Design of a framework for task allocation for the AgvSorter

van Wincoop, L.

Award date:
2019

Department of Mechanical Engineering

Design of a Framework for Task Allocation for the AgvSorter

Graduation Project

CST2019.065

L. van Wincoop 0849855

Supervisors:
Dr.Ir. M.A. Reniers (TU/e)
Dr.Ir. J.A.W.M van Eekelen (Vanderlande)

Eindhoven, August 1, 2019

Preface

For the past seven months I have worked on the graduation project for my masters degree in Manufacturing Systems Engineering within Mechanical Engineering. During my graduation project I have had the opportunity to explore a very interesting field of research and to get to know the company Vanderlande as a diverse and stimulating working environment. For all the support I have gotten throughout these past months I would like to thank some people.

First I want to start off by thanking my thesis supervisor at Vanderlande, Joost van Eekelen. I have enjoyed both the laughs and challenging discussions during our progress meetings. Next, I would also like to thank my supervisor at the TU/e, Michel Reniers for the support and criticism that helped me take my thesis to a higher level.

Also a thank you to all the people at Vanderlande who helped me progress through the analyses on airport systems, parcel and warehousing.

Last but not least I would like to thank my family for the support at home during this project. Especially my boyfriend who managed to keep me calm and collected during this process.

Samenvatting

Vanderlande is de ontwikkeling van een nieuw rooster-gebaseerd automatisch geleid voertuig (AGV) materiaalbehandelingssysteem aan het bestuderen, genaamd de AgvSorter. Dit systeem moet worden toegepast om processen binnen bagageafhandeling, pakket sorteercentra en magazijnen te ondersteunen. Naar deze systemen zal worden gerefereerd als primaire systemen. Een van de regel problemen die moeten worden opgelost tijdens het ontwikkelen van de AgvSorter is het vinden van een toepasselijke taakverdeling strategie die hoge prestaties kan garanderen op het gebied van bijvoorbeeld doorzet in ieder van de primaire systemen. Op dit moment bevat het AgvSorter model onvoldoende input om een taakverdeling strategie te ontwikkelen of om diepgaandere analyse te doen over de toepasbaarheid van het systeem binnen de primaire systemen.

Om een gedetailleerdere analyse te kunnen maken van de impact van het gedrag van een primair systeem op de AgvSorter en om de ontwikkeling van een taakverdeling strategie te faciliteren, wordt een generiek raamwerk ontwikkeld dat de processen en het gedrag kan modelleren van ieder primair systeem. De vereisten waar dit raamwerk aan moet voldoen zijn afgeleid van analyses van bagageafhandeling, pakket sorteercentra, magazijnen en taakverdeling strategieën.

Het resulterende conceptuele ontwerp van het raamwerk bevat een order-job-lading relatie die gebruikt wordt binnen de componenten waar het raamwerk uit bestaat. De componenten zijn ontworpen door gebruik te maken van de domein gedreven ontwerp strategie. Door de componenten te combineren en waardes toe te kennen aan de bijbehorende attributen kan een netwerk aan processen worden gedefiniëerd waardoor de doorstroom van jobs kan worden gemodelleerd met behulp van flowcharts.

Het ontwerp is gevalideerd door gebruik te maken van twee validatie strategieën: analyse van de vereisten en scenario analyse. De analyse van de vereisten is nodig om te laten zien dat de gevonden vereisten direct of indirect zijn meegenomen in het raamwerk of als input voor het raamwerk. De scenario analyse ondersteunt de bevindingen in de analyse van de vereisten door te laten zien dat het raamwerk de verschillende primaire processen en hun functionaliteit kan modelleren. Bij de scenario analyse worden representatieve configuraties van primaire systemen gemodelleerd in de componenten van het raamwerk. De flowcharts laten daarbij de doorstroom van jobs en informatie zien binnen het primaire systeem.

Concluderend laat deze scriptie een gevalideerd, conceptueel ontwerp zien van een generiek raamwerk dat gebruikt kan worden voor analyse van bagageafhandeling systemen, pakket sorteercentra en magazijnen en voor de ontwikkeling van een taakverdeling strategie voor ieder van deze toepassingsgebieden.

Abstract

Vanderlande is studying a new grid-based AGV material handling system, called AgvSorter, that should be employed to support the processes in baggage handling systems, parcel depots and warehouses. These systems are referred to as primary systems. One of the control problems that needs to be solved in the development of the AgvSorter is to find a suitable task allocation strategy that can guarantee high performance, such as high throughput, in each of the primary systems. Currently, the AgvSorter model contains insufficient inputs to develop a task allocation strategy or to conduct further analysis on the suitability of the system for application in each primary system.

In order to conduct more detailed analysis on the impact of primary system behavior on the AgvSorter and to aid in development of a task allocation strategy, a general framework is constructed that can model the processes and behavior in each primary system. The requirements for the framework are derived from analyses of baggage handling systems, parcel depots, warehouses and task allocation strategies.

The final conceptual design of the framework contains an order-job-load relationship that is used within the framework components. The components are designed with the Domain Driven Design strategy. By combining the framework components and assigning values to their attributes, a network of process steps can be defined through which a job flow can be modeled using job flow flowcharts.

The design is validated using two validation strategies: requirements analysis and scenario analysis. The requirements analysis is necessary to show that the derived requirements are included indirectly or directly in the framework or as input for the framework. The scenario analysis supports the requirements analysis by showing that the framework is able to model the different primary systems and their functionality. Representative configurations of primary systems are modeled in framework components. The flowcharts show for each scenario the job and information flow within the primary systems.

To conclude, this thesis presents a validated, conceptual design of a general framework to aid in analysis of baggage handling systems, parcel depots and warehouses and the development of a task allocation strategy for each of these applications.

Definitions

Term	Definition
AGV	Automated guided vehicle.
Blocking	A framework component is blocking when it reaches maximum capacity such that no new **jobs** can be moved to that component.
Die-back	**Blocking** resulting from processes downstream having a lower capacity than upstream processes.
Dispatching	Sending an **AGV** to a destination to fulfill a specific purpose.
Dispatching rule	A decision rule that determines which **material handling task** the **AGV** needs to execute.
Drop-off point	A grid location where an **AGV** can unload one or more loads.
Dwell time	The time a **load** spends in the system without being picked-up by an **AGV**.
Empty travel time	The time it takes for the **AGV** to execute the emptry travel **material handling action**. This does not include the time travelling empty to a parking spot.
Fleet size	The number of **AGVs** in the AgvSorter executing **material handling tasks**. This excludes AGVs parking or charging and includes AGVs being considered for **material handling tasks** by the **task allocator**.
Grid layout	A possible layout of the AgvSorter system.
Guide path	The path on the grid along which an **AGV** is allowed to move in the direction prescribed by the guide path.
Job	A specific combination of **loads** to which the processes in the **primary system** add (logistic) value.
Job flow	The sequence of framework components a **job** visits in the network of processes within a **primary system**.
Load	A load is an item of which there exists a certain quantity.
Load waiting time	The time a load spends in the system without being picked-up by an **AGV** (**dwell time** alias).
Material handling action	An action executed by the **AGV** as part of a **material handling task**. The available actions are: Empty travel to pick-up a load, loading, transporting, unloading.
Material handling system	A system responsible for supporting the **primary system** through physical movements of objects. The actions executed by the material handling system do not add (logistic) value.
Material handling task	A sequenced combination of **material handling actions**.

Term	**Definition**
Multi-load AGV	An **AGV** capable of carrying multiple **loads** at a time.
Operator	Human worker that has a function within the **primary system**.
Order	A part of the demand for the **primary system**.
Pick-up point	A grid location where an **AGV** can load one or more **loads**.
Primary system	A warehouse, a baggage handling system or a parcel depot.
Schedule	Information about when **jobs** are executed by the **primary system**.
Single-load AGV	An **AGV** capable of carrying one **load** at a time with the restriction that the quantity of the **load** is one.
Task	A goal that an **AGV** can be assigned to achieve.
Task allocation	The process of assigning a **task** to one or more **AGVs**.
Task allocator	Supervisory control component responsible for assigning **tasks** to **AGVs**.
Throughput	Number of **material handling tasks** executed by the **AgvSorter** per time unit. This is the total throughput of the system and not the throughput of one **AGV**.

Contents

Chapter 1

Introduction

Vanderlande Industries B.V. is a company focused on developing value-added logistic process automation in warehouses, airports and the parcel market. An essential part of these solutions are material handling systems. Currently, Vanderlande is researching and developing Automated Guided Vehicle (AGV) systems that can be employed as material handling system.

1.1 Automated Guided Vehicle Systems

Automated Guided Vehicles are vehicles that drive without any human interaction or interference. They are often controlled by a control system that determines their tasks, routing and scheduling. They are either free roaming, or they can move through facilities along a set guide path layout and perform material handling actions in a range of application areas [1]. For example, AGVs are deployed in industrial environments such as flexible manufacturing systems (FMS). An FMS is a manufacturing system with automated components such as machine tools, a storage and retrieval system and a material handling system. The material handling system can consist of a fleet of AGVs that transport parts between workstations [2] [3]. For the proper functioning of the FMS, the performance of the material handling system is crucial as it comprises approximately 30% of the production cost [2]. AGVs are also used in automated logistic systems such as container terminals, automated warehouses and cross-dock stations. In warehouses, AGVs are used to transport loads to and from storage areas and workstations. In port container terminals, the use of AGVs has allowed better performance in high-frequency environments and with large ships carrying many loads [4].

Due to the promising developments in industry with regard to AGVs, Vanderlande is developing a new grid based AGV system, internally currently called AgvSorter. The system consists of a grid over which a fleet of AGVs move to pick-up and deliver loads. The aim is to build a system that can be implemented for a variety of applications such as baggage handling at airports, automated warehousing or sorting parcels in the parcel industry.

A part of the control problem of the AGV system is to determine what task an AGV should be executing at a certain moment in time. This process is referred to as task allocation. The task allocation for the AGVs is one of the factors that greatly influences the performance of AGV systems [1] [5]. If inefficient task allocation decisions are made, this can negatively influence the throughput (material handling tasks executed per time unit) of the system and loads may dwell in the system for long periods of time. A material handling task is a task related to the transportation of one or more loads. This definition will be explained in more detail in Section 1.3.

1.2 Current AgvSorter Model

Currently, Vanderlande is in the early stages of the AgvSorter system development. No physical version or prototype of the system has been made yet. A simulation model of the system has been created in Matlab to design the AgvSorter and to visualize and assess its performance. A possible layout of the system with 100 AGVs is shown in Figure 1.2.1.

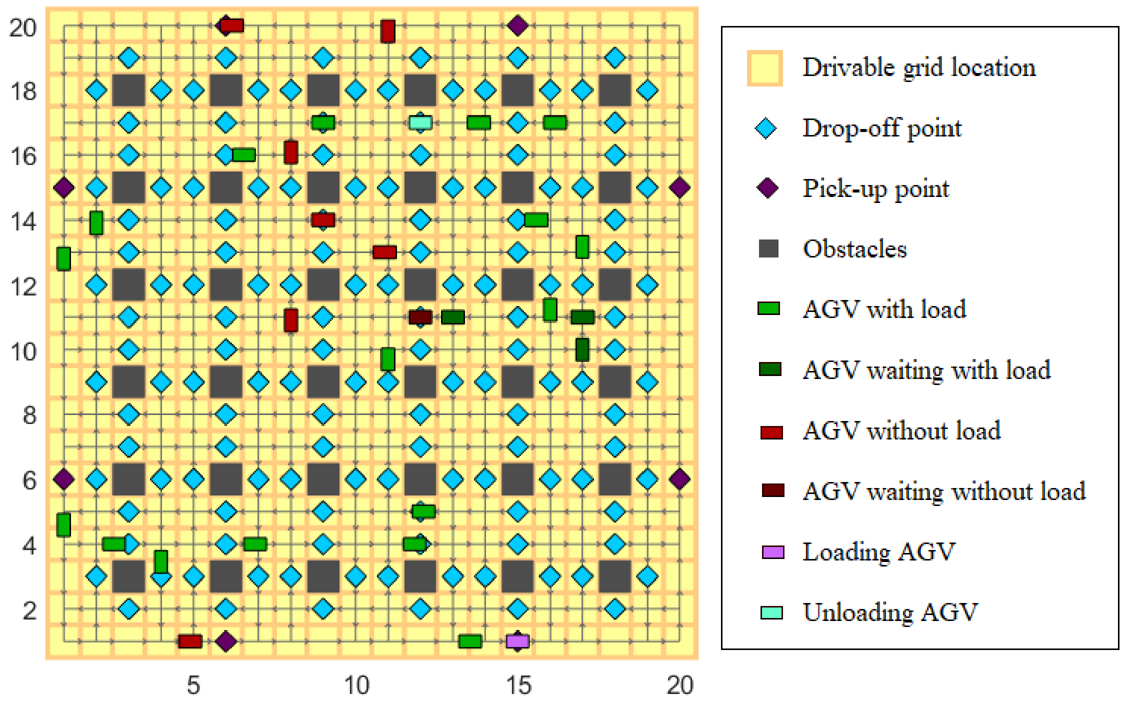


Figure 1.2.1: Square grid layout

In the simulation, it is possible to freely choose the number of AGVs between zero and the maximum number of AGVs as specified in the model. At the pick-up points, loads may be loaded and at the drop-off points loads may be unloaded. Obstacles indicate the chutes or containers or other types of non-reachable areas that the AGVs can encounter. The AGVs themselves are visualized using rectangles that change color based on their current status. Note that a red color does not mean that the AGV has not been assigned a task yet. It can be red while moving toward a pick-up point [6].

The arrows on the arcs of the grid indicate the direction in which the AGVs are allowed to move. Horizontally and vertically each row or column only allows the AGV to move in one direction. Directions can be changed only at intersection points. The grid can therefore be characterized as a uni-directional guide path layout [6].

1.3 Task Allocation Within the Current AgvSorter Model

The control problem for an AGV system consists of path planning, traffic control and task allocation. Path planning is concerned with planning a path along the grid between a starting location and a destination location. Traffic control is responsible for deadlock detection, avoidance and recovery. Lastly, task allocation is concerned with determining what task an AGV should be executing at a certain moment in time. Currently, the path planning and traffic control algorithms are developed and implemented in the AgvSorter model while the task allocation still needs to be developed [7] [8].

The task allocation differs from path planning and traffic control, as the task allocation is a higher level control problem in the sense that it requires knowledge on the processes the AgvSorter supports. Path planning and traffic control are concerned with the state of the grid and do not require any surrounding system knowledge. However, task allocation is highly dependent on information such as arrival rates of loads and queue statuses in order to make accurate decisions on the tasks it should assign to an AGV. A task is defined as a goal the AGV can be assigned to achieve. Within this definition, the following tasks can be distinguished for the AGV:

- Park on a designated parking spot

- Charge the battery
- Execute a material handling task (MHT)

A MHT is defined as a sequenced combination of material handling actions (MHAs). The possible material handling actions are listed and illustrated in Figure 1.3.1.

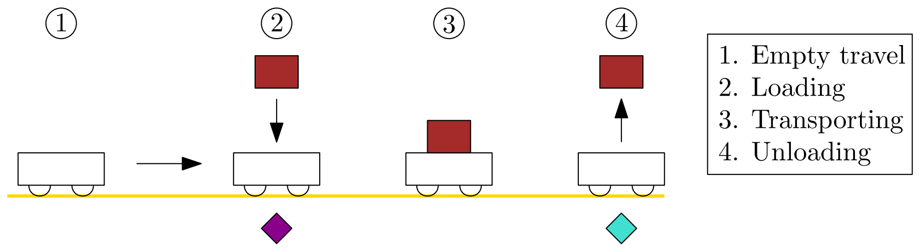


Figure 1.3.1: Material handling actions

By assigning tasks to an AGV, the task allocation strategy can directly or indirectly influence the fleet size. Having an optimal fleet size for the current demand can avoid unnecessary congestion, as well as long load waiting times [9]. Which fleet size is needed to transport all incoming loads is dependent on the processes that the AgvSorter supports as a material handling system. In case of the AgvSorter, the processes it supports are processes within a warehouse, a baggage handling system and a parcel depot. For generalization purposes, these systems are referred to as primary systems. These primary systems can employ the AgvSorter as material handling system.

The model of the AgvSorter system currently contains a task allocation strategy that is based on a continuous inflow of loads into the system. This task allocation strategy was implemented as a temporary solution such that other system behavior and algorithms could be developed. In the implemented task allocation strategy, the AGV selects a load to pick-up at a pick-up point. Whenever a load is reserved by an AGV, a new load is generated at the same pick-up point. This simulates a continuous inflow of loads at each pick-up point.

The process of reserving a load starts with an AGV becoming idle (empty and with no task assigned). It receives a list with pick-up points where it can pick-up loads. The AGV selects the closest pick-up point where the number of load reservations does not exceed a predefined threshold value. The threshold value is in place to ensure that it does not become overcrowded at a pick-up point.

An AGV does not reserve a specific load at a pick-up point. If this were the case then AGVs could hinder each other as illustrated by Figure 1.3.2. If AGV2 were only allowed to pick up the second reserved load (Res2) then it would have to wait for AGV1.

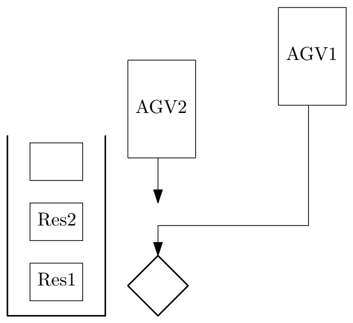


Figure 1.3.2: Illustration of the reservation of loads in the current AgvSorter model

The current assumption that there is a continuous inflow of loads into the system might not be accurate in real-world applications. If the assumption is relaxed, different system behavior may be observed. A continuous inflow of loads means that there are always at least as many loads available as AGVs. However, if for some time no new loads arrive, there may eventually be fewer available loads than AGVs. In literature on AGV dispatching these situations are shown to require a different task allocation approach [10]. Also, if the arrival flow of loads is not continuous, it could mean that some pick-up points become busier than others. This has an impact on the traffic on certain parts of the grid, which can influence performance indicators such as throughput or the waiting time of a load. As this behavior is not included in the AgvSorter model, it cannot be compensated for through the task allocation strategy.

In the current AgvSorter model, there is no way to simulate the processes and queues in the primary system. The impact of introducing aspects such as priority levels, multi-load AGVs and actively managing fleet size cannot be analyzed. The processes of the primary system not only create the demand for the AGV sorter, but can also result in blocking situations which the AgvSorter must be able to manage with its task allocation strategy.

1.4 Problem Statement

Developing a task allocation strategy requires more inputs on system behavior than the current AgvSorter model provides. It is possible to solve this by introducing several inputs in the AgvSorter model such as variable arrival patterns, a FIFO queue and the ability to track load waiting times. By testing task allocation strategies for different input-parameters, a task allocation strategy can be developed experimentally.

However, the three application areas that the AgvSorter should be able to support, exhibit different behavior that affects the choice of task allocations strategy. The processing times of machines and workstations, blocking and outages in these systems are examples of behavior that can have an impact on how task allocation is managed. It is possible that a task allocation strategy that results in high throughputs in a baggage handling system, reduces throughput in a warehouse setting. Next to that, due to physical constraints that differ and the availability of information in the form of sensor and camera data, it might not be possible to find a task allocation strategy that is applicable for all three types of systems. It is therefore desirable to have a more accurate model of the primary system processes.

Another reason why modeling the primary systems is important, is for Vanderlande to be able to show in the early development stage that the AgvSorter is capable of functioning as the material handling system within the customers systems. Several AgvSorter layouts have been developed for systems that other Vanderlande solutions are currently implemented in. This can show how the AGVs would drive through the facility, but not where blockage would occur or the state of the grid when the system becomes operational. By including primary system characteristics, it would be possible to simulate the resulting behavior of the primary system on these layouts.

Vanderlande has models and control systems for its warehousing solutions, baggage handling systems and parcel sorting systems. The choice can be made to adapt these models and simulations to suit the inputs and outputs of the AgvSorter model. However, it is time consuming to couple these models to the current AgvSorter Matlab model. Also, each time a new system needs to be analyzed the entire model would need to be rebuilt to fit the new configuration. Therefore, the main research question that arises is:

In what way can the primary systems be captured in one model, such that the model can be used in different simulations and analyses?

In order to answer this main research question, several research objectives need to be completed:

- Objective 1: Identify key characteristics of the baggage handling, parcel and warehousing primary systems.
- Objective 2: Construct a general framework that takes into account the key characteristics identified in the first objective.
- Objective 3: Validate the conceptual framework design.

1.5 Report Structure

Objective 1 is completed in Chapter 2. In this chapter the primary systems are analysed, from which the requirements are derived that need to be included in the framework. The conceptual design of this framework is presented in Chapter 3, which completes the second objective. Chapter 4 contains the validation of the framework through the application of two validation methods, which fulfills the third objective. Lastly, Chapter 5 contains the conclusion of this research and recommendations for future work.

Chapter 2

Identification of Primary System Requirements

In order to design the framework, the characteristics of the primary systems that the AgvSorter should be able to support must be analyzed. From this analysis, a list of requirements is derived for the design of the framework. As Vanderlande specializes in developing high-tech logistic systems for warehousing, parcel and baggage handling, the majority of information in this chapter is obtained from conversations with domain experts. This information is supported by Vanderlande documentation and scientific literature. In the first section, each type of primary system is introduced. The following sections describe different characteristics within the primary systems. Each section is concluded with a list of requirements that can be derived from the analysis in that section. Lastly, a literature study is done that presents several task allocation strategies and the information that is needed to execute them.

2.1 Introduction of the Primary Systems

In this section, an overview is given of the operations that are carried out within a warehouse, a parcel depot and a baggage handling system.

2.1.1 Warehousing

A warehouse is a location that replenishes and stores goods and fulfills orders. The type of goods that are stored is dependent on the market segment that the warehouse operates in. Possible market segments are fashion retail, general merchandise, food and spare parts. Warehouses can deliver either directly to the consumer or to a retailer [11].

Within a warehouse, goods either arrive to be stored or are removed from storage to be shipped [11]. The first is the inbound process and the latter is the outbound process. An example of a warehouse and its processes is shown in Figure 2.1.1.

The inbound process starts at the arriving trucks in the left of the figure. Pallets of items are unloaded (receiving) and moved to a decanting area where the items are placed in carriers that can fit in the warehouse storage.

The outbound process starts with a customer order. This customer order results in a picking process, after which the picked goods are delivered to packing and sorting workstations. After the items have been packed, they must be delivered to the shipping containers which can then be loaded into trucks. The process of sorting the packages to their shipping containers is called marshalling. Finally, the process ends with the orders being shipped to the customer.

Within a warehouse, the AgvSorter could be employed to move carriers with items between processes. It could also be used for order picking and marshalling.

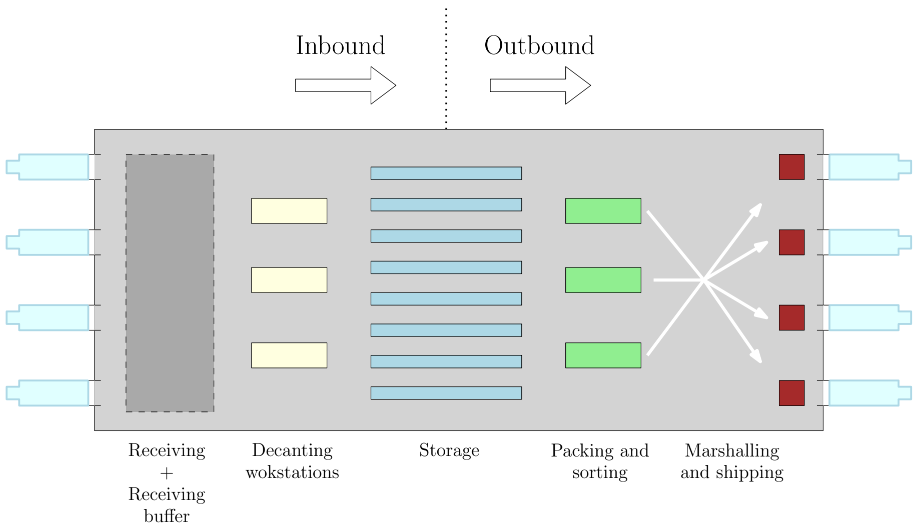


Figure 2.1.1: Schematic of a warehouse and its processes

2.1.2 Parcel

The parcel process starts with a customer shipping a parcel, which is picked up by a courier. At that point, the parcel enters a parcel network. In this network, the parcel is transported to one or more depots before reaching the customer at the receiving end. A parcel depot is a location where parcels arrive and are sorted to other vehicles and load carriers for transportation.

The processes within a parcel network can also be subdivided into an outbound process and an inbound process. The outbound process is the process of picking up parcels at the customer (post offices) and delivering them to the depot. For the inbound process, parcels are sorted at the depot to the delivery vans that deliver the parcels to the customer.

In a parcel depot, the AgvSorter can be implemented as the material handling system that sorts the parcels.

2.1.3 Baggage Handling

In airports there is a desire to automate the baggage handling systems. Losing baggage results in higher operating costs and lower passenger satisfaction. By eliminating the human errors through automation and more advanced baggage tracking systems, the number of lost bags should reduce. With this in mind, an AGV system can support the baggage handling process by performing the material handling actions required to transport and sort the bags [12].

Bags can arrive at a baggage handling system either as a transfer bag or as checked-in luggage that passengers drop-off upon arrival at the airport. A baggage handling system consists of several process steps. Figure 2.1.2 gives an overview of the different processes and the arrows indicate the order in which the processes can be visited by the bag. These are the basic process steps that are implemented in most airports. Between different airport sizes the order in which the processes are set up can vary.

Each bag has a label with a tag that contains all the necessary information about that bag. Usually, the bag information is obtained by an automatic scan while the bag is traveling from check-in or transfer. However, this automatic scan may fail. Then the bag information has to be entered at a manual-encoding station (MES) [13].

Every bag that arrives at the airport has to go through security at a screening station and/or machine. Screening consists of several levels. If a bag does not pass the first level, it enters the second level where other security checks are done and so on. All arriving bags need to go through screening [13].

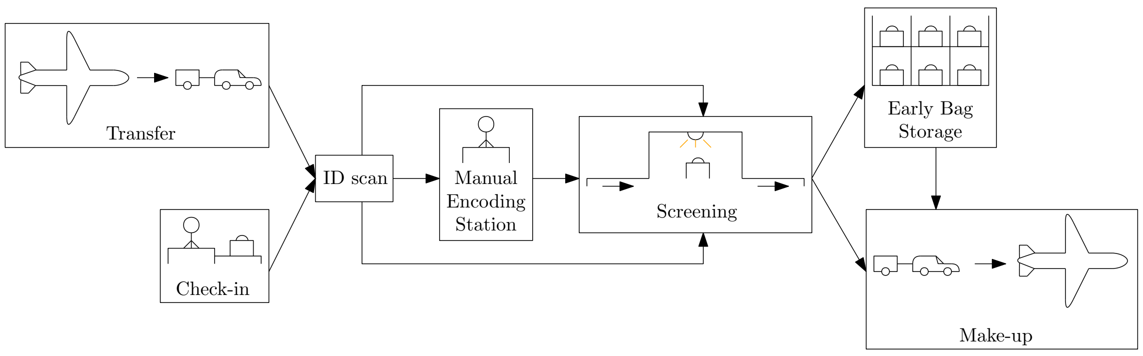


Figure 2.1.2: Schematic overview of the processes within an airport

After screening, the bag either goes to a buffer for early bags called Early Bag Storage (EBS) or make-up. The make-up process consists of final sort, loading and confirmation. When make-up opens for a flight, the bags bound for that flight may be sorted and loaded. Bags that were stored in the EBS may be retrieved from EBS.

In the baggage handling system, there is much material handling required and many places where the AgvSorter may be implemented. The AGV system could carry bags between each described process step, and it could also be employed only for the final sort in the make-up process.

2.2 Arrival Patterns

Each type of primary system has a distinct arrival pattern. Even between warehouse configurations or different airports, the arrival patterns may not be similar. Each configuration therefore has a unique influence on the behavior of the AGVs on the grid. It is possible to omit the complexity introduced by modeling accurate arrival patterns and opt to only design the system for maximum capacity use. However, real-life situations are dynamic and do not always require maximum capacity. When situations occur where less than maximum capacity is required, performance in terms of for example throughput can drop as the fleet size and tasks are not managed well for such a situation.

Within warehousing, arrivals can be frequent or infrequent and often highly unpredictable. Order arrival rates can vary each hour. An illustration of an arrival pattern for a warehouse fulfilling online customer orders is shown in Figure 2.2.1. The cut-off time is the time where the incoming orders can no longer be shipped the next day.

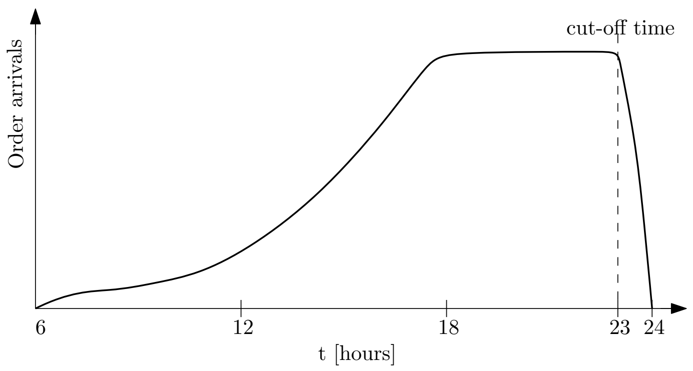


Figure 2.2.1: Warehousing order arrival profile

For a flight at an airport, the arrival pattern for bags can be shown in a dwell profile. An illustration of a dwell profile is shown in Figure 2.2.2. Dwell profiles such as these may vary in shape depending on the type of airport. The dwell profile may even differ between flights if a short-distance flight requires few bags to be handled. Some airports handle more transfer bags than others, which also creates a different dwell profile. As dwell profiles can differ for each flight within an airport, this shows that there is a high variability in arrival patterns within an airport.

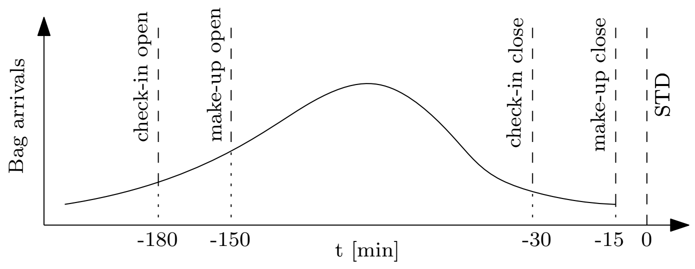
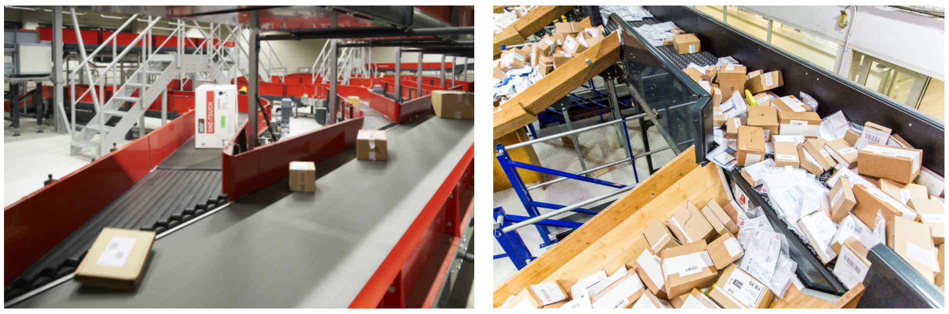


Figure 2.2.2: Bag arrival dwell profile

Parcels can arrive to the sorting either in bulk flow or in single-item flow. A single item flow means that the parcels traverse the conveyors one-by-one while in bulk flow parcels are heaped together. Figure 2.2.3 shows the two types of flow.



(a) Single item flow

(b) Bulk flow

Figure 2.2.3: Images of single item flow and bulk flow (source: Vanderlande)

The arrival of parcels happens in batches when trucks, vans or airplanes arrive at the depot delivering a number of parcels. This means that the arrival rate of parcels to the AgvSorter both increases and decreases over time depending on the arrival schedule of the vehicles delivering the parcels.

As can be seen, each of the primary system types have distinct arrival patterns. The continuous inflow of jobs currently implemented in the AgvSorter system does not hold for the primary systems it needs to support. It should be possible to use each type of arrival pattern seen in the primary systems as input for the framework. From the analysis of the arrival patterns, several requirements are identified that must be incorporated in the framework or used as input for the framework. In Table 2.2.1, the requirements are listed. Each requirement is given an ID. The capitalized letters in the ID refer to the type of characteristic they originate from and the last number is the requirement number.

Table 2.2.1: Requirements for incorporating arrival patterns in the framework

Requirement ID	**Requirement statement**
Req.AP.1	It shall be possible to use any arrival pattern encountered in the primary systems as input for the framework.
Req.AP.2	The framework shall be able to model bulk flows as well as single-item flows.

2.3 Primary System Processes

A primary system consists of multiple operations that add (logistic) value to either stock in a warehouse, bags or parcels. After loads have been processed, they may need to be transported by the AgvSorter system. Process times, random outages and the capacity of these processes influence the business of specific areas of the grid where these processes are located. They may also cause blocking, which the AgvSorter should be able to cope with.

There are different types of processes that can be distinguished:

- Automated processes
- Workstations

Conveyors are a type of automated process common in baggage handling systems, and in parcel depots. Conveyors often have deterministic travel times that depend on where the load is headed. Other automated processes are a depalletizing machine in a warehouse or a screening machine in a baggage handling system. Primary systems can also contain manual workstations such as a manual encoding station, a decanting station or a packing station.

Any process in a primary system may introduce variability. This can be in the form of random or planned outages, operators being called away or non-deterministic processing times [14]. Also, if a sequence of processes is present, this may cause die-back. Die-back means that blockages occur and that loaded AGVs are left waiting to unload at blocked processes. It happens when the downstream capacity is lower than the capacity of the upstream processes. In order to cope with capacity constraints and prevent die-back, primary system processes may have ingoing and outgoing buffers. These buffers can simply be empty spaces on the floor, conveyors or automated storage systems.

One or more AgvSorters may be in place to transport loads between some or all processes in a primary system. As the grid tiles are dependent on the size of the AGVs it may not be beneficial for performance if very small and very large AGVs are combined on a grid. In that case it is better to implement multiple AgvSorters with independent controls.

In Table 2.3.1 the requirements are listed that are elicited based on the analysis in this section.

Table 2.3.1: Requirements for incorporating different process types in the framework

Requirement ID	**Requirement statement**
Req.PSP.1	The framework shall be able to incorporate variable and deterministic processing times.
Req.PSP.2	The framework shall be able to specify the capacities for each process.
Req.PSP.3	The framework shall incorporate the possibility to model variability.
Req.PSP.4	The framework shall incorporate buffers.
Req.PSP.5	It shall be possible to model a sequence of processes within the framework.
Req.PSP.6	The framework shall be able to model a primary system that has multiple AgvSorters implemented.

2.4 Multi-load AGVs

A distinction can be made between single-load AGVs and multi-load AGVs. A single-load AGV is an AGV capable of only carrying one load due to mechanical constraints. A multi-load AGV can carry one or more loads at a time. If an AgvSorter is considered with multi-load AGVs, this may have an impact on the task allocation strategy [15].

Within warehousing, an AGV can be employed for order picking. This means picking multiple items from the storage racks and carrying them in a carrier such as a tote or crate. Order picking can be done using different strategies. An order picking strategy is batch picking where an AGV picks items belonging to multiple orders based on where the items are located in the warehouse.

While order picking, the sequence in which the items are picked can be of importance. Sequencing may be necessary to prevent fragile items to be placed under large, heavy items in a carrier or to ensure that cold items are picked together. Sequencing can also be the result of a business rule.

A distinction is made between strict sequencing and relaxed sequencing. Relaxed sequencing introduces sequence groups. The items belonging to the same sequence group may be picked in a random order, but all items belonging to one sequence group should be picked before proceeding to the next sequence group. Strict sequencing can be seen as a special case of relaxed sequencing where there is only one item per sequence group.

Within parcel and baggage handling, parcels and bags are often carried on top of the AGV instead of in a carrier. An AGV may be fitted with a conveyor or tilted top that aids the loading and unloading of parcels and bags. Also within these applications, multi-load AGVs may be applied. However, especially in parcel, the parcel dimensions are a deciding factor whether multi-load is possible as parcels can vary heavily in size. It depends on how the parcel network is set-up whether a depot sorts multiple different size categories or just one.

A way to apply multi-load for a parcel setting is to load the AGV with a parcel upon arrival at a pick-up point. If the parcel is small, another parcel may be placed on top of the AGV if that parcel is small too. If the first or second parcel are larger parcels, the AGV may proceed to carry out the material handling actions with one parcel. The same system could be applied for baggage handling, however the bag sizes are usually more constant in a baggage handling setting.

If an AGV has two parcels loaded, both parcels may need to be sorted to two different locations. This means that the AGV has to visit two drop-off points to deliver both loads. It can also be the case that an AGV that is only carrying one small load is instructed to pick up another small load at a different pick-up point.

It can be concluded from the different multi-load settings that an AGV can be instructed to visit n pick-up points and m drop-off points where n and m can be either one or more.

Based on the analysis on multi-load situations within each type of primary system, several requirements can be defined. These requirements are listed in Table 2.4.1.

Table 2.4.1: Requirements for incorporating multi-load AGVs in the framework

Requirement ID	**Requirement statement**
Req.ML.1	The framework shall support the use of different order picking strategies.
Req.ML.2	The framework shall incorporate many-to-many, many-to-one, one-to-many and one-to-one material handling tasks.
Req.ML.3	Strict and relaxed sequencing must be part of a material handling task.
Req.ML.4	Dimensions of loads shall be a factor that influences the material handling tasks.
Req.ML.5	The capacity needed to execute a material handling task shall not exceed the capacity constraints of an AGV.

2.5 Information Flow

Within each type of primary system, there are different sources of information available to base control decisions on, such as cameras, sensors, inventory management software and bag tracking systems. What information is available may differ per configuration of a primary system and it influences the accuracy and choice of task allocation strategies.

A type of information that is useful for a task allocation strategy, is priority information. Especially in baggage handling systems this information is often used. The priority level of a bag depends on the time that is left until make-up closes for the flight. Figure 2.5.1 shows the three priority levels in the dwell profile. In warehousing there may also be priority levels in place in case an order is an emergency shipment.

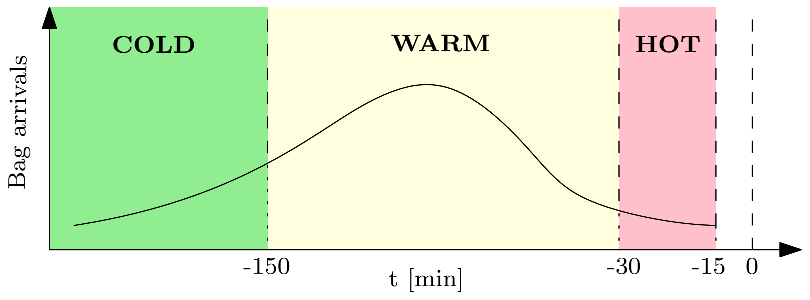


Figure 2.5.1: Priority levels in a baggage handling system

Not only is the type of information available relevant for the task allocation strategy, also the timing of when the information is available is important. Within baggage handling, reassignment may take place when the bag is on its way to a process if for example the results of a screening process are inconclusive. There is also a difference between timing in push processes and pull processes.

A pull process happens in the outbound process of a warehouse. When an AGV is order picking, the items are essentially pulled out of the warehouse stock by an external demand. It may be beneficial to decide as late as possible on which packing station to deliver the picked items to, depending on which packing station has free capacity. Push typically happens in the inbound process of a warehouse where goods arrive at a warehouse to be stored. Parcel and baggage handling systems can also be seen as push where parcels and bags are pushed into the system processes.

Within each configuration of primary systems, the type of information and the amount of available information differs. When choosing a task allocation strategy, it is important to only rely on information that can actually be obtained from the primary system configuration that is used. Therefore, the requirements listed in Table 2.5.1 should be taken into account.

Table 2.5.1: Requirements for information flow in the framework

Requirement ID	**Requirement statement**
Req.IF.1	The framework shall take into account priority levels.
Req.IF.2	The framework shall be able to adjust the timing of sending information based on the primary system configuration.
Req.IF.3	The information delivered by the framework shall correspond to the information flow as present in the primary system that is represented by the framework.

2.6 Task Allocation Strategies

In order to find a suitable task allocation strategy, the framework should facilitate the implementation of a range of strategies found in literature. In this section, a literature study of task allocation strategies is shown. The aim of this section is to identify inputs that are necessary to apply a range of task allocation strategies.

2.6.1 Introduction to Task Allocation

As described briefly in Section 1.3, task allocation consists of determining what task an AGV should perform. The tasks that are mentioned are:

- Execute a MHT
- Park
- Charge

When an AGV becomes idle, the task allocator should first decide which of the three tasks an AGV should perform. The task allocator is defined as the supervisory control component responsible for assigning tasks to AGVs. In case the task allocator decides that an AGV should execute a MHT, it must be decided which MHT this should be. A MHT is defined as a sequenced combination of MHAs. Note that the sequence in which the MHAs are executed as part of the MHT is not necessarily strict as the order in which multiple pick-up points and/or multiple drop-off points are visited is not prescribed. However, it must hold that all loads need to be picked up at pick-up points prior to conducting the unloading actions at drop-off points.

In literature, many strategies are defined over the years that address the assignment of MHTs to AGVs. Some of these strategies are discussed in this section.

2.6.2 Dwell Points

When an AGV is assigned a parking task, a parking location must be chosen. A parking location for an AGV can be referred to as a dwell point. Finding dwell points where an AGV can be idle, can be treated as an optimization problem that aims to minimize performance indicators such as the response time of a vehicle. The response time of the AGV is defined as the time it takes to reach the assigned load from the dwell point where it is parked [16] [17].

2.6.3 Fleet Size

Letting AGVs park or charge both reduces the fleet size. Managing the fleet size can be done actively or passively. The fleet size has impact on the energy consumption of the system as well as the efficiency. Blockage and congestion can occur if too many AGVs are active at the same time. Similarly, when the fleet size is too small, loads dwell in the system for long periods of time and in case of an FMS it can result in the underutilization of machines [18]. Early work on analytically determining the optimal AGV fleet size is done by Rajotia et al. [9]. They base the decision of the optimal fleet size on three time factors: the load handling time (loading, transporting and unloading), the empty travel time and the waiting and blocking time.

2.6.4 Dispatching

AGV dispatching is defined as sending an AGV to a destination to fulfill a specific purpose. An AGV dispatching rule is a decision rule that determines which purpose the AGV needs to fulfill. Egbelu and Tanchoco [10] laid important groundwork by making the distinction between work-station initiated dispatching rules and vehicle initiated dispatching rules. The former attempts to select a vehicle from a range of idle vehicles that can transport an arriving load. It is initiated when there are less loads to be transported than there are available AGVs. However, a vehicle initiated dispatching rule is initiated when there are more loads in the system than AGVs.

In early literature, many single-attribute rules are defined for AGV dispatching. These take into account one attribute that influences the performance of the AGV system such as empty travel time or the time a load has been in the system or the distance to a certain load [1] [10] [19] [20].

While single-attribute rules are simple and easy to implement, it is shown that multi-attribute rules can outperform the single-attribute rules. These rules consist of attributes such as number of loads in a queue, (empty) travel time and load waiting time. These attributes are multiplied by weights and combined in a single function. The weights determine how much each attribute contributes to the dispatching decision [1]. There are ways introduced in literature to dynamically tune the weights using fuzzy numbers, neural networks and genetic algorithms [21] [22] [23] [24].

2.6.5 Assignment Problem

Another approach to the task allocation for AGV systems is by solving the Assignment Problem (AP). The assignment problem is one of the fundamental combinatorial optimization problems. The AP considers the case where n material handling tasks need to be divided over n agents (or AGVs in this case) and each agent may only receive one material handling task [25].

Thakre et al. [26] apply the AP on a case where four candidates need to be divided over four jobs. The resulting AP is solved by using the Hungarian method, the matrix one's assignment (MOA) method [27] and a new method that is introduced by the authors. The Hungarian method is the first polynomial-time method developed to solve the AP [25].

Wu et al. [28] considered the General Assignment Problem (GAP) with min-max regret criterion under interval costs (MMR-GAP). The GAP is a version of the AP where the number of MHTs is not equal to the number of AGVs. Each MHT may only be assigned to one AGV and the AGVs are subject to a resource constraint. By executing a MHT, the AGV will deplete some of its resource. The MMR-GAP considers the case where costs can take on values within an interval and the aim is to minimize the maximum regret for choosing a certain assignment.

In both the AP and the GAP each agent is only assigned one task. This constraint can also be relaxed as done by Chauvet et al. [29]. In their research the number of tasks does not have to be equal to the number of AGVs. However, the number of tasks is smaller than the number of agents that the tasks are divided over.

Each of the AP and GAP variants could be applied for dividing material handling tasks over AGVs. Attributes such as load waiting time or travel distance can be used to divide the material handling tasks over AGVs. If there are fewer loads than AGVs in the system, one could take the approach suggested by Egbelu and Tanchoco [10] and divide loads over AGVs. A downside of using the GAP for MHT assignment is that the problem is NP-hard. The AP is less computationally expensive, but can also become complex when multiple attributes are used to determine which AGV is most suitable for which task.

2.6.6 Task Allocation Input Requirements

In the literature study presented in this section, two types of task allocation strategies are presented: strategies involving the AP and dispatching strategies. Additionally, sequencing and scheduling and bidding based strategies can also be considered as task allocation strategies. They require the same types of inputs and dispatching strategies. However, based on knowledge of the primary systems and the AgvSorter, it is found that these strategies are not suitable and are therefore not included in detail in this section. Sequencing and scheduling rely on thorough knowledge of when certain loads are planned to arrive. As each of the primary processes are unpredictable in terms of load arrivals, this strategy cannot be applied. Bidding based task allocation strategies often view AGVs as rational agents that bid for material handling tasks. This means that the control decisions become decentralized. As the AgvSorter can contain hundreds of AGVs, a decentralized control strategy is not desirable [30] [31] [32] [33] [34].

For each studied task allocation strategy, it is found that there are several recurring inputs that are needed to implement each task allocation strategy. The requirements identified are listed in Table 2.6.1.

Table 2.6.1: Requirements for task allocation strategies

Requirement ID	**Requirement statement**
Req.TA.1	The Task Allocator shall be able to obtain information on the distance between grid points.
Req.TA.2	The Task Allocator shall be able to obtain information on queue statuses.
Req.TA.3	The Task Allocator shall be able to obtain information on travel times between an AGV and a target location.
Req.TA.4	The Task Allocator shall be able to obtain information on load waiting time.
Req.TA.5	The Task Allocator shall be able to obtain information on load handling time.

2.7 Conclusion

In this section the characteristics of baggage handling systems, parcel depots and warehouses are analyzed. This has resulted in the derivation of a list of requirements that need to be incorporated indirectly or directly in the framework or used as input for the framework. In Chapter 3, the conceptual design of the framework is presented.

Chapter 3

Framework Design

In this chapter, the framework design is shown. The framework should incorporate the requirements that are derived in Chapter 2 in a general way. It should be possible to include complex primary system algorithms as well as simple deterministic models. The framework should therefore specify what parameters and inputs should be defined while the user defines what the values of these parameters should be. The user in this context is a person who wants to use the framework for analysis.

This chapter starts with explaining the distinction between orders, jobs and loads that is used throughout the framework. In the next section, the inputs to the framework in terms of order arrivals are introduced. Subsequently the components of the framework and their properties are explained. This is followed by a flowchart that shows when and how certain information needs to be sent and how jobs move through the framework.

3.1 Orders, Jobs and Loads

First, it is necessary to introduce orders, jobs and loads. Each of these terms are related to each other as shown in the entity relationship diagram (ERD) in Figure 3.1.1. The ERDs shown in this chapter show the cardinality of the relationships between entities based on Chens notation [35].

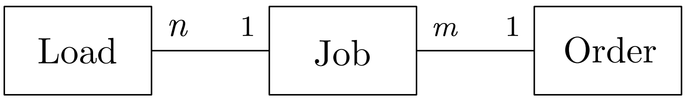


Figure 3.1.1: Order, job, load ERD

An order is a combination of jobs, which is a combination of loads. Loads are transported by the AgvSorter system as part of a MHT. A job corresponds to a specific (combination of) load(s) that is processed by the primary system processes. An order represents the external demand arriving at a primary system.

The load attributes are shown in the ERD in Figure 3.1.2. A load is an item of which there exists a certain quantity. The dimensions specify information on the length, width, height and weight of the load. The sequence group the load belongs to determines the order in which the loads need to be picked up as part of an MHT.

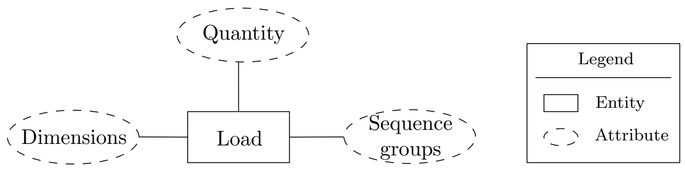


Figure 3.1.2: Load attribute ERD

The processes within the primary system process jobs and add (logistic) value to the job. The job attributes are shown in Figure 3.1.3. The priority level of a job may be taken into account by the task allocation strategy when assigning MHTs to AGVs and by the primary processes when processing the job. The waiting time shows how long the job spends waiting in a queue.

It should hold that the capacity required to transport the loads related to a job, never exceeds the capacity constraints of an AGV. An AGV may execute multiple MHTs at once if the total capacity needed does not violate any capacity constraints. Each MHT may only be carried out by one AGV.

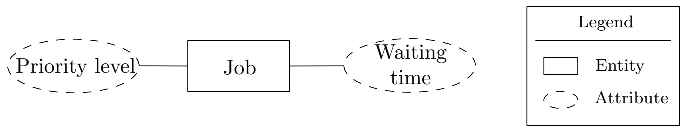


Figure 3.1.3: Job attribute ERD

An order is demand for the primary system. Based on the logistic concepts and constraints of the primary system, an order is split into a combination of jobs.

3.2 Framework Components

For the design of the framework components, the Domain Driven Design (DDD) strategy is used. This strategy allows complexity of the framework to be reduced by dividing the model components into subdomains with their own bounded context [36]. Vanderlande currently applies the DDD strategy in their warehousing platform.

3.2.1 Core Domain and Subdomains

For design of the framework, the primary system is defined as the core domain. Within the core domain, three subdomains are distinguished:

- The Enterprise Domain (ED)
- The Process Domain (PD)
- The Material Handling Domain (MHD)

Figure 3.2.1 shows the subdomains and a description of their contexts. Each of the subdomains has its own bounded context, which restricts the use of information and terminology for that domain. For the ED, the processes and logistics within the primary system are a black box. These logistics and processes are incorporated in the PD. The PD receives arrived orders from the ED, processes the resulting jobs and delivers finished orders as output to the ED. In short, the ED specifies what the primary system should do and the PD specifies how this is done.

For the PD the material handling system in the MHD is black box. The PD may take into account physical constraints from the MHD, without knowing what material handling system is in place. The MHD is not associated with jobs but with MHTs that result from jobs. It receives information from the PD that is necessary for task allocation without knowing about the logistic concepts in the primary system.

3.2.2 Enterprise Domain

The ED is concerned with what the primary system should do in terms of orders, which is captured in the component Order arrival processes, shown in Figure 3.2.2. The processes and logistics within the primary system are black box for the ED. Therefore, the ED forms the input for the framework.

In order to incorporate the arrival processes of the primary system, a load file is constructed.

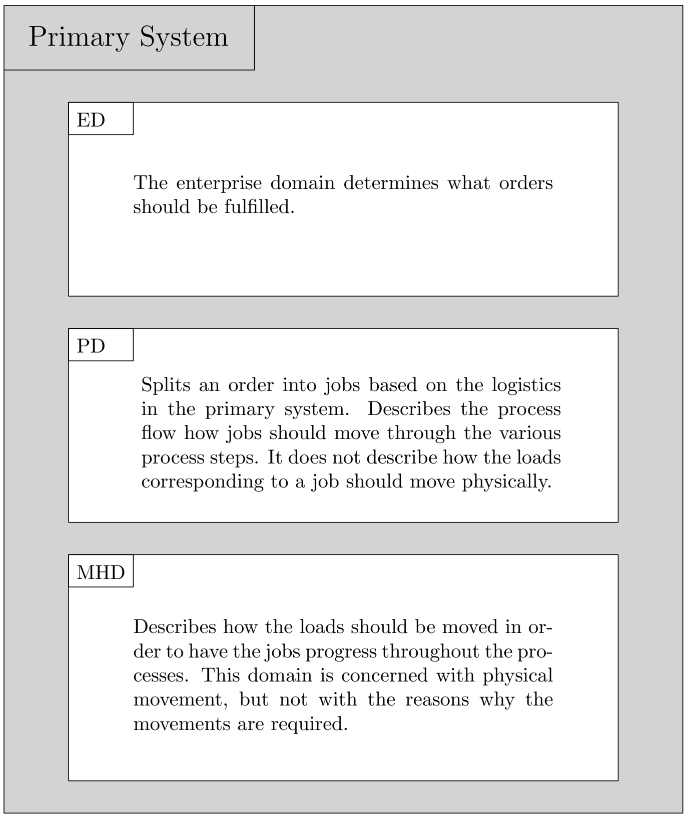

Primary System
ED
The enterprise domain determines what orders should be fulfilled.
PD
Splits an order into jobs based on the logistics in the primary system. Describes the process flow how jobs should move through the various process steps. It does not describe how the loads corresponding to a job should move physically.
MHD
Describes how the loads should be moved in order to have the jobs progress throughout the processes. This domain is concerned with physical movement, but not with the reasons why the movements are required.

Figure 3.2.1: Core domain and subdomains for the framework design

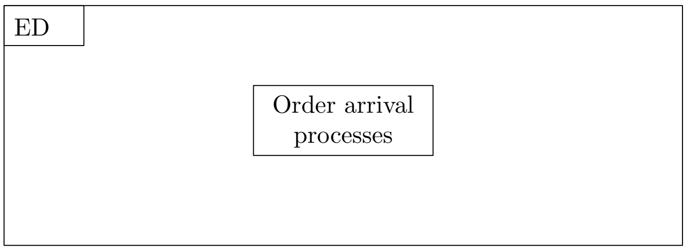


Figure 3.2.2: Overview of enterprise domain framework components

The load file consists of two parts. The first part is the arrival file, which lists the IDs of all arriving orders and the time at which they arrive. The attributes of the orders, jobs and loads are listed in the order database. For additional information, Appendix A gives an example of the layout of a load file.

3.2.3 Process Domain

The components of the PD are shown in Figure 3.2.3. Their relations and functionality are explained in this subsection.

The PD is concerned with the logistics and operations of the primary system. Once orders arrive, they are split into jobs by the component Order splitting logic. The algorithm behind Order splitting logic is user defined and may be based on logistics within the primary system and physical constraints imposed by the MHD. The jobs resulting from the split by Order splitting logic are processed by primary system processes. A combination of input-queues, a process and output-queues form an abstraction of a primary system process. The ERD of these components including their attributes is shown in Figure 3.2.4. Here $n = 1, 2, ...$ and $m = 0, 1, ...$ After leaving a process, a job is either done or moved to an output-queue.

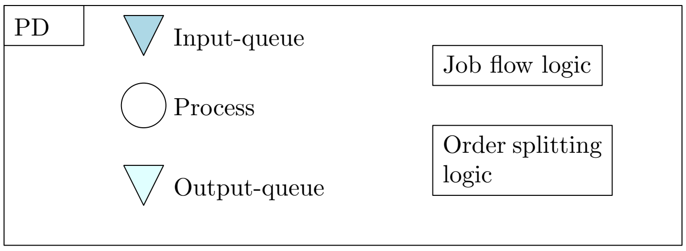


Figure 3.2.3: Overview of process domain framework components

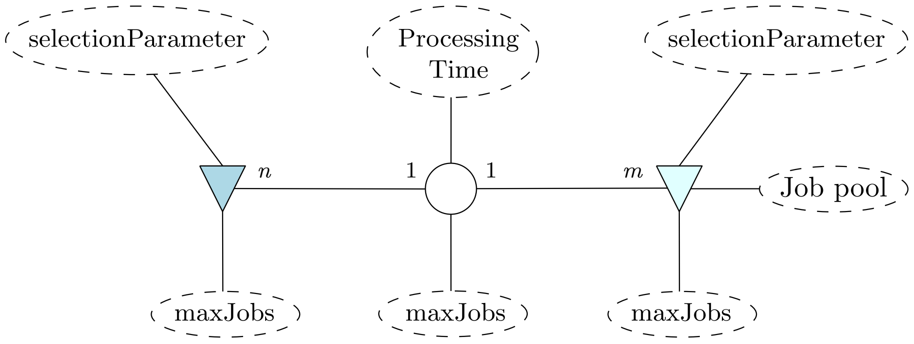


Figure 3.2.4: ERD for the input-queue, process and output-queue

Capacity constraints are indicated by the maxJobs attribute, which is an integer value that determines how many jobs may be in the queue or process at once. By taking into account finite capacity components, the impact of blocking on the AgvSorter can be analyzed. The process has a processing time parameter which determines how long a job will remain in the process before it is either done, or should move to an output-queue.

The selectionParameter for the queue determines which jobs can be moved from the queue if multiple jobs are located in the queue at once. The value of the selectionParameter is dependent on the process and the configuration of the queue in the primary system. Figure 3.2.5 shows five jobs located in two input-queues (indicated by the dashed rectangle). The first queue has selectionParameter 2 which means that Job1 and Job2 may be removed from the queue to enter the process. The second queue represents a FIFO queue, with selectionParameter equal to one, where only Job1 may be moved to the process. It is important to implement which jobs can be moved from the queue and which cannot as this impacts attributes such as priority levels that change over time and performance measures such as the time a job takes to go through the primary system processes.

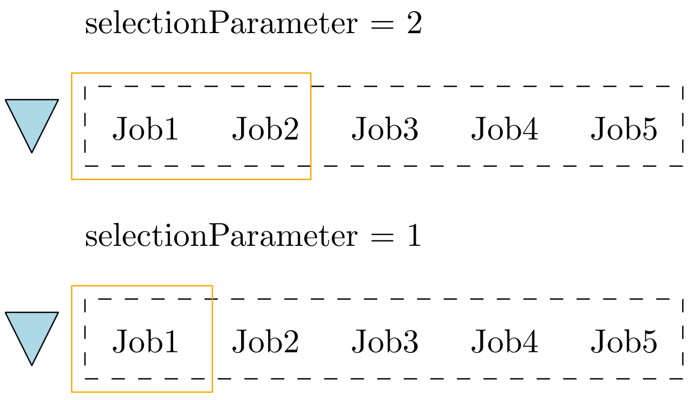


Figure 3.2.5: selectionParameter illustration

Jobs in an output-queue that are movable according to the selectionParameter and are not headed for an input-queue that is currently full (blocked) are included in a job pool. A job pool is either a list with jobs that can be moved directly to the next input-queue, or a list with jobs that result in MHTs that need to be executed by the AgvSorter in order to move the job to the next input-queue. The MHTs are communicated to the MHD.

A primary system may contain multiple processes that form a network as shown in Figure 3.2.6.

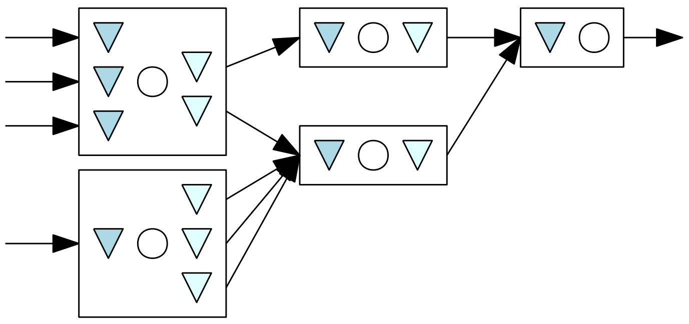


Figure 3.2.6: A network of processes

The network consists of a number of processes with their corresponding input-queues and output-queues. The arrows indicate the routes that a job may take through the various process steps depending on where the job is located.

The route a job should take is defined by the component Job flow logic. This component contains the decision rules behind choices that must be made in order to progress the jobs through the primary system processes. Job flow logic contains the logic that determines which job should be considered first if multiple jobs have finished processing simultaneously, or if multiple jobs can be selected from the input-queues. It also contains information on whether jobs should merge or split, which is introduced in Section 3.3.

3.2.4 Material Handling Domain

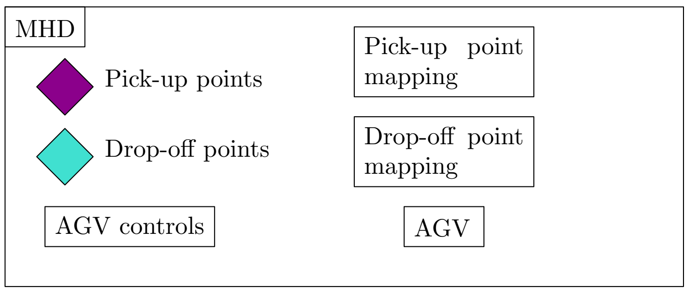
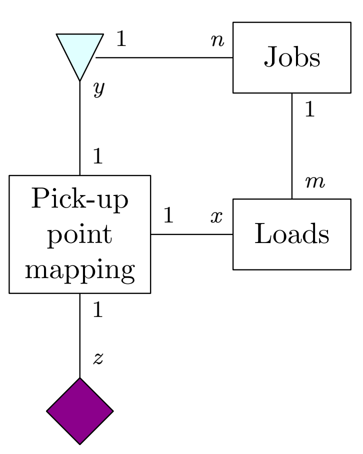


Figure 3.2.7: Overview of material handling domain framework components

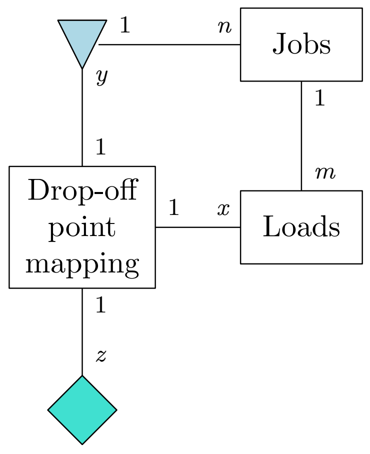
The MHD is concerned with the physical movement of loads by the AgvSorter system. MHD components are oblivious to the logistic reason why the movements are required. Figure 3.2.7 shows the different framework components that belong to the MHD.

What MHT a job results in, depends on the loads that need to be transported, from which grid points these loads can be collected and where they need to be dropped off. The pick-up points from which the loads can be collected are determined by the Pick-up point mapping. Similarly, the drop-off points at which the loads can be dropped off are determined by the Drop-off point mapping. Figure 3.2.8 shows the ERDs of the mappings. Pick-up points and drop-off points can be mapped to two or more queues simultaneously.

The task allocator is part of the AGV controls and receives information from the PD in order to make decisions on task assignments. The information and from which framework components this



(a) Pick-up point mapping



(b) Drop-off point mapping

Figure 3.2.8: ERDs for the pick-up point and drop-off point mappings

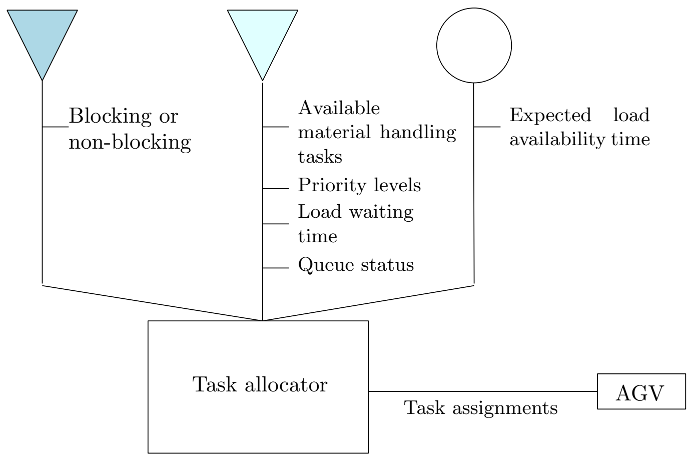


Figure 3.2.9: Information flow to the task allocator

information originates is shown in Figure 3.2.9. The available MHTs are derived from the job pools.

The information provided to the task allocator is given in a way that does not require the MHD to have any information on the logistic processes or the reasons why loads need to be transported. In the job flow flowcharts presented in Section 3.3, the information from the queues is referred to as queue information.

3.3 Job Flow Flowcharts

In the previous section, the framework components are introduced. It is shown that the primary system processes can be abstracted and modeled as a network of processes through which jobs progress. Moving the jobs through queues and processes requires a series of steps to be followed which are captured in the job flow flowcharts. For more detailed information on which framework components are related to certain flowchart steps, see Appendix B.

The flow of jobs, the timing, sending and updating of information is captured in the flowcharts. Certain events trigger the start of the flowchart, such as a job finishing a process. These events result in a series of actions that need to be taken in order to progress the jobs and MHTs. The flowchart is subdivided into several sections based on the functionality covered in that section. Figure 3.3.1 shows an overview of the sections. Ingoing and outgoing arrows indicate starting events and flowchart stops in that section. When sections are linked through arrows, steps in the first section leads to steps in the following section.

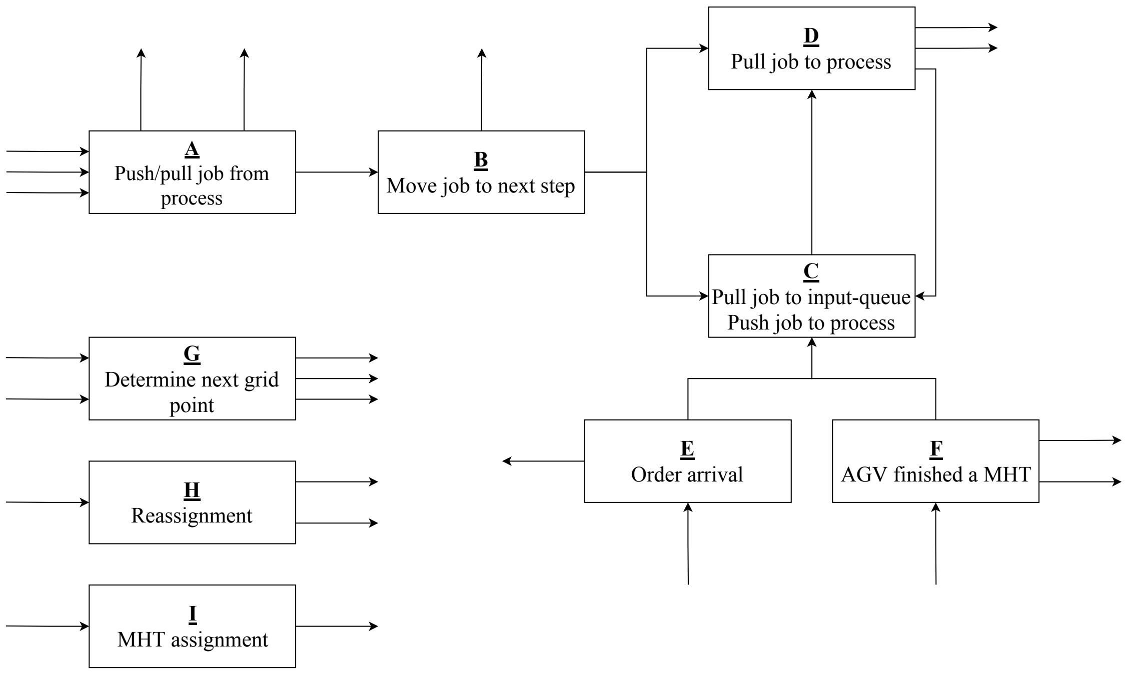


Figure 3.3.1: Overview of the sections of the jobflow flowchart

In Subsections 3.3.1-3.3.9, the different flowchart sections are presented.

3.3.1 Flowchart Section A: Push/Pull Job from Process

Section A of the flowchart is shown in Figure 3.3.2. In this figure a legend is included that holds for each flowchart section shown. For ease of reference, each of the elements in the flowchart is numbered based on their section. This section contains three starting events shown in A1, A3 and A7. These events can result in a job either being pulled from a process by an input-queue or pushed out of a process.

When an AGV loads a load at the first pick-up point for an MHT (A1), the corresponding job should be removed from the output-queue. Jobs that were previously blocked in the preceding process, could therefore be pulled into the queue. Similarly, a space becomes free in the output-queue when an input-queue directly pulls a job from the output-queue (A3).

A job can be pushed from a process when it is finished processing (A7). In A8 it is checked whether the job needs to merge with other jobs before continuing.

3.3.2 Flowchart Section B: Move Job to Next Step

Section B of the flowchart is shown in Figure 3.3.3. This flowchart continues where flowchart section A left off. If the job that is removed from the process is done, this leads to flowchart section D. Here, it is checked whether the process from which the job is removed, can pull a new job in from the preceding input-queue.

If the job is not done, the availability of an output-queue is checked in B3. If the selected output-queue is blocked, the job must remain in the process and the flowchart ends at B4. Otherwise, the job is moved to the selected output-queue where pick-up point and drop-off point information is determined for the loads corresponding to the job. This information can be used by the task allocator to determine travel times, travel distance and the cost for the material handling tasks for each AGV. In case the job needs to be registered in the job pool according to the selectionParameter, the job pool and queue information are updated. Otherwise, only the queue information is updated and sent together with the old job pool in B14. The recipient of this information is either

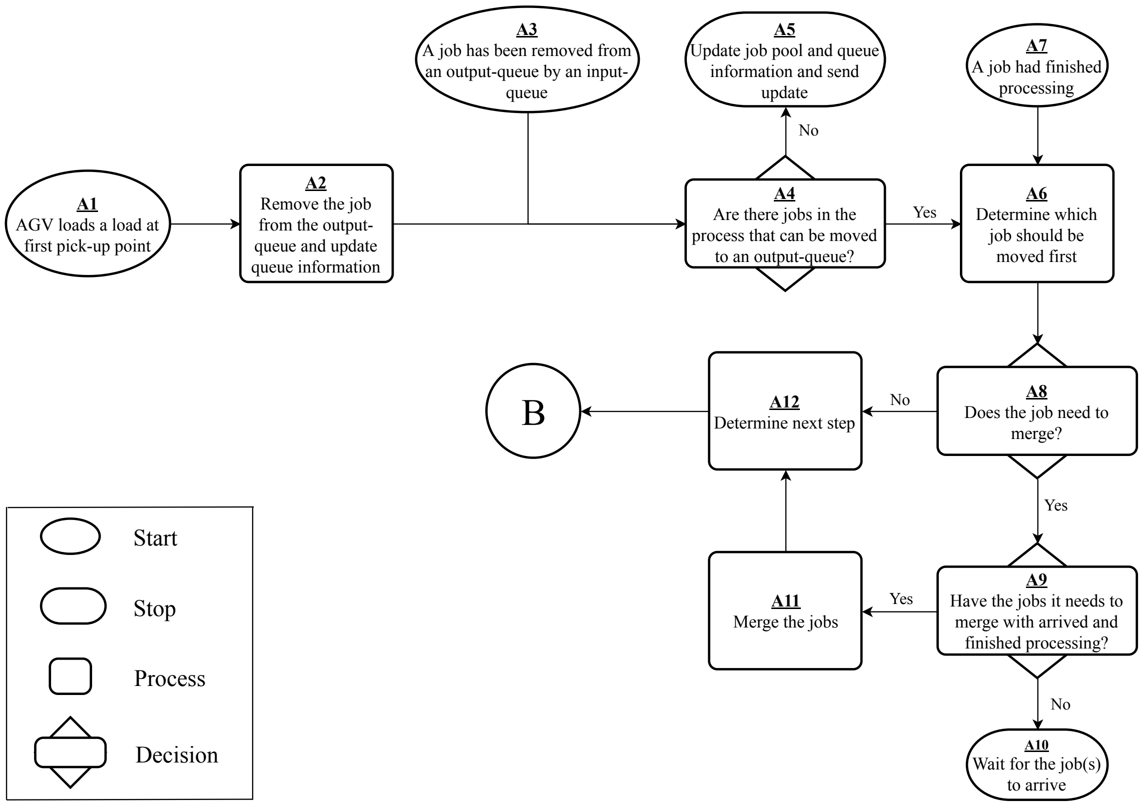

A3
A job has been removed from an output-queue by an input-queue
A5
Update job pool and queue information and send update
No
A7
A job had finished processing
A1
AGV loads a load at first pick-up point
A2
Remove the job from the output-queue and update queue information
A4
Are there jobs in the process that can be moved to an output-queue?
Yes
A6
Determine which job should be moved first
B
A12
Determine next step
No
A8
Does the job need to merge?
Yes
Start
Stop
Process
Decision
A11
Merge the jobs
Yes
A9
Have the jobs it needs to merge with arrived and finished processing?
No
A10
Wait for the job(s) to arrive

Figure 3.3.2: Jobflow flowchart section A

an input-queue or a task allocator.

There is a possibility that the output-queue is followed directly by an input-queue. Therefore, section C is entered after B14 in order to check if an input-queue can pull a job in.

3.3.3 Flowchart Section C: Pull Job to Input-queue, Push Job to Process

Section C of the flowchart is shown in Figure 3.3.4. An input-queue can pull a job in if it has received a non-empty job pool from an output-queue. The steps between C1 and C11 are executed until the job pool is empty or until the input-queue cannot pull any more jobs in.

After pulling a job in, it is checked in C5 whether this job can be moved directly through to the following process. If this is the case it is also checked if the job should be split into multiple jobs (C8) before starting the process. The split is necessary when the loads corresponding to a job proceed in the job flow as separate combinations of loads.

It is also possible to enter flowchart section C from C*, which happens when a job is pushed into the input-queue when the flowchart is started at sections E or F.

In C11 the processing time information is sent to the task allocator. This information can be used to time the pick-up of the loads for that job with the arrival of an AGV. After C11, it is checked whether the input-queue can pull another job in. If this is not the case, section D is reached.

3.3.4 Flowchart Section D: Pull Job to Process

Section D of the flowchart is shown in Figure 3.3.5. This flowchart section checks whether a process can pull a job in from the input-queue. This is possible if a job has been removed from a process in previous flowchart steps.

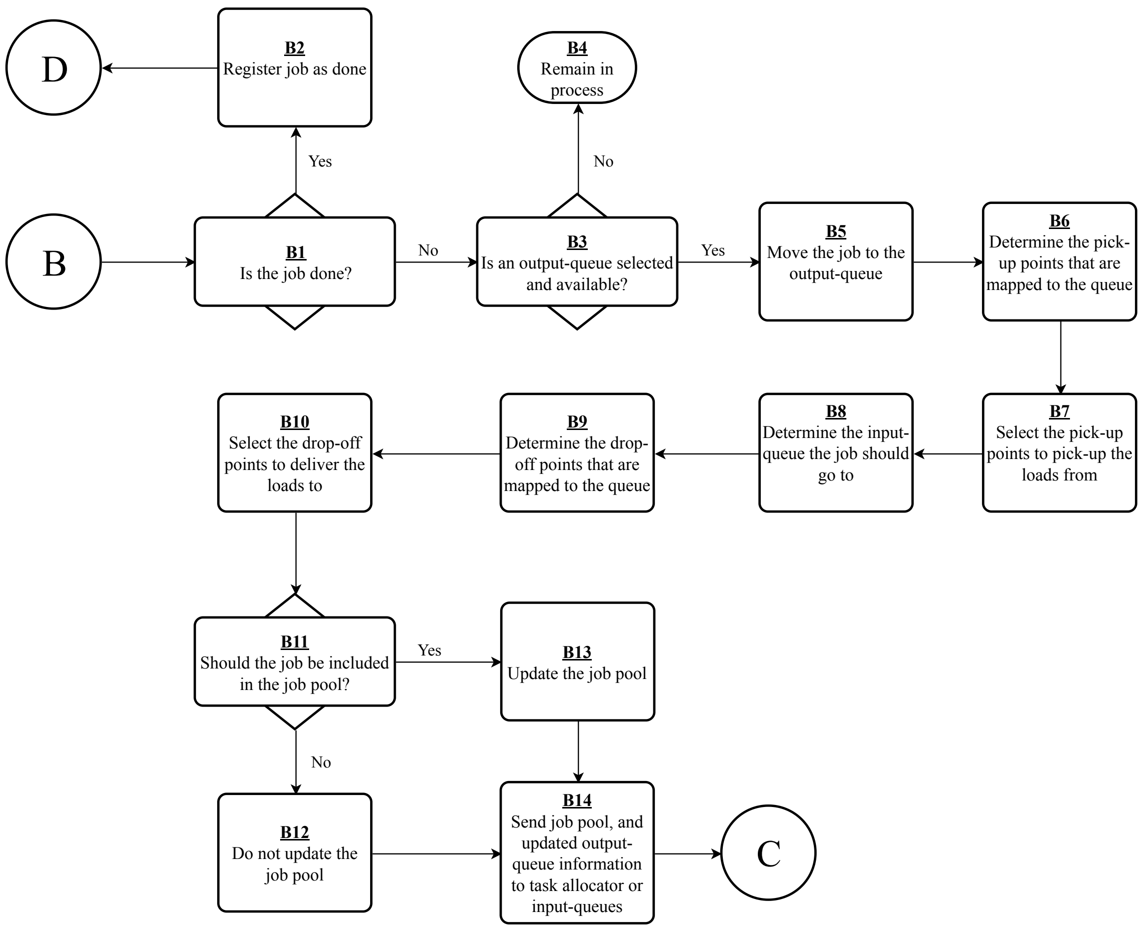

D
B2
Register job as done
B4
Remain in process
Yes
No
B
B1
Is the job done?
No
B3
Is an output-queue selected and available?
Yes
B5
Move the job to the output-queue
B6
Determine the pick-up points that are mapped to the queue
B10
Select the drop-off points to deliver the loads to
B9
Determine the drop-off points that are mapped to the queue
B8
Determine the input-queue the job should go to
B7
Select the pick-up points to pick-up the loads from
B11
Should the job be included in the job pool?
Yes
B13
Update the job pool
No
B12
Do not update the job pool
B14
Send job pool, and updated output-queue information to task allocator or input-queues
C

Figure 3.3.3: Jobflow flowchart section B

3.3.5 Flowchart Section E: Order Arrival

Section E of the flowchart is shown in Figure 3.3.6. When an order arrives and is split into multiple jobs, it is assumed that all jobs move to the same input-queue. All but one job are moved to a job pool that is sent to the input-queue (E4). This job pool does not originate from an output-queue, but is used to list jobs that arrive according to the load file.

For the remaining job it is checked whether there is space in the input-queue that the jobs arrive at. If this is not the case, the job under consideration is also moved into the job pool and waits until the input-queue pulls a job in. If there is space, section C is started at C*. As there are multiple jobs registered in the job pool, the steps in section C are carried out until either the job pool is empty or until the queue and process are both full.

3.3.6 Flowchart Section F: AGV Finished a MHT

Section F of the flowchart is shown in Figure 3.3.7. When an AGV finishes an MHT, the job to which this task corresponds, is moved to an input-queue. This happens when the AGV unloads the load at the final drop-off point that it needs to visit for that material handling task. If the queue is not blocked, section C is started at C*.

3.3.7 Flowchart Section G: Determine Next Grid Point

Section G of the flowchart is shown in Figure 3.3.8. This flowchart section is related to a special case where the information on the input-queue and related drop-off points is not available until after the AGV has picked up the loads for that job and driven to a grid point where a scan is located. This section also illustrates that the AGV must first visit all pick-up points for an MHT prior to visiting any drop-off points.

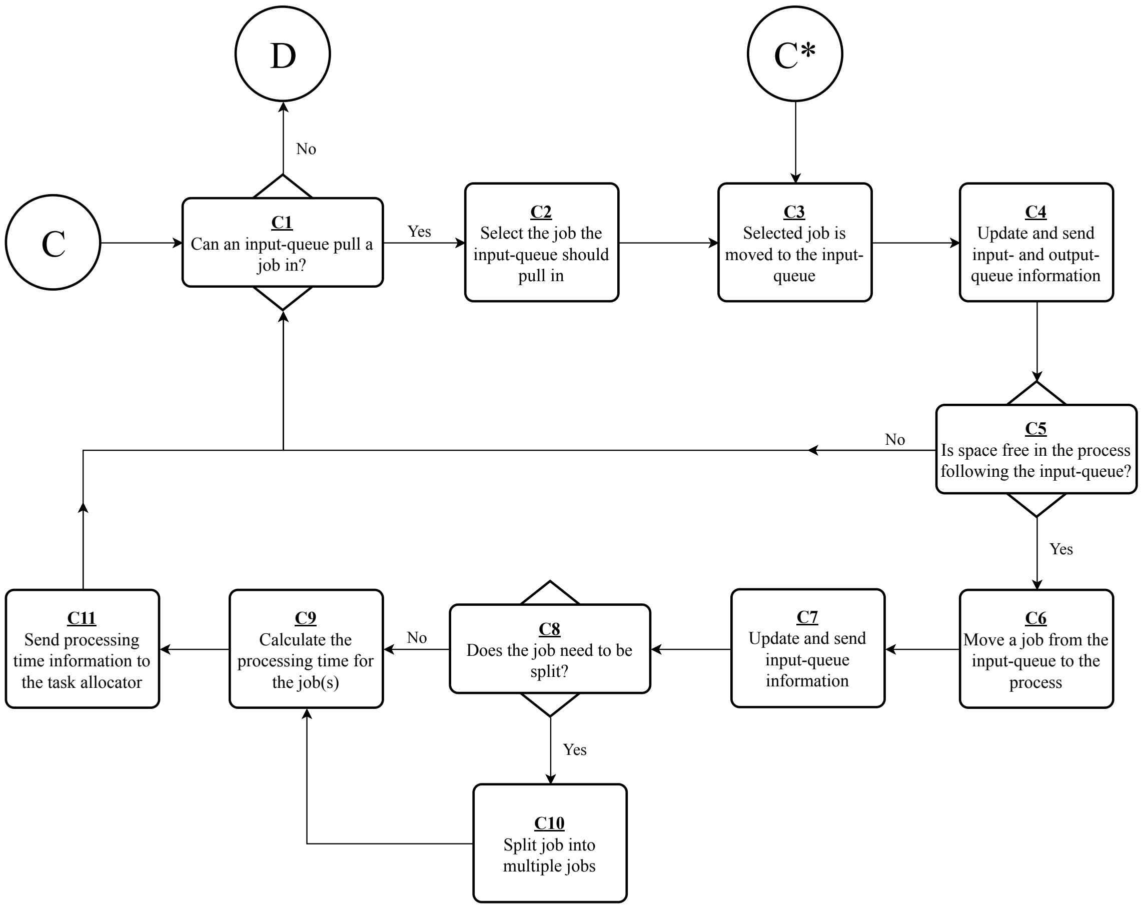

D
C*
C
No
C1
Can an input-queue pull a job in?
Yes
C2
Select the job the input-queue should pull in
C3
Selected job is moved to the input-queue
C4
Update and send input- and output-queue information
C5
Is space free in the process following the input-queue?
No
Yes
C6
Move a job from the input-queue to the process
C7
Update and send input-queue information
C8
Does the job need to be split?
No
Yes
C9
Calculate the processing time for the job(s)
C10
Split job into multiple jobs
C11
Send processing time information to the task allocator

Figure 3.3.4: Jobflow flowchart section C

3.3.8 Flowchart Section H: Reassignment

Section H of the flowchart is shown in Figure 3.3.9. Reassignment is something that can occur in baggage handling systems. When a bag is moved from the screening machine, the results of the screening process may not be processed yet. Meanwhile, the bag is traveling to the next screening level. If the results of the screening come through before the AGV that is carrying the bag reaches a drop-off point, it may reroute to another drop-off point belonging to either EBS or make-up. If the drop-off point is already reached, the reassignment has come too late and is ignored (H4).

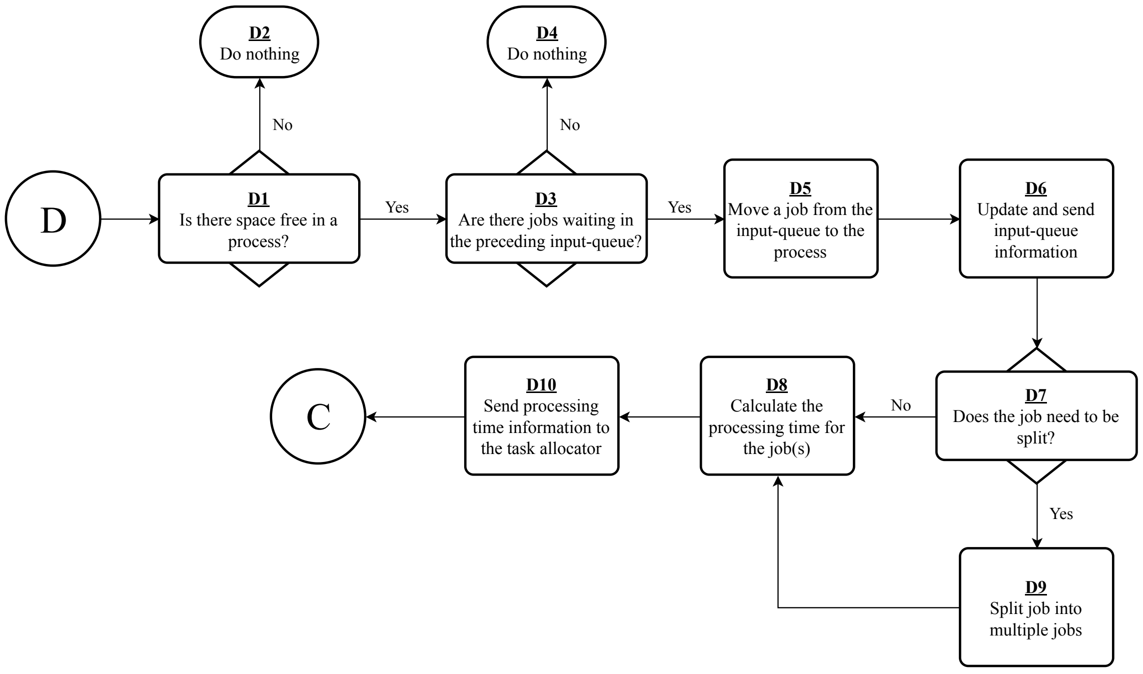
3.3.9 Flowchart Section I: MHT Assignment

Section I of the flowchart is shown in Figure 3.3.10. The task allocator should always request updated information before assigning MHTs. Some information is requestable by the task allocator, other information is imposed. Information that is imposed is sensor information that cannot be updated upon request.

In I2 it should also be possible to assign an AGV a single MHA instead of an MHT depending on the task allocation strategy. If the strategy is followed that is currently implemented in the AgvSorter, only an MHA would be assigned and after loading a load, the rest of the MHT is assigned.

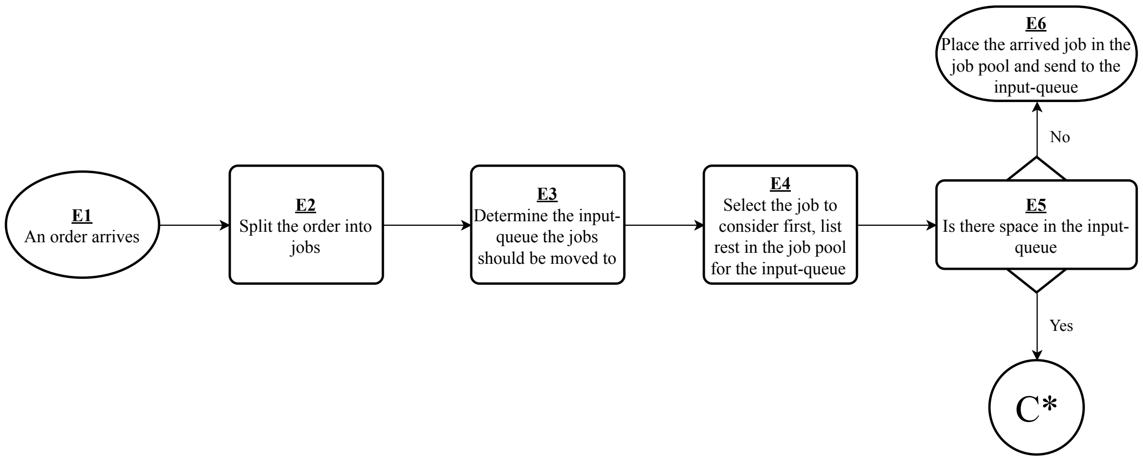
3.4 Conclusion

In this chapter, the design of the framework is presented. First, orders jobs and loads are introduced followed by the framework components. The framework components are designed using the DDD strategy. Lastly, a flowchart is presented that simulates the information and job flow that should result in primary system behavior. In Chapter 4, the design presented is validated.



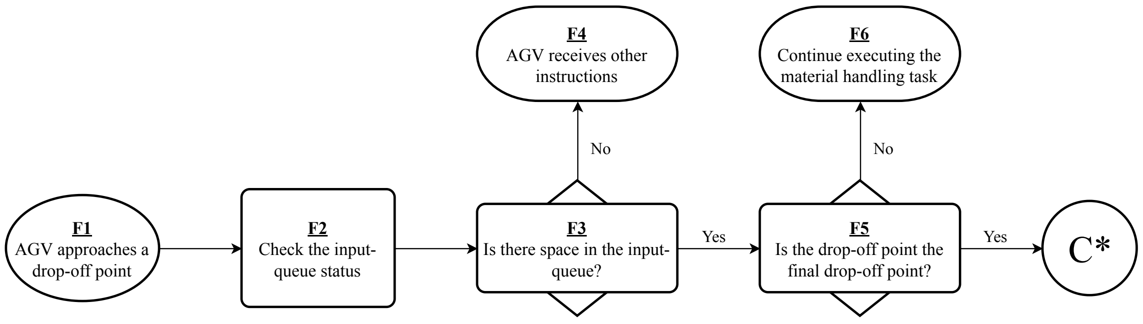
D2
Do nothing
D4
Do nothing
No
No
D
D1
Is there space free in a process?
Yes
D3
Are there jobs waiting in the preceding input-queue?
Yes
D5
Move a job from the input-queue to the process
D6
Update and send input-queue information
C
D10
Send processing time information to the task allocator
D8
Calculate the processing time for the job(s)
No
D7
Does the job need to be split?
Yes
D9
Split job into multiple jobs

Figure 3.3.5: Jobflow flowchart section D



E6
Place the arrived job in the job pool and send to the input-queue
No
E1
An order arrives
E2
Split the order into jobs
E3
Determine the input-queue the jobs should be moved to
E4
Select the job to consider first, list rest in the job pool for the input-queue
E5
Is there space in the input-queue
Yes
C*

Figure 3.3.6: Jobflow flowchart section E



F4
AGV receives other instructions
F6
Continue executing the material handling task
No
No
F1
AGV approaches a drop-off point
F2
Check the input-queue status
F3
Is there space in the input-queue?
Yes
F5
Is the drop-off point the final drop-off point?
Yes
C*

Figure 3.3.7: Jobflow flowchart section F

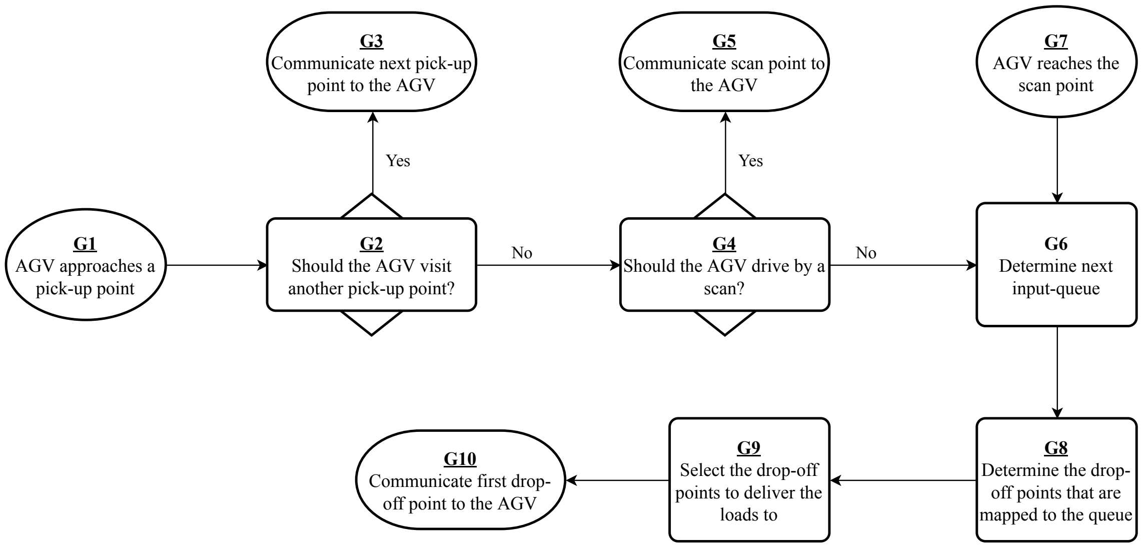

G1
AGV approaches a pick-up point
G2
Should the AGV visit another pick-up point?
Yes
G3
Communicate next pick-up point to the AGV
No
G4
Should the AGV drive by a scan?
Yes
G5
Communicate scan point to the AGV
No
G6
Determine next input-queue
G7
AGV reaches the scan point
G8
Determine the drop-off points that are mapped to the queue
G9
Select the drop-off points to deliver the loads to
G10
Communicate first drop-off point to the AGV

Figure 3.3.8: Flowchart section G

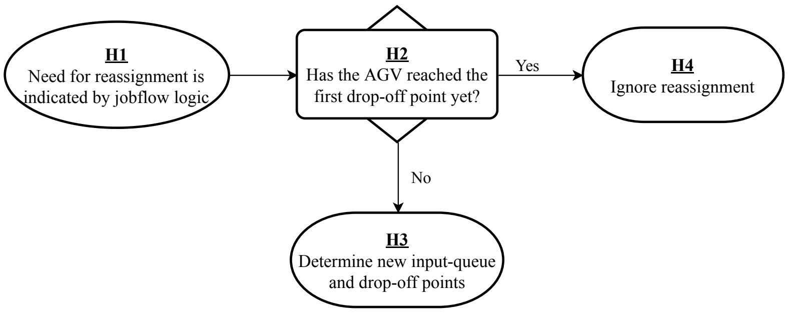

H1
Need for reassignment is indicated by jobflow logic
H2
Has the AGV reached the first drop-off point yet?
Yes
H4
Ignore reassignment
No
H3
Determine new input-queue and drop-off points

Figure 3.3.9: Flowchart section H

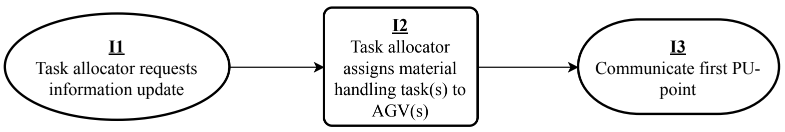

I1
Task allocator requests information update
I2
Task allocator assigns material handling task(s) to AGV(s)
I3
Communicate first PU-point

Figure 3.3.10: Flowchart section I

Chapter 4

Framework Validation

Now that the design of the framework is presented, it needs to be validated to determine whether the framework is able to capture the behavior of the primary systems. First, the validation strategy is presented and subsequently the validation and the results are shown.

4.1 Validation Strategy

The framework is intended to be a tool that can model different complex systems in a consistent, general way. The framework should be usable for analysis of these systems regarding several performance criteria, such as throughput and load waiting time. Next to that, the framework should be usable to derive a task allocation strategy for the AgvSorter. To show that the framework is able to achieve the mentioned goals, the validation is executed in two parts: requirements analysis and scenario analysis.

In the requirements analysis it is validated that each requirement presented in Chapter 2 is incorporated in the framework. This is necessary to show that enough information is available to make an informed decision on the task allocation strategy. It also shows that the behavior of the primary system can be incorporated, which could be used for further analysis of AgvSorter behavior. The scenario analysis supports this by showing that the framework represents the primary system behavior correctly.

Note that the requirements analysis can be seen as a necessary condition for validity and the scenario analysis as a sufficient condition. The requirements analysis shows that all behavior should be present. The scenario analysis supports this but is not capable of covering all possible scenarios or requirements.

The scenario analysis shows that the framework is valid for the chosen scenarios. It indicates that the conceptual design is functional and whether the timing of actions and provision of information in the flowchart incorporates the primary system behavior. The following scenarios are presented:

- Parcel: Single-item flow scenario
- Parcel: Bulk flow scenario
- Baggage handling: Overtaking and blocking scenario
- Baggage handling: Reassignment scenario
- Baggage handing: Conjoint processes scenario
- Warehousing: Inbound process scenario
- Warehousing: Batch picking scenario
- Warehousing: Automated packing and labeling scenario

These scenarios are chosen because they demonstrate a range of functionality of the flowchart that is encountered in real primary systems. The primary systems used for the scenarios are based on existing primary systems or fictive primary systems that Vanderlande has developed. The processes are modeled in framework components and examples are given on how their attributes can be chosen.

4.2 Requirements Analysis

In this section, it is shown for each of the requirements in Chapter 2 how they are included in the framework either as part of the framework or as input for the framework. The results are shown in Tables 4.2.1 till 4.2.5. Where needed, the inclusion of some requirements is explained in more detail.

4.2.1 Arrival Pattern Requirements Validation

In Table 4.2.1, for each arrival pattern requirement it is shown how it is integrated in the framework.

Table 4.2.1: Validation of the requirements for arrival patterns

Requirement ID	**Requirement statement**	**Integration in Framework**
Req.AP.1	It shall be possible to use any arrival pattern encountered in the primary systems as input for the framework.	By use of a load file for job arrival. The arrival file may contain any pattern of arrival times.
Req.AP.2	The framework shall be able to model bulk flows as well as single-item flows.	Through appropriate selection of maxJobs and the selectionParameter for the queues and process.

Note that the arrival patterns in Req.AP.1 are used as input for the framework and not as part of the framework. To highlight requirement Req.AP.2, a bulk flow is represented by allowing the maxJobs of the queues and processes to become very large. The selectionParameter should then incorporate a large portion of the queue as any parcel from a heap of parcels can be selected. This requirement is incorporated in the component attributes but not explicitly in the flowchart.

4.2.2 Primary System Process Requirements Validation

In Table 4.2.2, for each primary system process requirement it is shown how it is integrated in the framework.

Requirements Req.PSP.1 till Req.PSP.6 require the user to define the complexity of the model of the primary systems. The framework provides the inputs that must be delivered, which are the queue and process attributes. The user is responsible for determining the values of these attributes. A simple deterministic processing time can be used, but models including random or planned outages can also be included.

The incorporation of multiple AgvSorters there must be a distinction between which AgvSorter receives which process and queue information. The AgvSorter does not receive information on MHTs resulting from jobs that are located at processes or queues it does not support. By keeping all information related to processes the AgvSorter does not support black-box, multiple AgvSorters can be incorporated to support one primary system.

4.2.3 Multi-load AGV Requirements Validation

In Table 4.2.3, for each multi-load AGV requirement it is shown how it is integrated in the framework.

Table 4.2.2: Validation of the requirements for primary system processes

Requirement ID	Requirement statement	Integration in Framework
Req.PSP.1	The framework shall be able to incorporate variable and deterministic processing times.	Processing times can take on any value based on how the user chooses to calculate them.
Req.PSP.2	The framework shall be able to specify the capacities for each process.	Through the maxJobs attribute the capacity can be specified of each queue and each process.
Req.PSP.3	The framework shall incorporate the possibility to model variability.	Processing times are calculated separately for each job (C9, D8) and can therefore include delays for some jobs caused by random and planned outages.
Req.PSP.4	The framework shall incorporate buffers.	Through inclusion of the input-queue and output-queue components. Different types of buffers can be modeled through choice of the queue attributes.
Req.PSP.5	It shall be possible to model a sequence of processes within the framework.	Through the jobflow logic component and combinations of queues and processes following each other, each job can follow a sequence of processes.
Req.PSP.6	The framework shall be able to model a primary system that has multiple AgvSorters implemented.	This is done by sending information for each component separately. Each time information is sent, this can be to a different recipient. Information in the flowchart is sent in A2, B14, C4, C7, C11, D6, D10, I1.

Using a batch picking example, requirement Req.ML.1 is highlighted. An AGV can be instructed to execute the MHTs resulting from multiple jobs that belong to different orders. This corresponds to batch picking. When the AGV unloads the loads at a packing station, the loads can be sorted and packed into a package. This results in a new combination of loads, which can be seen as a new (merged) job.

Requirement Req.ML.2 is explained further. A material handling task may require picking-up multiple loads on different locations and moving them to one drop-off point. Multiple material handling tasks may also be assigned, which can correspond to picking up multiple parcels or bags at either the same pick-up point or multiple different pick-up points and delivering them to the same drop-off point or different drop-off points.

Requirement Req.ML.5 is not explicitly included in the framework itself. The user should define how the order split is carried out based on logistic rules and business rules and the capacity constraints of an AGV. This way, the user should guarantee that the constraint is met and the framework provides the user with the tools to meet the constraint.

4.2.4 Information Flow Requirements Validation

In Table 4.2.4, for each information flow requirement it is shown how it is integrated in the framework.

Table 4.2.3: Validation of the requirements for multi-load AGVs

Requirement ID	Requirement statement	Integration in Framework
Req.ML.1	The framework shall support the use of different order picking strategies.	By splitting orders into multiple jobs (E2), AGVs being allowed to execute multiple MHTs at a time and jobs being able to merge in a process, this requirement is satisfied.
Req.ML.2	The framework shall incorporate many-to-many, many-to-one, one-to-many and one-to-one material handling tasks.	An AGV is allowed to execute a combination of material handling tasks or one material handling task consisting of one or more loads.
Req.ML.3	Strict and relaxed sequencing must be part of a material handling task.	The sequence group a load belongs to is included as a load attribute. The specific combination of loads as part of an MHT defines the order in which the loads must be collected. Hence, strict or relaxed sequencing is captured in the framework.
Req.ML.4	Dimensions of loads shall be a factor that influences the material handling tasks.	Dimensions (length, width, height and weight) are included as a load attribute and can therefore influence the combination of loads included in a job.
Req.ML.5	The capacity needed to execute a material handling task shall not exceed the capacity constraints of an AGV.	The order is required to split into jobs such that the resulting MHTs do not exceed the capacity constraints of the AGV. This is done by taking into account the dimensions of the load combination of a job.

Table 4.2.4: Validation of the requirements for information flow

Requirement ID	Requirement statement	Integration in Framework
Req.IF.1	The framework shall take into account priority levels.	By inclusion of priority levels as a job property, the information on priority levels is available for the task allocator when requesting and receiving output-queue information.
Req.IF.2	The framework shall be able to adjust the timing of sending information based on the primary system configuration.	The flowchart incorporates multiple instances of sending (the same) information.
Req.IF.3	The information delivered by the framework shall correspond to the information flow that is present in the primary system as represented by the framework.	In the flowcharts all information is sent. In the task allocation strategy, usable information should be filtered.

An important example of how Req.IF.2 is incorporated is when the information on the drop-off points for an MHT becomes available. In B8, B9 and B10 this information is determined when the job enters the output-queue. If this information is available, a GAP task allocation strategy can be used as this allows the calculation of the total costs of an MHT for an AGV. However, if this information is not yet available in a primary system due to an identification scan happening after the output-queue, this information is not usable and is obtained later in flowchart section G.

Regarding requirement Req.IF.3, in the flowcharts, all information that could be available in a primary system is included. This is done as the framework must remain generic for each type of primary system. When the framework is used to develop a task allocation strategy for a specific primary system configuration, the user should identify which of the sent information should be used.

4.2.5 Task Allocation Strategy Requirements Validation

In Table 4.2.5, for each task allocation requirement it is shown how it is integrated in the framework.

Table 4.2.5: Requirements for task allocation strategies

Requirement ID	**Requirement statement**	**Integration in Framework**
Req.TA.1	The Task Allocator shall be able to obtain information on the distance between grid points.	Not explicitly included in the framework. Dependent on the layout of the grid.
Req.TA.2	The Task Allocator shall be able to obtain information on queue statuses.	According to the flowchart, queue information is communicated upon changes in the queue (B14, C4, C7, D7, E6).
Req.TA.3	The Task Allocator shall be able to obtain information on travel times between an AGV and a target location.	Pick-up points and drop-off points are determined when a job enters the output-queue (B6-B10). This can be used to determine the travel times.
Req.TA.4	The Task Allocator shall be able to obtain information on load waiting time.	Load waiting time is the time a job spends in an output-queue and is included as attribute of a job.
Req.TA.5	The Task Allocator shall be able to obtain information on load handling time.	The Task Allocator has access to all grid and AGV related information. By keeping track of when an AGV reaches a pick-up point until it reaches the drop-off point where the load is delivered, the load handling time is obtained.

For Req.TA.1 it is assumed that the task allocator has access to information on the distance between grid points as it is stated in Subsection 3.2.4, that the task allocator has access to all AGV and grid related information.

4.2.6 Discussion of the Requirements Validation

From the requirements validation it can be concluded that the requirements are either captured explicitly as part of the framework or as user responsibility. Requirements that are part of the framework are either incorporated in the order-job-load relationship and attributes, the framework components, the flowchart or as input for the framework in the form of a load file.

As there are many variables that need to be taken into account for each primary system, the framework does not provide elaborate models for each primary system process. Instead it provides attributes and components that the user can combine for either simple or more complex analysis.

4.3 Parcel Scenario Analysis

In the following three sections the scenario analysis is carried out for each type of primary system. In this section the parcel scenarios are presented.

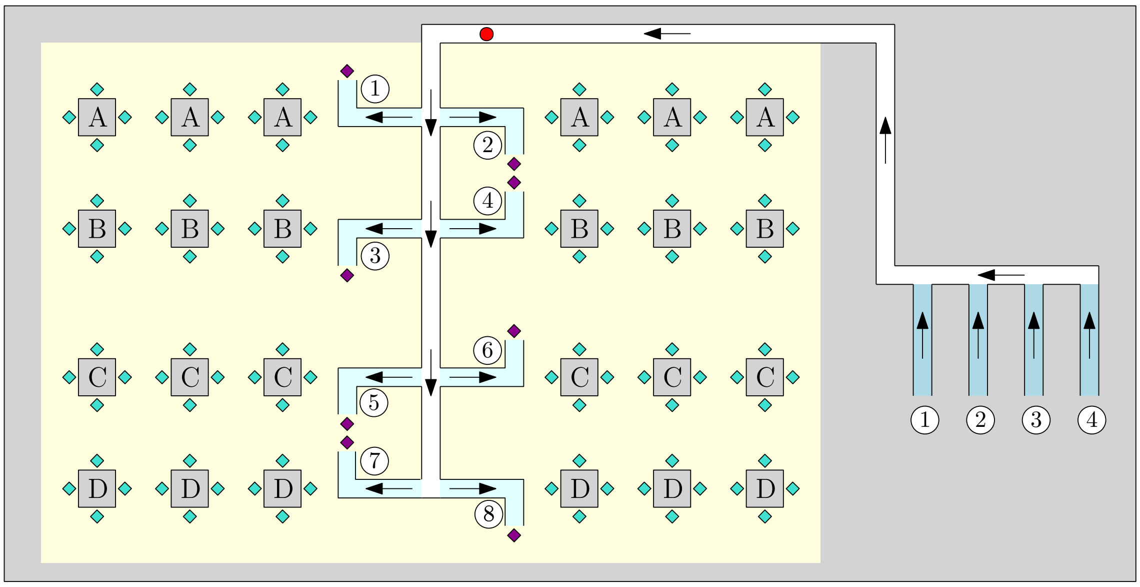


Figure 4.3.1: Parcel scenario setting for single item flow

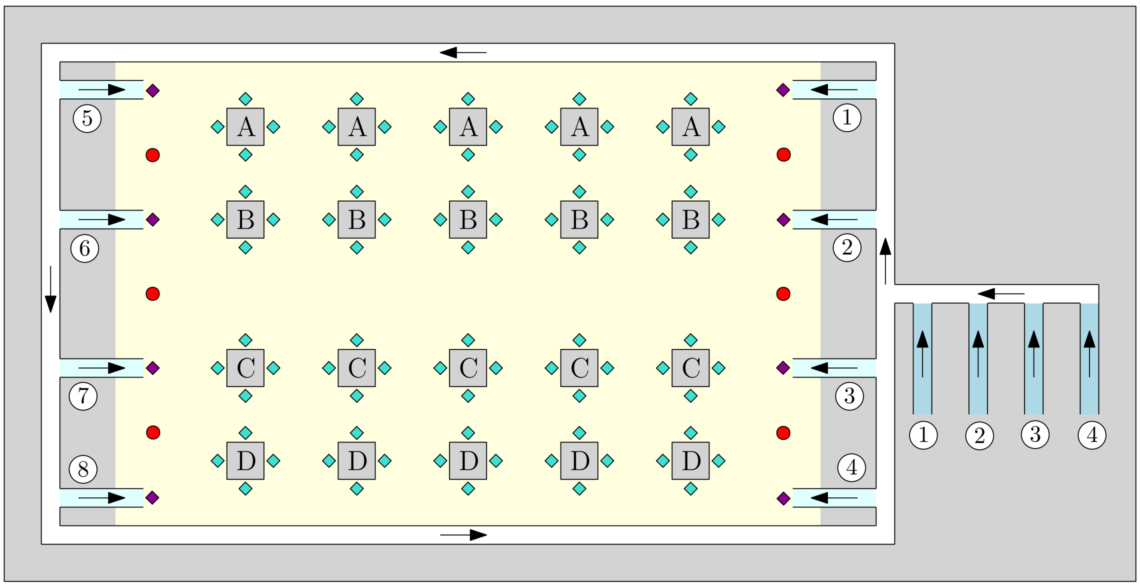


Figure 4.3.2: Parcel scenario setting for bulk flow

A top view of the parcel primary systems that are used for the parcel scenarios are shown in Figures 4.3.1 and 4.3.2. These figures are schematics based on an existing parcel depot. In this depot, the application of the AgvSorter system is researched by Vanderlande, which has resulted in the layouts shown in the figures. As these layouts are created based on a real parcel depot and it has been researched whether the AgvSorter could be applied here, these layouts are representative to conduct the validation with.

In the figures, a conveyor system transports the parcels from arriving trucks to the grid. The conveyor system is raised above the grid and the parcels in the bulk flow system arrive to the grid via slides. The parcels in single-item flow are moved down using conveyors.

In each of the scenario setting figures in this and the following sections, the light grey area represents the boundaries of the primary system and the yellow area is the AgvSorter grid. Processes and their queues are shown in the colors introduced in Chapter 3. The red dot is a scanner that can identify parcels (or bags in a baggage handling setting).

Figure 4.3.1 shows the parcel scenario primary system for a single-item flow. Trucks that arrive at the primary system, unload the parcels onto one of the four input-conveyors that lead up to the central transport conveyor. This conveyor leads all parcels through a scanner one by one. This allows de system to pre-sort the parcels such that the parcel is near its destination chute. Pick-up points are located at the bottom of the conveyors leading out from the central conveyor. There are 24 chutes located in this layout through which AGVs can drop parcels. Each chute is surrounded by four drop-off points. The destination the chute leads to is denoted by the letter in the chute.

In the bulk flow setting in Figure 4.3.2, the central conveyor consists of four input-queue conveyors leading up to the central conveyor system that loops around the grid. The output-queues are slides where photocells are located that can detect whether the slide is 25%, 50%, 75% and 100% full. Based on this sensor data, the conveyor divides the parcels over the slides. In this system, the AGV is required to drive under a scan that is located above certain grid points before being able to visit the drop-off point. Only after the scan, it is known which parcel the AGV is carrying. When implementing a task allocation strategy, the information sent to the task allocator in B8, B9 and B10 cannot be used. The only information that may be used from the sent queue information is how full the queues are.

4.3.1 Single-Item Flow Scenario

The abstract representation in framework components of the primary system processes is shown in Figure 4.3.3.

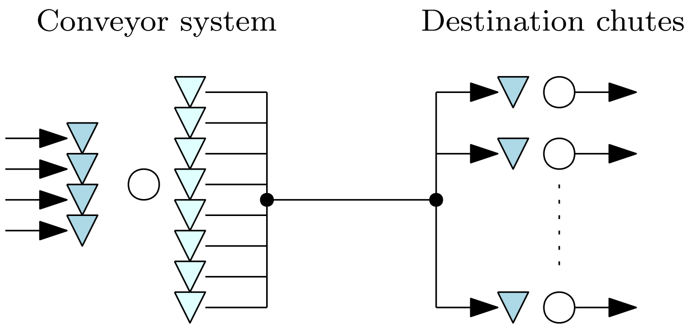


Figure 4.3.3: Parcel processes in framework components

It is chosen to model each chute as a separate input-queue and process. By doing so, Job flow logic can be used to control how many parcels are sorted to each chute in order to ensure even distribution. As all chutes are identical, they have the same attributes which are shown in Table 4.3.1. At the chutes, no physical input-queue is available. In this case, the input-queue component is used for the drop-off point mapping and as a hatch through which jobs are passed to the process in the same time instance the job enters the queue. This is guaranteed by the structure of the flowcharts and the fact that the chutes have an infinite capacity. In other words, a parcel can always be dropped in the chute upon arrival.

The conveyor system is modeled as indicated by the blue, light blue and white colors in Figure 4.3.1, with four input-queues and eight output-queues. To each output-queue one pick-up point is mapped. As the input-queues are not preceded by an AgvSorter system, no drop-off points are mapped here. The attributes for the conveyor system are shown in Table 4.3.1. The values in this table are assumptions based on the estimated size of the conveyors. In reality these numbers may differ. This has no impact on the validation of the framework.

It takes 60 seconds to go across the entire conveyor from conveyor input-queue 4 until output-queue 7 or 8. It depends on which input-queue a job arrives at and which output-queue the job is headed for how long the processing time takes. If the user wants to incorporate the traveling time on the input-queue and output-queue conveyors, these can be added as constants to the processTime. For this scenario analysis, the processing time for the central conveyor has a set value between 30 and 60 seconds depending on the queue the job arrives at.

Table 4.3.1: Attributes of the framework components for the single-item flow scenario

Framework component	**selectionParameter**	**maxJobs**	**processTime [s]**
Chutes process	-	Ample	1
Chutes input-queue	1	≥ 1	-
Conveyor process	-	500	[30,60]
Conveyor input-queues	1	30	-
Conveyor output-queue	1	30	-

For the single-item flow case the scenario that is studied concerns two parcels that each arrive at a different input-queue. Job1 is headed for destination A and Job2 is headed for destination D. There are other jobs in the processes, but they do not cause any blocking for the arriving jobs as the number of jobs in the process is less than 500 and are therefore ignored in this scenario analysis. For the rest of the scenarios studied in this section and the next sections, this assumption also holds. The first scenario steps are shown in Table 4.3.2. In the tables, the processing times for jobs are indicated with the abbreviation pt. The leftmost column contains the jobs affected by the flowchart steps. The second column shows the steps that need to be taken according to the flowchart. The third and fourth column show the locations of the jobs before and after the flowchart steps. Each row indicates an event starting the flowchart.

Table 4.3.2: Single-item flow scenario steps for t = 0

$t = 0$	Job1 and Job2 arrive simultaneously		
	Flowchart steps	**Location before**	**Location after**
Job1	E1, E2, E3, E4, E5, C3, C4, C5, C6, C7, C8, C9, C11, C1, D1, D3, D4	Arriving at conveyor input-queue 1	Conveyor process (pt = 30 s)
Job2	E1, E2, E3, E4, E5, C3, C4, C5, C6, C7, C8, C9, C11, C1, D1, D3, D4	Arriving at conveyor input-queue 4	Conveyor process (pt = 60 s)

At t = 0, the arrival of the two jobs at the input-queues for the conveyor are treated as two separate events as each parcel corresponded to a separate order. Should both parcels have been considered as one order, this would have resulted in one event where after C1, C2 would have followed to pull Job2 into the queue as well. The next event is scheduled at t = 30, as Job1 finishes processing at that time. The job is moved to an output-queue where several other jobs are located. Physically this means that the parcel has finished traveling along the conveyor and is diverted to the downward conveyor modeled by output-queue 2. The steps starting at t = 30 are shown in Table 4.3.3.

Job1 is moved from the process to conveyor output-queue 2. At B11 the job is not included in the job pool as there are other jobs waiting and the selectionParameter for the queue is 1. Job2 does not have any events scheduled at this time as it is still being processed. The next event is scheduled at t = 60 when Job2 is finished processing. At this time, Job1 is located in front of the queue and is included in the job pool. At t = 60, an AGV loads a load at the pick-up point that is mapped to the queue where Job1 is located. Table 4.3.4 shows the resulting flowchart steps.

Table 4.3.3: Single-item flow scenario steps for t = 30

$t = 30$	Job1 is finished processing		
	Flowchart steps	**Location before**	**Location after**
Job1	A7, A6, A8, A12, B1, B3, B5, B6, B7, B8, B9, B10, B11, B12, B14, C1, D1, D3, D4	Conveyor process	Conveyor output-queue 2

Table 4.3.4: Single-item flow scenario steps for t = 60

$t = 60$	Job2 is finished processing, Job1 is removed from the output-queue		
	Flowchart steps	**Location before**	**Location after**
Job2	A7, A6, A8, A12, B1, B3, B5, B6, B7, B8, B9, B10, B11, B13, B14	Conveyor process	Conveyor output-queue 7
Job1	A1, A2, A4, A5	Conveyor output-queue 2	Moving
AGV	G1, G2, G4, G6, G8, G9, G10	-	-

The steps taken in Table 4.3.4 depend on which of the two events is considered first. It is assumed that the event where Job2 finishes processing is considered first as this event was scheduled when Job2 entered the process. Note that Job2 is included in the job pool for conveyor output-queue 7 as there are no other jobs in the queue upon arrival of Job2. However, if the event removing Job1 from the output-queue happened first, the decision at A4 in the flowchart would detect that Job2 had finished processing and started the process of moving that job to an output-queue. This would result in A6 instead of A5.

The steps in flowchart section G are also included as the AGV approached the pick-up point to execute the MHT for Job1.

For the remaining validation steps, only Job1 is considered as the remaining validation steps are identical for Job2. The event that starts the flowchart steps is the AGV executing the material handling task corresponding to Job1 approaching a drop-off point. Tables 4.3.5 and 4.3.6 show the final steps for this scenario.

Table 4.3.5: Single-item flow scenario steps for t = 90

$t = 90$	AGV approaches drop-off point while executing the MHT for Job1		
	Flowchart steps	**Location before**	**Location after**
Job1	F1, F2, F3, F5, C3, C4, C5, C6, C7, C8, C9, C11, C1, D1, D3, D4	Moving	Chute process (pt = 1)

Table 4.3.6: Single-item flow scenario steps for t = 91

$t = 91$	Job1 is finished processing		
	Flowchart steps	**Location before**	**Location after**
Job1	A7, A6, A8, A12, B1, B2, D1, D3, D4	Chute process	Done

The steps show how the input-queue at the chutes directly passes the job through to the process.

This illustrates that the input-queue component does not have to be an existing, physical buffer.

For more information on several frequently encountered flowchart steps, the steps at t = 0 and t = 30 are elaborated in more detail in Appendix C.

4.3.2 Bulk Flow Scenario

The framework components form the same sequence of processes as shown in Figure 4.3.3. The difference lies in the calculation of the process times for the jobs on the central conveyor, the output-queue determination and the number of chutes.

In the single-item flow case it is relevant to take into account the distinct travel time for each parcel due to the pre-sort process. However, as the bulk flow results in a chaotic parcel flow, there is no pre-sort and the large number of parcels are divided equally over the slides. Exactly which parcel ends up at what output-queue is unknown to the primary system until the parcel is driven under the scan by an AGV. The choice can therefore be made to use a deterministic processing time and divide the jobs over the slides based on how many parcels there are in the slide. Table 4.3.7 shows the conveyor attributes for the bulk flow case. The chute attributes remain the same as the single-item flow case and are listed in Table 4.3.1.

Table 4.3.7: Attributes of the framework components representing the conveyor in the bulk flow scenario

Framework component	**selectionParameter**	**maxJobs**	**processTime [s]**
Conveyor process	-	1000	30
Conveyor input-queues	1	100	-
Conveyor output-queue	20	100	-

The operator can reach approximately 20 in the heap of parcels on the slide. Therefore, the selectionParameter is set to 20 here.

In the scenario that is studied it is assumed that there for the time being are no other parcels in the system. The scenario studied is a multi-load case where an AGV is executing two MHTs at once. Physically, this means that the AGV is carrying two parcels at the same time. The scenario starts at t = 0 when Job1 and Job2 are located at conveyor output-queue 2. Table 4.3.8 shows the steps taken at t = 0.

Table 4.3.8: Bulk flow scenario steps for t = 0

	Flowchart steps	**Location before**	**Location after**
$t = 0$	AGV loads a load at the pick-up point mapped to output-queue 2		
Job1	A1, A2, A4, A5	Conveyor output-queue 2	Moving
AGV	G1, G2, G3	-	-
Job2	A1, A2, A4, A5	Conveyor output-queue 2	Moving
AGV	G1, G2, G4, G5	-	-

First, the AGV loads the load corresponding to the MHT for Job1. Next, in flowchart section G the AGV receives instructions to go to another pick-up point (G3). In this case, these are instructions to load another load at the pick-up point it is already located on, which results in the flowchart starting at A1 for Job2. After loading the load for the second MHT, the AGV is instructed to go to a scan point. Next, the AGV arrives at the scan point as described in Table 4.3.9.

Table 4.3.9: Bulk flow scenario steps for t = 10

$t = 10$	AGV approaches the pick-up point at output-queue 2		
	Flowchart steps	**Location before**	**Location after**
AGV	G7, G6, G8, G9	Conveyor output-queue 2	Moving

It is assumed that when an AGV is carrying two parcels, it is sufficient to drive under the scan once in order to obtain the drop-off point information for both parcels. Each parcel in this scenario has to go to a different destination. Therefore, the AGV needs to visit two drop-off points to deliver both parcels. The arrival at the first drop-off point is shown in Table 4.3.10.

Table 4.3.10: Bulk flow scenario steps for t = 20

$t = 20$	AGV approaches a drop-off point		
	Flowchart steps	**Location before**	**Location after**
Job1	F1, F2, F3, F5, C3, C4, C5, C6, C7, C8, C9, C11, C1, D1, D3, D4	Moving	Chute process

Note that at steps F5 and F6 the drop-off point is denoted as the final drop-off point while still another drop-off point needs to be visited to finish executing the MHT for Job2. This is because both parcels belong to two different MHTs. Now, the AGV can either receive the instruction to drop-off the remaining parcel or to visit a pick-up point to visit another parcel. In [15] a task allocation strategy is shown for this type of multi-load case. For this scenario, it is assumed that the AGV is instructed to visit the drop-off point for the second MHT, resulting in the same steps for Job2 as for Job1 in Table 4.3.10.

4.4 Baggage Handling Scenario Analysis

The baggage handling scenario setting is based on a fictive airport setting. The layout is used by Vanderlande to illustrate the use of the Baxorter system, which is one of the baggage handling systems Vanderlande develops. It is representative of the size of a large number of airports and contains the characteristic components of a baggage handling system. It is therefore a representative setting for the validation. A schematic adaptation of the system is shown in Figure 4.4.1. This baggage handling system in framework components is shown in Figure 4.4.2.

Screening consists of three levels. The first two levels are carried out by three identical, parallel machines. It is chosen to model the three machines as three separate processes. Job flow logic is required to divide bags over the three machines. It is also possible to model the three machines as one process with one input-queue and one output-queue and five mapped drop-off points and pick-up points. However, choosing to do so assumes that the bags are automatically divided properly over the machines. The third screening level is a separate station that is automated in this setting. Should bags not pass this screening level, they are removed from the system and the job is considered done.

The EBS system is a lane-based EBS system where bags are stored in different lanes based on the time until make-up opens. When make-up opens, the bags are pushed out of the lanes and onto a conveyor that takes them back to the grid. The last part of the conveyor is considered the output-queue.

The make-up chutes in the floor lead to conveyors underneath the baggage handling system where they are loaded onto carts and transported to the flight. Note that the arrow in Figure 4.4.2 originating just before make-up and going back to MES is in place because the bag tracing system may lose tracking of the bag.

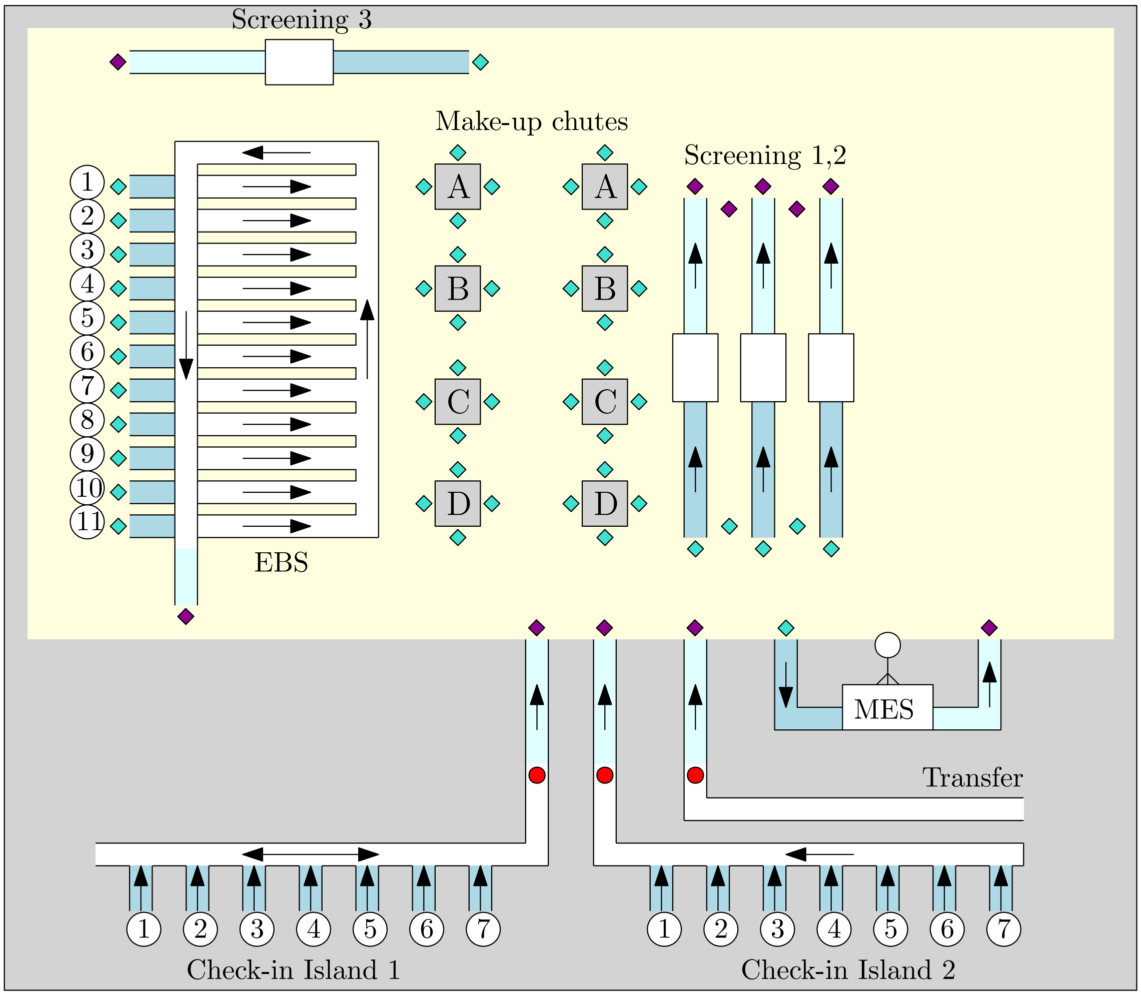


Figure 4.4.1: Baggage handling scenario setting

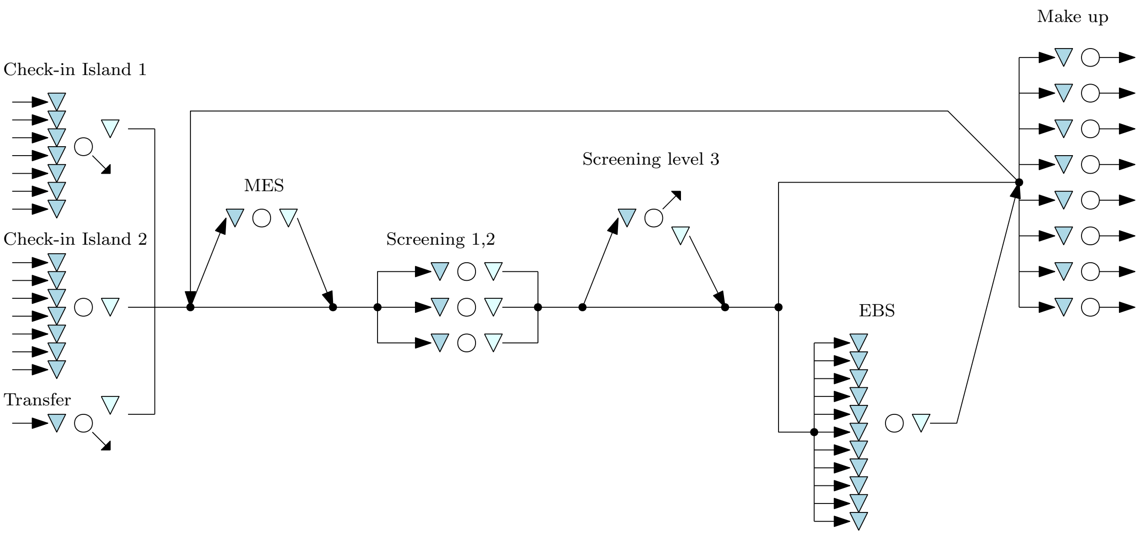


Figure 4.4.2: Baggage handling processes as framework components

Bags may arrive either as transfer bags or at one of the check-in islands. There is a scan located at each of these processes. However, if this scan fails, the bags are routed to the MES station, which is a manual workstation with an input-queue conveyor and an output-queue conveyor. Figure 4.4.2 shows the last part of the transfer process. Most of the transfer process takes place when the bag is unloaded from the airplane and transported to the conveyor shown in the figure. This is added to the processing time.

Bags that are checked in at Check-in Island 1 can either be odd sized bags or normal sized bags that the baggage handling system can handle. Odd sized bags go through the baggage handling system differently. They still need to be taken into account as arriving jobs as they take up space on the conveyor.

The attributes of the processes and queues are shown in Table 4.4.1. As with the parcel validation, these are fictive numbers to aid in the validation process.

Table 4.4.1: Attributes for the baggage handling framework components

Framework component	**selectionParameter**	**maxJobs**	**processTime** [s]
Check-in Island 1 and 2 processes	-	30	[5, 30]
Check-in Island 1 and 2 input-queues 1-7	1	2	-
Check-in Island 1 and 2 output-queue	1	4	-
Transfer process	-	Ample	1200
Transfer input-queue	1	≥ 1	-
Transfer output-queue	1	5	-
MES process	-	1	180
MES input-queue	1	5	-
MES output-queue	1	5	-
Screening 1,2 process	-	1	60
Screening 1,2 input-queue	1	7	-
Screening 1,2 output-queue	1	7	-
Screening 3 process	-	1	60
Screening 3 input-queue	-	7	1
Screening 3 output-queue	-	7	1
EBS process	-	300	[3600, 86400]
EBS input-queues 1-11	1	3	-
EBS output-queue	1	6	-
Make-up chute process	-	Ample	1
Make-up chute input-queue	1	≥ 1	-

The check-in islands and EBS are assumed to have variable processing times in this validation. The processing times for the check-in islands and EBS are dependent on the input-queues. The values shown in this table are examples of how this system can be modeled and is used for this particular validation. A user may choose to model the processes differently.

4.4.1 Baggage Handling Overtaking and Blocking Scenario

In this scenario, there are two bags arriving at Check-in Island 1. First Job1 arrives at t = 0 at input-queue 1 and at t = 15 Job2 arrives at input-queue 7. In this scenario, Job2 should overtake Job1 on the conveyor. Also incorporated in this scenario is blocking at the screening machines. The steps for t = 0 and t = 15 are shown in Tables 4.4.2 and 4.4.3.

Table 4.4.2: Baggage handling overtaking and blocking scenario steps for t = 0

$t = 0$	Job1 arrives		
	Flowchart steps	**Location before**	**Location after**
Job1	E1, E2, E3, E4, E5, C3, C4, C5, C6, C7, C8, C9, C11, C1, D1, D3, D4	Arriving at Check-in Island 1 input-queue 1	Check-in Island 1 process (pt = 30)

Table 4.4.3: Baggage handling overtaking and blocking scenario steps for t = 15

$t = 15$	Job2 arrives		
	Flowchart steps	**Location before**	**Location after**
Job2	E1, E2, E3, E4, E5, C3, C4, C5, C6, C7, C8, C9, C11, C1, D1, D3, D4	Arriving at Check-in Island 1 input-queue 7	Check-in Island 1 process (pt = 5)

At t = 30, both Job1 and Job2 are finished processing. In Table 4.4.4 the steps are shown.

Table 4.4.4: Baggage handling overtaking and blocking scenario steps for t = 30

$t = 30$	Job1 and Job2 finish processing simultaneously		
	Flowchart steps	**Location before**	**Location after**
Job1, Job2	Job1: A7, A6 $\rightarrow$ Job2: A8, A12, B1, B3, B5, B6, B7, B8, B9, B10, B11, B13, B14, C1, D1, D3, D4	Job1: Check-in Island 1 process, Job2: Check-in Island 1 process	Job1: Check-in Island 1 process, Job2: Check-in Island 1 output-queue
Job1	A7, A6, A8, A12, B1, B3, B5, B6, B7, B8, B9, B10, B11, B12, B14, C1, D1, D3, D4	Check-in Island 1 process	Check-in Island 1 output-queue

The event where Job1 finished processing is scheduled before Job2 finished processing. However, in A6 a decision rule is implemented that allows Job2 to overtake Job1 based on the input-queue Job2 originated from. This simulates bags overtaking each-other. As Job1 has still finished processing, it is considered again starting at A7 after Job2 is moved to the output-queue.

In the next part of the scenario, it is shown what happens when an AGV arrives at a blocked input-queue with a load. For this, only Job2 is considered as Job1 goes through the same steps. Job2 should go to one of the screening machines. When the decision is made at t = 30 to include the job in the job pool, there was still space free in the input-queues at the screening machines. Therefore, an AGV has started executing the MHT corresponding to Job2. Table 4.4.5 shows the steps when the AGV approaches the drop-off point.

Table 4.4.5: Baggage handling overtaking and blocking scenario steps for t = 40

$t = 40$	Job1 and Job2 finish processing simultaneously		
	Flowchart steps	**Location before**	**Location after**
Job2	F1, F2, F3, F4	Moving	Moving

The instructions that the AGV may receive at F4 could be to park somewhere until the task allocator receives information that space has become free in the input-queue. The AGV could also remain at the drop-off point until space becomes free in the queue.

Note that if the screening input-queues were blocked upon Job2 arriving at the output-queue, Job2 would not have been included in the job pool at the decision in B11.

4.4.2 Baggage Handling Reassignment Scenario

After a bag leaves screening levels 1 and 2, the results of the screening procedure may not be processed or be inconclusive. In order to increase the capacity of the screening process, bags are already removed from the process and routed to screening level 3. In case the results come in prior to the bag reaching screening level 3, the bag may be rerouted to make-up or EBS. In case reassignment should be considered, a trigger should be built into Job flow logic that determines whether a job should be reassigned to a new input-queue. This results in process flow steps H1, H2 and H3. In case the AGV has already reached a drop-off point, the reassignment may not take place and this result in steps H1, H2, H4.

4.4.3 Baggage Handling Conjoint Processes Scenario

This scenario requires a different setting than the one shown in Figure 4.4.1. The new setting concerns a screening machine followed by a conveyor leading to a second screening machine as shown in Figure 4.4.3 The framework components are shown in Figure 4.4.4.

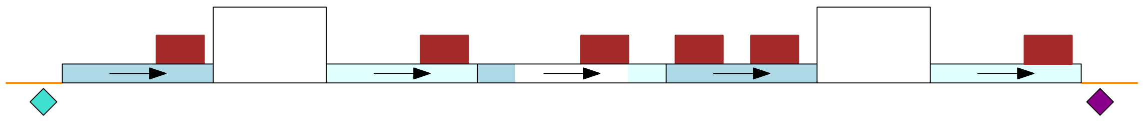


Figure 4.4.3: Baggage handling conjoint processes setting

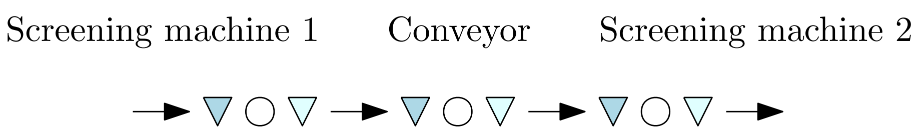


Figure 4.4.4: Baggage handling conjoint porocesses in framework components

The two screening machines and the conveyor linking them together are modeled as separate processes instead of one large process such that blocking and waiting behavior can be incorporated in more detail. The framework should automatically move the jobs through the process steps without interference of the AgvSorter system.

The scenario starts at t = 0 with Job1 being finished processing in screening machine 1. Meanwhile, Job2 is waiting in the input-queue preceding the first screening machine. Table 4.4.6 shows the steps at t = 0 for this scenario.

Table 4.4.6: Baggage handling conjoint process scenario steps for t = 0

$t = 0$	Job1 finishes processing

	Flowchart steps	**Location before**	**Location after**
Job1, Job2	Job1: A7, A6, A8, A12, B1, B3, B5, B6, B7, B8, B9, B10, B11, B13, B14, C1, C2, C3, C4, C5, C6, C7, C8, C9, C11, C1, D1, D3 → Job2: D5, D6, D7, D8, D10, C1, D1, D3, D4	Job1: Screening machine 1 process, Job2: Screening machine 1 input-queue	Job1: Conveyor process, Job2: Screening machine 1 process
Job2	→ D5, D6, D7, D8, D10, C1, D1, D3, D4	Screening machine 1 input-queue	Screening machine 1 process
Job1	A3, A4, A5	Conveyor process	Conveyor process

At D3 the perspective shifts to Job2 as this job is waiting to enter screening machine 1. The following steps move this job into the process and check whether the input-queue Job2 has been removed from can pull another job in. As this queue is preceded by the AgvSorter, this is not the case.

At B14 the job pool is sent to the input-queue of the conveyor process. As this queue is currently empty, it can pull the job directly from the output-queue. As Job1 is removed from the output-queue, an event is immediately scheduled that starts the flowchart at A3 to see if the output-queue can pull a waiting job from the process.

Once Job1 finishes processing on the conveyor, it is pulled from the output-queue of the conveyor to by the input-queue of screening machine 2 in section C of the flowchart. Also in section C, the job is directly pushed to the empty screening machine.

4.5 Warehousing Scenario Analysis

The warehousing scenario setting has been derived together with a domain expert from Vanderlande to form a representative setting based on existing warehousing processes and Vanderlande research into applying AGVs between the warehousing processes. The resulting setting is shown in Figure 4.5.1. This setting is abstracted into framework components as shown in Figure 4.5.2.

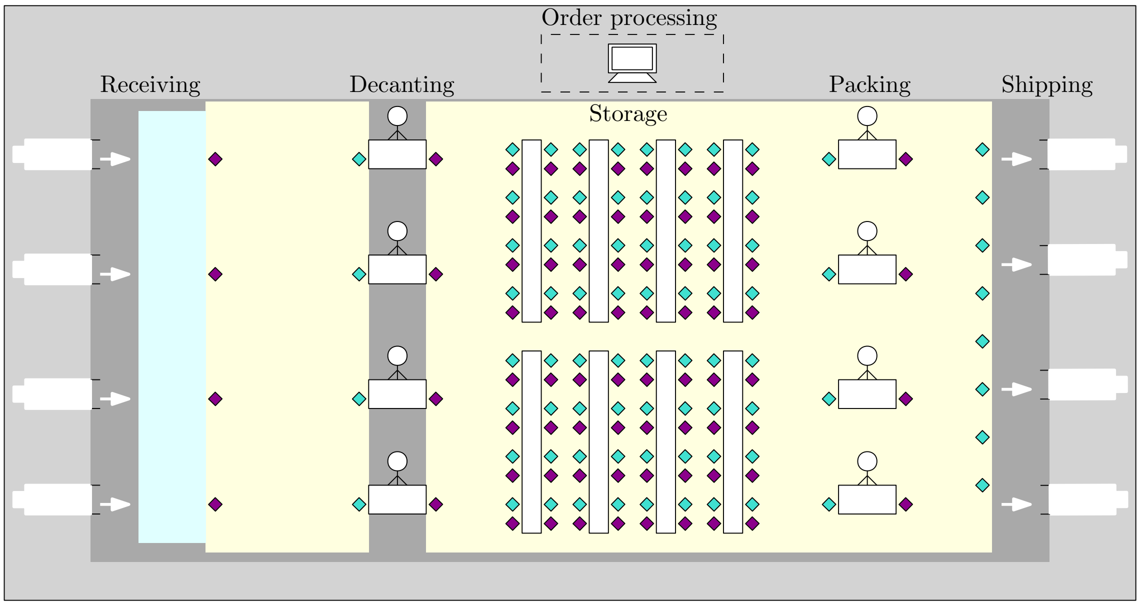


Figure 4.5.1: Warehousing scenario setting

The dark grey area in this figure represents the physical warehouse and processes. The light grey area form the primary system boundaries. It can be seen that both physical components and software components exist.

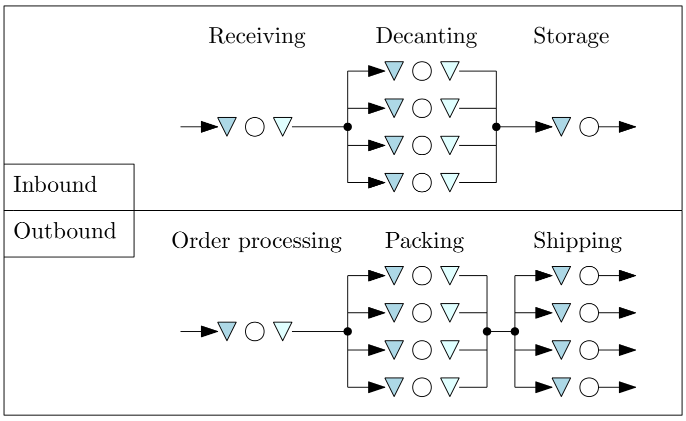


Figure 4.5.2: Warehousing processes as framework components

In Figure 4.5.2 both the inbound processes and outbound processes can be distinguished. The inbound process starts with the receiving process where an operator unloads pallets with goods from a truck. These pallets are placed in a receiving buffer from which the pallets can be collected by AGVs and delivered to one of the decanting workstations. At the decanting workstations the contents of the pallets is placed into smaller carriers, which can then be stored into the storage racks.

The outbound process starts with jobs arriving from customer orders, which are registered in the warehouse management system (order processing). The loads corresponding to these jobs can be picked up from the pick-up points located at the storage racks. After order picking, the collected items are brought to packing workstations. Lastly, the packages are sorted to shipping.

In this setting, there are two AgvSorter systems in place as the properties of the AGVs moving pallets are very different from the AGVs moving the smaller carriers from the decanting area. The AgvSorter between receiving and decanting is referred to as AgvSorter1 and the AgvSorter between decanting and shipping is AgvSorter2.

The attributes for the warehousing processes are shown in Table 4.5.1.

Table 4.5.1: Attributes for the warehousing framework components

Framework component	**selectionParameter**	**maxJobs**	**processTime [s]**
Receiving process	-	30	[180, 720]
Receiving input-queue	1	≥ 1	-
Receiving output-queue	4	20	-
Decanting process	-	1	[60, 600]
Decanting input-queue	1	1	-
Decanting output-queue	2	2	-
Storage process	-	Ample	10
Storage input-queue	1	≥ 1	-
Order processing process	-	Ample	3
Order processing input-queue	1	≥ 1	-
Order processing output-queue	queueLength	Ample	-
Packing process	-	Ample	60
Packing input-queue	1	≥ 1	-
Packing output-queue	2	2	-
Shipping process	-	Ample	300
Shipping input-queue	1	≥ 1	-

The decanting workstation is assumed to only be able to process one pallet at a time and it does not have a preceding input-queue as the receiving buffer stores pallets until the decanting workstations can receive more. However, the framework has the restriction that the maxJobs for an input-queue must always be one or more. If maxJobs would be zero, then no jobs would enter the input-queue. This means that the minimum capacity of a workstation without input-queue according to the framework becomes 2 jobs. Furthermore, it is assumed that the decanting areas have a small output-buffer of two carriers.

As the order processing software is not limited by physical constraints, it is assumed that the capacities of the process and output-queue are ample.

4.5.1 Inbound Process Scenario

The first scenario to be considered incorporates two AgvSorter systems as well as job splitting at the decanting workstation. For this scenario, one pallet (Job1) is followed that is unloaded from a truck at t = 0. As there are two AgvSorters involved with separate control systems, a distinction must be made in which AgvSorter receives what information. Table 4.5.2 shows the first steps at t = 0.

Table 4.5.2: Warehousing inbound process scenario steps for t = 0

$t = 0$	Job1 arrives		
	Flowchart steps	**Location before**	**Location after**
Job1	E1, E2, E3, E4, E5, C3, C4, C5, C6, C7, C8, C9, C11, C1, D1, D3, D4	Arriving	Receiving process (pt = 180)

The updated input-queue information in C4 and C7 is sent to the task allocator for AgvSorter1. However, this information is not used by the task allocator as there are no MHTs involved in moving the jobs to the receiving input-queue. The next event happens when the pallet has been unloaded and is moved into the receiving buffer (receiving output-queue). These steps are shown in Table 4.5.3.

Table 4.5.3: Warehousing inbound process scenario steps for t = 180

$t = 180$	Job1 is finished processing		
	Flowchart steps	**Location before**	**Location after**
Job1	A7, A6, A8, A12, B1, B3, B5, B6, B7, B8, B9, B10, B11, B13, B14, C1, D1, D3, D4	Receiving process	Receiving output-queue

The information in B14 is sent to the task allocator for AgvSorter1 as Job1 results in an MHT. Furthermore, in B8 the input-queue is already determined. This is a case where this information might not be used. In this case it is better to use the information determined later in G6 as this is based on the most up-to-date information on the status of the decanting workstations.

The rest of the scenario is considered when the AGV drops off the pallet at the decanting workstation as shown in Table 4.5.4.

Table 4.5.4: Warehousing inbound process scenario steps for t = 210

$t = 210$	AGV approaches a drop-off point		
	Flowchart steps	**Location before**	**Location after**
Job1	F1, F2, F3, F5, C3, C4, C5, C6, C7, C8, C10, C9, C11, C1, D1, D3, D4	Moving	Decanting process

Upon arriving at the decanting workstation, the job is split into multiple jobs. The one pallet arriving at the decanting workstation, results in multiple MHTs for AgvSorter2. It is assumed that the jobs resulting from the split together take up the capacity of one job such that the maxJobs for the process is not exceeded. Each job has a different processing time ranging from 60 seconds to 600 seconds based on when it is unloaded from the pallet. Only when all jobs are removed from the process, space becomes free in the process again. In the decanting output-queue, each job takes up one space that counts toward the maxJobs for the queue.

As AgvSorter2 follows after the decanting process, the processing time information in C11 is sent to the task allocator for AgvSorter2. The updated input-queue information in C7 is sent to the task allocator for AgvSorter1.

4.5.2 Batch Picking Scenario

This scenario shows steps in the outbound-process in a batch picking situation. Batch picking means that the AGV collects loads corresponding to multiple MHTs in one carrier. In other words, it is carrying out multiple MHTs in a many-to-one situation.

In Table 4.5.1 it is shown that the packing stations have ample capacity. This is to make sure that an AGV is able to unload all loads upon arrival at the packing station as this is required in a many-to-one setting. In a warehouse, complex algorithms are in place to ensure that a picking order is released only when the packing stations can process the arriving loads. Within the framework, the user can implement these algorithms in Job flow logic.

The scenario starts with an online customer order arrival. The steps are shown in Table 4.5.5.

Table 4.5.5: Warehousing batch picking scenario steps for t = 0

$t = 0$	Order arrives at the Order processing input-queue		
	Flowchart steps	**Location before**	**Location after**
Job1, Job2	E1, E2, E3, E4, E5, Job1: C3, C4, C5, C6, C7, C8, C9, C11, C1 $\rightarrow$ Job2: C2, C3, C4, C5, C6, C7, C8, C9, C11, C1, D1, D3, D4	Job1, Job2: Arriving	Job1, Job2: Order processing process (pt = 3)

Upon arrival of the order, the order is split into Job1 and Job2. At E3, Job2 is placed in the job pool for the order processing input-queue and Job1 is moved to the input-queue and then the process starting at C3. At C1 it is determined that Job2 is located in the job pool and is subsequently pulled into the input-queue and then the process.

At t = 3, both jobs are moved to the output-queue similarly to previous scenarios. In this case it does not matter which job is moved to the output-queue first as they are both included in the job pool according to the selectionParameter. The user can simply use a (default) FIFO rule and move Job1 first.

Based on the jobs located in the output-queue, an AGV needs to execute the MHT corresponding to Job1 and another MHT corresponding to Job3, which was already located in the output-queue prior to the arrival of Job1 and Job2. This means that the AGV must visit the pick-up points (repeat steps G1, G2, G3) for both MHTs prior to driving to the drop-off point where all loads are unloaded. Table 4.5.6 shows the scenario steps for the AGV visiting the final pick-up point. At this point it is determined which packing station the AGV should drop the loads off at. This depends on which packing station is responsible for which package and is dependent on warehousing algorithms that the user can include in Jobflow logic. Table 4.5.7 shows the steps when the AGV arrives at the drop-off point for a packing workstation.

Table 4.5.6: Warehousing batch picking scenario steps for t = 300

$t = 300$	AGV approaches a pick-up point		
	Flowchart steps	**Location before**	**Location after**
AGV	G1, G2, G4, G6, G8, G9, G10	-	-

Upon arriving at the drop-off point, F5 determines that this is the final drop-off point that needs to be visited for the MHT corresponding to Job1. Next, it is again registered that the AGV is also at the drop-off point for the MHT corresponding to Job3. Therefore, this MHT is moved to the process as well. This corresponds to delivering the carrier with batch picked items to the packing workstation.

Table 4.5.7: Warehousing batch picking scenario steps for t = 330

$t = 330$	AGV approaches a drop-off point		
	Flowchart steps	**Location before**	**Location after**
Job1	F1, F2, F3, F5, C3, C4, C5, C6, C7, C8, C9, C11, C1, D1, D3, D4	Moving	Packing process (pt = 60)
Job3	F1, F2, F3, F5, C3, C4, C5, C6, C7, C8, C9, C11, C1, D1, D3, D4	Moving	Packing process (pt = 60)

At t = 390, Job1 has finished processing, but it needs to wait for Job2 as the loads corresponding to Job1 need to be packed together with the loads corresponding to Job2. These steps are shown in Table 4.5.8. Job3 has also finished processing, but is not considered further for this scenario.

Table 4.5.8: Warehousing batch picking scenario steps for t = 390

$t = 390$	Job1 is finished processing		
	Flowchart steps	**Location before**	**Location after**
Job1	A7, A6, A8, A9, A10	Packing process	Packing process

The steps continue when Job2 has arrived and finished processing as shown in Table 4.5.9.

Table 4.5.9: Warehousing batch picking scenario steps for t = 450

$t = 450$	Job2 is finished processing		
Job	**Flowchart steps**	**Location before**	**Location after**
Job2	A7, A6, A8, A9, A11 $\rightarrow$ Job1,2: A12, B1, B3, B5, B6, B7, B8, B9, B10, B11, B13, B14, C1, D1, D3, D4	Packing process	Job1,2: Packing output-queue

At step A11, Job1 and Job2 are merged together as their loads are to be transported as one package. The constraint holds that the merge may not take place if the physical constraints of the AGV executing the resulting MHT are exceeded.

4.5.3 Automated Packing and Labeling Scenario

In this scenario, the packing stations are not needed as the AGVs collect the loads directly in the packages and drive by an automated labeling machine shown in Figure 4.5.3.

The machine has four input-queue spaces on the ingoing conveyor and four spaces in the output-queue. The selectionParameter for each queue is 1.

For this scenario, an order has arrived that is split into several jobs. Each job is a separate package and the order is fulfilled by shipping all packages on a pallet to the customer. This means that no merge is required and each AGV is carrying out one MHT at a time. The flowchart steps are therefore the same as in the previous scenario, only for one MHT per AGV and no merge.

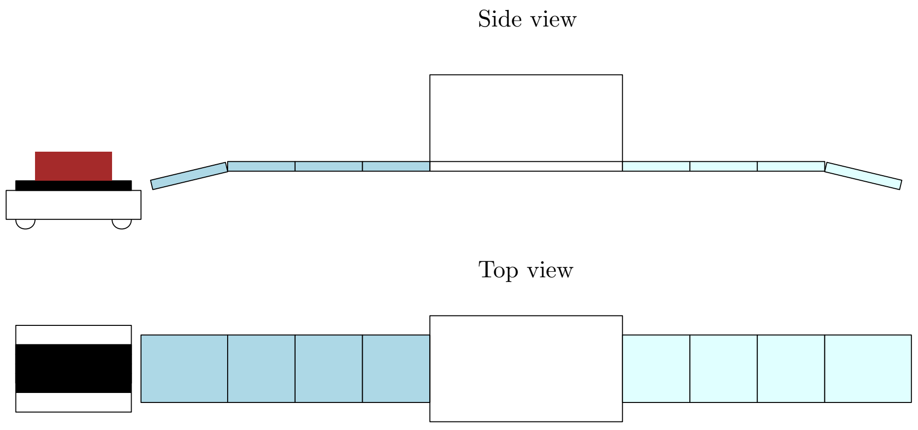


Figure 4.5.3: Warehousing automated labeling machine

4.6 Conclusion

In this chapter two validation strategies are carried out to validate the conceptual design of the framework. The requirements analysis showed that the requirements derived in Chapter 2 are met either through incorporation in the order-job-load relationship and attributes, the framework components, the flowchart or as input for the framework.

Next, the scenario analysis showed for each of the three types of primary systems that a wide range of functionality can be modeled in detail using the framework components and the flowchart. In the next chapter a conclusion is presented regarding the design of the framework and recommendations for future research are given.

Chapter 5

Conclusion and Recommendations

5.1 Conclusion

In this thesis, a framework is developed that can be used to model the primary systems with a wide range of functionality incorporated. This framework is developed with focus on the baggage handling systems, parcel depots and warehouses as primary systems. In Section 1.4, the main research question is presented and divided into three objectives. The objectives are discussed to draw a conclusion on the results and the main research question.

Identify key characteristics of the baggage handling, parcel and warehousing application areas.

In Chapter 2, this objective is treated by analyzing the baggage handling, parcel and warehousing applications. This showed that the three types of primary systems differ in terms of arrival patterns, multi-load AGV situations, processes and information flow. Next to that, task allocation strategies are studied to identify different inputs required to use the strategies encountered in literature. From the analysis in this chapter, requirements are derived that need to be incorporated indirectly or directly in the framework or as input for the framework.

Construct a general framework that takes into account the key characteristics identified in the first objective.

The conceptual design of the framework is presented in Chapter 3. Using the DDD strategy, the framework components are designed, with a distinction between the enterprise domain, process domain and material handling domain. The framework components in combination with the flowcharts and the order-job-load relationships form the framework that can be used to model the primary systems.

Validate the conceptual framework design.

In order to validate the conceptual framework design, Chapter 4 shows two validation strategies. In the requirements analysis it is shown that the framework incorporates most requirements directly either in the framework components, the order, job and load attributes or as input. Some requirements, such as Req.ML.5, are not included directly in the framework, but are part of user defined areas. The framework provides the necessary tools to meet these requirements and they should be seen as constraints for the user when implementing the framework to design a task allocation strategy. From the requirements analysis it is shown that the framework contains sufficient inputs for the development of a task allocation strategy.

The scenario analysis shows that the framework is able to represent a wide range of functionality within baggage handling systems, parcel depots and warehouses. The scenario analysis supported the requirements analysis in showing the functionality and information flow in the flowcharts. This indicates that the framework can be used for analysis of the impact of primary system configurations on the AgvSorter.

In what way can the primary systems be captured in one model, such that the model can be used in different simulations and analyses?

It can be concluded that by meeting the three research objectives the main research question can be answered. A conceptual framework is designed that can be applied for analysis of baggage handling systems, parcel depots and warehouses. This framework provides the user with a fixed set of tools to carry out both complex and simple analysis for all three types of primary systems. With this framework, task allocation strategies can be derived for specific configurations of primary systems that respect the constraints presented in each system.

5.2 Recommendations

In this section recommendations are given for future research and development of the framework. This thesis covers the conceptual design phase of the framework. The next step in the development is to implement the framework and test it in the AgvSorter model.

5.2.1 Full Functionality Comparison with Real Systems

In the scenario analysis in the validation a limited set of functionality is shown for simplified situations. This validation shows that the necessary functionality is incorporated in the framework. However, to conclude that the framework can be used for detailed analysis, the framework should be implemented and simulated and compared to fully functional primary systems.

5.2.2 Extend the Use of the Framework to More Types of Primary Systems

The framework has been developed for requirements derived for baggage handling systems, parcel depots and warehouses as primary systems. It should be investigated whether there are other types of primary systems that can also be analyzed with the help of the framework. Any system that can be modeled using a combination of input-queues, a process and output- queues could be applicable to the framework.

Flexible manufacturing systems in literature are often modeled using input-queues, output-queues and workstations and could therefore also be modeled using the framework [1] [31]. Port container terminals are also common in literature on AGV systems. This setting could also be used as primary system for the framework [37].

As the guide-path layout is black-box for the framework. The framework could be applied in combination with AGV systems that are non-grid based. The only grid related information that is required is which pick-up points and drop-off points are mapped to certain queues. This information could also be made available in non-grid-based systems.

5.2.3 Develop Default Component Models for the Framework

For each AgvSorter layout and primary system in place, the framework attributes and components need to be determined and assembled. In order to increase user friendliness, several default framework representations for common primary system processes should be defined. As this thesis covers the conceptual design phase of the framework, these default models have not yet been constructed. Defining default functions for common processes should be considered when implementing the framework. The default models should not contain complex analysis behavior. When a simple, deterministic model for certain processes is provided, the user can choose to extend these models to include the desired complexity.

Bibliography

[1] T. Le-Anh and M.B.M. De Koster. On-line dispatching rules for vehicle-based internal transport systems. *International Journal of Production Research*, 43(8):1711–1728, 2007.

[2] N. Jawahar, P. Aravindan, S.G. Ponnambalam, and R.K. Suresh. Agv schedule integrated with production in flexible manufacturing systems. *The International Journal of Advanced Manufacturing Technology*, 14(6):428–440, 1998.

[3] L. Kalinovcic, T. Petrovic, S. Bogdan, and V. Bobanac. Modified Banker's algorithm for scheduling in multi-AGV systems. In *2011 IEEE International Conference on Automation Science and Engineering*, pages 351–356. IEEE, 2011.

[4] D. Pjevcevic, M. Nikolic, N. Vidic, and K. Vukadinovic. Data envelopment analysis of AGV fleet sizing at a port container terminal. *International Journal of Production Research*, 55(14):4021–4034, 2017.

[5] W. Małopolski. A sustainable and conflict-free operation of AGVs in a square topology. *Computers & Industrial Engineering*, 126:472–481, 2018.

[6] Vanderlande. AgvSorter Model. `https://vikipedia.vanderlande.com/display/SIM/AgvSorter+Model`. Accessed: 31-07-2019.

[7] K.J.C Fransen. A path planning approach for AGVs in the dense grid-based agvsorter. Master's thesis, Eindhoven University of Technology, 2019.

[8] M. van Weert. Deadlock avoidance and detection for the grid-based AGV-sorter system. Master's thesis, Eindhoven University of Technology, 2019.

[9] S. Rajotia, K. Shanker, and J.L. Batra. Determination of optimal AGV fleet size for an FMS. *International Journal of Production Research*, 36(5):1177–1198, 1998.

[10] P.J. Egbelu and J.M.A. Tanchoco. Characterization of automatic guided vehicle dispatching rules. *The International Journal of Production Research*, 22(3):359–374, 1984.

[11] M. ten Hompel and T. Schmidt. *Warehouse management: automation and organisation of warehouse and order picking systems.* Springer Science & Business Media, 2007.

[12] A. Abdelghany, K. Abdelghany, and R. Narasimhan. Scheduling baggage-handling facilities in congested airports. *Journal of Air Transport Management*, 12(2):76–81, 2006.

[13] I. Alsyouf, U. Kumar, L. Al-Ashi, and M. Al-Hammadi. Improving baggage flow in the baggage handling system at a UAE-based airline using lean Six Sigma tools. *Quality Engineering*, 30(3):432–452, 2018.

[14] W.J. Hopp and M.L. Spearman. *Factory physics.* Waveland Press, 2011.

[15] Y.C. Ho and S.H. Chien. A simulation study on the performance of task-determination rules and delivery-dispatching rules for multiple-load AGVs. *International Journal of Production Research*, 44(20):4193–4222, 2006.

[16] J.A. Ventura, S. Pazhani, and A. Mendoza. Finding optimal dwell points for automated guided vehicles in general guide-path layouts. *International Journal of Production Economics*, 170:850–861, 2015.

[17] P.J. Egbelu. Positioning of automated guided vehicles in a loop layout to improve response time. *European Journal of Operational Research*, 71(1):32–44, 1993.

[18] C.I. Liu and P.A. Ioannou. A petri net based approach for AGV dispatch scheduling and fleet size determination. *IFAC Proceedings Volumes*, 35(1):19–24, 2002.

[19] M.M. Srinivasan, Y.A. Bozer, and M. Cho. Trip-based material handling systems: throughput capacity analysis. *IIE transactions*, 26(1):70–89, 1994.

[20] Y.A. Bozer and C. Eamrungroj. Throughput analysis of multi-device trip-based material handling systems operating under the modified-FCFS dispatching rule. *International Journal of Production Research*, 56(4):1486–1503, 2018.

[21] B.H. Jeong and S.U. Randhawa. A multi-attribute dispatching rule for automated guided vehicle systems. *International Journal of Production Research*, 39(13):2817–2832, 2001.

[22] K.K. Tan, K.C. Tan, and K.Z. Tang. Evolutionary tuning of a fuzzy dispatching system for automated guided vehicles. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 30(4):632–636, 2000.

[23] D. Naso and B. Turchiano. Multicriteria meta-heuristics for AGV dispatching control based on computational intelligence. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 35(2):208–226, 2005.

[24] V.F. Caridá, O. Morandin, and C.C.M. Tuma. Approaches of fuzzy systems applied to an AGV dispatching system in a FMS. *The International Journal of Advanced Manufacturing Technology*, 79(1-4):615–625, 2015.

[25] R.E. Burkard, M. Dell'Amico, and S. Martello. *Assignment problems*. Springer, 2009.

[26] T.A. Thakre, O.K. Chaudhari, and N.R. Dhawade. Placement of staff in LIC using fuzzy assignment problem. *International Journal of Mathematics Trends and Technology*, 53(4):259–266, 2018.

[27] S. Singh, G.C. Dubey, and R. Shrivastava. A comparative analysis of assignment problem. *IOSR Journal of Engineering*, 2(8):2250–3021, 2012.

[28] W. Wu, M. Iori, S. Martello, and M. Yagiura. Exact and heuristic algorithms for the interval min-max regret generalized assignment problem. *Computers & Industrial Engineering*, 125:98–110, 2018.

[29] F. Chauvet, J.M. Proth, and A. Soumare. The simple and multiple job assignment problems. *International Journal of Production Research*, 38(14):3165–3179, 2000.

[30] P. Lacomme, A. Moukrim, and N. Tchernev. Simultaneous job input sequencing and vehicle dispatching in a single-vehicle automated guided vehicle system: a heuristic branch-and-bound approach coupled with a discrete events simulation model. *International Journal of Production Research*, 43(9):1911–1942, 2005.

[31] Y.A. Bozer and C.K. Yen. Intelligent dispatching rules for trip-based material handling systems. *Journal of Manufacturing Systems*, 15(4):226–239, 1996.

[32] J.K. Lim, K.H. Kim, K. Yoshimoto, J.H. Lee, and T. Takahashi. A dispatching method for automated guided vehicles by using a bidding concept. *OR Spectrum*, 25(1):25–44, 2003.

[33] M.H.F. bin M. Fauadi, S.H. Yahaya, and T. Murata. Intelligent combinatorial auctions of decentralized task assignment for AGV with multiple loading capacity. *IEEJ Transactions on Electrical and Electronic Engineering*, 8(4):371–379, 2013.

[34] R. Cui, J. Guo, and B. Gao. Game theory-based negotiation for multiple robots task allocation. *Robotica*, 31(6):923–934, 2013.

[35] P.P.S. Chen. The entity-relationship model — toward a unified view of data. *ACM Transactions on Database Systems (TODS)*, 1(1):9–36, 1976.

[36] S. Millett. *Patterns, principles and practices of domain-driven design.* John Wiley & Sons, 2015.

[37] D. Pjevcevic, M. Nikolic, N. Vidic, and K. Vukadinovic. Data envelopment analysis of AGV fleet sizing at a port container terminal. *International Journal of Production Research*, 55(14):4021–4034, 2017.

Appendix A

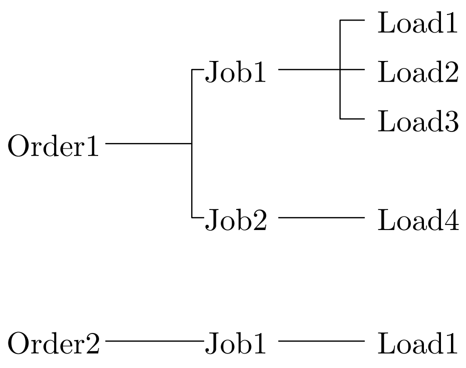
Load File

In this appendix, the two parts of the load file are illustrated. In Matlab a load file generator function is implemented that allows the generation of a load file for warehousing, baggage handling and parcel applications.

The load file consists of an arrivalFile and an orderDatabase. The arrivalFile contains information on order arrivals and the IDs of the arriving orders. In case of baggage handling, it also contains flight information such as departure times. The table below shows an example of the structure of an arrivalFile (without flight information).

t [s]	**Arriving order ID**
0	Order1
1	Order2
1.2	Order3
5	Order4
5	Order5

The jobs, loads and attributes related to an order are found in the orderDatabase. As shown in the figure below, the orders, jobs and loads are built up as structs. Their attributes are fields in the struct.



Appendix B

Framework Components Within the Flowcharts

This appendix shows an overview of where in the flowchart the following framework components are applied:

- Job flow logic
- Order splitting logic
- Pick-up point mapping
- Drop-off point mapping

These components include algorithms and decision rules that determine the progression of jobs through the various processes. The table below shows the overview of the flowchart steps and the related framework components.

Flowchart step	**Framework Component**
A6	Job flow logic
A8	Job flow logic
A12	Job flow logic
B6	Pick-up point mapping
B7	Pick-up point mapping
B8	Job flow logic
B9	Drop-off point mapping
B10	Drop-off point mapping
B11	Job flow logic, output-queue selectionParameter
C2	Job flow logic
C8	Job flow logic
C10	Job flow logic
D5	Job flow logic
D7	Job flow logic
D9	Job flow logic
E2	Order splitting logic
E3	Job flow logic
E4	Job flow logic
G6	Job flow logic
G8	Drop-off point mapping
G9	Drop-off point mapping
H1	Job flow logic
H3	Job flow logic, Drop-off point mapping

Appendix C

Scenario Elaboration

This appendix is intended to give some additional information on the use of the flowchart.

The steps at t = 0 and t = 30 in the single-item flow validation scenario from Subsection 4.3.1 are elaborated. The table below shows the scenario steps for t = 0. The same framework components and their attributes are used as in Subsection 4.3.1.

$t = 0$	Job1 and Job2 arrive simultaneously		
	Flowchart steps	**Location before**	**Location after**
Job1	E1, E2, E3, E4, E5, C3, C4, C5, C6, C7, C8, C9, C11, C1, D1, D3, D4	Arriving at conveyor input-queue 1	Conveyor process (pt = 30 s)
Job2	E1, E2, E3, E4, E5, C3, C4, C5, C6, C7, C8, C9, C11, C1, D1, D3, D4	Arriving at conveyor input-queue 4	Conveyor process (pt = 60 s)

Job1 and Job2 belong to two separate orders. From each order, there results one job at E2. It is not necessary or possible to split the order into multiple jobs as the order is represented by one parcel. As there is only one job resulting from the order, no jobs are placed in the job pool for the input-queue at E4. When step C1 is reached, the input-queue has no job to pull in and the flowchart moves to step D1. There is space in the process, but there are no jobs waiting in the input-queue. Therefore the flowchart steps and at D4. After leaving section C it should always be checked whether a process can pull a job in. If in previous process steps a job is removed from a process, this process may be able to pull a job in from its preceding input-queue.

At t = 30, Job1 has finished processing. The steps are shown in the table below.

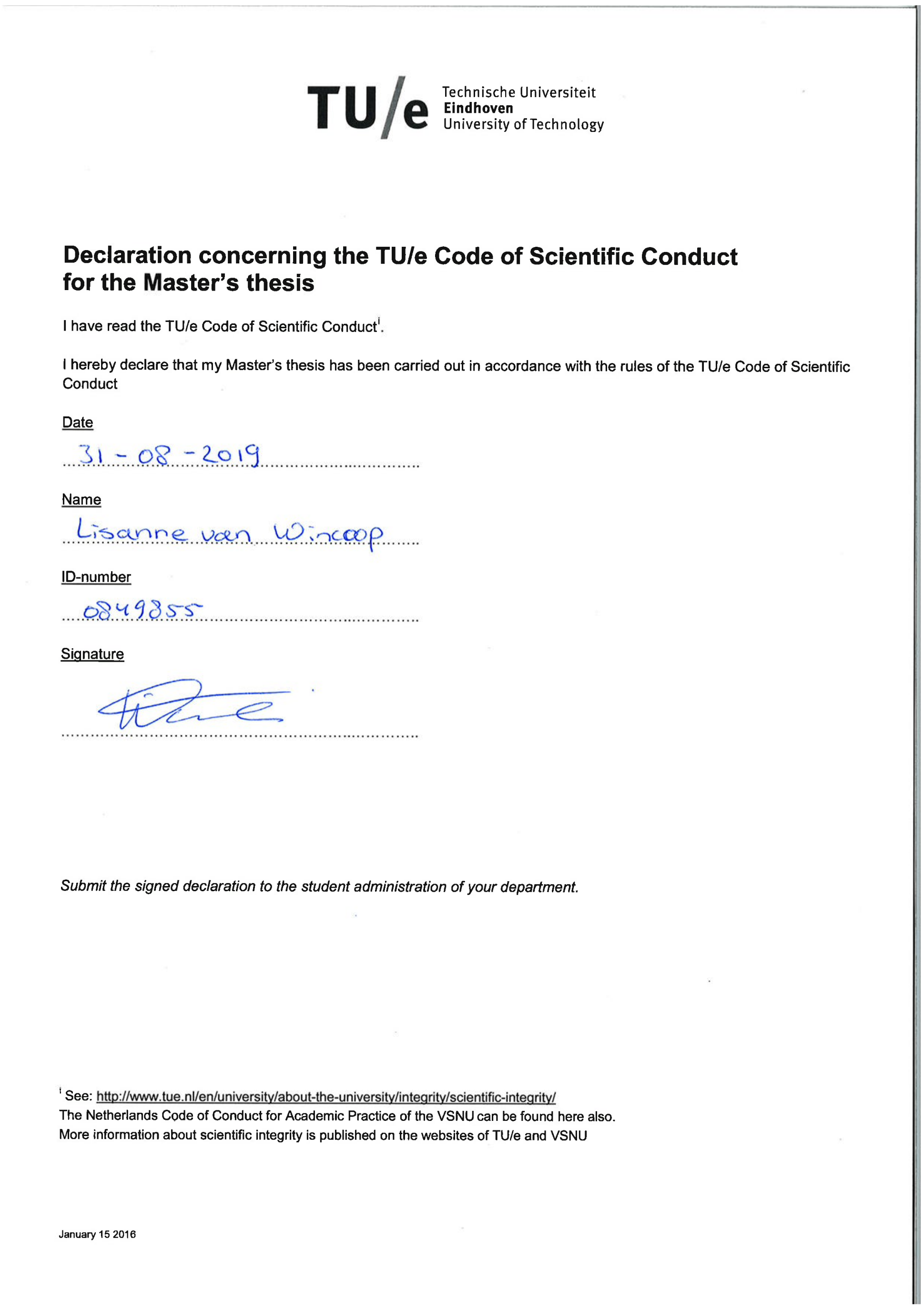
$t = 30$	Job1 is finished processing		
	Flowchart steps	**Location before**	**Location after**
Job1	A7, A6, A8, A12, B1, B3, B5, B6, B7, B8, B9, B10, B11, B12, B14, C1, D1, D3, D4	Conveyor process	Conveyor output-queue 2

As only Job1 is finished processing, the job selected at A6 is Job1. Whenever there are multiple jobs done with their processing simultaneously, a job needs to be selected at A6.

In this primary system, the information in B6-B11 is usable as the full information on the parcel is known from the scan that happened in the conveyor process. In the bulk flow scenario, this information is not usable.

Appendix D

Code of Scientific Conduct



TU/e Technische Universiteit **Eindhoven** University of Technology

Declaration concerning the TU/e Code of Scientific Conduct for the Master's thesis

I have read the TU/e Code of Scientific Conduct[i].

I hereby declare that my Master's thesis has been carried out in accordance with the rules of the TU/e Code of Scientific Conduct

Date

31-08-2019

Name

Lisanne van Wincoop

ID-number

0849855

Signature

[illegible]

Submit the signed declaration to the student administration of your department.

[i] See: http://www.tue.nl/en/university/about-the-university/integrity/scientific-integrity/
The Netherlands Code of Conduct for Academic Practice of the VSNU can be found here also.
More information about scientific integrity is published on the websites of TU/e and VSNU

January 15 2016