University of Waterloo

Faculty of Engineering

Nanotechnology Engineering

Optimizing Use of Relational Database Process Automation System for use in Quantitative Data Analysis

Curtiss-Wright Defense Solutions

333 Palladium Drive

Kanata Ontario, K2V 1C1

Prepared by

Leander Rodrigues

ID: 20670272

User ID: lgrodrig

2A Nanotechnology Engineering

April 30, 2018

Leander Rodrigues

3187 Malham Gate

Mississauga, Ontario

L5M 6K9

April 22, 2018

Siva Sivoththaman

Nanotechnology Engineering

University of Waterloo

Waterloo, Ontario

N2L 3G1

Dear Sir,

This report entitled “Optimizing Use of Relational Database Process Automation System for use in Quantitative Data Analysis” is my 2A Work Term Report which was prepared to fulfill the course requirements of NE 250. The purpose of this report is to evaluate the complex system architecture responsible for the process automation and determine its strengths and shortcomings with respect to general practice automation techniques.

During my 2A work term, I worked closely with Mr. Steven Dick (my directing supervisor) and the rest of the Quality Engineering department at Curtiss Wright Defense Solutions. Our responsibility as a team was to ensure the product (printed circuit boards) were adequately assembled, tested, recorded and inspected before being sold and shipped to the customer.

Mr. Dick and I were responsible for the process automation in this department, so I would like to thank him for the support and guidance he provided. His assistance aided in getting me through the steep learning curve. He gave me all of the available documentation, developer comments and necessary insights for the creation of this report, along with answering all of my many questions.

In addition to Mr. Dick, I would also like to thank my brother, Linford Rodrigues, a fifth-year electrical engineering student from McMaster for taking the time to proofread my report. I hereby confirm that I have received no further help other than that mentioned above in writing this report. I also confirm that this report has not previously been submitted at this or at any other academic institution.

Sincerely,



Leander Rodrigues

ID: 20670272

# Contributions

During my time with Curtiss Wright Defense Solutions, I was working closely with the Quality Engineering Department team. It consisted of six technical engineers, one department manager (a supervising engineer) and myself as the only coop student. Of that only my acting supervisor Mr. Steven Dick (a technical engineer) and I were responsible for contributing to the project outlined in this report. Mr. Dick would communicate with other departments and supervisors to determine the technical data that would like to have in the form of reports, and It was my responsibility to provide the information while maintaining/debugging the existing system architecture.

The goal of this process automation project was to do away with the necessity of repetitive manual tasks which would hinder the effectiveness of the Quality Engineering (QE) department. The QE department has a lot of responsibilities, the most prominent of which is direct-to-customer communication. Any issues they have pertaining to the production cycle, quality testing, or even payment/warranty/return processes are tasked to the QE department. The implementation of an automated report process would remove the inconsistencies of manual reporting, in addition to giving time back to the other busy engineers.

My specific responsibilities varied on a day to day, but I largely dealt with the programming side to the reporting. I was the only technical consultant for the system architecture, since Mr. Dick had other responsibilities. I was assigned with reading through the existing codebase to remove redundancies, debugging any errors/alerts mentioned by the rest of the department, and implementing additional features to the report, like filters/buttons/tables. Any time remaining was spent documenting the system architecture in a clear way to facilitate understanding for future coop students or other engineers.

There is a concrete relationship between this report and my obligations at Curtiss-Wright Defense Solutions. This report contains a critical analysis and evaluation of the existing process automation system, which I worked with daily during the work term. The insights, successes and shortcomings discovered in the current system have been found through my task of thorough documentation, whilst recommendations for future improvement are rooted in research and best practice. In writing this report, I have been able to enhance my critical thinking abilities and evidence based evaluations by way of my conclusions (section 5.0) and recommendations (section 6.0). This report will provide a permanent record of the knowledge acquired through this work term experience, as well as proof of the logical reasoning skills developed through the Nanotechnology Engineering program.

The contributions I have made to the present automation system will persist throughout the work terms of future coop students until the process automation falls to disuse or is rebuilt. My implementations are mainly in the form of additional documentation, a few report redesigns, and many altered/inserted code snippets. Using the system architecture which I have helped to maintain, the other quality engineers will continue to be able to interact with large amounts of quantitative data with ease, and focus on the more pressing tasks in their daily work. Additionally, future coop students will be able to read through my provided documentation and improve upon the automation system by way of newer reports and features.

# Summary

As a sophisticated computing product manufacturer, Curtiss-Wright Defense Solutions generates immense amounts of production, financial and processing data. The team of engineers and supervisors in the Quality Engineering department have the responsibility to communicate the effects, changes or consequences of this data directly to high volume customers, which requires a certain level of attention. Data extraction and analysis is time consuming for the department, so a process automation system is necessary to allow the department to maximize their productivity instead of routinely performing menial tasks. The process automation does away with much of this work; however, a critical workflow such as this, must be routinely evaluated to identify performance issues and design flaws, with the overall goal of optimizing the program.

The system architecture and its maintainability are some of the key topics that will be discussed in this report. The system architecture is discussed by way of applications/languages involved in the automation, along with their benefits and drawbacks. The process automation maintainability is evaluated by considering areas such as documentation, edge cases, and dependencies which could hinder the system in future development, even if sufficient at the time.

Through the evaluation discussed in this report, the process automation was found to be deficient in significant areas in terms of long-term maintainability, namely documentation, and dependencies. In order for the system to proceed in report generation, automated scripts pulled data from spreadsheets/working files from reports created manually by other departments. This, along with other potential flaws were only discovered through manual code tracking, which is evidence of weak original documentation (written by the initial programmer). This however, was countered by the additional documentation provided by past coop students which helpfully, identified each report’s data source and process timeline. Therefore, another conclusion was drawn that while initially the system had not been documented sufficiently, past coops have noted this, and supplemented explanatory documents and inline comments.

This report recommends that preventative actions be taken to avoid the possibility of an automation system failure. These actions could be, for example, employing a version control software to track changes and edits to scripts and macros, such as Git or Mercurial. Along with these tools, a larger technical team is proposed to be better capable of redesigning the existing complex system, perhaps by removing/rerouting dependencies. Lastly, those team members who depend on the automation (namely the quality engineers), are suggested to undergo basic scripting training to allow for basic debugging and hotfixes in critical situations.

# Table of Contents

[Contributions iii](#_Toc512502707)

[Summary v](#_Toc512502708)

[Table of Contents vi](#_Toc512502709)

[List of Figures viii](#_Toc512502710)

[List of Tables ix](#_Toc512502711)

[1.0 Introduction 1](#_Toc512502712)

[1.1 Software Automation 1](#_Toc512502713)

[1.2 The ‘Patchwork’ System Design 2](#_Toc512502714)

[2.0 Background 3](#_Toc512502715)

[2.1 Command Execution and Scope 3](#_Toc512502716)

[2.2 Relational Databases 3](#_Toc512502717)

[2.3 Constraints and Design Requirements 4](#_Toc512502718)

[3.0 Enhancing System Architecture 6](#_Toc512502719)

[3.1 Existing Automation Model 6](#_Toc512502720)

[3.1.1 Data Extraction 6](#_Toc512502721)

[3.1.2 Data Manipulation 7](#_Toc512502722)

[3.2 Alternatives Automation Models 8](#_Toc512502723)

[4.0 Improving System Maintainability 9](#_Toc512502724)

[4.1 Languages 9](#_Toc512502725)

[4.2 Documentation 9](#_Toc512502726)

[4.3 Dependencies 9](#_Toc512502727)

[4.4 Longterm Sustainability 10](#_Toc512502728)

[5.0 Conclusions 12](#_Toc512502729)

[6.0 Recommendations 13](#_Toc512502730)

[Glossary 14](#_Toc512502731)

[Appendix A – SQL code for use in Data Manipulation 15](#_Toc512502732)

# List of Figures

Asefase asetase

Asetsaet

# List of Tables

Saijfa seijfsa e

Asiejfa paoske

# Introduction

Every instance an individual repeats a task that they’ve completed countless times before, there exists an opportunity for it to be automated. In some fashion, the task can be reduced to a set of procedural instructions, which, if completed correctly, will achieve the end goal. By passing these instructions to a sophisticated program, or using the appropriate tool, the end goal can be reached faster with the same, if not more, precision. Specifically in the technology sector, this trend towards automation is what creates a fast-paced innovative environment. The power of modern computing allows for process automation to take place orders of magnitude faster than any individual could complete; however, inattention something

Curtiss-Wright Defense Solutions (CWDS) is a large manufacturer of sophisticated printed circuit boards, designed specifically for use in countless military sectors. The Quality Engineering (QE) department at CWDS is responsible for the interpreting the immense quantity of data that comes out of this process, and communicating it directly to the customer. Currently, the QE department mainly automates the data extraction portion of their workflow, leaving the interpretation and manipulation portion for each individual to tackle manually. Additionally, the system is based on an outdated codebase with little to no maintenance, which, in the case of a critical failure, could result in a complete halt in the department’s productivity. Therefore, there exists a need to evaluate the current automation process, and optimize it to more effectively suit the team’s needs.

## 1.1 Software Automation

Automation is a broad term for describing “the methods of controlling industrial processes often automatically, often by way of controlled systems” [http://www.dictionary.com/browse/automation], but this largely has colloquial connections to robotics, especially in the manufacturing sector. Software automation is a specific subset of automated processes referring only to the computation and logic operations, but may still rely on data extracted by way of robotic automation (see Figure 1). By passing the computer a set of sequential operations, once completed, provide some desired output to the user, about the information acquired from the other automated systems in the database (for example, financial records, machine operating times, component efficacy). In the case of this report, only software automation will be discussed as it was integral automation process being analyzed for possible improvement.

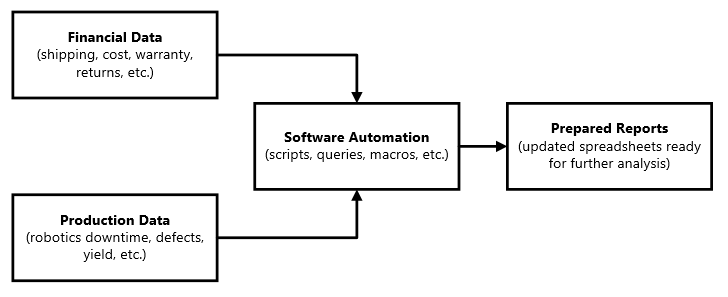


Figure 1: Simple software automation structure

## 1.2 The ‘Patchwork’ System Design

The process automation structure evaluated in this report is described as a `patchwork` as it has been assembled from multiple different individual developers, the majority of which being former coop students. The QE department started the automation project in summer of 2015, where a contracted developer was brought on for just over a year. Since their departure, coop students have rotated in an out of his position, contributing to the automation system with their own additions/edits. This has led to inconsistent documentation, little to no supervision and a lack of maintenance due to the difficulty of completing assigned projects while learning of every prior addition to the system. Recommended solutions to these persistent issues are discussed in section 6.0.

# Background

Automating software process depends on a variety of different applications and software. In order to begin to evaluate the efficacy or suggest improvements for a future system, it is integral to understand the general structure and process of the current one. In the following sections, important automation concepts will be discussed, as well as project limitations.

## 2.1 Command Execution and Scope

Ordering commands for the program to execute is a type of computer programming commonly referred to as ‘scripting’. Scripting operations allows for thousands of simple commands to take place in milliseconds by replacing removing the need for human interaction. The process automation evaluated in this report employs the use of multiple scripting languages to generate reports. Each language has a unique purpose and scope, but may interact with one another to accomplish a single task. For example, when generating any given report, a Windows Command Line (WCL) script opens and triggers a macro in a Microsoft (MS) Excel workbook. These macros are written Visual Basic for Applications (VBA) and completely recolor, resize and reformat the data contained in the workbook. While both WCL and VBA scripts complete an ordered procedure, VBA command execution only takes place in the one MS Excel file, while WCL can trigger events to start in any file or any folder.

## 2.2 Relational Databases

When storing large quantities of data, databases are one of the only viable methods. Databases store digital information in the form of tables, which organize the information. Using a database allows for querying and filtering without harming any data, and provides a universal location for teams to access the information. A relational database is even better in that it creates connections between tables to ensure the stored information is update and accurate.

Relational Databases are often coupled with specific applications known as relational database management systems (RDBMS). These programs provide an easy interactive way of getting data into and out of a relational database. Some examples of RDBMS’ are Oracle and MySQL, but for the purpose of this process automation, MS Access was used as the main database access point.

It is important to note that spreadsheets are not databases, as they provide less functionality and less safety. A spreadsheet tool (ex. MS Excel) provides functionality for mathematical analysis, formatting and chart generation, while an RDBMS (ex. MS Access) provides an avenue for project collaboration, data structuring/management and big data storage (see table 1).

<https://www.makeuseof.com/tag/excel-vs-access-can-spreadsheet-replace-database/>

|  |  |
| --- | --- |
| Common Uses | |
| MS Excel | MS Access |
| data analysis | data management |
| mathematics, logic, computations | display data subsets, data structuring, complex queries |
| calculations, statistical comparisons | automation of common events |
| simple, sharable output | database management with multiple users |
| conditional formatting, chart management | reports for data summarization |

## 2.3 Constraints and Design Requirements

The automation system architecture evaluated in this report is a critical element of the daily reporting structure and therefore came with quite a few constraints and important criteria. One of the more difficult constraints was technical development team size. As mentioned in section 1.2, there was only one developer working on the whole process automation system at any given time, and for the past few years it has only ever been a coop student. Besides the individual student, there was no other resource for technical consultation besides the limited documentation.

An additional constraint limiting the optimization of the automation was the necessity of credentials and security. In most cases, coop students are not given sufficient credentials in order to rework the code base as they please, due to the fact that CWDS works in close conduct with the Canadian military. Some source data or databases required permissions above the scope of work permitted for the student in order to write (make changes), which meant that they were limited to work with that legacy code.

The criteria required of the automation process was also quite demanding. The automation was critical in generating daily, monthly and quarterly reports that would be shared company wide, and be factored into decisions made by upper management. The reporting frequency needed to be maintained, meaning any changes made on a given day should not affect the following days report generation. Additionally, due to the importance of the reports, data accuracy was critical. Miscalculations could have radial affects throughout the company if not spotted and handled early.

The report frequency and company dependence made it necessary to perform this evaluation; however, all the aforementioned constrains and requirements were to be met in any optimized process automation system suggested from this report. Regardless, evaluating potentially outdated systems is pivotal in maintaining productivity and avoiding risky practices.

# Enhancing System Architecture

Understanding the existing data automation system is integral in identifying areas of improvement, suggesting other models and evaluating the overall efficacy. The current system architecture follows a complicated design flow of interconnected RDBMS.

## 3.1 Existing Automation Model

CWDS employs the use of two Enterprise Resources Planning (ERP) programs, Systems Applications and Products (SAP) and SAP’s separate module, Manufacturing Execution System (MES). Both ERP’s are integrated RDBMSs, which track component numbers and product numbers from production to final sale. MES tracks production defects, while SAP tracks financial records. The automation requires the data from these two programs in order to create the final reports.

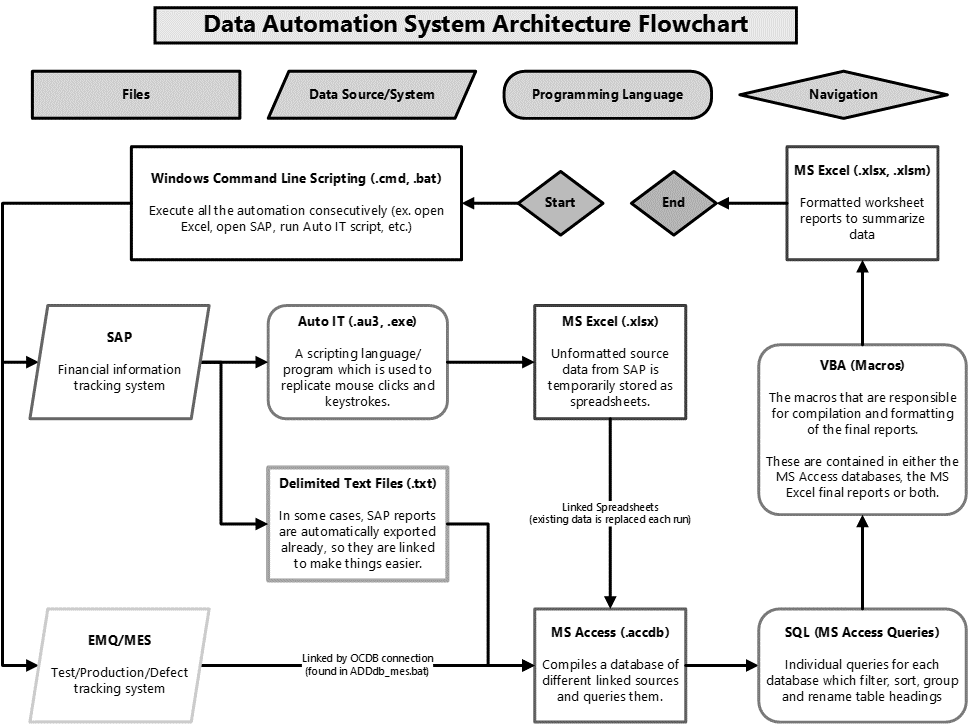


Figure 2 – Current automation system architecture flowchart

## 3.1.1 Data Extraction

In the case of SAP, the raw data comes from creating MS Excel spreadsheets and text files by using the Auto IT programming language. It mimics mouse clicks and keystrokes, signing into SAP under management credentials. A supervisor’s username and password are hardcoded into the script to gain access to the product financial data (due to the security clearance requirement).

In the case of MES, the raw data comes from an open database connection (OBDC) to the local RDBMS, in this case, MS Access. OBDCs enable an easy information transfer between the RDBMS and the data source, however, it must be instantiated outside of the RDBMS. This means an additional WCL script must be run before the MS Access database is able to update.

Each automated report is generated from its own MS Access database which has been configured ahead of time with the appropriate sources. For example, generating a report about the defect trend and its financial impact may require a spreadsheet from SAP, alongside a connection to MES. Both data sources are linked in the RDBMS (MS Access), automatically refreshing on viewing.

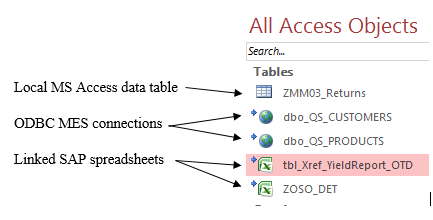


Figure 2: Example of linked source data in a Microsoft Access database

## 3.1.2 Data Manipulation

Once the source data is refreshed, Structured Query Language (SQL) create database queries, within the MS Access Databases. Since MS Access is an RDBMS, SQL can link the data sources based on similar field (ex. same product number, production date, etc.) and creates the tables that will be used in the reports (see Appendix A). Each query is saved locally in the MS Access database, available on the local network for public access, outside of report generation.

Once queried, Visual Basic for Applications (VBA) macros are used to export the data as MS Excel spreadsheets, fully accessible on a local network drive. In some cases, Additional MS Excel VBA Macros may be used for conditional formatting, creating graphs/pivot tables or adding supplementary information.

Having a large quantity of similar SQL queries on each MS Access slows the individual capabilities

## 4.1 Programming Languages

WCL scripting is the overarching initiator of the process automation system; opening files and calling the macros/queries they contain.

Auto IT is the automation languages used by CWDS in order to SAP’s graphical user interface to acquire source data, however, it has an inherent dependency, an application (.exe) compiler required to run it. Not only does this increase file size, but also processing power, in requiring another parallel application to be open and running.

An optimized solution would be to consolidate both the languages in some way, since both perform similar processes. An efficient alternative to either the existing methods is the usage of Python. Scripting with python similar to ‘natural language’ programming and is the basis programming language for modern engineers due to its ease. Python can do each operation faster due to its fasterness.

## 3.2 Alternative Automation Model

A split database is a data storage structure which separates the functional source components (tables, raw data, ODBCs) from the end user experience (queries, forms and reports). These are implemented in cases where full access is unnecessary for the database. In the CWDS automation process, the scripts only ever access the functional components for updates/edits, the queries are static and simply refresh for the next user.

To select the appropriate automation model, there are certain criteria which must be met: the economic cost of implementing the system change (which is calculated in terms of the hourly coop student rate (see Appendix something), the non-technical user friendliness, the automation performance change and the security risk associated with the change. These requirements are used in the decision making matrix in Table 1 to compare the suggested automation models. Comparing

|  |  |  |  |
| --- | --- | --- | --- |
| Criteria | Connected Database | Split Database | Comments |
| Automation speed | 0 | +1 | Split databases open and edit faster (less to open/render) |
| User experience | 0 | 0 | Users would have to export data to make changes, but would have much faster load times |
| Security | 0 | +1 | Users only permitted to view or download |
| Economic implementation | 0 | -1 | Dedicated time will cost the programmers wage, approximately a weak. |
| Total | 0 | +1 |  |

# Improving System Maintainability

For the existing process automation system to persist

The data processing requires the use of specific programming languages/applications for various tasks. Since programming languages are versatile, and may be combined in some cases, this leaves room for optimization. Consolidating languages use and scope also increases the overall ‘code cleanliness’ contributing to overall maintainability

## 4.2 Documentation

Documentation is a necessity for the long-term sustainability of programming systems, especially those with small technical teams or individuals. Providing adequate documentation allows for easier implementation and quicker transition times between teams and departments, in addition to a simplified debugging process.

In order to increase the efficacy of documentation some source suggests the implementation of mandatory README files, in the form of markdown, or text documents. These will allow for traceability and

## 4.3 Dependencies

Given any programming project, using an existing library is much easier and more cost effective than ground-up development. This however, results in project dependencies, which could result in complete system failures should those dependencies bug/become unsupported. Specifically in the case of process automation, these dependencies arise in the form of source data and automation applications/languages.

Source data is a matter of linking files to MS Access databases (see Figure 3). As per the maintainability, all it would take is file relocation, renaming, or privilege revocation to completely prevent report generation in the first place.

## 4.4 Additional Sustainability Systems

Currently, any sort of structured maintainability solution does not exist, and simply relies on user discretion. The criteria for a valid maintainability solution are as follows: allows active programmer to easily track changes, maintains system structure without (or with minimal) downtime, ensures data accuracy does not suffer, is safe (security-wise) and can be easily changed.

This may be accomplished by creating a separate full-credential user specifically for the purpose of automating report generation, with their access only permitted to the technical team. If this such redesign were possible, it would allow for self-contained table linking for each MS Access database, ultimately preventing any user from making changes which unbeknownst to them, crashes automation.

Another maintainability solution could be the implementation of version control software. Distributed version control software (such as Git) are : cite . Using version control will maintain report accuracy while creating a better testing environment for future system improvements even if the minimal team size does not change. Files can be restored without administrator access and all changes can be tracked to ensure project traceability.

Table 2: Decision matrix to determine the optimal long-ter

|  |  |  |  |
| --- | --- | --- | --- |
| Criteria | Full Access User | Version Control | Comments |
| Traceability | 0 | +1 | Version control tracks each edit/change permanently |
| Minimal downtime | 0 | +1 | Remote copies do not cause downtime until deployment |
| Maintains data accuracy | 0 | 0 | Both systems do not manipulate data directly |
| Security | 0 | +1 | Full user access for every team member is inherently risky |
| Versatility | 0 | -1 | User details can be changed/passed on |
| Total | 0 | +2 |  |

# Conclusions

With respect to the current automation system architecture, based on the report body’s analysis, it is concluded that:

1. Each generated report is connected to its own individual database MS Access database
2. Many of the MS Access databases handle both interface objects (queries, forms and reports) as well as data management (linked tables, ODBCs and local data)
3. Supervisor credentials are hard coded into system, leaving room for a critical security failure
4. Using split databases instead of connected databases would improve the overall articecture of the system
5. WCL and Auto IT scripting are similar scripting languages which can be consolidated (with respect to GUI

With respect to the process automation system maintainability, based on the report body’s analysis, it is concluded that:

1. The original system documentation was insufficient for long-term use; however, the additional documentation provided by past coop students
2. Many of the current reporting databases depend on manual files which could easily affect automation procedure due to human error, or
3. If a system failure were to come at an unfortunate point in time, (ex. coop changeover, employee vacation, etc.), there is an insufficient staff to maintain the system
4. Using the Python scripting language rather than WCL and Auto IT would increase code legibility, and simplify the existing process, while maintaining core functionality.

# Recommendations

Based on the analysis and conclusions contained in this report is recommended that more company resources be allocated to the simplification and redesign of the process automation system. This should

1. Employ the use of version control
2. Larger teams
3. More people should know what the software is made of
4. Switch to python as a the controlling language
5. Use a split database

# Glossary

**Curtiss Wright Defense Solutions (CWDS):** The Company whose process automation system is the subject of this report.

**Quality Engineering (QE):** The department of technical engineers who depended upon the process automation system.

**Microsoft (MS):** The technology company whose software and applications are employed in the automation process.

**Enterprise Resource Planning (ERP):** A manufacturing process which ties together production, sales, inventory, purchasing and supplies, most commonly by way of databases and serial numbers.

**Systems Applications and Products in Data Processing (SAP):** The critical ERP software used for increasing productivity in specific sectors, mainly financial records and product tracking.

**Manufacturing Execution System (MES):** An ERP production database which logs all the component defects and testing outcomes from every product manufactured on site.

**Visual Basic for Applications (VBA):** A subset of the programming language *Visual Basic* used by Microsoft Office Applications to create macros.

**Relational Database Management System (RDBMS):** A software/tool which is used to facilitate storing and interacting with relational databases. An example of a common RDBMS is the Microsoft Access application.

**Structured Query Language (SQL):** A programming language used when interfacing with databases to manipulate and query data.

# Appendix A – SQL code for use in Data Manipulation

Querying with SQL is the backbone of data manipulation; a key step in report generation and consequently process automation. SQL start off by selecting columns from tables that it will be manipulating, using the SELECT and FROM clauses. Using a relational database allows for effective JOIN clauses, connecting related data from different tables into a new selection. The WHERE clause determines what filtering will be applied to the selection, in order to gather only the relevant, accurate data. MS Access (or any RDBMS) allows for SQL queries to be saved as separate data files, and exported individually to reports. See Figure A-1 for an example query using SQL.

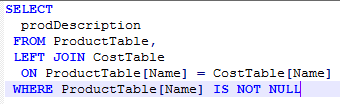


Figure A-1: An example SQL query joining data sources