Ossi Kotala

Evaluation of SAS Data Integration Studio as Tool to Improve Statistical Production

Helsinki Metropolia University of Applied Sciences
Master’s Degree
Information Technology
Master’s Thesis
08 November 2016
Preface

As more data becomes available, greater emphasis is placed on the data processing tools and processing environment. The tools, and means for using them, are required to be on a proficient level: It is an ongoing challenge how to maintain the level of quality in this regard, and with this study I will, for my own part, take part in resolving it. For me, this study offered a perfect possibility to strengthen my expertise with data processing and statistical information systems.

I wish to convey my gratitude towards the experts in subject matter as well as the stakeholders involved in the evaluation of the findings. All of them provided valuable feedback for my study. Furthermore I wish to extend my appreciation to my instructor at the case company Mr Perttu Pakarinen and at Metropolia University of Applied Sciences Mr Juha Kopu and Ms Zinaida Grabovskaia.

Last but not least I wish to acknowledge the strong support from my wife Ilona, who stayed in with the children for those evenings I spent away studying. Thank you. Now it’s your turn.

Refreshing your knowledge through education every once in a while is almost a mandatory requirement.

Ossi Kotala
Abstract

The purpose of this study was to examine and evaluate how the statistical software development would be enhanced in Statistics Finland by utilising SAS Data Integration Studio.

The case company is a government agency creating statistics for the use of the general public, government administration and enterprises. The statistics cover all aspects of society. In order to produce statistics, the case company uses SAS Institute’s statistical information systems to advance the development of statistical production. This study aims to improve the statistical production by ensuring the quality of development.

The tools of processing data are of a great significance, especially when large amount of data is being processed. The quality and performance factors are to be kept at a proper level to ensure efficient operations. Therefore, this study evaluates the chosen tool and the new SAS environment (which has already been used at the case company for some time) to find out how they could be utilised more effectively.

The tool is SAS Data Integration Studio (DIS), which is an ETL (Extract-Transform-Load) tool, whose purpose is to extract data from a given source and transform it so that necessary statistics can be created. Current state analysis was conducted to describe the statistical information system in the case company, going through the server based SAS architecture, metadata environment and the role of DIS in the aforementioned. The study conducted tests regarding statistical software development, examined the best practice of DIS processing, describing working methods, strengths and weaknesses and other technicalities, including the server/metadata environments. Additionally, project work, documenting, and other elements involved in statistical production are covered. Finally, an evaluation of the findings is provided, determining whether the objective was met. The end result is an in-depth evaluation and recommendation report on how to conduct statistical software development with SAS Data Integration studio.

Keywords

SAS, Data Integration Studio, ETL, Statistics
# Table of Contents

Preface  
Abstract  
Table of Contents  
List of Figures  
List of Tables  
Abbreviations  

1 Introduction  
1.1 Statistics Finland  
1.2 New SAS Architecture and Data Integration Studio (DIS)  
1.3 Research Question and Objective  

2 Method and Material  
2.1 Research Approach  
2.2 Research Design  

3 Statistical Data Processing, ETL and Data Integration  
3.1 General Statistical Business Process Model (GSBPM)  
3.2 ETL (Extract-Transform-Load) Concept  
3.3 Main Components of ETL and Data Integration  
3.4 SAS Data Integration Studio (DIS)  

4 Current Statistical Information System  
4.1 Introduction and Background  
4.2 Metadata Environment and DIS  
4.3 Starting with Data Integration Studio  
4.4 Data Integration Studio Transformations  

5 DIS Development: Evaluation of Best Practices and Working Methods  
5.1 Data Integration Studio Settings  
5.2 Data Integration Studio Development  
5.2.1 SQL Join versus SAS Merge  
5.2.2 Table Loader Transformation  
5.2.3 Summary Statistics  
5.2.4 Parallel Processing
List of Figures

Figure 1: Research Design ........................................................................................................... 8
Figure 2: The GSBPM Model (GSBPM, 2013) ............................................................................ 11
Figure 3: Data Processing (GSBPM, 2013) ............................................................................... 12
Figure 4: ETL (Extract-Transform-Load) .................................................................................. 14
Figure 5: ELT (Extract-Load-Transform) ................................................................................... 15
Figure 6: SAS Data Integration Studio Environment (SAS, 2016) ............................................ 17
Figure 7: SAS Architecture (The Case Company, 2016) ............................................................. 19
Figure 8: SAS Metadata Server Environment (SAS, 2016) ....................................................... 21
Figure 9: Example Transformations ........................................................................................... 23
Figure 10: Unfinished Work Flow Example .................................................................................. 24
Figure 11: Completed Work Flow Example .................................................................................. 24
Figure 12: Workflow Example - Source Tables .......................................................................... 24
Figure 13: Extract Transformation - Mappings .......................................................................... 25
Figure 14: Expression – SQL Case-When Structure .................................................................... 25
Figure 15: Kanban Method (Kaizen News, 2013) ..................................................................... 26
Figure 16: Usage of Sticky Notes in a Job ................................................................................... 26
Figure 17: Library Contents ........................................................................................................ 27
Figure 18: REST Options (SAS, 2016) ...................................................................................... 27
Figure 19: REST and SOAP Transformations ............................................................................. 28
Figure 20: Table Loader Transformation .................................................................................... 28
Figure 21: Summary Statistics Transformation .......................................................................... 28
Figure 22: Summary Statistics Options ....................................................................................... 29
Figure 23: Loop Transformation ................................................................................................ 29
Figure 24: Data Transformations ................................................................................................ 29
Figure 25: User Written Properties, Code ................................................................................ 30
Figure 26: SQL Transformations ................................................................................................ 30
Figure 27: SQL Join Transformation .......................................................................................... 30
Figure 28: SQL Merge Transformation ....................................................................................... 31
Figure 29: Transformation Status Handling ............................................................................... 31
Figure 30: Settings - Auto-mapping ........................................................................................... 32
Figure 31: Global Level Settings ................................................................................................ 33
Figure 32: Job Specific Settings .................................................................................................. 33
Figure 33: Settings - Pass-through ............................................................................................. 33
Figure 34: In-Database Processing ............................................................................................. 34
Figure 35: Check Database Processing ................................................................. 34
Figure 36: Settings - Join Conditions and Control Flow ......................................... 34
Figure 37: SQL Join versus SAS Merge, Server Environment .................................. 35
Figure 38: SAS Merge vs SQL Join Statistics, Server Environment ......................... 36
Figure 39: Application Created SQL Join Code ...................................................... 36
Figure 40: User Created SAS Merge Code ............................................................ 36
Figure 41: In-Database Processing, Server Environment ........................................... 37
Figure 42: In-Database Processing Statistics, Server Environment ......................... 37
Figure 43: Out-Database Processing, Server Environment ...................................... 37
Figure 44: Out-Database Processing Statistics, Server Environment ....................... 38
Figure 45: Table Loader Test, Server Environment .................................................. 38
Figure 46: Table Loader Test Statistics, Server Environment .................................... 39
Figure 47: Summary Statistics Work Flow ............................................................... 39
Figure 48: Summary Statistics Transformation ......................................................... 39
Figure 49: Summary Statistics Properties 1 ............................................................. 40
Figure 50: Summary Statistics Properties 2 ............................................................. 40
Figure 51: Summary Statistics Transformation 2 ....................................................... 40
Figure 52: Code in Base SAS .................................................................................. 41
Figure 53: Summary Statistics Work Flow 2 ............................................................ 41
Figure 54: CPU Code ............................................................................................. 42
Figure 55: Loop Job (Parallel Processing) ............................................................... 42
Figure 56: Parallel Processing Configuration ........................................................... 43
Figure 57: SQL Execute Example .......................................................................... 43
Figure 58: The Macro Job in DIS ........................................................................... 47
Figure 59: The Macro Job in DIS, Refined .............................................................. 48
Figure 60: Operational Concept of SOA ................................................................. 49
Figure 61: Web (Browser) Based Data Processing Application ................................. 52
Figure 62: Database Processing ............................................................................. 55
Figure 63: Metadata ............................................................................................... 55
Figure 64: Automatic Table Contents ..................................................................... 56
Figure 65: Web Services ....................................................................................... 56
Figure 66: Parallel Processing ............................................................................... 57
Figure 67: Work Flow ......................................................................................... 57
Figure 68: Performance Statistics ......................................................................... 58
List of Tables

Table 1: SAS Merge vs. SQL Join ................................................................. 37
Table 2: Summary Statistics ................................................................. 40
Table 3: Summary Statistics Output ......................................................... 41
Table 4: Parallel Processing Results ......................................................... 43
Table 5: Service Configuration ............................................................... 51
Table 6: URL of the Service ................................................................. 51
## Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACT</td>
<td>Access Control Template</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>CSV</td>
<td>Comma Separated Values</td>
</tr>
<tr>
<td>CTO</td>
<td>Chief Technical Officer</td>
</tr>
<tr>
<td>DBMS</td>
<td>Database Management System</td>
</tr>
<tr>
<td>DIS</td>
<td>Data Integration Studio</td>
</tr>
<tr>
<td>EG</td>
<td>Enterprise Guide</td>
</tr>
<tr>
<td>ETL</td>
<td>Extract-Transform-Load</td>
</tr>
<tr>
<td>FTP</td>
<td>File Transfer Protocol</td>
</tr>
<tr>
<td>GSBPM</td>
<td>General Statistical Business Process Model</td>
</tr>
<tr>
<td>HP</td>
<td>Hewlett Packard</td>
</tr>
<tr>
<td>HTML</td>
<td>Hypertext Markup Language</td>
</tr>
<tr>
<td>HTTP</td>
<td>HyperText Transfer Protocol</td>
</tr>
<tr>
<td>IBM</td>
<td>International Business Machines</td>
</tr>
<tr>
<td>ICT</td>
<td>Information and Communication Technology</td>
</tr>
<tr>
<td>IT</td>
<td>Information Technology</td>
</tr>
<tr>
<td>JMP</td>
<td>John's Macintosh Project</td>
</tr>
<tr>
<td>JSON</td>
<td>JavaScript Object Notation</td>
</tr>
<tr>
<td>ODBS</td>
<td>Open DataBase Connectivity</td>
</tr>
<tr>
<td>REST</td>
<td>Representational State Transfer</td>
</tr>
<tr>
<td>SAP</td>
<td>Systems Applications and Products</td>
</tr>
<tr>
<td>SAS</td>
<td>Statistical Analysis Systems</td>
</tr>
<tr>
<td>SAS_BI</td>
<td>SAS Business Intelligence</td>
</tr>
<tr>
<td>SOA</td>
<td>Service Oriented Architecture</td>
</tr>
<tr>
<td>SOAP</td>
<td>Simple Object Access Protocol</td>
</tr>
<tr>
<td>SQL</td>
<td>Structured Query Language</td>
</tr>
<tr>
<td>STP</td>
<td>Stored Process</td>
</tr>
<tr>
<td>UNECE</td>
<td>United Nations Economic Commission for Europe</td>
</tr>
<tr>
<td>UNIX</td>
<td>Uniplexed Information and Computing System</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
</tbody>
</table>
1 Introduction

This study examines and evaluates how the statistical software development would be enhanced in Statistics Finland by utilising SAS Data Integration Studio. The case company is a government agency creating statistics for different purposes. The case company and the statistical information system are described in this chapter.

1.1 Statistics Finland

The case company is Statistics Finland, a government agency. The purpose of the case company is to provide official statistics for the Finnish society. The mission is to support decision-making based on facts, as well as scientific research, by producing reliable statistics, studies and datasets describing society. (The Case Company, 2016). The case company’s tasks are to:

- Compile statistics and reports concerning social conditions
  - Collect and maintain data files on society
  - Provide information service and promote the use of statistics
  - Develop statistical methods and conduct studies supporting the development of statistics

- Develop the national statistics in co-operation with other government officials
  - Co-ordinate the national statistical service
  - Participate in and co-ordinate Finland’s international statistical co-operation

The case company produces around 160 sets of statistics, 550 publications yearly. A majority of the data for the statistics is derived from existing registers of general government. The case company additionally collects data with inquiries and interviews when the necessary data cannot be obtained via other means. The collection methods are:

- Telephone interviews
- Face-to-face interviews
- Postal inquiries
- Responding online
- Administrative data and registers
The statistics are used for planning of public services, enacting of legislation and monitoring the effects of decision making. In particular, central and local government organizations and research institutions are utilising the statistics. However, the media, enterprises and private individuals have found them useful as well. For example, municipalities use statistical data when they plan the locations of day-care centers, schools and health care services in residential areas.

Monitoring the development aspect of a society is an important factor; statistics can be used to compare wages and salary development in a given industry. Statistics provide information on price development regarding vehicles, apartments, as well as data on parliamentary elections, national accounts or households’ spending and saving.

In order to produce statistics, the case company utilises SAS Institute’s statistical information systems, by the statistical and ICT experts, to advance the development of statistical production. *This study aims to improve the statistical production by ensuring the quality of development.* The situation before the current (new) SAS architecture was such that the software development regarding statistical production was performed locally on individual workstations. The employee had in her/his use a personal computer, on which the user was running the information systems of statistical development. Client software, Base SAS, was used to perform the needed tasks and in some cases Enterprise Guide. These clients were able to connect indirectly to a UNIX server hosting statistical data (as well as to some databases) for processing of larger data files. However, no metadata server existed or a centralised way to perform the software development.

The above led to a situation in which individually created code was piling up, and after some time, it often happened that the code was challenging to support and dependent on a single developer. Also, when there were no commonly agreed conventions regarding the software development, a colourful range of solutions was introduced, each being uniquely different. This was not the best of situations regarding the supportability or documenting the solutions. Data Integration Studio has been used for statistical software development at the case company for some time; however, the new architecture brought with it a wider and more active usage of it.
1.2 New SAS Architecture and Data Integration Studio (DIS)

The aforementioned, workstation based architecture, has been recently replaced with a new architecture plan, which in the case company includes development of the statistical information systems and running the production in a metadata-based server environment, controllably and in a centralized manner. Also, a new developer’s tool, SAS Data Integration Studio (DIS), has been taken into more active use. DIS is a client application connecting to the metadata-based development environment.

The assumption from the case company is that centralizing the statistical processing will enhance the operations so that the development of statistical information systems will be more efficient and the statistical production itself will become more transparent and manageable. The test results and DIS programs can be used to verify the impact level of the effectiveness on statistical production. Using a more full-extent method of DIS development, programs would be perceivable, straightforward to develop and light to support. The statistical software will be developed and maintained utilising metadata on a common server, to which every developer has access. The programs need to be divided into smaller ensembles, all of which are documented on a detailed level (Kanban model). Given that the programs are as generic as possible, well documented, and not tied to a specific developer, the efficiency of support work will be increased. This means that a new developer is capable of proceeding with the work, with minimum extra effort in getting to know the new software.

The developer’s tool is SAS’s Data Integration Studio (DIS), which this study focuses on. DIS is an ETL (extract, transform, load) tool, used at the case company already for some time. Its purpose is to extract data from a desired source, transform it as needed, and load it up to the target system (for example a database table). It uses visualized programs called “jobs” to achieve the same as with EG’s tasks or Base SAS’s coding. These programs are created using metadata on a server, so called “transformations”, ready-made functionalities, which perform certain actions within the ETL chain. The transformations can be complemented or amended with Base SAS coding.

At this moment both EG and DIS are used in the case company in statistical software development and support work: DIS usually for larger and more complex statistics, and EG for smaller (however not consistently) statistics (by actuaries managing the statistical production). The objective of this study is to find out if and how Data Integration Studio
could be utilised to a fuller potential in the statistical software development and support work, so that maximum benefit is received. If/when proven beneficial, more active usage of DIS needs to be promoted and its purpose increased. If the potential capability of DIS could be employed effectively, it could be used as such to create the whole development and support (and even production) -line. If there would be an effective development, support and production -line from collection to statistics, using DIS, the efficiency of statistical operations would be ensured on this part by using the new IT-architecture in the case company.

However, for ad-hoc needs (data inspections, verifications), there can be another alternative in which they (ad-hoc tasks) are performed in concert with the actual production, using EG or other tools such as JMP (SAS). This is in a case where the usage of DIS is seen as too heavy or complex for quick data inspections. On occasion, and at least for the transition period, a need might still arise to use additional tooling. The actual development work with DIS would be performed to the extent possible. However, due to the complexity of DIS, nature of the statistic and time constraints, this might not be fully realised. In this case, EG would remain the primary developer’s tool.

1.3 Research Question and Objective

*The problem* is that the statistical software development and support need to be more efficient. The developed software is used to run statistical production (creating statistics) in the case company. The focus of this study is to evaluate if and how SAS Data Integration Studio, DIS) could be used to achieve this (more efficient development methods). DIS is already used in the case company, however, further studies are needed to discover how it can be utilised to a fuller potential.

This study answered the following *research question*:

<table>
<thead>
<tr>
<th>How to make the statistical software development more efficient with SAS Data Integration Studio?</th>
</tr>
</thead>
</table>

*Statistical software development with SAS Data Integration Studio* signifies that the tool (DIS) is used to create SAS programs via the use of a workflow-like graphical user interface. These programs are formed by the application on the background, based on the
user actions with the workflow. They can also be complemented or amended by the user manually. It is possible to create the whole program by manual coding (as with Base SAS), however, this is not the purpose of the tool. The purpose is to use the ready-made functionalities of the tool, so called transformations, and only when necessary, supplement it with manual coding. These programs are run from the user interface as such (running the work flow, which executes the code behind it), or a so called Stored Process is created out of the code, containing the SAS program. This Stored Process can be run from another tool such as SAS Enterprise Guide, from a web browser or from a Windows client. These programs create the statistics of the collected data.

Basically, it is necessary to ensure that the solutions (developed programs) would be:

- **Efficiently developed**, however, without risking quality.
- **Effortlessly supported**. The follow-up development/support needs to be made as easy as possible, so that the solutions would be lightly supported by any expert. Expert-dependency needs to be avoided as well as finding common working methods (using the development methods).
- **Generally applicable**. Reusable, when possible: One solution would feed the need of many statistics.
- **Well documented**. Each solution needs to be documented on a detail level and the documentation needs to be broken down to smaller units (e.g. one DIS job / documentation).
- **Performing well**, when it comes to larger data masses.
- **Utilising Service Oriented Architecture (SOA)** when possible. How can DIS advance the case company’s strategy regarding the Service Oriented Architecture?

The objective of the study is defined as such:

| The objective is to evaluate if and how it is possible to enhance the statistical software development and support in the case company using SAS Data Integration Studio (DIS). |

The case company’s ICT strategy outlines that the development and support of the statistical information system (DIS in this case) should be efficient and the amount of support work decreased. A clear need exists to find out ways to develop the programs most efficiently, in a manner in which they would be logical and easy to understand (easy to conduct follow-up development/support).
The outcome is an in-depth evaluation report and recommendations how to utilise Data Integration Studio (DIS) for improving statistical software development in the case company.

The structure of the study is as follows: Firstly, the introduction of the case company presents the research background and problem, defines the research question and points out the significance and scope of the study. Secondly, the study outline, structure of the study, is described, after which the method and material is reviewed. Existing knowledge is surveyed as a literature review, where the background theory and the main concepts of ETL are described, as is DIS’s relation to the ETL world. The study continues with current state analysis to describe the statistical information system in the case company, going through the server based SAS architecture, metadata environment and the role of DIS. As the study progresses, it provides an in-depth review and evaluation of the development aspect and best practice of DIS processing. The study describes working methods with DIS, strengths and weaknesses and other technicalities regarding DIS development and the server/metadata environments. Service Oriented Architecture (SOA) is described as well, as are the possible advancements to it by the use of DIS in the case company. Finally, the study sums up the conclusions and recommendations.
2 Method and Material

2.1 Research Approach

This “Method and Material” section acts partly as the research plan for this study. Part of the material which would normally fall under this section (Method and Material) is already included in the “Introduction” section.

The study was performed using a design science approach and is in the form of an evaluation study (assessment) of the ETL tool SAS Data Integration Studio. The mentioned tool is used at the case company for processing data and performing statistical production (creating statistics).

Definition of evaluation by Prof. William M.K. Trochim, 2006:

According to Trochim, probably the most frequently given definition is:

“Evaluation is the systematic assessment of the worth or merit of some object.”
(Trochim, 2006)

Trochim continues:

“This definition is hardly perfect. There are many types of evaluations that do not necessarily result in an assessment of worth or merit. - Descriptive studies, implementation analyses, and formative evaluations, to name a few. Better perhaps is a definition that emphasizes the information-processing and feedback functions of evaluation:

Evaluation is the systematic acquisition and assessment of information to provide useful feedback about some object.” (Trochim, 2006)

Above, Trochim emphasizes the informational aspect of the assessment of the artefact, so that actionable feedback could be obtained. This is the case also regarding the present study: The feedback is used to iteratively advance the evaluation of the artefact itself, in order to further strengthen the outcome values of the evaluation.
2.2 Research Design

The research design of this study includes the following steps, as Figure 1 illustrates:

![Research Design Diagram]

* Figure 1: Research Design

The Figure above (Figure 1) shows the different stages of the research design. The stages are described below.
“The Existing Knowledge” section will follow “Introduction” and “Method and Material” sections. In this part the study will continue with the literature review (Section 3: Statistical Data Processing, ETL and Data Integration). The mentioned review was conducted in order to discover the current existing knowledge about the subject matter.

“Current state analysis” will be presented in the succeeding section, in which the current statistical information system (emphasis on Data Integration Studio) is described (Section 4: Current Statistical Information System). The current working methods, tools, etc. are described, as well as literature reviews (the case company documentation and external) and knowledge transfer from colleagues.

“Evaluation of DIS” is reported next, where the best practice, working methods, test results and other essentially covered topics are evaluated (Section 5: DIS Development: Evaluation of Best Practices and Working Methods). The evaluation was conducted with the stakeholder group of the case company, and during the process, feedback was received, analysed and validated (see Section 6 for a description of the process). When found valid, the information was entered into the study. Evaluation feedback was gathered through one-to-one interviews, email and instant messaging communications, ad hoc meetings and workshops. The scope and objective was likewise formed with the stakeholders of the case company. Section 6 covers a description of the evaluation, as well as evaluation results, and analysis and validation. In this conjunction, the study offers evaluations on:

- Improving the development methods (efficiency, quality, reusability) of DIS, i.e. the ETL process logic, data flow, readability, performance, documentation, etc.
- Practical uses of DIS: What kind of challenges are associated with development/production with DIS? (What are the requirements for personal change and attitude adjustment towards new developments methods?)
- Service oriented architecture and its utilisation at the case company, especially the use of web services and how they can further advance reusability and distribution of commonly used solutions.
- The impact level of effectiveness on statistical production can be verified by some of the results, such as programs and revised working methods.
- Examining the impact of server-based SAS-architecture and the role of DIS as a server-based tool as opposed to workstation tools. (How can it further enhance the production of statistics?)
“The Solution” part offers conclusions and recommendations, which are based on the evaluations (Section 6: Conclusions and Recommendations). It includes a description of the iterative “Feedback Process” explaining the methods of obtaining and analysing the feedback, what was asked, who was involved, and what the evaluation results were. The conclusions and recommendations were also validated with relevant stakeholders and statistical / ICT experts of the case company. The developed solution is as described in Section 1.3, which is an in-depth evaluation report (this study) providing recommendations on how to utilise SAS Data Integration Studio for improving statistical software development in the case company.

Literature Search Sources: As noted, this study covered extensively available sources to discover the most suitable and up-to-date information regarding the scope of the study. Among the materials were online and printed materials from SAS Institute, however, not limited to this; the case company and external materials were also reviewed. Other sources included articles, white papers and additional publications. In-company material included product and configuration documentation, internal wikipedias and other specification.

Information Search Strategies: Different strategies were utilised in search for the relevant information. Papers and publications were examined on-site in Metropolia libraries as well as online. Used search terms were: “Data Integration”, “SAS Data Integration Studio”, “ETL”, “Data Processing”, “Service Oriented Architecture” and “SOA”, as well as other combinations. The information obtained was scrutinized through the core pipeline of the study, thus placed in proper context. Information from different sources was merged and validated against each other. Whilst author’s aspiration was in retrieving recently proven results (given the fact that latest trends in ICT need to be followed), primary sources were not neglected.

Research Material: As mentioned, this study is performed using a design science approach of iterative nature when necessary (literature survey and analysis), mostly from online materials and materials provided by SAS institute (online and printed) and the case company. Data analysis methods are of both qualitative and quantitative nature, likewise are the evaluation methods regarding results. The issues and matters at hand were studied using in-company data (specifications, internal wikipedias, databases, datasets) and data from online sources (SAS, other). Methods of data analysis included content and statistical analysis with SAS Data Integration Studio (ETL tool).
3 Statistical Data Processing, ETL and Data Integration

This section describes the current context (acts as a literature review into the existing knowledge) of data processing, ETL (Extract-Transform-Load) concept, data integration in general and Data Integration Studio.

3.1 General Statistical Business Process Model (GSBPM)

Essentially data processing is about extracting information from the data, preferably useful. Any form of advanced data mining or simpler methods such as cleansing, validating or aggregating, is a form of data processing; it is done in order to get value out of data. Business decision making, finding direction in politics or creating statistics, are all examples of a situation where specific data processing is needed. They all appreciate the value derived from the data. The outcome of data processing is a report, diagram, statistic, etc. Data processing follows a distinct pattern - dependent on the usage scenario - and in this study, it is tied to the statistical context. Therefore, data processing in this study is tied to the GSBPM, or the General Statistical Business Process Model as Figure 2 illustrates:

![Figure 2: The GSBPM Model (GSBPM, 2013)](image-url)
The GSBPM model, as a whole, is illustrated in Figure 2 above, which describes the model in its entirety, from requirement specification to evaluation. However, the present study focuses on the *processing* part of the model, as demonstrated in Figure 3.

GSBPM is created to describe statistical processes in a coherent way. The model has been created by the Statistical Division of UNECE (United Nations Economic Commission for Europe). GSBPM defines a set of business processes needed to produce official statistics. It provides a standard framework and common terminology, in order to help statistical organizations to modernize their statistical production processes. (GSBPM, 2013).

*Data processing* of GSBPM is designed as such (Figure 3):

![Data Processing Table](image)

Figure 3: Data Processing (GSBPM, 2013)

Figure 3 above depicts more closely the process part of the model, and to be more precise, the *data process*. This is a proper example of what is meant by data processing in the context of *statistical data processing*.

According to the GSBPM model, the typical statistical business process includes the collection and processing of data to produce statistical output. However, the GSBPM model is also applied to cases where existing data is revised (or time-series are re-calculated), for example as a result of better source data quality. In these cases, according to the GSBPM model, the input data is the previous released statistic, which is then processed and analysed to produce revised outputs. The model suggests that in these cases, certain sub-processes and even some phases (the early ones) could be omitted. (GSBPM, 2013)

The above statement from GSBPM is rather vague: If there are changes in the source data quality this means that the data is being gathered / received from somewhere, so
collection has to exist. One cannot have “improved source data” and not do the collection; this would leave the new data unnoticed. Therefore the new input data is not the previous released statistic, but the improved source data itself.

*The sub-sections* of the data processing phase of the GSBPM model are described below.

*Data Integration (5.1):* Data Integration takes place right after the collection phase. The purpose of data integration is rather self-evident: to combine data from different sources. For example in creation of statistics such as national accounts, integration is needed (GSBPM, 2013).

*Data Classification (5.2):* A proper example of classification is the industry classification. In this case, there might be an incrementally changing numerical value in a variable, which is then converted into a proper name of the industry. A classification database is needed for this conversion to take place.

*Data Validation (5.3):* All data needs to be validated. If not, its quality and integrity cannot be confirmed. Usually data is validated against a set of rules, which is performed automatically. Examples of automatic validation are misclassification and outlier detection. Manual data validation needs to be avoided, which is seen as an unfavourable way of working. If data is not passing the validation, it is discarded, in which case manual validation would be acceptable.

*Data Editing (5.4):* The editing section includes also data imputing. Editing and imputing is, again, performed on rule set based actions. The purpose of this sub-section is to make data more coherent and uniformed, by completing or modifying it.

*Deriving New Variables (5.5):* In addition to modifying the existing data, new variables need to be derived and drawn from it. These variables are results of data analysis, which brings forth information, which is needed to be carried forward. They are therefore output into new variables.

*Weights Calculation (5.6):* Usually a good overall sampling of the target population is needed, meaning the representation of groups needs to be balanced. For example, if there is a survey respondent in under-represented group, they get a weight larger than
1, and respondents in over-represented groups get a weight smaller than 1. These weighted values create reliability to the outcome of the survey, and can make the survey results of the target population “full” and subject to comparison. (Applied Survey methods, 2015)

Aggregation (5.7): Often, data is in a unit level form, which might mean that a single individual can be identified from it. This data can be aggregated, or in another words summarized by the common characteristics of it. A proper example would be to aggregate material form the tax administration by the classifying variables, in order to get results regarding paid and received taxes on a city, region or a country level.

Finalizing Data Files (5.8): When the data has been processed, it needs to be finalized (made ready) for the following phase. Data needs to be joined, appended and formulated into a form and format, which is required by the next part of the model.

3.2 ETL (Extract-Transform-Load) Concept

The ETL (Extract-Transform-Load) concept can be seen as a process, which extracts data from needed sources, transforms it into a desired format, and loads it up to the target system (usually an operational data store or a data warehouse). This process is generally used when data is being extracted from several different sources, requiring data integration. Figure 4 illustrates the concept:

Also, a different approach is sometimes used when describing data processing and the ETL concept. When the performance becomes a notable factor, the data is loaded up to a so called staging area, which is a part of the target system database, before the transformation takes place. After this the transformation is being performed in-database,
bringing along with it performance improvements\textsuperscript{1}, especially when complex data manipulations with large data masses are performed (for example credit card fraud detection). From the staging area the transformed data is being transferred to the proper place in the data warehouse. This approach is referred to as ELT (Extract-Load-Transform) and demonstrated in Figure 5:

![Figure 5: ELT (Extract-Load-Transform)](image)

The phases of ETL (and ELT) processes are described in the following section.

3.3 Main Components of ETL and Data Integration

This section presents the main components of the ETL (Extract-Transform-Load) and Data Integration processes.

*Extract:* Firstly, the data is being extracted from wanted sources, which can practically be in any format, such as relational / non-relational database, XML or flat files. The purpose of this phase is to convert the data into a unified format, make it ready for processing and send it forward. Data validation often takes place already in this phase, which is a crucial step in assuring the quality in the process.

*Transform:* Secondly, the data needs to be transformed as required. This usually happens against a set of rules specified within the ETL software. Generally in this phase data integration is performed, however, also transposing and deduplication takes place. Classification, editing (imputing), deriving new variables and aggregation are part of this phase, likewise are weights calculation, change percentages and outlier detection. Additional validations of the data and quality assurance can be reiterated here.

\textsuperscript{1} In-database processing enables blending and analysis of large sets of data without moving the data out of a database, providing significant performance improvements. (Alteryx, 2016)
**Load:** Thirdly, the data needs to be loaded up into the target system, which usually is an operational database, data warehouse or a data mart. The practicalities regarding this vary between different companies, some load up daily, some monthly or yearly. Problems generally are prone to gather up when loading the data into the database, thus bulk operations and parallel processing are encouraged when possible. The ways to perform optimization regarding database processing or loading the data to the target system include memory handling and configuration (regarding for example parallel processing).

It is interesting to note the capabilities of ETL: Syncsort and Vertica recently broke the Database ETL World Record Using HP BladeSystem c-Class: Extracting, transforming and loading 5.4 terabytes under one hour. (PR Newswire, 2016)

Some, for example Phil Shelley (CTO at Sears Holdings), imply that the traditional ETL is nearing its end, because of new ways of processing data. For example Hadoop is on the rise in this regards; its capabilities of processing data are stronger according Shelley, given that the data never leaves Hadoop. Others, such as James Markarian (CTO at Informatica), are saying that this is still business as usual and it will not change the way traditional ETL process is conducted. (Information Week, 2012). An example of data integration is merging of two companies, and especially their databases (Data Integration Info, 2015). Another example would be when scientific data is being compared against each other or merged, in order to form joint follow-up research.

The challenges of data integration are technical ones, when merging two large and often mainly different databases from disparate sources. An even more demanding challenge is the design and implementation process of the data integration itself; the design process needs to be initiated and driven by the business side, not IT. Then, after a successful implementation, testing needs to be conducted, in which both parties, business and IT, are closing in on the common goal: successfully integrated and tested information system. (Data Integration Info, 2015). Data Integration usually takes place in the transform phase of the ETL process. However, this may vary and take place iteratively also in another phase. There are several ways of handling data integration regarding the technicalities of it. Some prefer manual integration, however, this study is focusing on the application based integration, which takes place with SAS Data Integration Studio (SAS DIS), being introduced in the following section.
3.4 SAS Data Integration Studio (DIS)

This section covers the fundamentals of SAS Data Integration Studio, its purpose and capabilities.

SAS Data Integration Studio (DIS) is an ETL (Extract-Transform-Load) tool used to consolidate data from different sources, modifying it, and sending it forward to the target system. A variety of similar tools exist: SAP Business Objects Data Services, Oracle Data Integrator, IBM Cognos Data Manager and Informatica PowerCenter, to name a few. SAS is describing DIS as a visual design tool for implementing and managing data integration processes, using a variety of data sources (SAS, 2016).

DIS is utilising metadata. According to SAS, DIS is using metadata in a following way: It enables users to build data integration and automatically create and manage standardized metadata. The users are able to visualize enterprise metadata, which helps in forming the data integration process. A shared metadata environment provides consistent data definition for every kind of data. (SAS, 2016). Figure 6 illustrates the SAS Data Integration Studio environment:

*Figure 6: SAS Data Integration Studio Environment (SAS, 2016)*
Figure 6 is an illustration of the Data Integration Studio environment; it displays the main clients and servers. Administrators are using the Management Console to create and modify the metadata on the Metadata Server, which is then used by DIS.

The metadata consists of network resources, libraries, and datasets, and is saved to a repository. DIS users connect to the same server and register additional libraries and tables to the repository. After this, they create process flows (jobs), which read source tables and create target tables in the physical storage. (SAS, 2016). When functionality is created with DIS, it can be exported into bundle of metadata and created implementation. This bundle can be moved from an environment to another, without the need to recreate the metadata in the target environment.

This study examines the strengths, weaknesses and challenges of DIS in Section 5.6, however, according to SAS, the advantages of DIS are: a) SAS Data Integration Studio reduces development time by enabling rapid generation of data warehouses, data marts, and data streams. b) DIS controls the costs of data integration by supporting collaboration, code reuse, and common metadata. c) DIS increases returns on existing IT investments by providing multi-platform scalability and interoperability. d) DIS creates process flows that are reusable, easily modified, and have embedded data quality processing. The flows are self-documenting and support data lineage analysis. (SAS, 2016)

The case company is using SAS Data Integration Studio to process data from different sources such as Excel, CSV files, SAS datasets, etc. The size of the input data varies and can be considerably large, therefore performance issues are at an important role. At the case company, Data Integration Studio is most suited for more complex statistical software development using larger data masses, which are integrated and processed (modified as needed) and loaded up to a database or data warehouse.
4 Current Statistical Information System

This section presents findings from the current state analysis conducted in the case company environment regarding the current statistical information system. “Current”, in this context, signifies the new SAS architecture taken into use at the case company recently, as described in the “Introduction” section of the study. It is part of the objective of the study to evaluate this system and the tools of it. The “old architecture” (or the workstation based architecture) is covered in the “Introduction” section.

4.1 Introduction and Background

Figure 7 provides an architectural picture of the statistical information system of the case company:

As Figure 7 depicts, the current SAS architecture includes the SAS servers (UNIX AIX) running the Metadata-, Workspace- and Stored Process Servers. The client software has been installed on user workstations. The statistical data is usually located either on the SAS server as SAS data, in the relational database as database tables, or on a network drive in excel, CSV, or other format. SASDEV, SASTEST and SASTPROD represent the SAS servers. UNIX AIX operating system has been installed onto these servers, hosting the SAS environment.
As mentioned, client software, such as Data Integration Studio, Enterprise Guide or Management Console, has been installed on user workstations. These clients contact the server environment on which the development, testing or production work takes place. Exception to this is Enterprise Guide which also makes it possible to work with local SAS installed on the workstation (as is the case with Base SAS). SAS servers connect to the SQL Servers, which are hosting hundreds of relational databases. These databases contain the core of the statistical data. The connection from the SAS servers to the SQL servers is using the ODBS interface.

The material (data) description stored on the UNIX server, which is hosting the eXist XML database, contains additional information regarding the data. This adds information to the statistical data being published. Classification information is also stored on the same server. In communication between the SAS server and the eXist server, HTTP protocol is used.

The goal of a successfully practical statistical production would be that the actuary has in her or his possession the tools (server environment, DIS) and the methods, which they can use to analyse the data as necessary, despite the technical skills of the actuary. On a general level, it is not the purpose of the actuary to learn how to develop software rather than just utilising the tools offered to her / him, although some exceptions apply. This will enable the actuary to focus on the substantial matters, which are quality assurance, editing (imputing) and taking corrective actions. Another assumption as mentioned in the “Introduction” section is that the amount of support work will be decreased. This is made possible by utilising the potential of the new environment and applications to the fullest. Ways to achieve this are described later in this study.

Having said the above, it must be noted than in some cases, even the actuary is participating on the development aspect of the statistical production. The division of labour of statistical production and software development is fading. The actuary (or a dedicated person, someone on the team) might be using development methods on an ad-hoc basis to form some functionality regarding how the data is being viewed or processed. As is part of the bigger objective picture of this study, by placing emphasis on the server environment in the architectural design, the joint use of resources on statistical production is enabled. Resources in this context mean statistical data, as well as implemented data streams; they are both created and saved into the metadata. Reusable solutions, such
as common functionality, are benefits of this, as described in the “Introduction” section of the study.

**ICT Related Strategy:** The new SAS architecture complements the bigger overall ICT enterprise architecture and is in line with it. The mentioned generality and reusable solutions make it possible to diminish individuality regarding the software development and move towards universal solutions. When one solution feeds the need of many statistics, it can be described as successful software development. As also stated in the ICT strategy, it is an important factor to decrease the support work at the case company. This expectation could be met naturally when the use of the new (current) SAS architecture is enforced and advanced further into the work force. When there are shared rules of engagement and work methods, it would be effortless to follow up on someone else’s work. More information regarding universally agreed practices, support work and ways to enhance the software development in Section 5.

### 4.2 Metadata Environment and DIS

SAS describes the Metadata Server (Figure 8) as a *centralized resource pool for storing metadata* for SAS applications and other servers. The Metadata Server is the foundation; without it the whole system is non-functioning. It enables centralized control so that all users access consistent and accurate metadata. (SAS, 2016)

![Figure 8: SAS Metadata Server Environment (SAS, 2016)](image)
One Metadata Server works with all SAS applications in the environment, supporting hundreds of concurrent users. This architecture enables the following: a) Exchange of metadata between applications, enabling collaboration. b) Centralized management of metadata resources. Because of a common framework for creating, accessing, and updating metadata, it is easier to manage applications relying on it. (SAS, 2016)

SAS Metadata Server stores information about: a) Enterprise data sources and data structures which are accessed by SAS applications. b) Resources created and used by SAS applications including: report definitions, Stored Process definitions, and scheduled jobs. c) Servers that run SAS processes. d) Users, and groups of users, who use the system, and the levels of access that users and groups have to resources. (SAS, 2016)

The environment is a highly meta-driven system, using a centralized processing environment. At the case company this means development of the statistical information system and running the production in a metadata rich server environment, controllably in a systematized manner. As mentioned in the “Introduction” section, centralizing the statistical processing in the presented way, would enhance the operations so, that the development of statistics will be more efficient and the statistical production itself will become more transparent and manageable. When every Data Integration Studio developer has the same access to: metadata, SAS data on the server, database tables, libraries and other resources, and they are all working jointly on the aforementioned, the assumption is that the efficacy of statistical development is strengthened.

DIS utilises metadata and works in tandem with it. When a developer uses Data Integration Studio, she or he has to make sure that the administrator has created the necessary folder structure, libraries, etc. to the Metadata Server in order to start with the development. Also the necessary tables (database and SAS data) need to be registered and the developer needs to have a read access to them (issued by the database team).

At the case company, the above means that the statistical software development with Data Integration Studio cannot proceed until the administrator and the developer have jointly achieved to set up the environment properly. This environment is set up using the mentioned metadata objects in the server repository (creating new ones and modifying the existing). Once the environment is set up, the user creates a new job (data stream), which itself is a metadata object, and includes the registered input / output tables. After this, the data flow is formed between the tables, with needed modifications.
4.3 Starting with Data Integration Studio

*Attitude Adjustment:* As mentioned, in some cases it can be challenging for a traditional programmer to start development with Data Integration Studio. The leap to use ready-made functionalities may feel sizable, especially when manual coding is strongly accustomed to. The sentiment shared often reflects the question: “Why drag and drop boxes trying to “develop” code, instead of just coding it?” Therefore the first change one will make is the adjustment of one’s stance on accepting and using the new tool. This will be aided by informing the work force about the benefits of the change. Figure 9 illustrates the example transformations.

*Transformations:* The actual development work is started with the transformations. There is a larger variety of transformations (as Figure 9 demonstrates) in Data Integration Studio as illustrated here in the Figure above. Access and analysis transformations are there for self-explanatory reasons, as well as control, data and quality transformations. In addition to the more traditional ETL transformations, some newer transformations are in place, such as Hadoop and High Performance Analytics transformations. Data can be handled via SQL transformations and pushed into the database or manipulated as was performed in Base SAS, using the User Written transformation.

*Starting with the Workflow:* The training for DIS is usually targeting for the development to take place backwards, starting with the target. When the metadata of the target table
is created (and the source table is put in place), the developer starts working on the intermediary transformations. However, usually in practical solutions, the developer or designer does not have the end result clearly defined and will start the software development from the beginning, as Figure 10 depicts:

Sequential Development: It is encouraged to start the development sequentially, testing the functionality block by block. The data needs to be driven one transformation (or a node) at a time, as the data flow disturbances are shown imminently, instead of waiting for the completion of the whole job and then running the data. The red and white circle indicates that there are still situations to be fixed in the metadata, after which the data will pass through the flow correctly.

Keeping It Simple: One of the key issues in software development regarding Data Integration Studio, is to keep it simple and short, complexity needs to be avoided. This is not easily achieved in Base SAS, however, DIS offers better tools for this. Keeping it plain and straightforward makes the process more perceivable, as the developer is not creating it for her / himself rather than for a future developer (see Figure 11).

Workflow Tune-Up:

As shown in Figure 11, it is advisable to add an Extract transformation after each source table within a job rather than drawing the connecting straight from the source (Figure 12).
The Extract transformation (Figure 13) preserves the metadata definition allowing the developer to change source tables without breaking the succeeding mappings.

**SQL Case-When Algorithm:** The usage of logical SQL case-when algorithms and other expressions in transformations is advised. The expression language in the transformations (even those ones which are not grouped under SQL Transformations) is SQL, therefore it is possible to write SQL language as such into the expression window, as demonstrated in Figure 14 below:

**Notes and Documentation:** From the start, the developer is encouraged to use sticky notes and documentation. By keeping the jobs as short and understandable as possible, it is also easier to finish the documentation for them. One document per job would be advisable, however, it does not matter if there would be multiple documents, as long as they are clear and easy to follow. This does not mean that they cannot be detailed on a technical level; when the development is broken down so that there are many smaller jobs, it is quite natural to write the technical functionality for each individual job. The preferable way of producing the documentation is to follow the Kanban method. The concept of Kanban method is illustrated in Figure 15:
Kanban, in the context of software development, is a method for process management. Kanban is utilised also at the case company. Kanban whiteboard starts off at the case company with creating the specification. Every note coming from the backlog is a small piece of functionality turned into a proper functional description, providing a solid platform for the developer to work on. Using the Kanban model realises the creation of proper actionable documentation.

In addition to the documentation, simple sticky notes are available in Data Integration Studio. The usage of those is strongly encouraged, for example at the start of each job. It is advisable to include a note, which describes the function of the job in question, as Figure 16 illustrates:

**Debugging:** It is beneficial to realise the capabilities of Data Integration Studio debugging as early as possible. Debugging the implemented functionality is relatively straightforward with DIS. When running data through the work flow, error and warning messages are displayed by the user interface and when the “Line <number>” link is clicked, it will lead to the line of the code, which caused the error. Full code and log of the transformation, can also be viewed when clicking the links respectfully.
4.4 Data Integration Studio Transformations

This section describes statistical data processing transformations. Not all transformations of DIS are used at the case company. However, out of those who are, some are analysed here. Some basic transformations such as File Reader and Writer are disregarded.

**Library Contents:** The Library Contents transformation (Figure 17) is useful when multiple input files need to be processed through the same functionality. In this case a list of input files is first created with this transformation which are then looped through the functionality with the Loop transformation. Here “CHECKLIB” acts as a control table to the Loop transformation.

![Figure 17: Library Contents](image1)

**REST and SOAP Transformations:** These transformations (as displayed in Figure 19) enable the user to use the chosen approach (at the case company it is REST) to read and write to a third-party web service. Representational State Transfer (REST) is a set of architectural principles for designing web services that access a system's resources. A resource is accessed with a Uniform Resource Identifier (URI). The REST transformation generates SAS HTTP procedure code to read from and write to a web service in the context of a job. (SAS, 2016)

![Figure 18: REST Options](image2)

<table>
<thead>
<tr>
<th>Example Options for the REST Transformation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tab Selection Path</td>
</tr>
<tr>
<td>Options ⇒ REST Properties</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Options ⇒ Input and Output</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Options ⇒ Authentication</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

(SAS, 2016)
The above options (Figure 18) are example options when setting up the REST approach when accessing web services.

**Figure 19: REST and SOAP Transformations**

**Table Loader Transformation:** This transformation (Figure 20) writes a source table into a target table (dataset or a database table). Several loading options exist.

Load methods are: Replace, append to existing and update/insert, of which the latter takes place according to columns defined by the user. The Insert Buffer –setting needs to set to “3000” as described in Section 5.2.2, when loading data into a database table.

**Summary Statistics:** This transformation (Figure 21) creates an output table that contains descriptive statistics in tabular format, using some or all of the variables in a data set. It computes many of the same statistics that are computed by other descriptive statistical procedures such as MEANS, FREQ, and REPORT. (SAS, 2016)

When calculating summary statistics with this transformation (percentiles, other) it is advisable to use the CLASS statement (Figure 22), for it enables summing by subgrouping:
Loop Transformation: As described, the Loop transformation enables iterative processing, for example, looping of similarly structured input files through common functionality. A Control table needs to be fed to the iterative process, containing a list of tables to be processed. “CHECKLIB” acts as the control table in Figure 23. Loop transformation also realises the use of parallel processing, see Section 5.2.4.

Data Validation: Data Validation enables deduplication, checking for invalid or missing values, or to create custom validation rules. If there are erroneous situations (values) in the data, it is possible to choose the actions accordingly, for example, to stop the processing, change the value, or write the erroneous row into an error table to be processed or checked later. With custom validation, it is possible to write user written expressions to handle the data.
**User Written Code:** User Written Code transformation (Figure 25) makes it possible for the user to create custom code for situations where the ready-made transformations are not sufficient or suitable. Basically any kind of SAS code (also imported from Base SAS) can be run under this transformation. At some occasions this approach works well (when adding quick custom code) and can be used, however, the primary purpose is to use the ready-made transformations (or to create a new commonly used transformation, using the transformation generator wizard).

Two **SQL transformations** (shown in Figure 26) are described here: SQL Join and SQL Merge.

**SQL Join:** Selects sets of rows from one or more sources and combines them according to a column and user defined matches, to the target. The user is able to build statements and clauses which constitute an SQL query.

SQL Join also makes it possible to create subqueries, use parameters and enable pass-through processing. Basically it does the same as SAS merge, however, there is no need for sorting or renaming. Basic functionality with SQL Join such as where, having or order by -clauses apply, as demonstrated in Figure 27:
**SQL Merge**: SQL Merge is not to be confused with SAS Merge (as described in Section 5.2.1). SQL Merge updates existing rows and inserts new ones according to the match defined by the user. This takes place inside a database, hence the operation is for database tables only. Often when there is a need to update a database table using a dataset (flat file) as a source, the efficiency could be rather poor. When the operating is moved to take place wholly in a database, as shown in Figure 28 below, the efficacy is strengthened up considerably (from a database table to a database table).

![Figure 28: SQL Merge Transformation](image)

**Status Handling in a Transformation**: If a transformation ends in an error situation, there are several ways to handle it automatically. An email could be sent alerting about the situation, or the entry could be sent into a file or a dataset, as demonstrated in Figure 29. The processing can also be aborted. This is a proper way to ensure a needed automated alerting system for the job owner.

![Figure 29: Transformation Status Handling](image)

**User Created Transformations**: The user can create transformations using the transformation generator wizard and store them under a specific folder, as shown below. The user creates the code by him/herself for these transformations, similarly as with the User Written Code transformation. However, the difference is that these transformations can be commonly used by other developers.
5 DIS Development: Evaluation of Best Practices and Working Methods

This section discusses and evaluates the best practices and working methods with SAS Data Integration Studio (DIS). The purpose is to offer recommendations for efficient ways to perform DIS development, as described in previous sections.

In addition to presenting findings on technical solutions and enhancements to statistical software development, this section also focuses on the practical mentalities of operating with DIS. Meaning, trying to seek out common ground on universally adaptable development methods and applicable solutions.

This section does not intent to cover all the technical aspects of Data Integration Studio rather than those discovered to hold meaning regarding statistical software development.

Testing Environments
Comparing results between the workstation (old architecture: Base SAS v9.4) and server environment (new architecture: SAS DIS v4.9), following environments are at use:

- **Workstation**: Windows 7 (64 bit), Intel i5 (4-core) 2.3 GHz, 16 GB RAM
- **Server Environment**: IBM AIX Power 7 (64 bit) virtual machine: 6 CPUs / 3.3 GHz, 128 GB RAM

5.1 Data Integration Studio Settings

This section covers the rules about the Data Integration Studio settings. These settings will either help the developer or cause some harm, if not set correctly.

**Auto-Mapping**: Firstly, auto-mapping (Figure 30) is advised to be turned off:
The reason is that auto-mapping in DIS automatically maps the columns when the source and target columns have the same name, data type and length. This would not be an ideal way to proceed when there are, for example, tens of transformations in one job, some of them having custom expressions. This leads to a situation where the custom expressions will be wiped clean and the developer will have to go through all the nodes manually losing days (or more) work.

It is important to note that a developer can adjust the settings either on a global (all jobs) level, or on a job specific level, as shown in Figures 31 and 32 below. All settings for a developer are workstation specific.

**Pass-Through**: The idea behind pass-through (Figure 33) is that the data processing is conducted in a database rather than consuming the server’s resources. The total processing time with in-database processing is considerably shorter, when compared to out-database processing.

When the developer ticks the above setting, it will result in the SQL operations to take place automatically inside the SQL database, as is shown in the Figure below. The letter “Q” on the join transformation illustrates that the operation is performed in SQL database.
However, this requires that both input tables and the output table reside in a database (Figure 34):

![Figure 34: In-Database Processing](image)

By right clicking the job diagram window and choosing “Check Database Processing” (Figure 35) DIS will go through all transformations and inform if the operation can or cannot be performed in database.

![Figure 35: Check Database Processing](image)

**Join Conditions and Control Flow:** As Figure 36 illustrates, by setting the “Automatically create join conditions in Join”, it will do as suggested: This will cause the Join transformation to behave so that it will create the join condition automatically by those variables, which hold the same name and data type (length may vary).

![Figure 36: Settings - Join Conditions and Control Flow](image)

By setting the “Automatically create control flow” (Figure 36) new transformations are automatically attached as a part of the processed stream.
5.2 Data Integration Studio Development

This section examines efficient ways to perform Data Integration Studio development, placing emphasis on the performance factor (especially when it comes to running larger data masses). Findings are presented from conducted tests regarding development and performance related issues.

An overview of the system (DIS and the server environment) has been given in Sections 3.7 and 4, similarly the analysis of it and the test results (evaluation), is given in Section 6.

5.2.1 SQL Join versus SAS Merge

This section reports on processing power between SQL Join -transformation and SAS Merge. This test was conducted with relatively large input files, as shown below, which are SAS dataset files (flat files). The comparison is firstly conducted in a server environment, after which it is repeated in a workstation environment.

The test setup (Figure 37) is as follows: a test was conducted joining two SAS identical datasets (flat file) using one by-variable. The input files contained ~ 2 million rows and 165 columns. The join was performed as a full join and was done using SAS Merge (user created code) and SQL Join transformation as such (application created code). Input files are both sorted according to the by-variable.

![Figure 37: SQL Join versus SAS Merge, Server Environment](image)

The result (Figure 38) of the server environment test shows that with SAS Merge the duration time (real time) of the operation was 99 percent of SQL Join. The CPU time was 97 percent of SQL Join. Hence SAS Merge and SQL Join are practically equally efficient in the server environment.
It must be noted that the merge combines data somewhat differently when it comes to missing data. However, at this occasion the output was as expected. Also, other tests were done with other combinations of data and the advantage was marginally with SAS Merge. The point here is that the conventional way (Base SAS coding, such as SAS Merge) can be done (If so wanted) with DIS quite simply using the User Written transformation, as was now done. However, regarding supportability and readability of the implementation, SQL Join would be preferable, as it is done via the ready-made functionalities and no need for manual coding. Also, the advantage of SQL Join is that there is no need for sorting or renaming. Below is the code used in the test: Application created code (SQL Join, Figure 39), and user created code (SAS Merge, Figure 40):

```
proc sql;
    create table output2 as
    select * from input1
    full join input2
    on input1.variable = input2.variable;
quit;
```

```
data output1;
    merge input1 input2;
    by variable;
run;
```

When the same test was repeated on a workstation with Base SAS, using same input files and code, the efficiency was similarly marginally with SAS Merge. The results between the environments (workstation and server) were as shown in Table 1 (given is the duration of the operation, CPU time is in brackets):
Table 1: SAS Merge vs. SQL Join

<table>
<thead>
<tr>
<th>Environment</th>
<th>SAS Merge</th>
<th>SQL Join</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server Environment, DIS</td>
<td>10.22 (4.71) s</td>
<td>10.30 (4.88) s</td>
</tr>
<tr>
<td>Workstation, Base SAS</td>
<td>22.11 (7.16) s</td>
<td>22.20 (11.04) s</td>
</tr>
<tr>
<td>Difference*</td>
<td>53.8 %</td>
<td>53.6 %</td>
</tr>
</tbody>
</table>

According to the test, both SAS Merge and SQL Join had noticeably faster duration and CPU time in the server environment. Difference was calculated using duration.

**In-Database Processing**: The same operation as above (using the same tables, only a different name) is now performed so that the operation is pushed into a relational SQL database, see Figure 41. All tables (input and output) reside in a database and the transformation is pass-through enabled.

![Figure 41: In-Database Processing, Server Environment](image)

It is noticeable (Figure 42) that the duration increases further, however, the load on the server CPU is distinctively light:

![Figure 42: In-Database Processing Statistics, Server Environment](image)

When taking into account the full processing time of the ETL chain, processing in-database is faster than out-database, given that the data does not need to be transferred or sorted in and out of the database.

![Figure 43: Out-Database Processing, Server Environment](image)
When in-database processing is compared to a situation where the same processing takes place out-database (Figure 43), the following result is produced:

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Records</th>
<th>Start Time</th>
<th>End Time</th>
<th>Duration</th>
<th>CPU Time</th>
<th>Current Memory</th>
<th>System Memory</th>
<th>Current I/O</th>
<th>System I/O</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extract and sort - 8</td>
<td>Completed successfully</td>
<td>2000000</td>
<td>6/22/2016</td>
<td>8/22/2016</td>
<td>19,417</td>
<td>21,18</td>
<td>2922794,016</td>
<td>16969,728</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Extract and sort - 9</td>
<td>Completed successfully</td>
<td>2000000</td>
<td>6/22/2016</td>
<td>8/22/2016</td>
<td>30,696</td>
<td>21,18</td>
<td>2922794,016</td>
<td>16969,728</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SAS Merge - 10</td>
<td>Completed successfully</td>
<td>2000000</td>
<td>6/22/2016</td>
<td>8/22/2016</td>
<td>9,955</td>
<td>4,7</td>
<td>2922794,016</td>
<td>16969,728</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Table Loader - 11</td>
<td>Completed successfully</td>
<td>2000000</td>
<td>6/22/2016</td>
<td>8/22/2016</td>
<td>114,581</td>
<td>43,51</td>
<td>2922794,016</td>
<td>16969,728</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>whole job</td>
<td>Completed successfully</td>
<td>2000000</td>
<td>6/22/2016</td>
<td>8/22/2016</td>
<td>225,635</td>
<td>120,94</td>
<td>2922794,016</td>
<td>16969,728</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 44: Out-Database Processing Statistics, Server Environment

The result (Figure 44) shows that out-database processing takes considerably more real and CPU time. This is because the data needs to be extracted out of the database and sorted for SAS Merge, after which it is loaded back up to the database. Therefore, according to this test, in-database processing seems to be more performant regarding the total processing time of the ETL chain.

5.2.2 Table Loader Transformation

The Table Loader transformation (Figure 45) was tested using two methods (identified below) for loading the same amount of identical SAS dataset rows into an SQL database.

Firstly, loading 300 000 rows into an SQL database (Table Loader 1) with customised settings: Insert Buffer (INSERTBUFF): 3000. Insert Buffer specifies the number of rows in a single DBMS insert.

Secondly, loading the same identical 300 000 rows into an SQL database (Table Loader 2), default settings. Here, the INSERTBUFF parameter is not set.
The result (Figure 46) shows that the Insert Buffer setting (Table Loader 1) decreased the real time (duration) down to ~ 15 % and CPU time to ~ 50 % compared to Table Loader 2. Therefore it is strongly advisable to set the INSERTBUFF setting appropriately.

5.2.3 Summary Statistics

Summary statistics (Figure 47) were calculated out of the data and the processing was compared between the environments. Performance related issues are pointed out.

The test setup is as follows: an input file (SAS dataset) the size of one million rows (and 7 columns) is run through the summary statistics transformation, noting the duration and CPU time. Statistics is generated out of one variable according to six classifying variables. Calculated are: sum, number of observations, mean and some percentiles such as: median, Q1, Q3, P10, P90. Firstly the test is conducted using default settings of the transformation, Figure 48 shows the results:
The next step is to do the same processing with customized settings such as setting NOPRINT to “yes” (Figure 49) and Other Proc Means options to clear (remove what is shown in Figure 50).

![Figure 49: Summary Statistics Properties 1](image1)

![Figure 50: Summary Statistics Properties 2](image2)

It can be noticed (Figure 51) that the duration and CPU time are decreased considerably:

![Figure 51: Summary Statistics Transformation 2](image3)

This is because now the job does not produce unnecessary log and output information, only the needed result, the output data file, is produced.

When the same operation is performed in the workstation environment with Base SAS, using the same code as with DIS, the results (Table 2) between environments are as follows (given is the duration, CPU time is in brackets):

<table>
<thead>
<tr>
<th>Environment</th>
<th>Summary Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server Environment, DIS</td>
<td>11.52 (7.53) s</td>
</tr>
<tr>
<td>Workstation, Base SAS</td>
<td>4.41 (4.83) s</td>
</tr>
<tr>
<td><strong>Difference</strong></td>
<td><strong>61.7 %</strong></td>
</tr>
</tbody>
</table>

Here it can be noted that the performance is better on the workstation with Base SAS. However, when analysing the results it needs also to be noted that only smaller data (as with this test) could be run on a workstation. When the operation on the workstation was redone with larger input data (as usually is the case), the processing failed with an error message (not enough memory). Therefore another environment is needed to conduct
the development (other than workstation) and the new server environment offers the possibility for this.

An additional summary statistics test was conducted to showcase both ways of development, (Base SAS and DIS). The performance was about the same (1.2 second duration in both environments). However, this test is not about performance, more importantly just to show the differences between mentioned environments. An input file the size of ~300 000 rows was ran through a summary function (proc means) in Base SAS, after which the same is done with DIS and its transformations. Number of observations, mean, standard deviation, minimum and maximum was calculated for a one variable in the data. The end result was as displayed below in Table 3 (the same in both environments):

<table>
<thead>
<tr>
<th>Incomegroup</th>
<th>N Obs</th>
<th>N</th>
<th>Mean</th>
<th>Std Dev</th>
<th>Minimum</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>18025</td>
<td>12581</td>
<td>645504.96</td>
<td>8521296.55</td>
<td>-167734.53</td>
<td>302701666</td>
</tr>
<tr>
<td>2</td>
<td>30015</td>
<td>25439</td>
<td>24933395.22</td>
<td>557520642</td>
<td>-379581.10</td>
<td>2879279685</td>
</tr>
<tr>
<td>3</td>
<td>27305</td>
<td>20576</td>
<td>3566682.34</td>
<td>55607087.79</td>
<td>-1732672.14</td>
<td>2522898028</td>
</tr>
<tr>
<td>4</td>
<td>26242</td>
<td>17723</td>
<td>4913691.47</td>
<td>8466372.18</td>
<td>-99663.90</td>
<td>3627021405</td>
</tr>
<tr>
<td>5</td>
<td>25900</td>
<td>17678</td>
<td>5032697.75</td>
<td>70027071.04</td>
<td>-215472.54</td>
<td>3637885624</td>
</tr>
<tr>
<td>Y</td>
<td>33181</td>
<td>31480</td>
<td>88473399.93</td>
<td>2106985241</td>
<td>-2000150.37</td>
<td>1335748829.13</td>
</tr>
</tbody>
</table>

Table 3: Summary Statistics Output

Figure 52 below shows the code needed to be created in Base SAS (on workstation).

```
PROC MEANS DATA= lib.ok_input;
CLASS incomegroup;
VAR sum;
output out=ok_output;
run;
```

Figure 52: Code in Base SAS

The implementation in DIS (in a server environment) is as shown in Figure 53:
The implementation in DIS is created so that SAS code is created automatically on the background by the application. The code is very different than with Base SAS and considerably longer, hence it is not displayed here. The differences between environments (in development methods) are that with Base SAS the code is always uniquely written by a specific developer, so it differs between implementers. Also the code is being stored on a local computer and usually is not documented in any way. When the code gets longer, up to thousands of lines, it becomes impossible to perform follow-up development, as it is too much developer dependent.

With DIS, the implementing practicalities follow same conventions (given that jointly agreed guidelines are respected) and no manual coding is required. Work flow is logical and follow-up development by another developer straightforward. Even though the code behind the implementation can be rather complex, it is not an issue in this context. Also the workflow is self-documenting, which is easy to turn into a technical document.

5.2.4 Parallel Processing

The effects of parallel processing are examined in this section. The possibility for parallel processing comes with DIS and the server environment: it actualizes the use of more efficient CPU resources. The system administrator needs to set up the system so that there will be reasonable amount of CPUs allocated for each environment: development, test and production (according to the license agreement with the system provider).

Firstly, a small program (Figure 54), which stresses the CPU, is run through the Loop transformation as such (one process only = parallel processing not enabled*). The Loop transformation enables the use of parallel processing (see Figure 55).

![Figure 55: Loop Job (Parallel Processing)](image-url)
Secondly, the parallel processing is switched on from the properties of the Loop transformation, see Figure 56. Maximum number of concurrent processes ranged from 2 to 6, 12 and all. One process equals no parallel processing (first part of the test).

![Figure 56: Parallel Processing Configuration](image)

### Table 4: Parallel Processing Results

<table>
<thead>
<tr>
<th>Processes</th>
<th>Processing Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 *</td>
<td>88 s</td>
</tr>
<tr>
<td>2</td>
<td>63 s</td>
</tr>
<tr>
<td>3</td>
<td>34 s</td>
</tr>
<tr>
<td>4</td>
<td>34 s</td>
</tr>
<tr>
<td>5</td>
<td>34 s</td>
</tr>
<tr>
<td>6</td>
<td>34 s</td>
</tr>
<tr>
<td>12</td>
<td>34 s</td>
</tr>
<tr>
<td>All</td>
<td>34 s</td>
</tr>
</tbody>
</table>

It can be seen in the results (Table 4) that the system allows three processes to be run simultaneously. After that, (a greater number of processes run at the same time) there are no processing benefits, possibly due to some platform system restrictions. However, the server environment brings with it the possibility to conduct parallel processing and according to this test the benefits are noticeable (between sequential and parallel processing). When running three processes concurrently the processing time dropped from 88 seconds to 34 seconds, 61.4 percent.

#### 5.2.5 Proc SQL

SQL is a powerful query language which can be utilised easily in DIS with Proc SQL (Figure 57). Below are some examples of Proc SQL.

![Figure 57: SQL Execute Example](image)

Figure 57 above depicts a scenario in which the Proc SQL is written into the SQL Execute transformation (code found from below). The code enables relatively easy manipulation of data in a table, according to another table. How to delete data in this manner is presented below:
An additional SQL execute transformation test was conducted. When a many-to-many match update (from table to table) is performed, the query is optimized so that a dynamically created key will identify the rows to be updated:

```sql
proc sql noprint;
  update &_input2 as b set workamount = 0, workamountsource = 'ZZZ'
  WHERE cats(put(b.companyid,11.),put(b.year,4.)) in (select cats(put(a.companyid,11.),put(a.year,4.)) from &_input1 as a) and b.workamountsource = '444';
quit;
```

There are different methods of performing SQL operations with DIS, basically any SQL structure is acceptable. This study has emphasized the use of the ready-made transformations to perform data manipulations, however, when necessary, SQL queries (modifications) are acceptable when done in a clear way and a description of the functionality is added to the DIS job.

5.2.6 Case Example: Company Level SAS Macros

This section introduces a way to perform statistical software development as it currently stands with the case company level SAS macros, without the use of server environment and DIS. Secondly, the process of moving it to take place in the server environment is described, utilising DIS and noting the advantages / disadvantages of it.

**SAS Macros, Current Situation**

Currently, there are the case company level SAS macros used to perform the tabulation of statistical data processing. Statistical data is usually published in PC-Axis file format, which consists of keywords which can be either mandatory or optional. Keywords such
as Contents, Data, Decimals, Heading, Units and Values are used. The file consists of two blocks: data and metadata. The metadata is always described first, after which comes the data part.

*Publication* takes place when the ETL process is finished and the needed statistical output data (SAS format) exists. This takes place by turning the SAS dataset into a PC-Axis file using the tabulating macros. When the PC-Axis file is formulated correctly, it is ready to be used via all tools in the PC-Axis product family, such as: PC-Axis, PX-Win, PX-Web (used at the case company), PX-Map and PX-Edit. PX-Web is a web browser based application, which creates user interfaces automatically based on the metadata found from the PC-Axis file. If the metadata is inadequate, this translates directly into a poor user experience on the web browser interface. From PX-Web the end user is able to examine the released statistic as wanted.

*These macros* are maintained on a network drive and usually developed by a dedicated person. When the developer starts the work, she or he will take a copy of the macro and paste it into the Base SAS (or EG) to perform the needed changes. During this, a local copy of the macro is in use. After the changes are done and tested, the macro will be copied back to the network drive and placed generally available for the whole work force. Statistical production using the tabulating macros is conducted on an individual workstation locally (code example below). The user runs the macros using Base SAS or Enterprise Guide (a SAS dataset works as the input file) and the macros are producing the PC-Axis output file (table_out.px).

*Challenges* in the development work are that the work has mostly been handled by a single individual. Knowledge transfer should be performed and more developers acquainted with the macros, assuring continuous development capabilities, even during holiday seasons and sick leaves. Another issue is that there is no version control in use, hence the latest saved copy is always the only existing copy of the macro. If, a specific version of the macro needs to be saved, it needs to be manually copied under another name, indicating the version number and an identifiable string.

Likewise, there is no development or testing environment, only production. The risk is that the implemented changes are tested in production with a large audience: this has created problematic situations when the macros are not working as expected (in a real life multi user environment). When running the macros on a workstation, a challenge
might be that the limitations of the system are promptly met; the macros behave in such a fashion that the disk space (work library) is often running out. This makes it impossible to perform the tabulation, creating pressure on getting the statistics released on time.

An example SAS software which firstly retrieves metadata from eXist (an XML database containing data description and classification) is displayed below. The necessary key variables are set, in order for the tabulation macro (called on the last line) to produce correct PC-Axis output file:

*acquire the metadata from eXist and create the px-file:

```sas
%let description = http://<host>:8080/exist/rest/db/bookkeeping123.xml;
%xml_parser(&description);

Data px_meta.px_tables;
Length Matrix Charset Language Languages $ 20 Path_text $ 120 Decimals ShowDecimals 8;

*table description
Matrix      = "table_out";
Charset     = "ANSI";
Language    = "fi";
Languages   = "fisven";
Path_text   = "&egp_path\tmp\table_out.px";
Description_fi = "Description";
LAST_UPDATED = "%sysfunc(compress(2016-06-06,-)) 09:00";
SOURCE_fi   = "Source";
CONTACT_fi  = "";
COPYRIGHT   = "YES";
INFOFILE_fi = "";
NOTEX_fi    = "";
CREATION_DATE = "";
NEXT_UPDATE  = "";
PX_SERVER   = "px.server.address.fi";
DIRECTORY_PATH = "Path";
SUBJECT_AREA_fi = "tax01";
SUBJECT_CODE_fi = "tax01";
TITLE_fi    = "tax01";
CONTENTS_fi = "tax01";
UNITS_fi    = "euros";
AGGREGALLOWED = "NO";
AUTOPEN     = "";
DESCRIPTIONDEFAULT = "";
DATABASE    = "";

*SAS description for creating the px-table
SumVar      = "information";
Variables   = "value c_value change c_change portion num";
TimeValVar  = "";
Rows        = "variable information";
Columns     = "industry tax year month";
Clao        = "";
Datafile    = "lib.table_out";
Run;

*create tabulation
%do_px_ml (table_out);
```
As mentioned, the above code is run in Base SAS or EG, after which the PC-Axis file is produced on the user’s local hard drive. This file needs to be transferred to the PX-Web Windows server via FTP, manually or timed. Note that the above code is not the tabulating macro, it is only called via this program (the last line). When Base SAS or EG is started, the macros are loaded automatically into the current environment, from the network drive where they are stored. This enables them to be used so that they are called by their name only.

**SAS Macros, Server Environment**

An experiment was conducted where the tabulating macros were transferred into the server environment. This section describes the required steps and what are the noted advantages / disadvantages of performing the development and running the production in the server environment.

Firstly, the tabulating macros themselves needed to be ported into a UNIX environment. This meant changing some used syntaxes, for example folder paths, into a UNIX format. Windows environment is using backslashes in paths instead of forward slashes, which are to be used with UNIX. Some other functionality was also changed, which dealt with accessing network drives; this was changed so that the server drives were accessed instead. The macros were transferred into a common location within the server.

The following step was to write a similar code as above, sets the scene for the tabulating macros. The code was placed into the node number one (first transformation after the input data) in Figure 58. The code does the same as with Base SAS processing: retrieving metadata from eXist and performing the tabulation.

![Figure 58: The Macro Job in DIS](image-url)
The specified key variables are written to a separate parameter table called “PX_Tables”, after which the table is transposed and a control table is created out of it. If there is a need to make changes to the variables, the control table can be edited. However, after this it is transposed back into original position and loaded back up to the parameter table. Now, it is possible to redo the tabulating with the updated parameter table (re-valued key variables) by running node number five.

The created job above was refined (Figure 59) so, that a logical work flow was formed ending with a data transfer transformation. This enables running the processing wholly from start to finish without any intermediary phases in DIS (assuming there is no need to make any changes to the control table). The data transfer step at the end included user written code which transferred the created PC-Axis file to the PX-Web server.

![Figure 59: The Macro Job in DIS, Refined](image)

The conducted test acts as a showcase example on how the tabulating procedure can be moved from a workstation to the server environment. This enables appropriate development process cycle, including the use of version control and proper operating environment for each distinctive phase: development, test and production (eliminating the need to test in production). Additionally, the server environment realises the use of a venue which offers the capabilities regarding resources such as: disk space, CPU power, etc. When a Stored Process is created out of the above functionality (and the preceding ETL processing), it can be used as a web service by calling it from a web browser (see Section 5.3). This would enable “one click” processing from data to statistics, all the way to the PX-Web server. This would be an “ideal” situation regarding statistical production. However, often a need exists to edit and adjust the data before the statistic is released, requiring manual intervention.
Generally the development of the statistical software is started on the server environment with DIS (the normal ETL process). However, when the need to perform tabulation (create PC-Axis files for the publication) becomes present, the files need to be transferred from the server environment to a local workstation. After this, the files are run through the tabulation macros using Base SAS or Enterprise Guide. Finally, they are transferred via FTP to the PX-Web server manually or on a timer. This causes some unnecessary manual labour when compared to running the whole processing, from start to finish, on the server side. No distinctive disadvantages were noted during the test.

5.3 Service Oriented Architecture (SOA) at the Case Company

*Service Oriented Architecture* is an architectural style in software design and development, which is based on the use of services: small iterative functions of daily business activities. Services perform one action (usually) per service, examples are: validating a user, storing a user record, loading up a report on a web browser. Figure 60 illustrates the operational concept of SOA.

![Figure 60: Operational Concept of SOA](image)

The services are independent (unassociated) and reusable, the technology used to create the service is hidden from the user. The user sees a simple interface through which the service is used, for example a web page.

*The usage of SOA in software development at the case company:* While the service oriented architecture has already been examined for some time at the case company,
the actual practical implementation of it has lagged behind. Recently, some implementations have been realised. Services, especially *web services* are implemented by using REST (Representational State Transfer) at the case company to perform actions within the statistical production. An example of web services is the invoking of macros through the use of *Stored Processes* (STP) on a server. This Stored Process acts as the web service which is called upon by the web browser, through the REST interface.

According to the GSBPM model, the aim is to align the enterprise architectures of different organisations, creating an “industry architecture” for the whole statistics industry. The result is the *Common Statistical Production Architecture* (CSPA), first released at the end of 2013. (GSBPM, 2013). The usage of Service Oriented Architecture is advanced throughout these institutions by CSPA, which is promoting the sharing of the commonly implemented functionality (created with SOA), between government agencies.

*Advancement of SOA by the use of Data Integration Studio:* SAS Data Integration Studio can be used to implement Stored Processes which are used by the web browser to provide services for the user. In the context of this study, statistical production can be aided via the use of web services, for example cleaning and verifying the data and loading it up to the database. Web services in this context mean SAS services which are created with DIS and executed in SAS environment, and which can be used device independently, utilising the HTTP protocol.

“By utilising the service oriented architecture (SOA) the objective is that each information system project produces at least one common service (web service) that is defined as a task of the project already when the project is established.” (The Case Company, 2016)

In practice, SAS service will be developed using the SAS BI Web Services software (utilised through DIS), which can represent the Stored Processes as HTTP services, without additional configuring. SAS BI Web Services only support GET and POST HTTP calls. If parameters will be transmitted to the Stored Process, then POST is used. With GET it is possible to call a Stored Process, which does not have any parameters defined or parameters are not mandatory. The service can be called using JSON or XML syntax. If the parameters and results of the service are simple, either one can be used, however, complex syntaxes require XML syntax. When the service is started using the POST method, the service (HTTP) header will have to specify the content type for the parameters. When JSON is used also the result type needs to be specified (see Table 5):
Table 5: Service Configuration

<table>
<thead>
<tr>
<th>Service Syntax</th>
<th>Content-Type</th>
<th>Accept</th>
</tr>
</thead>
<tbody>
<tr>
<td>XML</td>
<td>application/xml</td>
<td>application/xml</td>
</tr>
<tr>
<td>JSON</td>
<td>application/x-www-form-urlencoded</td>
<td>application/json</td>
</tr>
</tbody>
</table>

The syntax used in the service call also affects the URL of the service (see Table 6):

Table 6: URL of the Service

<table>
<thead>
<tr>
<th>Service Syntax</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>XML</td>
<td>&lt;host&gt;/SASBIWS/rest/storedProcesses</td>
</tr>
<tr>
<td>JSON</td>
<td>&lt;host&gt;/SASBIWS/json/storedProcesses</td>
</tr>
</tbody>
</table>

Examples of SOA (running Stored Processes as a service):

When the Stored Process is saved onto a hard drive, it can be used via the REST interface as described previously, using a web browser. An example of a code which calls the Stored Process “check_file”, is displayed below. Normally the Stored Process is called from the web browser, however, it can be simulated (tested) as done below, using Proc STP:

```plaintext
libname test_load meta library=folder (test_load) metaout=data;

%let rc=;
%let load_id=;
%let log=;

Proc stp program="/Stats/Stat/program/test/check_file";
  inputparam data="data_25JUN16";
  inputparam system="LJ_VT2016";
  inputparam SASENV;
  outputparam rc load_id log;
run;

%put stp reply: &=rc &=load_id &=log;
```

The Stored Process itself can simply be a piece of a code which handles input / output parameters calling up additional data handling macros. The below example Stored Process calls up two additional macros for handling data: load_check and load_code2id:
Example stored process "check_file":

*input parameters:
%global data system check_combo;
*output parameters:
%global rc load_id log_file;
%macro check_file;
  %load_check(&data, _checked, &system, ccheck=on);
  %if &error ne 0 %then %do;
  %let rc=&error;
  % goto end_now;
  %end;
  %load_code2id(_checked,_load.&scoreboard,&system);
  %if &error ne 0 %then %do;
  %let rc=&error;
  % goto end_now;
  %end;
%end:
%mend check_file;
%check_file;

Here, the user would choose a data file from the web browser (below) and submit it to a check by clicking “Check File”. This calls the Stored Process on the server, which again executes the necessary macros for checking the data. The output is a checked data file, which can be loaded up to the database by using another Stored Process executed when the corresponding button is clicked: “Upload to database” (see Figure 61):

![Data Process Application](attachment:image)

**Figure 61: Web (Browser) Based Data Processing Application**
As demonstrated in Figure 61, the Stored Processes may contain embedded data processing macros, which are commonly available on the server, thus advancing the general applicability (reusability).

The alternative would be, and has been, to run the implemented functionality directly from EG or DIS (or from Base SAS, when on workstation environment). It has been possible also to create Stored Processes and run them through a .net application or manually by Enterprise Guide or Data Integration Studio. However, by creating web-based applications, it will be possible to create browser-based portals, from which the statistical production (multiple statistics) can be conducted, in a centralized manner.

The Hello World –example:

Hello World is a simple Stored Process, which is producing streaming output (creating one web page saying “Hello”, which is sent back to the browser):

```sas
/* simply write out a web page that says "Hello World!" */
data _null_;  
file _webout;  
put '<HTML>';  
put '<HEAD><TITLE>Hello World!</TITLE></HEAD>';  
put '<BODY>';  
put '<H1>Hello World!</H1>';  
put '</H1>';  
put '</BODY>';  
put '</HTML>';  
run;
```

The GET call can be made as follows using the server command line interface. The environment parameter $USER holds the SAS user id.

```
$ curl -G -k -u $USER -o result.xml \  
https://<host>/SASBIWS/rest/storedProcesses/Products/SAS Intelligence Platform/Samples/Hello World
```

When a Stored Process is created, one must describe any output that is returned to the client. There are four types of client output: None, Streaming, Transient package and Permanent package. In this context the Stored Process is producing streaming output:
XML document (or HTML page) is delivered as streaming output to the client. The XML output is placed into a temporary file reference "_WEBOUT", which is base64 encoded:

```xml
<sampleHelloWorldResponse>
  <sampleHelloWorldResult>
    <Streams>
      <WEBOUT content-Type="text/html; charset=windows-1252">
        <Value>
          PEhUTUw+CjxIRUFEPjxUSVRMRT5IZWxsbyBXb3JsZCE8L1RJVEEX-FPjwSEVBRD4KPEJPRFk+CjxIMT5IZWxsbyBXb3JsZCE8L0gxPgo8L0JPRFk+CjwvSFRNTD4K
        </Value>
      </WEBOUT>
    </Streams>
  </sampleHelloWorldResult>
</sampleHelloWorldResponse>
```

When the "_WEBOUT" is pointed at directly as follows:

```bash
$ curl -G -k -u $TKSAS_USER -o hello.html \ https://<host>/SASBIWS/rest/storedProcesses/Products/SAS Intelligence Platform/Samples/Hello World/dataTargets/_WEBOUT
```

The file hello.html is produced and is now containing the “Hello World” HTML:

```html
<HTML>
  <HEAD><TITLE>Hello World!</TITLE></HEAD>
  <BODY>
    <H1>Hello World!</H1>
  </BODY>
</HTML>
```

The above is an example how to stream information back to the web browser. Any data written to the "_WEBOUT" file reference is streamed back to the client application automatically. What needs to be remembered is that streaming output is supported only on the Stored Process Server. Stored Processes executed on the Workspace Server, cannot use streaming output.

5.4 Strengths, Weaknesses and Challenges of DIS

Findings on strengths, weaknesses and challenges of Data Integration Studio are presented in this section. Below are the strengths:

SAS Administrator can set user- and group level access as is chosen, providing access for example only to one statistic or groups of statistics. Access Control Templates (ACTs) can be used with the imported/exported DIS packages, retaining the associations between the user and metadata objects. This eliminates the need to repeatedly set the
same explicit permissions for the same identities on multiple objects. DIS offers an automated back-up system, which secures the DIS used metadata server environment. It can be backed up automatically via scheduling, without the need for administering the system. In addition to importing/exporting DIS packages, an ability exists to import SAS code: Old SAS code can be transferred (as SAS files or copy-pasted) into the new environment and complement / amend it with new functionality. The SAS code can be turned into a job automatically, which can be scheduled via Management Console or SAS Platform Scheduler. After a successful execution of a job, an alert can be sent (for example an email), notifying the completion of the job.

Base SAS has over 200 functions for handling numeric, character or date formats, allowing complex manipulations of data. These are all available in Data Integration Studio, due to it having a built-in access to SAS functions. With DIS, a Base SAS developer can do the same with manual coding if so chosen. However, it is the intention to use the ready-made functionalities (transformations). A possibility also exists to push data processing into the database, using SQL pass-through. Pass-through (Figure 62) enables the data processing to be conducted in a database rather than on the server side:

Using a metadata rich environment is an asset to be noted: Every object used in DIS is a metadata object, which means having an ID attached to it which uniquely identifies it (see Figure 63).

**Figure 62: Database Processing**

**Figure 63: Metadata**
Figure 63 above shows that the object holds a unique ID assigned to it (circled in red), through which the object is used and identified. The metadata also offers a one click access to the content of the table, displaying the characteristics of it, as shown below in Figure 64:

![Figure 64: Automatic Table Contents]

Data Integration Studio realises the possibility to create Stored Processes and run them on a Workspace- or a Stored Process Server and use sessions. Stored processes enable the possibility to implement functionality according to Service Oriented Architecture (SOA), and create services for a web page (web services), as shown in Figure 65:

![Figure 65: Web Services]

With DIS, rapid development processes, such as fast creation of tables, are actualized. One can use a template from an existing table and promptly create the new needed tables. Also, the supportability of the implemented functionality (the data flow) is clear and easily supported. For a new developer this realises an effortlessly follow-up on another person’s implementation. DIS makes it possible to use the resources intelligently: Implementations are transparent and evident to every developer, as they reside on a common Metadata Server, enhancing the software development.
Parallel processing (Figure 66) has been brought to the attention of the developer with the server environment. It is an efficient way to perform the development regarding large data processing. (System administrator needs to be aware when setting up parallel processing and running larger data masses.)

![Figure 66: Parallel Processing](image)

A flow chart like data stream capabilities in the user interface provide a visual and informative view (Figure 67), especially compared to Base SAS coding:

![Figure 67: Work Flow](image)

The system is collecting performance statistics out of the data stream (job) and transformation and displaying it in a clear tabular format. Especially useful (when processing larger amounts of data) is the memory and CPU related information. With this information the DIS developer or designer can identify the bottlenecks in a job regarding hardware resource sensitive data streams. Example statistics are illustrated in Figure 68:
Impact and reverse impact analysis (not depicted here) show the developer, for example, which table is used in which job. This is a usable tool also for keeping the metadata environment clean: for example removing unneeded tables not used by any job, or for tracking the usage of different metadata objects.

Weaknesses of Data Integration Studio are viewed below:

Data Integration Studio is developed with Java, and one of the findings of examining memory consumption is that the garbage collection is still the bottle neck regarding memory handling (in Java related products): At least 16 gigabytes of memory is needed for the DIS developer’s workstation, when processing larger amounts of data or running multiple DIS clients simultaneously.

Regarding other weaknesses: DIS still has a notable amount of bugs in its system (at the time of the writing). For example copy-paste not working as one would expect. Additionally, DIS is a client-server application and is a part of a larger package: Installation of it is not the easiest task to perform. In development work, DIS developer always needs a metadata administrator to work with in tandem. Developing with Data Integration Studio is not a solo operation; therefore ensuring smooth team work is essential.

Challenges of introducing Data Integration Studio into the work community:

One of the major challenges is the natural resistance to change one’s working methods. Individual resistance derives from universal fears; fear of new and something unknown. If it is not known that one can actually perform the work, and as easily as before, it is not desirable to commence it. If the work has been performed in a similar and well known way, for a longer time, the developer might just want to perform as she or he has always done, not taking “an extra tool to make the work difficult”.

Figure 68: Performance Statistics
DIS might be seen as a slowdown or a punishment as the true purpose (as described earlier) remains clouded. Hence, it is important, through class room studies and infomercials to educate the work force regarding the benefits of the chosen tool. One challenge is the code DIS generates; it could be seen as complex, hard to interpret. However, the purpose is not to change the application code over the use of ready-made functionalities (transformations). However, if so wanted, DIS is realising also the possibility to write personal code with a user written transformation, although as noted, this is preferable only in a situation which cannot be handled via the conventional transformations.

5.5 Working Methods, Project Work and Documentation

*Unified development methods*, including documentation, lead to effortless supportability (continuity of work). This study has introduced a set of common ways of conducting the statistical software development. As mentioned, the created functionality may not be dependent on a specific developer, expert dependency needs to be avoided. Thus, clear and logical process flows (small), as shown in the examples, are to be utilised. Following the presented guidelines in this study should result into a decreased amount of support work.

“In the project to combine the architecture of small statistics, information systems are unified based on SAS architecture so that more standardized system structure, implementation technology and work processes are reached.” (The Case Company, 2016)

As described in the ICT strategic plan of the case company, the architecture work relies on the GSBPM (Section 3.1). The case company’s ICT strategy is a part of the enterprise architecture, which again is derived from the government administration. GSBPM and especially the data processing part of it, guide the software development operations at the case company. By following the generic process models (such as GSBPM), more common operations and quality in development work can be ensured.

The goal at the case company, regarding data warehousing, should be to aim for a uniform and easy-to-use data warehouse, divided to logical ensembles. Data warehouses should utilise metadata excessively (to create visibility) and be technologically more available, through joint use of interfaces. Created datasets should be topic area based, enabling construction of larger information systems (however more perceivable and logically comprehensible).
The service oriented architecture should be promoted within the case company, for its active utilisation will create more common and reusable solutions and working methods (as demonstrated in Section 5.3). The web services are to be developed so that they would be device independent: creating solutions enabling different purposes and usage methods of data. The end result of this would be dynamically constructed thematic pages meeting the user’s custom needs.

_The project work_ at the case company follows the Lean model. Usually Scrum or Kanban is utilised. This study has been focusing on Kanban. The work is broken down to smaller iterations which all are thoroughly visible to everyone. In each phase work load limits are used, making it possible to detect upcoming bottlenecks in development work. The project work at the case company should be enhanced by hiring / training professional project managers, thus aiming for improving time management (holding the set timetable). The extension for the timetable (if needed) should be limited to one or two months after which more extension is granted only with proper arguments. This is due to the fact that recently some overlapping projects have existed, which has resulted into resources being tied down (not freed for other work), especially from ICT management.

Projects should incorporate one-to-one discussions and middle- and final evaluations regarding the proceedings of the project, learning from the gathered information. Also, the project work should be defined so that the project plan is strictly monitored and followed; often projects start to lose substance and focus shift. It is also often noticed that a project will inherit other smaller off-spring projects which are not in scope or something is taken off the agenda. This kind of vague planning and clouded focus should be eliminated and a firm (realistic) project plan formulated. Also, the project work and production work are mixing up: The running production work is often halting the project work, which should be prevented. Individuals need to be freed more clearly from traditional production work for the duration of the project. One way to achieve this is to advance the use of Lean methods such as Kanban, holding more intensive sprint-like project work periods.

For the above-mentioned reason the project schedule is prolonging, and work effort put into the project work not sufficient. This in many cases leads to inefficiency.

“The efficiency of development work is increased by developing project work. This happens, for example, by using agile methods and increasing professionalism in project work.” (The Case Company, 2016)
The case company's ICT Management architects should participate in all statistical information system projects from early on. It should be one of the architect's priority tasks to place emphasis and put the ICT related strategy into practice. In addition to the qualified project manager, the architect is the conduit between Company’s ICT strategy and the implementing factor.

“Utilisation and building of common solutions, and the compliance of information systems with the architecture are ensured through the Information Technology Department's architects participating in all information system projects already starting at the planning stage. The aim is to develop topic area specific statistical systems and reduce the share of solutions tailored for individual statistics.” (The Case Company, 2016)

Documentation has an important role also in statistical software development. The Kanban model brings with it a documentation system which emphasizes the creation of smaller (however technical) documentation. Thus, it would be more perceivable for the developer to grasp the idea behind the functionality and create the implementation. One documentation per DIS job would be preferable. When the documentation is stored on a network drive and the designed software is located on the centralized metadata environment, it realises general applicability (use of shared resources) for DIS development.

It should be made clear that there exists a “demand” from ICT Management towards customers (statistical departments) for a specific level of quality regarding requirement documentation. The statistical departments are the substance experts in their own field (topic area) and need to express clearly what is required from the statistical software, in order for the ICT Management to implement it efficiently. More desirable would be to create documentation on a topic level (regarding statistics) within the case company so that more general and reusable statistical systems would be born. The documentation may cover a wider area of functionality, for example several statistics under a common topic. However, it needs to be divided into clear sub-documentation, in order to keep the viability of development.

The design documentation (or requirement) which reflects the used methodology (of DIS development), should be a type of documentation, which describes what will be accessed, defines formats and lays out the rules for expected (target) values, specifying how the data will be moved or transformed. The target and its structure should be clearly defined. This type of a design documentation is needed to ensure smooth operations for everyone involved. When the requirement documentation is finalised, it needs to be com-
plete as described above. However, it can be dynamically changed under certain circum-
stances: Given that often a need might arise during the implementation phase that the
requirements need to be clarified. This is something which is unforeseen for the re-
quester. In this case the documentation can be revisited.

Generally, the risks at the case company regarding improper statistical documentation
are inefficiency at development and support work, such as unnecessary/faulty changes
to the statistical software implementation. The starting point for good documenting prac-
tise would be to store documentation in a common and easily accessible location, also:

- Each statistics division should have a responsible person dealing with documen-
tation regarding their statistical production.
- A plan should be drafted and followed (with deadlines) by divisions, for how to
  bring documentation up-to-date.
- Proper documentation templates should be formulated (and kept updated), which
  are to be used when creating specification.
6 Conclusions and Recommendations

This section formulates the conclusions and recommendations of the study, containing a description of the feedback / iteration cycle (covering the conducted process), summary, outcome vs. objectives and evaluation.

6.1 Feedback Process

Firstly, the feedback process is described: covering the methods of acquiring information from the case company regarding the findings of the study, and analysing it.

Stakeholders from the case company were involved from the start, in defining the scope and objective of the study. Stakeholders included IT managers, architects and other colleagues. The first communication with the stakeholders was a round of one-to-one interviews, based on a series of questions, which covered scoping out the technology / business problem in the case company. The questions asked were: What kind of problem(s) the case company is interested in solving, why it is important to the company, what is the current situation at the moment (regarding the problem, i.e. the starting point), and how the company sees it should be developed. Additionally, the case company was asked to identify the specific elements (issues) in the problem area to be developed, and what were the requirements for the outcome / solution. The above lead to defining the scope, and forming the research question and objective, as presented in this study.

The communication dialogue was kept open throughout the whole procedure, as ad hoc interviews and workshops were conducted and test result validated iteratively through the pool of stakeholders, especially technical experts. During the process, valuable feedback was obtained which was fed back into this study. Information was also received through email and instant messaging. The answer from a member of the stakeholder group was validated against other members by inquiring further thoughts on the same matter, especially for test results in Section 5 (however not limited to this). When the feedback was targeted at the conducted tests, it was analysed by re-running the tests and verifying the validity of it. When found to be accurate, the feedback was entered into the study and taken notice of.
As described in the “Introduction” section, the evaluation methods of this study aimed at both qualitative and quantitative level: The test results in Section 5 are of quantitative nature, when other results are more qualitative. The evaluation was performed jointly with the stakeholder group during follow-up meetings and based on the received input. An assessment of worth was reached based on the evaluation, which is brought forth further on in this section.

6.2 Summary

The research question was: How to make the statistical software development more efficient with SAS Data Integration Studio? The objective of this study was tied to the research question and was: To evaluate if and how it is possible to enhance the statistical software development and support in the case company, using Data Integration Studio (DIS). The assumption was that centralizing the statistical processing will enhance the operations so that the development of statistics will be more efficient and the statistical production itself will become more transparent and manageable.

The current context of ETL world was introduced as base information for the analysis of the ETL and SAS Data Integration Studio (DIS) processing at the case company. This acted as the literature review into the existing knowledge. At this conjunction the Global Statistical Business Process Model (GSBPM) was introduced, which is a vital part of statistical processing and followed also by the case company. The current statistical information system was described and compared with the previous way of conducting statistical operations. Section 5 (DIS Development) described and evaluated best practice and working methods with SAS Data Integration Studio. The purpose was to find out and offer recommendations for efficient ways to perform DIS development, as described in the “Introduction” section. In addition to finding technical solutions and enhancements to statistical software development, Section 5 focused on the practical mentalities of operating with DIS: trying to seek out common ground on universally adaptable development methods and applicable solutions.

This study has introduced a series of best practices and working methods with DIS, which have been discovered through a two year period of examining the application itself, and how it could be utilised to the fullest potential. The technical aspects cover ways to perform operations the most efficient way in the server environment and compared
against the workstation based environment. The aim was to find the most performant yet supportable ways of creating logical and comprehensible implementations.

6.3 Outcome vs. Objectives

As mentioned, the objective of this study was to evaluate if and how it is possible to enhance the statistical software development and support in the case company using Data Integration Studio (DIS). This included project work, unified ways of working and documentation. This study is the outcome of this work, including best practice, working methods, performance related matters and productivity issues regarding development work in general.

According to the findings and test results in this study, it can be summed up that the research question was answered by the outcome, likewise the objective, as described above, was reached.

6.4 Evaluation

As mentioned, the evaluation of the study is conducted based on both quantitative (Section 5) and qualitative approach. The findings and test results of this study were evaluated with relevant stakeholders and statistical / ICT experts of the case company, in which point feedback and improvement proposals were also gathered. This was iteratively fed back into the study. The evaluation rounds helped to form a coherent picture of the statistical information system at the case company and conducting Data Integration Studio development.

It can be summed up that based on the evaluations, the presented results can be considered relevant. It is a notation supported also by the stakeholder group, especially the test results in Section 5 favour the use of the new server environment. These results prove in the majority of cases that the new environment is more performant than the old one. However, also more capable of offering more versatile development methods, such as service oriented architecture or creating general - commonly usable - functionality out of individually created code. This study also suggested changes to the working methods, project work and documentation, which were seen by the stakeholder group to be in line with the company ICT strategy. The study has offered a series of best practises which were similarly evaluated as described above: they are found by the stakeholder group to
be noteworthy and valid, proving on their behalf that there exists potential with the new server environment and Data Integration Studio which should be harnessed more vigorously. This should be done by increasing the usage of DIS according to the presented guidelines and best practices.

**Strengths, Weaknesses and Challenges**

When evaluating the strengths, weaknesses and challenges it can be concluded that the advantages outweigh the disadvantages of Data Integration Studio greatly. Regarding the larger learning curve, it is not an obstacle given that the productivity will increase noticeably after the initial use.

**Initial Phase and Settings**

This study also offered ways to handle the initial phase (transfer from old to new) when starting DIS development and accepting it as established element in the current system. In this phase the practical mentalities of operating with DIS were examined. Also, some application (DIS) specific settings were covered. This was seen by the stakeholder group as a helpful way for a new developer to get acquainted with the new system.

**Development and Performance**

When evaluating the development and performance related test results and findings (Section 5.2), valuable feedback was obtained from colleagues. Conducted results and created program examples were evaluated by the stakeholder group and found to be effective on statistical software development and production. This feedback was fed iteratively back into the study. Although this section addressed also performance issues, the productivity in general was reviewed here, such as using SQL language in queries (Proc SQL), SAS Merge and SQL Join, or utilising in-database processing. The quantitative evaluation of the test results by the stakeholders revealed that, the server environment seems to be clearly more performant (compared to the workstation based environment) regarding efficiency of processing, especially when dealing with in-database processing, however, also in other tasks. Some examples exist which seem to indicate the workstation based environment is more performant, such as calculating summary statistics. However, an explanation was given why these results should be examined critically. On a generalized level, according to the evaluation, the benefits of utilising the server environment more actively (through DIS) outweigh clearly the disadvantages, which were found to be practically marginal.
The question is not only about performance issues, but also the bigger picture of the development environment and what it brings with it. The new server based environment realises common usability of the development methods. As described in the “Introduction” section, the statistical software will be developed and maintained using metadata on a common server, to which every developer has access. When the programs are as generic as possible, well documented, and not dependent on a specific developer, the amount of support work will be decreased. This means that a new developer is capable of conducting follow-up development relatively effortlessly. Efficiency of development will be guaranteed with the new server environment.

An explanation was given on how development methods differ between environments (Base SAS and DIS) and it can be concluded that the traditional way of coding is too volatile (developer dependent) as opposed to DIS’s way of creating commonly agreed work flows. It was mentioned that when the storage place for the code is transferred from a personal computer to the server environment, the continuity of the development work is secured. When the created implementation is documented as described in this study, supportability is further enhanced. More advanced development methods were described also in Section 5.2.4 regarding parallel processing. This was seen to offer a noticeable boost to efficiency when dealing with code which distinctively placed stress on the CPU.

Case Example: Company Level SAS Macros
The conducted experiment, transferring the tabulating macros into the server environment and developing the functionality to utilise them, is supporting the assumption and starting premise, as set in the “Introduction” section. When the functionality is turned into a Stored Process, it can be used as a web service by calling it from a web browser, enabling efficient processing from data to statistics. Noting of course that this would be an ideal situation regarding statistical production, for often a need exists to edit the data before the statistic is released, forcing manual intervention. By making the macros available in the server environment, not only the statistical development and support, but also production (running statistical production) are strengthened regarding the operational efficiency.

Service Oriented Architecture (SOA) and Reusability
It is part of the bigger objective to find ways for utilisation of Service Oriented Architecture at the case company. The question needed to be answered was: How can DIS advance
the case company’s strategy regarding the Service Oriented Architecture? This question was answered with concrete examples in Section 5.3. It can be concluded that services, especially web services, are a viable way to conduct statistical software development and production. The use of web services is encouraged, for it fulfils on its behalf the reusability aspect of the developed implementation. Another way to fulfil the objective, are user created transformations and placing them generally available.

Working Methods, Project Work and Documentation
Ways to enhance working methods, project work and documentation were recommended in this study. These recommendations were evaluated by the stakeholders attached to the study and found to be proficient, therefore the study is recommending the realisation of the suggested changes.

The use of SAS Data Integration Studio has shown to be beneficial in terms of statistical software development within this study. It is the advisable action that more active usage of DIS would be promoted and its purpose increased.

6.5 Future Steps

During the evaluation of the study some issues were marked for further studies. These included: Presenting further findings of “best practice” and “working methods”, more detailed performance related issues (efficient data processing etc.) and show cases of example statistics (conducting software development from start to finish for a specific statistic). Additionally, statistical software development can be orchestrated so that similarly structured statistics could be combined under a specific topic area. Extended examples of service oriented architecture with real life implementations is seen to be of value, especially the use of web services and their utilisation in statistical production.

Other issues to be covered would be version control (how to work with different operational environments) and the meaning of a suitable environment at every work phase (development/testing/production). The support process and change management was left out of the scope of this study. However, this is something which could be covered in order to ensure the supportability aspect of operations. The presented actions in this study - the ways for decreasing the amount of support work - need a proper actionable plan on how to formulate the support process into something which is built on solid and found-effective practices.
References

[Accessed 19 04 2016].

[Accessed 20 05 2016].

[Accessed 13 03 2016].

[Accessed 10 02 2016].

[Accessed 13 03 2016].


[Accessed 06 04 2016].

[Accessed 09 02 2016].


