UNIVERSITY OF TWENTE.

BRICK LOG Logisch in Logistiek

A standardised layout data model and visualisation tool to showcase the performance of class-based storage policies in warehouses

Author
E.H. Huisman
s2506415

Supervisors
Dr. B. Alves Beirigo (University of Twente)
Dr. A. Trivella (University of Twente)
Hubert Benneker (Bricklog B.V.)

A thesis presented for the bachelor degree of
Industrial Engineering and Management

in the
Faculty of Behavioural, Management and Social Sciences

June 28, 2023

PREFACE

Before you lies my bachelor thesis with the title “a standardised layout data model and visualisation tool to showcase the performance of class-based storage policies in warehouses”. I would like to thank Breno Alves Beirigo for his support throughout my thesis journey. He is my first supervisor whose keen eye for detail greatly helped me improve this thesis. I would also like to thank my second supervisor, Alessio Trivella, for providing valuable feedback.

The third supervisor who deserves my gratitude is Hubert Benneker. Hubert is one of the founders of Bricklog and his experience helped me understand warehouse operations more and more. Hubert is also one of the reasons why the office days in Apeldoorn were never boring.

Looking back at the graduation thesis, I am grateful to have been part of the small Bricklog Team. From the beginning I have been warmly welcomed. Although the busy schedules of all employees, they made time for me.

This graduation thesis marks for me the end of my bachelor’s degree. For Bricklog, I hope this thesis serves as the beginning of the development of a top-selling product. Enjoy the read.

Best,
Elise Huisman

June 28, 2023

MANAGEMENT SUMMARY

Introduction of the company and the problem

This research project focuses on the development of a warehouse layout data model and visualisation and reporting tools for Bricklog, a company offering business intelligence (BI) products to logistics companies. The dashboards that Bricklog sells are different compared to offering BI services. The dashboard *product* implies the requirement of developing a *standardised* method to produce the product for the customer. Their target customers are small, manual warehouses that operate with basic warehouse management systems which do not enable extensive analyses.

The new product of Bricklog should provide Bricklog's customers with insight into the implementation and efficiency of a class-based storage assignment policy in a manual warehouse. The class-based storage assignment policy assigns products to a certain zone in a warehouse to reduce order picker travel. However, Bricklog needs to solve two main problems before the product can be released. First, there is a partial absence of warehouse location data. Second, there is an absence of standardised visualisation and reporting tools. This thesis has a practical contribution by developing a method to fill a standardised layout data model with the absent data on storage locations. Second, visualisation and reporting tools are developed for Bricklog enabling warehouse managers to get more understanding on the implementation and efficiency of a class-based storage policy.

The development of the standardised data layout model

The data on the relative positions of warehouse storage locations is missing a standardised data format; there is no standardised format in which the data is present. In this thesis the data format is developed. The data format is transformed to output a standardised data layout model. The data model is a table that is added to the existing data model of Bricklog. The table consists of each storage location's x-coordinate, y-coordinate, z-coordinate, distance to the depot, and the class the location belongs to. Based on this data, the development of the visualisation tools is possible for a class-based storage policy analysis.

The development of the visualisation and report tools

The visualisation and reporting tools are designed to provide insights into product class allocation (the "product" dashboard), location class allocation (the "location" dashboard), and the travel distance of order pickers for the current storage policy and the class-based storage policy (the "analysis" dashboard). These tools offer features such as date filtering, filtering based on product or location characteristics, and 3D visualisation capabilities.

The products dashboard provides an overview of all products, calculates class assignments based on order frequencies, and determines the percentage of locations reserved for each class. The locations dashboard displays distance information, average distances of classes, and a 3D model illustrating class assignments. Both dashboards serve the function of enhancing warehouse managers with the means to implement a class-based storage policy. The warehouse manager can derive from the dashboard which products should be stored in a certain zone of locations.

The analysis dashboard utilises outbound mutation data to calculate the travel distance of order pickers for the class-based storage policy. The historic data is used to calculate the historic order pick travel distances. The percentage difference for the class-based storage policy and the historic storage policy is given on the analysis dashboard. Besides the percentage difference, the warehouse manager can see the absolute difference in order picker travel, the absolute difference in order

picker time, and the percentage of time used by order pickers while maintaining the same productivity.

Demonstration of the tools on data of a demonstration warehouse

The designed adapter and visualisation/reporting tools are tested and showcased using a fictitious warehouse called the “demo-warehouse.” Dummy data, generated to simulate a warehouse management system, is used for the demonstration. The demo-warehouse is a unit-load warehouse consisting of 1600 pallet racking locations. The data of the demo-warehouse represents a random storage policy.

The demonstration compares the results between the random storage policy and the class-based storage policy for different variables. As expected, the demonstration confirms that class-based storage policies effectively reduce travel distance in the warehouse, with a reduction of approximately 32%. In other tests, input variables are changed to analyse the resulting travel distance reduction. The differences in the results are all rationally explainable. This confirms the correct development of the tools.

Recommendations

The following recommendations are proposed for Bricklog:

- Integrate the product with a warehouse management system to identify potential flaws and issues with implementation in real-world scenarios.
- Conduct a market analysis to assess the demand for a BI product focused on class-based storage analysis. Understanding customer requirements and market needs will aid in refining and optimising the product.
- Gather specific requirements from potential customers to guide the product’s development trajectory.

CONTENTS

List of terms, acronyms and abbreviations

SME small and medium-sized enterprise

MPSM managerial problem-solving method

I/O point inbound/outbound point, or input/output point

CPO cube-per-order index

AS/RS automated storage/retrieval systems

RSS required storage space

WMS warehouse management system

KPI key performance indicator

MHE material handling equipment

SKU stock keeping unit

DSRM design science research methodology

SLAP storage location assignment problem

BI business intelligence

COL closest open location

1 | INTRODUCTION

This thesis is written at Bricklog, a business intelligence company. Section 1.1 provides information about the company. A theoretical framework on warehouse operations is presented in Section 1.2, while Section 1.3 outlines the problem-solving approach employed.

1.1 Background of the company

The company involved in my graduation assignment is Bricklog Holding B.V., commonly known as "Bricklog". Bricklog is a 20-employee company based in Apeldoorn (main office) and Enschede (a satellite office with a capacity of four employees). The private limited company was founded in May 2015 to sell a data-driven product in the transport and logistics sector. The two founders combine 25 years of experience in the transportation and logistics branch, along with technical knowledge and change management [1].

Bricklog's target market primarily consists of small and medium-sized enterprises (SMEs) operating in the logistics sector. A significant portion of these enterprises currently lacks an effective approach to data processing. Bricklog offers a business intelligence (BI) product designed to address this challenge. Business intelligence refers to a range of tools and processes that enable companies to manage their data from collection to reporting. Bricklog aims to make the customer's operation a data-driven operation by implementing their business intelligence tools.

In recent months, Bricklog has started developing a BI product specifically for warehouse operations. Warehouse management systems (WMS) store vast amounts of data related to warehouse operations. However, many smaller warehouses fail to analyse this data effectively to enhance their operations. The bachelor assignment is rooted in Bricklog's development trajectory, which aims to create a BI product tailored for warehouse operations.

1.1.1 Bricklog Data Factory

The Bricklog product outline comprises three main components: the source (co1), the "Bricklog Data Factory" (co2), and the end product (co3). The primary source of data is the customer's WMS. The Bricklog Data Factory consists of various elements that transform the data from the source into the desired end product. The end product is a comprehensive framework for the customer, encompassing reports and dashboards.

The Bricklog Data Factory is operated by data engineers and data analysts. At the core of the factory lies a standardised data model. Data engineers develop an adapter, which can be understood as a concept from computer science. In this context, an adapter is responsible for converting the interface of one class into another interface that clients expect. The purpose of the adapter is to enable the collaboration of classes that would otherwise be incompatible due to differing interfaces [2, p. 139]. In the case of Bricklog, the adapter consists of pipelines that transform the data from the source, ensuring that the data model is populated accurately. The pipelines are standardised to enable re-usability.

Data analysts utilise the standardised data model to create graphics, tables, and figures. These visualisations, along with generated dashboards and reports, are automatically generated. The data visualisation software used by Bricklog is "Power BI" of Microsoft. The data model and corresponding reporting and visualisation tools adhere to standardised formats. As a result, Bricklog is able to offer a product that provides standardised insights for warehouse managers, rather than custom-made solutions.

In summary, the source (co1) and the end product (co3) are unique to each customer. On the other hand, the Bricklog Data Factory (co2) operates in a standardised manner. Data engineers transform the source data to populate standardised data models for data analysts. In this context, the source acts as the adaptee, the data pipelines serve as the adapter, and the data models act as the client. Data analysts utilise these standardised data models to develop visualisations and reporting tools. The outline of the different components is visually represented in Figure 1.1.

With this background information, the term “adapter” in the latter part of this thesis refers to the configuration of source data into standardised data models or datasets. “Standardised visualisation and reporting tools” encompass various calculations, transformations, and analyses based on the generic data model.

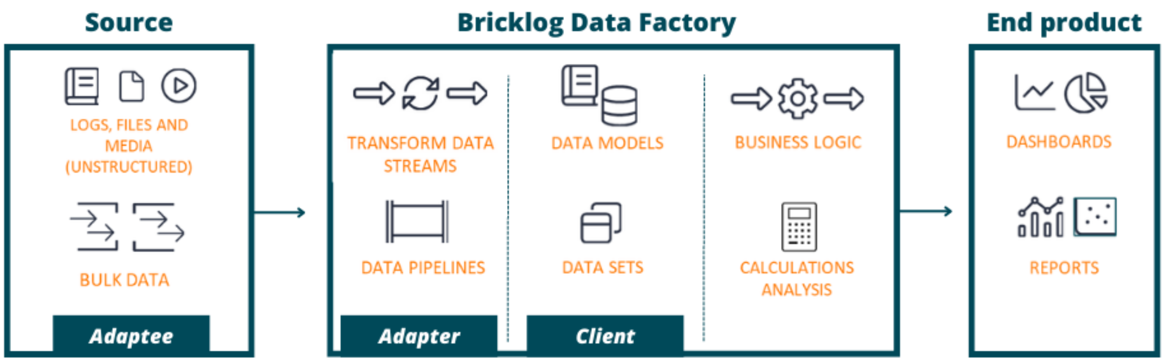


Figure 1.1: The key components of the product of Bricklog (co1, co2, and co3)

1.2 Theoretical framework on warehouses

This section aims to establish a foundational understanding of warehouse concepts and thereby providing the necessary background information. Section 1.2.1 introduces and defines two distinct types of warehouses. Additionally, Rouwenhorst et al. [3] further characterises warehouses by discussing three key perspectives: processes, resources, and organisation. Section 1.2.2 delves into the various processes involved in handling products within a warehouse. Furthermore, Section 1.2.3 elaborates on the means and equipment required for efficient warehouse operations. Subsequently, Section 1.2.4 discusses the essential procedures employed in managing warehouse operations. Finally, Section 1.2.5 presents vital information on warehouse activity profiling, which serves as a crucial step in virtually every warehouse project.

1.2.1 Type of warehouses

In logistics networks, warehouses fulfill a critical function as storage locations. They serve as repositories for inventory until the goods are required. Two primary types of warehouses are distinguished: *distribution centers* and *production warehouses*. Distribution centers serve as hubs where products from one or multiple suppliers are collected for subsequent delivery to various customers. On the other hand, production warehouses accommodate a production environment where raw materials, semi-finished products, and finished products are stored and distributed [4].

1.2.2 Warehouse processes

Upon arrival at a warehouse, products undergo a sequence of defined processes. The literature commonly identifies four key processes in warehouse operations: *receiving, storage, order picking,* and *shipping* [3, 5, 6].

In addition to the four main processes mentioned earlier, three additional processes can be added: *cross-docking, replenishment,* and *sorting and/or consolidation.* Cross-docking involves the direct transfer of products from the receiving area to the shipping area [7, p.73]. Consolidation refers to the practise of combining multiple individual items from an order into a single shipment. *Replenishment* entails moving products from the reserve area to the forward area [3]. The reserve area is designed for cost-effective bulk storage, while the forward area houses products for the order picking process. *Sorting and/or consolidation* is an optional process that may occur after the initial picking process. Figure 1.2 provides an overview of the theoretical framework for warehouse processes and organisation, which will be discussed in detail in the following paragraphs.

Three additional processes are potentially added: cross-docking, replenishment, and the sorting and/or consolidation process. Cross-docking is a process in which products are moved directly from the unloading docks to the loading docks [7, p.73]. A warehouse dock is a designated area where the trucks are loaded and unloaded. Consolidation means sending several individual items from an order in a single shipment. *Replenishment* is the process that moves products from the reserve area to the forward area [3]. The *reserve area* stores products in the most economical way; the bulk storage. The *forward area* stores product for the order-picking process. *The sorting and/or consolidation process* is optionally performed after the other picking process. Figure 1.2 provides an overview of the various processes involved in a warehouse. The organisational policies depicted in the figure will be discussed in Section 1.2.4.

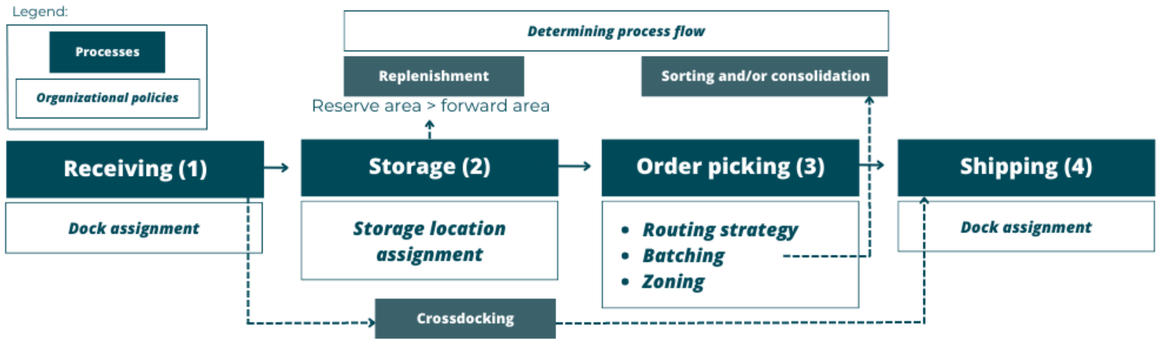


Figure 1.2: Overview of main warehouse processes including possible organisational decisions (based on Gu et al. [6]).

1.2.3 Warehouse resources

Resources encompass all the assets and infrastructure within the warehouse, constituting the essential elements for efficient operations. The key resources encompass the storage units, storage systems, material handling equipment (MHE), computer system, and personnel.

Storage units

A product is defined as a type of good, such as a battery. Each individual battery is referred to as an *item* (or unit), while a box or container containing multiple batteries is known as a *case*. A *pallet* consists of multiple cases stored together on a (wooden or plastic) frame. In practice, the terms “euro pallets or EUR-pallets” and “block pallets” are commonly used. By default, an EUR-pallet has a floor dimension of 80 x 120 cm, whereas block pallets have a size of 100 x 120 cm. In a traditional unit-load warehouse, products are handled on pallets and considered as a whole as one unit [8].

Storage system numbering methods are used to identify and locate products within a warehouse. The most commonly used numbering system for pallet racks is explained in Figure 1.3. In case of multiple halls, the numbering system adds a number in front to specify the hall.

The combination of several items from different products requested by a customer is referred to as a customer order [3, p. 516]. The order consists of one or multiple order lines, with each order line containing information about a specific item, such as the Stock Keeping Unit (SKU) and the ordered quantity. The Stock Keeping Unit (SKU) is a unique code assigned to a specific product.

If a customer requests a quantity that is less than a full case, it is termed a *broken-case pick*. On the other hand, if a customer requests a quantity that is an integer multiple of a case quantity but less than a pallet (unit) load, it is referred to as a *full-case pick* or a *case pick*. A *pallet pick* represents an order quantity that is a multiple of a pallet load quantity [9, p. 240].

Storage Systems

When it comes to storing pallets, two commonly used terms are block stacking and pallet racks. The simplest form of storage is *block stacking*, where pallets are stored on the floor and stacked on top of each other in an open space within the warehouse. *Floor storage* is sometimes used as a synonym for block stacking or as a specific term when the pallets are not typically stacked. Another storage method is *pallet racks*, which are metal constructions that enable stacking of pallets while still allowing manual access to pallets on lower levels directly [4, p. 7]. Pallet racks can be either single-deep or multi-deep.

In a single-deep configuration, pallets are stored in a single row, providing direct access to each pallet from the aisle. On the other hand, multi-deep storage involves multiple rows of pallets positioned behind one another. Additionally, warehouses can be classified as low-level or high-level based on the vertical extent of the storage space. A low-level warehouses enables operators to order pick on foot. In contrast, a high-level warehouse features taller storage systems.

Storage system numbering methods are employed to identify and locate products within a warehouse. An often used numbering system for pallet racks is to identify rows, bays, levels and places. The way of numbering is portrayed in Figure 1.3. In the case of multiple halls, the numbering system incorporates a hall number as a prefix to specify the specific hall.

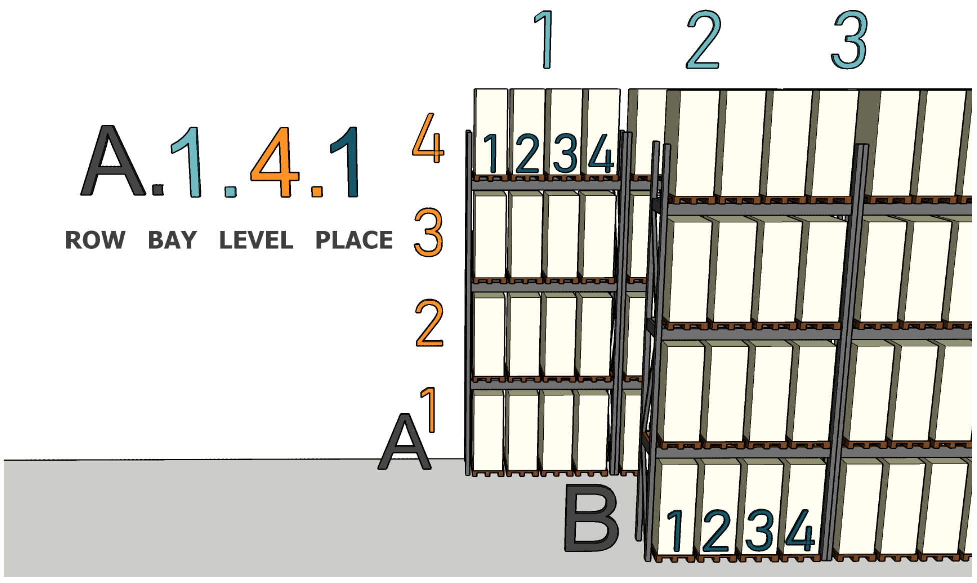


Figure 1.3: A warehouse numbering system for pallet racks.

Material handling equipment

The storage units are moved with use of material handling equipment (MHE). There are many different types of equipment, and the selection depends, among other reasons, on the type of storage systems, the available space, and the type of storage units. Table 1.1 provides an overview of commonly used handling equipment.

Table 1.1: Overview of a number of material handling equipment for the transport of pallets in a manual warehouse, based on overview of Richards et al. [7].

MHE for horizontal movement	MHE for horizontal & vertical movement
Hand pallet trucks (HPTs)	Pallet stacker
Conveyors	Counterbalance forklift trucks (CBT)
Low-level order picker lift truck	Reach trucks
	Narrow aisle trucks
	Medium and high level order picker lift truck

Computer system
Companies require information technology tools to enhance their business operations and establish reliability, speed, control, and flexibility in their warehouse processes [7, p.188]. While some warehouses still rely on paper-based warehouse management systems to manage and control their operations, introducing software technology can significantly improve productivity. The market offers a wide range of Warehouse Management Systems (WMSs), with annual fees varying from a few thousand euros to several million euros.

Personnel
Personnel in a warehouse can include various roles such as warehouse manager, order picker, and forklift operator. The number and ratio of employees required can depend on factors such as the size of the warehouse, the type of operation that is being carried out and the level of automation in place.

1.2.4 Warehouse organisation

Organisation includes all planning and control procedures used to run the warehouse. All the procedures come with certain design decisions. The design decisions are often considered at strategic, tactical, and operational levels. Strategic decisions have a long-term impact, tactical decisions have a medium-term impact, and operational decisions have a short-term impact [3].

Dock assignment and storage policies
Each process has its own organisational policies. For the receiving and shipping process, a dock assignment policy determines the allocation of trucks to docks. In the storage process, items are transported to storage systems and assigned to storage locations. A dedicated storage policy prescribes a particular location, whereas a random storage policy leaves the decision to the operator. In between, a class-based storage policy (ABC zoning) allocates zones to specific product groups, often based on their turnover rate [3, p.517].

Routing policies
Three distinguished operating policies for order picking are routing strategies, zoning, and batching. *Routing strategies* determine the sequence of retrievals and the route to visit the retrieval locations [3, p.517]. Routing policies can be categorised into two groups. Firstly, there are *optimal algorithms* that determine the shortest routes. Secondly, there are routing *heuristics* that determine a feasible route, which may not necessarily be the shortest route [4, p.32].

For a pick-list with one or two items, routing policies are not of importance, because the travel distance will always be the same. For pick-lists of three or more items, the s-shape (or transversal) heuristic, largest gap heuristic and aisle-by-aisle heuristic are common in literature.

Routing policies are focused on just the order picking process. However, the storage and the order picking could also be combined. A pick-list of one item could be picked in a single command cycle or a dual command cycle. Single command cycles means to move a load from the input point into a rack, or a load is moved from a rack to the output point. Dual command cycles means that load is picked up at the input point and put in the rack, then another load is retrieved from the rack and deposited at the output point [4, p.19]. Route length will be the same regardless of the sequence of the visits.

Zoning and batching
Zoning occurs if each order picker only picks products from an order if those products are located in his assigned zone. *Batching* means bundling orders and retrieving them simultaneously by one picker. This means that a sorting and/or consolidation process is required afterward. The alternative is single-order picking, in which orders are picked one by one without batching.

Design of the warehouse flow
In addition to the specific policies for each process, the design of the process flow is considered the most crucial decision, as highlighted by Rouwenhorst et al. [3]. This decision encompasses factors such as incorporating separate reserve and forward storage areas or implementing batching, which would necessitate adjustments to the sorting process.

1.2.5 Warehouse activity profiling

For almost any significant warehouse project, warehouse activity profiling is the first necessary step. It is the careful measurement and statistical analysis of warehouse activity [9, p.233]. There are three main types of data required to support profiling: data pertaining to each SKU, data pertaining to customer orders and data pertaining to locations within the warehouse [9, p.237]. The SKU and customer order data is in a standard WMS included in the data. Warehouse location data is generally most challenging to obtain in a standardised manner. The data related to warehouse locations typically includes maps, blueprints, or sketches of the warehouse.

1.3 Problem-solving approach

Bricklog is dedicated to developing a business intelligence product specifically tailored for warehouses. The primary focus of this product is to address practical challenges and enhance operational efficiency within warehouse operations. In this context, the Design Science Research (DSR) approach is deemed more suitable compared to the commonly used Managerial Problem-Solving Method (MPSM) advocated by Heerkens et al [10]. Unlike the MPSM, which typically focuses on solving immediate managerial issues, DSR offers a research framework that emphasises the creation and evaluation of innovative solutions to practical problems.

To guide the research process, the chosen DSR process model is the widely recognised model presented by Peffers et al. [11]. This model cycles through six phases as listed in Table 1.2.

Table 1.2: The six phases of the Design Science Research Methodology as constructed by Peffers et al. [11]

Phase	Documentation
Identify problem and motivation	Chapter 2
Define objectives of a solution	Chapter 3
Design and development	Chapter 4, 5, 6
Demonstration	Chapter 7
Evaluation	Chapter 8
Communication	Chapter 8

2 | PROBLEM IDENTIFICATION

In this chapter, the management problem (Section 2.1), the theoretical background on efficiency improvement in warehouses (Section 2.2) and the theoretical framework on storage policies (Section 2.3) come together to shape the formulation of the problem statement (Section 2.4). Following the problem statement, the motivation for conducting the research is presented in Section 2.5.

2.1 Management problem

The problem statement of Bricklog management wrapped in one sentence is:

"Develop a business intelligence (BI) product offering actionable insights to warehouse managers to drive efficiency improvement in their warehouse operations."

The warehouse managers are the customers of Bricklog. The BI product should assist warehouse managers in analysing their data and acquiring actionable insights to make informed business decisions. Ideally, the BI product facilitates strategic and operational decision-making within a warehouse. The ultimate decisions always lie with the warehouse manager themselves.

In the problem statement the term"efficiency improvement" is very broad. There are endless means to improve efficiency for warehouse managers. Clarity in defining "efficiency improvement" is necessary to establish a specific objective for improvement. Therefore, a rational and structural approach on specifying improving efficiency for the BI product is necessary to be able to formulate a problem statement.

2.2 Outline of efficiency improvement in a warehouse

Improving efficiency in a manual warehouse involves the reduction of time spent on various processes within the facility. Section 1.2.2 provides an overview of the four main processes conducted in a warehouse. To minimize time consumption, warehouse managers can implement numerous interventions. The forthcoming BI product should ideally provide valuable insights in interventions that offer significant efficiency improvements relative to their cost. Consequently, it is crucial to acquire information regarding the duration of each process in the warehouse and the potential efficiency enhancements associated with different interventions.

2.2.1 Most time-consuming process in a warehouse

In the case of a manual process, order picking operations are estimated to account for approximately 55% of the total cost of warehouse operations, as stated by Tompkins et al. [12]. Coyle et al. [13] also mention that this percentage ranges between 50% and 75%. Given this significant contribution, improving order picking becomes the primary focus in enhancing warehouse productivity [14, p.1].

2.2.2 Interventions to improve the order picking process

Bottani et al. investigated the factors that influence the efficiency of the order picking process in literature, which are illustrated in Figure 2.1 [5]. These factors include layout design, material handling equipment, routing strategy, picking policy, and storage assignment policy.

For Bricklog's BI product, placing a strong emphasis on strategic-level decisions such as layout design or material handling equipment may be less suitable. These decisions are typically one-time choices and may not directly align with the BI product's objective of utilising daily operational data to support warehouse managers' decision-making. The other three factors can be categorised as organisational policies, as discussed in Section 1.2.4. By aligning the knowledge on organisational policies with the findings of Bottani et al., it can be concluded that four policies have an impact on the order picking process: the storage assignment policy, routing strategy, batching, and zoning.

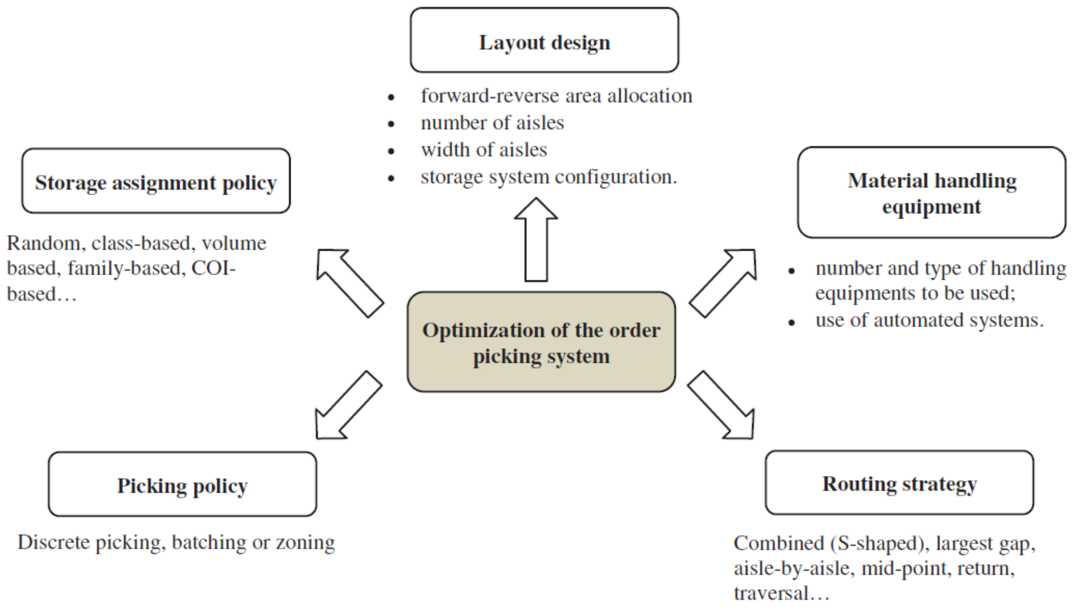

Layout design
forward-reverse area allocation
number of aisles
width of aisles
storage system configuration.
Storage assignment policy
Random, class-based, volume based, family-based, COI-based…
Material handling equipment
number and type of handling equipments to be used;
use of automated systems.
Optimization of the order picking system
Picking policy
Discrete picking, batching or zoning
Routing strategy
Combined (S-shaped), largest gap, aisle-by-aisle, mid-point, return, traversal…

Figure 2.1: Factors affecting the optimisation of an order picking system [5]

Petersen and Aase conducted a study to investigate the impact of process decisions on the travel distance of order pickers [15]. The results of their research revealed that the most significant savings in travel distance can be achieved through the implementation of batching. Class-based or volume-based storage policies provides nearly the same level of savings as batching. Zoning and routing policies yield lower efficiency improvement results.

While batching has been shown to offer substantial efficiency improvements, it is not be the most suitable intervention for Bricklog's BI product. Batching involves the implementation of sorting and/or consolidation processes, which can inadvertently limit the market for Bricklog by excluding warehouse managers who prefer not to utilize such processes. Instead, focusing on providing insights on storage policies aligns with the preferences of a larger pool of potential customers.

The next step to determine the detailed problem statement involves selecting a specific type of storage policy. Considerable contributions have been made in research regarding storage policies, commonly known as the storage location assignment problem (SLAP). Numerous storage assignment methods have been developed and studied. In the following paragraphs, a classification and overview of storage assignments will be provided, followed by an explanation of the selected storage policy for Bricklog's BI product.

2.3 Theoretical framework on storage policies

Storage policies improve the order picking process by reducing travel time for the order pickers. In manual warehouses, it is commonly assumed that travel time increases with travel distance [4, 15, 14]. As a result, optimising travel distance is often considered as an objective in warehouse operations. Storage policies play a crucial role in achieving this objective by assigning product items to specific locations [3, p.517]. In the theoretical framework the random, dedicated and class-based storage policies will be outlined.

2.3.1 Class-based storage

Product items can be bundled in groups, and then assigned to location zones. Such groups are called classes. For example, locations close to the inbound/outbound (I/O) point are for the product group of fast moving products (class A), and the other locations are for slow-moving products (class B). Each item in the product is classified in class A or B based on the mean expected storage time. Within a storage class, items are randomly stored. This is an example of a class-based storage assignment with two classes based on the popularity of a product.

Traditional class-based storage assignment is based on three classes with turnover as the classification criterion. These classes are commonly referred to as A, B, and C, hence the term ABC-storage method. The ABC-storage assignment utilises Pareto's principle to partition product items into classes. Pareto's principle suggests that 20% of the population holds 80% of the wealth in Italy, and similarly, approximately 20% of the products in a warehouse typically contribute to 80% of the turnover (in terms of monetary value) [16]. Class A comprises the top 20% of products with the highest turnover, following Pareto's principle. However, specific guidelines for partitioning into classes have not been established in the literature [14]. The extreme cases of class-based storage are the random storage assignment method and the dedicated storage assignment method.

2.3.2 Random storage

Random storage assignment involves having a single class for all products, allowing flexibility in storing products anywhere within the warehouse. To determine the exact storage locations during the inbound process, an open location selection rule can be implemented, such as the closest open location (COL) rule [17]. This rule helps in selecting the nearest available location for storage.

Random storage provides a straightforward and adaptable approach as products can be placed in any open location within the warehouse. However, it may result in sub-optimal travel distances for order pickers as there is no deliberate organisation.

2.3.3 Dedicated storage

Dedicated storage assignment assigns a separate class for each product. Each product is assessed based on a performance measurement, and a specific location is assigned accordingly. This approach requires more extensive data processing as the precise locations for each product need to be recorded and known during the inbound process.

2.3.4 Comparison of the three storage assignment methods

Randomised storage outperforms the other two storage methods based on the utilisation of storage space. Dedicated storage significantly reduces the travel distance compared to randomised storage [17]. The downside of dedicated storage is that the process requires more and accurate data processing.

In the middle, class-based storage is considered as an alternative that has the benefits of both random and dedicated storage assignment methods [6, 18]. Class-based storage can realise with a few classes travel times that are similar to dedicated storage assignment, while maintaining ease of use [19, 20]. Therefore, class-based storage is considered to be the most popular assignment method that comprises material handling costs, ease of use, and utilisation of storage space [21].

2.3.5 Storage assignment for Bricklog

Bricklog will make visualisations in their product that can offer insight into class-based storage assignment. Class-based storage assignment policy is in literature the storage assignment with overall the best performance [14, 17, 22, 23, 21]. The class-based storage assignment has the benefits of both random and dedicated storage assignment. Class-based storage achieves near optimal efficiency performance, while being easier to implement and work with than dedicated storage.

2.4 Problem statement

Utilising the background information pertaining to the management problem, the clarification of efficiency improvement, and the theoretical framework on storage policies, it becomes possible to define more specifically the requirements of the BI product, referred to as the norm (Section 2.4.1). Subsequently, an evaluation of the current state of affairs, commonly referred to as the status quo, will be presented (Section 2.4.2). Through an analysis of the gap between the desired norm and the current reality (Section 2.4.3) a research question (Section 2.4.4) and its corresponding sub-questions (Section 2.4.5) can be formulated.

2.4.1 The norm

The desired outcome for Bricklog is a BI product offering actionable insights on class-based storage policies with the goal to reduce travel distance in a manual warehouse. The end-user of the BI product are warehouse managers. The warehouse managers should get a basic understanding of class-based storage policies. The BI product should help to implement a class-based storage policies. The warehouse manager should gain insights in the possible potential for his warehouse of class-based storage policy by getting analysis on efficiency improvement.

2.4.2 The status quo

Bricklog has developed a standardised data model which can store all the data of different WMSs. Currently in the data model a "location" table is filled with the following data:

1. Location ID: this is the primary key of the table; a unique number serving as identifier of each storage location.
2. Location specification: the hall, row, bay, level and place and each location.
3. Dimensions of the location: the height, length, width and the maximum weight possible to store at the location.
4. Type of location: examples include pallet rack or block stacking.

There is location data missing to offer analysis on the travel distance of order pickers. Data on the locations of storage units is typically not stored in a standard WMS. Commonly, the data source for this type of data is not existing, or a type of map/floor plan of the warehouse is available. If a map is available, the type of map and its level of detail differs for each customer. This means that a standardised data source for warehouse storage locations is missing as well.

Besides the missing data source, there are no visualisation and reporting tools to develop dashboards for the customer. Bricklog has developed dashboards for the transportation sector. The design can be made in the same style and logic.

2.4.3 The gap between the norm and the status quo

The status quo is lacking on two mains fronts compared to the norm. First, there is a partial absence of warehouse location data. Second, the absence of standardised visualisation and reporting tools restricts the ability to extract actionable insights from the existing data to drive efficiency improvement.

To fill the first gap, data on the relative positions of all storage locations is required. To ensure alignment with the current outline of Bricklog's product (mentioned in Section 1.1.1), it is necessary to address the absence of warehouse location data by developing an adapter. This adapter is responsible for configuring the warehouse layout data into the existing standardised data model. The adapter starts with a standardised data soure format. For instance, SAP Extended WMS software requires X and Y coordinates for the start and end of the network to calculate travel distances [24]. The adapter transforms the data in the data source format to the desired outcome. The adapter's outcome is a populated *data model* that includes information about the *layout* of a warehouse. The adapter follows a *standardised* approach. With this knowledge the adapter will from this point on be known as the development of a "standardised layout data model".

To fill the second gap, there should be standardised visualisation and reporting tools on top of the data model to bridge the gap between the norm and the status quo. The tools should provide the end-user with insights into the reduction of travel distance through a class-based storage policy based on the historical data of the warehouse. The insights should enhance the knowledge of the warehouse manager about an implementation of a class-based storage policy.

2.4.4 Research question

The research question derived from the gap between the norm and the status quo is:

How can visualisation and report tools be developed, utilising a standardised layout data model, to provide Bricklog's customers with insights into the implementation and operational efficiency of a class-based storage assignment policy within their warehouse?

2.4.5 Sub-questions

In addition to the main research question, different sub-questions are formulated.

1. What are the design objectives for the standardised layout data model and the visualisation and report tools?
2. What are necessary decisions for the implementation of a class-based storage policy?
    - What decisions are made for the visualisation and report tools for the BI product of Bricklog?
3. How to develop the standardised data layout model to complete the current data model?
    - Which input variables are necessary for the standardised data source on warehouse layout data?
4. How to develop the visualisation and report tools for the class-based storage policy in a warehouse?
    - What type of visualisations are suited for the dashboards?
    - Which information is stored in reports to enhance the understanding of the warehouse manager about the class-based storage policy?
    - What calculations are made in Power BI to enable the visualisations?
5. What are the results of the standard warehouse layout model and the visualisation and report tools in a demonstration warehouse operating with a random storage policy?
    - Based on the generated sample data, what are the efficiency results of a class-based storage policy compared to the random storage policy?

2.5 Motivation

This thesis contributes to practice through the development of a BI product on the efficiency of class-based storage policy. The main contributions can be summarised in three points.

Firstly, the thesis addresses the issue of affordability and accessibility of storage placement analyses in warehouse management systems (WMS). Traditional WMS systems offering such analyses are often expensive, making them inaccessible for small or medium-sized enterprises with limited budgets. However, the Bricklog product developed in this thesis offers a more affordable alternative to the current market. By providing an add-on option to existing basic WMS systems, the thesis enables a greater segment of people to benefit from storage placement analyses, thereby expanding the customer base for such services.

Secondly, the thesis enhances the knowledge and understanding of warehouse managers by delving into the intricacies of class-based storage policies. While existing products and WMS analyses may provide a class-based storage policy, they often do not focus on providing a deeper understanding to warehouse managers regarding the policy itself. In contrast, the thesis not only presents a class-based storage policy but also offers additional insights and knowledge about its implementation. This empowers warehouse managers with a more comprehensive understanding of the policy, enabling them to make informed decisions and optimise their warehouse operations effectively.

Lastly, the thesis introduces a practical and systematic approach for gathering data on warehouse layouts. Accurate data on the physical layout of a warehouse is crucial for conducting storage placement analyses and implementing effective storage policies. However, obtaining such data can be challenging, time-consuming, and prone to errors. The thesis addresses this issue by presenting a methodical approach to collect data on warehouse layouts. By providing clear guidelines and procedures, the thesis facilitates the process of data gathering, making it more efficient and reliable.

In summary, the bachelor thesis contributes to practice in three ways:

1. The resulting products offers an affordable and accessible storage allocation analysis for warehouse managers.
2. Product enhancing the knowledge of managers on the implementation of class-based storage policies.
3. Thesis presents a practical method for gathering data on warehouse layouts.

Through these contributions, the thesis provides valuable insights and tools that can be applied in real-world warehouse management scenarios, benefiting both warehouse managers and Bricklog.

3 | OBJECTIVES

The chapter establishes the objectives for the design and development phase of the BI product. In Section 3.1, Bricklog's strategy of offering visualisation products instead of services is discussed, and the corresponding objectives are outlined. Additionally, Section 3.2 focuses on the objectives for the data model, while Section 3.3 elaborates on the objectives for the visualisation tools. A summary of all the objectives can be found in the conclusion presented in Section 3.4.

3.1 Strategy of Bricklog

Bricklog aims to offer *productised* services. One characteristic of a product is modularity; the reusability of components. The aim is to minimise the number of individual customised steps. The operations of Bricklog become scalable when a product is offered instead of a service. Bricklog competitors offer mostly services; for each customer, the BI reports are built from scratch. Bricklog wants to set itself apart on the market by offering a product, instead of tailor-made reports. One of the accessory objectives of offering a product, is that in the development the need of most customers in their target market should be taken into account. Instead of responding to a specific customer need. This objective keeps the pool of potential customers as large as possible

Secondly, *change management* plays an important role in the relationship between Bricklog and its customers. Bricklog has experienced in the past that customers do not see the added value of the product and ask many specific questions when they do not understand the product. This shifts Bricklog's operations towards becoming more service-focused in stead of selling a product. Therefore it is key to make the customer understand the product. For example, the customers should understand that gathering correct data is the first crucial step for optimising the operations through data analysis. Making the product understandable for customers, requires Bricklog to start simple with data analysis. Furthermore the different dashboards and reports should all have the same look and feel to increase the learning curve of customers in the use of the product.

3.2 Objectives for the warehouse layout data model

The standardised warehouse layout data model (as discussed in Section 2.4.3) is a data model populated with data about relative positions of storage location. The data model is populated in a generic way based on the standardised data source format. The standardised data source format should be user-friendly, even in cases where no map or only a basic warehouse layout is provided. The data source format should be as small as possible while being sufficient to populate the data model. The final objective of the standardised warehouse layout data model is to establish a method for configuring the input into extra columns for the existing location table.

3.3 Objectives for the visualisation and report tools

After the standardised data sets are filled with data, visualisations and reports need to be developed. Bricklog requires the use of Power BI as visualisation application. The main requirement of the visualisations is the enhancement of the understanding of warehouse managers on the implementation of class-based storage policies for their own warehouse. The report tools should enable an overview of all products and the assigned class in the class-based system. The warehouse manager should understand the performance indicator(s) that determine the class assignment to SKUs.

Once the standardised data sets are populated, the focus shifts towards creating visualisations and reports utilising Power BI as the selected visualisation application by Bricklog. The key objective of these visualisations is to enhance the understanding of warehouse managers regarding the implementation of class-based storage policies tailored to their respective warehouses. Essentially this means that not just the result of the class-based storage policy should be visualised, but also the logic towards the result. The report tools should provide an overview of the difference of order picking travel between a class-based storage policy and the current policy.

One of the requirements of Bricklog is that the visualisation tools should include a 3D sketch of the warehouse. The 3D sketch serves as a strong visual compared to tables and reports. The selling point for Bricklog is with a 3D visualisation a lot better, because with a 3D visual the dashboards look more professional.

3.4 Chapter 3 conclusion

In summary, all the objectives are listed below. The objectives provide the answer to the sub-question: *What are the design objectives for the standardised layout data model and the visualisation and report tools?*

The general Bricklog product requirements are:

- Reduction of individualization for customers through modularity.
- Consider the needs of the majority of customers, ensuring a broader appeal and maximising the potential customer base.
- Implementation of a user-friendly approach to make insights easily understandable for end-users.
- Maintenance of the same look and feel as the existing Bricklog products.

The objectives for the standardised warehouse layout data model are:

- Development of a data source format that can be easily populated with data regarding the relative positions of locations.
- Creation of a query that configures the data source to the standardised data model suitable for visualization purposes.

The objectives for the visualisation and reports tools are:

- Utilisation of Power BI as the visualisation application.
- Provision of insights about class-based storage policies for each SKU and each location which enhance the knowledge of the warehouse manager.
- Creation of visualisations to highlight the differences between the current and class-based storage assignments.

4 | DESIGN DECISIONS ON CLASS-BASED STORAGE ASSIGNMENT

Endless variations on the class-based storage policies exists. To implement a class-based storage policy, five decisions should be made. An overview of the decisions is given in Section 4.1. In the subsequent sections the decisions are explained. Section 4.2 addresses the determination of the number of classes. Section 4.3 explores the selection of a suitable performance indicator for class allocation. Section 4.4 establishes the distribution ratio of products across each class. Section 4.5 discusses the storage boundaries for the products in the warehouse. Section 4.6 mentions the review date of the class-based storage assignment policy. Section 4.7 includes a conclusion of the chapter.

4.1 Overview of choices and decisions

The choices in Table 4.1 give an understanding of the implementation of class-based storage necessary for the subsequent design chapters. The decisions do not influence the design of the standardised warehouse layout data model (see Chapter 5). The decisions are made for the design of the visualisation and report tools (discussed in Chapter 6). These decisions are necessary to enable visualisations in Power BI. To align the objective of Bricklog to make a visualisation that consider the needs of the majority of the customers, the decisions are made to represent a most common class-based storage analysis. If customers have specific requirements and want to differ from the decisions made, then individualised adjustments are possible against a surcharge.

Table 4.1: Decisions made for modelling class-based storage assignment.

No.	Choices	Decisions
1	Number classes	3
2	Performance indicator for products	Frequency product on a sales order line
3	Ratio of products to classes	A: 80%, B: 15%, C: 5%
4	Zone boundaries	Based on the average inventory level
5	Review date of allocation	None

4.2 Number of classes

There are no strict guidelines in literature to help determine the right number of classes. However, any small number of classes provides a near optimal solution on travel time of order pickers [25]. Since the travel time is not sensitive for the number of classes, the number of classes can be reduced to reduce the required storage space. Therefore, typically between two and six classes are recommended [6, 26, 27].

For the visualisation tools, the number of classes is set to three, which is similar to conventional ABC-storage policies. To underpin the findings of Yu et al. it is simulated that an AS/RS class-based storage with only three classes already provides nearly the same savings as storage policies with more classes [20, 19]. Besides a lower required storage space for less classes, there is another advantage of keeping the classes limited. The segmented customers of Bricklog are small warehouses that still do many activities manual. The lower the number of classes keeps the inbound process and the data processing simpler; it is easier for order pickers to remember three zones than six zones.

4.3 Performance criteria for product classes

There are three common characteristics (or termed criteria or performance indicators) of product items to use for class partitioning: *(1) product affinity, (2) popularity* and *(3) space* [28]. The correlation between products (also referred to as correlation) is about the frequency that product items are in the same order. The travel distance in a multi-order picking process is reduced if items in the same order are close to each other. Popularity is, for example, about the average number of picks per day, about the mean storage time of each product, or about the sales of an item over time. Space refers to the dimensions of the product. Finding an optimal allocation of products that coincides with all the characteristics is not possible. Therefore, some compromises should be made.

Introduced by Heskett in 1963 [29], the cube-per-order index (COI) is a method that combines criteria to allocate products based on their space requirements and popularity. Kallina et al. [28] were the first to show that the COI policy is optimal in some cases. The COI is used in many storage assignment models, where the storage process acts with a single command cycle. Schuur [30] researched whether there is a worst-case scenario of the COI rule. He concluded that COI is an excellent choice if single command retrievals are dominant. However, if the COI rule is implemented in a multi-command system, then it is possible to get far from optimal travel distance results. There are other methods that combine the size of products and popularity [31, 32], or product affinity and popularity [33].

If only one performance indicator is used to divide the SKUs between classes, then it is often popularity [34]. Popularity can be expressed as monetary turnover or turnover frequency. The turnover frequency is, for example, measured by the number of pick lines per time unit or the number of items of product per time unit [14].

The visualisation tools will use as *popularity* as performance indicator. Specifically, popularity measured as the number of times a product is on a sales order line over a certain time period. Using popularity as a performance indicator is the most common for ABC-analysis [34]. Furthermore, the data on sales order lines is most likely always available from the warehouse. Unlike data on the volume of products, which is another performance indicator that could have been selected. The history data available for the calculation the popularity of one product stretches one year. To get the most representative results, the data of that one year will be used to calculate the frequency of a product on a sales order line.

4.4 Ratio between products over the classes

Similar for the number of classes, there are also no strict guidelines for partitioning products into classes. The optimal ratio also depends on the selected performance indicator. In the case of three classes, *A products* commonly represents 5-33%, *B products* 15 - 33% and *C products* 25-50% of the product items [16].

For the visualisation tools, the ratio of 80-15-5 for assigning products to the A-B-C classes is used. These are common used values based on the 80/20 demand pattern [35]. The 80-15-5 ratio for the allocation of products to classes means that *class A* is filled with products until the products in the class account for 80% of the products. *Class B* is filled until 15% of the products have a class allocated. The remaining 5% of products will form *class C*.

4.5 Zone partitioning

The next choice is about the storage boundaries, or zones, of the classes. The optimal solution for determining the class locations depends on the layout of the warehouse, the routing policies, and the pick-list sizes [14]. For a single-order picking system, the closest locations to the I/O point are for A products.

The boundaries for the zones depend on the required storage space (RSS) of all products in the class. Guo et al. points out that existing research assumes that the required storage space (RSS) of items equal the average inventory level [36]. Often the adaption of arbitrary zone sizes is also a common practice, because zone simulation optimisation require relatively high computation times [35]. An example of an arbitrary zone size is 50/30/20 for three classes with a 80/20 demand curve.

The zones in the visualisation tools are determined by the average inventory level of all the products in the items. The average inventory level of an item in a class is denoted as $\overline{I}_{\text{item, class}}$. The average inventory level of all the products in a class can be calculated by summing up the average inventory levels of each item within that class:

$$\overline{I}_{\text{class}} = \sum_{\text{item}} \overline{I}_{\text{item, class}} \tag{4.1}$$

Similarly, the sum of the average inventory levels of all products across all classes is denoted as $\sum \overline{I}_{\text{item, class}}$. To determine the zones in the visualisation tools, the percentage for each class is calculated based on the average inventory levels:

$$\text{Percentage}_{\text{class}} = \frac{\overline{I}_{\text{class}}}{\sum \overline{I}_{\text{item, class}}} \times 100\% \tag{4.2}$$

By dividing the average inventory level of a class by the sum of the average inventory levels of all products, a percentage value is obtained that represents the contribution of that class to the overall inventory. This ensures that the total percentage across all classes adds up to 100%.

4.6 Review frequency of class-based storage assignment

Dependent on the types of products in a warehouse, there can be a strong trend in the popularity of product items over time. This is also called “seasonality”. If the performance indicator “popularity” is used as a performance indicator for storage assignment, then it can be optimal to periodically review the allocation of product items to a class. This could result in the periodic relocation of SKUs to match the demand pattern [26].

For the visualisation tools the review of the class allocation is eliminated. The class allocation will be based on all available historic data. This is a one year period.

4.7 Chapter 4 conclusion

The five choices and decisions for the implementation of class-based storage form the answer to sub-question 2: *What are necessary decisions for the implementation of a class-based storage policy?* In summary, for implementation warehouse managers need to decide on (1) the number of classes, (2) the performance indicator, (3) the ratio of product class allocation, (4) zone boundaries and (5) the review frequency.

5 | DEVELOPMENT OF A STANDARDISED WAREHOUSE LAYOUT DATA MODEL

The standardised warehouse layout data model is a data model populated with data about relative positions of storage location. Section 5.1 introduces the set-up of the development of the data model. The development of the data model consists out of three steps. The first step in Section 5.2 sets the Location ID numbering and a coordinate system for all the locations. The second step in Section 5.3 generates the distance from each Location ID to the depot. The third step in Section 5.4 allocates a class to each location ID. The import of the table in Power BI is mentioned in Section 5.5. The concluding statements on the development of the layout data model is given in Section 5.6.

5.1 Set-up of the development

The development of the standardised layout data model consists of two elements, (1) a data source format and (2) calculations that configures the data source to a standardised data model. Both objectives are fulfilled in one Excel file that is used as a data source format, and runs a query using Visual Basic Code. The query consists out of three steps. Each step requires some input variables. Together, these variables form the data source format. The variables are listed in Table 5.1 for each step. Based on the input variables, calculations generate a table with data (see Table 5.2). The table is the standardised data layout data model. The data consists the missing data on the relative positions of storage locations within the warehouse. The generated table in Excel is loaded into Power BI as an additional table with an automatic query.

Table 5.1: The input variables of the data source format of the adapter.

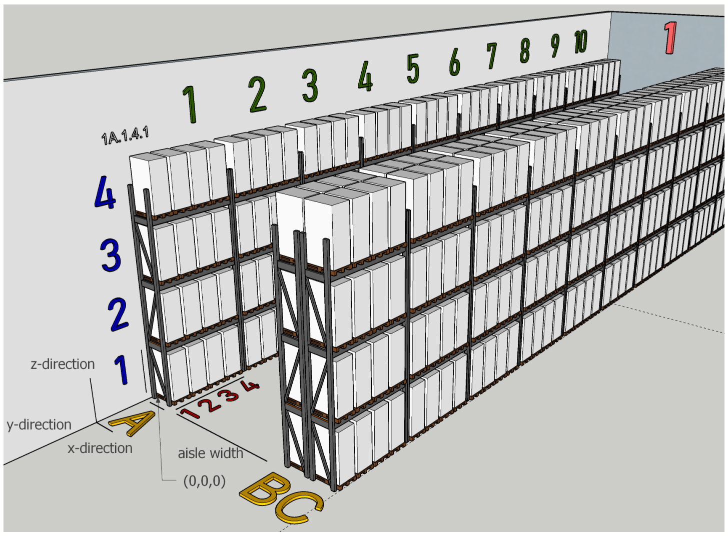
Step 1: location IDs and coordinates	Step 2: distance to depot	Step 3: class allocation
The number of the hall	X-coordinate of the depot (cm)	Percentage class A
Number rows	Y-coordinate of the depot (cm)	Percentage class B
Number bays	Z-coordinate of the depot (cm)	Percentage class C
Number levels	Z-factor	Percentage class D
Number places		Percentage class E
Aisle width (cm)		Percentage class F
Level length (cm)		
Level width (cm)		
Level heigth (cm)		

5.2 Setting a location identification system (step 1)

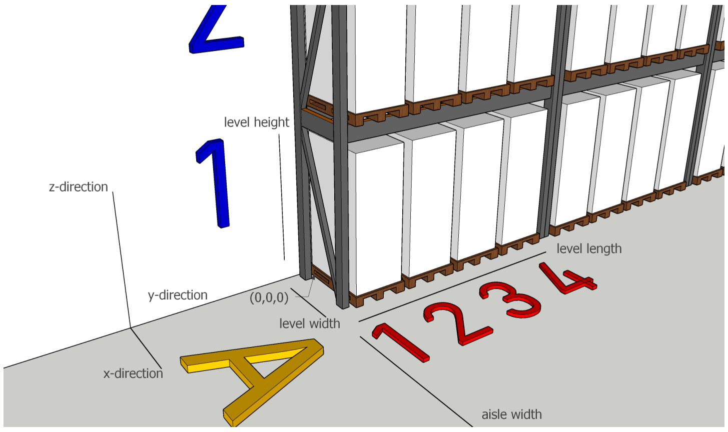
Step 1 sets the location identification system. The input variables that need to be filled in for step 1 are shown in Figures 5.1a and 5.1b. An example of the numbering system is given: 1A.1.4.1. The first 1 is the number of the hall, indicated pink in Figure 5.1a. This number can be any data type.

The next four input variables are integer numbers. The number of rows are the yellow letters. In the example figure the number of rows would be 3. The next number is the number of bays, indicated by the green numbers (1-10). The blue numbers form together the number of levels (1-4). The last number represents the number of places (1-4).

The aisle width is the width in centimeters between the rows. The level length, width and height are given in Figure 5.1b. In total these 9 variables make it possible to set a location numbering system, and to define the x, y and z-coordinate of each location.



(a) Input variables zoomed out.



(b) Input variables zoomed in.

Figure 5.1: Input variables given for a warehouse.

The assumptions made for the calculations of step 1 are:

- The storage system is pallet racking.
- The first row is always single. After the first aisle the second and third row are double. The fourth and fifth row appear after the second aisle. This logic repeats itself till the total number of rows is reached.
- The aisle width is constant.
- Each row starts on the x-axis and has the same direction and length.

If a warehouse layout does not adhere to the underlying assumptions, customised modifications can be implemented to the outcome of the adapter. Nevertheless, in order to develop a standardised tool, efforts should be directed towards minimising the need for customised interventions, whenever feasible.

The pseudo code for step 1 is given in Algorithm 1. The full code of all the steps is given in Appendix A. The calculations used in the code for the x-, y- and z-coordinates are given in Equations 5.1 - 5.3.

Algorithm 1 Step 1: transformation of the location IDs and coordinates of the standard data source format.

```
Input: Hall identifier, number of rows, number of bays, number of levels, number of places,
aisle width, level length, level width, level height.                    ▷ See Table 5.1
Output: Array with row index equal to the number of storage locations and column index equal
to 4.
    Column index 1: Unique location identifier.
    Column index 2: X-coordinate of the storage location.
    Column index 3: Y-coordinate of the storage location.
    Column index 4: Z-coordinate of the storage location.

1: Initialization of variables
2: Read the input values from the data source format
3: Set the number of rows to letters

4: for number of rows do
5:     for number of bays do
6:         for number of levels do
7:             for number of places do
8:                 Set the location numbering identification
9:                 Set x-coordinate                               ▷ See Equation 5.1
10:                Set y-coordinate                               ▷ See Equation 5.2
11:                Set z-coordinate                               ▷ See Equation 5.3
```

The number of rows, bays, levels and places are used as input for the calculations of the coordinates. As origin of the coordinate system the location 1A.01.1.1 is taken. The coordinate system is such set that x increases with an increase in number of rows. Y increases with the number of bays. Z increases with the number of levels. Figure 5.1 also indicates the direction of the axis.

$$\text{x-coordinate} = (\text{rows} - 0.5) \times \text{level width} + \text{RoundDown}(\left\lfloor \frac{\text{rows}}{2} \right\rfloor \times \text{aisleWidth}) \tag{5.1}$$

$$\text{y-coordinate} = (\text{bays} - 1) \times \text{level length} + \frac{\text{level length}}{\text{number of places}} \times (i - 1) \tag{5.2}$$

$$\text{z-coordinate} = \text{level height} \times (\text{levels} - 1) \tag{5.3}$$

5.3 Distance from the storage location to the depot (step 2)

The goal of this step is to generate the one way distance to the depot for all location IDs. As input variables the coordinates (x, y, z) of the depot should be given. As fourth input, the z-factor is added. The z-factor is an integer number. In the calculations for the distance from the locations to the depot, the z-factor is multiplied with the travel distance in the z-direction. The time it takes for a reach truck to travel distance in the z-direction, can be substantial longer than the travel distance on the ground. The z-factor can be used to increase the weight of the travel direction in the z-direction.

In literature most of the analysis of storage policies is calculated for AS/RS systems [3], for which travelling in the z-direction is no differently than traveling horizontally. Analysis and research for manual warehouses is often done with a simplified two-dimensional warehouse. In practice however, will an order picking in a manual warehouse almost always prefer to travel horizontally instead of vertically. Therefore, it is important to enable the possibility of putting more weight on the z-direction.

For warehouse distance calculations, two methods are common to use: the rectilinear calculation or the Chebyshev calculation. The distance calculations for the rectilinear case means that the movement of a pallet always occurs along one Cartesian axis at a time. The Chebyshev calculations allow travel to occur in multiple directions and also at different speeds [37]. The rectilinear calculations will be used for the calculations between the storage locations and the depot, because in manual warehouse the reach trucks will first move the pallets to ground level, and then start moving horizontally on the floor.

The pseudo code of the VBA code for step 2 is given in Algorithm 1. The calculations used in the code for the rectilinear distance is in Equation 5.4.

Algorithm 2 Step 2: adding distance to the depot for each storage location.

```
Input: array step 1 (output of Algorithm 1), x-coordinate depot, y-coordinate depot, z-coordinate depot, z-factor.
Output: Array with row index equal to the number of storage locations and column index equal to 5.
    Column index 1-4: equal to array of Algorithm 1.
    Column index 5: distance to the depot.
1: Run step 1
2: Initialisation of variables
3: Read the input values
4: for all locations do
5:     Calculate the rectilinear distance to the depot.
```

The distance to the depot for a storage location is given in Equation 5.4. “Location” refers to a storage location.

$$\text{Distance to depot} = \left|\text{Depot}_x - \text{Location}_x\right| + \left|\text{Depot}_y - \text{Location}_y\right| + \left|\text{Depot}_y - \text{Location}_z\right| \times \text{Factor}_z \tag{5.4}$$

5.4 Allocating a class to each storage location (step 3)

The percentages for the classes are used as input variables. These percentages add up to 100%. The adapter is at the moment able to handle six different classes. If necessary the adapter can easily extended to handle more classes. However, using more classes does not improve the order picking process significantly. Therefore, limiting the adapter to six classes is sufficient. The pseudo code for step 3 is given in Algorithm 3.

Algorithm 3 Step 3

```
Input: array step 2 (output of Algorithm 2), A percentage, B percentage, C percentage, D percentage, E percentage, F percentage.
Output: Array with row index equal to the number of storage locations and column index equal to 6.
    Column index 1-5: equal to array of Algorithm 2.
    Column index 6: class of the location.

 1: Run step 1 and 2
 2: Initialisation of variables
 3: Read the input values
                                                  ▷ Sort the locations on distance to depot
 4: for the first location to (last location - 1) do
 5:     for the second location to the last location do
 6:         if the latter location has a greater distance then Switch locations

 7: for each class do
 8:     Multiply percentage class with total locations and round down.
 9:     if locations are not filled due to round down then assign these locations to the last class

10: for all sorted locations do
11:     Assign the right class letter

12: Print all results to the Excel worksheet
13: Make a table of the results
```

The output of the last step is the standardised data layout model in the form of a table in Excel. These data set forms the input for the visualisation and report tools. The headers and two sample rows of the output of the adapter are portrayed in Table 5.2

Table 5.2: The table that is the result of the adapter. Two sample rows are added for illustration.

locationID	x-coordinate	y-coordinate	z-coordinate	distanceToDepot	class
1A.01.1.1	55	0	0	1055	A
1A.01.1.2	55	90	0	1145	A

5.5 Query for connecting the adapter with Power BI

The table from the Excel file should appear in the standardised data sets of Bricklog. The excel file is added as a source. The table is extracted from the file and the columns are automatically given the right data formats. The table should have an one-to-one connection, or one-to-many connection with the existing locations table to ensure a correct data flow.

5.6 Chapter 5 conclusion

The chapter answers the sub-question: *How to develop the standardised data layout model to complete the current data model?* In summary, the layout data model is developed by transformations of the input variables of the developed data source format. The result is an extra data table in the existing data model that consists the extra data necessary for the visualisation of the class-based storage analysis.

6 | DEVELOPMENT OF THE VISUALISATION AND REPORT TOOLS

The visualisation and reports tools are split into three categories: products (Section 6.1), locations (Section 6.2), analysis (Section 6.3). Existing Bricklog dashboards have the same kind of three-layer setup. For each category a separate dashboard is designed. The concluding remarks on the chapter are added in Section 6.4.

The dashboards are all designed to fulfill the requirements:

- Include navigation buttons to switch between the categories.
- Have filter options at the top of the page to enable the user to analyse the data themselves.
- Have a design in the Bricklog style.

The following assumptions are in place for the visualisation tools:

- Unit-load warehouse (which means a single order picking policy is in place).
- The retrieval of pallets in a single command cycle.
- Storage system: only pallet racking system.
- All products are EURO pallet sized.

6.1 The development of the "products" dashboard

The *products* reports and visualisations are designed to fulfill the following requirements:

1. Give an overview of all the products in the WMS.
2. Give standard characteristics about the products (such as name and category).
3. Return the amount of times a product has occurred in a sales order line.
4. Calculate the running total of orders for each product.
5. Assign classes to each product by using the running total and the ratio A products add up to 80% of the order lines, B up to 95% and C up to 100%.
6. Calculate summary statistics, such as the percentage products in the different classes, the percentage of orders that the classes represent.
7. Calculate the percentage of locations that should be reserved for the classes by making use of Equation 4.2.

The calculations on the running total of orders in percentage requires multiple steps. The calculations in Power BI are performed by writing Data Analysis Expressions, or in short DAX. The pieces of DAX code are written in "measures". Algorithm 4 gives the pseudo code for the DAX measure on the running total, Appendix C gives the full DAX measure.

Algorithm 4 Calculation for running total of the orders for each product in as percentage.

```
Input: all sales order lines, product ID
Output: running total of occurrence in sales order line for the product ID

1: Count number times product in sales order line.
2: A new "virtual" table that stores product ID and the count
3: Store the maximum order number of the "virtual" table
4: for each product do Sum the order counts between the order frequency of the product and the
   maximum
5: The running total of one product / total orders.
```

6.2 The development of the "locations" dashboard

The *locations* reports and visualisations are designed to fulfill the following requirements:

1. Give an overview of all the locations in the warehouse.
2. Return the distance to the depot from the table that is the output of the adapter.
3. Calculate the number of locations for each class and the percentage to check the output of the adapter.
4. Calculate the average distance and relative distance of the classes to class A.
5. Show a 3D model that illustrates which class is assigned to each location in the warehouse.

The 3D model is made with the software "SketchUp 2023". The software allows to make components. These components can be grouped to form a new component. The 3D model is set up by making first the pallet racking systems. Then one pallet is placed at the first location of the warehouse. That forms the first place. The first place is copied to the right number of places. The places are grouped to form a level components. The levels are grouped to bays components. The bays are grouped to rows components. The rows form a hall. The different components can be numbered by giving the components instance names. 3DBI for SketchUp software of KG-Dev is used as tool to make the connection between the SketchUp model and the Power BI dashboards. The SketchUp model in Figure 6.1 is selected on the fourth bay in one row. The relation with the other bays is visible. The grouping of components makes quick warehouse SketchUp modelling possible.

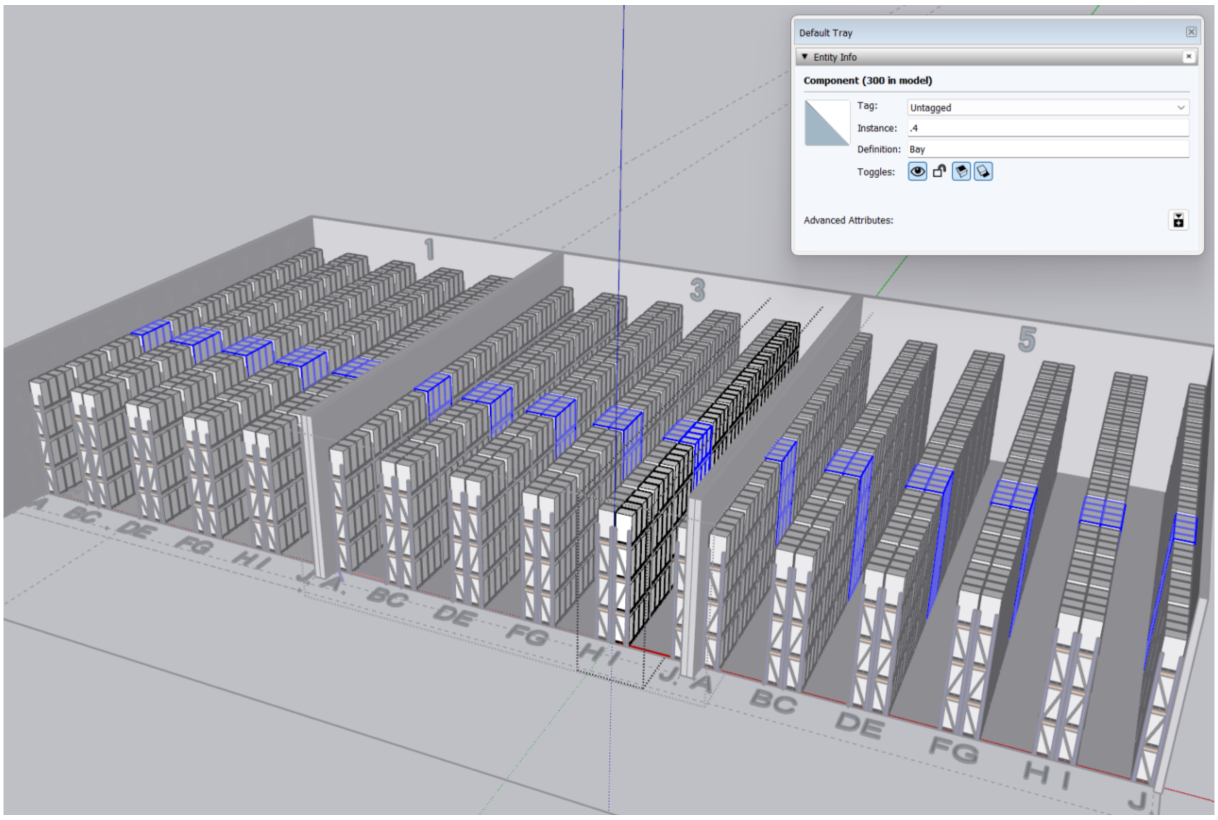

Default Tray
Entity Info
Component (300 in model)
Tag: Untagged
Instance: .4
Definition: Bay
Toggles:
Advanced Attributes:

Figure 6.1: SketchUp model built from different components, such as the bay component in the figure.

6.3 The development of the "analysis" dashboard

The analysis is based on information about "mutations" in warehouse. Mutations can take three forms: inbound, outbound and movement. The *analysis* reports and visualisations are designed to fulfill the following requirements:

1. Give an overview of all outbound mutations in the warehouse.
2. Give some standard characteristics of the mutation, such as outbound date, location from which is picked, the product that is picked.

3. Calculate the distances that the order pickers have traveled for the outbound locations by summing all the distances of the locations.
4. Calculate the distance that the order pickers would have travelled if the product that is picked is not located at the location of the mutation, but somewhere on a location inside the class of the product.
5. Calculate the difference between the historic travel distance and the class-based travel distance for order pickers.
6. Translate the difference to efficiency improvement statements that are of value to a warehouse manager.
7. Filter all the data in the report and visualisations tools on a selected year, month or day.

All calculations are basic calculations measures, except for the point 6. At the end a percentage reduction is given on the travel distance for a filtered year, month or day. For the last point, the difference in travel distance should be used to generate additional statistical measures that go beyond a percentage. Literature is used to add besides the percentage more statistics on the efficiency improvement of class based storage policy.

The time of an order picker is typically 50% spent on travel, see Figure 6.2. This means that if you were able to decrease the travel time with 25%, then one employee can pick the same number of orders in 87,5% of the time. The general formula for the statement: "Order pickers accomplish the same productivity level in X % of the regular time," is $X = (1 - k \cdot 0.5)$, for which k represents the ABC reduction parameter, where $0 < k \leq 1$.

Furthermore, to transfer the travel distance to time, the amount of meters is divided by 45.72. This is based on the same assumption as Petersen et al. used in a warehouse simulation for a manual warehouse with 10 picking aisles and a total storage capacity of 1000 SKUs [15].

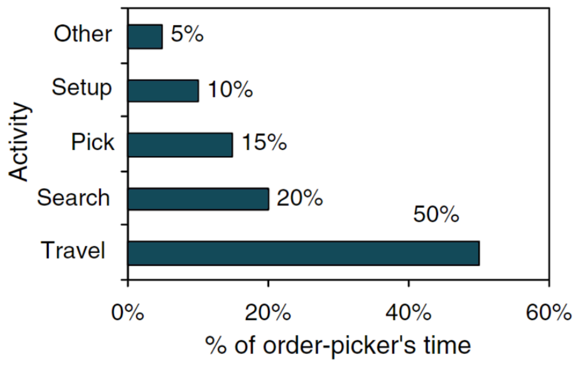


Figure 6.2: Typical distribution of an order picker's time [6, 22]

6.4 Chapter 6 conclusion

In this chapter, the fourth sub-question is answered: *How to develop the visualisation and report tools for the class-based storage policy in a warehouse?*. The *products* and *locations* dashboards showcase all the elements of the implementation of class-based storage. The *analysis* gives insight in the efficiency improvement of the class-based storage policy compared to the current implement policy.

7 | DEMONSTRATION

To demonstrate and test the designed adapter and the visualisation and report tools, a non-existent warehouse has been made up; “the demonstration warehouse”, or in short “demo-warehouse”. Section 7.1 describes the characteristics of the demo-warehouse. In Section 7.2 the design of the standardised layout model will be demonstrated for the demo-warehouse. Section 7.3 demonstrates the visualisation and report tools for the demo-warehouse. The chapter serves the purpose of testing the developed layout model and the visualisation and report tools. The conclusion is given in Section 7.4.

7.1 Characteristics of the demonstration warehouse

The demo-warehouse has 10 aisles, each consisting of 10 bays, with 4 levels and each level on a bay has 4 places. This sums up to a total of 1600 storage locations. To be able to test the adapter properly, two different I/O points can be considered. Depot 1 is the I/O point in the centre. Depot 2 is the I/O point in the left bottom corner. Figure 7.1 portrays the 3D-sketched demo-warehouse.

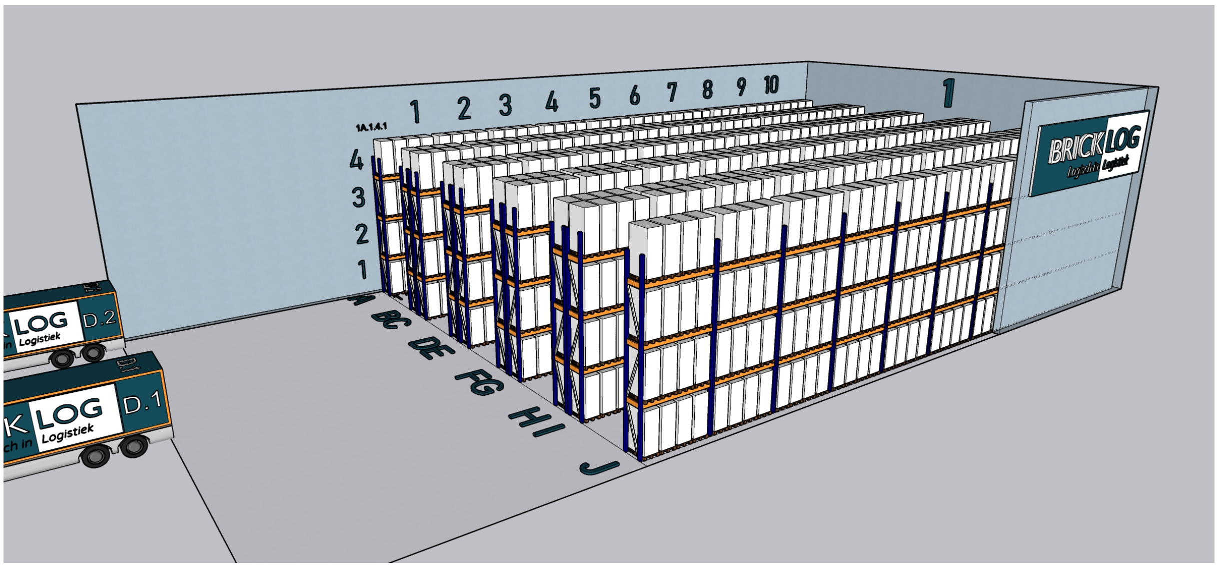


Figure 7.1: 3D-sketch of the demo-warehouse including depot 1 (D.1) and depot 2 (D.2).

7.1.1 Dummy data

The data that would normally be retrieved from a WMS has been generated by scripts made by a Bricklog employee. The WMS data that is generated is so called “dummy data”. The dummy data is random, except for the occurrence of Product ID in the sales order lines. Approximately 20% of the products account for 80% of the order lines, the next 30% products account for 15% of the product, and the last 50% products make up 5% of the order lines. The random dummy data indicates that the analysis in the demonstration will be a comparison between random storage policy and class-based storage policy.

7.2 Demonstration of the standardised layout data model

The adapter is tested in three scenarios to be able to evaluate the use and result of the adapter. Table 7.1 gives the input values for the scenarios. The values with an “x” indicate that these values differ between the scenarios.

Table 7.1: The input values for the test scenarios of the adapter.

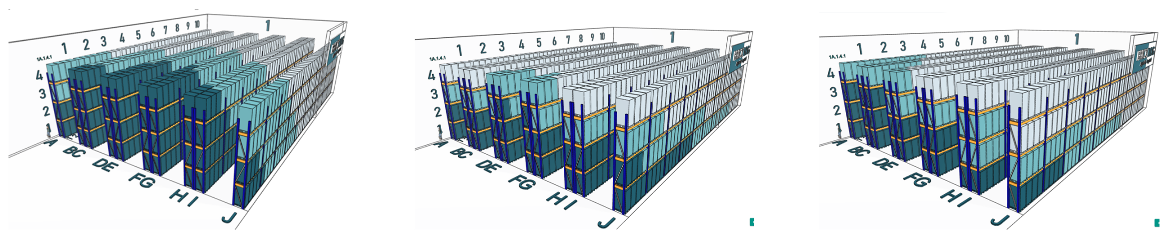
Step 1		Step 2		Step 3	
The number of the hall	1	X-coordinate of the depot (cm)	**x**	Perc. class A	70
Number rows	10	Y-coordinate of the depot (cm)	-1000	Perc. class B	20
Number bays	10	Z-coordinate of the depot (cm)	0	Perc. class C	10
Number levels	4	Z-factor	**x**	Perc. class D	0
Number places	4			Perc. class E	0
Aisle width (cm)	305			Perc. class F	0
Length of 1 level (cm)	360				
Width of 1 level (cm)	110				
Height of 1 level (cm)	205				

In the first scenario, depot 1, the depot centre, is taken as I/O point. The x-coordinate is 1312. The z-factor is set at 1. In the second scenario, the z-factor is changed to 6. In the third scenario, depot 2 is used as an I/O point. The x-coordinate is 0.

Table 7.2: The test scenarios for the adapter.

	Depot location	Z-factor
Scenario 1	Center, x = 1320	1
Scenario 2	Center, x = 1320	6
Scenario 3	Left, x = 0	6

To test if the adapter results make sense, the classes can be modelled in the 3D sketch of the warehouse, see Figure 7.2. The depot is in front of the start of the rows. Therefore the darkest color is class A, the lightest color class C.



(a) Scenario 1 **(b)** Scenario 2 **(c)** Scenario 3

Figure 7.2: Scenarios to test the adapter. The dark blue locations that belong to class A are representing the closest locations to the different configurations of the depot.

In scenario 2 compared to scenario 1, the z-factor is increased. This results in less high-level A-class locations. In Scenario 3 compared to Scenario 1, the depot is altered to the left. This results in more A-class locations to the left of the warehouse.

The results of the warehouse layout data model give the expected results. It can be concluded that the method is working properly. Generating the different visualisations of the visualisations is also done within a few minutes. This emphasises the ease of use of the standardised data source format and the corresponding standardised data model.

7.3 Demonstration of the visualisation and report tools

The dashboards in Appendix B have been enlarged for better visibility. Each dashboard shares a consistent layout, including a navigation option, general filter options, an overview component, and statistics components. This layout ensures a clean and organized data analysis for users. The overview component offers a snapshot of key metrics and trends, while the statistics components offer information on the overview component.

7.3.1 The demonstration of the "products" dashboard

The results of the dashboard is that for the ratio 80-15-5 percentage filled in the DAX measure ProductClass (see Appendix C), the result is a product class allocation in which 74% is class A, 21% class and 5% class C. Table 7.3 gives an overview of the results which are as expected. The average product percentage is used as input for the adapter to determine the class allocation for the locations.

Table 7.3: Results of the products visualisation and report tool. The 80/20 demand curve is visible in the results, as 74% of the orders make up 18% of the products.

	Percentage	Percentage orders	Percentage products	Average product percentage
A	80%	74%	18%	31%
B	15%	21%	32%	38%
C	5%	5%	50%	31%

7.3.2 The demonstration of the "locations" dashboard

The input variables are set to the right values for the demonstration warehouse. The depot is assumed to be in the centre, with x-coordinate of 1320 cm. The percentages are set to 31% for class A, 38% for class B and 31% for class C. The result is a division of the locations as portrayed in Figure 7.3.

Portraying the class locations in a 3D visual is of especial added value for the order pickers in the warehouse. These employees are often less educated and for them it is of great value to see the warehouse that they work. It transfers the theory about class-based storage assignment to an understandable visualisation.

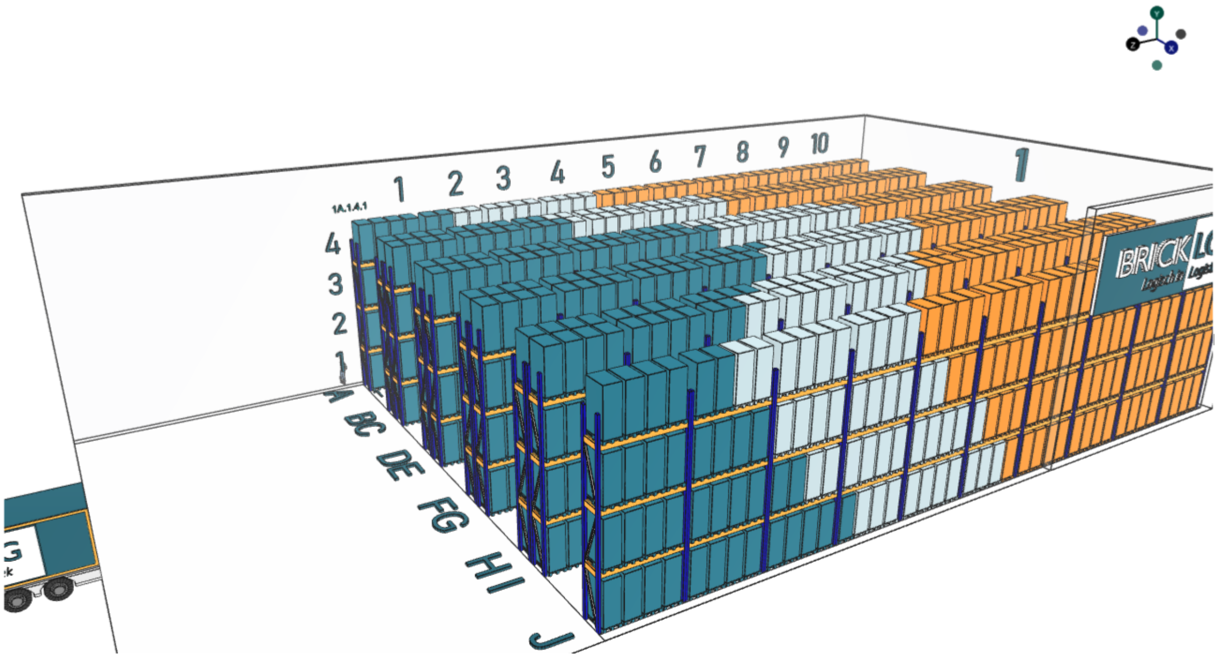


Figure 7.3: Result of the visualisation tool in the locations dashboard. The 31% closest locations for class A are given in dark blue. Light blue are the locations for class B, the orange locations represent class C.

7.3.3 The demonstration of the "analysis" dashboard

The analysis dashboard's results depend on the selected date, with the data from the first year, 2020, used as the selection criteria. The input variables remain the same as in Table 5.1, or they are modified to test the analysis. Table 7.4 presents the results of various tests, providing further insights into the efficiency improvement achieved by implementing class-based storage policies compared to random storage policies.

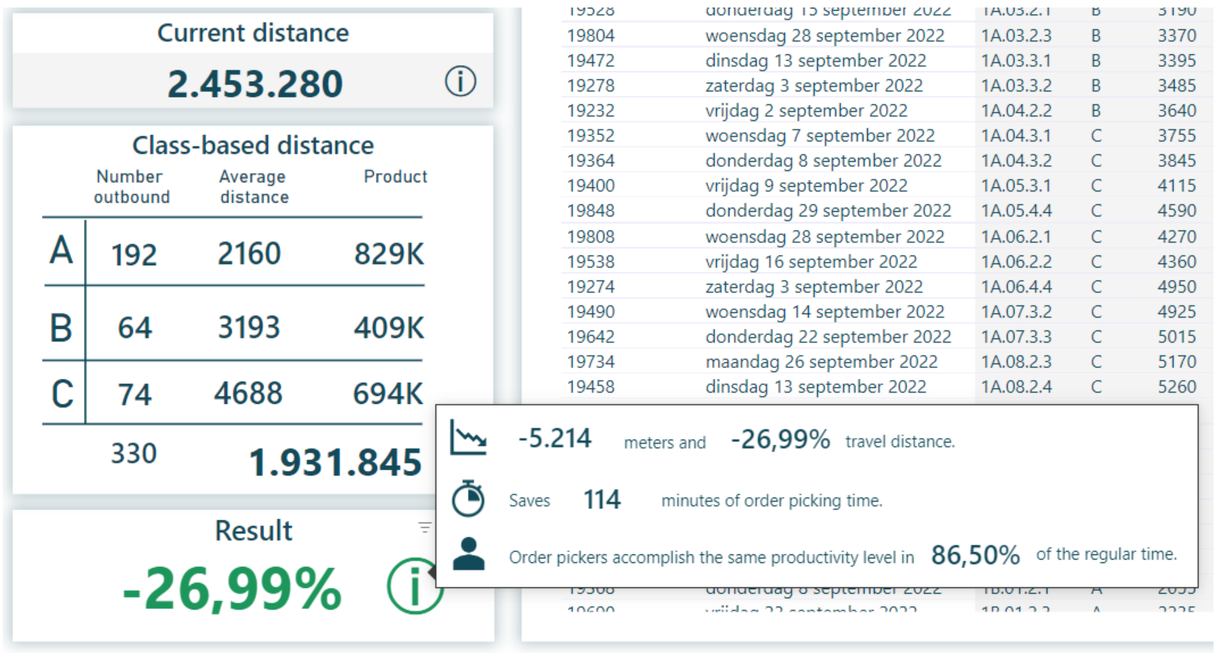
Test 1 is the analysis with the standard input variables. The difference for one year is a reduction of approximately 32% on the travel distance of order pickers. Test 2 and 3 increases the z-factor to 5 and 10 respectively. Increasing the z-factor lets the percentage reduction increase. The distance travelled for order picking in generally increases, because the travel distance in the z-direction is multiplied with the z-factor. The higher the distances the more reduction is possible with class-based storage policies. Test 6 also increases the travel distance by decreasing the Y-coordinate from -1000 to -2000. However, the extra 1000 meters is for every outbound mutation in place.Therefore the reduction in kilometers between test 1 and 6 is the same, but the reduction increase is lower for test 6 as the total travel distance increases.

Test 4 changed the percentage to 20, 30 and 50% respectively for class A, B and C. Using the arbitrary class location increases the reduction from 32% reduction to 49% reduction. Test 5 changes the depot from the centre to the left. The reduction is a bit lower, because the relative distance between the B and C classes compared to the A class locations is a bit lower.

Table 7.4: Results of the analysis visualisation and report tool. The reduction in travel distance of order pickers over a one-year pick history time frame is 32%.

Test	Location A%	Location B%	Location C%	Z-factor	X-coordinate depot	Y-coordinate depot	Reduction in travel distance in %
1	31%	38%	31%	1	1320	-1000	-32.39%
2	31%	38%	31%	**5**	1320	-1000	-34.49%
3	31%	38%	31%	**10**	1320	-1000	-46.51%
4	**20%**	**30%**	**50%**	1	1320	-1000	-49.04%
5	31%	38%	31%	1	**0**	-1000	-31.45%
6	31%	38%	31%	1	1320	**-2000**	-23.91%

In addition to the travel reduction percentage, the warehouse manager can hover over the information icon next to the percentage. Figure 7.4 displays the box that provides a more relatable number for the warehouse managers, translating the percentage into a concrete value. This allows the warehouse managers to gain insights not only into the reduction in travel distance but also the absolute decrease in travel distance. Furthermore, the distance in meters is translated into saved minutes of order picking time. Lastly, the warehouse manager can determine the fraction of time required to achieve the same productivity level.


Current distance

2.453.280

Class-based distance

	Number outbound	Average distance	Product
A	192	2160	829K
B	64	3193	409K
C	74	4688	694K
	330		1.931.845

Result

-26,99%

19528	donderdag 15 september 2022	1A.03.2.1	B	3190
19804	woensdag 28 september 2022	1A.03.2.3	B	3370
19472	dinsdag 13 september 2022	1A.03.3.1	B	3395
19278	zaterdag 3 september 2022	1A.03.3.2	B	3485
19232	vrijdag 2 september 2022	1A.04.2.2	B	3640
19352	woensdag 7 september 2022	1A.04.3.1	C	3755
19364	donderdag 8 september 2022	1A.04.3.2	C	3845
19400	vrijdag 9 september 2022	1A.05.3.1	C	4115
19848	donderdag 29 september 2022	1A.05.4.4	C	4590
19808	woensdag 28 september 2022	1A.06.2.1	C	4270
19538	vrijdag 16 september 2022	1A.06.2.2	C	4360
19274	zaterdag 3 september 2022	1A.06.4.4	C	4950
19490	woensdag 14 september 2022	1A.07.3.2	C	4925
19642	donderdag 22 september 2022	1A.07.3.3	C	5015
19734	maandag 26 september 2022	1A.08.2.3	C	5170
19458	dinsdag 13 september 2022	1A.08.2.4	C	5260
19368	donderdag 8 september 2022	1B.01.2.1	A	2055
19600	vrijdag 23 september 2022	1B.01.2.2	A	2225

-5.214 meters and -26,99% travel distance.

Saves 114 minutes of order picking time.

Order pickers accomplish the same productivity level in 86,50% of the regular time.

Figure 7.4: Box appears as extra information to the travel distance reduction percentage. The warehouse manager sees the reduction of order pick travel in minutes, and gets a productivity index. The Dutch dashboard settings of Bricklog turns the period into decimal as number separator.

7.4 Chapter 7 conclusion

The chapter answers the last sub-question: *What are the results of the standard warehouse layout model and the visualisation and report tools in a demonstration warehouse operating with a random storage policy?*

The functionality of the layout model is tested by analysing the results for different depot locations. As anticipated, the generated results from the data model are obtained within a minute, demonstrating its efficiency. Similarly, the visualisations and reports generated by the tools align logically with various input variables. By utilising dummy data that simulates picking data from a random order picking policy over one year, it is observed that the order pick travel is reduced by 32% with the implementation of a class-based storage policy. These diverse tests effectively demonstrate the capabilities and flexibility of the standardised visualisation and reporting tools.

8 | CONCLUSION

In the concluding chapter, the main findings are summarised in Section 8.1. The limitations of the research performed are outlined in Section 8.2. Section 8.3 critically assesses the flaws and deliverables of the thesis. Recommendations for Bricklog are provided in Section 8.4. Additionally, potential areas for further development of the deliverables are listed in Section 8.5.

8.1 Main findings

The main findings provide the answer to the research question: *How can visualisation and report tools be developed, utilising a standardised layout data model, to provide Bricklog's customers with insights into the implementation and operational efficiency of a class-based storage assignment policy within their warehouse?*

The research question is answered through the development of the following three components:

- A standardised layout data model

A method was devised to transform data from a standardised data source format (Table 5.1) into a table that captures all the necessary data for a class-based storage policy analysis (Table 5.2). This standardised layout data model serves as the foundation for generating insightful visualisations and reports.

- Visualisation and report tools for insights into the *implementation* of a class-based storage assignment policy.

Separate dashboards are created to guide warehouse managers in allocating classes to products and locations. The 3D visualisation of class locations enhances their understanding by providing a visual representation of the warehouse layout.

- Visualisation and report tools for insights into the *operational efficiency* of a class-based storage assignment policy.

The third dashboard provides insights into the reduced travel distance for order pickers when implementing a class-based storage policy compared to the current storage policy. The analysis shows that, on average, there is a 32% reduction in travel distance for order pickers when comparing a class-based storage policy with a random policy. It is important to note that this analysis assumes a single order picking, single command, unit load warehouse scenario.

8.2 Limitations

While this research has provided valuable insights into the implementation and operational efficiency of a class-based storage assignment policy, it is important to acknowledge limitations. Throughout the research, only dummy data was available for analysis. The absence of real data from an actual warehouse management system (WMS) of a customer limited the ability to evaluate the effectiveness of the developed tools in a real-world scenario.

In addition, the list of requirements for the product was relatively limited, allowing for considerable freedom in the development process. While this provided flexibility to explore various design possibilities, it may have also constrained the functionality of the final product. Conducting a more comprehensive analysis of customer requirements and expectations could have resulted in a more tailored and targeted solution.

Future studies should aim to address these limitations by incorporating real data from customer WMSs and conducting in-depth requirement analyses to further enhance the product's practicality and effectiveness. These recommendations are explained in more detail in Section 8.4.

8.3 Discussion

The data source format used in the research demonstrated its capability to configure location IDs for one hall at a time. However, it is important to note a limitation to configure a coordinate system for multiple halls at the same time. The best alternative for the use of a data source format would be the ability to automatically extract the coordinates from the SketchUp model and input them into Power BI. However, extracting the precise coordinates from SketchUp is currently a complex task that involves exploding all components in the model. It is a possiblity that SketchUp comes with a software improvement to address this issue.

While the developed dashboards provide valuable insights into class-based storage analysis, certain aspects warrant further consideration. One notable limitation is the absence of an analysis on optimal class and zone boundaries for products and locations. The dashboards rely on existing literature findings rather than conducting independent research in this area. Incorporating empirical research on optimal class and zone boundaries would enhance the precision and effectiveness of the class-based analysis.

Moreover, the class-based analysis could be further improved by incorporating seasonal considerations. By utilizing review dates for class allocation, warehouse managers could receive guidance on periodically relocating products within the warehouse. This would enable better adaptation to changing seasonal demands and improve overall operational efficiency.

It is important to acknowledge that the precise implementation of the product in practical warehouse settings may still require further refinement. The current level of clarity regarding its implementation and the extent of added value it provides could be further enhanced. Future iterations should focus on addressing these aspects to ensure seamless integration and maximise the benefits for warehouse managers.

8.4 Recommendations

Bricklog is currently in the initial stages of developing their BI product for warehouses. The primary recommendation for Bricklog is to connect the product to a Warehouse Management System (WMS) that is currently in use, rather than relying on self-developed dummy data. By automatically integrating the product with a WMS, new flaws and issues can be identified.

Furthermore, it is advisable for Bricklog to conduct a market analysis on the demand for a BI product that drives efficiency improvement. Currently, there is little to no research or documentation on the market's need for a BI product. Obtaining a list of requirements from potential customers regarding their preferred features in the product would greatly benefit its development. Currently, Bricklog's management is overseeing the product development process, but it could be enhanced by gaining more specific insights into the requirements of potential end users.

During this market research, it is particularly recommended to focus on understanding how the end users ultimately intend to utilize the BI product. For instance, a possible requirement could be that warehouse managers desire a print option for the 3D layout of the warehouse. This would enable order pickers to easily identify the corresponding location of each item. However, this remains unclear and should not be assumed without further investigation.

8.5 Further development

In this chapter, a list is given of potential further development extensions. These extensions aim to improve the functionality and applicability of the current developed BI product for warehouses.

- The standardised source data can be extended to include the transformation of input variables on storage systems other than pallet racks.
- In situations where multiple blocks of different pallet racking systems are present, it would be beneficial to add the capability to incorporate both systems into a unified coordinate system.
- Currently, the insights derived from the analysis focus on single order picking scenarios. To enhance the system's capabilities, it is recommended to enlarge the analysis to include multiple order picking in non-unit warehouses.
- While the current BI product generates visualisations based on the class-based storage policy, there is potential to expand the visualisations to encompass other organizational policies. By incorporating a wider range of policies, users can evaluate and compare the effectiveness of different storage strategies.
- Further research could involve expanding the analysis to consider seasonality factors. Currently, the research is limited to a class allocation at a single moment in time. By accounting for seasonal fluctuations in product demand, the system can provide more dynamic and adaptable recommendations for optimizing warehouse operations throughout the year.

REFERENCES

[1] Home - Bricklog.

[2] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Longman Publishing Co., Inc., USA, 1995.

[3] B. Rouwenhorst, B. Reuter, V. Stockrahm, G. J. Van Houtum, R. J. Mantel, and W. H.M. Zijm. Warehouse design and control: Framework and literature review. *European Journal of Operational Research*, 122(3):515–533, 5 2000.

[4] Kees Jan Roodbergen. *Layout and Routing Methods for Warehouses*. PhD thesis, 4 2001.

[5] Eleonora Bottani, Margherita Cecconi, Giuseppe Vignali, and Roberto Montanari. Optimisation of storage allocation in order picking operations through a genetic algorithm. *International Journal of Logistics Research and Applications*, 15(2):127–146, 4 2012.

[6] Jinxiang Gu, Marc Goetschalckx, and Leon F. McGinnis. Research on warehouse operation: A comprehensive review. *European Journal of Operational Research*, 177(1):1–21, 2 2007.

[7] G Richards and an O'Reilly Media Company Safari. *Warehouse Management, 2nd Edition*. Kogan Page, 2014.

[8] Kevin R. Gue, Goran Ivanović, and Russell D. Meller. A unit-load warehouse with multiple pickup and deposit points and non-traditional aisles. *Transportation Research Part E: Logistics and Transportation Review*, 48(4):795–806, 7 2012.

[9] John J Bartholdi and Steven T Hackman. Warehouse & Distribution Science. 2019.

[10] Hans Heerkens and Arnold van Winden. *Solving Managerial Problems Systematically*. Noordhoff Uitgevers, 2017.

[11] Ken Peffers, Tuure Tuunanen, Marcus A. Rothenberger, and Samir Chatterjee. A design science research methodology for information systems research. *Journal of Management Information Systems*, 24(3):45–77, 12 2007.

[12] J A Tompkins. *Facilities Planning*. Facilities Planning. J. Wiley, 2003.

[13] J J Coyle, E J Bardi, and C J Langley. *The Management of Business Logistics: A Supply Chain Perspective*. SWC-Management Series. South-Western/Thomson Learning, 2003.

[14] Tho Le Duc. *Design and Control of Efficient Order Picking Processes*. 4 2005.

[15] Charles G Petersen and Gerald Aase. A comparison of picking, storage, and routing policies in manual order picking. *International Journal of Production Economics*, 92(1):11–19, 2004.

[16] Alfred Ultsch. Proof of Pareto's 80/20 Law and Precise Limits for ABC-Analysis. 2002.

[17] Warren Hausman, Leroy Schwarz, and Stephen Graves. Optimal Storage Assignment in Automatic Warehousing Systems. *Management Science*, 22:629–638, 5 1976.

[18] Jinxiang Gu, Marc Goetschalckx, and Leon F. McGinnis. Research on warehouse design and performance evaluation: A comprehensive review. *European Journal of Operational Research*, 203(3):539–549, 6 2010.

[19] Meir J Rosenblatt and Amit Eynan. Deriving the Optimal Boundaries for Class-Based Automatic Storage/Retrieval Systems. *Management Science*, 35(12):1519–1524, 1989.

[20] Amit Eynan and Meir J Rosenblatt. ESTABLISHING ZONES IN SINGLE-COMMAND CLASS-BASED RECTANGULAR AS/RS. *IIE Transactions*, 26(1):38–46, 1 1994.

[21] Tho Le Duc and René De Koster. Determining Number of Zones in a Pick-and-pack Orderpicking System. 5 2005.

[22] René de Koster, Tho Le-Duc, and Kees Jan Roodbergen. Design and control of warehouse order picking: A literature review. *European Journal of Operational Research*, 182(2):481–501, 10 2007.

[23] Kees Jan Roodbergen. Storage Assignment for Order Picking in Multiple-Block Warehouses. pages 139–155, 5 2012.

[24] SAP Documentation. Settings for the Travel Distance Calculation (SAP Library - Travel Distance Calculation), 2013.

[25] Yugang Yu, René B M de Koster, and Xiaolong Guo. Class-Based Storage with a Finite Number of Items: Using More Classes is not Always Better. *Production and Operations Management*, 24(8):1235–1247, 2015.

[26] Charles Petersen, Gerald Aase, and Daniel Heiser. Improving order-picking performance through the implementation of class-based storage. *International Journal of Physical Distribution & Logistics Management*, 34:534–544, 8 2004.

[27] Jeroen Berg and Noud Gademann. Simulation study of an automated storage/retrieval system. *International Journal of Production Research*, 38:1339–1356, 4 2000.

[28] Carl Kallina and Jeffrey Lynn. Application of the Cube-per-Order Index Rule for Stock Location in a Distribution Warehouse. 7(1):37–46, 1976.

[29] J.L. Heskett. Cube-Per-Order Index: A Key to Warehouse Stock Location. *Transportation and Distribution Management*, 3:27–31, 1963.

[30] Peter C. Schuur. The worst-case performance of the Cube per Order Index slotting strategy is infinitely bad – A technical note. *International Journal of Production Economics*, 170:801–804, 12 2015.

[31] Rommert Dekker, René De Koster, Kees Jan Roodbergen, and Herco Van Kalleveen. Improving order-picking response time at Ankor′s warehouse. *Interfaces*, 34(4):303–313, 2004.

[32] Seval Ene and Nursel Öztürk. Storage location assignment and order picking optimization in the automotive industry. *The International Journal of Advanced Manufacturing Technology*, 60(5):787–797, 2012.

[33] Jiaxi Li, Mohsen Moghaddam, and Shimon Y Nof. Dynamic storage assignment with product affinity and ABC classification—a case study. *The International Journal of Advanced Manufacturing Technology*, 84(9):2179–2194, 2016.

[34] Felix T.S. Chan and H. K. Chan. Improving the productivity of order picking of a manual-pick and multi-level rack distribution warehouse through the implementation of class-based storage. *Expert Systems with Applications*, 38(3):2686–2700, 3 2011.

[35] Allyson Silva, Kees Jan Roodbergen, Leandro C Coelho, and Maryam Darvish. Estimating optimal ABC zone sizes in manual warehouses. *International Journal of Production Economics*, 252:108579, 2022.

[36] Xiaolong Guo, Yugang Yu, and René B M De Koster. Impact of required storage space on storage policy performance in a unit-load warehouse. *International Journal of Production Research*, 54(8):2405–2418, 2016.

[37] Gilles Cormier and Eldon A Gunn. A review of warehouse models. *European Journal of Operational Research*, 58(1):3–13, 1992.

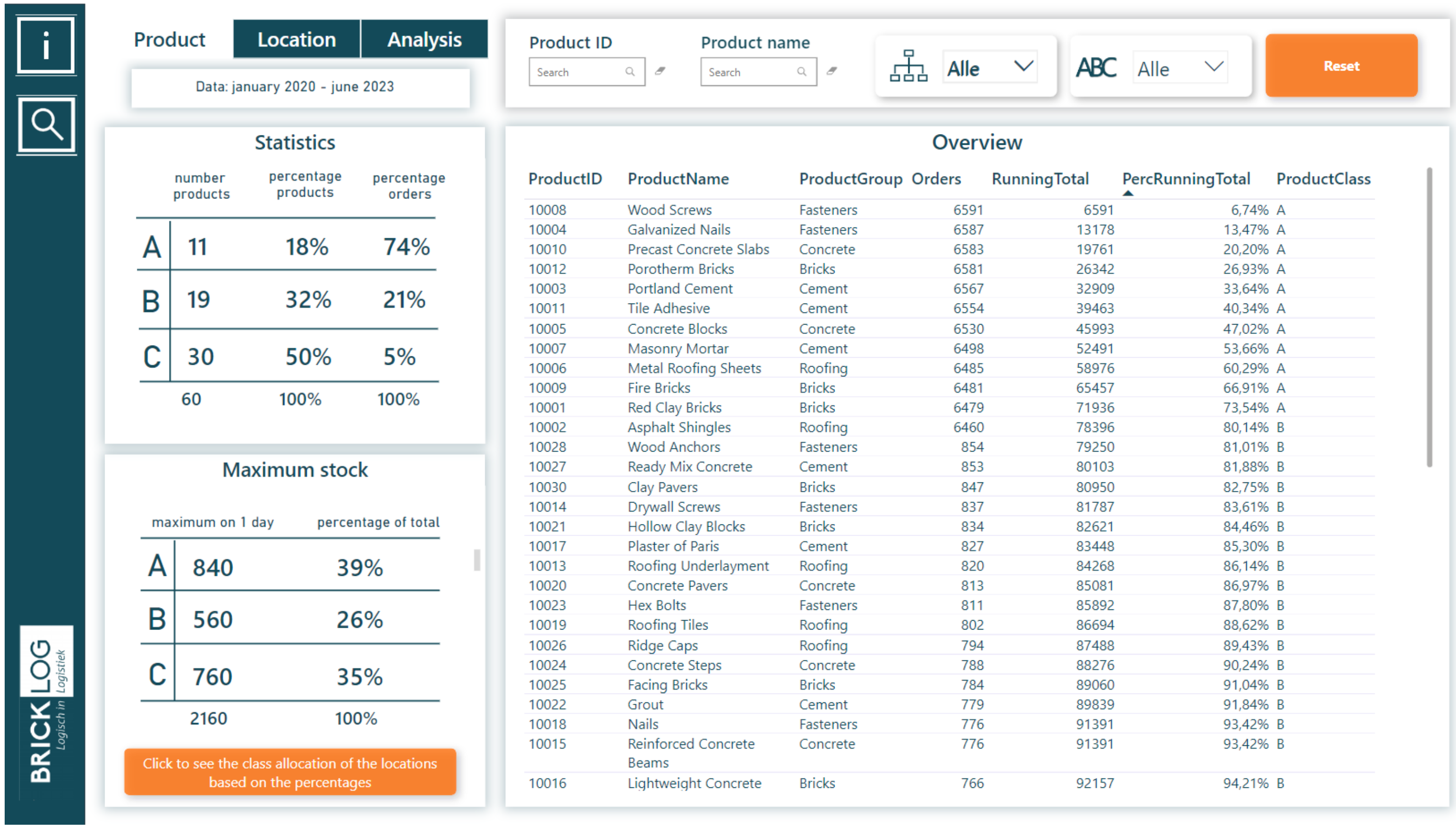
A | VBA CODE OF THE ADAPTER

In this thesis Visual Basic Application (VBA) is used. VBA is the programming language behind Microsoft Excel.

```
'Goal: the adapter for generating a data source on storage locations
'Date: 01-06-2023
'Made by Elise Huisman (2506416)

Option Explicit
Private Locations() As Variant

Sub step1retrieveCoordinates()
'Goal of the sub: for each locationID get x, y, z, coordinate

'Setting all variables
    'variable that store a static input value
Dim inputDict As Object
Set inputDict = CreateObject("Scripting.Dictionary")
Dim newDictionaryItem As String

Dim halls As Integer
Dim numberRows As Integer
Dim numberBays As Integer
Dim numberLevels As Integer
Dim numberPlaces As Integer
Dim length As Double
Dim width As Double
Dim height As Double
Dim aisleWidth As Double

    'loop variables
Dim i As Integer
Dim rows As Integer
Dim bays As Integer
Dim levels As Integer
Dim places As Integer

    'other variables
Dim totalNumberLocations As Integer
Dim AisleNumbering() As String
Dim placesDone As Integer

Call clearField

'Reading in the settings data
For i = 1 To 6
    newDictionaryItem = Worksheets("Input").Cells(2 + i, 1)
    inputDict.Add newDictionaryItem, Worksheets("Input").Cells(2 + i,
        2)
Next i

halls = inputDict("Hall")
numberRows = inputDict("#_Rows")
numberBays = inputDict("#_Bays")
numberLevels = inputDict("#_Levels")
numberPlaces = inputDict("#_Places")
aisleWidth = inputDict("Aisle_width")

length = Worksheets("Input").Cells(12, 1)
width = Worksheets("Input").Cells(12, 2)
height = Worksheets("Input").Cells(12, 3)

'Reading in the letters for the number of rows
ReDim AisleNumbering(1 To numberRows)
For i = 1 To numberRows
    AisleNumbering(i) = Worksheets("Alphabet").Cells(i, 1)
Next i

'adding headers for the data on the worksheet
Worksheets("Input").Cells(1, 8) = "locationID"
Worksheets("Input").Cells(1, 9) = "x-coordinate"
Worksheets("Input").Cells(1, 10) = "y-coordinate"
Worksheets("Input").Cells(1, 11) = "z-coordinate"

'Setting up array for all LocationIDs
totalNumberLocations = inputDict("#_Rows") * inputDict("#_Bays") *
   inputDict("#_Levels") * inputDict("#_Places")
ReDim Locations(1 To totalNumberLocations, 3)

'looping over all locations
placesDone = 0


For rows = 1 To numberRows
    For bays = 1 To numberBays
        For levels = 1 To numberLevels
            For places = 1 To numberPlaces
                'get numbering correct for the locationIDs
                If bays < 10 Then
                    Locations(placesDone * numberPlaces + places, 0)
                       = halls & AisleNumbering(rows) & ".0" & bays &
                        "." & levels & "." & places
                    Worksheets("Input").Cells(1 + placesDone *
                       numberPlaces + places, 8) = Locations(
                       placesDone * numberPlaces + places, 0)
                Else
                    Locations(placesDone * numberPlaces + places, 0)
                       = halls & AisleNumbering(rows) & "." & bays &
                    "." & levels & "." & places
                Worksheets("Input").Cells(1 + placesDone *
                    numberPlaces + places, 8) = Locations(
                    placesDone * numberPlaces + places, 0)
            End If

            'x-coordinate
            Locations(placesDone * numberPlaces + places, 1) = (
                rows - 0.5) * width + Application.
                WorksheetFunction.RoundDown(rows / 2, 0) * (
                aisleWidth)
            Worksheets("Input").Cells(1 + placesDone *
                numberPlaces + places, 9) = Locations(placesDone *
                 numberPlaces + places, 1)

            'y- coordinate
            Locations(placesDone * numberPlaces + places, 2) = (
                bays - 1) * length + length / numberPlaces * (
                places - 1)
            Worksheets("Input").Cells(1 + placesDone *
                numberPlaces + places, 10) = Locations(placesDone
                * numberPlaces + places, 2)

            'z-coordinate
            Locations(placesDone * numberPlaces + places, 3) =
                height * (levels - 1)
            Worksheets("Input").Cells(1 + placesDone *
                numberPlaces + places, 11) = Locations(placesDone
                * numberPlaces + places, 3)
        Next places
        placesDone = placesDone + 1
      Next levels
   Next bays
Next rows


End Sub

Sub step2locationToDepot()
'Goal: add distance of each location to the depot

Call step1retrieveCoordinates

'add header
Worksheets("Input").Cells(1, 12) = "distanceToDepot"

Dim xDepot As Double
Dim yDepot As Double
Dim zDepot As Double
Dim zFactor As Integer
Dim locationRange As Range
Dim numberLocationIDs As Integer
Dim i As Integer

xDepot = Worksheets("Input").Cells(22, 1)
yDepot = Worksheets("Input").Cells(22, 2)
zDepot = Worksheets("Input").Cells(22, 3)
zFactor = Worksheets("Input").Cells(28, 2)

'get the amount of location IDs
Set locationRange = Worksheets("Input").Range("H2").CurrentRegion
numberLocationIDs = locationRange.rows.Count - 1 'minus the header
   row

'add extra column to the array
ReDim Preserve Locations(1 To numberLocationIDs, 4)

For i = 1 To numberLocationIDs
    'the x and y distance is the (absolute value) difference between
       the depot and the point
    'the z direction is multiplied by the direction
    Locations(i, 4) = Abs(xDepot - Locations(i, 1)) + Abs(yDepot -
       Locations(i, 2)) + (Abs(zDepot - Locations(i, 3)) * zFactor)
    Worksheets("Input").Cells(i + 1, 12) = Locations(i, 4)
Next i

End Sub

Sub step3classAllocation()
'Goal: adding a class to all locations depending on the distance to
   the depot

Call clearField
Call step1retrieveCoordinates
Call step2locationToDepot

Worksheets("Input").Cells(1, 13) = "Class"

'declare array that stores the input values
Dim classes(5, 2) As Variant

Dim numberLocationIDs As Integer
Dim i As Integer
Dim j As Integer
Dim k As Integer
Dim temporaryVar(4) As Variant
Dim numberLocationsWithClass As Integer
Dim LocationsFilledIn As Integer


numberLocationIDs = UBound(Locations, 1)
ReDim Preserve Locations(1 To numberLocationIDs, 5)
```

```
'Sorting array, based on bubbleSort method
For i = 1 To numberLocationIDs - 1
    For j = i + 1 To numberLocationIDs
        If Locations(i, 4) > Locations(j, 4) Then
            For k = 0 To 4
                temporaryVar(k) = Locations(j, k)
                Locations(j, k) = Locations(i, k)
                Locations(i, k) = temporaryVar(k)
            Next k
        End If
    Next j
Next i

'get percentage of classes, the corresponding number of locations for
    each class, and the letter of each class
For i = 0 To 5
    classes(i, 0) = Worksheets("Input").Cells(37 + i, 2)
    classes(i, 1) = Application.WorksheetFunction.RoundDown(classes(i
        , 0) * numberLocationIDs, 0)
    classes(i, 2) = Worksheets("Input").Cells(37 + i, 1)
    numberLocationsWithClass = numberLocationsWithClass + classes(i,
        1)
Next i

    'if the last locations do not have a class assigned (because of
       roundDown) then the following code assures all locationIDs
       have a class
If numberLocationsWithClass < numberLocationIDs Then
    For i = 5 To 0 Step -1
        If classes(i, 0) <> 0 Then
            classes(i, 1) = classes(i, 1) + (numberLocationIDs -
                numberLocationsWithClass)
            Exit For
        End If
    Next i
ElseIf numberLocationsWithClass > numberLocationIDs Then
    MsgBox ("The␣sum␣of␣percentages␣of␣the␣classes␣should␣equal␣100%"
        )
    Exit Sub
End If


For i = 5 To 0 Step -1
    If classes(i, 1) > 0 Then
        For j = numberLocationIDs - LocationsFilledIn To
            numberLocationIDs - LocationsFilledIn - classes(i, 1) Step
             -1
            If j = 0 Then
                Exit For
                End If

                Locations(j, 5) = classes(i, 2)
            Next j
            LocationsFilledIn = LocationsFilledIn + classes(i, 1)
            If LocationsFilledIn >= numberLocationIDs Then
                Exit For
            End If
        End If
Next i

Call printWholeArray(UBound(Locations, 1), UBound(Locations, 2))
Call MakingTable

End Sub

Sub clearField()

Worksheets("Input").Range("H2").CurrentRegion.Clear

End Sub

Sub printWholeArray(rowsLocations As Integer, columnsLocations As
    Integer)

Dim i As Integer
Dim j As Integer

Worksheets("Input").Cells(1, 8) = "locationID"
Worksheets("Input").Cells(1, 9) = "x-coordinate"
Worksheets("Input").Cells(1, 10) = "y-coordinate"
Worksheets("Input").Cells(1, 11) = "z-coordinate"
Worksheets("Input").Cells(1, 12) = "distanceToDepot"
Worksheets("Input").Cells(1, 13) = "class"

i = UBound(Locations, 1)

For i = 1 To rowsLocations
    For j = 1 To columnsLocations + 1
        Worksheets("Input").Cells(i + 1, j + 7) = Locations(i, j - 1)
    Next j
Next i

End Sub
```

B | DASHBOARDS (ENLARGED)



Data: january 2020 - june 2023

Statistics

	number products	percentage products	percentage orders
A	11	18%	74%
B	19	32%	21%
C	30	50%	5%
	60	100%	100%

Maximum stock

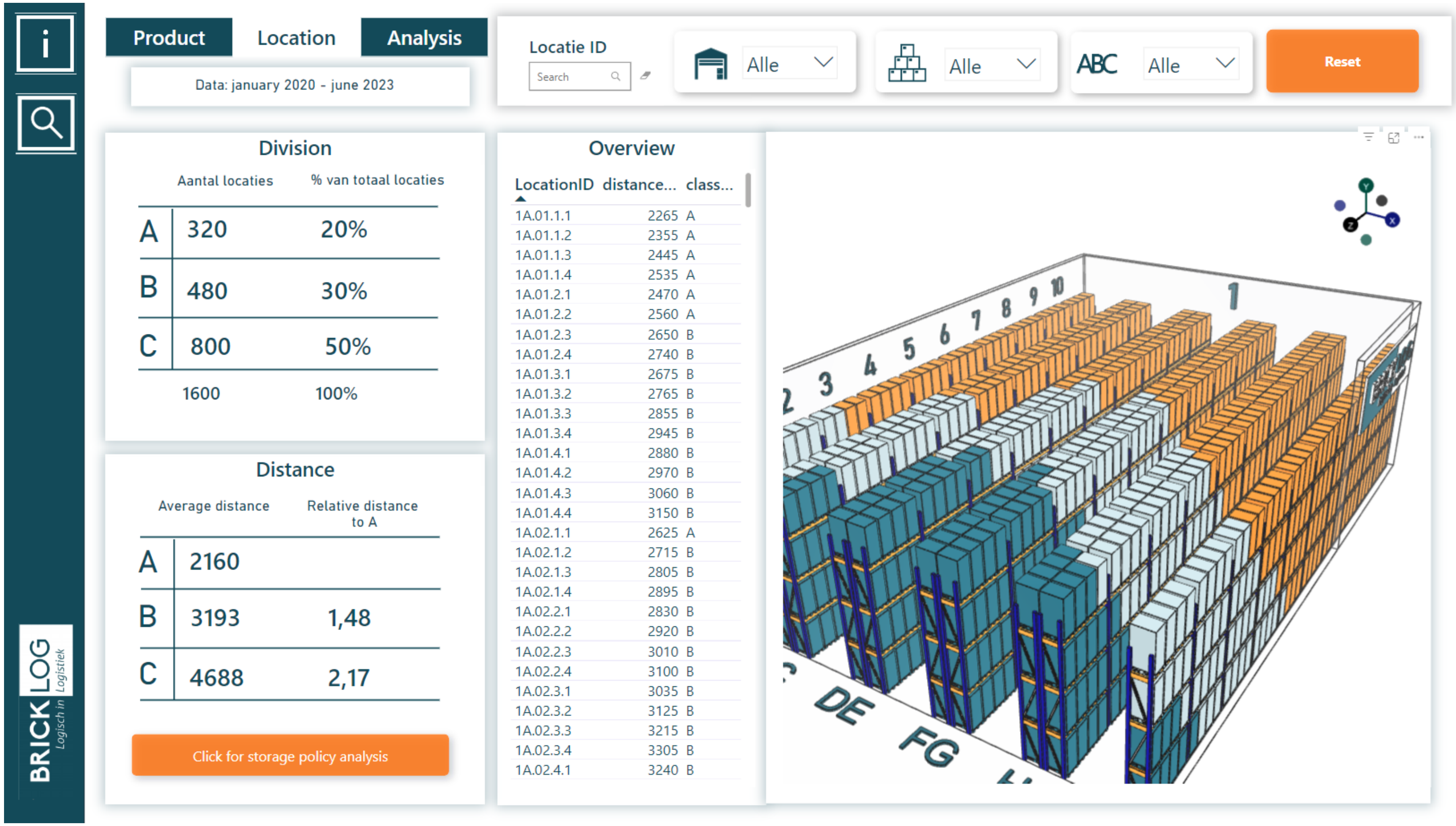
	maximum on 1 day	percentage of total
A	840	39%
B	560	26%
C	760	35%
	2160	100%

Click to see the class allocation of the locations based on the percentages

Overview

ProductID	ProductName	ProductGroup	Orders	RunningTotal	PercRunningTotal	ProductClass
10008	Wood Screws	Fasteners	6591	6591	6,74%	A
10004	Galvanized Nails	Fasteners	6587	13178	13,47%	A
10010	Precast Concrete Slabs	Concrete	6583	19761	20,20%	A
10012	Porotherm Bricks	Bricks	6581	26342	26,93%	A
10003	Portland Cement	Cement	6567	32909	33,64%	A
10011	Tile Adhesive	Cement	6554	39463	40,34%	A
10005	Concrete Blocks	Concrete	6530	45993	47,02%	A
10007	Masonry Mortar	Cement	6498	52491	53,66%	A
10006	Metal Roofing Sheets	Roofing	6485	58976	60,29%	A
10009	Fire Bricks	Bricks	6481	65457	66,91%	A
10001	Red Clay Bricks	Bricks	6479	71936	73,54%	A
10002	Asphalt Shingles	Roofing	6460	78396	80,14%	B
10028	Wood Anchors	Fasteners	854	79250	81,01%	B
10027	Ready Mix Concrete	Cement	853	80103	81,88%	B
10030	Clay Pavers	Bricks	847	80950	82,75%	B
10014	Drywall Screws	Fasteners	837	81787	83,61%	B
10021	Hollow Clay Blocks	Bricks	834	82621	84,46%	B
10017	Plaster of Paris	Cement	827	83448	85,30%	B
10013	Roofing Underlayment	Roofing	820	84268	86,14%	B
10020	Concrete Pavers	Concrete	813	85081	86,97%	B
10023	Hex Bolts	Fasteners	811	85892	87,80%	B
10019	Roofing Tiles	Roofing	802	86694	88,62%	B
10026	Ridge Caps	Roofing	794	87488	89,43%	B
10024	Concrete Steps	Concrete	788	88276	90,24%	B
10025	Facing Bricks	Bricks	784	89060	91,04%	B
10022	Grout	Cement	779	89839	91,84%	B
10018	Nails	Fasteners	776	91391	93,42%	B
10015	Reinforced Concrete Beams	Concrete	776	91391	93,42%	B
10016	Lightweight Concrete	Bricks	766	92157	94,21%	B

Figure B.1: The ”products” report and visualisation overview.



Division

	Aantal locaties	% van totaal locaties
A	320	20%
B	480	30%
C	800	50%
	1600	100%

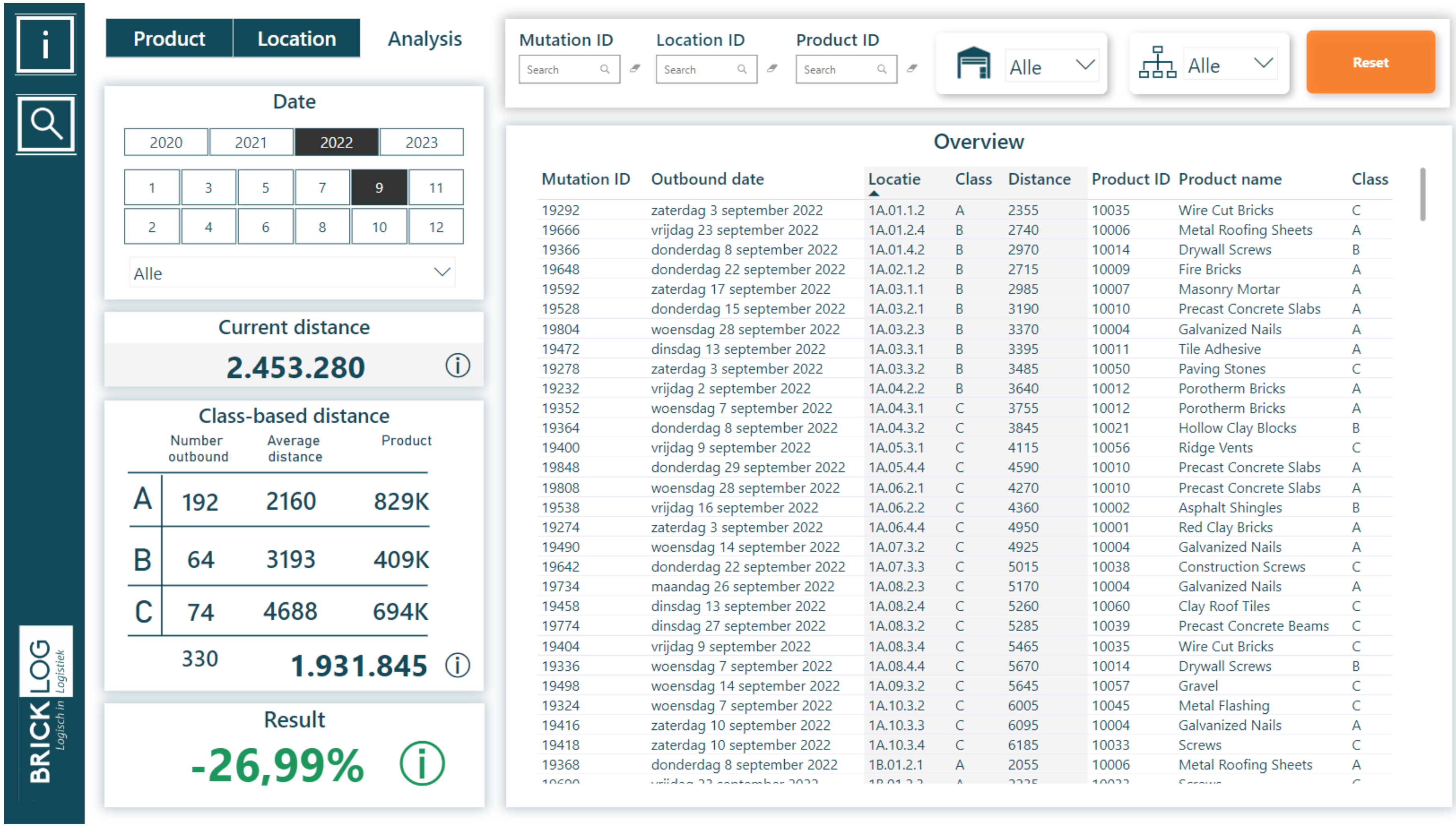
Distance

	Average distance	Relative distance to A
A	2160	
B	3193	1,48
C	4688	2,17

Overview

LocationID	distance...	class...
1A.01.1.1	2265	A
1A.01.1.2	2355	A
1A.01.1.3	2445	A
1A.01.1.4	2535	A
1A.01.2.1	2470	A
1A.01.2.2	2560	A
1A.01.2.3	2650	B
1A.01.2.4	2740	B
1A.01.3.1	2675	B
1A.01.3.2	2765	B
1A.01.3.3	2855	B
1A.01.3.4	2945	B
1A.01.4.1	2880	B
1A.01.4.2	2970	B
1A.01.4.3	3060	B
1A.01.4.4	3150	B
1A.02.1.1	2625	A
1A.02.1.2	2715	B
1A.02.1.3	2805	B
1A.02.1.4	2895	B
1A.02.2.1	2830	B
1A.02.2.2	2920	B
1A.02.2.3	3010	B
1A.02.2.4	3100	B
1A.02.3.1	3035	B
1A.02.3.2	3125	B
1A.02.3.3	3215	B
1A.02.3.4	3305	B
1A.02.4.1	3240	B

Figure B.2: The "locations" report and visualisation overview.



Date

2020 | 2021 | **2022** | 2023

1, 2, 3, 4, 5, 6, 7, 8, **9**, 10, 11, 12

Alle

Current distance

2.453.280

Class-based distance

	Number outbound	Average distance	Product
A	192	2160	829K
B	64	3193	409K
C	74	4688	694K
	330		1.931.845

Result

-26,99%

Overview

Mutation ID	Outbound date	Locatie	Class	Distance	Product ID	Product name	Class
19292	zaterdag 3 september 2022	1A.01.1.2	A	2355	10035	Wire Cut Bricks	C
19666	vrijdag 23 september 2022	1A.01.2.4	B	2740	10006	Metal Roofing Sheets	A
19366	donderdag 8 september 2022	1A.01.4.2	B	2970	10014	Drywall Screws	B
19648	donderdag 22 september 2022	1A.02.1.2	B	2715	10009	Fire Bricks	A
19592	zaterdag 17 september 2022	1A.03.1.1	B	2985	10007	Masonry Mortar	A
19528	donderdag 15 september 2022	1A.03.2.1	B	3190	10010	Precast Concrete Slabs	A
19804	woensdag 28 september 2022	1A.03.2.3	B	3370	10004	Galvanized Nails	A
19472	dinsdag 13 september 2022	1A.03.3.1	B	3395	10011	Tile Adhesive	A
19278	zaterdag 3 september 2022	1A.03.3.2	B	3485	10050	Paving Stones	C
19232	vrijdag 2 september 2022	1A.04.2.2	B	3640	10012	Porotherm Bricks	A
19352	woensdag 7 september 2022	1A.04.3.1	C	3755	10012	Porotherm Bricks	A
19364	donderdag 8 september 2022	1A.04.3.2	C	3845	10021	Hollow Clay Blocks	B
19400	vrijdag 9 september 2022	1A.05.3.1	C	4115	10056	Ridge Vents	C
19848	donderdag 29 september 2022	1A.05.4.4	C	4590	10010	Precast Concrete Slabs	A
19808	woensdag 28 september 2022	1A.06.2.1	C	4270	10010	Precast Concrete Slabs	A
19538	vrijdag 16 september 2022	1A.06.2.2	C	4360	10002	Asphalt Shingles	B
19274	zaterdag 3 september 2022	1A.06.4.4	C	4950	10001	Red Clay Bricks	A
19490	woensdag 14 september 2022	1A.07.3.2	C	4925	10004	Galvanized Nails	A
19642	donderdag 22 september 2022	1A.07.3.3	C	5015	10038	Construction Screws	C
19734	maandag 26 september 2022	1A.08.2.3	C	5170	10004	Galvanized Nails	A
19458	dinsdag 13 september 2022	1A.08.2.4	C	5260	10060	Clay Roof Tiles	C
19774	dinsdag 27 september 2022	1A.08.3.2	C	5285	10039	Precast Concrete Beams	C
19404	vrijdag 9 september 2022	1A.08.3.4	C	5465	10035	Wire Cut Bricks	C
19336	woensdag 7 september 2022	1A.08.4.4	C	5670	10014	Drywall Screws	B
19498	woensdag 14 september 2022	1A.09.3.2	C	5645	10057	Gravel	C
19324	woensdag 7 september 2022	1A.10.3.2	C	6005	10045	Metal Flashing	C
19416	zaterdag 10 september 2022	1A.10.3.3	C	6095	10004	Galvanized Nails	A
19418	zaterdag 10 september 2022	1A.10.3.4	C	6185	10033	Screws	C
19368	donderdag 8 september 2022	1B.01.2.1	A	2055	10006	Metal Roofing Sheets	A

Figure B.3: The "analysis" report and visualisation overview.

C | DAX MEASURES IN POWER BI

```
PercRunningTotal =

VAR CurrentProductFrequency =
    CALCULATE(COUNT(WH_Salesorder_Line[SalesOrderLineID]), ALL(
        Ref_dates))
RETURN

VAR MaxOrderAantal =
        CALCULATE(
            MAXX(
                SUMMARIZE(ALL(WH_Salesorder_Line), WH_Salesorder_Line
                    [ProductID], "Count", COUNT(WH_Salesorder_Line[
                    ProductID])),
                [Count]
            )
        )
RETURN

VAR FilteredTable =
FILTER(
    SUMMARIZE(
        ALL(WH_Salesorder_Line),
        WH_Salesorder_Line[ProductID],
        "Count",
        COUNT(WH_Salesorder_Line[SalesOrderLineID])
        ),
    [Count] >= CurrentProductFrequency && [Count] <= MaxOrderAantal
    )
RETURN

    IF(CurrentProductFrequency <> 0,
        CALCULATE(SUMX(FilteredTable, [Count]) / [TotalOrders]),
        ""
    )

ProductClass =
SWITCH(TRUE(),[PercRunningTotal]<=0.80, "A", [PercRunningTotal]>0.80
   && [PercRunningTotal]<=0.95, "B", [PercRunningTotal] >0.95, "C")
```