Comparing different approaches for retrieving and visualizing big data

University of Oulu
Information Processing Science
Master's Thesis
Anniina Salow
04.03.2024

Abstract

Big data is a huge part of companies' success nowadays and therefore the big data needs to be retrieved and analysed to benefit from it. Big data has taken a remarkable role in business and therefore it is important that companies put effort into their big data analyses. With the data analyses companies can follow data trends in different areas, and notice increasing or decreasing trends quickly. These allow the company to act prior and predict what will be happening in the future. Therefore, companies can act on upcoming changes beforehand.

For the data retrieval and analysis, there are multiple ways to execute that. In this research, there are two methods that are considered for data retrieval for the purpose of data analysis. This research will build missing management reports to bring even more benefit from the big data. The two methods will be covered in this research and consider their possibilities and limitations. Elasticsearch, APIs and Python are at the key position when investigating and building data retrieval for the management reports. As Elasticsearch, APIs and Python are for the data retrieval the data also needs to be visualized to create understandable management reports. The visualization will be done in Power BI to visualize the data in graphs and to create useful filters for the users to limit the graphs as they want them to. This research is done following the Design Science Research (DSR) research method.

Keywords
Elasticsearch, Big data, Data retrieval, Index, API, Visualization, Design Science Research.

Supervisor
PhD, University Lecturer Dorina Rajanen

Foreword

In today's increasingly data-driven world, the importance of big data and data analytics cannot be overstated. It is important for companies to realise the benefits of the data they produce. This thesis researches the best method to retrieve, analyse, and visualize data for management report purposes.

My decision to explore the topic was sparked by my interest in learning about interesting aspects of handling big data and learning to use new analytical tools. The topic was interesting from the beginning, but it got even more fascinating as the research went forward.

I am deeply grateful for my supervisors Dorina Rajanen from Oulu University and Hannu Kaarela from Nokia, whose expertise and guidance have been extremely valuable. I want to also thank my family and friends who supported and encouraged me during this journey.

Oulu, 08.02.2024

Anniina Salow

Contents

1. Introduction

Today's world produces countless amounts of data every second (Akdogan, 2015). The amount of data that is produced is so extremely large that we cannot handle it by ourselves anymore. This ocean of data that is produced every second is called big data, which is largely unstructured and scattered (Akdogan, 2015). The concept of big data is used when the amounts of data are so massive that it faces difficulties with storage and further processes. According to Sagiroglu and Sinanc (2013), "big data is a term for massive data sets having large, more varied and complex structure with the difficulties of storing, analysing and visualizing for further processes or results." The growing amount of data is described as an explosion of data by Kaisler & others (2013), which has led to new challenges in processing it. Just for our personal computer in 2008, there was only a need for ~10 000 gigabytes of storage room, but fast forwarding only five years to 2013 the need was raised to ~10 000 terabytes (Kaisler et al. 2013). These numbers are only for personal computers for personal use, the amount of storage space that a large global company could need is beyond possible to have reasonable storage for all their data and to be able to search and analyse the data efficiently.

When dealing with this ocean of scattered data the manual searching and finding is not good enough anymore. The disk-based data storage and search cannot offer efficient enough responses due to high latency (Zhang et al., 2015). The processing pace needs to be way faster when getting real-time data analytics, and the fast pace is not efficient enough, the processing phase should support so-called ultra-low latency service (Zhang et al., 2015). For this reason, there has been an increasing need for search engines that are efficient, fast, and significant for accessing, recording, processing, and analysing data fast and effortlessly from the users' point of view (Akdogan, 2015). To meet these requirements and find solutions to these issues of handling big data, Elasticsearch has been developed to answer these needs. Elasticsearch is a full-text search engine and tool for analysing data (Akdogan, 2015). Elasticsearch is one of the layers of the ELK stack which is developed by the company called Elastic (Elastic, 2023). ELK stack consists of Logstash, Elasticsearch and Kibana which are all presented later in the research.

For just pointing out issues of the big data, there might be raised questions about why the big data is so important that it even needs to be dealt with and developed assisting tools to handle all the scattered data. Some general points about how big data can create value are addressed in a conference article by Kaisler & others (2013). The first point was how big data creates transparency by making data available to be analysed for business and functional analysis i.e. quality, lower costs etc. The second point was how big data can support experimental analysis for example of specific market programs. The third addressed point was how big data can be an assisting factor in identifying market segmentations at narrow levels. The fourth point was how big data can support real-time analysis and decisions which can be based on sophisticated analytics. The fifth and last point was big data being able to help facilitate computer-assisted innovations in products based on embedded product sensors indicating customer responses.

To understand the big data a visualization is needed to present the data in an understandable format. The data comes in large volumes, and it is not possible to interpret without some visualization anymore (Berinato, 2016). The data visualization will help the users to see the performance and effects of the data. It is described as an impossible task to see all the nuances and patterns hidden in the data without any visualization (Berinato, 2016).

1.1 Aim of study

Nokia being a large global telecommunication company indicates, by just being a large company, that the amount of data that it reproduces is huge. With the current tools, there has been able to recognize lack of summative analysis of different results. The current tools and solutions have been seen as not enough and a need for new development was identified. For this issue, there was created a tool that is an API between Elasticsearch and visualization tools which were already in use. The Elasticsearch and visualizations tools were not enough to summarize data efficiently which is why the new API was created inside Nokia by Sonja Rytinki (2023). Therefore, for this research, the aim is to investigate if this new tool is actually efficient enough to replace the old method by just using Elasticsearch and the chosen visualization tool.

This research is done because of Nokia's need to analyse and visualize specific data that has not been visualized yet. The data that has been collected is not in readable format due to high amounts of scattered data overflowing and presented in unstructured and unreadable format. The aim is to compare methods for finding the efficient way to retrieve the data from indexes and retrieving it for the visualization which will then act as management reports. When one needed information has been retrieved and visualized using the most efficient method, it can be used as an example in the future for creating other data visualizations. There are now two ways to implement this goal, an original one using only Elasticsearch and the new one using the newly created API, which is why the need is to find out the best one to be utilized in the future.

The two methods are retrieving the data by creating an index in Elasticsearch and saving it to another index where the data can be visualized with the chosen visualization tool. This method of using only Elasticsearch is the old way of summarizing the data into graphs. The other method is to use Nokia's own API. Nokia's own API has been developed to make this data retrieval more efficient (Rytinki, 2023). Although, the goal has been to research if this is actually an easier and faster method.

1.2 Research question and method

This research contains a comparison of the two methods of retrieving data, and the goal is to find out which is the most efficient method for this purpose. Therefore, this thesis will show the way how these data retrievals should be handled in the future to avoid the more time-consuming methods and visualize the data in an understandable format. From these requirements, the following research question was formed. *Which is the most efficient way to retrieve and visualize big data when comparing two different methods?*

This research is done as a Design Science Research (DSR) and the results are gained from relying on previous scientific research from similar solutions, studying the two methods, and the experience gained from implementing the data retrieving, summarizations, and visualization. The DSR approach has been chosen as a research method for this research due to the reason it is a good option when finding new design solutions and implementing the new design. The DSR is conducted for understanding and solving a design problem (Hevner et al., 2004). For this research, the end result is to implement a summarization of chosen data and visualize it, after researching the best way to do this. Therefore, DSR supports this new implementation process the best.

The research starts with the introduction of the tools that are relevant to the research. How do the tools work and what is the intention behind the tools? Then the research will present

the research method. How the material can be found and what search phrases are to be used for accessing the articles that are referred. The research method part also explains the steps that need to be taken to perform the design science research properly. After these, the sections go through the design science research steps. The first step is to write the motivation and describe the issue. What the new research and design will solve and why it is important.

The second part is about the design and mental model of the outcome of this research. This means describing what the new design will contain. After that, the design part includes also the step where the two methods are researched to be able to say which of them is the more efficient one and which can be recommended for use in the future. Next is the demonstration part where the research results are ready, and the data retrieval and summarization are executed with the chosen method. From the data retrieval, the visualization is implemented to be able to visualize what the outcome is. After the part where the design has been implemented, the evaluation of the artifact is done by the researcher and users. This structure is based on the Peffers et al. (2007) nominal process and it is used throughout this research.

1.3 Concepts

This chapter presents the key concepts used in this research. These key concepts are: Big data, Data retrieval, API, Index, Data Visualization. These concepts are explained what they are and what they mean regarding this research.

Big data: Big data is a collection of data that is so huge and scattered that it has difficulties with storage, analysis, and visualization (Sagiroglu & Sinanc, 2013).

Data retrieval: Data retrieval means from a database point of view an identification of the correct data and extracting that data from the database for the user (Rouse, 2023). The data retrieval happens by users search queries or applications designed for data retrieval (Rouse, 2023).

API: API means Application Programming Interface, which is developed to connect, integrate, and extend a software system. The APIs provide an interface where different applications can connect. APIs are not usually in interaction with people but those people who are interacting with APIs are developers who are creating applications or solutions with APIs. APIs are created for software to connect with other software or similarly, machines to connect to other machines. (Biehl, 2015).

Index: Index in this research is referred to as Elasticsearch index which is similar to a database. Data is stored in an index or several indexes. (Andhavarapu, 2017)

Indexing: In this research, indexing means inserting/updating documents into an Elasticsearch index (Andhavarapu, 2017).

Data visualization: Data visualization means representing data through graphs and visualizations to be able to communicate the data in a more understandable format (IBM, 2023). Data visualization helps users to see patterns in data and gain insights from the data (e.g., increase understanding, gain knowledge, solve problems) (Keim, 2002).

2. Background

This section goes through the tools used in this research. The first part is about diving into the Elastic tool, what the Elastic contains and what are the most important components regarding this research. After the general presentation about the Elastic tool, there will be an explanation of the components that the Elastic tool contains, such as Logstash and Elasticsearch. The next section is about how the data retrieval is done from the Elasticsearch. There are a few examples of the commands and results to present more carefully how the data retrieval happens in general. After that, there is the presentation of Nokia's own API which is another possible method for retrieving data. The section presents what the API is and for what purpose it is created. Then the visualization tool is presented, and lastly, the final section is about the evaluation.

2.1 Diving into Elastic

What is Elastic? The Elastic that is discussed in this research is a company originally founded by Shay Banon in the early 2000s (Kuć & Rogoziński, 2015). The company Elastic covers their multiple tools for example Elasticsearch, Kibana, etc. and forms a platform for search-powered solutions (Elastic, 2023). What is important in this research is to understand what are, ELK Stack, Elasticsearch, and Kibana. ELK stack name comes from Elasticsearch, Logstash, and Kibana, which means the ELK Stack is built from these components (Elastic, 2023). Another component called Beats was added later, whose purpose is to send data from edge devices to Logstash (Bajer, 2017).

Figure 1 shows how the ELK stack is built with four different components, first at the bottom Logstash and Beats, Elasticsearch, and lastly Kibana. What these components are, is the next important question. Elasticsearch is the engine that searches and analyses data, Logstash covers the server-side, and it is the data processing pipeline whose job is to collect data, transform the data, and send it to Elasticsearch, and Kibana is the UI for visualizing the Elasticsearch results into graphs and tables (Elastic, 2023). The data is already collected because Nokia already uses the service, but the data is not in an understandable or visualized format. Therefore, the use of Elasticsearch to find the correct data and bring it to be visualized is important. In this research, these components are under main focus, which is why these components need to be presented.

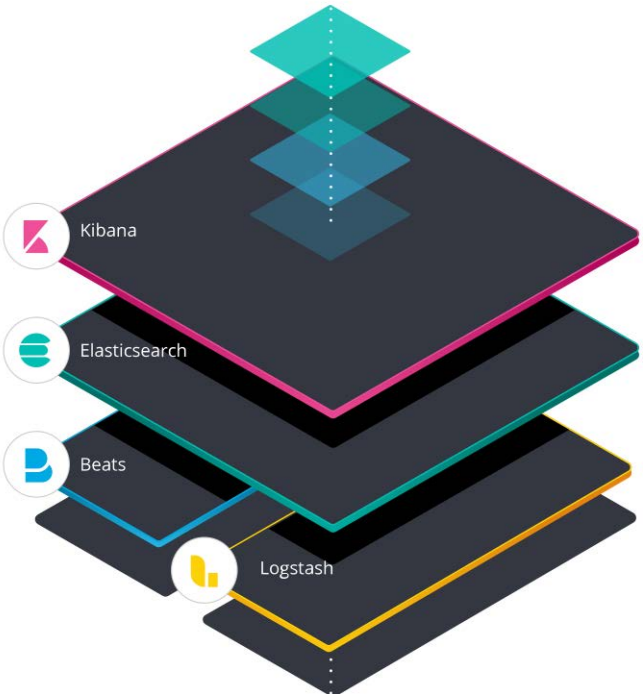

Figure 1. ELK stack with the layers it consists of in order (Elastic, 2023).

2.1.1 Logstash and Beats

Logstash is the bottom component in the ELK stack, as Figure 1 presents. Logstash is already mentioned to be a component for the server-side, and a component for the processing pipeline that collects, transforms, and sends data. While this is true, in this section the Logstash will be explained more carefully. Logstash has an important structure, which is what the Logstash is based on. The structure is 1. Input, 2. Filter, 3. Output (Bajer, 2017). This structure is based on the fact that Logstash swallows loads of data from different sources simultaneously, alters it, and then sends it forward (Bajer, 2017). Hence, follows the presented structure, and that is what the Logstash is based on.

Logstash supports different forms of input data. There can be many variations, but as an example, the input data can be a TCP/UDP socket, HTTP API endpoint, CSV files, and Elasticsearch (Bajer, 2017). After getting the inputted data, the altering phase begins. The altering in this case means that the data is stripped of redundant data or adds needed information to the existing data (Bajer, 2017).

In Elastic there was built a new product that became part of the ELK stack, called Beats (Elastic, 2023). This new product Beats was built to serve the purpose of a lightweight way to send network data to Elasticsearch (Elastic, 2023). This lightweight way means that the process gets a bit easier with the new Beats development. How this Beats has been described, is as data shippers and server agents for sending operational data to Elasticsearch (Elastic, 2023). Why these Logstash and Beats are visualized side by side and not on top of each other? The point is that Beats can send the data through Logstash to Elasticsearch where there is a possibility to parse and enhance the data or straight from Beats to Elasticsearch (Elastic, 2023). The beats support capturing multiple different data forms and have categories that are meant for capturing different data types. There are Auditbeat for audit data, Filebeat for Log files and journals, Functionbeat for Cloud data,

Heartbeat for availability, Metricbeats for Metrics, Packetbeat for Network traffic, and Winlogbeat for Windows event logs (Elastic, 2023).

2.1.2 Elasticsearch

What does the Elasticsearch then actually do? It is a JSON-based search engine that can search data from large amounts of files, logs, business analytics, website searches, Enterprise searches, security analytics, application performance monitoring, etc. (Elastic, 2023). Elasticsearch is developed to be a full-text search engine and data analysis tool (Akdogan, 2015). One could think that why need Elasticsearch when you can just read the files and search specific parts. The point of Elasticsearch is to help with fast searches from large amounts of data that would be a slow process for a human to do or even impossible. Elasticsearch is for companies or private people who need fast answers and fast data retrieval from structured or even unstructured data (Elastic, 2023).

One important thing to mention about the Elasticsearch is the connection to Apache Lucene. The Elasticsearch uses Apache Lucene which is even called the heart of the Elasticsearch (Akdogan, 2015). But then we are in front of a question, why does the Elasticsearch need to use the Apache Lucene? Apache Lucene is a text search engine library, and because of that Elasticsearch uses the Apache Lucene library to read and write from indexes (Akdogan, 2015). Therefore, Apache Lucene is actually the one doing the searching against the indexed documents, and Elasticsearch is the one using Apache Lucene (Kuć & Rogoziński, 2015).

Elasticsearch works based on nodes, clusters and shards as Figure 2 represents. The different nodes form different clusters (Kuć & Rogoziński, 2015). The new Elasticsearch node tries to connect with different nodes by sending requests to a cluster's master node (the master node is the one that communicates on behalf of the nodes in that one cluster) with the same cluster name (Kuć & Rogoziński, 2015). From Figure 2 it is visible that inside the nodes there are shards and replicas of the shards. These shards and replicas are Apache Lucene indexes and play the role of distribution, managed automatically by Elasticsearch (Akdogan, 2015). Hence, explained in easier form is that the new Elasticsearch node tries to connect all the information with the same name or ID to form a summary of the wanted information. Therefore, the new node succeeds in finding all the information with the same name/ID and can present summarized information about the already collected data. This Elasticsearch process can be described as data ingestion as it parses, normalizes, and enriches the raw data (Elastic, 2023). Because of this data ingestion, the data can be indexed which creates the possibility to run complex queries and use aggregations to retrieve data summaries (Elastic, 2023).

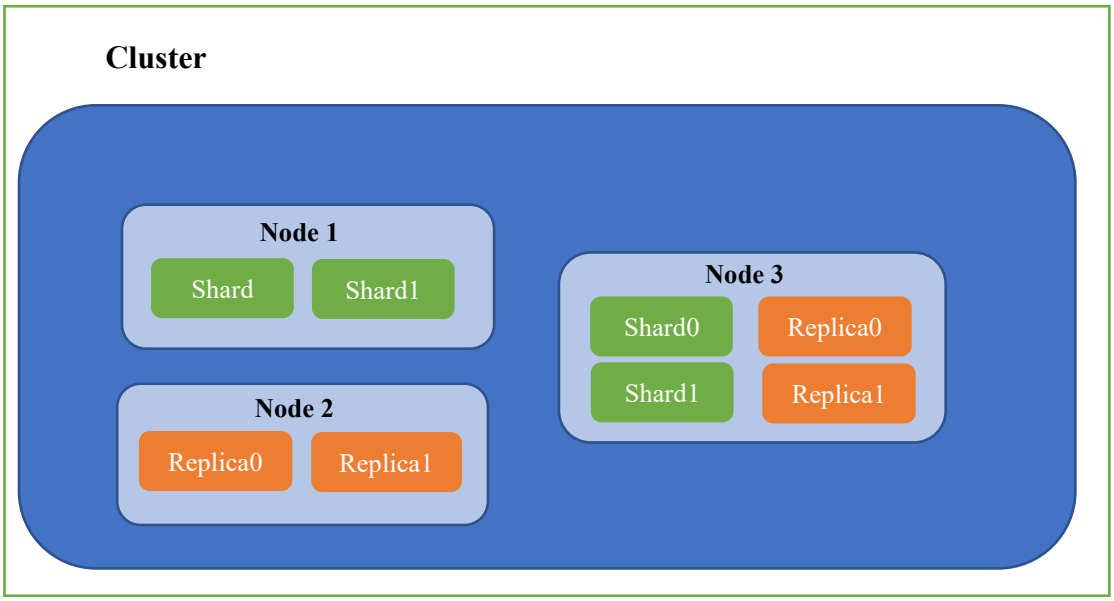

Figure 2. Example of a cluster. The blue box here is the cluster and it consists of nodes (light blue boxes), in this example there are three nodes. Nodes consist of different shards (green boxes) and possible replicas of those shards (orange boxes).

Elasticsearch works via indexes, which are there to store documents and read them from the storage (Kuć & Rogoziński, 2015). The indexes are actually a collection of documents that have similar characteristics (Elastic, 2023). These characteristics can be different names of fields or properties and they have their own values which can be strings, numbers, dates, arrays, Booleans, etc. (Elastic, 2023). These indexes operate in a way that all data is stored in a library, but the library stores the data in another data structure, and these are called inverted indexes (Akdogan, 2015). This process of inverted indexing is done by Apache Lucene (Kuć & Rogoziński, 2015). These inverted indexes store lists of unique words or terms from the stored documents and, in addition, the indexes have a list of documents in which these terms are appearing (Akdogan, 2015). From these mapped documents and terms, complete text-searching can be performed efficiently (Akdogan, 2015). A good way to understand this inverted indexing more is the way Kuć and Rogoziński describe it in their book Mastering Elastic Indexes (2015) as a term oriented not document-oriented data structure. With all these indexes, storages, and data structures, this all starts to remind and look like a database. Elasticsearch actually works like a relational database and is a highly document-oriented tool (Akdogan, 2015). In addition, the indexes work just like a database, and they use analogies from SQL (Kuć & Rogoziński, 2015).

When handling huge amounts of distributed data, there will be some delays, but one most important Elasticsearch features is the Near Real Time (NRT) searching and versioning (Kuć & Rogoziński, 2015). Elasticsearch is still considered fast because the latency is usually just around one second from the time the document is indexed to the point that it becomes searchable (Elastic, 2023). This only one second latency shows how the search engine performs fast and efficiently and makes its way of being extremely valuable for companies and for private people as well. There is no way that a human could do this faster when searching and analysing numerous unstructured documents.

But, as Elasticsearch sounds very technical and probably a little complex from time to time, the basic idea still just comes down to as basic as documents. Elasticsearch at the end of the day is just searching for documents, searching from the documents, and analysing them (Kuć & Rogoziński, 2015). It is just created to be able to do this process

efficiently and work as it should. But all in all, the basic idea is just about going through documents.

2.1.3 Retrieving the data from the Elasticsearch

As already discussed, the indexes play a main role when retrieving specific data from the storage. The stored data should be indexed and categorized properly to be able to make a successful search. The data should contain requirements such as title, category, content, date and tags (Akdogan, 2015). With these categories, the data can be mined with success connected with the right indexes and retrieved with the matching categories.

To make this possible, the key is to run queries against a field to match a term to a specific document. “In order to run a query against a field, you need to provide the field name, add the colon character, and provide the clause that should be run against that field.” (Kuć & Rogoziński, 2015). The following is an example to clarify this. If you want to match documents with the term *Home* in the term title you need to execute the following query:

```
title: Home                                                    (1)
```

Furthermore, next is an example of the Elasticsearch GET-command and results, an example that is retrieved from elastic.com (2023). The first step is to give the GET-command for the index my-index.000001with the ID 0

```
GET my-index-000001/_doc/0                                     (2)
```

The results are the following:

```
{                                                              (3)
  "_index": "my-index-000001",
  "_id": "0",
  "_version": 1,
  "_seq_no": 0,
  "_primary_term": 1,
  "found": true,
  "_source": {
    "@timestamp": "2099-11-15T14:12:12",
    "http": {
      "request": {
        "method": "get"
      },
      "response": {
        "status_code": 200,
```

```
        "bytes": 1070000
      },
      "version": "1.1"
    },
    "source": {
      "ip": "127.0.0.1"
    },
    "message": "GET /search HTTP/1.1 200 1070000",
    "user": {
      "id": "kimchy"
    }
  }
}
```

If there is a need to index a document to an existing index the next example shows how to do this. An example is provided by Andhavarapu (2017) in the book *Learning Elasticsearch : distributed real-time search and analytics with Elasticsearch 5.x*: (pp. 84-85). Here in the code example (4) the PUT command "puts" a new person, person 1, into a chapter4 index and creates the id, name, age, gender, email, and timestamp of the modification date for the person.

```
PUT chapter4/person/1                                                    (4)
{
        "id":1,
        "name": "user1",
        "age": "55",
        "gender": "M",
        "email": "user1@gmail.com",
        "last_modified_date": "2017-02-15",
}
```

Here in code examples (2) and (3) is visible how with the GET command you can retrieve an already created index. The example (4) shows how to index your data to an existing index.

In this research for Nokia two methods are being compared. The first method requires the creation of a completely new index where there is fetched information from current indexes. The other method is based on API where there is a need for reading data from

several current indexes. Both methods use the GET command to get the index data to be retrieved and visualized. For this reason, the GET command is presented in this background. The retrieval is possible by Elasticsearch APIs that makes it possible to feed the data and build queries (Kuć & Rogoziński, 2015). Elasticsearch provides multiple different APIs that perform different tasks. According to Kuć and Rogoziński (2015), "In general, it would not be a surprise if we would say that every feature of Elasticsearch has an API."

2.2 The Nokia's own API

For the next part, the Nokia's own API that is used in this research is introduced. The API was designed and built by Sonja Rytinki for Nokia purposes in 2023. The purpose of the API is to help users retrieve needed data quickly and easily. In particular, the API is a "new analysis tool" that is "designed to help the testers and project management who need to have a quick and clear access to the results of the test runs." (Rytinki, 2023). The API is designed to retrieve data from selected data indexes from the Elasticsearch analyse and summarize the data, and save it to another data index (Rytinki, 2023). The API basically does the work for the user rather than the user having to perform GET commands to retrieve data by themselves. The API is created as a REST API, and it uses HTTP requests (Rytinki, 2023).

The API retrieves data that is important for Nokia, and only that data is analysed (Rytinki, 2023). The big data that Nokia reproduces is not all that important for these summarization purposes, which is why there has been selected only the highly useful data to be analysed. This tool was created for the need for a summarization tool for analysing and visualizing data (Rytinki, 2023). There was a lack of good tools that helped to retrieve data to be analysed and summarized to be easily reviewed and get conclusions for success and quality. There was raised an issue with big data and the scattered printing of data that was impossible to view and keep track of. This new tool created by Rytinki, has been successful in retrieving data for creating different graphs from the chosen categories that are useful for the need of Nokia's testing and help Nokia employees, especially managers, to get conclusions of the needed aspects and learn how for example a new product is performing in different settings. Therefore, the managers can be able to conclude important information from the results for further use.

This API contains five different categories/use cases Hardware, KPI (key performance indicators), Test line, Software and default which all have their own GET command for the data retrieval from Elastic (Rytinki, 2023). All the categories have their own purpose from a testing point of view. These categories are selected for Nokia's testing purposes and to visualize the summary of testing success and performance from different points of view. The API's purpose is to provide easy and clear views of common issues that need to be monitored to see the performance of Nokia's products (Rytinki, 2023). To be able to keep track of the performance and quality of the large scale of different and continuous testing, this summarization is highly needed. The aftermath of this tool is completed and presented to the people that will use it concluded the API is suitable to fix the raised issue of the lack of summarization of analysed data, and the tool has been set to run on the company server (Rytinki, 2023).

Even though the tool has been introduced to make the data retrieval process to become easier, Rytinki has mentioned limitations (2023). There have been found issues with missing data values and data that is not analysed. These missing values and results affect the overall results because they cannot be counted with other results such as pass or fail.

There are also identified issues with percentages being selected as indicators of errors and warnings, but the percentage indicators might lead to misunderstandings due to users not considering the full data set. Even though, these limitations exist there has been seen as a possibility for this tool to make this data retrieval process simple and with good visualization there is a great possibility to achieve easier tracking of the success and quality of the testing results.

This possibility will be researched in this thesis by digging deeper into how this tool really works. The purpose is to investigate if this API could be a more efficient way for users to get the summarization of needed data, rather than users themselves creating indexes from Elastic to retrieve data and save it to the indexes. This research aims to get clarity on the API's efficiency and to examine if the API makes the data retrieval process more simplified and easier than the original way of only using Elasticsearch.

2.3 Visualization tool

The information that is collected via the Elasticsearch can be visualized with different tools such as Kibana, Power BI, and Grafana. The data that needs to be visualized is currently not in a format that is clear and easily readable and comparable to other summarizations. There is a definite need for visualization and presentation of the data in a format that can be read easily and understood. This research will retrieve data in a format that is readable by the visualization tools, which will present it in a way that users can understand.

For this purpose, Power BI is chosen as a tool to visualize the data summarizations. Power BI is one of the most popular visualization tools and a “hot topic” in data visualization (Ferrari & Russo, 2016). It is also an option in Nokia as it is a tool that Nokia uses for data visualizations. Power BI was created by Microsoft in 2009 and its purpose from the beginning was to be a self-service business intelligence tool (Ferrari & Russo, 2016). The Power BI job in this scenario is to present the data in a readable format. Hence, Power BI can read queries and then display the data that has been retrieved (Ferrari & Russo, 2016). Figure 3 presents a graph that is presented as an example of data visualization.

Figure 3 Example of visualization of the retrieved data.

2.4 Evaluation

While the research process goes on, evaluation of the process needs to happen continuously and iteratively. The evaluation can be done in many ways but, in this research, Sonnenberg's and Vom Brocke's (2012) evaluation model and Nielsen's (1992) heuristics are used. The process of DSR and evaluation goes through six steps which are problem identification and motivation, definition of the objectives, design and development, demonstration, evaluation, and communication (Peffers et al. 2007). The evaluation step is performed to ponder if the design has been successfully made toward a working artifact. These DSR steps are defined by Sonnenberg and Vom Brocke (2012) as problem identification, design, construction, and use, which all are followed by the evaluation phase.

According to Sonnenberg and Vom Brocke (2012), the evaluation consists of ex ante and ex post evaluation. The ex ante evaluation is the design part and goes through the phases before the construction of the artifact and the ex post evaluation goes through the artifacts phases after the construction such as the use of the artifact (Sonnenberg & Vom Brocke, 2012). This Sonnenberg's and Vom Brocke's evaluation model fits with the Peffers et al. (2007) nominal process as the Sonnenberg's and Vom Brocke's (2012) model just adds evaluation phases after every step in the Peffers et al. (2007) nominal process steps.

The other evaluation method that is used in this research is the Jakob Nielsen (1992) heuristics. These heuristics are for the data visualization part of this DSR process. Jakob Nielsen has defined five key heuristics which are learnability, efficiency of use once the system is learned, ability of infrequent users to return to the system without having to learn it all over, frequency and seriousness of user errors, and subjective user satisfaction (Nielsen, 1992). But, as Nielsen (1992) has explained, rarely designs cover all these five objectives as their top priority. Usually, only a few top heuristics are picked, and the design is evaluated based on those chosen top priorities. Therefore, in this research, the top few goals are chosen from these five objectives and evaluated the visualization design based on the chosen priorities.

3. Research Method

This chapter presents the research method Design Science Research which is the research approach applied in this research. After that, the design process is described, how the design process goes through, and what models are used going forward with the research process. Also, there are figures to represent the process in a more understandable format.

3.1 Design science research

This research is done by using the Design Science Research (DSR) method. The method was chosen because it is a problem-solving process. The DSR is suitable for the issue of comparing two methods of data retrieval and implementing the best method for an actual data retrieval, summarization, and visualization process. Other research methods would not have been able to support the implementation part of the process as well as the DSR method does. According to Hevner and Chatterjee (2010), DSR "seeks to develop and justify theories (i.e. principles and laws) that explain or predict organizational and human phenomena surrounding the analysis, design, implementation, management, and use of information systems."

The DSR is, as said, a problem-solving process, but it is created for understanding and solving a design problem from where the results are used in building an artifact (Hevner et al., 2004). The DSR is also explained as a paradigm, where the designer answers relevant questions by creating artifacts for the purpose of creating new knowledge (Hevner et al., 2004).

What is important with the DSR is to be able to find the following three objectives from the research (Peffers et al., 2007). First, it is important to provide a nominal process for the conduct of DSR. Next is to build the research on top of previous literature about design science in information systems and reference disciplines. The third one is to provide researchers with a mental model or template for the structure of research outputs.

For this process of solving a design problem and implementation, there are presented six steps to follow to be able to complete a DSR project as it should. These six steps are problem identification and motivation, definition of the objectives, design and development, demonstration, evaluation, and communication (Peffers et al., 2007). These six steps as a whole, are called nominal process sequences (Peffers et al., 2007). In the following, the DSR process is described based on Peffers et al. (2007). The nominal process begins with the problem definition.

The first step in a DSR project is to identify a problem and think about its importance, what is the actual problem, what it solves, and who benefits from the new solution. Can there be found a group that actually benefits from the solution and is it really a problem to someone? The work should not be wasted on an issue that is not really an issue. The second step is to define the objectives of the new solution, to identify the artifact and what that artifact would be accomplishing. The third step is to design and develop the artifact and the fourth step is to demonstrate how it works or how it can be implemented. Steps two, three and four are answering the questions of what should be done, how it would be executed, and what would be the outcome of these. The last two steps are evaluation and communication. The evaluation is where the solution is observed and evaluated on how it performs in the opinion of others and possibly identifying issues. Lastly, the communication is meant for publication.

What the previously mentioned mental model then covers is the fact that there needs to be a hint of reality. The mental model is there to present a model or diagram to be able to create an understanding of what would be to come (Peffers et al., 2007). The mental model is actually something that will help the researcher. It helps the researcher to conduct the research efficiently and will help with getting the results done without too much latency (Peffers et al., 2007).

What the DSR produces is an artifact. The artifact is something created in DSR which comes from knowledge of a task or situation (March & Smith, 1995). DSR can be divided into two sections. These are "build" and "evaluate", where "build" includes the building of the artifact for the specific use and "evaluate" refers to evaluating the artifact's performance in that specific use it was built to (March & Smith, 1995). The artifact represents the end goal of the research, and it will develop during the DSR process.

There are four types of artifacts. These are construct, model, method, and implementation (instantiations) (March & Smith, 1995). March & Smith (1995) explain what these mean as follows. Constructs are for the need for basic language concepts, models are for describing tasks, situations, or artifacts, methods are for presenting ways to present activities, and implementation (instantiations) are for concrete implementations for executing specific tasks. In this research, there can be identified two different artifacts. These are a method which defines the research of choosing a method to perform the data retrieval, and the implementation artifact which is the visualization part of this research and will be an actual implementation as a management report.

3.2 Research process

The research was started with a search of scientific literature for introducing the tools that will be used in this research. The search was started by using Google Scholar and forming search phrases that could give good hits on the topic. The first search that brought up results that are used in this research was *elasticsearch.* This search resulted in 19,800 hits, which was a lot. One notable result of the search was a book called Mastering Elasticsearch written by Rafał Kuć and Marek Rogoziński (2015). This book is cited a lot in this research and was an important find regarding this research. The next search phrase was *elasticsearch indexing* which resulted in 11,800 hits in Google Scholar. This search also resulted in a relevant finding regarding this research which was the book called Elasticsearch Indexing written by Huseyin Akdogan (2015).

Another important tool to find important information was using Elastic's official website. The Elastic official website was an important source for this research because it provided information about Elastic's different features and possibilities that are investigated and used in this research. The next search was about big data and finding articles about the basics of big data. One search word was *big data management*. It led to too many results, more specifically over 4 million hits, but it had one good article at the top of the list that was good for the explanation of the big data in this research. Another search for finding basic information about big data was searching with the phrase big data. Again, it resulted in too many matches, over 10 million, but it had a good article describing basic information about big data. Some of the articles were not accessible from Google Scholar, however, the articles were possible to be accessed via other databases such as Scopus.

The search process continued throughout the research process to find specific articles to support the claims. This was done by multiple different search words and therefore picked

the most suitable ones. Also, some of the articles were picked from references of other articles, or articles that were already familiar and therefore searched by the article's name or writer.

Because of the fact that this research is done for Nokia and one of the tools created by Nokia, the research material was also searched inside Nokia and not only by using public search engines such as Google. One of the research methods was to get access to documents by asking the authors themselves. Also, getting materials from Hannu Kaarela, as he is the one initiating this research problem and the supervisor of this research. Also, other experts at Nokia were a source of information about the specific issues that were related to tools and access to Nokia's tools and data.

As this research is a DSR project, the structure will follow the DSR process structure. First of all, the model called nominal process is went through. The nominal process is presented by Peffers et al. (2007) where the steps have been divided into a six-step iterative process. The nominal process starts with identifying the problem and writing the motivation to show the importance of the new design created in the process (Peffers et al., 2007). The need for Nokia's purposes is presented and defined the reasons for this research. Why Nokia would benefit from this and what is the need that has initiated this research. The next step will define the objectives of the solution and present the artifact of the design (Peffers et al., 2007). Then there will be a presentation of the artifact and a demonstration of the artifact as the two next steps in the nominal process (Peffers et al., 2007).

After these steps, there will be visible the design of the new solution for Nokia's need for summative analysis and the method of how this will be performed. There will be a visualization of how this summarization is presented and what is the outcome of this research. The process of creating the visualization and researching the methods used to obtain the data to be visualized represents a mental model of the DSR process carried out in this thesis. Figure 4 represents this mental model, and it also shows how the new design proposed in the thesis is demonstrated. What will also be presented is the reasons behind the solution. The demonstration part contains the research behind selecting the best method to perform the data retrieval, namely the investigation of both methods and the result of their efficiency and easiness. Both methods are investigated by performing them both and relying on previously done research about the topic. This part is one of the three important objectives of design science research (Peffers et al., 2007).

What is important to understand is that the visualization will be done only for the method that is chosen in this research and investigated to answer better the needs and goals. The result from both methods is the same so it is not necessary to do visualization from both methods as the Figure 4 presents. The Figure 4 presents how the Peffers et al. (2007) demonstration step is done, by retrieving data, then executing the better method, and then visualizing it. The visualization performs the summary that has been retrieved from the existing data using the chosen method.

Lastly, there will be evaluation and communication steps, which in an iterative process can initiate new iterations (Peffers et al., 2007). The result of the previous steps will be evaluated by the researcher and 2-5 experts from Nokia who are familiar with the topic. From their comments, there will be possible iterations for improvement. The last step named communication in the nominal process is about publications and will be the last step in this research (Peffers et al., 2007). Figure 5 will present the whole research process as a diagram based on the Peffers et al. (2007) nominal process.

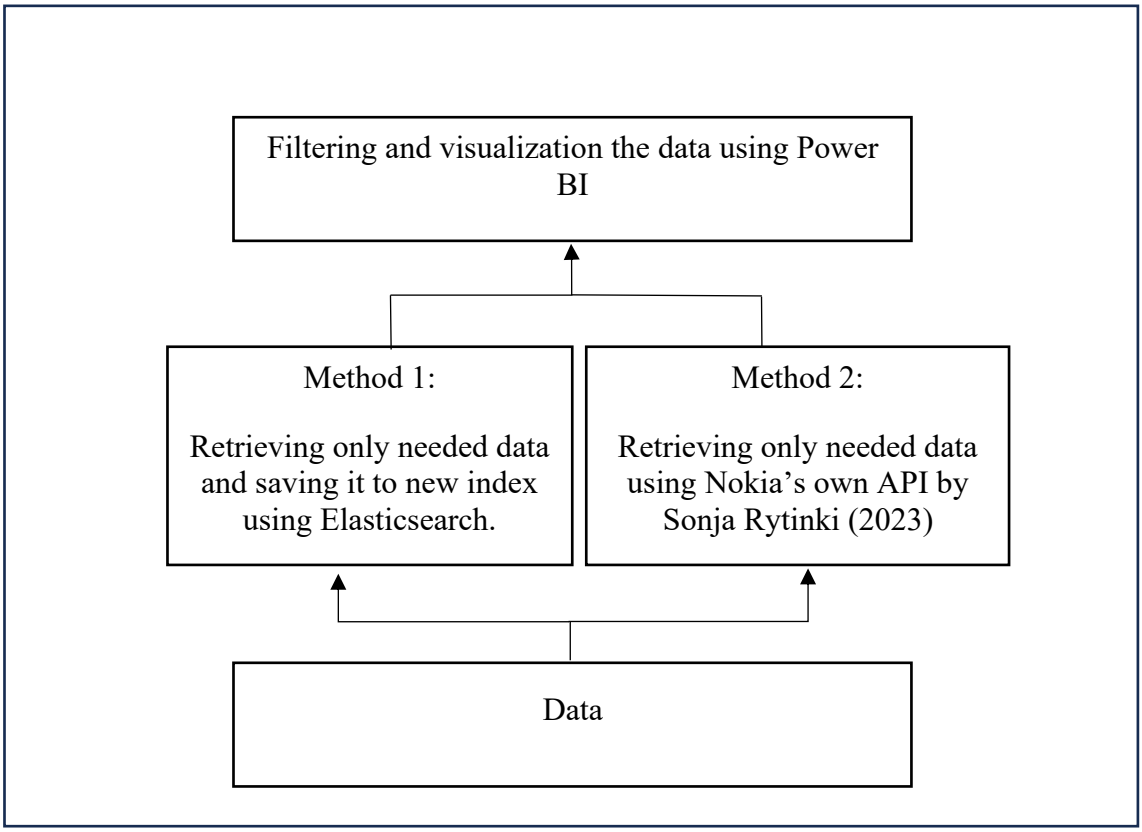

Figure 4 The mental model in this research (also the process flow of the demonstration part in this research according to Peffers et al., 2007).

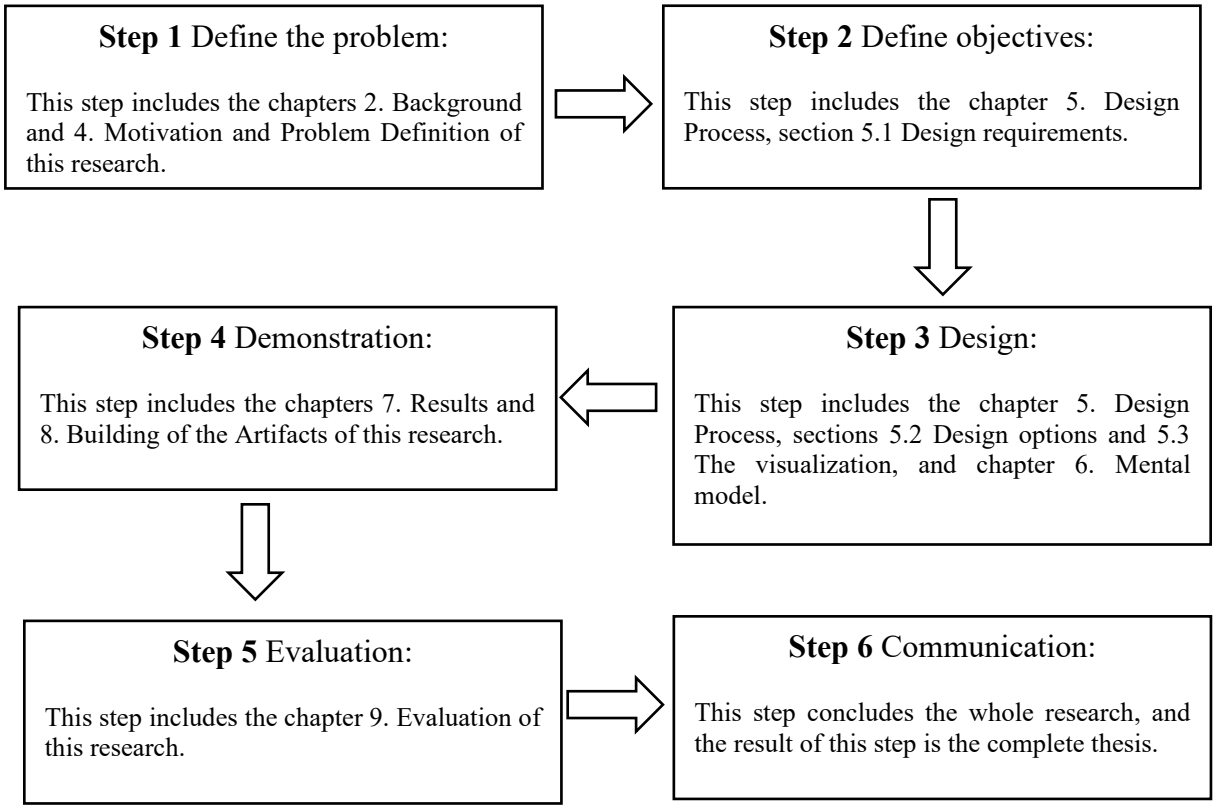

Figure 5 The process flow of the whole research process based on the Peffers et al. (2007) nominal process.

4. Motivation and Problem Definition

In this chapter, the motivation of this study and the problem definition of this study are presented. This chapter is part of the first step of the Peffers et al. (2007) nominal process. This chapter defines why the research problem is relevant and what are the possible methods to solve the problem. Therefore, this chapter defines the research issue and answers the questions of why it is important, and what needs to be solved.

4.1 Defining the problem

Big data analytics is an important factor in being a competitor in today's business markets. By analysing the company's big data, the company can get competitive leverage to business. As Thomas Davenport says in his article, with the help of analytics companies can squeeze every last drop of value from their business processes (Davenport, 2006). What Davenport (2006) also explains about data analytics is that with the analytics they will get information such as what will trigger them to buy more items or predict how many items the customer could be buying in their lifetime. To understand customer behaviour and possible needs, it gives the company the possibility to predict what the customer could be buying in the future. Big data analytics has been said to be a breakthrough in business communities because its performance gains can be life-changing for a company (Mikalef et al., 2019).

Big data analytics is so powerful that measuring it and managing it, is better than ever before and something that people could only dream of just a generation ago (Mikalef et al., 2019). Nokia is no exception to this. The data analytics from many different areas are crucially important to understand performance from every point of view, "Simply put, because of big data, managers can measure, and hence know, radically more about their businesses, and directly translate that knowledge into improved decision making and performance." (McAfee et al., 2012). In this research, the focus is software testing and its performance. Analysing big data from the testing, it will give managers an understanding of how well the product is working, how it looks, whether is it sufficient enough, and a lot more. Therefore, the research and implementation of new summative analysis in Nokia is important. There are already many analytical tools in use at Nokia, but right now there is a need for creating a management report for specific data coming from Elasticsearch that has not yet been analysed or visualized at all.

The need for even more management reports has risen in Nokia. Nokia's data from Elasticsearch should be summarized in a way that it is easily visible for everyone to understand. The current issue is that data flow into Elastic is extremely high, and the data is scattered and in an unreadable format. It is no longer a sufficient way to retrieve information just by looking at the printing or retrieving with GET commands. What is needed is a summarization of the data in Elasticsearch and visualizing it in a readable format. What is now the issue is that summarizations have been done for some key topics, but more is needed. And as the importance of analysing big data was already stated, the need for more summarizations of the data is important and beneficial to Nokia.

There are also two different ways to execute this and not too much information on which one is the preferred way to use. It is important to further develop knowledge on what is the method that should be preferred. This research is devoted to solving the future issue of balancing between two methods and wasting time on time-consuming and complex methods. The current situation is that data retrieval is possible to do on Elasticsearch

indexing. Indexing means that data is being stored into an index (similar to a database) (Andhavarapu, 2017). So, for example, software test result data is stored in an index called Results. This method operates with commands for example creating indexes, retrieving existing documents, saving documents to and indexing etc. with just using commands such as GET, PUT, POST (Andhavarapu, 2017).

For this, Sonja Rytinki has developed an API for Nokia to use. The API is a tool that is built to do all this indexing for the user and not the user having to do all this themselves. The API retrieves the data from the index of Elasticsearch, analyses it, summarizes it, and saves it to another index (Rytinki, 2023). This tool was freshly made in 2023 and is still not yet investigated of the promising performance that it has. Does this tool help create the summarizations more efficiently than it would be done without this tool? Does the tool have limitations that make the solution unsuitable for this specific scenario? These questions are solved in this research and many other issues that need to be investigated to be able to answer the research question.

5. Design Process

This chapter goes through the design process of the new artifacts and the comparison between the two methods. The first section describes the requirements of what the artifacts have and what is needed from Nokia's point of view. The next two sections discuss the benefits and challenges of the two methods and rely on previous research about similar topics. This chapter is part of Step 2 Identifying the objectives of this research (section 5.1) and Step 3 Design the solution artifact (Sections 5.2 and 5.3) as the Figure 5 presents.

5.1 Design requirements

The new design has specific requirements that have been defined from Nokia's point of view, what is missing and what needs to be developed. The need that has defined these requirements is the summarization of scattered data collected to a different Elastic index and not visualized in a readable format. The design has technical requirements, but the outcome should also be designed to fulfil general design requirements.

The technical requirements include data retrieval from Elastic. Retrieving the documents and indexes from the Elasticsearch that Nokia's test environments produce. Retrieving the needed data and not filling up the system with irrelevant data. The data volume is so large that storing, accessing, searching, and analysing are challenging. In this case, the data volume needs to be considered. Because the goal is exactly to achieve what is challenging with big data, analyse, store, access, and search. The solution to achieve Nokia's needs can only be reducing the size of the data that needs to be analysed. The answer is to retrieve only some of the data and analyse only that. For that Elasticsearch is specifically the answer, "In particular, Elasticsearch – a distributed full-text search engine – explicitly addresses issues of scalability, big data search, and performance that relational databases were simply never designed to support" (Kononenko et al. 2014). One important point is also the fact that the data needs to be in a format that Power BI can read it, which is currently not the situation.

Something that was raised as a quality point in Nokia's meeting about this topic was that result to become reasonable for Nokia, it needs and should include automatic refresh and data updating. Hence, automation would make the process easier when the data is updated by itself. Another thing emphasized about this topic and wanted results was to provide the solutions in a way that it creates as small a number of documents as possible due to the issue of storage space that big data causes.

One of the design options is also to get data from different indexes to be saved into a new index. The data should be retrieved from the new index that is created to contain the wanted information to conclude management reports. The information needs to show results and tell the source of the results to become relevant information. This new index should not be bigger than the original index. The other option is just getting data using Nokia's own API. The data that the API has retrieved is then visualized by Power BI which retrieves the data from the API.

The visualization of the analysed data needs to fulfil user requirements as well as the technical aspects. The widely used usability model created by Jakob Nielsen (1992) is used to define user requirements. The characteristics that need to be met are learnability, efficiency of use of the learnt system, ability of users to return to the system without

having to learn it all over again, frequency and seriousness of user errors, and subjective user satisfaction (Nielsen, 1992). These five characteristics are taken into consideration during the design process. But, as Nielsen also says, not all 5 characteristics are equally important, and in every design, some of the characteristics are more important than others (Nielsen, 1992).

For effortless data visualization and clarity for the visualization, there were selected design principles to help to achieve that. For this purpose, Tufte's (1983/2007) design principles were considered. Tufte (1983/2007) defined several design principles of which the following are selected to be considered when designing the visualization for this project: Show the data with clarity, precision and efficiency, Make the viewer think about the substance of the data rather than about the visualization technique (design and computational issues), Encourage the eye to make comparisons between data, Use clear, detailed, and thorough labelling in order to avoid graphical distortion and ambiguity, Show data variation, not design variation, Have a properly chosen format and design, Avoid content-free decoration (i.e., user of colours and other graphical attributes). These principles are listed in Table 1.

Regarding Nielsen's (1992) and Tufte's (1983/2007) design principles, this research will follow all of these for the visualization. Some of them are high priority and some of them are low priority, but all characteristics are gone through and considered. The visualization design will be provided in a way that it is easily learnt what the visualization presents. The data summary visualization will also be presented with correct titles and easy graphs so that it is not needed to learn again and again what the data is there to visualize or how to read the graphs. The view will not have any possibilities for the user to destroy or hide the visualization by accident. The result of the visualization will be something that users are satisfied with, it is pleasant to look at, and it visualizes the exact needed information, not anything less or something unnecessary. The visualization will also be created in a way that user will intuitively make comparison between data and notice the data variation. The visualization will also present the data that it encourages the user to think about the data results rather than the design, and colours and formats will be used properly.

Table 1. Tufte's (1983/2007) design principles.

1.	Show the data with clarity, precision and efficiency.
2.	Make the viewer think about the substance of the data rather than about the visualization technique (design and computational issues).
3.	Encourage the eye to make comparisons between data.
4.	Use clear, detailed, and thorough labelling in order to avoid graphical distortion and ambiguity.
5.	Show data variation, not design variation.
6.	Have a properly chosen format and design.
7.	Avoid content-free decoration (i.e., user of colours and other graphical attributes)

5.2 Design options

There are a few different possibilities which could be used in this research to design and implement the new artifacts. Those are for method 1. Rollup API, Transform API, Reindex API, and for method 2. Nokia's own API. These are all researched and investigated for their possibilities and limitations to be able to see which method should be used and what elements to use in the chosen method. The possibilities and limitations will be the ones to affect the decision of the method for this research. All these are presented in this chapter.

5.2.1 Rollup API

First, the Rollup API is presented. The Rollup API is Elastic's API which can be set to execute within wanted intervals (Elastic, 2023). The Rollup API can attach live and historic data into one index, "A useful feature of Rollup is the ability to query both "live", real-time data in addition to historical "rolled" data in a single query." (Elastic, 2023). The Rollup API works in a way that it rolls up the raw data into these so-called historical summaries and does not include the irrelevant data from filling up the limited storage. The function that is important in Nokia's point of view is the possibility to roll up historical and live data, where there is a possibility to see data for example from the beginning of the month to this day without retrieving the data separately.

The Rollup API is time-based and identifies new documents by their timestamps (Elastic, 2023). The Rollup API has an object called cron, which defines the intervals when the Rollup API is performed, for example once in a day or once in an hour (Elastic, 2023). The Rollup API creates a new index where the rolled-up data is stored (Elastic, 2023). This new index creation is one of the requirements in this new design, and in the code example (5) the "rollup_index": "sensor_rollup" is the new index where the data is stored after the rollup is done. What is great about the Rollup API is that it can limit the data from accumulating into masses. The Rollup API updates the new index and updates only the fields that are chosen for the new index. There is a retention policy where users can define the age of the documents and older ones are thrown away (Elastic, 2023).

Example of the Rollup API retrieved from Elastic (2023):

(5)

```
curl -X PUT "localhost:9200/_rollup/job/sensor?pretty" -H 'Content-Type:application/json' -d'

{

  "index_pattern": "sensor-*",

  "rollup_index": "sensor_rollup",

  "cron": "*/30 * * * * ?",

  "page_size": 1000,

  "groups": {

    "date_histogram": {

      "field": "timestamp",
```

```
      "fixed_interval": "60m"
    },
    "terms": {
      "fields": [ "node" ]
    }
  },
  "metrics": [
    {
      "field": "temperature",
      "metrics": [ "min", "max", "sum" ]
    },
    {
      "field": "voltage",
      "metrics": [ "avg" ]
    }
  ]
}
```

In this example (5) the PUT command starts the creation of the roll up-index and the rest are the specifics of what the rollup actually does and where the data is retrieved. Index_pattern and rollup_index are the source and destination from which the rollup job retrieves the rolled-up data and where the rolled-up data is saved. Cron defines the interval of how often the rollup is performed, in this example it is every 30 minutes, and the groups define how the data is grouped into the destination index. In this example, it is grouped by date and nodes. Metrics are there to do max, min, sum, and average metrics from the data.

Even though Elastic provides this query there are limitations in the Rollup API as well. The Rollup API does not support everything such as some search features are disabled i.e., highlights, and only one index using Rollup API is allowed to be specified (Elastic, 2023). Also, the issue discussed with Nokia's experts is the fact that the testing of the Rollup API would be time-consuming. The need is to perform the Rollup API only once a day, and the testing will always take hours. What the Rollup API can do is defined in three sections. Logistical details such as schedule, grouping fields, and metrics can be collected from each group (Elastic, 2023). For grouping Rollup API supports only date histograms which buckets data based on time, histograms which buckets numerical fields into numeric histogram intervals, and terms which can group based on keywords (Elastic, 2023).

Even though Rollup API solves the problem of getting updated data this is not solving the issue of summarization of the data as specifically that it is required. The data can be retrieved by grouping and counting metrics (Elastic, 2023). The Rollup API only retrieves

data, but that data is not filtered in any way. In this new design, the requirement is to retrieve data and filter it to per specific analysis area that is significant to Nokia. This Rollup API does create a new index with significant data, but it is not filtered as specifically as needed. There are two possibilities to filter the data. If the Rollup API is chosen to be used, then the visualization tool Power BI does the filtering. The issue is only that the index created from the Rollup API will be huge and could face the issue of limited memory. The other option is to use the Transform API which has more possibilities to filter the data and summarize only the needed data with a specific timeline.

5.2.2 Transform API

For summarizing and filtering the data together and creating a new index, Elasticsearch has an API called Transform. “This API defines a transform, which copies data from source indices, transforms it, and persists it into an entity-centric destination index” (Elastic, 2023). This Transform API solves the question in this research of filtering the specific data together to form a management report. The Transform API can summarize and retrieve complex and specific insights from the big data that is collected by Nokia. According to Elastic (2023), one can “summarize complex things like the number of web requests per day on a busy website, broken down by geography and browser type.”

Pivot object in Transform API groups fields into wanted groups using the “group_by” command (Elastic, 2023). This method can retrieve and summarize the needed information together and answer the need for summarization for management reports such as pass/fail percentage of specific issues or count of reports from a specific team. The Transform API supports multiple properties in pivot which makes it possible to retrieve and count the needed data. Transform API’s pivot method supports such aggregations and the group_by object, which are needed for this design (Elastic, 2023). The Transform API is a useful job to do filtering and summarizing altogether and not taking so much memory space. The Transform job needs a pipeline for the transformations (Elastic, 2023). It is a bit more complex to execute than the Rollup API and the filtering with Power BI. The point is to think about if the transform is necessary to execute because the power BI does already the filtering itself.

Example of the Transform API retrieved from Elastic (2023):

```
(6)

 curl -X PUT “localhost:9200/_transform/ecommerce_transform1?pretty” -H
‘Content-Type: application/json’ -d’

{

  “source”: {

    “index”: “kibana_sample_data_ecommerce”,

    “query”: {

      “term”: {

        “geoip.continent_name”: {

          “value”: “Asia”

        }
```

```
    }
  }
},
"pivot": {
  "group_by": {
    "customer_id": {
      "terms": {
        "field": "customer_id",
        "missing_bucket": true
      }
    }
  },
  "aggregations": {
    "max_price": {
      "max": {
        "field": "taxful_total_price"
      }
    }
  }
},
"description": "Maximum priced ecommerce data by customer_id in Asia",
"dest": {
  "index": "kibana_sample_data_ecommerce_transform1",
  "pipeline": "add_timestamp_pipeline"
},
"frequency": "5m",
"sync": {
  "time": {
    "field": "order_date",
    "delay": "60s"
  }
```

```
  },
  "retention_policy": {
    "time": {
      "field": "order_date",
      "max_age": "30d"
    }
  }
}
```

Here in this example (6), the PUT command starts the creation of the new transform index and under "source" there is the source index from which the data is retrieved and a main filter which the data is filtered through. In this example, only data that has Asia in the source index field called geop.continent_name is retrieved. Pivot contains the grouping and the aggregation wanted to the summarization; therefore, the data will be grouped by the fields that are specified in the pivot and the aggregation retrieved. The "dest" contains the destination where the data will be stored, a new index created by Transform API. "Sync" on the other hand is the one needed for a continuous summarization and specifies the intervals when the data is updated to the summarization. "Retention policy determines how old data in the new index will be and how older will be deleted.

One important element is to start the Rollup API or the Transform API. The PUT command only creates the job, therefore the job needs to be started (Elastic, 2023). The following commands retrieved from Elastic (2023) start the Rollup API (7) and the Transform API (8).

```
POST _rollup/job/<job_id>/_start                                    (7)

POST _transform/<transform_id>/_start                               (8)
```

To these commands, the index created in Rollup API needs to be added to the *<job_id>* and index created in Transform API needs to be added to the *<transform_id>* and otherwise the command will look like the example.

5.2.3 Reindex API

The third option considered in this research is simply by just reindexing the wanted data into a new index. This can be done with the Reindex API in Elasticsearch, which simply copies documents from a source index and saves them to a completely new destination index (Elastic, 2023). This Reindex API supports indexes, aliases, and data streams as a source and the only limitation is that the destination index needs to be a separate index. "The destination must differ from the source. For example, you cannot reindex a data stream into itself." (Elastic, 2023).

The Reindex API supports several possibilities, which in this research is crucial. Important in this research is to get all needed data, and not to leave anything out. Without all the needed data fields, management reports cannot be created to visualize the data in a correct form and filter the data so that it is actually useful. The Reindex API supports for example features called query, size, sort, and index (Elastic, 2023). These are

something that possibly needs to be used when creating a new index for the management reports. The query parameters that are supported in this Reindex API are for example max_docs, timeout, slices, etc. which also might be needed to be used to build the method artifact (Elastic, 2023). There are also a lot of other possibilities with this Reindex API. An example of this Reindex API is code example (9) and it is retrieved from Elastic's website (2023).

```
POST _reindex                                                    (9)

{

  "source": {

    "index": "source",

    "query": {

      "match": {

        "company": "cat"

      }

    }

  },

  "dest": {

    "index": "dest"

  }

}
```

Here in this example the POST _reindex creates the new index with the given restrictions. The source index is defined to be named source, and the destination index is defined to be called dest. This can be seen from the line where there is "index": "source" and from the line where there is "index": "dest". These define the source and destination indexes. Then there is a query part which is there to specify the documents to the new index (Elastic, 2023). In this case, the query identifies with the operation "match" all companies which have the name "cat"; this is seen in example (9) from the "match" object.

5.2.4 The Nokia's own API

Nokia's own API is created for data retrieval. It consists of 10 use cases which all retrieve specific data that is determined in the use cases (Rytinki, 2023). The API is created in a way that the future work of this API can be creating new use cases (Rytinki, 2023). Nokia's own API is the second possible method that could be used for data retrieval in this research. The API is created by using Python and Python libraries (Rytinki, 2023). Python is a programming language and is developed as an open-source language with an OSI-approved license (Python, 2023). Therefore, this second method in this research is going to be investigated by using Python. How this API then works with Power BI is as follows. Power BI sends a query to the API, and the user-selected parameter will be used to retrieve the correct use case from the API (Rytinki, 2023).

This API shows a lot of potential to solve this research issue, but limitations have been raised. One crucial issue that was raised by Nokia's experts was that the API has issues with data refreshing. Hence, the API can only summarize data but there is not a possibility to update results; this function was not completely implemented as would have been needed. Therefore, the API has one element missing that would be useful for Nokia's purposes. This lack of updating possibility is crucial and something that needs to be considered as a point that discards this method from the possibility of using it. Another thing that was raised as a concern by Nokia's experts was that the source index that the API uses has some of the data needed in this research in a format that cannot be read correctly from Power BI. Therefore, the API could not be compatible with the data that is needed to be retrieved.

5.3 The visualization

After getting all the data in a reasonable format and all the needed information, the data is then to be visualized with the Power BI tool. The Power BI tool will receive the data in a format that Power BI can read. Power BI will filter the data the way the user wants to see it. The visualization will be performed by using Power BI and the visualization will display the data with a graph which has text, colour, numbers, title, and the time period of the results.

The visualization will be designed in a way that the user does not have to do anything else than look at the graph to understand what it represents. It is important that the graph does not have to be learnt over and over again or studied in any way. The text, numbers, colours, and titles are there to indicate all the information that is there to be presented and the design is designed in a way that the user experience is easy and effortless. An example of visualization is the Figure 6 where the example results have been collected into four categories, separated the categories with an intuitive colour, counted the sum of every result in percentage within a month, and made the timeline and filter of what the results show visible. The filters of the visualization (at the top of Figure 6) will provide the users the possibilities to interact with the visualization. The interaction with the visualization helps the user to simplify the discussion, compare, and contrast the information gotten from the visualization (Card et al., 1999).

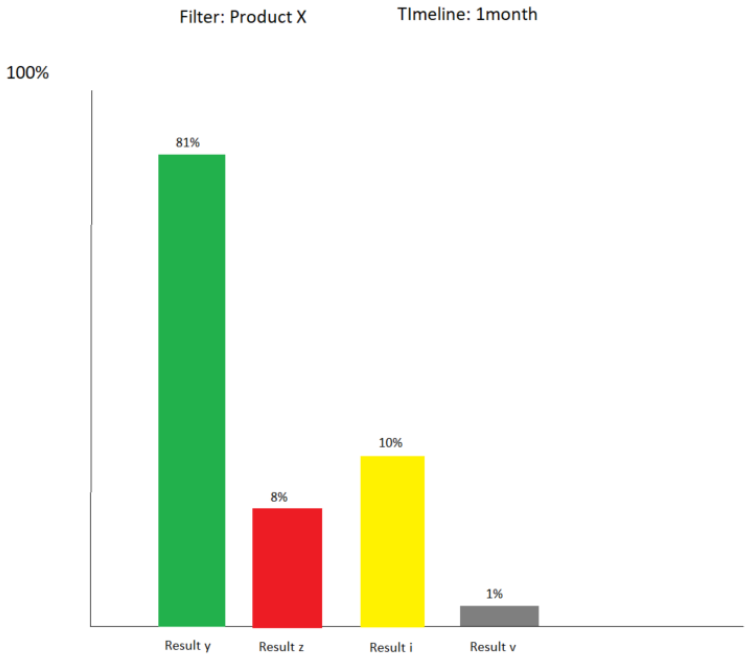

Figure 6 Example of the visualization of what the implementation artifact would look like.

6. Mental Model

The mental model is explained as something to model the characteristics and research output of the DSR process (Peffers et al., 2007). The mental model is supposed to help the research process and to know what to expect from it. As Peffers et al. (2007) explain, the mental model will help the researchers to conduct the process more effectively. The mental model is presented in this chapter by providing an overview of what can be expected from this DSR process. The construction of the mental model is part of step 3 called Design phase of this research as shown in Figure 5.

There are three key elements in the building of both artifacts, method artifact and implementation artifact, which are presented in the Figure 4. The first one is the Data that has already been saved to an index. This source index contains data that is unreadable for the visualization tools and there are huge amounts of data that it is not reasonable to retrieve it all. The second element is the data retrieval and summarization from the source data. This part retrieves only the necessary data about the tests such as a timeline of the tests, results, and amount of the tests done. This second element is the most complex one and will have the key research in it. The method was chosen by relying on previous research and information about the possibilities of Elastic. The limitations of Elasticsearch and the limitations of Nokia's own API are the crucial aspects of this research and will be the aspects that affect the result of this research. The third element is getting the new summarization of the data to be visualized and visualizing the data into a readable format. This part will rely on Jakob Nielsen's (1992) usability goals and Tufte's (1983/2007) design principles, and therefore the visualization will conclude a working design for the visualization.

In the visualization design, the top priority goals are learnability and the ability of infrequent users to return to the system without having to learn it all over. As Nielsen also says, all five goals cannot be a top priority in every design (Nielsen, 1992), therefore in this design there are chosen few that are kept at the centre of the design all the time. The learnability means that users may not use the system frequently, therefore the system should be so easy to use that the users do not have to learn the system every time they access it (Nielsen, 1992). In this design, there will most likely be many infrequent users because the topic of these management reports is specific. It is something that will not be in daily review for every user, but maybe for some can be. Hence, the design will follow the rules of high learnability.

In addition, the design will follow Tufte's (1983/2007) seven design principles which are presented in Table 1. Based on these principles the visualization will be presented in a way that it encourages user to compare the different results in the visualization and show the variation of the data clearly between the results. From both Nielsen's (1992) and Tufte's (1983/2007) design principles the visualization will get a direction for clear and understandable look. The colours, formats, labels and titles will be thought carefully to be able to achieve the most clarity that is possible to achieve.

The visualization of this retrieved data will represent the data patterns and nuances that can be found in the data. The visualization will present the data in a way that is comparable to other results and will show the user the data in an understandable format. Figure 6 represents the data visualization example of how the data would be presented and the visualization executed. There needs to be filters for users to choose from and for example product and/or timeline and see the data in graphs that are intuitively presented with colours, names of the columns, and numbers to show the data in relation to

percentages or total amounts of the data. The filters bring interaction between the user and the visualization. The filters help the user to compare and contrast the information, but also to simplify the discussion with the gotten information (Card et al., 1999).

7. Results

This part will present the results of the investigation. Both methods were researched, and all the limitations were investigated. This chapter is part of step 4 called demonstration, as the Figure 5 presents.

7.1 Method using only Elasticsearch

The first method investigated in this research was for using only Elasticsearch. The goal was to create a new index where there would have been all the needed data for the method artifact. The point of this method was to investigate if this would have been the best option for Nokia to use in the future rather than the new API that was created by Sonja Rytinki. For this method, at first, there were two potential Elasticsearch APIs called Rollup API and Transform API. These were presented as potential in a meeting with Nokia's experts either one of them used or possibly both of them being used. Third Elasticsearch API found later in the research process was Reindex API.

The first part presents the results of the Rollup API, the possibilities and limitations. There will be results if the Rollup API is even possible to use, and if it has been used at the creation of the method artifact. The second part presents the results of the investigation done for Transform API. There are also presented the success, possibilities, and limitations of the Transform API.

7.1.1 Testing the Rollup API

First in this research is the Elasticsearch Rollup API. The Rollup API was created for the source index that already existed. The Rollup API was built to retrieve data and summarise the data under a timestamp and update the data in 30-minute intervals. The Rollup API was promising for this research because it updated the data and retrieved data. There was latency with the Rollup API, but it was not so huge that it could not be used. The Rollup API was created by using the example of code example (5) that was found in the guide of Elastic's website (Elastic, 2023).

Even though the Rollup API showed great potential, it did not have the possibilities that would have been needed to create the management reports for this specific topic. The issue that was faced was the fact that some of the data was unindexed inside the source index. The Rollup API is created to only retrieve indexed data from source indexes (Elastic, 2023) Therefore, because of the fact that in this case there was unindexed data inside the source index that needed to be retrieved, the Rollup API could not do it. This led to the fact that Rollup API needed to be excluded from this project.

7.1.2 Testing Transform API

The next step was to investigate the Transform API. The Transform API had a source index either the Rollup API's new index or the Nokia's already existing source index. Both methods were tested and created the Transform API. The Transform API could have also answered the criteria of diminishing the data load by creating a new destination index where there would have been only the specifically chosen data fields. Transform API had also great potential by having a lot of different query parameters that could have been used. These are the "filters" that would have retrieved only the needed fields, by chosen

a timeline and grouping the data. For example, the Transform API supports about 20 different aggregations, groups the data with 4 different possibilities, defines the acceptable data age to the new index etc. (Elastic, 2023).

The Transform index was created to the point that the Transform API retrieved and grouped all the indexed data and printed in a correct form which Power BI could have been able to read. Despite all these possibilities, the Transform API could not handle the unindexed data and therefore the Transform API also had to be excluded from this research. At this point of the research, the clear issue was the unindexed data.

7.1.3 Testing the Reindex API

As it was noticed during the research, the unindexed data did not have the same possibilities as the indexed data. For this unindexed data, there is a command that can be used to retrieve the data. It is called the _source field. According to Elastic (2023), "the source field itself is not indexed (and thus is not searchable), but it is stored so that it can be returned when executing fetch requests, like get or search."

An example of the _source field in use is the code example (10).

```
GET my_index/_search                                                    (10)
{
  "size": 5,
  "_source": ["temperature.june.daily"],
  "query": {
    "match_all": {}
  }
}
```

Here in this code example (10) the GET command searches from an index called my_index and retrieves the unindexed data called "daily" which is now saved under a directory called temperature and a subdirectory called June. This will show the temperature in June and print the daily temperature. Here the June and daily are unindexed data and can be retrieved by using _source.

What was then discovered in this research by using the _source field with GET command and /_search as example (10), was that the unindexed data that needed to be retrieved was finally accessed and retrieved in a correct format which Power BI can read. This information that was searched can be implemented into a Reindex API to be able to create a new index which has the same outcome as this search command. By using a similar query to the code example (9), it is possible to save this needed data into a new index by using Reindex and also the _source elements. An example of this is the following code example (11) where there are combined the Reindex API and the _source field.

```
POST _reindex                                                           (11)

{

  "source": {

    "index": "source"

    "size": 5,

    "_source":    ["temperature.june.daily,    temperature.july.daily,
temperature.august.daily "],
```

```
    "query": {
    "bool": {
      "must": [
        {  "range": {"temperature.timestamp": { "gte": "now-70d"}}}
      ]
    }
  },
    "sort": [
      {
        " temperature.timestamp ": {
          "order": "desc"
        }
      }
  ]
  },
    "dest": {
    "index": "dest"
  }
}
```

In this code example (11) there are retrieved the daily temperatures from the months June, July and August. Assuming the query is done in August, the timeline of these daily temperatures are 70 days from the day of when the query is run. Also, the sort-element is there to arrange the output in descending order.

Now that the research proved that the method of creating a new index with just the needed data can be executed the next question is whether the new index can be updated automatically to contain live data in the index. The reindexing only puts the current data when creating the new index, as it is seen in the code example (11).

How this automatic data update can be solved is by using Python. Because the investigation excluded the Transform API and the Rollup API and left only the Reindex API as an option in the first method, there was needed an investigation of how the Reindex API could be used periodically. After investigating Elastic's official guide page (Elastic, 2023), there was only left to try to find a solution from other programming languages. Python language was used already in Nokia's own API. Therefore, Python was the option the research took immediately to investigate if Python could offer commands for solving the periodical reindexing.

First, when starting to use Python in this research there was a need for installing Elasticsearch client, using pip install as the code example (12) shows (Elastic, 2023).

Installing the Elasticsearch client to use by Python included the commands that were needed to connect to the Elastic client and to give commands for the indexes (Elastic, 2023). From there it was clear that the indexing commands are possible to run from Python and there would be possible that Python would be a solution for the periodical reindexing.

```
Python -m pip install elasticsearch                                    (12)
```

Python Elasticsearch client includes Elasticsearch helpers which could be imported to the Python file (Elastic, 2023). From these helpers, there was found the reindexing helper. The reindexing helper reindexes documents from the source document to another document with the given query (Python Elasticsearch client, 2023). After a deep dive into Python queries with OpenAI ChatGPT the Python schedule.every().hour.do() operation was found to perform the reindexing operation at the given time period (OpenAI, 2023).

7.2 Testing the Nokia's API

It was shown in the previous section that the reindexing can be performed periodically using Python. It is then needed to investigate if Nokia's own API created by Sonja Rytinki (2023) is a better option or not. Is the Nokia's own API more useful to use in Nokia's future purposes for different analyses rather than this Reindexing API in Elasticsearch? Nokia's own API is Python-based API. The research began by accessing the source code and reviewing all the Python files that were created for the API. What was found was that to use this API, it was needed to create a new use case file for this research results to be retrieved to the API. The use cases were written in Python in a way that Power BI could read the files, but the data needed to be transformed in the Power BI to be able to make the visualization (Rytinki, 2023).

What was also found from the source files was that the API did percentage counting on every file and counted percentages on the different results. It was already stated that the unindexed data that this research investigates cannot be used against all the queries. The unindexed data can only be retrieved from the source data. Therefore, this percentage counting was to be eliminated from these management reports in this new use case that would be created for the API.

But the most important finding that was already discussed as a concern was the lack of possibility to refresh the data effortlessly. The API does provide an update possibility, but it did not refresh it to chosen filters and time periods. This is not so convenient for the end user and therefore will need to be excluded as an option to provide these management reports which is the end goal of this design science research (DSR).

The issue raised by experts from Nokia, that some of the data are in a format that Power BI cannot read, was investigated also. Power BI was able to read the data when retrieving the whole source index. However, some of the aggregations used in Power BI advanced tools were not possible to use since some of the data in the source file was only saved and not indexed properly. Therefore, the same issues were faced in some ways that earlier with other procedures that data that is only saved and not indexed was harder to handle. What is important to understand is that this issue could have been solved somehow. However, it was not necessary to further investigate it because there were already found other issues that determined this method to be not as efficient as the method using Elasticsearch's reindexing operation and actually reindexing all of the necessary data to a new index.

7.3 Summary

The method that was chosen for this execution of the management report is the first method of creating a new index in Elasticsearch. Even though Nokia's own API has a lot of potential, the reason for this result is the lack of easy data refreshing possibilities for the end user. This method is not as efficient as it is hoped. The newly created index using Elasticsearch's Reindex API and Python's schedule elements can be set to automatically reindex i.e., in every hour, every twelve hours, or once a day and the data is refreshed to show correct filtering results.

One concern with the new index is memory outage issues. There is a huge amount of data that is being handled, even though the reindexing will only retrieve part of the source index to the new destination index. But, to keep the amount of data at a reasonable level, the new index will limit the data only to have data from the past year. With this limitation, the size of the index will be kept at a reasonable level.

8. Building the artifacts

This chapter describes both artifacts and how they were built as a result of this research. This chapter is part of the demonstration phase in this research as Figure 5 presents. Section 8.1 presents the building of the method artifact, what elements were used and why. The chapter also presents an example of the execution. Section 8.2 goes through the implementation artifact, how it was implemented and what were the necessary elements to be provided in the visualization. The chapter also provides examples of how the visualization was built and what elements it provides to the user.

8.1 Building the method artifact

The method artifact is part of the demonstration of this DSR project as the Figure 5 presented. In Figure 4 there is a representation of how the data is retrieved, going either the route of Method 1 or the route of Method 2. This research showed that Method 1 is the better option in this case and therefore the method artifact contains the Method 1 option.

Chapter 7 presented how the Reindex API can be used to retrieve data from a source index to a destination index. The method artifact is built by using this function. However, only Reindex API is not enough. Therefore, Python programming language needed to be taken into the mix. To explain how the method artifact was built is by a few key elements. Python Elasticsearch client where the Elasticsearch helpers contained the Reindex API. Python also provided the schedule.every().hour.do() where the periodic reindexing was possible to implement. An example retrieved from OpenAPI ChatGPT (2023) is the code example (13).

```
Import schedule                                                    (13)

import time

from elasticsearch import Elasticsearch

from elasticsearch.helpers import reindex


# Connect to Elasticsearch

client = Elasticsearch([{'host': 'localhost', 'port': 9200}])


# Specify source and destination indices

source_index = 'source_index'

destination_index = 'destination_index'


# Define the date range for the reindexing operation
```

```
start_date = "2023-01-01T00:00:00"
end_date = "2023-01-15T23:59:59"

# Define the reindex query with a date range filter
reindex_body = {
    "source": {
        "index": source_index,
        "_source": ["field1", "field2"],
        "query": {
            "range": {
                "date_field": {
                    "gte": "now-365d"
                }
            }
        }
    },
    "dest": {
        "index": destination_index,
    }
}

# Function to perform the reindex operation
def perform_reindex():
    print("Performing reindex operation…")
    response = reindex(
        client=client,
        body=reindex_body,
        timeout="5m"  # Adjust the timeout value as needed
    )
    print("Reindex operation completed.")
```

In this code example (13) first there are imported the necessary libraries and operations to be able to perform all the necessary queries. After that, there is the connection to Elasticsearch and the definition of the source and destination indexes. Then there is the reindexing operation body with "_source" limiting only the wanted fields from the source index and the "range" limiting the data to be retrieved only from now to year-old data. After that the def perform_reindex(): performs the actual reindexing. To keep the data updated, there needs to be a program to reindex the new day and add it to the existing index. This is done similarly as the code example (13), but the range query includes only one day "now-1d" rather than 365 days. To keep the index always only at the size of 365 days, the oldest day needs to be deleted at the same time as the new day is added. This can be done as the code example (14) shows retrieved from Open AI ChatGPT (2023)

```
def delete_data(destination_index):                                    (14)

    print("Deleting oldest date...")

    days = 365

    threshold_date = (datetime.now() - timedelta(days)).isoformat()

    result = client.delete_by_query(

        index=destination_index,

        body={

            "query": {

                "range": {

                    "time_field": {

                        "lt": threshold_date

                    }

                }

            }

        }

    )
```

Here in the code example (14) the "days" define the number of days included in the destination index and the "threshold_day" defines the day that will be deleted. Then the day will be deleted in the "client.delete_by_query". In the code example (15) the schedule.every(24).hours.do can be set to schedule the reindexing and old data deletion to be set to loop on every 24 hours until stopped manually. With this function, the data will be updated automatically, and users do not have to do the updating manually.

```
schedule.every(24).hours.do()                                          (15)
```

While building and testing this method artifact, there were found also issues with size limitations. It was learned that Elasticsearch accepts only a limited amount of data to be handled in the queries and therefore some of the data was being dropped (Elastic, 2023). This was noticed when the data was brought to Power BI and compared to the Elastic

index. The Power BI retrieves the data by Elasticsearch queries and therefore the Elasticsearch limit of 10,000 hits was faced.

This led to the fact that the index needed to be cut down to pieces in some way to the Power BI. This was possible to be done in Power BI Advanced Editor and editing queries to retrieve smaller parts of the index. The first method that was tried was counting the data with aggregations which can do different counts and summaries of data (Elastic, 2023). This method did not work with the index structure that was handled. This method had issues with values that had identical names. The values that had the same name were grouped in a way that the group was counted as one and not all the values it had inside. Therefore, the results were incorrect, and this method was excluded.

Something that was found working was scripting in the JSON queries. These scripts are able to take one month, week, day, or even hour from the index and in that way slice the index into smaller pieces (Elastic, 2023). However, this needed to be figured out how many "slices" the index needed to be sliced for the Elastic queries to be able to handle it. An example of a piece of this query that could be used in the Power BI advanced editor to slice the index into months is the code example (16).

(16)

```
{
    ""size"": 10000,
    ""query"": {
        ""bool"": {
          ""must"": [
            {
                ""script"": {
                    ""script"": {
                        ""source"":
""doc['mytimestamp'].value.getMonthOfYear() == params.month"",
                        ""params"": {
                            ""month"": 12
                        }
                    }
                }
            }
        ]
    }
  }
}")
```

Here in this code example (16) is formatted for Power BI to be able to read it. This example is only one part of what the Power BI needs to be able to read the data, but here the query with its script slices the index into one month. The month is December due to the “params.month” being set to value 12.

8.2 Building the implementation artifact

After the method artifact is created, the implementation artifact is possible to be created. How is this implementation artifact built then? It is created using Power BI as Figure 4 presents. In Power BI the index is retrieved to the Power BI tool using the Elasticsearch local host address and port but also the new index name. The Power BI retrieves all the data from the newly created index and the visualization of the data can be created.

What the visualization needs to have, are interactive elements so that users can adjust the views based on their needs (Card et al., 1999). One such interactive element is a timeline, which allows the user to adjust the wanted period. This is executed as shown in Figure 7. Another necessary element was different filters to filter the data with important options such as specific products or teams. Figure 8 represents the filter option. Something that was required for some graphs was to show the data daily for example by week or by month period and average values for creating a goal and something to compare the results.

What was kept in mind while creating these visualizations were Nielsen’s (1992) heuristics that were used for evaluation and target usability. The visualization needed to be learnable and created in a way that infrequent users know how to use them when they access the visualization. To this, of course, Power BI’s existing elements that were used had already good UX and UI. However, the colours, titles, placement of the elements, etc., needed to be adjusted in a way that the user would understand what the elements are and what they can do. What was important to keep in mind was that there were a lot of options to put into the graphs. However, it was not the best option to try and fit several different filters into one graph. This would have looked too chaotic, and the user experience would have suffered.

It needed to be thought, out which of the filters are the most important, what can be left out, if none, could another graph be useful. Figures 9a and 9b represent two examples of graphs that were created. The elements presented in Figure 7, Figure 8, Figure 9a and Figure 9b are all individual elements which will not be ever presented by themselves. The finished management report will include all or some of these elements based on the need for the management report. In this research, there were created nine management reports, from which only a few will be published for actual use. This choice is not included in this research.

Figure 7 Example of the timeline that is provided to the user to adjust the way they want.

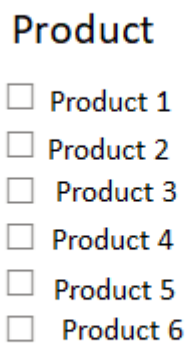

Figure 8 Example of the Filters that can be chosen by user.

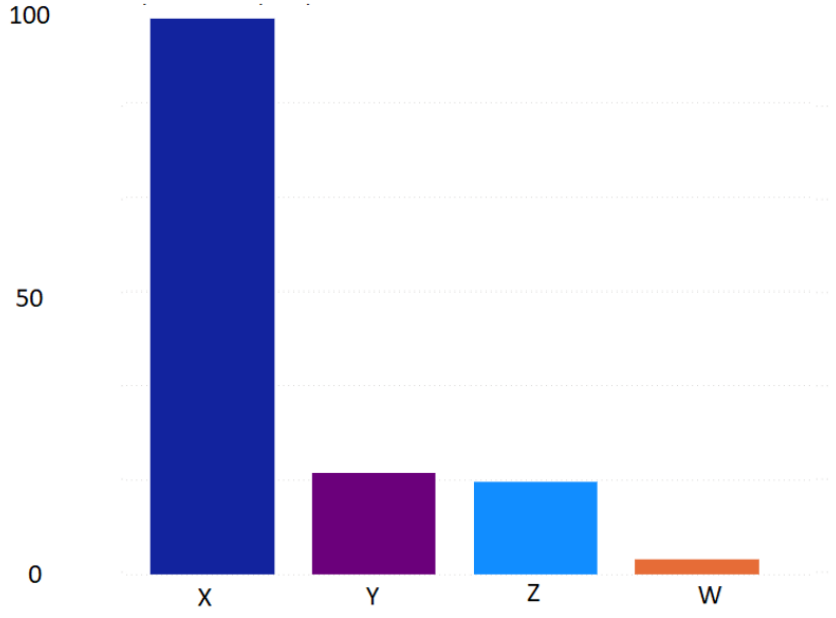

Figure 9a Example of the graphs that were built to represent results and the total count of the results.

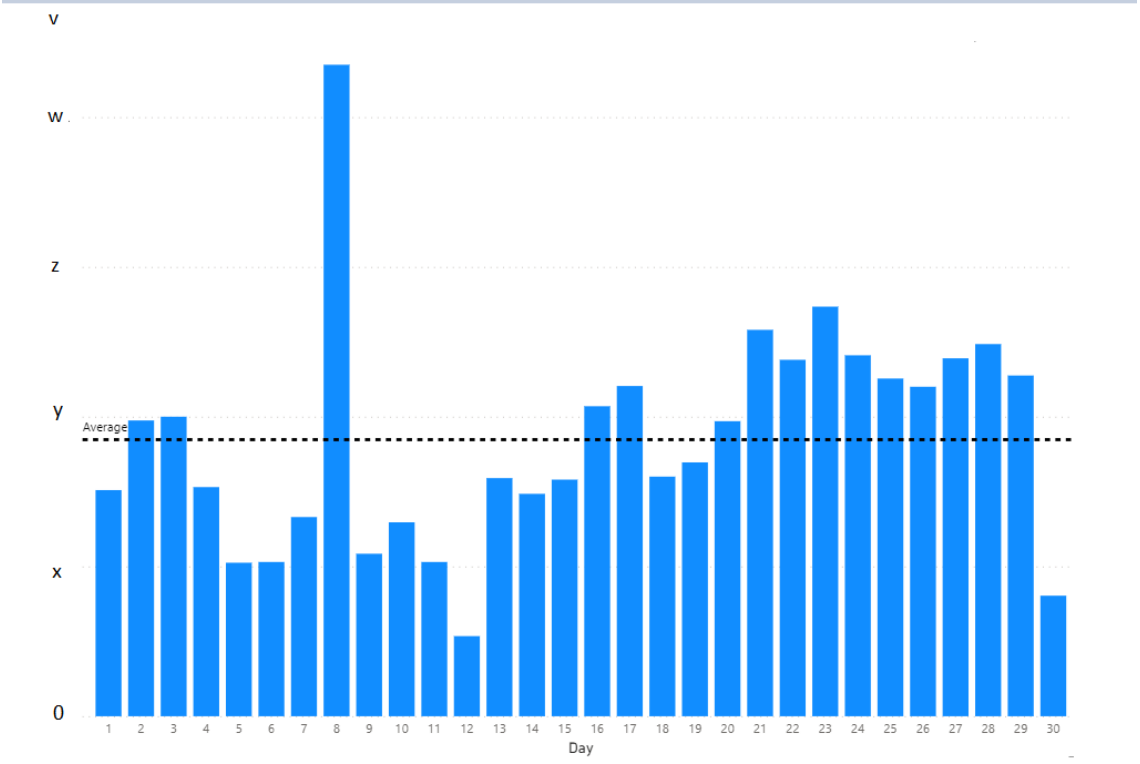

Figure 9b Example of the graphs that were built to represent results daily from chosen month.

9. Evaluation

This chapter covers the evaluation of both method artifact and implementation artifact. This chapter is part of the Evaluation phase of this research as can be seen in the Figure 5. The evaluation is done using Sonnenberg's and Vom Brocke's (2012) evaluation model for the method artifact and the implementation artifact is evaluated by Nielsen's (1992) usability goals model. There is also evaluation done by users, user feedback, which comes from Nokia managers and staff who are the end users for these artifacts.

9.1 Evaluation of the method artifact

The Sonnenberg's and Vom Brocke's (2012) evaluation model starts with the ex ante evaluation phase as the Figure 10 presents, where there are evaluated the problem definition and the design of the artifact. A similar evaluation approach was used in this research. The problem definition is evaluated in the motivation and problem definition chapter of this research (Chapter 4). This part needed to provide a justified problem definition, research gap, and justified design objectives (Sonnenberg & Vom Brocke, 2012). The research gap was discovered to be the lack of management reports for some areas at Nokia and the unknown method by which it should be executed. The management reports that were missing were defined as reasonable issues to be solved and something that should be researched. The new management reports would be beneficial for following performance and success. The design objectives were defined as the two methods that needed to be researched to find out the best option to execute the creation of these management reports (Chapter 5). This concludes the Figure 10 Evaluation 1 part.

After this is the design phase of the research which is executed in the Design Process and Mental Model chapters. The evaluation of the method artifact's design which was done in the Results part (Chapter 7). In the Results chapter the design options were evaluated and explained the options what will not work and why they do not work in this research. Therefore, it resulted in an explanation of what method does work in this execution of the method artifact and at the same time the justified design method was explained.

The evaluation of the design ended with the choice of the Elasticsearch index creation method and the use of the Reindex object and Python scheduler to execute the management reports. The design evaluation concluded that the resulting method was the most efficient one and had the best user experience and efficiency in having automatically updated data. The evaluation of the design contained an explanation of why the second method was not sufficient with the parameter update from the API to get fresh data. This was also at the same time the specification of the design. This was the end of Evaluation 2 part seen in the Figure 10.

After the evaluation 1 and evaluation 2 parts were completed, the ex ante evaluation was finished. The method artifact and the construction of the method artifact could be started. Even though the ex ante evaluation phase is completed it does not mean that the ex ante evaluation phase cannot come back to. As the Figure 10 represents the evaluation process is iterative and there are two ways of the process, forward and backward. Therefore, if there is a need the evaluation can go a bit backwards to complete the product as well as possible.

After the construction of the method artifact, the artifact could be built as a prototype. The prototype construction was described in the chapter 8 on building the artifacts

(Chapter 8) and there was an example of how the method artifact was built. The evaluation of construction phase needed a prototype which could be evaluated to see the applicability of the prototype (Sonnenberg & Vom Brocke, 2012). The prototype was then set to run locally on a computer to see the performance. The prototype reindexed hourly and created the test index. The prototype was set to see if the Python schedule.every.hour().do() actually executes the reindexing command hourly and if the reindexing is successful. The prototype was seen to successfully create a new index and update it every hour and the index containing the needed data was the proof of the prototype's applicability. This was also the end of Evaluation 3 which is visible in the Figure 10.

When the prototype was successfully seen as applicable to this research, the use of the artifact was then the next step. The last evaluation of Sonnenberg's and Vom Brocke's (2012) evaluation model is after the use of the artifact. Therefore, the last evaluation needed a completed artifact, and its output was to see the usefulness of the artifact in a natural setting (Sonnenberg & Vom Brocke, 2012). The prototype was set to reindex every 24 hours and therefore the new results were updated to the destination index every 24 hours. From this, Power BI could get data for the implementation artifact, in other words, data visualization. This was the Evaluation 4 which is presented after the Construction phase of Figure 10.

This next evaluation identified two issues. The first major issue that was found, was that the index created duplicates of the index. Therefore, after a week, there were 7 large indexes and after a month there would have been 31 large indexes. This was a huge issue and would have caused major problems if it was not noticed. What was done to solve this problem, was adding deletion of every index with the same name before the new reindex period. After this, the Python program does iteration by first deleting old indexes with the same name, then performing the reindexing, saving the index, and then waiting for 24 hours to do this all over again until stopped manually. However, although this was a successful solution it was not seen as the most optimal one.

The second issue was faced due to the Elasticsearch query size limits. It was realized that Power BI could not get all the data from the past year and dropped data randomly. This was due to Elasticsearch's query limits that are used to retrieve data to Power BI. The solution for this was to use Power BI advance editor to retrieve only part of the data to one graph. For example, the data could be brought month by month or week by week to the Power BI and therefore there would be a graph for every month or a week. The Elasticsearch queries size limit was a difficult issue, even though the new index's size was only part of the source index's size.

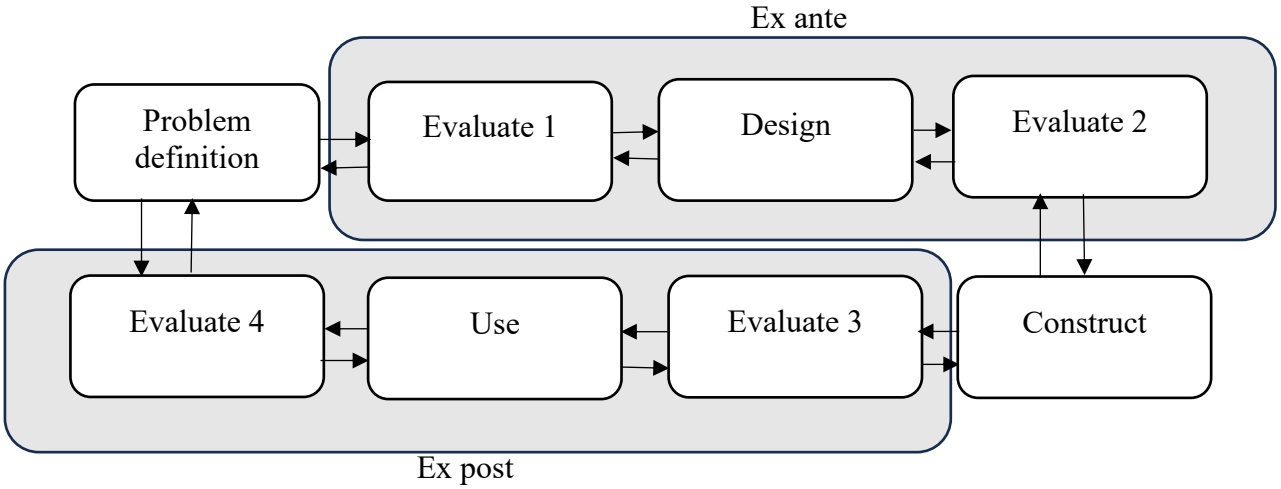

Figure 10 The evaluation process (Sonnenberg & Vom Brocke, 2012).

9.2 Evaluation of the implementation artifact

The implementation artifact is evaluated by the researcher based on Nielsen's heuristics (1992). From Nielsen's heuristics (1992), two top priorities were chosen, 1. learnability and 2. that the infrequent users come back to use the system without having to learn how to use it all over again. Also, the other usability heuristics are gone through in this evaluation which are, efficiency of use once the system is learnt, frequency and seriousness of human errors, and subjective user satisfaction (Nielsen, 1992).

The learnability of the visualization is evaluated to make the visualization as easy to learn as possible. The visualization includes representative titles, numbers, and intuitiveness. The visualization has titles that represent the visualization as well as possible. The X-axis and the Y-axis are named such as "Total count of test results" and "Test results". The filters that the user can use are named and presented in a way that all the filters are visible without any effort from the user. The filters are also titled in a way that users will understand right away what they are filtering such as "Timeline". The visualization will also include graphs like there are presented in Figures 9a and 9b. The columns in the graphs are titled to represent what they are. Also, the figures have numbers to represent the graph's size in relation to numbers. If the Y axis last number is 100 (like the Figure 9a) and all the columns are under that, it means all the result's total amount is under the value 100.

The colours are default colours that are used in other graphs as well to keep similarity and users' intuitiveness. Also, therefore the comparability is easier for the user if all of the graphs are coloured similarly. The default colours are picked for the columns to represent intuitively the results for example errors are marked with orange because it is closest to red which is usually colour for faults and errors. The blue colour is for success because it does not indicate anything negative and is not usually used for any errors or faults. Therefore, colours between these two, orange and blue, are used for partly successful or not complete results. For example, colour purple presents the result of partly correct, which means it has faults and success. The colour is chosen for these results because it can be considered as a mixture of orange and blue.

Also, the visualization uses familiar elements such as lines for setting the timeline which are generally used, date numbers to define what the timeline is for, filter options with blank squares and chosen filters with black squares which are familiar indicators in general for choosing. Therefore, the visualization will be intuitive to use and therefore easy to learn. Also, with this intuitiveness and familiarity the heuristic of infrequent users coming back to use the system without having to learn how to use it all over again is also solved. The graph will be intuitive to use, and the titles and numbers will explain themselves to the user which will make it easy to remember and understand what those are representing.

The other heuristics, efficiency of use once the system is learnt, frequency and seriousness of human errors, and subjective user satisfaction, were thought about when creating the intuitive visualization which is effortless to understand. Intuitiveness with familiarity, titles, colours, etc., affect also the heuristic of the remaining heuristics. The efficiency of use once the system is learnt is exactly solved with a clear design and understandable objects and titles. The frequency and seriousness of human errors-heuristic are solved by clearly designing and placing the graphs under the correct name and colour, making the filters visible to see what is chosen and when, showing clearly what the filters' values are, and giving the user a possibility to clear the filters to easily go back if the wrong filter is chosen. Also, if the colour cannot be named in the graph there needs to be an explanation of the colour shown to the user with a dot or box of the colour and the meaning of it next to it or under it. The subjective user satisfaction heuristic is achieved with the effortless design to look at and understand. Also, the elements need to be placed in a way that they can be reached easily and the text with reasonable font size.

9.3 Evaluation from users

Throughout the research process, there were evaluations gotten from users. The comments were related to users' point of view, what is needed and what is not. The method artifact first evaluation by the user resulted in two main points. The first point that was raised was related to data fields that are important and relevant to be reported to managers. It was identified in Evaluation phase 1 (see Figure 10). This was important due to the fact, that the new index was created to retrieve only the data that is relevant to be created in the management reports but to also make possible future developments with the index. Another thing that was raised was the size of the index and the pace of refreshing the data. This was pointed out in the Evaluation phase 2 (see Figure 10). Those were thought of a few times and ended up being 365 days of data and an update period of 24 hours. This was a discussion between the researcher and Nokia's experts who would also be users for these management reports. The result was decided to be able to limit the size of the data to a reasonable level.

Another issue seen by users was the optimization of the program. The reindexing of the whole 365 days every 24 hours would cause unnecessary operations and load. Therefore, due to this comment, the program was modified in a way that the reindexing for the past year was done only once, and after that, every 24 hours there was reindexed only the missing data from the past day, and the oldest day were deleted. These operations, adding the new day and deleting the oldest, were set to run every 24 hours to decrease the unnecessary operations and load that the first solution caused. This issue was recognized after the construction phase in the Evaluation 3 (see Figure 10).

After the use of the program there was the last evaluation phase, Evaluation 4 (see Figure 10). However, this only resulted small changes to the comments of the code, defining

better name for the new index, and a discussion about which time zone the code should be set. Hence, in the Evaluation phase 4, there were no more major changes, but were fine-tuning. In the Evaluation phase there were recognized few major improvement possibilities, but those were decided to be left out of this research and define them for future research. These are presented in the Chapter 10.3.

The implementation artifact's first evaluation was related to the data that it showed and in what format. Power BI did not present all of the data in a correct format and those were decided to be left out of this research and left to the future development. What was also commented on was the fact that the Power BI queries could be modified to exclude some of the fields and therefore limit the data fields to be able to retrieve as much data as possible. The second evaluation of the implementation artifact noticed few missing graphs. Other already created graphs were passed and those achieved requirements. Therefore, the missing graphs needed to be created and evaluated. After passing the missing graphs, the implementation artifact was also completed.

10. Discussion

This study aimed to compare two methods, their possibilities, and their limitations. From the comparison the goal was to choose the right method to create management reports for the data retrieval and visualization. The management reports' goal was to present relevant information that was still missing and present it in an understandable format. Because this research included the method comparison and the data visualization, those created two different artifacts. These artifacts were method artifact and implementation artifact. This research was done using the Design Science Research (DSR) method. The research answered the research question *Which is the most efficient way to retrieve and visualize big data when comparing two different methods?*

The issue in these management reports that were created in this research was that they did not exist yet. Some management reports were but the goal of this research was to find out what is the best way to do these missing management reports and show the way for future developments. The research covered multiple limitations for these specific management reports and the route to creating these reports was changed many times. This means, that many of the possible ways of creating these reports ended up excluded due to limitations that were found during the investigation. These were for example Rollup API having a lot of potential but could not handle unindexed data and Nokia's own API having huge potential but was excluded due to issues with data refreshing.

All these limitations did lead the investigation further even though it seemed that every single one of the possible APIs and features that were provided by the two different methods had crucial limitations. The issue was always the unindexed data or the lack of updating possibilities. For example, Rollup API had limitations with unindexed data, and Reindex API and the API created in Nokia had a lack of automatic update possibility. The research ended up going in a direction which was not anticipated in any way. The Reindexing was possible to do with Python programming language and Python was the key to the periodical update for the new index. What was meant by the anticipated results was the fact that Python was not considered to be used at all at the start of the research for the first method.

This automatic periodical update was a crucial element because old data would not be useful for a long time. Without the update the data would always be the same. Big data analysis is crucial for many companies and that can be life-changing for the companies' success (Mikalef et al., 2019). But to achieve all these possibilities in action, the data analysis needs to be accurate and up to date. Also, for this purpose that this research aims to achieve, the data is extremely crucial to be up-to-date and accurate. There are no real benefits of managers analysing the same old data over and over again. It is crucial to be up to date.

When handling big data, the size is always an issue on the table. Tools, storage, etc. cannot handle everything and the size becomes quickly a problem. In this case, even though the destination index was created to only take the necessary data and leave out everything else, the size of the destination index was too big for the Elastic queries to handle. Even though there was a solution found to bring the index into Power BI in pieces with smaller Elasticsearch queries, the data size always can grow even more. Then there is a risk that the pieces that are now created become again too large. The index will handle the growing amount of data, but Elastic queries might not. Therefore, it needs to be monitored if the amount of data is known to grow.

What is also important is the understandability of the data visualization. The implementation artifact aimed to visualize data in an understandable format and therefore create an easy and effortless user experience for users. The data visualization followed Jacob Nielsen's (1992) heuristics to achieve better user satisfaction. Also, the data visualization aimed for an intuitive user experience for creating effortless use. The visualization included filters and adjustable timelines for the user to adjust the visualization for their needs, but to also change the selections to see other results as well. All these results are simplified to Figure 11 to clarify all the findings that were made during this research.

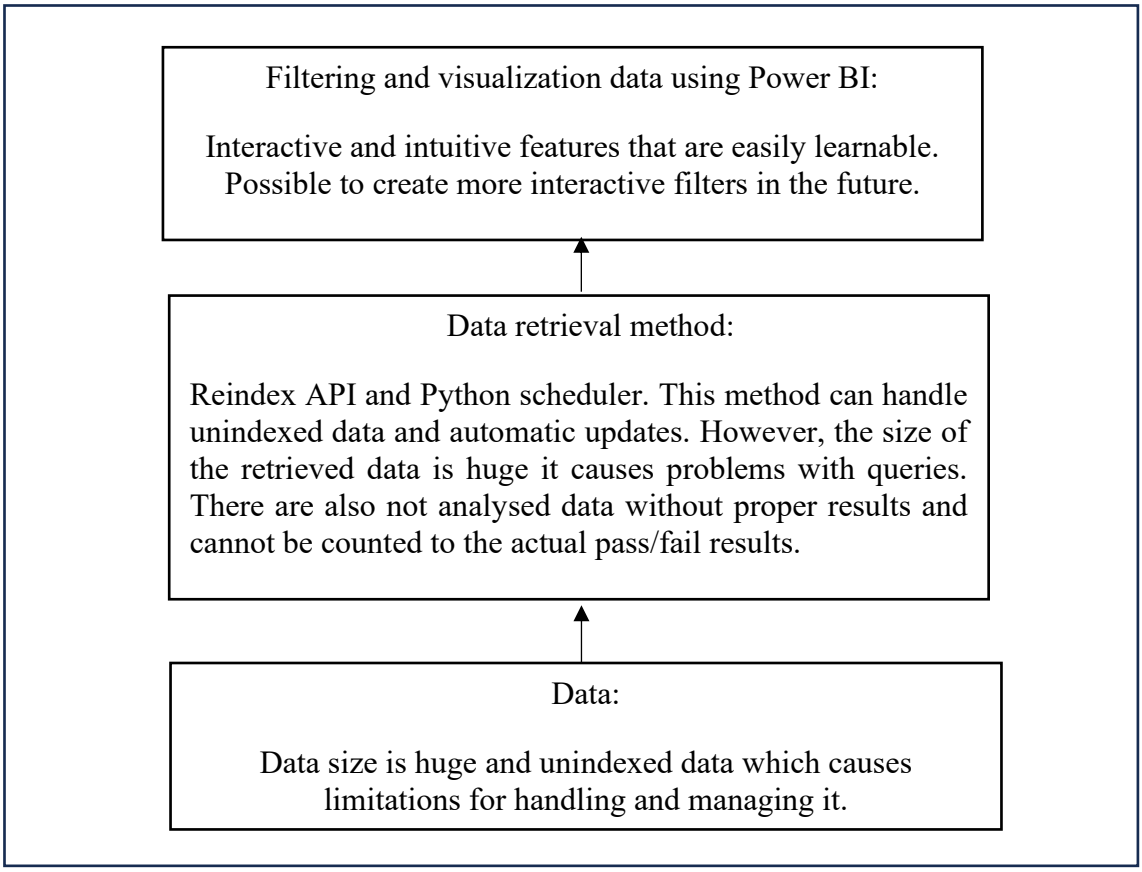

Figure 11 Summary of the research results.

10.1 Implications to theory and practice

As Mikalef et al. (2019) said we are now living in the "Age of Data", referring to the fact that there is data produced and gathered massively every second. This has directed organizations to the point where they have realized that the gathered data could have massive benefits. Organizations have started to invest in projects where the benefits of the produced data can be made into a form that creates value (Mikalef et al., 2019). This is one of the sources in this research that shows the benefits of big data and emphasizes the importance of big data retrieval and analysis. This research is more of a concrete example of how this data retrieval, visualization, and analysis can be performed and what issues will be faced. Hence, its main point is to explore and investigate how these processes are actually done, and not just to show the value of the big data.

Findings from the research show that handling big data is not easily done, and the size and form of the big data cause the most issues. The big data is scattered and comes with large data flows, which is what limits the possibilities of working with real data. As the previous literature shows, for example, Sagiroglu's and Sinanc's (2013) article, the big data's issues are storage, analysis, and visualization. This research is a concrete example of how all of these difficulties were faced with different tools and methods. However, these difficulties were also solved in this research. This research adds valuable examples of how big data handling is causing difficulties and bringing even surprising limitations. It has been proven and discussed how big data handling is not easy, but this research provides a clear example of what issues can be faced and if there are solutions or not.

The research discovered limitations. Hence, there was found the need to think about what enough data analysis is, and should resources be used for future development of something even more complex. In this research, it was found out that even more data could be brought to the new destination index and therefore get even more benefit from the data. However, the research also discovered limitations with the Elastic queries and counting possibilities. The large amount of data does not fit well with the Elastic queries and Power BI. The possibility of bringing the data to Power BI in a reasonable format and not exhausting the Power BI would be jeopardized. Another possibility would be counting the data before bringing the data to Power BI. However, there are values with identical names and therefore the counting would not give answers in a needed way. This brings an issue that the managers need to consider. If this is worth investigating further in the future even with the risk of not finding a solution. Managers need to consider if the benefit from the not included data is worth starting a new investigation.

10.2 Limitations

Even though the method was successfully chosen and implemented, there are limitations. One of the limitations that was noticed was the missing data in the source file. There were null values visible and not analysed data. These data results affect the overall data results, but to indicate these limitations, these results were presented as well. These limitations of the source files missing data are something that this research cannot resolve. Therefore, the results were shown and indicated to the user that there are possible null-values and also data that are not analysed. Meaning those will not have results such as pass or fail.

Another relevant limitation was the size of the new index. As the data flow is huge, the index that was created during this research needed to be thought precisely. The size limitation was decided to be limited to contain only year-old data and not any older. There the size could be limited but the decision created a limitation for the management report to only include data that is max one year old. There is also size limitation with the Elasticsearch queries that were already faced in the research process. In the future, the amount of data could grow even more. This could create new problems and the queries that retrieve the data to the Power BI, could start dropping data again. Even though the data is brought to the Power BI already in small slices, there is still the risk that the data grows too much, and the Elastic query size limitations are faced again. Elasticsearch query size limitations are always relevant topics for concern and limitations that will be faced now and in the future.

What was also discovered was that some of the data from the indexes came as lists and single values inside a column. Therefore, there was left a situation where the column contained single values and lists, which were not able to be extracted anymore. This was

a clear limitation to be able to create filters from that column, and in this case the column was left out completely to be solved in the future.

10.3 Future work

The created artifacts, method artifact and implementation artifact, can be developed further and would be preferred to be developed in the future. There were already found few important future development targets that were decided to be excluded from this research. First was that there could be data retrieved from two different indexes and combined the results and create even more specific results. The two indexes could be combined with different ways possibly to create two destination indexes and combine them in Power BI or however would be seen as the best option. These two indexes could make the management reports even more detailed and give data that has not been retrieved yet to be visualized in management reports. This would be a good way to improve the data analysis even more and get to squeeze even more benefits from big data.

Another thing that could be developed in the future is adding more filters to the data visualization; more options to filter the data to be visualized and create even more advanced visualizations. This would bring even more interaction between the user and the visualization. By adding more possibilities to the visualization, the visualization will provide more simplifying elements and help to understand and contrast the visualization (Card et al., 1999). The implementation artifact is limited to only creating a few visualizations which are management reports for those specific topics. But it is already possible to create more, and different visualizations for other management reports. The method artifact was created in a way that would make additional management reports possible, even though this research would not cover them. The new index that is created does have data that makes it possible to create other relevant management reports, these possibilities were left to the future work.

In addition, as the data flow is huge and there is a lot of data that is gathered, the size of the destination index will always be something to consider in the future. This research does not cover all the specific data that would be useful to visualize in the future. However, the additional specific data will grow the dataflow so large that it creates issues again with the Elastic query size limits. The data can be easily retrieved into the destination index created in this research. The issue is that the amount of data will be so huge that there is a risk of exhausting the Power BI. One option would have been counting the data before bringing it to Power BI. However, the data is complex and has identical values which causes the issue of Elasticsearch counting methods not working as needed. Therefore, in the future there should be researched if there is a way to grow the amount of data and still get the data visualized with Power BI.

11. Conclusion

This research is done based on the Design Science Research (DSR) method and answered the research question *Which is the most efficient way to retrieve and visualize big data when comparing two different methods?* The research investigated two methods, their possibilities and limitations. The methods were: Method 1. Using only Elasticsearch retrieve data and create new index and Method 2. Using API created in Nokia by Sonja Rytinki (2023) to retrieve data. The data retrieved using one of these methods needed to be visualized as well. This researcher created two different artifacts, a method artifact and an implementation artifact. The method artifact contained two different methods of data retrieval and the choice of the best one. The implementation artifact contained the data visualizations. Those visualizations were created to be necessary management reports that were still missing.

Method 1 had different possibilities for solving the data retrieval and index creation that were investigated. The investigation resulted in limitations with unindexed data that needed to be retrieved. Even with these limitations, the Reindex API was found to be able to solve data retrieval and index creation. The research found that with Python queries there was also the possibility to update the index periodically which concluded the fact that method 1 can be used for this research.

Even though there was the conclusion of Reindex API and Python queries to be a way to finish this research, Nokia's own API needed to be investigated if it was possible to be used as well and if it was a better fit for this purpose. Even though Nokia's own API had great potential it was found to lack efficient data refreshing as needed and therefore was excluded for this purpose. This concluded the fact that the first method was the correct one to be chosen for this research. Therefore, the data retrieval was executed using the first method.

The size of the index that was created using the chosen method, caused issues with Elasticsearch query limitations that are used for retrieving data to Power BI. Elastic queries can handle only a certain amount of data and the index exceeded the limitation. This changed a lot how the graphs could be built because the data was decided to be brought in pieces to Power BI. The graphs eventually were a representation of those pieces and therefore needed to be changed to look a bit different than first imagined. Even though, the graphs were able to be created to visualize the needed data as wanted.

The data visualization, in other words, implementation artifact, was created using the Power BI tool. The data visualization was created to follow the collective design of Nokia's other visualizations. The goal was to not depart from other visualizations too much to keep the new visualizations in line with others. The design was made to be intuitive, understandable, and clear. The visualizations included filters and the relevant information as those were required from Nokia. The design was created to keep in mind Nielsen's heuristics (1992) which were chosen as an evaluation method for the implementation artifact.

The artifacts were evaluated separately to be able to implement the best evaluation methods for both of them. The method artifact was evaluated by using Sonnenberg's and Vom Brocke's (2012) evaluation model with ex ante and ex post phases. The implementation method was evaluated by evaluating the Nielsen's heuristics (1992) implementation of the visualizations. From the heuristics, there were two top heuristics to be achieved to have a clear and reasonable design for this purpose. The two top

heuristics are learnability and that the infrequent users come back to use the system without having to learn how to use it all over again. These were chosen to fit the best to the purpose of these visualizations.

There are different ways and reasons to execute data retrieval and visualize the data in management reports. One of the biggest reasons is the benefit of analysing big data. Therefore, companies should really consider if there is still some data left to be analysed and made as a report. This research is a clear example of how this need for data analysis was noticed and put to work and how it should be executed to become a management report.

References

Akdogan, H. (2015). *Elasticsearch Indexing*. Packt Publishing Ltd.

Andhavarapu, A. (2017). *Learning Elasticsearch*. Packt Publishing Ltd.

Bajer, M. (2017, August). Building an IoT data hub with elasticsearch, Logstash and Kibana. *In 2017 5th International Conference on Future Internet of Things and Cloud Workshops* (FiCloudW) (pp. 63-68). IEEE.

Berinato, S. (2016). Visualizations that really work. *Harvard business review*, 94(6), 93-100.

Biehl, M. (2015). API Architecture (Vol. 2). API-University Press.

Card, S.K., Mackinlay, J.D., and Shneiderman, B., (1999). *Readings in Information Visualization - Using Vision to Think*. Chapter 1. Information visualization (pp. 1-34.)

Davenport, T. (2006). Competing on Analytics. *Harward business review* 84(1), 98

Elastic (23.7.2023) *Elastic* https://www.elastic.co/

Ferrari, A., & Russo, M. (2016). *Introducing Microsoft Power BI*. Microsoft Press.

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design Science in Information Systems Research. *MIS quarterly*, 28(1), 75-105. https://doi.org/10.2307/25148625

Hevner, A., & Chatterjee, S. (2010). Design research in information systems: theory and practice (Vol. 22). *Springer Science & Business Media*.

IBM (Publish time unknown) *What is data visualization?* Retrieved in 16.11.2023 from https://www.ibm.com/topics/data-visualization

Kaisler, S., Armour, F., Espinosa, J. A., & Money, W. (2013, January). Big data: Issues and challenges moving forward. *In 2013 46th Hawaii international conference on system sciences* (pp. 995-1004). IEEE.

Keim, D. (2002). Information visualization and visual data mining. *IEEE transactions on visualization and computer graphics, 8*(1), 1-8. https://doi.org/10.1109/2945.981847

Kononenko, O., Baysal, O., Holmes, R., & Godfrey, M. W. (2014, May). Mining modern repositories with elasticsearch. *In Proceedings of the 11th working conference on mining software repositories* (pp. 328-331).

Kuć, R., & Rogoziński, M. (2015). *Mastering Elasticsearch*. Packt Publishing Ltd.

March, S. T., & Smith, G. F. (1995). Design and natural science research on information technology. *Decision Support Systems*, 15(4), 251-266. https://doi.org/10.1016/0167-9236(94)00041-2

McAfee, A., Brynjolfsson, E., Davenport, T. H., Patil, D. J., & Barton, D. (2012). Big data: the management revolution. *Harvard business review*, 90(10), 60-68.

Mikalef, P., Boura, M., Lekakos, G., & Krogstie, J. (2019). Big data analytics and firm performance: Findings from a mixed-method approach. *Journal of Business Research*, 98, 261-276.

Nielsen, J. (1992). The usability engineering life cycle. *Computer* (Long Beach, Calif.), 25(3), 12-22. https://doi.org/10.1109/2.121503

OpenAI. (2023). *ChatGPT* (GPT-3,5 version) [English]. https://chat.openai.com/chat

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of management information systems*, 24(3), 45-77.

Python (Publish time unknown). *Python*. Retrieved in 27.10.2023 from https://www.python.org/

Python Elasticsearch client (Publish time unkown). *Python Elasticsearch client*. Retrieved in 13.11.2023 from https://elasticsearch-py.readthedocs.io/en/stable/helpers.html#reindex

Rouse, M. Data Retrieval (16.11.2023). *Techopedia* What is Data Retrieval? - Definition from Techopedia

Rytinki, S. (2023). *Summative analysis as a tool for prioritizing development efforts* (Master's thesis, University of Oulu). Retrieved from http://urn.fi/URN:NBN:fi:oulu-202305161775

Sagiroglu, S., & Sinanc, D. (2013, May). Big data: A review. *In 2013 international conference on collaboration technologies and systems* (CTS) (pp. 42-47). IEEE.

Sonnenberg, C., & Vom Brocke, J. (2012). Evaluations in the science of the artificial–reconsidering the build-evaluate pattern in design science research. *In Design Science Research in Information Systems. Advances in Theory and Practice: 7th International Conference, DESRIST 2012, Las Vegas, NV, USA*, May 14-15, 2012. Proceedings 7 (pp. 381-397). Springer Berlin Heidelberg.

Tufte, E. R., (1983/2007). *The visual display of quantitative information. Cheshire*, CT: Graphics press.

Zhang, H., Chen, G., Ooi, B. C., Tan, K. L., & Zhang, M. (2015). In-memory big data management and processing: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 27(7), 1920-1948.