for both the customer and the developer.

3. Requirements Specification

This is the process of describing what a system will do. It involves scoping the requirements so that it meets the customer’s vision. Requirements Specification serves as a foundation software, hardware and database design. It describes the function (Functional and Non-Functional) and performance of the system and the constraints that will govern its development. It specifies the inputs and also describes the outputs. These specifications need to be

- Complete
- Comprehensive
- Testable
- Consistent
- Unambiguous
- Writable
- Modifiable
- Implementable

4. Requirements Management

This is the process that helps the project team identify, control, and track changes to the requirements at any time as the project proceeds. Requirements Verification, Validation and Traceability examine the specification to ensure that all system requirements have been stated unambiguously and that inconsistencies, omissions and errors have been detected and corrected. Thus, ensuring that the work products confirms to the standards established for the process, the project and the system.

4.2 Requirement Capture

The aim of the system was to:

1. Provide functionality for Reviewing match sets that have been designated for clerical review.
2. Provides functionality for Filtering match sets to easily focus on a subset.
3. Mark records as dups or residuals.

In order to achieve these, the following fact finding activities were carried out to identify the requirements for the system.

1. Formal meetings with problem owner
2. Reading from sample documents
3. Reviewing similar systems
4. Throughout feedback mails from clients of the IBM IIS QualityStage

4.3 Functional Requirements

Functional requirements describe what a system does or are expected to do. From the above requirement capture section; users can use the systems to perform activities that provided values to him or her to enhance the provision of such values. Those are the requirements of the project.

In the diagram, potential users including programmers, programmer analysts, business analysts, and others re-engineer data to meet business objectives and data quality management standards, and their corresponding tasks within the system were defined and presented in Use Case Diagram.

A high level functional view of the system is shown as figure 4.1. The diagram shows potential users of the system with the associated tasks they will perform with the system.

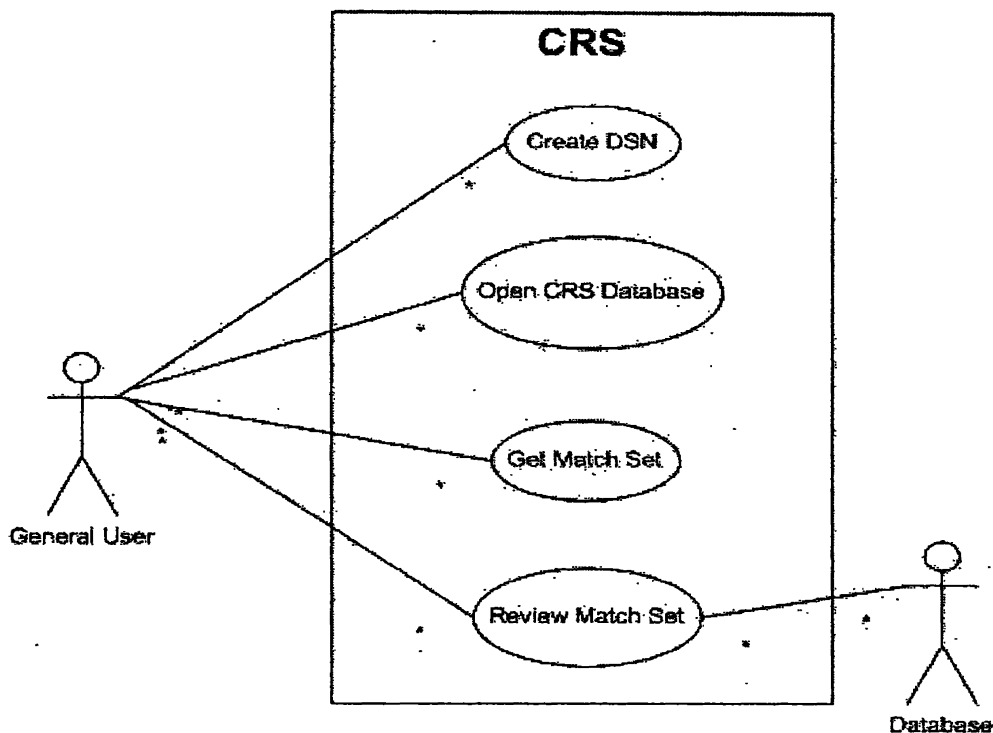


Figure 4.2 System Use case Diagram

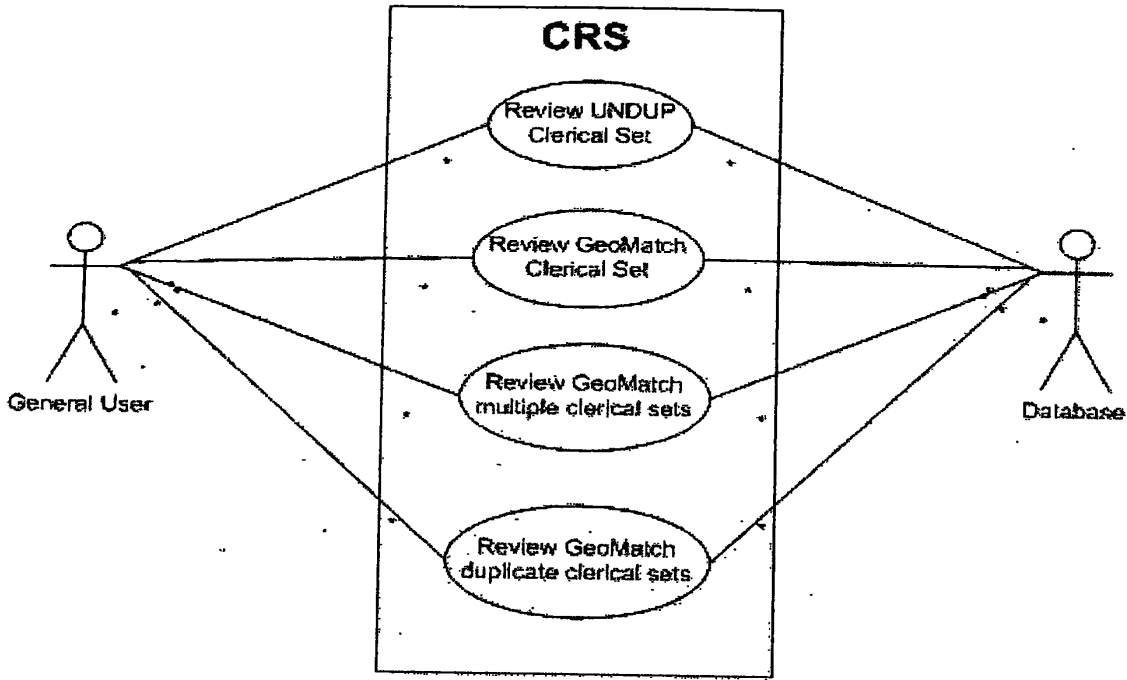


Figure 4.3 Use case for Review Match Set

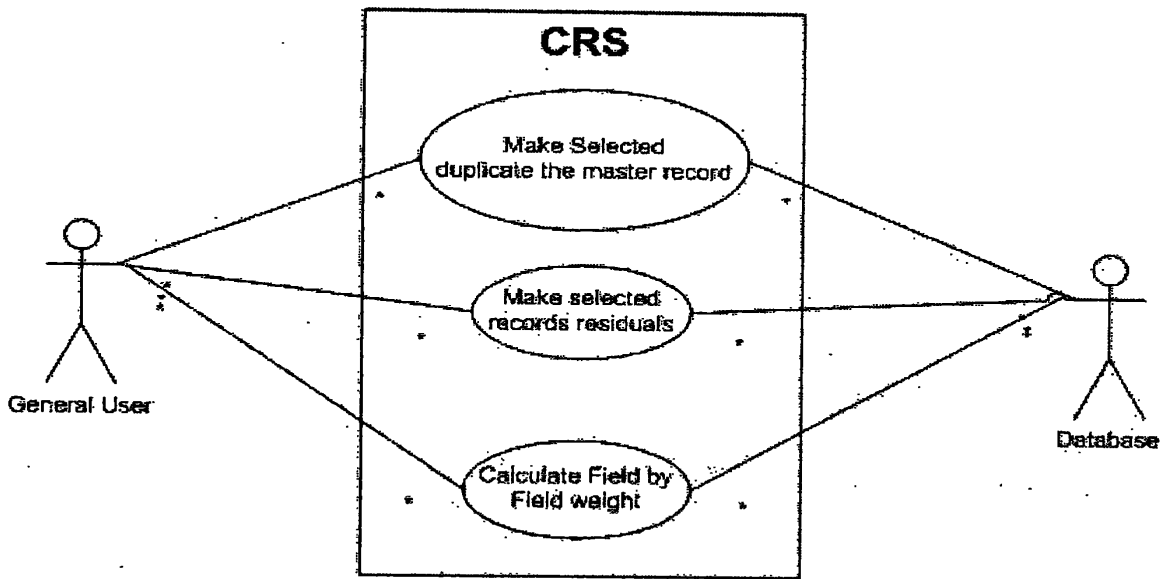


Figure 4.4 Use case for Review Unduplication Clerical Set

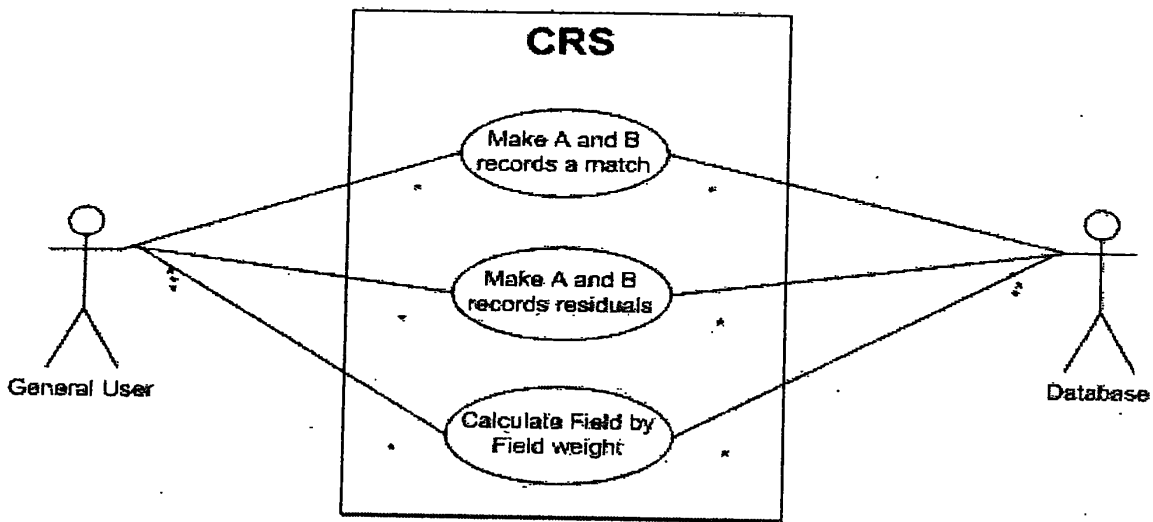


Figure 4.5 Use case for Review GeoMatch Clerical Set

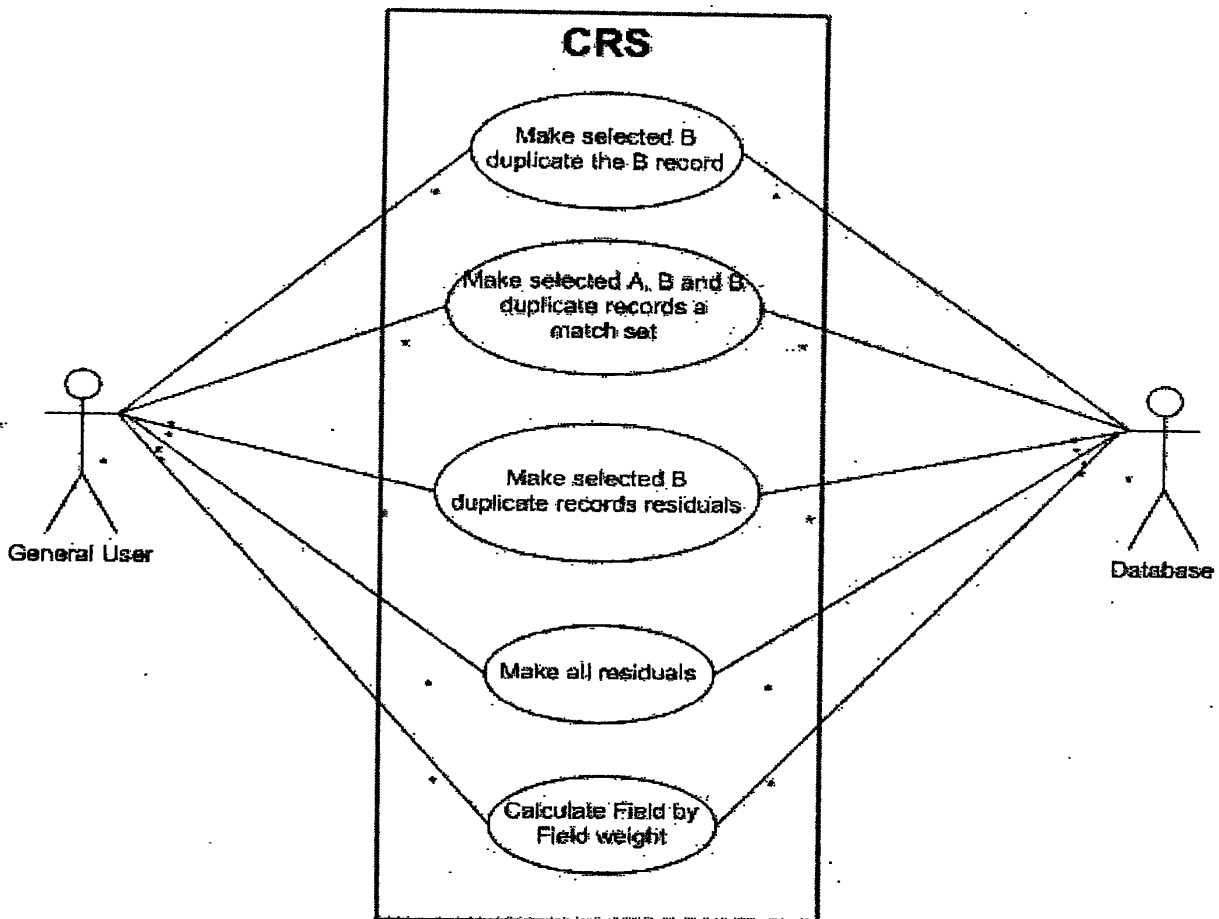


Figure 4.6 Use case for Review GeoMatch Multiple Clerical Set

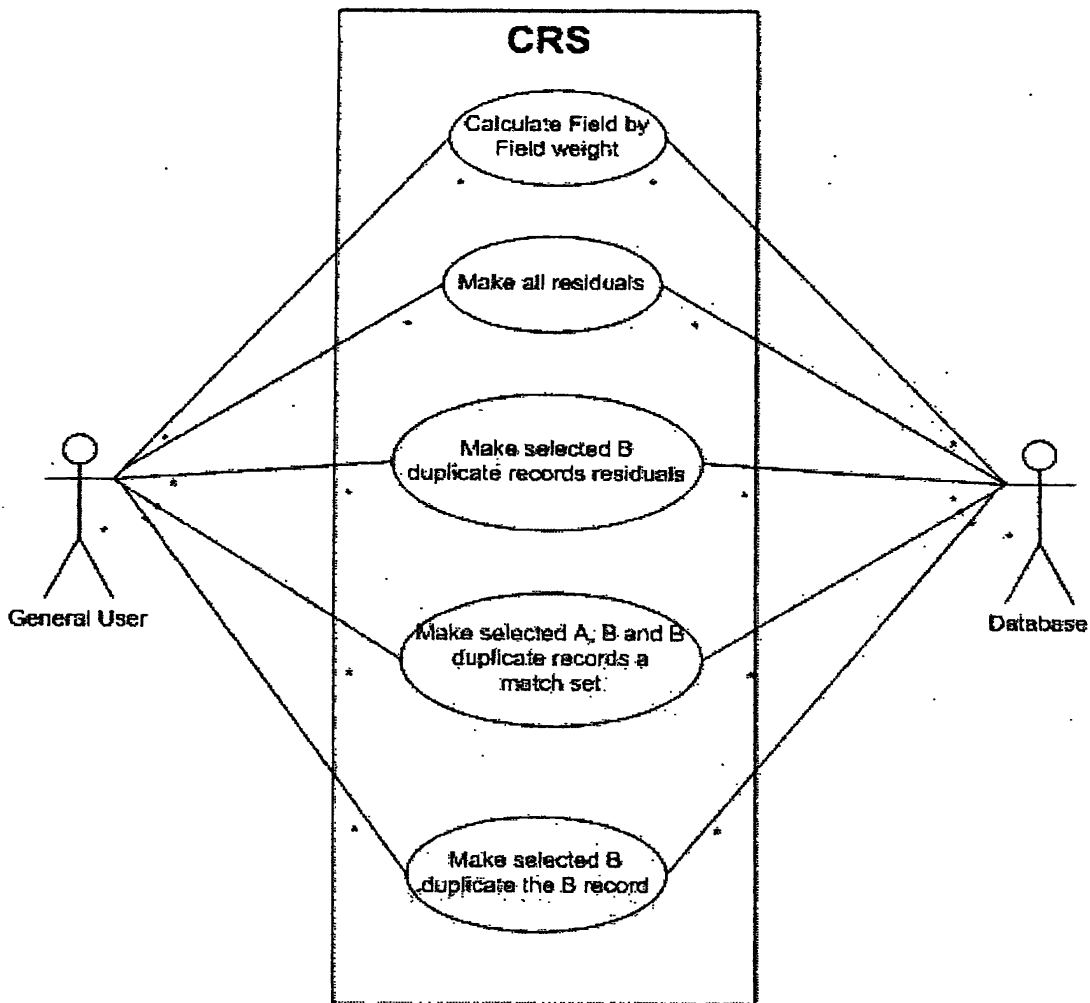


Figure 4.7 Use case for Review GeoMatch Duplicate Clerical Set

4.4 Design Goals and Constraints

In this project, the Clerical Reconciliation Station enables users to process clerical records within the IBM IIS Enterprise Application environment. Currently the IBM IIS Enterprise applications do not provide a mechanism for reviewing and reconciling clerical records. Clerical records are records that were assigned a match weight between the clerical cutoff weight and the match cutoff weight by QualityStage Server's automatic matching.

Through the Clerical Reconciliation Station, users will elevate clerical sets to match sets, demote clerical sets to residuals or divide them in to subsets as is meaningful for the type of match.

4.5 Non-Goals

Clerical reconciliation station will not:

- Maintain a history of changes to apply to subsequent runs.
- Provide rules for automatically handling sets or records.
- Support interwoven review and match design changes.

The Clerical Reconciliation Station is not meant to provide direct support for QualityStage Match design.

4.6 Assumptions

- There will be no major changes to the Match process in the QualityStage Designer or QualityStage Server applications.
- There will be no changes to the QualityStage Designer and QualityStage Server control file formats. This includes Dictionary files (.DIC), Extraction Files (.EXT) and frequency files (.FRQ).
- The database server's data/time value will be used to determine date/time values used with the CRS database. It will be assumed that CRS database server's date/time value is correctly adjusted.
- To calculate the field-to-field MATCH weights between records, we are making an assumption about the non-changing nature of the MATCH specification and frequency information here. If the frequency file or MATCH specification changes, CRS could produce different weights. If the MATCH specification changes, CRS could even have different comparisons and fields from when the clerical file was generated. It is assumed for this release that the frequency and MATCH specification are not changed from the first run of the match that produced the clerical results for loading into the Clerical Reconciliation Station database.

Chapter 5: Design

5.1 Introduction

The aim of the design stage of the development process was to translate the analysis models from chapter 4 to a design model. Activities concerned in this stage included high-level system architecture, platform selection, attributes and operations, human computer interactions, user interfaces, data storage and component frameworks.

To suit the usability and maintainability requirements, iterative development approach were used to produce a better solution to the system. Each iteration was gone through a meeting with problem owner. Re-consideration, new construction and modification were taken place through this phase so as to get user satisfaction.

Here are the artifacts produced to verify the suitability of the system design:

1. High Level Architecture
2. User Interface Description
3. Process Logic Description
4. CRS.DBAccess Library design
5. CRS Database Design, Load and Extract
6. CRS Weight Calculations
7. Quality Stage Designer Changes

5.2 High Level Architecture

This section describes the system architecture in high level. It depicts all major components, technologies and how they interface to external systems. Some of the Clerical Reconciliation Station's initialization processes occur in the QualityStage Designer and QualityStage Server applications. Hence, they are considered in the high level architecture diagram.

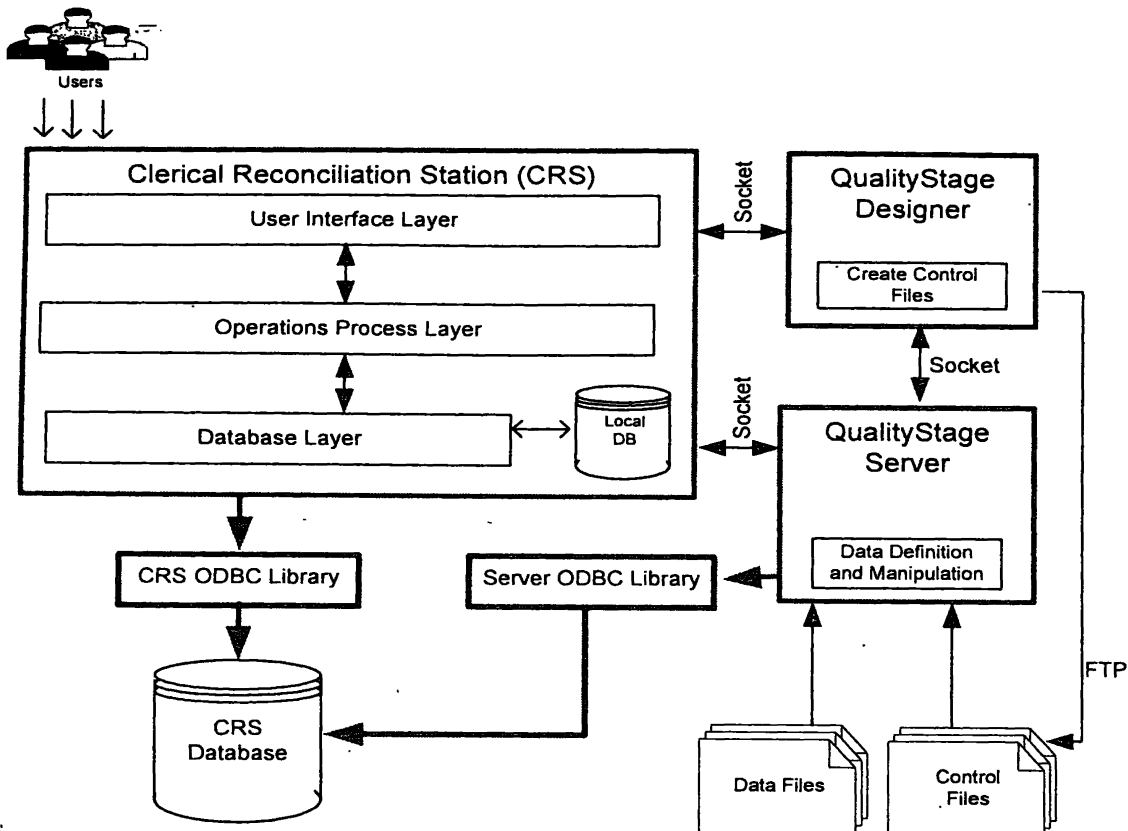


Figure 5.1 Clerical Reconciliation Station high level architecture view

5.2.1 QualityStage Designer Process

Although the Clerical Reconciliation station works as standalone application, some of its initial process are defined and performed in the QualityStage Designer and Server applications.

5.2.1.1 Specify Use of CRS

A clerical file generation option will be on the MATCH extract stage wizard screen. Upon user selection of this option (selected by default) and completion of the wizard process, all MATCH variable fields and only those fields that appear in the residual, master, and duplicate match extract specification will be added to the clerical extract specification. This will minimize the storage of unnecessary information in the CRS database, but changes to the match extract could result in fields in the match extract not

being in the CR database. As assumptions about the static nature of production environment match runs have been made for the CRS, this will not be an issue.

The QS Server will then populate the CRS database.

New screen controls and functionality will be added to the 'QS designers' MATCH extract specification wizard screen.

5.2.1.2 Specify Relational Database to store clerical data

New functionality will be added to the Quality Stage Designer's Advanced Run Options window to enter relational database information. This information includes the DSN name, database user id and password for the given DNS, etc. The given DSN should be pre-configured on the QualityStage server and known to QualityStage Designer user.

The relational database specified by the DSN will be the Clerical Reconciliation Station's data repository. Both the Clerical Reconciliation Station (CRS) and QualityStage Server will access this database via an ODBC driver library.

The QualityStage Designer adds the DSN and database connectivity information into a new control file, *<Stage Name>.CRI* for use by the Clerical Reconciliation Load stage. This stage is described in detail in a later portion of this document. This control file is placed in the Project directory along with the pre-existing control files. The QualityStage Server will use this information to access the CRS database.

The ".CRI" Control file will be created for all MATCH stages associated with the RUN during deployment process. The DSNNAME and other parameter values will be empty for MATCH-stages which does not have a CR DSN associated with.

5.2.2 QualityStage Server Process

The QualityStage Server will be responsible for populating the Clerical Reconciliation (CR) database used by CRS application.

5.2.2.1 Create CRS Data tables

This operation will be performed by the QS Server. The CR database table creation step will use the MATCH stage's control files. These files contain DSN information, clerical output fields, their types, their sizes, etc. The QS Designer creates these control files during the match job deployment [Staging] and uploads them to the QS Server during the match job deployment [Staging]. The only exception is the .env file which is uploaded to the QS Server during the match job run [Run].

The match jobs that populate the CR database are assumed not to change, as the expectation is that the CRS will only be used in a production environment where changes to the match should not occur. Neither the blocking nor matching fields should change from match job run to match job run. The match extract specification is also assumed to be static. Although the expectation of a non-changing match is assumed, the QS server will attempt to protect against the insertion of records into the CR table that are incompatible with existing records from previous runs of the match.

During the CR database table creation step, the QS Server looks for existing tables for a given match. If there is an existing table, the QS server will ensure that the schema is identical before inserting clerical records. The QS Server will also check the match design specification, the match extract specification and the frequency files to determine if they differ from the any pre-existing versions in the CR database for the match. If the schema or any of these other specifications differ, the user will be informed that the match must be renamed or the existing tables in the CR database should be purged before the match is run again. The purging should occur after completing clerical reconciliation on the existing entries in the CR database and extracting the entries from the CR database, or after a conscious decision is made by the user to destroy the existing clerical entries in the CR database that have not been processed or extracted.

Functionality will be available on the CRS to purge the existing match tables from the CRS database.

5.2.2.2 Populate CRS tables with Clerical data

The CR database population will occur following the match extraction process, during the Clerical Reconciliation Load stage. At this time all the control files were uploaded to the QS project directory and a successful match run generated clerical data records.

The QS Server's CRI stage processes clerical data streams and parse the streams with the use of data dictionary [.DIC] files. This data will be used to populate clerical database tables.

These data dictionary files are in the QS Project directory.

5.3 User Interface Description

The Clerical Reconciliation Station's main window contains a menu and a toolbar with standard windows functionality. On the main screen, the user has the option to select pre-configured ODBC DSNs for CRS Relational Databases.

When connected to the database, the application searches for CRS tables. Upon a successful search, the CRS main screen will be displayed. Upon an unsuccessful search, the CRS tables' not available message will be displayed. This state allows user to select another DNS from the CRS or add a new DSN to CRS.

The main screen displays match sets in a data grid and allows user to select them one by one. The match set filter option available on the top of the main screen allows user to filter clerical sets. Upon review Task bar/button/menu item/toolbar option click, data items belongs to the selected match set will be displayed in the detail area in the same screen.

Reconciliation operations can be done on the data items and match type will determine the possible operations on them. These operations can be done through the toolbar options, task bar options, menu options or available buttons.

Database admin functionality will be available on the CRS application to purge/clear CRS database.

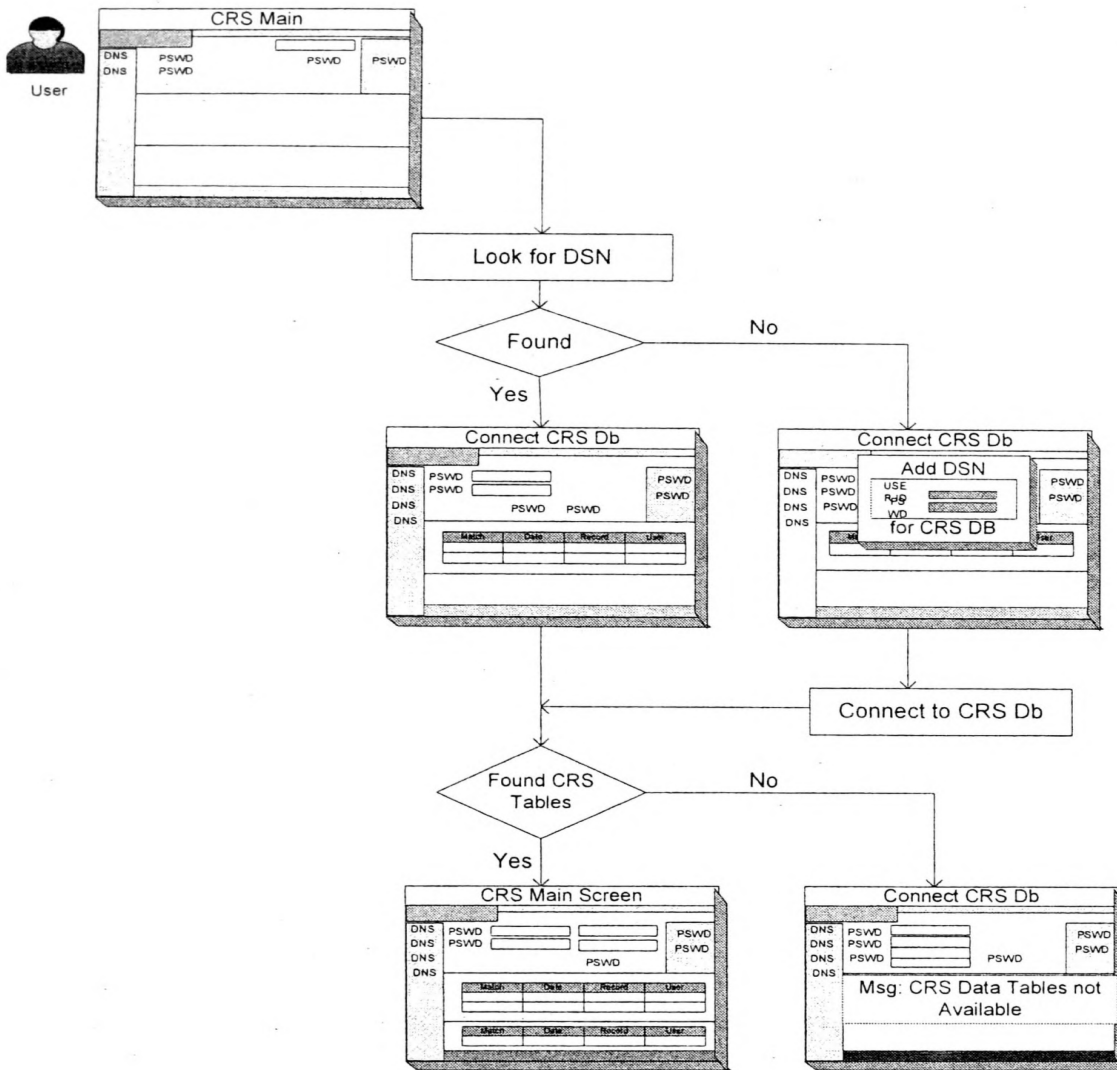


Figure 5.2 Clerical Reconciliation Station's screen layout

5.4 Process Logic Description

This section describes the operation process of the CRS. The Decomposition view and the Process view are two different views of the same system.

5.4.1 Decomposition View

This section describes how the system (Clerical Reconciliation Station, QualityStage Designer's Match Specification area and QualityStage Server) is decomposed to smaller subcomponents.

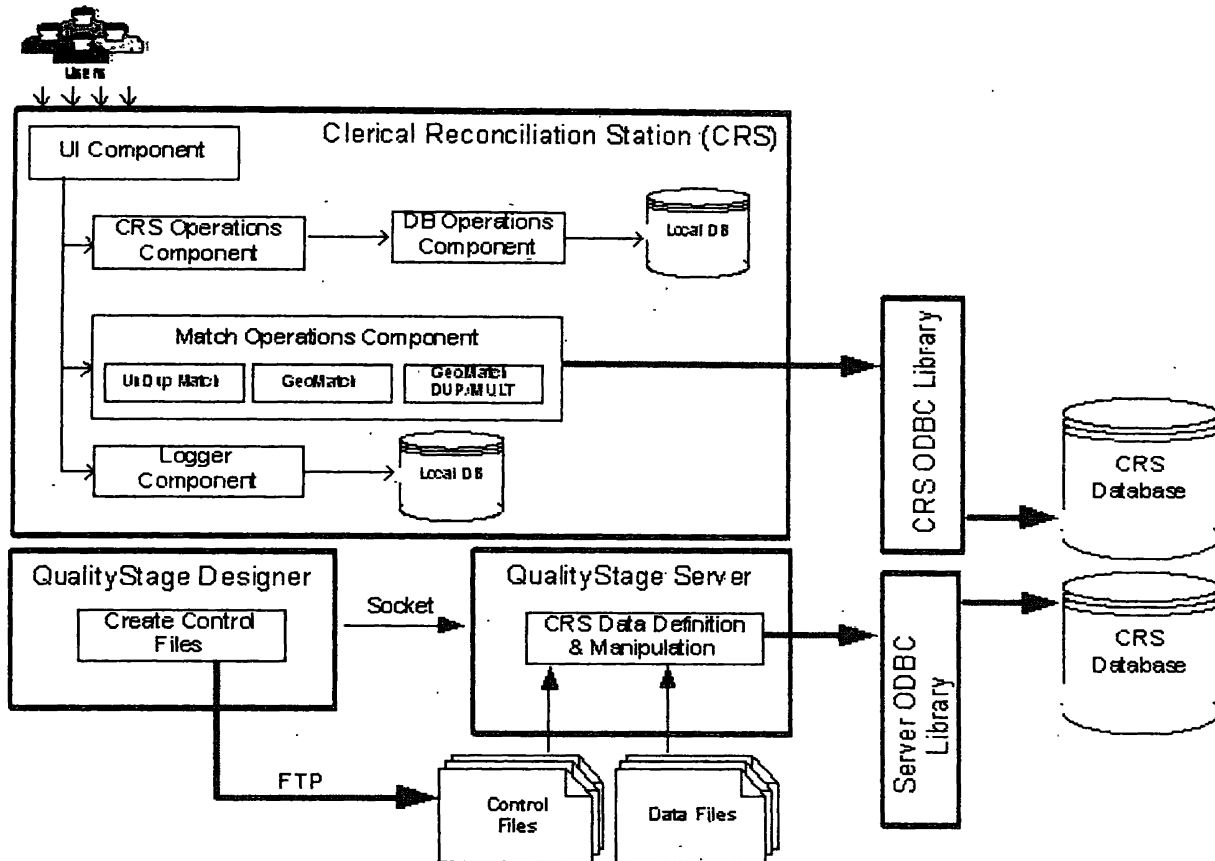


Figure 5.3 CRS Decomposition View

5.4.1.1 CRS Components

Several CRS Components are described under this section.

5.4.1.1.1 UI Component

This component handles code behind functionalities to the CRS UI screens. This includes display clerical sets information, filter clerical sets and review & reconciliation clerical data items.

5.4.1.1.2 CRS Operations Component

This component handles all the CRS application specific operations. This includes manage DNS information, maintain application settings, etc. CRS Operations component uses data in the local database and retrieves them via DB operation component.

5.4.1.1.2.1 DB Operations Component

This provides data manipulation functionality to the Clerical Reconciliation Station. DB Operations component will be provided direct db operations into the local database. ODBC library will be used to perform CRS database (external) operations.

The design of the CRS DBAccess Library will be described under the section 5.5 CRS DBAccess Library design, in this chapter.

5.4.1.1.3 Match Operation Component

This provides functionality to manage Clerical review and Reconciliation process. Class interfaces review and reconciliation methods/functions to layers above. UnDup, GeoMatch and GeoMatch MULT/DUP processing functionalities will be implemented with in these methods/functions.

Reviewed records will be marked in the database and removed form the next review screen. There will be new functionality implemented on the QualityStage server to extract reviewed match sets.

5.4.1.2 Class Diagrams

This describes the class structures, methods and properties

5.4.1.2.1 Review and Reconciliation Classes

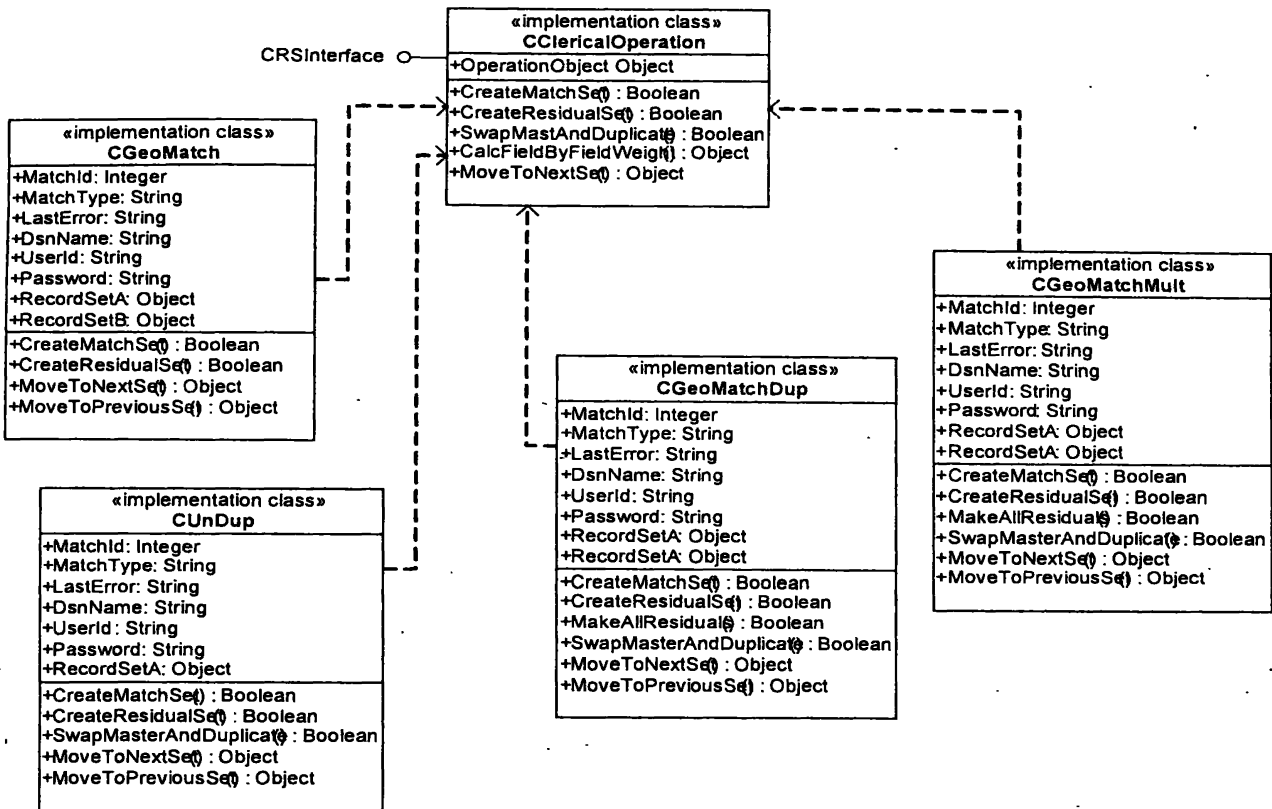


Figure 5.4 CRS Review and Reconciliation Class Diagram

5.4.1.2.2 CRS Operations Classes

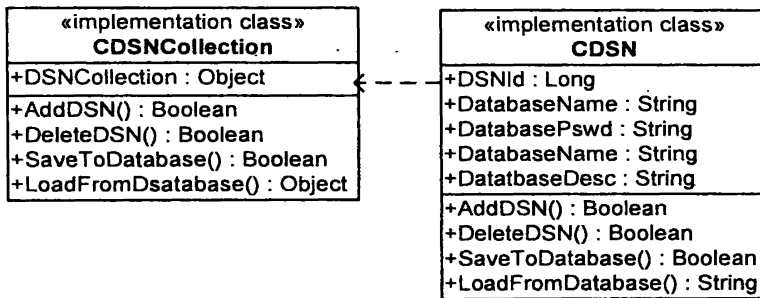


Figure 5.5 CRS Operation Classes

5.4.1.2.3 Logger Component Class

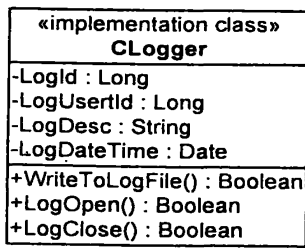


Figure 5.6 CRS Logger Component Class

5.4.2 Process View

This section describes the process flow of the system. (Clerical Reconciliation Station and QualityStage Designer's Match Specification area)

5.4.2.1 Review and Reconciliation Process

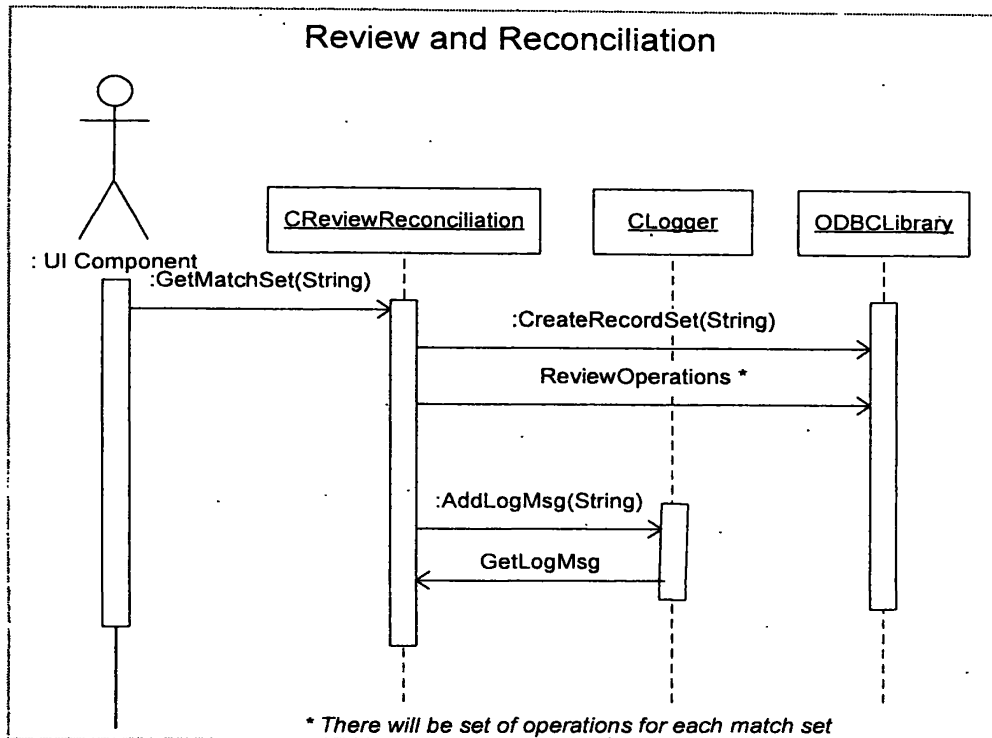


Figure 5.7 Sequence Diagram for GetMatchSet process

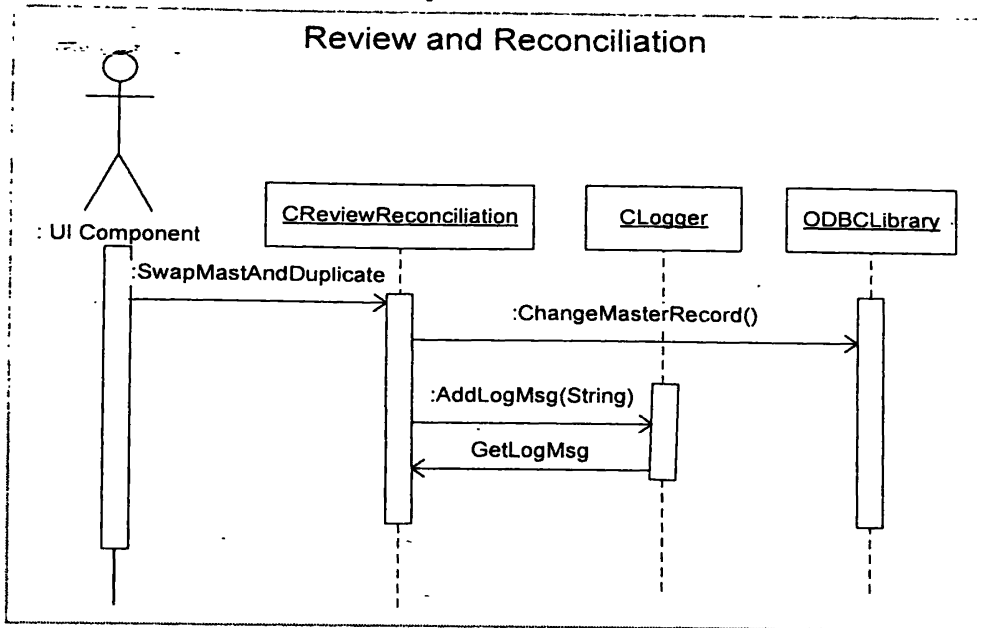


Figure 5.8 Sequence Diagram for SwapMasterAndDuplicate process

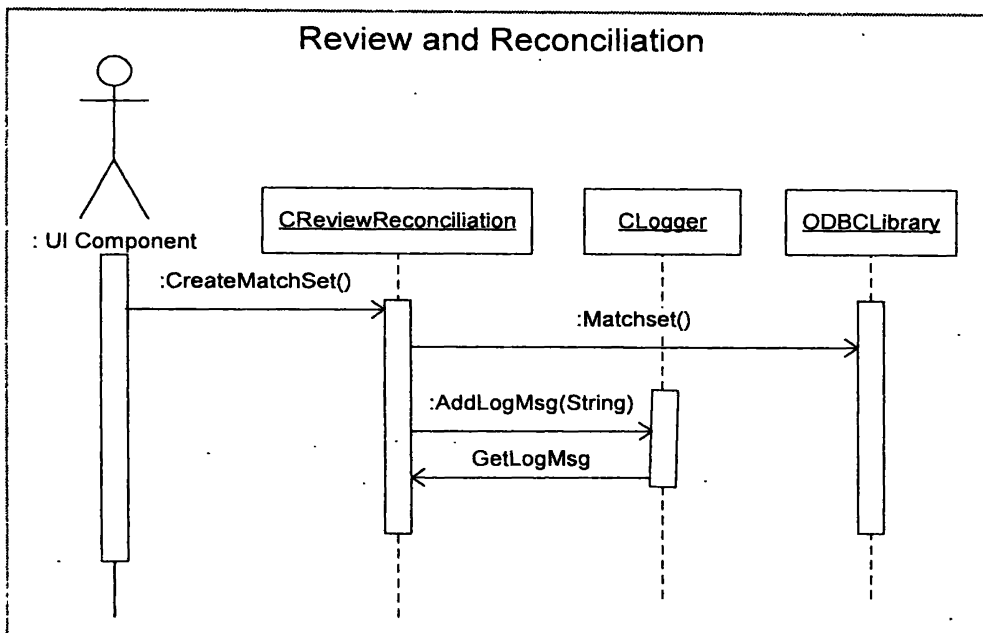


Figure 5.9 Sequence Diagram for Create a Match set process

5.5 CRS DBAccess Library design

5.5.1 Over view

The CRS DBAccess Library will handle all the database related functionalities of the CRS system. This will work as a separate VB COM object. Making it a separate COM object will enable the reusability of CRS DBAccess Library even when the CRS client is migrated to .NET. The CRS DBAccess Library will be compiled for Binary compatibility in order to maintain the class ID unchanged. This will ensure that only one version of the COM will be registered in the client machine.

5.5.2 Class Diagram

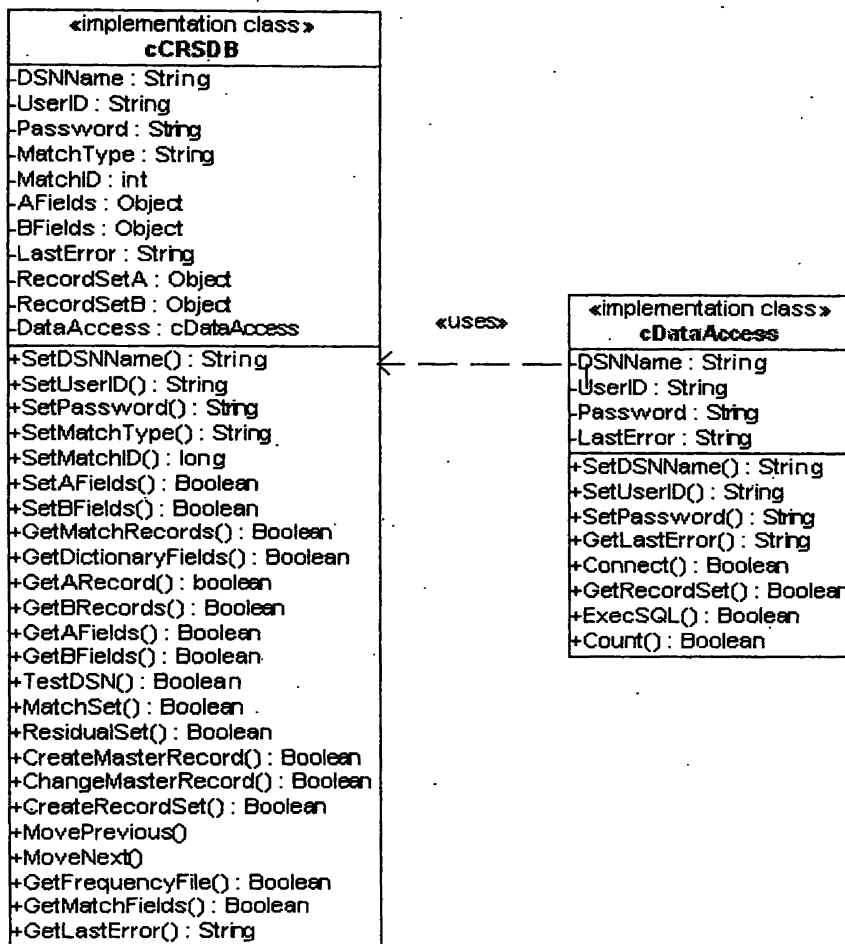


Figure 5.10 CRS Database Access Library Class Diagram

5.5.3 Functionality

The CRS DBAccess Library will use ODBC for database access. It will mainly support for Data Direct ODBC driver. All the public interfaces will be available in the cCRSDBAccess class. The cCRSDBAccess class will internally use cDataAccess class for database communication.

Database access will be achieved via a configured DSN in the user machine. DSN information should be provided to the DBAccess library. Initially user has to select a MATCH for processing. Once the MATCH is selected user will get one clerical set at a time for reviewing. Once the given record is processed, system will provide the next clerical set for reconciliation.

The CRS DBAccess Library supports concurrent users. This will ensure that no two users will edit the same clerical set at one time. User will get only the unprocessed clerical sets for reconciliation and the system will lock the master record before passing it for reconciliation. Moving to the next record or previous record without processing the current record will unlock the current record making it visible to other users for editing. If user processes the given master record and duplicates they will be marked as processed records. User will not get access to processed records.

Generally data files contain many data fields that user will not be interested for viewing during reconciliation process. The CRS ODBC Library running on the QS server will populate data relevant to all MATCH variable fields and only those fields that appear in the residual, master, and duplicate match extract specification.

The CRS DBAccess Library will send all the field data available and the CRS Client will display them according to the user requirement.

Following table describes the public interfaces provided by the CRS DBAccess Library and their expected functionality.

Interface	Functionality
SetDsnName	Set the name of the DSN that DBAccess library should use for Database connectivity
SetUserID	Provide the userID required for the DSN
SetPassword	Provide the password required for the DSN
SetMatchType	Set the MATCH type of the user selected MATCH.
SetMatchID	Set the match_id of the MATCH user selected
SetBFields	Provide the Field Names of the B file that displays in the GUI. If this is empty DBAccess library will populate only the MATCH fields in the B file.
GetMatchRecords	Will return all the MATCH records available in the CRS_MATCH_INFO table
GetDictionaryFields	Will return all the dictionary fields that user can use to filter clerical records available for a given MATCH
GetARecord	Return the master record data
GetBRecords	Return the duplicate record data available for the master record.
GetAFields	Return the field names of the A file
GetBFields	Return the field names of the B file
TestDSN	Will test the provided DSN for database connectivity
MatchSet	Make given A record and B records a MATCH SET.
ResidualSet	Make given A record (optional) and B records a residual set.
CreateMasterrecord	Set a given B record a master record
ChangeMasterRecord	Set the given master record as duplicate and the given duplicate record as master record
CreateRecordSet	Create a record set for the selected MATCH according to the provided filter criterion. If the record set creation is successful it will populate the first master record and its duplicate record data
MovePrevious	Get the previous unprocessed clerical set
MoveNext	Get the next unprocessed clerical set
GetFrequencyFile	Get the frequency file data of the selected MATCH
GetMatchFields	Get the MATCH field mapping

Table 5.1 public interfaces provided by the CRS DBAccess Library and their expected functionality

5.5.4 Supporting Concurrent Users

The Concurrent user access to the clerical sets will be provided using cursor level record locking. A cursor created for master records will be utilized for this process. The master record will be locked at cursor level before giving it to the user. Duplicate record locking is not required since the access to duplicate records is mapped via respective master record.

The cursor will be an ADO record set created using a KeySet cursor type and Pessimistic lock type. Pessimistic lock type was selected in order to have more control over record locking. Cursor location will be set to server in order to manipulate record locking at database level.

Advantages

- Lock will be rollback in unexpected system failures avoiding permanent locking of the record.
- Prevent concurrent editing of the same record.
- No physical lock will be applied to the database.

Disadvantages

- Entirely depend on the database and ODBC driver support provided by the vendor. Most of the standard ODBC drivers support but cannot guarantee for all.
- Some databases may not support for record level locking (MS Access support only page level locking).

The CRS client will support for SQL, Oracle, and DB2 databases. In order to make sure all these databases are supported for record level locking a survey was carried out and following table displays the results.

DataBase	Locking Support
SQL	Support for Record level locking (Version 7.0 upwards)
Oracle	DataDirect driver document confirm record level locking for Oracle 7.3 upwards
DB2	DataDirect driver document confirm record level locking for DB2

Table 5.2 Record Level Locking Support in each Database

5.5.5 Cursor creation and database update

Some of the queries was used for database updating in each screen are as given below. The table names will change according to the match_id of the match type selected.

5.5.5.1 Clerical Reconciliation – Select Record Screen

1. Get All Match Records to Display

Get Match Records

```
Select match_id, match_name, match_type, match_description, match_creation_date,
match_insert_mod_date, match_extract_mod_date, match_review_date, Project_Name
From CRS_MATCH_INFO
```

Get Total Records for the Match

UNDUP

```
Select count (*)
From CRS_MATCH_1
Where type='MP'
```

MATCH/GEOMATCH and GEOMATCH MULT/DUP

```
Select count (*)
From CRS_MATCH_A_2
```

Get Total Records Reviewed for the Match

UNDUP

```
Select count (*)
From CRS_MATCH_1
Where type='MP' and review_flag = 1
```

MATCH/GEOMATCH and GEOMATCH MULT/DUP

```
Select count (*)
From CRS_MATCH_A_2
Where review_flag = 1
```

2. Get Meta Data to Show all Dictionary fields

UNDUP

```
SELECT *  
From CRS_MATCH_1  
Where 1=0
```

MATCH/GEOMATCH and GEOMATCH MULT/DUP

```
SELECT *  
From CRS_MATCH_A_2  
Where 1=0
```

5.5.5.2 Un-Duplication Match Screen

- Create a cursor to get unprocessed master records and their duplicates

Get Master Records

Without Filters

```
Select *  
From CRS_MATCH_1  
Where review_flag = 0 and type='MP'  
Order by clerical_set_id
```

With Filters

```
Select *  
From CRS_MATCH_1  
Where review_flag = 0 and type='MP'  
and GIVEN FILTER CRITERION  
Order by clerical_set_id
```

Get Duplicate Records for a given master record ID

```
Select *  
From CRS_MATCH_1  
Where review_flag = 0 and type='CP'  
and clerical_set_id = GIVEN CLERICAL SET ID  
Order by record_id
```

- **Make a MATCH Set**

Use ADO Update method

(type = 'MP', review_flag = 1)

- **Make a Residual Set**

Use ADO Update method

(type = 'RA', review_flag = 1)

- **Swap the Master Record**

Use ADO Update method

(Existing Master record type = 'CP', New Master record type = 'MP', Update new match weights for duplicate fields)

- **Set a new record as the Master record**

Use ADO Update method

(New Master record type = 'MP', clerical_set_id = NEW CLERICAL SET ID, Update new match weights for duplicate fields)

5.6 CRS Database Design, Load and Extract

5.6.1 CRS Database Design

Clerical Reconciliation Station maintains a local Access database to store DSN & external CRS database (CRS_DATABASE) information and Application settings (CRS_SETTINGS).

Clerical Reconciliation Station maintains an external database. Stored within this database is database version information (CRS_INFO), match information

(CRS_MATCH_INFO), match run information (CRS_MATCH_RUN_IDS), clerical record information (CRS_MATCH_A_<Match Identifier> and CRS_MATCH_B_<Match Identifier>), match frequency information (CRS_FREQ_<Match Identifier>), and clerical set identifiers (CRS_CLERICAL_SET_IDS).

Local Database:

- CRS_DATABASE: Information on external clerical (CRS) databases which are connected to the CRS application.
- CRS_SETTINGS: Information specific to users and match sets. (e.g.: Data Grid display specifications, Show/hide columns, Size columns and Order Columns etc.)

External Database:

- CRS_INFO: database version information related to each match
- CRS_MATCH_INFO: match information including match name, description, creation date, etc
- CRS_CLERICAL_SET_IDS: assigned set identified for each match run that uniquely and sequentially distinguishes match sets
- CRS_MATCH_A_<Match Identifier>: clerical record information including match variables and data fields for the A file of a two file match or the only file in an Undup
- CRS_MATCH_B_<Match Identifier>: clerical record information including match variables and data fields for the B file of a two file match
- CRS_FREQ_<Match Identifier>: match frequency information acquired from the .FRQ files
- CRS_DIC_<Match Identifier>: match frequency information acquired from the .DIC files
- CRS_SPEC_<Match Identifier>: match specification information acquired from the .MAT file
- CRS_EXTRACT_<Match Identifier>: match extract information acquired from the .EXT file.

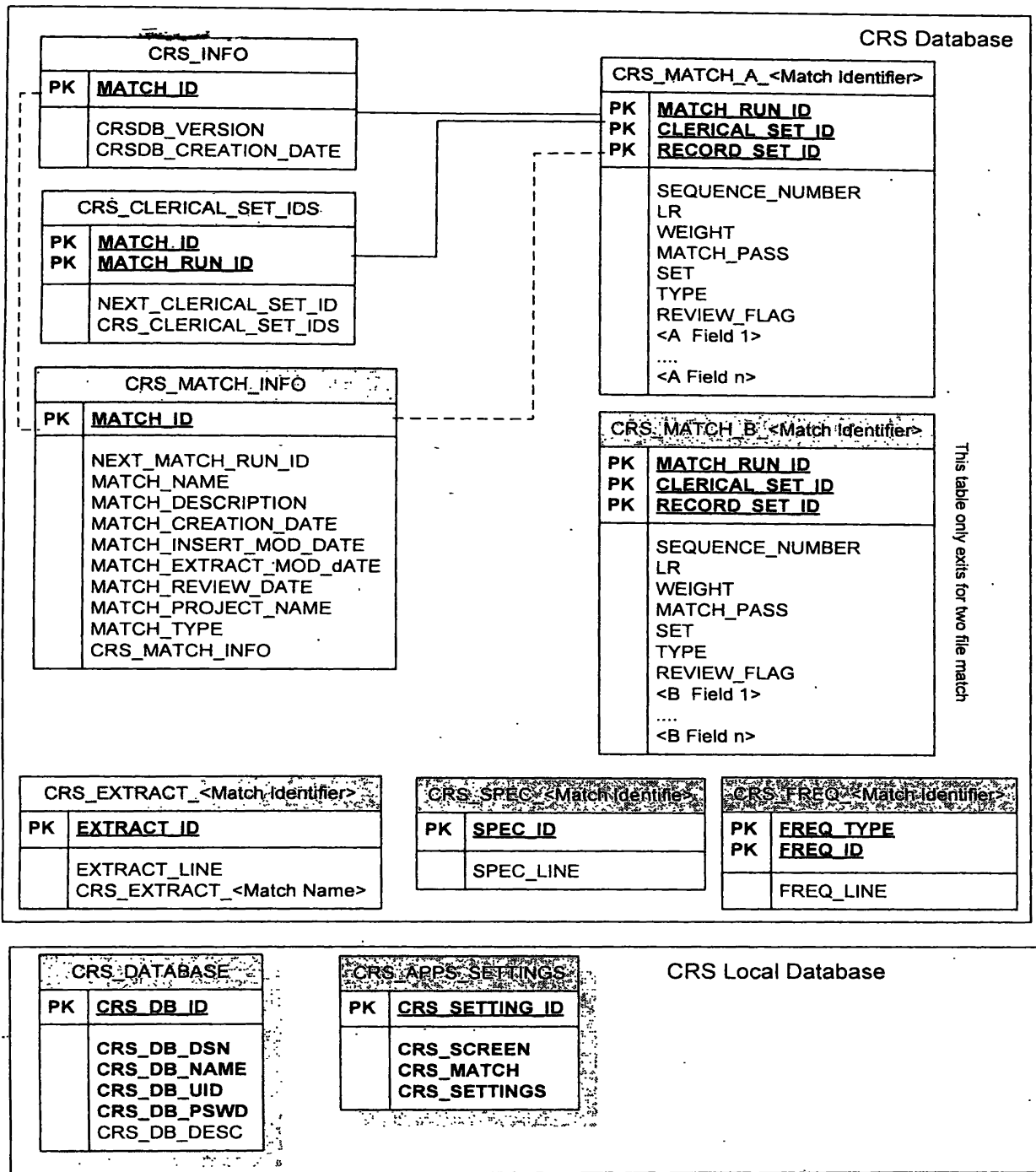


Figure 5.11 Entity Relationship Diagram for CRS Database Access Library

5.6.2 CRS Database Load

5.6.2.1 QS Server Extract Modifications Preparing for Database Load

At the beginning, the QS Server match extract process did not include the duplicates of clerical pairs/sets in the clerical extract file or data stream. These duplicates, both those from match sets and clerical sets, were directed to any file or data stream the user chooses. For integration with CRS, the extract process will be modified to include the duplicates for clerical sets in the clerical extract file or data stream instead of (or possible in addition to) the user specified duplicate record file or data stream.

5.6.2.2 Database Load Platform Support

CR database load, extraction, and use from the CRS platform is supported on AIX, Solaris, Linux, HP-UX, Tru64 and Windows using the IBM IIS DataDirect ODBC drivers for ODBC support.

On the OS-390, CR load and extraction is currently being investigated. The CRS use of the database will not be supported by the IBM IIS DataDirect ODBC drivers, but either DB2 II Classic Federation or IBM IIS Connect (iWay) could be used. If the CR database load and extraction are not feasible, the options are to say CRS is not available on the OS-390 or to provide users with detailed instructions on how to load and extract clerical records from the CR database.

5.6.2.3 QS Server Database Load Overview

Following the QS Server match extract process, another user scheduled stage will direct the output stream of a match extract process destined for clerical review into the CRS. This stage, Clerical Reconciliation Insert (CRI), will perform database creation and the insertion of clerical records. As a separate stage not directly incorporated into the match, users can replace this ODBC stage with a more performance DB-specific implementation.

5.6.2.4 QS Server Creates CRS Database

If a user wants to direct the clerical output from a match into the CR external database for reconciling in the CRS, the project's Controls directory will contain a file named <Stage Name>.CRI. From this file the QS Server will gather the DSN information, and connect to the database during CRI processing.

The QS Server determines if the CR database has been initialized. This determination is made based upon the existence of non-match run specific tables exist in the CR database. These tables include CRS_INFO, CRS_CLERICAL_SET_IDS, CRS and CRS_MATCH_INFO. If these tables do not exist, the QS Server creates these tables in the CRS database.

5.6.2.5 QS Server Populates CR Database with Clerical Data

The CR database population occurs following the match extraction process in a match run during the CRI stage. At this time all the control files were uploaded to the QS project directory and a successful match run generated clerical data records.

The QS Server process clerical data streams and parse them with the use of data dictionary [.DIC] files. If the QS Server CRI process determines that the CRS database exists, the database is loaded with the clerical records created during the match.

The QS Server CRI process determines if a previous run of this match has occurred by checking for the existence of the match name and match project name in the CRS_MATCH_INFO table. If the match does not exist, the QS Server acquires an auto-generated match identifier, inserts the match details into the CRS_MATCH_INFO table, and initializes the next_match_run_id in CRS_MATCH_INFO table to zero.

Once a match identifier has been determined, the value of the next_match_run_id in the CRS_MATCH_INFO is selected. If this value is not zero, the CRI process compares the all the fields in the clerical match extract specification (this includes all match variables and input fields) and the columns in the CRS_MATCH_(A/B)_<Match Identifier> table. These must be identical or an error message is issued.

The option to compare the db_version in the CRS_INFO associated with the match_id also exists. An error message is issued if the QS server's current notion of the CRS database version differs from the match identifier's db_version within the existing database.

Then the next_match_run_id in CRS_MATCH_INFO table is incremented. The next_clerical_set_id in the CRS_CLERICAL_SET_IDS is initialized to zero for this match run.

The QS Server's CRI process populates the CRS_MATCH_(A/B)_<Match Identifier> table while assigning a sequential clerical set identifier to each match set of clerical records. The clerical_set_id differs from the @SET in the match extract specification because it is a sequential count that can be easily tracked and used by the CRS to break sets of records into distinct subsets with their own unique set identifier.

Before the clerical records for that match are loaded into the CRS database, the next_clerical_set_id in the CRS_CLERICAL_SET_ID table is incremented by the maximum number of records that will be allowed in a bulk database load. The clerical records are then loaded into the CRS_MATCH_(A/B)_< Match Identifier >. After all loads the current largest clerical set identifier is stored in the next_clerical_set_id in the CRS_CLERICAL_SET_ID. This could result in a gap in the clerical set identifier that is equivalent to the maximum number of records in a bulk load minus one. The next clerical set identifier is now available for use by the CRS to create subsets from larger sets that with a distinct set identifier for the match run.

The CRS_FREQ_<Match Identifier> table is loaded with the information contained in the .FRQ files for each of the match input files.

The CRS_DIC_<Match Identifier> table is loaded with the information contained in the .DIC files for each of the match input files.

The CRS_SPEC_<Match Identifier> table is loaded with the information contained in the .MAT file for the match.

The CRS_EXTRACT_<Match Identifier> table is loaded with the information contained in the .EXT file for the match.

5.6.3 CRS Database Extraction

To extract the reviewed former clerical records from the CRS database and produce a flat file or data stream that's for use in downstream processing, a new stage will be added to the QS Server and Designer. The Clerical Reconciliation Extract (CRE) stage allows users to extract the match specific reviewed records from the CRS database. The extraction of all the information from the CRS_MATCH_(A/B)_<Match Identifier> table includes all match variables and all data file fields.

As the CRS database schema allows for inclusion of multiple runs a single match job, the extraction process generates a unique set identifier that is composed of the match run identifier pre-pended to the clerical set identifier. This identifier will be used in place of the current ten characters set identifier generated by the match process.

When extracting reviewed records from the CRS database, the QS Designer user will have the same filter options available in the CRS. These options will be filter records based on match type (masters, duplicates, and residuals), processing dates, etc.

5.6.3.1 Clerical Reconciliation Extract Stage Control File

A control file for use in the extract will specify the DSN specific information as well as the extract layout. The options for the extract layout will be to

An example file would be name MATCH.CRE and contain the following information:

```
QualityStage v7.0
DSNNAME CRS_DB
VSERIAL (an encrypted version of the username and password)
FILTER WHERE TYPE=MP
MATCHNAME UTEST
PROJECTNAME SAMPLE
```

5.6.4 Clerical Set Identifier

The match set identifier that is assigned by the QS Server is problematic in the context of the CRS. The match set identifier is replaced with a clerical set identifier when data is introduced into the Clerical Reconciliation database.

The issue :

The match set identifier assigned by the QS Server is a means to allow for grouping or clustering of the related match masters, duplicates and clericals. The match set identifier is unique only within the context of a match run. A user may run multiple iterations of the same match and direct all of the resulting clericals into the CRS. This could result in unrelated sets with the same match set identifier. As the CRS GUI will be using a set id to group records together to display to the user, this would group unrelated set of records from different iteration of the match together. This grouping would be incorrect.

The solution:

A clerical set identifier would supplement the match set identifier when records are inserted into the CRS database. This clerical set identifier together with the match run identifier would be unique identifier within the context of the match. This would allow for correct grouping in the CRS GUI display and in the file resulting from the CRS database extraction.

Any user who uses the match set identifier in a meaningful way in downstream processes would be adversely impacted. It is unlikely that the batch (file, stream, or PX mode) user is using this match set identifier in a meaningful way. The real-time user may be doing so, but this issue could be resolved with a set of "Best Practices" instructions. The match set identifier in this case would need to have been acquired from another place in the record, so the user should use that field instead of using the match set identifier. For some time, users have been warned about the transient nature of the match set identifier, so ideally no one should be treating it as a meaningful value.

The proposed clerical set identifier and match run identifier would have a length identical to the match set identifier. Records extracted from the CRS external database would then have a length and format identical to those in the extract files produced during the match extract, so they could participate in any downstream processes without modification.

5.7 CRS Weight Calculations

5.7.1 Re-Packaged QS Server Match Weigh Calculation Utility

The Clerical Reconciliation Station will need to provide dynamic match weight calculations to users in two separate scenarios. The first use is to provide weights on a field by field basis for an existing pair of clerical records. This will provide users with more metadata that may assist in their decision making. In the second scenario, when the user switches a master record with a duplicate in the Clerical Reconciliation Station, the weight for the duplicate records must be recalculated. The CRS will perform both of these functions using a re-package version of existing QS Server functionality.

5.7.1.1 QS Server Weight Calculation

The QS Server match process calculates the field and composite weights assigned to each clerical record in a process that has been repackaged into a library (.dll) that ships with the CRS.

This library, crs.dll, is a repackaged version of the functionality currently found in the QS server match processes.

This library does not require the presence of the QS server .integconfig license file. It will require the ICU 2.4 libraries that currently ship with the QS Designer and Server.

5.7.1.2 CRS use of crs.dll

The Clerical Reconciliation Station will reconstruct the frequency and control files for the match using the information stored in the CRS_FREQ_<Match Identifier>.

CRS_SPEC_<Match Identifier>, and CRS_DIC_<Match Identifier> tables in the CR database.

Input :

- Frequency files – A string specifying the fully qualified name of the A and B frequency files, null shall be provided for the B frequency file in an Undup run.
- Match specification – A string specifying the fully qualified name of the match specification file
- Dictionary files – A string specifying the fully qualified name of the A and B dictionary files, null shall be provided for the B dictionary file in an undup run
- A/B records – A string containing the A and B record, or in the case of an undup the A and A record, shall be passed to the library.

Output:

- CRS Weight Structure – A structure that specifies the composite weight for the record, the number of match variables, and an array of match A variable name and the corresponding weight.
- Error Code – A string specifying success or failure

The VB declaration of the function:

```
Declare Sub crs_calc_weight Lib "crs.dll" (ByVal freqA As String, ByVal freqB As String, ByVal mtchSpec As String,
ByVal dicA As String, ByVal dicB As String, ByVal recA As String, ByVal recB As String, CrsMtchWgt)
```

The C functions:

```
char * __far __pascal __export crs_calc_weight(char __far * freqA, char __far * freqB, char __far * mtchSpec, char
__far * dicA, char __far * dicB, char __far * recA, char __far * recB, struct CrsMtchWgt);
```

Where

```
struct CrsVarWgt{
    UCHAR *varname[];
    float wgt;
}
struct CrsMtchWgt{
    float compWgt;
    int nMtchVars;
    struct CrsVarWgt varwgt[];
}
```

5.7.1.3 Updating Composite Weights in the CR Database

The composite weights shall be updated in the CR database by the CRS as a consequence of the following actions.

- When a master and a duplicate record are swapped in an Undup match, where swapping is defined as making the chosen duplicate record the master record for the set and making the existing master record a duplicate.
- When a clerical set is broken into multiple match sets in an Undup match, the weights for the duplicates of the newly created master record must be recalculated and stored in the CR database.
- When a master and duplicate record in the B file are swapped in a GeoMatch Multiple/Duplicate match.

5.8 Quality Stage Designer Changes

In order to support Clerical Reconciliation Process, New functionality has introduced to the Quality Stage designer. The expected changes are as follows.

- Provide a GUI for Users to input CRS DSN information for the selected MATCH.
- Create “.CRS” control file using the user selected DSN information during the deploying stage.
- Add text to “.ENV” file during the run stage to inform Quality stage Server to process clerical reconciliation or not.

5.8.1 Process

The MATCH wizard functionality will be changed to allow users to define CR DSN information for the MATCH. The idea is to allow users to direct clericals created by different MATCH stages to different databases in situations where more than one MATCH stage is associated with a RUN. Also this will enable users to ON and OFF clerical reconciliation process for the selected MATCH.

The DSN ID and the status of the Clerical Extraction (ON or OFF) will be stored with the MATCH specific information in the suprop table of the local database. Following fields

that are not currently in use have been selected to store these values in order to minimize database changes and CMTCOp class structure.

investigation_freq_cutoff (long integer) - to store the CRS odbc DSN

idinvestigation_options (integer) - to save the clerical review ON, OFF state

During the Deployment process System will check MATCH stages associated with the RUN for clerical extraction information and create a “.CRI” control file for each one of them. For MATCH stages that do not have a Clerical Reconciliation assigned, system will create a “.CRI” control file with empty values for variables

The RUN wizard will have a check box for user to activate clerical extraction. This check box will work as a central switch to all MATCH stages associated with the RUN. When the clerical extraction is activated, Quality Stage will process clerical extraction for all the MATCH stages which has a clerical reconciliation process defined. It is mandatory for users to deploy the MATCH before running it, if they do any changes to the Clerical Extraction process associated with the MATCH.

During the RUN process System will check for the “Disable Clerical review Processing” check box value. If it is OFF system will set DOCRSEXTRACT environment variable value in the “.ENV” file to YES. The QS server will check for the “DOCRSEXTRACT” variable value in the “.ENV” file and if it is set to YES, system will process Clerical Extraction for all available “.CRI” control files with DSN information. The “.CRI” files with empty variable values will be ignored.

Chapter 6: Development Environment

A brief introduction of the development environment is given below.

IDE	Microsoft Developer Studio 6 (Visual Studio 6)
Languages	Visual Basic 6 – Service Pack 6
Operating system	Windows XP Professional
Third Party Components and Tools	<ul style="list-style-type: none">• Infragistics Active Tool Bars Plus• Xceed SmartUI v2.0• Sheridan Components
Database	Access – local database Oracle/DB2/SQL Server– CRS database

Table 6.1 CRS Development Environment

Chapter 7: Deployment Environment

A brief introduction of the deployment environment is given below.

Operating System for CRS Client	Windows NT/2000/XP
Packaging Tool	InstallShield 6.3 Professional
Third Party Components and Tools (These tools packed into installer package)	<ul style="list-style-type: none">• Infragistics Active Tool Bars Plus• Xceed SmartUI v2.0• Sheridan Components

Table 7.1 CRS Deployment Environment

Chapter 8: Conclusion

The aim of this project was to enhance data cleansing features in the IBM IIS QualityStage according to the Object Oriented Technology. By using this architecture, reusability and extensibility issues could be achieved to evolve the system. Changes in business logic could be made by modifying the existing components or by adding new components to the system. User interfaces could adapt to the changes by modifying the program code. Although time spent on designing and implementing components was longer than traditional design and implementation, the time spent on future changes would be saved.

8.1 Summary of Objectives, Achievements, Problems and Challenges

The objectives of the project were:

1. To capture documents and prioritize requirements for the Clerical Reconciliation Station.
2. To learn and apply the technology of VB6 Enterprise Edition
3. To apply an iterative approach to increase user involvement.
4. To design the Clerical Reconciliation Station while paying specific attention to usability and maintainability.
5. To implement and test the system.
6. To evaluate the system with some user trials.

Objective 1, 2, 4, 5 were successfully achieved while the other one partially completed in this project. Capturing requirements were achieved and presented in Use Case Diagram in Chapter 2. The design was described in Chapter 3, 4 and 5. However, prototyping approach was not applied enough due to the project currently goes under its maintenance part and including the implementation was not allowed due to the customer restrictions. Evaluations are carried out continuously with its potential users according to the iterative development approach. The others were artifacts in evaluation that were produced in each phase.

The major challenges of this project were:

1. To learn how to use VB6, this is required to develop the system.
2. Designing and developing the potential user's requirements.
3. How to enable VB6 technology to enhance the system.
4. How to reuse the components in the right way.

8.2 Future Consideration

Future consideration for this project is to enhance some of the present features and add new features for the system.

The clients' comments on the prototype evaluation must be first fulfilled. The usability is the main issue on how comfortable potential users are of using this system. For example, if they find that it is difficult to communicate with the system, no matter how good the functionalities are, the system should not go live. Therefore user interface based on client evaluation should be modified to suit users' requirements.

Secondly, extra features should be added to the system so that it can meet the client's requirements.

References

1. Eliens, A. (2000) Principle of Object-Oriented Software Development, 2nd Edition, Pearson Education Limited 2000.
2. Jacobson et al (1999), The Unified Software Development Process, Addison Wesley Longman, Inc.
3. Shackel, B.(1991), Usability-context, framework, definition, design and evaluation. In B. Shackel and S. Richardson (eds) Human factors for information usability, Cambridge, Cambridge University Press, 21-37.
4. MSDN Home and OR Home Page, URL: <http://msdn.microsoft.com>
5. Subversion Project and OR Home Page, URL: <http://www.tigris.org> [25th August 2003]
6. Szyperski C., (1997), Component Software: Beyond Object-Oriented Programming, Addison-Wesley Longman.
7. The Rational Software Corporation and OR Home Page, URL: <http://www.rational.com> [20th August 2003]
8. Pearson Home and or Home Page, URL: <http://www.cpearson.com>
9. Quantumstate Home and or Home Page, URL: <http://www.tutorialized.com>
10. VB Helper Home or Home Page, URL: <http://vb-helper.com>
11. Abstractvb Home or Home Page, URL: <http://abstractvb.com>
12. Matthew J. Curland. Advanced Visual Basic 6, Pearson Education Limited 2000
13. Steve Brown. Visual Basic 6 Complete, SYBEX Inc 1999

Appendix A Reflection

At the beginning of the project, I was confused by using VB6 technology in an industrial standard manner to enhance the IBM Data Integration Suit. During the ramping up process, I learnt a lot on what technologies was and how to apply it. However, it was little easy to capture most of the things by reviewing systems which are already developed under IBM account.

Due to the project was going under its maintenance part, I had to familiar with technologies which are already used in the project and had to enhance the system keeping its usability, maintainability, flexibility, efficiency, etc by using those technologies.

During the implementation, Object Oriented Technology helped to minimize the extent of bugs, debugging process could be much more efficient then traditional architecture. The debugging process was frustrated and the time spent on it was out of my expectation. But I still learn a lot in this project.

Lastly, I would like to share my experience to junior programmers who works with Object Oriented Technology.

- Remember to get concrete understanding the Object Oriented Technology and its features before designing and development
- Develop component one by one according to the design.
- Test each component before integration.
- Get use of components to minimize redundant

National Digitization Project
National Science Foundation

Institute : Sabaragamuwa University of Sri Lanka

1. Place of Scanning : Sabaragamuwa University of Sri Lanka, Belihuloya

2. Date Scanned : 2017-09-25

3. Name of Digitizing Company : Sanje (Private) Ltd, No 435/16, Kottawa Rd,
Hokandara North, Arangala, Hokandara

4. Scanning Officer

Name : B.A.C. Badarwan

Signature : 

Certification of Scanning

I hereby certify that the scanning of this document was carried out under my supervision, according to the norms and standards of digital scanning accurately, also keeping with the originality of the original document to be accepted in a court of law.

Certifying Officer

Designation : Librarian

Name : T. N. Neighsoorei

Signature : 

Date : 2017-09-25

Mrs. T. N. NEIGHSOOREI
(MSSc, PhD, ASLA, BA)
Librarian
Sabaragamuwa University of Sri Lanka
P.O. Box 02 Belihuloya, Sri Lanka
Tele: 094 45 2280045
Fax: 094 45 2280045

"This document/publication was digitized under National Digitization Project of the National Science Foundation, Sri Lanka"